Methods for Converging Solutions of Differential Equations: Applying Imaginary Time Propagation to Density Functional Theory and Unsupervised Neural Networks to Dynamical Systems

Methods for Converging Solutions of Differential Equations: Applying Imaginary Time Propagation to Density Functional Theory and Unsupervised Neural Networks to Dynamical Systems

A DISSERTATION PRESENTED

BY

CEDRIC WEN FLAMANT

TO

THE DEPARTMENT OF PHYSICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

PHYSICS

HARVARD UNIVERSITY

CAMBRIDGE, MASSACHUSETTS

APRIL 2020

Dissertation Advisor: Professor Efthimios Kaxiras Cedric Wen Flamant

Methods for Converging Solutions of Differential Equations: Applying Imaginary Time Propagation to Density Functional Theory and Unsupervised Neural Networks to Dynamical Systems

Abstract

Reliable and robust convergence to the electronic ground state within density functional theory (DFT) Kohn-Sham (KS) calculations remains a thorny issue in many systems of interest. Here, we use an approach based on transforming the time-dependent DFT equations to imaginary time, followed by imaginary-time evolution, as a reliable alternative to the self-consistent field (SCF) procedure for determining the KS ground state. We discuss the theoretical and technical aspects of this approach and show that the KS ground state should be expected to be the long-imaginary-time output of the evolution, independent of the exchange-correlation functional or the level of theory used to simulate the system. By maintaining self-consistency between the single-particle wavefunctions (orbitals) and the electronic density throughout the determination of the stationary state, our method avoids the typical difficulties encountered in SCF. To demonstrate dependability of our approach, we apply it to selected systems which struggle to converge with SCF schemes. In addition, through the van Leeuwen theorem, we affirm the physical meaningfulness of imaginary time TDDFT, justifying its use in certain topics of statistical mechanics such as in computing imaginary time path integrals.

The time evolution of dynamical systems is frequently described by ordinary differential equations (ODEs), which must be solved for given initial conditions. Most standard approaches numerically integrate the ODEs, producing a solution whose values are computed at discrete times. For every set of initial conditions and system parameters, the calculation has to be repeated from scratch, adding significant computational overhead to methods which require varied solutions to the ODE. We extend the Lagaris method of creating an approximating neural network solution to a set of differential equations, proposing instead that a neural network be used as a solution bundle, a collection of solutions to an ODE for various initial states and system parameters. The neural network solution bundle is trained with an unsupervised loss that does not require any prior knowledge of the sought solutions, and the resulting object is differentiable in initial conditions and system parameters. The solution bundle exhibits fast, parallelizable evaluation of the system state, facilitating the use of Bayesian inference for parameter or trajectory estimation in real dynamical systems.

Contents

I Imaginary Time Propagation in Density Functional Theory **1**

1 DENSITY FUNCTIONAL THEORY **3**
- 1.1 Introduction . . . 3
- 1.2 Ground-State Density Functional Theory . . . 4
  - 1.2.1 Hohenberg-Kohn Theorem . . . 6
  - 1.2.2 Kohn-Sham Formalism . . . 8
  - 1.2.3 Discussion of Kohn-Sham Formalism . . . 12
  - 1.2.4 DFT in Practice . . . 13
- 1.3 Time-Dependent Density Functional Theory . . . 14
  - 1.3.1 Runge-Gross Theorem . . . 16
  - 1.3.2 The van Leeuwen Theorem . . . 20
  - 1.3.3 Time-Dependent Kohn-Sham Formalism . . . 26
  - 1.3.4 Time-Dependent Exchange-Correlation Potential Approximations . . . 27
- 1.4 Linear Response in TDDFT . . . 28
  - 1.4.1 Linear Response Review . . . 29
  - 1.4.2 The Density-Density Response Function . . . 30
  - 1.4.3 Kohn-Sham Density-Density Response . . . 32
- 1.5 Conclusion . . . 34

2 IMAGINARY-TIME TIME-DEPENDENT DENSITY FUNCTIONAL THEORY **36**
- 2.1 Introduction . . . 37
- 2.2 Methodology . . . 39
  - 2.2.1 Imaginary-Time Propagation . . . 39
  - 2.2.2 Implementation within the Kohn-Sham Formalism . . . 40
- 2.3 Theoretical Considerations . . . 42
  - 2.3.1 van Leeuwen Theorem in Imaginary Time . . . 42
  - 2.3.2 Maintaining Orthonormalization . . . 43
  - 2.3.3 Monotonically Decreasing Energy . . . 46
  - 2.3.4 Alternative Theoretical Foundation for Stationary States in DFT . . . 47
  - 2.3.5 Practical Advantages of it-TDDFT . . . 48
- 2.4 Example Calculations . . . 49
- 2.5 Conclusion . . . 55
- 2.6 Supporting Information . . . 56
- 2.7 Acknowledgments . . . 56

II Solving Differential Equations with Unsupervised Neural Networks **57**

3 INTRODUCTION TO NEURAL NETWORKS **59**
- 3.1 Introduction . . . 59
- 3.2 Artificial Neural Networks . . . 61
  - 3.2.1 Multilayer Perceptron . . . 61
- 3.3 Universal Approximation Theorem . . . 65
  - 3.3.1 Shallow Networks . . . 66
  - 3.3.2 Deep Networks . . . 71

3.4 Learning . . . 72
3.4.1 Loss Function . . . 73
3.4.2 Stochastic Gradient Descent . . . 76
3.5 Automatic Differentiation and Backpropagation . . . 78

4 SOLVING ORDINARY DIFFERENTIAL EQUATIONS WITH NEURAL NETWORKS **82**
4.1 Introduction . . . 82
4.2 Solution Bundles . . . 84
4.2.1 Method Description . . . 84
4.3 Propagating a Distribution . . . 90
4.3.1 Planar Circular Restricted Three-Body Problem . . . 91
4.4 Bayesian Parameter Inference . . . 97
4.4.1 Simple Harmonic Oscillator . . . 98
4.4.2 Rebound Pendulum . . . 100
4.4.3 FitzHugh-Nagumo Model . . . 102
4.5 Conclusion . . . 112

A JANAK'S THEOREM **119**

B SUPPORTING INFORMATION FOR CHAPTER 2 **125**
B.1 Cu_{13} Example Calculation Details . . . 125
B.2 Ru_{55} Example Calculation Details . . . 126

To my grandparents, particularly Ming-Ping Feng, who always took special interest in my academics.

Acknowledgments

First I would like to thank Efthimios Kaxiras for his guidance on giving effective technical presentations and for maintaining such a great group. I am also very grateful for his advice regarding my career plans, and for being exceedingly supportive in helping me be prepared for them. Oscar Grånäs was a joy to be around in lab, entertaining us all with his stories and sharp sense of humor. I will fondly remember hanging out with him at the various APS March Meetings. Grigory Kolesov was a great mentor in the finer points of linux and high-performance computing, and we had countless exciting discussions about physics while waiting for jobs to run. He always had strong opinions about computational tools, whether it be about a brand of laptop or a programming language. I (nearly) always eventually learned that his opinion was right. Matt Montemore was very helpful in finding sufficiently poorly-behaved systems to try my method on, and his cheerful attitude was always a boon to Cruft 406. I would also like to thank Robert Hoyt for great friendship and the useful advice about everything from preparing for quals, searching for jobs, to the latest technology in speakers. Daniel Larson was also a great source of conversation on a wide variety of topics over lunch, and he was also the bringer of one of my favorite "holidays" of grad school, the group's annual liquid nitrogen ice cream party. Stephen Carr had an uncanny ability to identify the most important point of any technical discussion, and speaking to him about my projects was always an elucidating experience. Zoe Zhu and Steven Torrisi were also great to talk to over lunch, and I wish them the best of luck with their own grad school adventures.

Outside of Cruft there were many other sources of guidance and motivation. Pavlos Protopapas was incredibly useful in my explorations of data science and machine learning, and he put together a great data science class that was fun to help teach. He also brought unique opportunities and boosted spirits with fun gatherings and events. Just when I thought I had seen everything Harvard had to offer, he filled my final year with new experiences. David Sondak was also very helpful in my time at the IACS, and I always enjoyed his humorous commentary during group meeting. I also appreciate Michael Emanuel's insightful perspectives on my projects. I also want to thank Philip Kim and Eric Heller for serving on my thesis committee.

Thank you Jacob Shenker for the valuable technical conversations about my projects at the gym, always the last place I would expect to have a breakthrough. Most of my projects had one of their lynchpin ideas emerge during our workouts from us bouncing ideas back and forth and taking longer breaks between sets than we should. Laura Kulowski, thank you for all your support and for making grad school a doubles game. If I could even manage to return half of the support you have given me, I would be accomplishing an

incredible feat. Finally, I would like to thank my wonderful parents Wan-Fan Feng and Sylvain Flamant for shaping me into the person I am today, and Etienne Flamant for being the best brother I could ask for.

Part I

Imaginary Time Propagation in Density Functional Theory

Page intentionally left blank

1

Density Functional Theory

1.1 Introduction

Density functional theory, (DFT), is a formally exact approach to the electronic many-body problem. It is widely used both in industry and in academia for determining the properties of molecules and materials, ranging from determining molecular structures, estimating mechanical properties, computing activation barriers, to calculating optical spectra. The general idea is to use the electronic density as the fundamental mathematical object, instead of the far more intricate many-body wavefunction. Remarkably, despite appearing like a simplifying approximation, DFT is formally exact, and is a different, equivalent view of the many-body quantum theory. Note that this view does not allow us to exactly solve the many-body problem in practice as its intricacies and challenges remain. However, the mathematical structure of DFT is more amenable to incremental levels of approximation, a boon for practical application. In particular, we commonly use the Kohn-Sham system of noninteracting quasielectrons as a tractable way to calculate the electronic density of the true many-body interacting system, which can then be used to compute desired observables. Time-dependent density functional theory (TDDFT) extends the methods of DFT to dynamical systems such as those found in optical excitations, electronic transport, spectroscopy, electron-phonon coupling and more. Like its stationary counterpart, implementations of TDDFT often replace the many-body electronic wavefunction with a noninteracting Kohn-Sham system that reproduces the same density. Through the Runge-Gross and van Leeuwen theorems, it is proven that this replacement results in the same physics for a large class of systems, with more general proofs remaining a current topic of research.

DFT is powerful due to its favorable scaling with system size, a useful trait for the increasingly complex systems of great interest in material science, chemistry, and biology. Much of the economy of DFT arises from eliminating the need for the many-body wavefunction, a construction that contains far more information than one could need or want[45]. Usually we are only interested in the probability densities of one or two electronic coordinates, integrating over the rest, so much of the information in the wavefunction

is discarded anyways. By using the electronic density as the fundamental state describing the system, we reduce the number of relevant coordinates from ND to just D, where N is the number of electrons, and D is the dimension of the system.

In this chapter we will examine some of the underpinnings of both the static and time-dependent density functional theories and provide a few basic examples of calculations that can be performed with these methods. This sets the stage for the novel contribution to the theory presented in Chapter 2, imaginary-time time-dependent density functional theory, which enables the connections between quantum and statistical mechanics afforded by Wick rotations, as well as providing a practical method for solving the Kohn-Sham equations in static DFT.

1.2 Ground-State Density Functional Theory

Consider a system of N interacting electrons described by the non-relativistic Schrödinger equation

$$\hat{H}_0\Psi_j(\mathbf{r}_1,\ldots,\mathbf{r}_N) = E_j\Psi_j(\mathbf{r}_1,\ldots,\mathbf{r}_N), \qquad j = 0,1,2,\ldots \tag{1.1}$$

The jth eigenstate of the Hamiltonian is $\Psi_j(\mathbf{r}_1,\ldots,\mathbf{r}_N)$, and of particular interest is the ground state, $\Psi_0 \equiv \Psi_{\text{gs}}$.

The Hamiltonian is given by

$$\hat{H} = \hat{T} + \hat{V} + \hat{W}, \tag{1.2}$$

where the kinetic energy and scalar potential operators are

$$\hat{T} = \sum_{j=1}^{N} -\frac{\nabla_j^2}{2}, \qquad \hat{V} = \sum_{j=1}^{N} v(\mathbf{r}_j), \tag{1.3}$$

and the electron-electron interaction operator is

$$\hat{W} = \sum_{i,j=1,i\neq j}^{N} w\big(|\mathbf{r}_i - \mathbf{r}_j|\big), \tag{1.4}$$

where $w\big(|\mathbf{r}_i - \mathbf{r}_j|\big)$ is the interaction between two electrons, usually taken to be the Coulomb interaction, $w\big(|\mathbf{r}_i - \mathbf{r}_j|\big) = 1/|\mathbf{r}_i - \mathbf{r}_j|$. By writing this Hamiltonian, we are implicitly invoking the Born-Oppenheimer approximation, treating nuclear degrees of freedom classically in the scalar potential $v(\mathbf{r})$.

The core idea of the density functional approach is to extract all physically relevant information from the system through the electron density, circumventing the need to calculate the full wavefunction. The ground state density is given by

$$n_0(\mathbf{r}) = N \int \mathrm{d}^3 r_2 \ldots \int \mathrm{d}^3 r_N \left|\Psi_{\mathrm{gs}}(\mathbf{r}, \mathbf{r}_2, \ldots, \mathbf{r}_N)\right|^2.$$

To gain some insight on how knowing the density gives sufficient information to extract all useful information from a quantum system, first let's consider a nice example given by Ullrich[80]. Suppose we have a one-electron system; it will satisfy the Schrödinger equation (in atomic units)

$$\left[-\frac{\nabla^2}{2} + v(\mathbf{r})\right] \varphi_j(\mathbf{r}) = \varepsilon_j \varphi_j(\mathbf{r}). \tag{1.5}$$

Typically, one would solve this equation for a given potential $v(\mathbf{r})$ and determine the ground state probability density as $n_0(\mathbf{r}) = |\varphi_0(\mathbf{r})|^2$. Suppose we were instead given a density function $n_0(\mathbf{r})$ to start, and were asked to find in which potential $v(\mathbf{r})$ this would be the ground state density. If we assume that the wavefunction is real so that we can write $\varphi_0 = \sqrt{n_0(\mathbf{r})}$, and we also shift the energy scale so that $\varepsilon_0 = 0$, we can invert the Schrödinger equation, Eq. (1.5) to get

$$v(\mathbf{r}) = \frac{\nabla^2 n_0(\mathbf{r})}{4 n_0(\mathbf{r})} - \frac{|\nabla n_0(\mathbf{r})|^2}{8 n_0(\mathbf{r})^2}. \tag{1.6}$$

So, in this simple one-electron system we have reconstructed the potential given the ground-state density $n_0(\mathbf{r})$. This also means that we have reconstructed the Hamiltonian of the system, since we already know the kinetic energy operator. Then, using the Schrödinger equation we can solve the eigenvalue problem to obtain all the wavefunctions. The chain is represented as follows:

$$n_0(\mathbf{r}) \to v(\mathbf{r}) \to \hat{H} \to \{\varphi_j\}. \tag{1.7}$$

Since everything we could want to know about the ground state of the system is stored in the wavefunction, we see from the above chain that the unique progression from the density means that all the information also has to be in the density. This was easy to show in a one-electron system, but it is not the case in a general, interacting, many-electron system. Incredibly, even in this far more complex case, there exists a unique potential for each well-behaved density function such that it is the ground state density in this potential. This leads us to the Hohenberg-Kohn theorem.

1.2.1 HOHENBERG-KOHN THEOREM

The Hohenberg-Kohn theorem[24] reveals the surprising amount of information stored in the ground-state density of a many-body system through a proof of its one-to-one mapping to external potentials and the eigenfunction spectrum of the system.

Hohenberg-Kohn theorem. *In a finite, interacting N-electron system with a given particle-particle interaction, there exists a one-to-one correspondence between the external potential $v(\mathbf{r})$ and the ground-state density $n_0(\mathbf{r})$. In other words, the external potential is a unique functional of the ground-state density, $v[n_0](\mathbf{r})$, up to an arbitrary additive constant.*[79]

Proof. The proof hinges on a contradiction arising from the application of the Rayleigh-Ritz variational principle. We consider two potentials $v(\mathbf{r})$ and $v'(\mathbf{r})$ different if they are not related by a constant shift, $v'(\mathbf{r}) \neq v(\mathbf{r}) + c$.

For the first part of the proof, we show that two different potentials cannot reproduce the same ground-state wavefunction (*i.e.* Ψ_0 and Ψ_0' must differ more than a trivial phase factor). To prove this, assume that Ψ_0 and Ψ_0' are the same and subtract their respective many-body Schrödinger equations (1.1). This results in the equality $\hat{V} - \hat{V}' = E_0 - E_0'$, in contradiction with the requirement that the potentials differ by more than just a constant shift. Thus, the relationship between potentials and wavefunctions is unique.

For the second part of the proof, we need to show that two different ground-state wavefunctions produce different ground-state densities, *i.e.* Ψ_0 and Ψ_0', differing by more than just a phase factor, cannot both map to the ground-state density $n_0(\mathbf{r})$. Again, we use a proof by contradiction. Suppose that both Ψ_0 and Ψ_0' (coming from Schrödinger equations with different potentials v and v') produce the same density $n_0(\mathbf{r})$. The ground-state energy associated with $v'(\mathbf{r})$ is given by

$$E_0' = \left\langle \Psi_0' \middle| \hat{H}' \middle| \Psi_0' \right\rangle . \tag{1.8}$$

Using the Rayleigh-Ritz variational principle and the fact that Ψ_0 and Ψ_0' differ nontrivially, we have the inequality

$$E_0' < \langle \Psi_0 | \hat{H}' | \Psi_0 \rangle = \langle \Psi_0 | \hat{H} + \hat{V}' - \hat{V} | \Psi_0 \rangle \tag{1.9}$$

$$= E_0 + \int \mathrm{d}^3 r \left[v'(\mathbf{r}) - v(\mathbf{r}) \right] n_0(\mathbf{r}). \tag{1.10}$$

Note that we are using a strict inequality; we are assuming that the ground state is not degenerate. The proof

can be extended to degenerate ground states [14]. Now, we can interchange the primes and unprimes in Eq. (1.10) to find the other inequality that holds for the ground-state energy of the unprimed system:

$$E_0 < \langle \Psi_0' | \hat{H} | \Psi_0' \rangle = \langle \Psi_0' | \hat{H}' + \hat{V} - \hat{V}' | \Psi_0' \rangle \tag{1.11}$$

$$= E_0' + \int \mathrm{d}^3 r \left[v(\mathbf{r}) - v'(\mathbf{r}) \right] n_0(\mathbf{r}). \tag{1.12}$$

Note that we are supposing that both wavefunctions have the same density in the last equality. Adding Eqs. (1.10) and (1.12) gives

$$E_0 + E_0' < E_0 + E_0', \tag{1.13}$$

a contradiction. Hence, we have proven that Ψ_0 and Ψ_0' must give different densities n_0 and n_0'. In the first part of the proof we have also shown that Ψ_0 and Ψ_0' come from different potentials v and v', so taken together we have the existence of a unique one-to-one correspondence between the potentials and ground-state densities:

$$n_0(\mathbf{r}) \leftrightarrow v(\mathbf{r}), \tag{1.14}$$

which is the Hohenberg-Kohn theorem. □

Formally, we can write $\hat{V}[n_0]$, *i.e.* the external potential is a functional of the ground state density. Note that the kinetic and electron-interaction operators $\hat{T}$ and $\hat{W}$ are fixed, implying that the Hamiltonian $\hat{H}$ is a functional of the ground-state density, and that all eigenstates of the system also become ground-state-density functionals, $\Psi_j[n_0]$, in the same logical chain as in the one-electron system considered earlier, Eq. (1.7),

$$n_0(\mathbf{r}) \to \hat{V}[n_0] \to \hat{H}[n_0] \to \left\{ \Psi_j[n_0] \right\}. \tag{1.15}$$

This is quite remarkable as it shows that in principle the ground-state density contains all the information you could want to know about a stationary many-body system. For example, Eq. (1.15) shows that any observable can be written as a density functional. In particular, we can write the total energy of a system as a

density functional:

$$E_{v_0}[n] = \langle \Psi[n] | \hat{T} + \hat{V}_0 + \hat{W} | \Psi[n] \rangle \,, \tag{1.16}$$

where n is some N-electron density and $\Psi[n]$ is the ground-state wavefunction which reproduces this density. The energy functional (1.16) is minimized by the *actual* ground-state density n_0 corresponding to v_0, and then becomes equal to the ground state energy:

$$\begin{aligned} E_{v_0}[n] &> E_0 \quad \text{for} \quad n(\mathbf{r}) \neq n_0(\mathbf{r}), \\ E_{v_0}[n] &= E_0 \quad \text{for} \quad n(\mathbf{r}) = n_0(\mathbf{r}). \end{aligned} \tag{1.17}$$

1.2.2 KOHN-SHAM FORMALISM

So, now we have shown that our understanding of a many-body system can be formulated in terms of the ground state density, which is an enormous computational simplification since the density is only a function of D variables while the wavefunction was a function of DN variables. However, unless we can find a shortcut to obtaining the density, we would still be stuck solving the full many-body problem to get the ground state wavefunction anyways.

This is where the *Kohn-Sham formalism* comes into play. The essential idea is to create a noninteracting system with an *effective* potential which captures the many-body interactions, *defined* to be the the potential which reproduces the exact ground-state density of the *interacting* system. This formalism relies on the Hohenberg-Kohn theorem, Eq. (1.14), and the Rayleigh-Ritz minimum principle formulated in terms of densities, Eq. (1.17).

First of all, let us write the total energy functional of the interacting system, Eq. (1.16), a different way:

$$\begin{aligned} E_{v_0}[n] &= T[n] + \int \mathrm{d}^3 r \, v_0(\mathbf{r}) n(\mathbf{r}) + W[n] \\ &= T_s[n] + \int \mathrm{d}^3 r \, v_0(\mathbf{r}) n(\mathbf{r}) + (T[n] - T_s[n] + W[n]) \\ &\equiv T_s[n] + \int \mathrm{d}^3 r \, v_0(\mathbf{r}) n(\mathbf{r}) + E_{\mathrm{H}}[n] + E_{\mathrm{xc}}. \end{aligned} \tag{1.18}$$

Here, $T[n]$ is the kinetic energy functional of the *interacting* system, whereas $T_s[n]$ is the kinetic energy functional of a *noninteracting* system. We do not know $T[n]$ or $T_s[n]$ as explicit density functionals, but we will soon see how to deal with them. Furthermore, in the last step of the above equation, Eq. (1.18), we

introduce the Hartree energy,

$$E_{\mathrm{H}} = \frac{1}{2}\int \mathrm{d}^3 r \int \mathrm{d}^3 r' \, \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}, \tag{1.19}$$

and the so-called *exchange-correlation* (xc) energy,

$$E_{\mathrm{xc}} = T[n] - T_s[n] + W[n] - E_{\mathrm{H}}[n], \tag{1.20}$$

which collects the remaining energy terms. Note that the Hartree energy is simply the classical electrostatic energy of the density configuration, while the xc energy is the difference between the kinetic energy functionals in the noninteracting and interacting systems, plus the electron-electron interactions without the classical Hartree component.

Now, we know from the Rayleigh-Ritz minimum principle (1.17) that the true ground state density for the interacting system will be the minimum of the total energy functional (1.18). Let us now introduce a general form for the density, and then use the Rayleigh-Ritz minimum principle to determine when it becomes equal to the *exact* ground state density:

$$n(\mathbf{r}) = \sum_{j=1}^{N} \left|\varphi_j(\mathbf{r})\right|^2, \tag{1.21}$$

where yet-to-be-determined single-particle orbitals are normalized to one,

$$\int \mathrm{d}^3 r \left|\varphi_j(\mathbf{r})\right|^2 = 1. \tag{1.22}$$

Note that this form for the density is general since the functions φ_j can certainly reproduce any density $n'(\mathbf{r})$ that describes N particles,

$$\int \mathrm{d}^3 r \, n'(\mathbf{r}) = N, \tag{1.23}$$

as long as the orbitals are orthonormal. There are many ways that the single-particle orbitals φ_j can reproduce an N-particle density, and that is the space we wish to minimize $E_{v_0}[n]$ over in order to find the ground-state density.

So, now we wish to minimize $E_{v_0}[n]$ using the form for $n(\mathbf{r})$ given in Eq. (1.21). We could set the functional derivative of $E_{v_0}[n]$ with respect to $n(\mathbf{r})$ equal to zero, and there are proofs of the Kohn-Sham

equation that do this, but instead we will use a nice trick that can arguably lead to a cleaner proof.

First let us make a short digression to investigate a trick that is often used but infrequently explained. Note that despite $\varphi_j(\mathbf{r})$ possibly being complex, the energy functional (1.18) always has to be real.

Consider minimizing a function $f(z)$ of a single complex variable z. This is effectively minimizing a real function of two independent real variables, the real and imaginary parts of $z \equiv z_r + \mathrm{i}z_i$. In order to minimize $f(z)$, we need to compute the two partial derivatives $\frac{\partial f(z_r,z_i)}{\partial z_r}\big|_{z_i}$ and $\frac{\partial f(z_r,z_i)}{\partial z_i}\big|_{z_r}$ which are both real since f is real. We can equivalently think of f as a function of z and z^*, noting that $z_r = (z+z^*)/2$ and $z_i = (z-z^*)/2\mathrm{i}$. Using the chain rule to differentiate $f(z,z^*)$ with respect to z^* while treating z as a constant,

$$\begin{aligned}\left.\frac{\partial f}{\partial z^*}\right|_z &= \left.\frac{\partial f}{\partial z_r}\right|_{z_i}\left.\frac{\partial z_r}{\partial z^*}\right|_z + \left.\frac{\partial f}{\partial z_i}\right|_{z_r}\left.\frac{\partial z_i}{\partial z^*}\right|_z \\ &= \frac{1}{2}\left(\left.\frac{\partial f}{\partial z_r}\right|_{z_i} + \mathrm{i}\left.\frac{\partial f}{\partial z_i}\right|_{z_r}\right).\end{aligned} \tag{1.24}$$

Notice that the real and imaginary components of $\frac{\partial f}{\partial z^*}\big|_z$ give us both derivatives $\frac{\partial f(z_r,z_i)}{\partial z_r}\big|_{z_i}$ and $\frac{\partial f(z_r,z_i)}{\partial z_i}\big|_{z_r}$ *simultaneously*. In particular, in order to minimize over all possible values of $z = z_r + \mathrm{i}z_i$, we need just one equation:

$$0 = \left.\frac{\partial f}{\partial z^*}\right|_z. \tag{1.25}$$

Now, returning to our proof, we saw above that minimizing (or, strictly speaking, finding stationary points of) a real function of a complex variable boils down to setting its derivative with respect to the complex conjugate z^* equal to zero, while holding the other variable z fixed. The same holds for functional derivatives. So, requiring the variation in the energy functional (1.18) with respect to the density to equal zero amounts to requiring that the variation in energy due to each φ_j^* equals zero. Additionally, we include the normality constraint of each orbital $\varphi_j(\mathbf{r})$ with the Lagrange multipliers λ_j and take the variation:

$$\begin{aligned}0 &= \frac{\delta E_{v_0}[n]}{\delta\varphi_j^*(\mathbf{r})} - \frac{\delta}{\delta\varphi_j^*(\mathbf{r})}\sum_i^N \lambda_i \int \varphi_i^*(\mathbf{r}')\varphi_i(\mathbf{r}')\,\mathrm{d}^3r' \\ &= \frac{\delta T_s[n]}{\delta\varphi_j^*(\mathbf{r})} + v_0(\mathbf{r})\varphi_j(\mathbf{r}) + \frac{\delta E_{\mathrm{H}}[n]}{\delta\varphi_j^*(\mathbf{r})} + \frac{\delta E_{\mathrm{xc}}[n]}{\delta\varphi_j^*(\mathbf{r})} - \lambda_j\varphi_j(\mathbf{r}),\end{aligned} \tag{1.26}$$

and now we evaluate each of the functional derivatives. First note that it is straightforward to write down

$T_s[n]$ as an explicit functional of the orbitals,

$$T_s[n] = -\frac{1}{2}\int \mathrm{d}^3\mathbf{r} \sum_{j=1}^{N} \varphi_i^*(\mathbf{r})\nabla^2\varphi_i(\mathbf{r}), \tag{1.27}$$

so that its functional derivative with respect to $\varphi_j^*(\mathbf{r})$ can be easily evaluated:

$$\frac{\delta T_s[n]}{\delta \varphi_j^*(\mathbf{r})} = -\frac{1}{2}\nabla^2\varphi_j(\mathbf{r}). \tag{1.28}$$

As for the Hartree term,

$$\begin{aligned}\frac{\delta E_{\mathrm{H}}[n]}{\delta \varphi_j^*(\mathbf{r})} &= \frac{1}{2}\int \mathrm{d}^3r' \mathrm{d}^3r'' \sum_{i,\ell=1}^{N} \frac{\delta}{\delta \varphi_j^*(\mathbf{r})} \frac{\varphi_i^*(\mathbf{r}')\varphi_i(\mathbf{r}')\varphi_\ell^*(\mathbf{r}'')\varphi_\ell(\mathbf{r}'')}{|\mathbf{r}'-\mathbf{r}''|} \\ &= \int \mathrm{d}^3r' \sum_{i=1}^{N} \frac{\varphi_i^*(\mathbf{r}')\varphi_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\varphi_j(\mathbf{r}) \qquad (1.29) \\ &= v_{\mathrm{H}}(\mathbf{r})\varphi_j(\mathbf{r}), \qquad (1.30)\end{aligned}$$

where the Hartree potential is defined as

$$v_{\mathrm{H}}(\mathbf{r}) = \int \mathrm{d}^3r' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}. \tag{1.31}$$

Finally, we simply *define* the exchange-correlation potential to be

$$v_{\mathrm{xc}}[n](\mathbf{r}) \equiv \frac{\delta E_{\mathrm{xc}}[n]}{\delta n(\mathbf{r})}, \tag{1.32}$$

so that

$$\frac{\delta E_{\mathrm{xc}}[n]}{\delta \varphi_j^*(\mathbf{r})} = \frac{\delta E_{\mathrm{xc}}[n]}{\delta n(\mathbf{r})}\frac{\delta n(\mathbf{r})}{\delta \varphi_j^*(\mathbf{r})} = v_{\mathrm{xc}}[n](\mathbf{r})\varphi_j(\mathbf{r}). \tag{1.33}$$

So, substituting Eqs. (1.28), (1.30), and (1.33) back into our stationary point equation (1.26), we are left with the *Kohn-Sham equation*,

$$\begin{aligned}&\left[-\frac{\nabla^2}{2} + v_0(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r}) + v_{\mathrm{xc}}[n](\mathbf{r})\right]\varphi_j(\mathbf{r}) = \lambda_j\varphi_j(\mathbf{r}), \\ &\left[-\frac{\nabla^2}{2} + v_s[n](\mathbf{r})\right]\varphi_j(\mathbf{r}) = \varepsilon_j\varphi_j(\mathbf{r}),\end{aligned} \tag{1.34}$$

where in the final line we have defined the single-particle effective potential,

$$v_s[n](\mathbf{r}) \equiv v_0(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r}) + v_{\mathrm{xc}}[n](\mathbf{r}), \tag{1.35}$$

the density is given in terms of the eigenfunctions

$$n(\mathbf{r}) = \sum_{j=1}^{N} \left|\varphi_j(\mathbf{r})\right|^2, \tag{1.36}$$

and we have also identified the Lagrange multipliers λ_j as the Kohn-Sham eigenenergies ε_j.

Solving the Kohn-Sham equations (1.34) and (1.36) self-consistently, *i.e.* finding sets of $\{\varphi_j\}$ such that both equations simultaneously hold, given that the Kohn-Sham Hamiltonian is dependent on $\{\varphi_j\}$ through the density $n(\mathbf{r})$, gives the densities that are stationary points of the total energy functional (1.18). Of these possible sets of $\{\varphi_j\}$, one of them will minimize $E_{v_0}[n]$, and this set thus produces *the* ground state density of the *interacting* many-body system, $n_0(\mathbf{r})$.

In practice, by performing a self-consistency loop on the Kohn-Sham equations, oftentimes aided by clever density mixing schemes, one usually ends up with the ground state density (as opposed to an excited state which can also be a stationary state of the total energy functional). This is possibly due to the global minimum of the energy functional being easier to reach by chance.

1.2.3 Discussion of Kohn-Sham Formalism

Here we note a few properties of the Kohn-Sham formalism that are also applicable to the TDDFT case we will eventually discuss. The Kohn-Sham system is noninteracting so its total N-particle wavefunction can be written as a Slater determinant:

$$\Psi_{\mathrm{gs}}^{\mathrm{KS}}(\mathbf{r}_1,\ldots,\mathbf{r}_N) = \frac{1}{\sqrt{N!}}\det\{\varphi_j\} = \frac{1}{\sqrt{N!}}\begin{vmatrix} \varphi_1(\mathbf{r}_1) & \varphi_2(\mathbf{r}_1) & \cdots & \varphi_N(\mathbf{r}_1) \\ \varphi_1(\mathbf{r}_2) & \varphi_2(\mathbf{r}_2) & \cdots & \varphi_N(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{r}_N) & \varphi_2(\mathbf{r}_N) & \cdots & \varphi_N(\mathbf{r}_N) \end{vmatrix}. \tag{1.37}$$

It is also important to remember that the Kohn-Sham system is only designed to reproduce the correct ground state density. It is tenuous to assign physical meaning to the Kohn-Sham eigenstates φ_j and its energy eigenvalues ε_j. Same goes for the eigenenergy differences $\varepsilon_\alpha - \varepsilon_j$, although they are frequently interpreted as excitation energies when α indexes an unoccupied state. However, in extended systems,

Janak's theorem (Appendix A) gives an approximate interpretation of the single particle energy ε_j in relation to the first vertical ionization energy and electron affinity[28]. However, ionization energies computed in this manner are typically quite poor with present-day exchange-correlation functionals[22].

As it will be useful later on, we note that the external potential behaves as $v_0(\mathbf{r}) \to -N/r$ and the Hartee potential as $v_\mathrm{H}(\mathbf{r}) \to N/r$ for $r \to \infty$. Thus, if an electron is found far away, in the outer regions of the system, it will still feel a Coulomb potential of the remaining positive ion, so asymptotically the xc potential has to behave as

$$v_\mathrm{xc}(\mathbf{r}) \sim -\frac{1}{r}. \tag{1.38}$$

Note that this makes sense in the context of a one-electron system. The Hartree and xc potential would cancel exactly, leaving just the external potential acting on the electron.

Furthermore, it is easy to extend the Kohn-Sham formalism to two spin channels, resulting in a density for up spin and a density for down spin. This allows for systems that break spin symmetry to be calculated as well, and for specific molecular magnetization states to be computed.

1.2.4 DFT IN PRACTICE

DFT is in principle exact, but the difficulty lies in the exchange-correlation energy and the corresponding potential. These are not known exactly, so they must be approximated.

We can write the xc energy of a system in terms of the exchange-correlation energy density,

$$E_\mathrm{xc}[n] = \int \mathrm{d}^3 r\, e_\mathrm{xc}[n](\mathbf{r}), \tag{1.39}$$

where the xc energy density $e_\mathrm{xc}[n](\mathbf{r})$ is, in general, nonlocal. We wish to approximate the xc energy density. A simple approximation that has had much success in a variety of systems is the local density approximation (LDA). The LDA has the following form:

$$E_\mathrm{xc}^\mathrm{LDA}[n] = \int \mathrm{d}^3 r\, e_\mathrm{xc}^h(n(\mathbf{r})). \tag{1.40}$$

The LDA xc energy density is just the xc energy density of a homogeneous electron liquid of electron density $\bar{n}$, evaluated at the local density of the inhomogeneous system, $e_\mathrm{xc}^h(n(\mathbf{r})) = e_\mathrm{xc}^h(\bar{n})\,|_{\bar{n}=n(\mathbf{r})}$. This approximation becomes increasingly valid the slower a system varies in space, since it would be

approaching a true homogeneous electron gas. We need the xc energy density of a homogeneous electron liquid, which can be split up into an exchange part and a correlation part:

$$e^h_{\mathrm{xc}}(\bar{n}) = e^h_{\mathrm{x}}(\bar{n}) + e^h_{\mathrm{c}}(\bar{n}). \tag{1.41}$$

The exchange energy density can be calculated exactly using Hartree-Fock theory, giving

$$e^h_{\mathrm{x}}(\bar{n}) = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\bar{n}^{4/3}. \tag{1.42}$$

The correlation energy density $e^h_{\mathrm{c}}(\bar{n})$ is not exactly known, but very accurate numerical results exist from quantum Monte Carlo calculations[80].

One significant drawback of the LDA approximation is that the xc potential goes to zero exponentially fast instead of as $-1/r$, as we saw was required in Eq. (1.38). This indicates the presence of self-interactions, and generally causes the Kohn-Sham eigenvalues to be too low in magnitude.

The LDA can be improved by considering gradients of the density, leading to the generalized gradient approximation, (GGA), which has the generic form

$$E^{\mathrm{GGA}}_{\mathrm{xc}}[n] = \int \mathrm{d}^3 r\, e_{\mathrm{xc}}(n(\mathbf{r}), \boldsymbol{\nabla} n(\mathbf{r})). \tag{1.43}$$

It still suffers from the wrong asymptotic behavior, so there exist schemes to mix in some exact-exchange calculated from Hartree-Fock theory, resulting in the so-called hybrid functionals. Another class of hybrid functionals, the *range-separated hybrids*, separate the Coulomb interaction into a short-range and long-range part. As such, one can have the correct long-range xc potential asymptotic behavior.

1.3 Time-Dependent Density Functional Theory

In a time-dependent system of N interacting electrons, the total Hamiltonian becomes

$$\hat{H}(t) = \hat{T} + \hat{V}(t) + \hat{W}, \tag{1.44}$$

where the only difference with Eq. (1.2) is that the potential operator is now time-dependent,

$$\hat{V}(t) = \sum_{j=1}^{N} v(\mathbf{r}_j, t). \tag{1.45}$$

The time evolution of the system is governed by the time-dependent many-body Schrödinger equation,

$$\mathrm{i}\frac{\partial}{\partial t}\Psi(\mathbf{x}_1,\ldots,\mathbf{x}_N,t) = \hat{H}(t)\Psi(\mathbf{x}_1,\ldots,\mathbf{x}_N,t), \tag{1.46}$$

which propagates an initial state $\Psi(t_0) \equiv \Psi_0$ to some final time t.

Just like how the foundations of ground state DFT are established by the Hohenberg-Kohn theorem (1.14), the theoretical basis of TDDFT is currently the Runge-Gross theorem[63]. It is more difficult in the time-dependent case because we no longer can make use of the Rayleigh-Ritz minimum principle given that the system will not be in the ground state. The general idea behind the Runge-Gross theorem, in analogue to the Hohenberg-Kohn theorem, is that if two N-electron systems start from the same initial state but are subject to different time-dependent potentials, their respective time-dependent densites will be different.

In this case we consider time-dependent potentials to be different if they differ more than just a time-dependent constant,

$$v(\mathbf{r},t) - v'(\mathbf{r},t) \neq c(t), \tag{1.47}$$

for $t > t_0$. One can easily show that if they did just differ by such a constant, it would only materialize as a time-dependent phase factor difference for all wavefunctions which would cancel out in the calculation of observables.

There are some limitations to the Runge-Gross theorem however. It only applies to potentials that can be expanded in a Taylor-series about the initial time,

$$v(\mathbf{r},t) = \sum_{k=0}^{\infty} \frac{v_k(\mathbf{r})}{k!}(t-t_0)^k. \tag{1.48}$$

But, for such potentials, there exists a unique 1-to-1 correspondence between potential $v(\mathbf{r},t)$ and density $n(\mathbf{r},t)$, for a fixed initial wavefunction Ψ_0.

Note the dependence on the initial state Ψ_0, however, which means that the potential will be a functional of both the density and the initial state wavefunction Ψ_0:

$$v(\mathbf{r},t) = v[n,\Psi_0](\mathbf{r},t), \tag{1.49}$$

and similarly for any observables formed (recall that a one-to-one correspondence between density and

potential leads to the wavefunctions being density functionals, Eq. (1.15)):

$$O(t) = \langle \Psi[n,\Psi_0](t)|\hat{O}|\Psi[n,\Psi_0](t)\rangle = O[n,\Psi_0](t). \tag{1.50}$$

In the case where our initial state is the ground state of the system, and when the potential is turned on at t_0,

$$v(\mathbf{r},t) = v_0(\mathbf{r}) + \theta(t-t_0)v_1(\mathbf{r},t), \tag{1.51}$$

the Hohenberg-Kohn theorem tells us that the initial state $\Psi_{\mathrm{gs}}[n]$ becomes a functional of the density as well, making all observables purely density functionals.

Runge-Gross is only an existence theorem. It shows that TDDFT exists in the sense that, in principle, all one needs to know is the density of a system in order to obtain observables. It does not guarantee that we can use the same trick as in the Kohn-Sham formalism, calculating the density of the full interacting system with a much simpler noninteracting system. This is where the van Leeuwen theorem steps in, paving the way towards the time-dependent Kohn-Sham equations.

1.3.1 Runge-Gross Theorem

Before we launch into the Runge-Gross theorem[63], we need a few equations resulting from continuity and local conservation laws.

Recall that the density operator of an N-electron system is defined in position space as

$$\hat{n}(\mathbf{r}) = \sum_{l=1}^{N} \delta(\mathbf{r}-\mathbf{r}_l), \tag{1.52}$$

and the (paramagnetic) current-density operator is

$$\hat{\mathbf{j}}(\mathbf{r}) = \frac{1}{2\mathrm{i}}\sum_{l=1}^{N}[\nabla_l\delta(\mathbf{r}-\mathbf{r}_l) + \delta(\mathbf{r}-\mathbf{r}_l)\nabla_l]. \tag{1.53}$$

The expectation values of these operators give their time-dependent counterparts,

$$\begin{aligned} n(\mathbf{r},t) &= \langle\Psi(t)|\hat{n}(\mathbf{r})|\Psi(t)\rangle \\ \mathbf{j}(\mathbf{r},t) &= \langle\Psi(t)|\hat{j}(\mathbf{r})|\Psi(t)\rangle\,. \end{aligned} \tag{1.54}$$

Consider the equation of motion of the expectation value of an operator $\hat{O}(t)$:

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}\langle\Psi(t)|\hat{O}(t)|\Psi(t)\rangle = \left\langle \Psi(t)\middle|\mathrm{i}\frac{\partial}{\partial t}\hat{O}(t) + \left[\hat{O}(t),\hat{H}(t)\right]\middle|\Psi(t)\right\rangle. \tag{1.55}$$

Applying this equation to the density operator gives

$$\mathrm{i}\frac{\partial}{\partial t}n(\mathbf{r},t) = \left\langle\Psi(t)\middle|\left[\hat{n}(\mathbf{r}),\hat{H}(t)\right]\middle|\Psi(t)\right\rangle, \tag{1.56}$$

by noting that the density operator does not have explicit time-dependence. We can easily work out the commutator and write the result in terms of the current density:

$$\frac{\partial}{\partial t}n(\mathbf{r},t) = -\boldsymbol{\nabla}\cdot\mathbf{j}(\mathbf{r},t), \tag{1.57}$$

recovering the continuity equation.

We can also obtain the equation of motion for the current density,

$$\mathrm{i}\frac{\partial}{\partial t}\mathbf{j}(\mathbf{r},t) = \left\langle\Psi(t)\middle|\left[\hat{\mathbf{j}},\hat{H}(t)\right]\middle|\Psi(t)\right\rangle. \tag{1.58}$$

It is more difficult to work out the commutator, but still straightforward, giving the following result:

$$\frac{\partial}{\partial t}j_\mu(\mathbf{r},t) = -n(\mathbf{r},t)\frac{\partial}{\partial r_\mu}v(\mathbf{r},t) - F_\mu^{\mathrm{kin}}(\mathbf{r},t) - F_\mu^{\mathrm{int}}(\mathbf{r},t), \tag{1.59}$$

where μ,ν are Cartesian coordinate indices, and the vectors F_μ^{kin} and F_μ^{int} correspond to internal force densities of the many-body system due to kinetic and interaction effects. For the explicit expressions of these two vectors, refer to Ullrich[79], since we will not need them for our purposes.

Now we move on to the Runge-Gross theorem. The time-dependent Schrödinger equation maps from a potential $v(\mathbf{r},t)$ to a time-dependent density $n(\mathbf{r},t)$, for a given initial state Ψ_0. The point of the Runge-Gross theorem is to show that the reverse mapping is unique, *i.e.* a given potential generated in a physical system maps back to a single potential $v(\mathbf{r},t)$. As seen back in the ground state mapping Eq. (1.15), the existence of such a unique reverse mapping allows any observable of the system to be expressed as a functional of the density. The existence of the one-to-one relationship between potential and density is the formal backbone of TDDFT.

The following theorem statement and proof are from Ullrich[79], reworded for our purposes, and with some

contributions from the proof by Marques *et al.*[45].

Runge-Gross Theorem. *Two densities $n(\mathbf{r},t)$ and $n'(\mathbf{r},t)$, evolving from a common initial many-body state Ψ_0 under the influence of two different potentials $v(\mathbf{r},t)$ and $v'(\mathbf{r},t) \neq v(\mathbf{r},t)+c(t)$ (both assumed to be Taylor-expandable around t_0), will start to become different infinitesimally later than t_0. Therefore, there is a one-to-one correspondence between densities and potentials, for any fixed initial many-body state.*

Proof. First of all, note that if the two potentials v and v' only differ by a time-dependent constant, then the wavefunctions that they correspond to will only differ by a phase which will cancel out in the calculation of densities. Next, this proof is only formulated for potentials that can be expanded in a Taylor series about the initial time:

$$v(\mathbf{r},t) = \sum_{k=0}^{\infty} \frac{1}{k!} v_k(\mathbf{r})(t-t_0)^k, \tag{1.60}$$

such potentials are referred to as t-TE. The primed potential must also be t-TE, naturally, with expansion coefficients $v'_k(\mathbf{r})$. With this form established, we can then express the assumption that v and v' differ by more than just a function $c(t)$ through the requirement that there exists a smallest integer $k \geq 0$ such that

$$v_k(\mathbf{r}) - v'_k(\mathbf{r}) \neq \text{const.} \tag{1.61}$$

We do not make any assumptions about the radius of convergence of the potential series expansions, other than it being greater than zero. That said, we did not require the initial state of the system Ψ_0, to be an eigenstate of the initial potential $v(\mathbf{r},t_0)$, so the case of sudden switching is included in the proof. The Runge-Gross proof will proceed in two steps. First we will establish the uniqueness of the current densities, and then proceed with this knowledge to show that the densities differ.

Step 1. Starting with the equation of motion for the current density, (1.58), we subtract the primed equation from the unprimed to obtain

$$\begin{aligned} \frac{\partial}{\partial t}\{\mathbf{j}(\mathbf{r},t) - \mathbf{j}'(\mathbf{r},t)\}\Big|_{t=t_0} &= -\mathrm{i}\left\langle \Psi_0 \middle| \left[\hat{\mathbf{j}}, \hat{H}(t_0) - \hat{H}'(t_0)\right] \middle| \Psi_0 \right\rangle \\ &= -n(\mathbf{r},t_0)\boldsymbol{\nabla}\big\{v(\mathbf{r},t_0) - v'(\mathbf{r},t_0)\big\}, \end{aligned} \tag{1.62}$$

where the second step results from applying the evaluated commutator form of the equation of motion (1.59), noting that since we are starting from the same initial state, the internal kinetic and interaction forces in Eq. (1.59) are identical in both systems and cancel out. Now, if the condition (1.61) is satisfied for $k = 0$,

we already see that the right-hand side of Eq. (1.62) cannot vanish identically and hence the currents $\mathbf{j}$ and $\mathbf{j}'$ will become different infinitesimally later than $t = 0$. But, it may be that at $t = t_0$ the two potentials are the same, but only diverge later in time. However, due to the restriction on the form of the potentials, namely that they are t-TE, this difference must manifest itself in differences in higher time derivatives of the potential. So, if the smallest integer k for which the condition (1.61) holds is greater than zero, then we use the general equation of motion (1.55) $(k+1)$ times. That is, as for $k = 0$ above where we used $\hat{O}(t) = \hat{\mathbf{j}}(\mathbf{r})$ in Eq. (1.55), for $k = 1$ we take $\hat{O}(t) = -i\left[\hat{\mathbf{j}}(\mathbf{r}), \hat{H}(t)\right]$; for general k,
$\hat{O}(t) = (-\mathrm{i})^k \left[\left[\left[\hat{\mathbf{j}}(\mathbf{r}), \hat{H}(t)\right], \hat{H}(t)\right] \ldots, \hat{H}(t)\right]_k$ meaning there are k nested commutators to take. After some algebra[45]:

$$\frac{\partial^{k+1}}{\partial t^{k+1}} \left\{\mathbf{j}(\mathbf{r},t) - \mathbf{j}'(\mathbf{r},t)\right\}\Big|_{t=t_0} = -n(\mathbf{r},t_0)\boldsymbol{\nabla}\left\{v_k(\mathbf{r}) - v_k'(\mathbf{r})\right\}, \tag{1.63}$$

which tells us that $\mathbf{j}(\mathbf{r},t) \neq \mathbf{j}'(\mathbf{r},t)$ for $t > t_0$.

Step 2. Now, we need to show that having different current densities (*i.e.* Step 1) means that the densities themselves will be different. To achieve this, we start with the continuity equation (1.57) and calculate its $(k+1)$th time derivative:

$$\begin{aligned}\frac{\partial^{k+2}}{\partial t^{k+2}} \left\{n(\mathbf{r},t) - n'(\mathbf{r},t)\right\}\Big|_{t=t_0} &= -\boldsymbol{\nabla}\cdot\frac{\partial^{k+1}}{\partial t^{k+1}} \left\{\mathbf{j}(\mathbf{r},t) - \mathbf{j}'(\mathbf{r},t)\right\}\Big|_{t=t_0} \\ &= -\boldsymbol{\nabla}\cdot(n_0(\mathbf{r})\boldsymbol{\nabla} w_k(\mathbf{r})),\end{aligned} \tag{1.64}$$

where we have defined $w_k(\mathbf{r}) = v_k(\mathbf{r}) - v_k'(\mathbf{r})$ and $n_0(\mathbf{r}) = n(\mathbf{r},t_0)$. We now need to show that the right-hand side of Eq. (1.64) must be nonzero as long as the condition (1.61) holds, *i.e.* $w_k(\mathbf{r})$ is nonzero. For this, consider the following relation which follows from the divergence theorem and the product rule for derivatives (to produce the left-hand side and the first term on the right-hand side):

$$\int \mathrm{d}^3 r\, n_0(\mathbf{r})(\boldsymbol{\nabla} w_k(\mathbf{r}))^2 = -\int \mathrm{d}^3 r\, w_k(\mathbf{r})\boldsymbol{\nabla}\cdot(n_0(\mathbf{r})\boldsymbol{\nabla} w_k(\mathbf{r})) + \oint \mathrm{d}\mathbf{S}\cdot(n_0(\mathbf{r}) w_k(\mathbf{r})\boldsymbol{\nabla} w_k(\mathbf{r})). \tag{1.65}$$

The surface integral on the right-hand side is crucial to the validity of the Runge-Gross proof. It has to vanish for all physically realistic potentials in order for the proof to hold. Fortunately, it can be shown that all potentials which arise from finite normalizable external charge distributions go to zero at least as fast as $1/r$ so that the surface integral indeed vanishes[21]. Note that the vanishing of this term also requires that the system is finite so that the density vanishes at infinity. However, it can be proven to work for periodic

systems as well, provided the external potential is also periodic.

So, suppose we have gotten rid of the surface integral. The left hand side of Eq. (1.65) does not vanish, since $w_k \neq 0$, and the integrand is therefore nonnegative everywhere. As a consequence, the first integral on the right is also nonvanishing, which, taken alongside $w_k \neq 0$, immediately implies that $\boldsymbol{\nabla} \cdot (n_0(\mathbf{r}) \boldsymbol{\nabla} w_k(\mathbf{r})) \neq 0$. Thus, from Eq. (1.64) we conclude that the densities must differ infinitesimally later than t_0. This concludes the Runge-Gross proof. □

1.3.2 The van Leeuwen Theorem

The Runge-Gross theorem formally establishes the existence of TDDFT, allowing us to speak of density functionals in a meaningful way when it comes to the time-dependent problem. However, it would be great if we could use the same trick as in the ground state theory, namely, have a noninteracting system that can reproduce the same density as the interacting system. The Runge-Gross theorem does not guarantee that this is possible, so we need to go further.

Here we provide the statement of the van Leeuwen theorem[82] in the words of Ullrich[79].

Van Leeuwen Theorem. *For a time-dependent density $n(\mathbf{r},t)$ associated with a many-body system with a given particle-particle interaction $w(|\mathbf{r}-\mathbf{r}'|)$, external potential $v(\mathbf{r},t)$, and initial state Ψ_0, there exists a different many-body system featuring an interaction $w'(|\mathbf{r}-\mathbf{r}'|)$ and a unique external potential $v'(\mathbf{r},t)$ [up to a purely time-dependent $C(t)$] which reproduces the same time-dependent density. The initial state Ψ_0' in this system must be chosen such that it correctly yields the given density and its time derivative at the inital time. Both the external potential and densities have to be t-TE (Taylor expandable around t_0) in this proof.*

The following proof is from the original van Leeuwen paper[82] with minor modifications and commentary.

Proof. Consider the Hamiltonian

$$\hat{H}(t) = \hat{T} + \hat{V}(t) + \hat{W} \tag{1.66}$$

of a finite many-particle system, where $\hat{T}$ is the kinetic energy, $\hat{V}(t)$ is the external potential, and $\hat{W}$ is the

two-particle interaction. We work in the second quantization formalism where

$$\hat{T} = -\frac{1}{2}\int \mathrm{d}^3 r\, \hat{\psi}^\dagger(\mathbf{r})\nabla^2\hat{\psi}(\mathbf{r}), \tag{1.67}$$

$$\hat{V} = \int \mathrm{d}^3 r\, v(\mathbf{r}t)\,\hat{\psi}^\dagger(\mathbf{r})\,\hat{\psi}(\mathbf{r}), \tag{1.68}$$

$$\hat{W} = \int \mathrm{d}^3 r\, \mathrm{d}^3 r'\, w\big(|\mathbf{r}-\mathbf{r}'|\big)\,\hat{\psi}^\dagger(\mathbf{r})\,\hat{\psi}^\dagger\big(\mathbf{r}'\big)\,\hat{\psi}(\mathbf{r}')\,\hat{\psi}(\mathbf{r}). \tag{1.69}$$

The external potential $v(\mathbf{r}t)$ is assumed to have the form

$$v(\mathbf{r}t) = \int \mathrm{d}^3 r'\, \frac{Z(\mathbf{r}'t)}{|\mathbf{r}-\mathbf{r}'|}, \tag{1.70}$$

where $Z(\mathbf{r}t)$ describes a finite but arbitrarily large charge distribution. This form is sufficient to describe the potential due to general time-dependent ionic configurations, and it can also be used to simulate uniform external fields in limit configurations with large finite charge disributions. We also assume that $v(\mathbf{r}t)$ is an analytic function of time t. Let us specify an initial state $|\Psi_0\rangle$ at $t = t_0$ and evolve the wavefunction with the Hamiltonian $\hat{H}(t)$. This gives $|\Psi(t)\rangle$, from which the density $n(\mathbf{r}t)$ can be calculated. We can write down the continuity equation

$$\partial_t n(\mathbf{r}t) = -\mathrm{i}\,\langle\Psi(t)|\big[\hat{n}(\mathbf{r}),\hat{H}(t)\big]|\Psi(t)\rangle = -\boldsymbol{\nabla}\cdot\mathbf{j}(\mathbf{r}t), \tag{1.71}$$

where the current operator is given by

$$\hat{\mathbf{j}}(\mathbf{r}) = \frac{1}{2\mathrm{i}}\big\{\,\hat{\psi}^\dagger(\mathbf{r})\boldsymbol{\nabla}\hat{\psi}(\mathbf{r}) - \big[\boldsymbol{\nabla}\hat{\psi}^\dagger(\mathbf{r})\big]\,\hat{\psi}(\mathbf{r})\big\} \tag{1.72}$$

and has expectation value

$$\mathbf{j}(\mathbf{r}t) = \langle\Psi(t)|\hat{\mathbf{j}}(\mathbf{r})|\Psi(t)\rangle\,. \tag{1.73}$$

We can further consider a continuity equation for the current itself,

$$\partial_t \mathbf{j}(\mathbf{r}t) = -\mathrm{i}\,\langle\Psi(t)|\big[\hat{\mathbf{j}}(\mathbf{r}),\hat{H}(t)\big]|\Psi(t)\rangle\,. \tag{1.74}$$

Working out this equation in terms of the momentum-stress tensor

$$\hat{T}_{ik}(\mathbf{r}) = \frac{1}{2}\left\{ \partial_i \hat{\psi}^\dagger(\mathbf{r})\partial_k \hat{\psi}(\mathbf{r}) + \partial_k \hat{\psi}^\dagger(\mathbf{r})\partial_i \hat{\psi}(\mathbf{r}) - \frac{1}{2}\partial_i\partial_k\left[\hat{\psi}^\dagger(\mathbf{r})\hat{\psi}(\mathbf{r})\right] \right\} \tag{1.75}$$

and

$$\hat{W}_k(\mathbf{r}) = \int \mathrm{d}^3 r'\, \hat{\psi}^\dagger(\mathbf{r})\hat{\psi}^\dagger(\mathbf{r}')\partial_k w\big(|\mathbf{r}-\mathbf{r}'|\big)\,\hat{\psi}(\mathbf{r}')\hat{\psi}(\mathbf{r}), \tag{1.76}$$

where the derivatives ∂_k are taken with respect to $\mathbf{r}$, we obtain

$$\partial_t j_k(\mathbf{r}t) = -n(\mathbf{r}t)\partial_k v(\mathbf{r}t) - \sum_i \partial_i T_{ik}(\mathbf{r}t) - W_k(\mathbf{r}t). \tag{1.77}$$

In the above expression the expectation values are defined as

$$T_{ik}(\mathbf{r}t) = \langle \Psi(t)|\hat{T}_{ik}|\Psi(t)\rangle\,, \tag{1.78}$$

$$W_k(\mathbf{r}t) = \langle \Psi(t)|\hat{W}_k|\Psi(t)\rangle\,. \tag{1.79}$$

By taking the divergence of Eq. (1.77) and using Eq. (1.71) we find

$$\partial_t^2 n(\mathbf{r}t) = \boldsymbol{\nabla}\cdot[n(\mathbf{r}t)\boldsymbol{\nabla} v(\mathbf{r}t)] + q(\mathbf{r}t), \tag{1.80}$$

where

$$q(\mathbf{r}t) = \langle \Psi(t)|\hat{q}(\mathbf{r})|\Psi(t)\rangle\,, \tag{1.81}$$

with the operator $\hat{q}(\mathbf{r})$ defined as

$$\hat{q}(\mathbf{r}) = \sum_{i,k}\partial_i\partial_k \hat{T}_{ik}(\mathbf{r}) + \sum_k \partial_k \hat{W}_k(\mathbf{r}). \tag{1.82}$$

Notice that Eq. (1.80) directly relates external potential $v(\mathbf{r}t)$ to $n(\mathbf{r}t)$; it will be the central equation of the proof.

Now we consider a second system with a different two-particle interaction $w'(|\mathbf{r}-\mathbf{r}'|)$. We will seek an external potential $v'(\mathbf{r}t)$ such that the system produces the same density $n(\mathbf{r}t)$ as the original system, subject to the constraint that $v'(\mathbf{r}t)$ vanishes at infinity (which is also satisfied by $v(\mathbf{r}t)$ given the form Eq. (1.70)).

The Hamiltonian of the second (primed) system is

$$\hat{H}'(t) = \hat{T} + \hat{V}'(t) + \hat{W}'. \tag{1.83}$$

Let the initial state of the primed system be denoted $|\Phi_0\rangle$ at $t = t_0$, evolving as $|\Phi(t)\rangle$. It is assumed that $\hat{W}'_k(\mathbf{r}t)$ (as in Eq. (1.76)) and its derivatives are finite. Following the same steps as before, we can produce an equation analogous to Eq. (1.80) from the primed Hamiltonian. We further assume that the primed system has the same density trajectory, so $n'(\mathbf{r}t) = n(\mathbf{r}t)$ and thus

$$\partial_t^2 n(\mathbf{r}t) = \boldsymbol{\nabla} \cdot \left[n(\mathbf{r}t) \boldsymbol{\nabla} v'(\mathbf{r}t) \right] + q'(\mathbf{r}t), \tag{1.84}$$

where $q'(\mathbf{r}t)$ has the expectation value

$$q'(\mathbf{r}t) = \langle \Phi(t) | \hat{q}'(\mathbf{r}) | \Phi(t) \rangle . \tag{1.85}$$

We can subtract Eqs. (1.80) and (1.84) to obtain

$$\boldsymbol{\nabla} \cdot [n(\mathbf{r}t) \boldsymbol{\nabla} \omega(\mathbf{r}t)] = \zeta(\mathbf{r}t), \tag{1.86}$$

with $\omega(\mathbf{r}t) = v(\mathbf{r}t) - v'(\mathbf{r}t)$ and $\zeta(\mathbf{r}t) = q'(\mathbf{r}t) - q(\mathbf{r}t)$. We will use Eq. (1.86) to construct the required $v'(\mathbf{r}t)$ to support the density equality between the primed and unprimed systems supposed earlier. As Eq. (1.86) is a differential equation, to determine a solution for $v'(\mathbf{r}t)$ given the rest of the known quantities we need boundary conditions. The first obvious condition is that the initial states in the two systems must yield the same starting density in order for the density trajectories to be equal:

$$\langle \Phi_0 | \hat{n}(\mathbf{r}) | \Phi_0 \rangle = \langle \Psi_0 | \hat{n}(\mathbf{r}) | \Psi_0 \rangle . \tag{1.87}$$

Since the equation relating density and potential, Eq. (1.80), is second-order in time for $\mathbf{n}(\mathbf{r}t)$, we also need to match the first-order derivatives for the densities in the two systems, *i.e.* $\partial_t n'(\mathbf{r}t) = \partial_t n(\mathbf{r}t)$ at $t = t_0$. We can rewrite this with the usual continuity equation, Eq. (1.71), resulting in

$$\langle \Phi_0 | \boldsymbol{\nabla} \cdot \hat{\mathbf{j}}(\mathbf{r}) | \Phi_0 \rangle = \langle \Psi_0 | \boldsymbol{\nabla} \cdot \hat{\mathbf{j}}(\mathbf{r}) | \Psi_0 \rangle . \tag{1.88}$$

Given the two initial conditions above, we return to discussing the solution of Eq. (1.86). The equation has

no time derivatives, so the time variable can be treated as a parameter (a useful feature for the discussion in Section 2.3.1 where we extend this proof to imaginary time.) The equation is also of a well-known Sturm-Liouville type, with a unique solution for $\omega(\mathbf{r}t)$ if $n(\mathbf{r}t)$ and $\zeta(\mathbf{r}t)$ are given, and we further specify the boundary condition that $\omega(\mathbf{r}t)$ approaches zero at infinity. At $t = t_0$ we have

$$\boldsymbol{\nabla} \cdot [n(\mathbf{r}t_0)\boldsymbol{\nabla}\omega(\mathbf{r}t_0)] = \zeta(\mathbf{r}t_0). \tag{1.89}$$

Since $n(\mathbf{r}t)$ is known at all times and $\zeta(\mathbf{r}t_0)$ can be calculated from the initial states $|\Psi_0\rangle$ and $|\Phi_0\rangle$ there is a unique solution $\omega(\mathbf{r}t_0)$. This also gives $v'(\mathbf{r}t_0) = v(\mathbf{r}t_0) - \omega(\mathbf{r}t_0)$. Now we take the time derivative of Eq. (1.89) at $t = t_0$ resulting in

$$\boldsymbol{\nabla} \cdot \Big[n(\mathbf{r}t_0)\boldsymbol{\nabla}\omega^{(1)}(\mathbf{r})\Big] = \zeta^{(1)}(\mathbf{r}) - \boldsymbol{\nabla} \cdot \Big[n^{(1)}(\mathbf{r})\boldsymbol{\nabla}\omega(\mathbf{r}t_0)\Big], \tag{1.90}$$

where we introduced the following notation for the kth time derivative at $t = t_0$:

$$f^{(k)}(\mathbf{r}) = \partial_t^k f(\mathbf{r}t)\Big|_{t=t_0}. \tag{1.91}$$

All the quantities on the right-hand side of Eq. (1.90) are known since $n(\mathbf{r}t)$ is known at all times and $\omega(\mathbf{r}t_0)$ was determined from Eq. (1.89). The final term $\zeta^{(1)}(\mathbf{r})$ is calculated from the commutators:

$$\zeta^{(1)}(\mathbf{r}) = \partial_t \zeta(\mathbf{r}t)\Big|_{t=t_0} \tag{1.92}$$

$$= \mathrm{i}\,\langle\Psi_0|\big[\hat{q}(\mathbf{r}), \hat{H}(t_0)\big]|\Psi_0\rangle - \mathrm{i}\,\langle\Phi_0|\big[\hat{q}'(\mathbf{r}), \hat{H}'(t_0)\big]|\Phi_0\rangle\,, \tag{1.93}$$

where $\hat{H}'(t_0)$ is known from $v'(\mathbf{r}t_0)$. The new PDE Eq. (1.90) is of the same Sturm-Liouville type as Eq. (1.89), so we can calculate the unique solution $\omega^{(1)}(\mathbf{r})$ and hence $\partial_t v'(\mathbf{r}t)$ at $t = t_0$. We can take the second derivative of Eq. (1.89) and repeat the above procedure to determine $\partial_t^2 v'(\mathbf{r}t)$ at $t = t_0$.

In general, taking the kth time derivative of Eq. (1.89) gives

$$\boldsymbol{\nabla} \cdot \Big[n(\mathbf{r}t_0)\boldsymbol{\nabla}\omega^{(k)}(\mathbf{r})\Big] = Q^{(k)}(\mathbf{r}), \tag{1.94}$$

where the inhomogeneity $Q^{(k)}(\mathbf{r})$ is given by

$$Q^{(k)}(\mathbf{r}) = \zeta^{(k)}(\mathbf{r}) - \sum_{l=0}^{k-1} \binom{k}{l} \boldsymbol{\nabla} \cdot \left[n^{(k-l)}(\mathbf{r}) \boldsymbol{\nabla} \omega^{(l)}(\mathbf{r}) \right]. \tag{1.95}$$

The term $\zeta^{(k)}$ can be computed from multiple commutators of $\hat{q}(\mathbf{r})$ and $\hat{q}'(\mathbf{r})$ with the Hamiltonians $\hat{H}$ and $\hat{H}'$, respectively, and their time derivatives up to order $k-1$, sandwiched between the initial states $|\Psi_0\rangle$ and $|\Phi_0\rangle$. Notice that the inhomogeneity $Q^{(k)}(\mathbf{r})$ is completely determined by the density $n(\mathbf{r}t)$, the potential $v(\mathbf{r}t)$, the initial states $|\Psi_0\rangle$ and $|\Phi_0\rangle$, and the time derivatives $\partial_t^{(l)} v'(\mathbf{r}t)$ at $t = t_0$ up to order $k-1$. Eq. (1.94) therefore allows complete determination of $\partial_t^k v'(\mathbf{r}t)$ at $t = t_0$ for arbitrary k. We can use this to construct $v'(\mathbf{r}t)$ from its Taylor series,

$$v'(\mathbf{r}t) = \sum_{k=0}^{\infty} \frac{1}{k!} \partial_t^k v'(\mathbf{r}t) \bigg|_{t=t_0} (t - t_0)^k. \tag{1.96}$$

This expansion determines $v'(\mathbf{r}t)$ completely within the convergence radius. If the convergence radius is nonzero but finite, we can propagate $|\Phi_0\rangle$ to $|\Phi(t_1)\rangle$, where t_1 lies within the radius of convergence about t_0, and repeat the whole process above considering $|\Phi(t_1)\rangle$ as the initial state. This is an analytic continuation along the whole real time axis and a complete determination of $v'(\mathbf{r}t)$ at all times. We disregard the possibility of a convergence radius of zero since that implies nonanalyticity of $v'(\mathbf{r}t)$ and hence $n(\mathbf{r}t)$ and $v(\mathbf{r}t)$ at $t = t_0$, which we do not consider in this proof. Finally, note that in Eq. (1.94) the determination of the unique $\omega^{(k)}(\mathbf{r})$ depended on the boundary condition that $\omega(\mathbf{r}t)$ approach zero at infinity. In general, we could add an arbitrary time-dependent constant $C(t)$ to $v'(\mathbf{r}t)$, resulting in $\tilde{\omega}(\mathbf{r}t) = \omega(\mathbf{r}t) - C(t)$ that would still satisfy Eq. (1.94) for every k since

$$\boldsymbol{\nabla} \tilde{\omega}^{(k)}(\mathbf{r}) = \boldsymbol{\nabla} \omega^{(k)}(\mathbf{r}) - \boldsymbol{\nabla} C^{(k)}(t) = \boldsymbol{\nabla} \omega^{(k)}(\mathbf{r}). \tag{1.97}$$

This additional additive degree of freedom $C(t)$ in $v'(\mathbf{r}t)$ is not surprising since adding a uniform constant to the potential only affects the phase of the wavefunction, which does not affect the density. However, as will be seen in Section 2.3.1 where we extend this proof to imaginary time, a specific choice of $C(t)$ plays an important role in maintaining the unit norm of a wavefunction undergoing imaginary time evolution. We can now make the following statement: We specify a given density $n(\mathbf{r}t)$ obtained from a many-particle system with Hamiltonian $\hat{H}$ and initial state $|\Psi_0\rangle$. If one chooses an initial state $|\Phi_0\rangle$ of a many-particle system with two-particle interaction $\hat{W}'$ in such a way that it yields the correct initial density and initial time derivative of

the density, then, for this system, there is a unique external potential $v'(\mathbf{r}t)$ [determined up to a purely time-dependent function $C(t)$] that reproduces the given density $n(\mathbf{r}t)$. □

This theorem is rather remarkable. Note that if we are to choose the same particle-particle interaction in the primed system, i.e. $w' = w$, we recover the Runge-Gross theorem (but slightly less general since the density also has to be t-TE while in the Runge-Gross proof only the potential had to be t-TE). If we are to choose no particle-particle interactions, $w' = 0$, we are then referring to a noninteracting system, and the van Leeuwen theorem guarantees the existence of some noninteracting system which reproduces the time-dependent interacting density. This permits the use of a Kohn-Sham system in the time-dependent case.

Despite the success of the van Leeuwen theorem, there is still work to be done in this area. The restriction to t-TE densities can lead to some issues in real world systems. For example the density can frequently have cusps in space due to diverging Coulomb potentials, which can result in non-t-TE densities due to the action of the kinetic energy operator $\hat{T}$, a differential operator in space. To clarify the point, though using a contrived example, consider an electron held in place by a Coulombic nuclear potential. The initial density has a cusp at the location of the nucleus. If the nucleus were to suddenly disappear, by solving the time-dependent Schrödinger equation we can show that the cusp immediately becomes rounded, and the peak starts to dissipate as the electron no longer has a reason to be localized. However, when using a Taylor expansion in time, the cusp cannot disappear and the wavefunction remains stationary in time, which is the wrong behavior. A more physical example of a non-t-TE density would be the one associated with a nuclear fission event, where a single cusp has to break into two.

1.3.3 Time-Dependent Kohn-Sham Formalism

In light of the van Leeuwen theorem, we can simply propagate our Kohn-Sham system forward in time, using the proper single-particle potential guaranteed by the van Leeuwen theorem to reproduce the interacting system's density and its time dependence.

The exact time-dependent density can be calculated from a noninteracting system with N single-particle orbitals:

$$n(\mathbf{r},t) = \sum_{j=1}^{N} \left|\varphi_j(\mathbf{r},t)\right|^2. \tag{1.98}$$

The orbitals $\varphi_j(\mathbf{r},t)$ satisfy the time-dependent Kohn-Sham equation:

$$\mathrm{i}\frac{\partial}{\partial t}\varphi_j(\mathbf{r},t) = \left[-\frac{\nabla^2}{2} + v_s(\mathbf{r},t)\right]\varphi_j(\mathbf{r},t), \tag{1.99}$$

where the time-dependent effective potential is given by

$$\begin{aligned} v_s[n,\Psi_0,\Phi_0](\mathbf{r},t) = v(\mathbf{r},t) + v_{\mathrm{H}}(\mathbf{r},t) \\ + v_{\mathrm{xc}}[n,\Psi_0,\Phi_0](\mathbf{r},t). \end{aligned} \tag{1.100}$$

Like in the ground state theory, $v(\mathbf{r},t)$ is the time-dependent external potential, which we assume to have the form

$$v(\mathbf{r},t) = v_0(\mathbf{r}) + \theta(t-t_0)v_1(\mathbf{r},t). \tag{1.101}$$

The time-dependent Hartree potential,

$$v_{\mathrm{H}} = \int \mathrm{d}^3 r' \, \frac{n(\mathbf{r}',t)}{|\mathbf{r}-\mathbf{r}'|}, \tag{1.102}$$

depends on the instantaneous time-dependent density only (so it is classical and nonrelativistic). The time dependent xc potential v_{xc} formally has functional dependence on the density, the initial many-body state Ψ_0 of the exact interacting system, and the initial state of the Kohn-Sham system Φ_0.

1.3.4 Time-Dependent Exchange-Correlation Potential Approximations

Like the ground state theory, the approximation in the otherwise exact TDDFT comes in through the fact that we do not know the exchange-correlation potential. Formally, the time-dependent xc potential is a functional of the time-dependent density as well as the initial states, $v_{\mathrm{xc}}[n,\Psi_0,\Phi_0](\mathbf{r},t)$. Usually we are interested in systems that start off in the ground state, so thankfully we can use the Hohenberg-Kohn theorem to eliminate the dependence on the initial states Ψ_0 and Φ_0 which we know to be functionals of the density. This helps simplify things a bit, and we can write the xc potential as $v_{\mathrm{xc}}[n](\mathbf{r},t)$.

However, the complications do not end there. Technically the density dependence of the xc potential is nonlocal in space and in time: the xc potential at a space-time point $(\mathbf{r},t)$ depends on densities at all other points in space and at all previous times, $n(\mathbf{r}',t')$, where $t' \leq t$. This is a substantial increase in complexity over the already challenging nonlocal dependence of the xc potential in ground state DFT.

Due to the extraordinary complexity of the "memory" feature of the exchange-correlation potential, the most widely used approximation for the xc potential is the *adiabatic approximation*:

$$v_{\mathrm{xc}}^{\mathrm{A}}(\mathbf{r},t) = v_{\mathrm{xc}}^{\mathrm{gs}}[n_0](\mathbf{r})\Big|_{n_0(\mathbf{r})=n(\mathbf{r},t)}, \tag{1.103}$$

where $v_{\mathrm{xc}}^{\mathrm{gs}}$, the ground-state xc potential defined in Eq. (1.32), is evaluated at the instantaneous time-dependent density. This approximation becomes exact as the system varies slower and slower in time, as one would expect since this limit is just taking us to the stationary ground state. Through this adiabatic approximation, we eliminate all of the memory aspect of the xc potential.

Surprisingly, the adiabatic equation works quite well in many cases. Many time-dependent Kohn-Sham calculations use the adiabatic LDA (ALDA),

$$v_{\mathrm{xc}}^{\mathrm{ALDA}}(\mathbf{r},t) = v_{\mathrm{xc}}^{\mathrm{LDA}}(n(\mathbf{r},t)), \tag{1.104}$$

which is just the adiabatic approximation applied to the local density approximation. Other popular choices are adiabatic GGA functionals, where the adiabatic approximation is applied to common GGA functionals.

Very few applications to date have been carried out with nonadiabatic, explicitly memory-dependent xc functionals[79].

1.4 Linear Response in TDDFT

In the previous sections we have seen that it is possible to have a noninteracting system which reproduces the density of a given interacting system. Through Runge-Gross, we also have that all observables are functionals of the density, so in principle, we could now calculate anything we want. In practice, however, computing observables in DFT comes with its own challenges. Some observables that are directly obtained from the density, such as the dipole moment, are fairly straightforward to extract from the Kohn-Sham system. Other quantities of interest, such as photoelectron spectra and state-to-state transition probabilities, are difficult to calculate and call for their own research and levels of approximations.

Often, in many systems of practical interest, we do not have to worry about solving the full time-dependent Schrödinger or Kohn-Sham systems since we are only subjecting the system to a small perturbation. In such situations, the system does not deviate strongly from its initial state and it becomes sufficient to calculate the response to first order in the perturbation. This is referred to as *linear response theory*, whose goal is to directly calculate the change in a certain variable or observable to first order without

having to calculate the change in the wavefunction.

As an example, in most applications of spectroscopy, the response to a weak probe is used to determine the spectral properties of a system. In these cases, linear response is sufficient to calculate the quantities we are interested in.

1.4.1 Linear Response Review

Here we briefly review linear response in TDDFT[79]. Consider a quantum mechanical observable $\hat{\alpha}$, whose ground state expectation value is given by

$$\alpha_0 = \langle \Psi_0 | \hat{\alpha} | \Psi_0 \rangle , \tag{1.105}$$

where Ψ_0 is the ground state many-body wavefunction associated with the static Hamiltonian $\hat{H}_0$. Now, assume that the system is acted upon by a time-dependent perturbation

$$\hat{H}_1(t) = F(t)\hat{\beta}, \qquad t \geq t_0, \tag{1.106}$$

where $F(t)$ is an external field that couples to an observable $\hat{\beta}$ and which is switched on at a time t_0. This perturbation affects the wavefunction of the system, and thus the expectation value of the observable α, which now becomes time-dependent:

$$\alpha(t) = \langle \Psi(t) | \hat{\alpha} | \Psi(t) \rangle , \qquad t \geq t_0. \tag{1.107}$$

The difference between the time-dependent expectation value of $\hat{\alpha}$ and its initial static value $\alpha(t) - \alpha_0$, is called the *response* of $\hat{\alpha}$ to the perturbation. The response can be expanded in powers of the field $F(t)$:

$$\alpha(t) - \alpha_0 = \alpha_1(t) + \alpha_2(t) + \alpha_3(t) + \ldots , \tag{1.108}$$

where $\alpha_1(t)$ is the linear response, $\alpha_2(t)$ is the quadratic response, and so on.

Using the first-order approximation to the time evolution operator and the interaction picture for the operators $\hat{\alpha}$ and $\hat{\beta}$, we obtain the linear response as[8]

$$\alpha_1(t) = -\mathrm{i} \int_{t_0}^{t} \mathrm{d}t' \, F\left(t'\right) \left\langle \Psi_0 \middle| \left[\hat{\alpha}(t), \hat{\beta}(t') \right] \middle| \Psi_0 \right\rangle . \tag{1.109}$$

Since the initial-state Hamiltonian $\hat{H}_0$ is time-independent, we can replace the commutator $\left[\hat{\alpha}(t),\hat{\beta}(t')\right]$ with $\left[\hat{\alpha}(t-t'),\hat{\beta}\right]$. This is shown explicitly in Eq. 5.43 of Bruus and Flensberg[8], where the cyclic property of the trace in the expectation value is used to merge exponentials. We can now define the retarded response function

$$\chi_{\alpha\beta}(t-t') = -i\theta(t-t')\left\langle\Psi_0\middle|\left[\hat{\alpha}(t-t'),\hat{\beta}\right]\middle|\Psi_0\right\rangle. \tag{1.110}$$

The linear response $\alpha_1(t)$ is therefore given by

$$\alpha_1(t) = \int_{-\infty}^{\infty} \mathrm{d}t'\,\chi_{\alpha\beta}(t-t')F(t'). \tag{1.111}$$

The response function $\chi_{\alpha\beta}(t-t')$ only depends on properties of the system in the absence of the probe. We now proceed to the most important case in the context of TDDFT, the density-density response. Given that the electronic density is the most easily accessible observable in density functional theory, the density-density response is often desired.

1.4.2 The Density-Density Response Function

Suppose we start with a system of interacting particles in the ground state, and at $t = 0$ a perturbation is switched on. The total potential is thus given by

$$v(\mathbf{r},t) = v_0(\mathbf{r}) + \theta(t-t_0)v_1(\mathbf{r},t). \tag{1.112}$$

Again like in Eq. (1.108), we expand the response, which in this case is the density:

$$n(\mathbf{r},t) = n_0(\mathbf{r}) + n_1(\mathbf{r},t) + n_2(\mathbf{r},t) + \ldots, \tag{1.113}$$

where in linear response we are only concerned with the first order term $n_1(\mathbf{r},t)$. From Eq. (1.111), we can formally write the linear density response as

$$n_1(\mathbf{r},t) = \int_{-\infty}^{\infty} \mathrm{d}t' \int \mathrm{d}^3r'\,\chi(\mathbf{r},\mathbf{r}',t-t')v_1(\mathbf{r}',t'), \tag{1.114}$$

where

$$\chi\big(\mathbf{r},\mathbf{r}',t-t'\big) = -\mathrm{i}\theta(t-t')\,\langle\Psi_0|\,\big[\hat{n}\big(\mathbf{r},t-t'\big),\hat{n}\big(\mathbf{r}'\big)\big]\,|\Psi_0\rangle\,. \tag{1.115}$$

We are typically interested in the frequency-dependent response, which is given by a simple Fourier transform,

$$n_1(\mathbf{r},\omega) = \int \mathrm{d}^3r'\,\chi\big(\mathbf{r},\mathbf{r}',\omega\big)v_1\big(\mathbf{r}',\omega\big). \tag{1.116}$$

Furthermore, by inserting complete set of eigenstates $\sum_{n=1}^{\infty}|\Psi_n\rangle\langle\Psi_n| = \hat{1}$, and Fourier transforming the response function, we obtain the Lehmann representation for the density-density response function,

$$\chi\big(\mathbf{r},\mathbf{r}',\omega\big) = \sum_{n=1}^{\infty}\Big\{\frac{\langle\Psi_0|\hat{n}(\mathbf{r})|\Psi_n\rangle\,\langle\Psi_n|\hat{n}(\mathbf{r}')|\Psi_0\rangle}{\omega-\Omega_n+\mathrm{i}\eta} - \frac{\langle\Psi_0|\hat{n}(\mathbf{r}')|\Psi_n\rangle\,\langle\Psi_n|\hat{n}(\mathbf{r})|\Psi_0\rangle}{\omega+\Omega_n+\mathrm{i}\eta}\Big\}, \tag{1.117}$$

where the limit $\eta\to 0^+$ is to be applied to pick the appropriate contour upon integration, and where we define the energy difference

$$\Omega_n = E_n - E_0. \tag{1.118}$$

Note that in this representation, we see explicitly that the response function has poles at the exact excitation energies of the system. If we are to apply a perturbation $v_1(\mathbf{r},\omega)$ with a frequency that matches one of the excitation energies, we expect to see a very large response.

If we knew the response function χ of the many-body system, calculating the density response would just be a matter of performing the integral in Eq. (1.116). Then, from the density response we could obtain spectroscopic observables.

To give an example, consider a monochromatic dipole field along the z direction:

$$v_1(\mathbf{r},t) = \mathcal{E}z\sin(\omega t). \tag{1.119}$$

From the density response $n_1(\mathbf{r},\omega)$ that follows, we can calculate the dynamic dipole polarizability,

$$\alpha(\omega) = -\frac{2}{\mathcal{E}}\int \mathrm{d}^3r\,z n_1(\mathbf{r},\omega), \tag{1.120}$$

and the photoabsorption cross section $\sigma(\omega)$ is then given by

$$\sigma(\omega) = \frac{4\pi\omega}{c}\,\mathrm{Im}\,[\alpha(\omega)]. \tag{1.121}$$

1.4.3 KOHN-SHAM DENSITY-DENSITY RESPONSE

In general it is very difficult to calculate the response function χ of an interacting system, so it would not be a very promising approach to try to calculate the linear response directly from an interacting many-body system. Instead, we will again make use of the Kohn-Sham system.

In TDDFT, the linear response of a general interacting many-body system can be calculated, in principle exactly, as the response of the noninteracting Kohn-Sham system due to an *effective* perturbation:

$$n_1(\mathbf{r},t) = \int \mathrm{d}t' \int \mathrm{d}^3r'\, \chi_s\big(\mathbf{r},\mathbf{r}',t-t'\big) v_{1s}\big(\mathbf{r}',t'\big). \tag{1.122}$$

Here $\chi_s(\mathbf{r},\mathbf{r}',t-t')$ is the density-density response function of the Kohn-Sham system, and $v_{1s}(\mathbf{r},t)$ is the effective perturbation that we will discuss shortly.

First, note that the density-density response function can be directly obtained from the expression in the Lehmann representation that we derived earlier, Eq. (1.117):

$$\chi_s\big(\mathbf{r},\mathbf{r}',\omega\big) = \sum_{j,k=1}^{\infty} (f_k - f_j)\frac{\varphi_j(\mathbf{r})\varphi_k^*(\mathbf{r})\varphi_j^*(\mathbf{r}')\varphi_k(\mathbf{r}')}{\omega - \omega_{jk} + \mathrm{i}\eta}, \tag{1.123}$$

where f_j and f_k are the occupation numbers referring to the configuration of the Kohn-Sham ground state (1 for occupied, 0 for unoccupied), and ω_{jk} are defined as

$$\omega_{jk} = \varepsilon_j - \varepsilon_k. \tag{1.124}$$

At this notice that the Kohn-Sham response function has poles at the excitation energies of the Kohn-Sham system, which in Section 1.2.3 we had stated have no supported physical meaning. Furthermore, the numerators, which give the strengths of the poles, are related to the optical absorption intensities of the KS system and not those of the true system. This is disconcerting, since we are interested in the properties of the real system. However, this is where the *effective* perturbation comes into play. Since the Kohn-Sham interaction contains both components from the actual many-body external potential as well as the effective single-particle interactions that describe the many-body interactions, correlations, and exchanges, it makes

sense that we could not simply apply the external perturbation in Eq. (1.122). A variation in the external potential will affect the electronic configuration, which would then affect the many-body interactions encoded in the Kohn-Sham single-particle potential v_s.

So, expanding the definition of the Kohn-Sham effective single-particle potential, Eq. (1.100), up to first order in the density response $n_1(\mathbf{r},t)$, we obtain the following first-order perturbation of the Kohn-Sham potential:

$$v_{s1}(\mathbf{r},t) = v_1(\mathbf{r},t) + \int \mathrm{d}^3 r' \, \frac{n_1(\mathbf{r}',t)}{|\mathbf{r}-\mathbf{r}'|} + \int \mathrm{d}t' \int \mathrm{d}^3 r' \, f_{\mathrm{xc}}\big(\mathbf{r},t,\mathbf{r}',t'\big) n_1\big(\mathbf{r}',t'\big). \tag{1.125}$$

Here, f_{xc} is the so-called *xc kernel*, the functional derivative of the xc potential with respect to the density, evaluated at the ground state density:

$$f_{\mathrm{xc}}\big(\mathbf{r},t,\mathbf{r}',t'\big) = \left.\frac{\delta v_{\mathrm{xc}}[n](\mathbf{r},t)}{\delta n(\mathbf{r}',t')}\right|_{n_0(\mathbf{r})}. \tag{1.126}$$

As expected, the effective potential perturbation (1.125) contains a part of the actual potential perturbation, v_1, as well as a term from the Hartree potential variation, as well as the variation in the xc potential.

Plugging in Eq. (1.125) into Eq. (1.122) results in an equation that features $n_1(\mathbf{r},t)$ on both sides. Solving the equation self-consistently for the density response ends up giving the right answer. In some sense, the self-consistent nature "cancels out" the wrong poles and restores the correct poles of the many-body system.

We can also use these equations to solve for the actual many-body system's response function in terms of the KS response function by setting Eq. (1.122), which contains a reference to the KS response function, equal to Eq. (1.114), which contains the true many-body response function, giving:

$$\chi\big(\mathbf{r},t;\mathbf{r}',t'\big) = \chi_s\big(\mathbf{r},t;\mathbf{r}',t'\big) + \int \mathrm{d}t_1 \, \mathrm{d}^3 r_1 \, \mathrm{d}t_2 \, \mathrm{d}^3 r_2 \, \chi_s(\mathbf{r},t;\mathbf{r_1},t_1) \left[\frac{\delta(t_1-t_2)}{|\mathbf{r}_1-\mathbf{r}_2|} + f_{\mathrm{xc}}(\mathbf{r}_1,t_1;\mathbf{r}_2,t_2)\right] \chi\big(\mathbf{r}_2,t_2;\mathbf{r}',t'\big). \tag{1.127}$$

Notice the resemblence to a Dyson equation,

$$G = G_0 + G_0 \sum G, \tag{1.128}$$

where χ_s plays the role of G_0. This gives more credence to the "correcting" of the poles of the density-density response function, since we recall that the Dyson equation shifts the poles of the bare propagator G_0 to yield the dressed propagator G.

There are many approximations to the xc kernel required in Eq. (1.125), the simplest of which is to simply set the xc kernel to zero:

$$f_{\mathrm{xc}}^{\mathrm{RPA}}\left(\mathbf{r},t;\mathbf{r}',t'\right)=0. \tag{1.129}$$

this seemingly trivial kernel is called the random phase approximation (RPA), which comes from many-body theory where one sums up all the "bubble"-type diagrams. This makes sense since such a substitution would only leave the Hartree correction to the effective KS perturbation (1.125). The form becomes similar to time-dependent Hartree, but one has to remember that TDDFT RPA is fundamentally different since it is using the Kohn-Sham system instead.

The xc kernel inherits the difficulties of the exchange-correlation potential since it is just its functional derivative with respect to density. Just like how the vast majority of xc potentials in use are adiabatic, so are practical xc kernels f_{xc}. Such kernels do not have any explicit time dependence and only depend on the instantaneous density. The approximation is often carried further by assuming that the kernel is also local in space. An important example is the ALDA xc kernel,

$$f_{\mathrm{xc}}^{\mathrm{A}}\left(\mathbf{r},\mathbf{r}'\right)=\left.\frac{\mathrm{d}^2 e_{\mathrm{xc}}^{\mathrm{h}}(\bar{n})}{\mathrm{d}\bar{n}^2}\right|_{\bar{n}=n_0(\mathbf{r})}\delta\left(\mathbf{r}-\mathbf{r}'\right). \tag{1.130}$$

1.5 Conclusion

DFT is a powerful formalism for solving many-body quantum problems. By simplifying the task of solving the full Schrödinger equation to just solving for a density, many complex systems become accessible given the present limits in computational power. Time-dependent DFT is shaping up to play a crucial role in biology for studying the links between structure and functionality due to its favorable scaling with system size, a feature that already allows systems of thousands of atoms to be simulated. Building off of the success of ground state density functional theory for the study of materials and their optical and electronic properties, TDDFT enables the investigation of fundamentally time-dependent systems which have been challenging to study in the past. This class of systems contains the rich world of chemical reactions, photovoltaic processes of biological materials, optoelectronic devices, nonequilibrium thermodynamics, and much more.

There is much room for improvement, however. Despite being formally exact, any application of TDDFT still has two major approximations. First, since it is not known how to obtain exact exchange-correlation energies, potentials, or kernels, simplifying schemes are used, oftentimes rendering the xc potential local in

space and time. Many interesting systems in physics, such as excitations involving charge separation, do not yield correct results when the xc potential lacks the proper long-range behavior. Second, TDDFT systems have to be solved numerically, often with self-consistency loops that may be tricky to converge, or that sometimes converge to a state that was not desired. That said, TDDFT has already proven successful in many systems of physical interest, ranging from the study of interactions of molecules with light to the improvement of catalysts in industrial applications.

In the following chapter, we extend TDDFT to imaginary time, enabling the use of some convenient computational methods in many-body systems. In particular, imaginary-time TDDFT can be used as a robust and reliable method for solving the Kohn-Sham equations in ground-state DFT, a task that can prove challenging in some large systems, or those with metallic character. We also present an extension of the van Leeuwen theorem in imaginary time, providing theoretical backing for using the Kohn-Sham system for certain topics in quantum statistical mechanics where a Wick rotation to imaginary time is employed.

2

Imaginary-Time Time-Dependent Density Functional Theory

Apart from minor modifications and elaborations, this chapter originally appeared in the following publication:

> **Cedric Flamant**, Grigory Kolesov, Efstratios Manousakis, and Efthimios Kaxiras. "Imaginary-Time Time-Dependent Density Functional Theory and Its Application for Robust Convergence of Electronic States." *J. Chem. Theory Comput.* **15**, 11, 6036-6045 (2019).

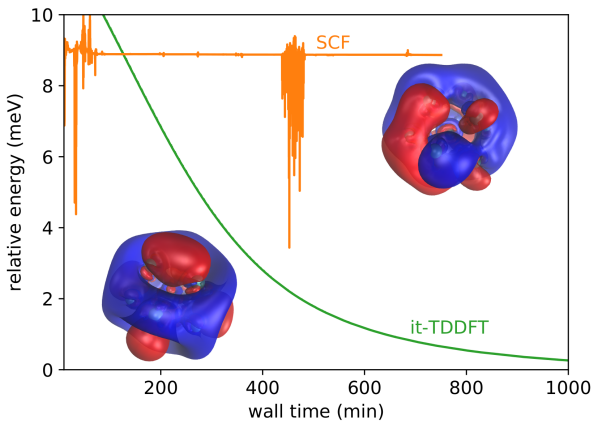

Figure 2.1: Visual abstract submitted to *J. Chem. Theory Comput.*. The smooth, monotonic convergence of a Cu_{13} nanocluster electronic state using imaginary-time time-dependent density functional theory is compared to the conventional approach, SCF, which does not converge to the desired lowest-energy state. Magnetization density isosurfaces are depicted for both states.

Abstract

Reliable and robust convergence to the electronic ground state within density functional theory (DFT) Kohn-Sham (KS) calculations remains a thorny issue in many systems of interest. In such cases, charge sloshing can delay or completely hinder the convergence. Here, we use an approach based on transforming

the time-dependent DFT equations to imaginary time, followed by imaginary-time evolution, as a reliable alternative to the self-consistent field (SCF) procedure for determining the KS ground state. We discuss the theoretical and technical aspects of this approach and show that the KS ground state should be expected to be the long-imaginary-time output of the evolution, independent of the exchange-correlation functional or the level of theory used to simulate the system. By maintaining self-consistency between the single-particle wavefunctions (orbitals) and the electronic density throughout the determination of the stationary state, our method avoids the typical difficulties encountered in SCF. To demonstrate dependability of our approach, we apply it to selected systems which struggle to converge with SCF schemes. In addition, through the van Leeuwen theorem, we affirm the physical meaningfulness of imaginary time TDDFT, justifying its use in certain topics of statistical mechanics such as in computing imaginary time path integrals.

2.1 Introduction

Density functional theory (DFT) is a widely used approach enabling *ab initio* calculations of electronic and material properties. Unlike direct approaches to studying quantum systems through the Schrödinger equation where the wavefunction is the central object, DFT uses the electron density $n(\mathbf{r})$ as the fundamental physical quantity. In principle, through the Hohenberg-Kohn theorem[25], the ground state $n(\mathbf{r})$ of a system uniquely determines all of its observables. It is standard practice to use the Kohn-Sham (KS) system[31] of non-interacting fermions as a shortcut to obtaining the ground state density, employing specially formulated potentials[9,56] that are functionals of $n(\mathbf{r})$ to approximate the electron-electron interactions.

There are many techniques to find the ground state of the KS equations, including methods which: (a) aim at direct determination of the minimum of the KS total energy functional[71,85,86]; and (b) use iterative methods based on diagonalization of the KS Hamiltonian in conjunction with iterative improvements of the ground state charge density through mixing. The present work is distinct from both these approaches. We focus on comparing our method to type (b) approaches.

For a system with N electrons, the lowest N eigenstates to the KS equations determine $n(\mathbf{r})$, which itself appears in the KS equations through an effective single-particle potential. In general, finding the set of N eigenstates that satisfy the KS equations involves an iterative process known as self-consistent field (SCF) iterations that produce successively better approximations to the solution. In its simplest conceptualization the iterative approach involves solving the eigenvalue problem for an initial density distribution, then using the resulting eigenstates to produce the next approximation to the density. When this approach is iterated, except for the simplest systems, it rarely converges to a self-consistent solution. In order to stabilize the SCF

loops and improve the convergence rate, various mixing schemes are typically employed. These schemes take advantage of the information contained in multiple previous trial densities to select the next one. A popular mixing scheme is direct inversion of the iterative subspace (DIIS), also known as Pulay mixing[58,59].

When SCF schemes require many iterations to reach an acceptable solution, or fail to converge, the choices are to change the mixing scheme or its parameters, start with a different density, or fractionally occupy states[49,60] which some methods implement by introducing a fictitious electronic temperature (Fermi smearing[48,84]). Other methods like level-shifting[66] can also fix oscillating or divergent SCF steps. If these fail, one can resort to computationally-intensive direct minimization methods[71,85,86] to find a solution. Convergence difficulties for SCF usually arise in systems with large unit cells and in metallic systems[2], or when an excited state is desired. The small differences in eigenenergies of the KS orbitals, as well as the presence of many states near the Fermi level, can cause very different eigenstates to be occupied from step to step. This can lead to large variations in the density, causing the phenomenon known as charge sloshing[36] where a fluctuating charge density from step to step is observed with insufficient attenuation to reach convergence.

In the present paper we transform the time-dependent KS (TDKS) equations of time-dependent density functional theory (TDDFT)[57,64,83] to imaginary time[33]. We use these equations to propagate an initial state to very long imaginary time, refining it down to the KS state corresponding to its lowest energy component. The idea of using imaginary-time propagation (ITP) to find eigenstates is well-known, and it is frequently used to find ground state solutions to the Schrödinger equation describing single-particle systems with a fixed potential[3,40]. Imaginary time-steps have also been used to find self-consistent solutions to the Hartree-Fock equations[12] and for nuclear energy density functional calculations[65]. It has also been employed in a DFT context as an alternative to the diagonalization step to find the orbitals for a fixed electronic density[1,23]. However, imaginary-time evolution has yet to be examined as a stand-alone substitute to iterative density updating in solving the KS equations. In the present method both the density and wavefunction evolve together towards the ground state according to the imaginary time TDKS equations, remaining consistent with each other throughout the calculation. We discuss the theoretical foundation of the imaginary-time evolution of the KS system, a procedure which is non-unitary, requiring re-orthonormalization of the states at each imaginary time-step. We show that the proof provided by van Leeuwen[83] for TDDFT can be extended to imaginary-time TDDFT (it-TDDFT), affirming in principle that the density of a KS system will evolve in imaginary time in the same manner as the true many-body interacting system. The imaginary-time propagation method in DFT has attractive theoretical and practical benefits when applied to systems that are challenging to study using standard methods of solving the KS

equations, as we demonstrate on model systems.

We benchmark our approach by applying it to the benzene molecule and show that it converges to the same ground state energy as other SCF-based methods. Next, we apply our method to systems with known difficulties in achieving convergence. We chose to examine a copper nanocluster Cu_{13} with fixed magnetization and a spin-unpolarized Ru_{55} nanocluster. We show that self-consistent solutions are hard to realize in both systems using the most popular standard approach, SCF with Pulay mixing. In general, we find that while requiring more computation, our method is more dependable and more autonomous compared to SCF. It provides a good alternative to existing methodologies when the latter fail to converge in challenging systems, or if a user wishes to find an unfamiliar system's ground state with minimal intervention; this can be particularly useful when computations are carried out in an automated fashion on large clusters of processors.

2.2 Methodology

2.2.1 Imaginary-Time Propagation

First, let us take the Hamiltonian $\hat{H}$ to be time-independent. Under the substitution $t \to -\mathrm{i}\tau$, where τ is real, the time evolution operator transforms from $e^{-\mathrm{i}t\hat{H}}$ to $e^{-\tau\hat{H}}$. When $|\Phi_i\rangle$ is an eigenstate of $\hat{H}$,

$$|\Phi_i(\tau)\rangle = e^{-\tau\hat{H}}|\Phi_i\rangle = e^{-\tau E_i}|\Phi_i\rangle\,. \tag{2.1}$$

For an arbitrary initial wavefunction $|\Psi(0)\rangle$, imaginary-time propagation amounts to

$$|\tilde{\Psi}(\tau)\rangle = \sum_{i=0}^{\infty} A_i(0)e^{-\tau E_i}|\Phi_i\rangle\,, \tag{2.2}$$

where $A_i(0)$ is the amplitude of the eigenstate component initially present. As imaginary time goes to infinity, $\tau \to \infty$, the eigenstate $\left|\Phi_j\right\rangle$ corresponding to the lowest energy eigenvalue with $A_j(0) \neq 0$ will dominate. We can choose to keep the state $|\Psi(\tau)\rangle$ normalized by dividing by the norm $\Omega(\tau) \equiv \sqrt{\langle\tilde{\Psi}(\tau)|\tilde{\Psi}(\tau)\rangle} = \sqrt{\sum_{i=0}^{\infty}|A_i(0)|^2 e^{-2\tau E_i}}$,

$$|\Psi(\tau)\rangle = \sum_{i=0}^{\infty}\frac{A_i(0)e^{-\tau E_i}}{\Omega(\tau)}|\Phi_i\rangle\,, \tag{2.3}$$

which then yields $\lim_{\tau\to\infty}|\Psi(\tau)\rangle = |\Phi_j\rangle$. Since an arbitrary initial state generated by randomizing the coefficients in some basis is likely to have a nonzero ground state component, ITP is often used to find ground state wavefunctions and energies. While an initial state with a strong ground state component will converge earlier than an arbitrary starting state, ITP rapidly eliminates all but the lowest energy components so the decrease in the total computational time from a well-chosen starting state is modest and not worth undue effort to attain.

2.2.2 Implementation within the Kohn-Sham Formalism

In TDDFT, starting from an initial state, the KS system obeys the equations of motion (in atomic units):

$$\mathrm{i}\frac{\partial}{\partial t}\phi_j(\mathbf{r},t) = \hat{H}_{\mathrm{KS}}[n(\mathbf{r},t)]\phi_j(\mathbf{r},t), \tag{2.4a}$$

$$\hat{H}_{\mathrm{KS}}[n(\mathbf{r},t)] \equiv \left(-\frac{\nabla^2}{2} + v_s[n(\mathbf{r},t)]\right), \tag{2.4b}$$

with time-dependent effective potential

$$v_s[n(\mathbf{r},t)] = v(\mathbf{r}) + v_{\mathrm{H}}[n(\mathbf{r},t)] + v_{\mathrm{xc}}[n(\mathbf{r},t)]. \tag{2.5}$$

In these expressions, $v(\mathbf{r})$ is the external potential and

$$v_{\mathrm{H}}[n(\mathbf{r},t)] = \int \mathrm{d}\mathbf{r}'\,\frac{n(\mathbf{r}',t)}{|\mathbf{r}-\mathbf{r}'|}, \tag{2.6}$$

$$v_{\mathrm{xc}}[n(\mathbf{r},t)] = \frac{\delta E_{\mathrm{xc}}[n(\mathbf{r},t)]}{\delta n(\mathbf{r},t)}, \tag{2.7}$$

$$n(\mathbf{r},t) = \sum_{j=1}^{N} \left|\phi_j(\mathbf{r},t)\right|^2. \tag{2.8}$$

The Kohn-Sham time-evolution can be reformulated in terms of a time-propagator which acts on orbitals and is given by

$$\left|\phi_j(t)\right\rangle = \hat{U}(t,t_0)\left|\phi_j(t_0)\right\rangle, \tag{2.9}$$

$$\hat{U}(t,t_0) = \hat{\mathcal{T}}\exp\left(-\mathrm{i}\int_{t_0}^{t}\hat{H}_{\mathrm{KS}}[n(\mathbf{r},t')]\,\mathrm{d}t'\right), \tag{2.10}$$

where $\hat{\mathcal{T}}$ is the time-ordering operator. In imaginary time, applying the substitution $t \to -\mathrm{i}\tau$ results in

$$\left|\phi_j(\tau)\right\rangle = \hat{\mathcal{U}}(\tau,\tau_0)\left|\phi_j(\tau_0)\right\rangle, \tag{2.11}$$

$$\hat{\mathcal{U}}(\tau,\tau_0) = \hat{\mathcal{T}}_\tau \exp\left(-\int_{\tau_0}^{\tau} \hat{H}_{\mathrm{KS}}[n(\mathbf{r},\tau')]\,\mathrm{d}\tau'\right), \tag{2.12}$$

where $\hat{\mathcal{T}}_\tau$ now time-orders in imaginary time. Note that the imaginary-time propagator is not unitary.

Employing the same numerical scheme used for real time propagation of KS states on an atomic basis[32], we evolve in imaginary-time the orbitals using finite time-steps $\Delta\tau$ and we approximate the instantaneous imaginary-time propagator with the second-order Magnus expansion:

$$\hat{\mathcal{U}}(\tau+\Delta\tau,\tau) \approx \exp\left[-\Delta\tau \hat{H}_{\mathrm{KS}}\left(\tau+\frac{\Delta\tau}{2}\right)\right], \tag{2.13}$$

$$\hat{H}_{\mathrm{KS}}(\tau) \equiv \hat{H}_{\mathrm{KS}}[n(\mathbf{r},\tau)]. \tag{2.14}$$

The Hamiltonian at the midpoint is approximated as the average of the Hamiltonians at τ_i and τ_{i+1}, $\hat{H}_{\mathrm{KS}}\left(\tau_i+\frac{\Delta\tau}{2}\right) \approx \frac{1}{2}\left[\hat{H}_{\mathrm{KS}}(\tau_i)+\hat{H}_{\mathrm{KS}}(\tau_{i+1})\right]$. Each step is iterated to self-consistency in order to make use of the Hamiltonian at τ_{i+1}. We use the Padé rational polynomial approximation of arbitrary degree to obtain the general matrix exponential. Further details of the numerical propagation can be found in our earlier work[32], which describes TDAP-2.0, a TDDFT code we used, built on top of SIESTA[69], a DFT package which uses strictly localized basis sets. While the midpoint Hamiltonian greatly aids stability and energy conservation in real time propagation, in practice we have found that for imaginary-time propagation we can just use the first step in the iterative procedure, which simply applies the approximation $\hat{H}_{\mathrm{KS}}\left(\tau_i+\frac{\Delta\tau}{2}\right) \approx \hat{H}_{\mathrm{KS}}(\tau_i)$. This explicit propagation is faster since the Hamiltonian only needs to be evaluated once per propagation step, and the effect on the size of the maximum stable time-step appears negligible compared to the implicit method using the midpoint Hamiltonian. This is expected since imaginary-time propagation is inherently more stable than the real-time propagation the TDAP-2.0 code was originally designed to solve. The most time-consuming process per step in our implementation is the matrix exponentiation. If the accuracy of the trajectory through imaginary time is not the primary concern of a calculation, lower-quality approximations to the matrix exponential can be used.

Because the imaginary-time propagator is not unitary, the orbitals lose their normalization and generally cease to be orthogonal. The simple expression for density in Eq. (2.8) becomes more complicated if the orbitals ϕ_j are non-orthonormal. It is convenient to reorthonormalize the orbitals at each time step. The details of how the orthogonalization is achieved do not affect the physics, as we show in Section 2.3.2. We

use the modified Gram-Schmidt algorithm to orthonormalize the states.

While we employ a localized atomic basis for our calculations, the method we propose is independent of the basis used to represent the Kohn-Sham orbitals, and can easily be implemented in other popular bases, like plane waves or Gaussians.

2.3 Theoretical Considerations

2.3.1 van Leeuwen Theorem in Imaginary Time

The van Leeuwen theorem states that a time-dependent particle density $n(\mathbf{r},t)$ belonging to a many-particle system with two-particle interaction $\hat{W}$ can always be reproduced by a unique (up to an additive purely time-dependent constant) external potential $v'(\mathbf{r},t)$ in another many-particle system that uses a different two-particle interaction $\hat{W}'$, under the mild restriction that the density has to be analytic in time[83]. If we choose the two-particle interaction in this other system to be $\hat{W}'=0$, the theorem guarantees the existence of the effective potential $v_s(\mathbf{r},t)$ for a Kohn-Sham system that reproduces the same time-dependent density as the interacting system of interest. Here we point out the modifications to the original theorem in order to make it compatible with imaginary-time evolution.

A complex t value does not pose any problems with the operations performed in the original proof, where t appears in some time derivatives but otherwise is treated as a parameter. We add time-dependent uniform potentials $\lambda(t)$ and $\lambda'(t)$ to the unprimed and primed Hamiltonians to conserve the norm of the wavefunctions. The origin of these terms will be discussed in the next section. The Hamiltonian $\hat{H}$ of a finite many-particle system is then given by

$$\hat{H}(t)=\hat{T}+\hat{V}(t)+\hat{W}+\lambda(t), \tag{2.15}$$

expressed in terms of creation and annihilation operators

$$\hat{T}=-\frac{1}{2}\int \mathrm{d}^3\mathbf{r}\,\hat{\psi}^\dagger(\mathbf{r})\nabla^2\hat{\psi}(\mathbf{r}), \tag{2.16a}$$

$$\hat{V}(t)=\int \mathrm{d}^3\mathbf{r}\,v(\mathbf{r},t)\,\hat{\psi}^\dagger(\mathbf{r})\hat{\psi}(\mathbf{r}), \tag{2.16b}$$

$$\hat{W}=\int \mathrm{d}^3\mathbf{r}\,\mathrm{d}^3\mathbf{r}'\,w\big(|\mathbf{r}-\mathbf{r}'|\big)\,\hat{\psi}^\dagger(\mathbf{r})\hat{\psi}^\dagger(\mathbf{r}')\hat{\psi}(\mathbf{r}')\hat{\psi}(\mathbf{r}). \tag{2.16c}$$

Since $\lambda(t)$ commutes with everything, it does not affect any of the commutators involving $\hat{H}(t)$ in the various Heisenberg equations of motion underpinning the proof of the van Leeuwen theorem. There is only

one detail to note, regarding the freedom to add an arbitrary $C(t)$ to the potential of the primed system, $v'(\mathbf{r},t)$, in the original proof. From Eq. (2.16b) a time-dependent constant in the potential modifies the Hamiltonian by an additional term $C(t)\hat{N}$, where $\hat{N}$ is the number operator. For the systems of interest, the number of particles N is fixed so $C(t)N$ is a time-dependent uniform potential like $\lambda'(t)$, which means that a norm-conserving $\lambda'(t)$ will cancel any effect from the choice of $C(t)$. Thus, with $\lambda(t)$ and $\lambda'(t)$ chosen to ensure that the norm of states in both the unprimed and primed systems is held at unity, the van Leeuwen theorem holds in imaginary time. This is a useful result since it allows us to think about imaginary-time propagation in the Kohn-Sham system in terms of what it does in the real system, allowing the Wick-rotation connections from quantum mechanics to statistical mechanics to be employed. For example, it justifies the use of the Kohn-Sham system as a stand-in for the interacting system in our calculations performed for imaginary time path integrals[33].

2.3.2 Maintaining Orthonormalization

Orthonormalization of the orbitals is equivalent to adding a purely time-dependent function $\lambda(t)$ to the many-body Hamiltonian. This takes care of holding the wavefunction normalized, both in the interacting and Kohn-Sham systems, as well as accounting for the orthogonalization step we use in the Kohn-Sham state propagation.

We first consider the interacting system. In real time propagation, the choice of $\lambda(t)$ does not affect the dynamics of density since this spatially-constant offset in energy only results in changing the phase of the wavefunction:

$$|\Psi(t)\rangle = \hat{U}(t,t_0)\,|\Psi(t_0)\rangle\,,$$

$$\hat{U}(t,t_0) = \hat{\mathcal{T}}\exp\left(-\mathrm{i}\int_{t_0}^{t}\hat{H}(t')+\lambda(t')\,\mathrm{d}t'\right) = U_\lambda(t,t_0)\;\hat{\mathcal{T}}\;\hat{U}_{\hat{H}}(t,t_0), \tag{2.17a}$$

$$U_\lambda(t,t_0) \equiv \exp\left(-\mathrm{i}\int_{t_0}^{t}\lambda(t')\,\mathrm{d}t'\right), \tag{2.17b}$$

$$\hat{U}_{\hat{H}}(t,t_0) \equiv \exp\left(-\mathrm{i}\int_{t_0}^{t}\hat{H}(t')\,\mathrm{d}t'\right). \tag{2.17c}$$

In imaginary-time propagation, $\lambda(\tau)$ modifies the imaginary-time propagator $\hat{\mathcal{U}}(\tau,\tau_0)$ by a time dependent

magnitude,

$$\hat{\mathcal{U}}(\tau,\tau_0) = \mathcal{U}_\lambda(\tau,\tau_0)\,\hat{\mathcal{T}}_\tau\,\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0), \tag{2.18a}$$

$$\mathcal{U}_\lambda(\tau,\tau_0) \equiv \exp\left(-\int_{\tau_0}^{\tau}\lambda(\tau')\,\mathrm{d}\tau'\right), \tag{2.18b}$$

$$\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0) \equiv \exp\left(-\int_{\tau_0}^{\tau}\hat{H}(\tau')\,\mathrm{d}\tau'\right). \tag{2.18c}$$

If $\lambda(\tau)$ is arbitrary, the norm of the wavefunction will change in time, incorrectly scaling the expectation values of observables like density and energy. The norm of the wavefunction can be held fixed by choosing $\lambda(\tau)$ to counteract the norm-altering effect of $\hat{\mathcal{T}}_\tau \exp\left(-\int_{\tau_0}^{\tau}\hat{H}(\tau')\,\mathrm{d}\tau'\right)$ when it acts on $|\Psi(\tau_0)\rangle$. Note that such a $\lambda(\tau)$ will also depend on the starting state. For example, in the time-independent Hamiltonian case presented in Section 2.2.1, from Eq. (2.3)

$$\mathcal{U}_\lambda(\tau,\tau_0) = \exp\left(-\int_{\tau_0}^{\tau}\lambda(\tau')\,\mathrm{d}\tau'\right) = \left[\sum_{j=0}^{\infty}\left|A_j(0)\right|^2 e^{-2\tau E_j}\right]^{-1/2}. \tag{2.19}$$

The equation can be rearranged and differentiated to reveal an interpretation of $\lambda(\tau)$,

$$\begin{aligned}
\frac{\partial}{\partial\tau}\int_{\tau_0}^{\tau}\lambda(\tau')\,\mathrm{d}\tau' &= \frac{1}{2}\ln\left[\sum_{j=0}^{\infty}\left|A_j(0)\right|^2 e^{-2\tau E_j}\right] \\
\lambda(\tau) = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}\tau}\ln\left[\sum_{j=0}^{\infty}\left|A_j(0)\right|^2 e^{-2\tau E_j}\right] &= -\frac{\sum_{j=0}^{\infty}E_j\left|A_j(0)\right|^2 e^{-2\tau E_j}}{\sum_{j=0}^{\infty}\left|A_j(0)\right|^2 e^{-2\tau E_j}} = -\langle E(\tau)\rangle,
\end{aligned} \tag{2.20}$$

that is, to keep the wavefunction normalized, $\lambda(\tau)$ is such that the energies of the Hamiltonian are measured relative to $\langle E(\tau)\rangle$. This result holds more generally for time-dependent Hamiltonians as well, which can be shown by using $\hat{\mathcal{U}}(\tau,\tau_0)$ from Eq. (2.18a) and differentiating the norm-conserving equation $1 = \langle\Psi(\tau)|\Psi(\tau)\rangle = \langle\Psi(\tau_0)|\hat{\mathcal{U}}^\dagger(\tau,\tau_0)\hat{\mathcal{U}}(\tau,\tau_0)\,|\Psi(\tau_0)\rangle$ to solve for $\lambda(\tau)$:

$$\begin{aligned}
0 &= \frac{\partial}{\partial\tau}\left[\exp\left(-2\int_{\tau_0}^{\tau}\lambda(\tau')\,\mathrm{d}\tau'\right)\langle\Psi(\tau_0)|\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)^\dagger\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)|\Psi(\tau_0)\rangle\right] \\
&= -2\lambda(\tau)\mathcal{U}_\lambda(\tau,\tau_0)\left\langle\tilde{\Psi}(\tau)\middle|\tilde{\Psi}(\tau)\right\rangle + \mathcal{U}_\lambda(\tau,\tau_0)\,\langle\Psi(\tau_0)|2\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)^\dagger\hat{H}(\tau)\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)|\Psi(\tau_0)\rangle \\
\lambda(\tau) &= -\frac{\left\langle\tilde{\Psi}(\tau)\middle|\hat{H}(\tau)\middle|\tilde{\Psi}(\tau)\right\rangle}{\left\langle\tilde{\Psi}(\tau)\middle|\tilde{\Psi}(\tau)\right\rangle} = -\langle E(\tau)\rangle,
\end{aligned} \tag{2.21}$$

where we use the unnormalized wavefunction $\left|\tilde{\Psi}(\tau)\right\rangle = \left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)|\Psi(\tau_0)\rangle$, and the fact that $\partial/\partial\tau\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right) = \hat{H}(\tau)\left(\hat{\mathcal{T}}_\tau\hat{\mathcal{U}}_{\hat{H}}(\tau,\tau_0)\right)$ since τ is the latest time so $\hat{H}(\tau)$ can be pulled out in front of the

time-ordering operator. We will assume that a norm-conserving $\lambda(\tau)$ is used in the interacting system so that the system always remains normalized.

In the Kohn-Sham system the propagator is given by

$$\hat{\mathcal{U}}(\tau,\tau_0) = \mathcal{U}_{\lambda_{\mathrm{KS}}}(\tau,\tau_0)\ \hat{\mathcal{T}}_\tau\ \hat{\mathcal{U}}_{\hat{H}_{\mathrm{KS}}}(\tau,\tau_0). \tag{2.22}$$

where $\hat{H}_{\mathrm{KS}}$ acts on the entire Kohn-Sham many-body wavefunction $|\Phi\rangle$ through its constituent orbitals $\left|\phi_j\right\rangle$, see Eq. (2.4). In general $\lambda_{\mathrm{KS}}(\tau)$ differs from the constant $\lambda(\tau)$ of the interacting system, and in addition to normalizing the many-body state, it can account for orthonormalization of the constituent orbitals.

Orthonormalization of the occupied orbitals is an invertible transformation preserving the subspace spanned by these linearly-independent states. Representing the orthonormalization by matrix $\mathbf{S}$ and given a single-particle Slater determinant wavefunction $\Phi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)$, orthonormalization results in $\tilde{\Phi}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N) = \Phi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)\det\mathbf{S}$. This can be seen as follows: first we write out the Slater determinant wavefunction $\Phi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)$,

$$\Phi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N) = \frac{1}{\sqrt{N!}}\det\{\mathbf{A}\}, \qquad \mathbf{A} = \begin{pmatrix} \phi_1(\mathbf{r}_1) & \phi_2(\mathbf{r}_1) & \cdots & \phi_N(\mathbf{r}_1) \\ \phi_1(\mathbf{r}_2) & \phi_2(\mathbf{r}_2) & \cdots & \phi_N(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{r}_N) & \phi_2(\mathbf{r}_N) & \cdots & \phi_N(\mathbf{r}_N) \end{pmatrix}. \tag{2.23}$$

The new many-body wavefunction $\tilde{\Phi}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)$ composed of orbitals $\{\tilde{\phi}_i\}$ is reached by the invertible transformation $\mathbf{S}$ which transforms the single-particle states according to $\tilde{\phi}_i(\mathbf{r}) = \sum_{j=1}^N \phi_j(\mathbf{r})S_{ji}$. We then have that $\tilde{\Phi}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N) = \frac{1}{\sqrt{N!}}\det\tilde{\mathbf{A}}$, where $\tilde{A}_{ik} = \tilde{\phi}_k(\mathbf{r}_i) = \sum_{j=1}^N \phi_j(\mathbf{r}_i)S_{jk} = \sum_{j=1}^N A_{ij}S_{jk} = (AS)_{ik}$, which implies $\tilde{\mathbf{A}} = \mathbf{AS}$. Thus, $\tilde{\Phi}(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N) = \frac{1}{\sqrt{N!}}\det(\mathbf{AS}) = \frac{1}{\sqrt{N!}}\det(\mathbf{A})\det(\mathbf{S}) = \det(\mathbf{S})\Phi(\mathbf{r}_1,\mathbf{r}_2,\ldots,\mathbf{r}_N)$, meaning that the two wavefunctions differ by at most a complex scalar. The orthonormalization step merely amounts to changing the phase and rescaling the many-body wavefunction.

At the starting time τ_0, we assume the Kohn-Sham wavefunction is properly normalized. Following the application of the imaginary-time propagator up to a particular time τ, we represent a particular orthonormalization of the orbitals by an invertible transformation $\mathbf{S}(\tau)$. In order for $\lambda_{\mathrm{KS}}(\tau)$ to act like the orthonormalization procedure, we require that

$$\mathcal{U}_{\lambda_{\mathrm{KS}}}(\tau,\tau_0)\ = \det\mathbf{S}(\tau). \tag{2.24}$$

Note that $|\det \mathbf{S}(\tau)|$ will be continuous since it is the reciprocal of the norm of the unnormalized propagated wavefunction. The phase of $\det \mathbf{S}(\tau)$ is not important since it changes the phase of the wavefunction, which will not affect the density. We can therefore use any orthonormalization procedure at each time-step without concern about the continuity of the phase, and a purely real $\lambda_{\mathrm{KS}}(\tau)$ satisfying $\mathcal{U}_{\lambda_{\mathrm{KS}}}(\tau,\tau_0) = |\det \mathbf{S}(\tau)|$ for all $\tau > \tau_0$ is guaranteed to exist.

The above definitions for norm-conserving $\lambda(\tau)$ and $\lambda_{\mathrm{KS}}(\tau)$ conclude the proof of the imaginary time extension to the van Leeuwen theorem presented in Section 2.3.1.

2.3.3 Monotonically Decreasing Energy

In the Kohn-Sham system the Hamiltonian depends on the density, and thus will in general have eigenenergies and eigenvalues that depend on time. In particular, for the density at time τ_ℓ, $n(\mathbf{r},\tau_\ell)$, we are considering a quantum system with the Hamiltonian $\hat{H}_{\mathrm{KS}}[n(\mathbf{r},\tau_\ell)]$. By propagating the state of interest in imaginary time using this instantaneous Hamiltonian, we are amplifying the low-energy eigenstates of the current Hamiltonian $\hat{H}_{\mathrm{KS}}[n(\tau_\ell)]$, which in general are different than the low-energy eigenstates of the new Hamiltonian, $\hat{H}_{\mathrm{KS}}[n(\tau_{\ell+1})]$, and the resultant state could have a higher energy than the previous state. A good example of this is the commonly-observed divergence of SCF loops without a mixing scheme: the N-lowest eigenstates of the Hamiltonian $\hat{H}_{\mathrm{KS}}[n_i]$ are directly used to compute the next density n_{i+1}. This also reveals an interesting limiting case of it-TDDFT. If a KS state is propagated to infinite imaginary time before the density used in the instantaneous Hamiltonian $\hat{H}_{\mathrm{KS}}[n(\tau_\ell)]$ is updated, the propagated state will become the ground state of the present Hamiltonian, which is equivalent to populating the N-lowest eigenstates of $\hat{H}_{\mathrm{KS}}[n(\tau_\ell)]$. In this way basic SCF can be thought of as it-TDDFT with infinitely large time-steps when using explicit propagation. Indeed, if the time-step in it-TDDFT is taken to be too large, the total energy will diverge, just like in SCF performed without a mixing scheme.

With a reasonable time-step, usually around 2 atomic units of time or smaller, it-TDDFT monotonically decreases the total energy of the system. The van Leeuwen theorem, which connects the KS system to the interacting system, provides the theoretical backbone for this result. While propagation of the Kohn-Sham system is complicated by the dependence on density, in the true interacting system the evolution in imaginary time has the simple form given in Eq. (2.3).

2.3.4 Alternative Theoretical Foundation for Stationary States in DFT

The first step in the majority of DFT calculations is to find a density corresponding to a stationary state. A stationary state is an eigenstate of the Hamiltonian, or equivalently, a state that only changes by a phase when evolved in real time or by a multiplicative factor when evolved in imaginary time. Only the first definition is used in KS systems, as it is implicitly assumed by SCF schemes. In systems that are difficult to converge with SCF, owing to their size or metallic character, the second definition becomes more useful, and it can be applied through the it-TDDFT method.

The KS equations are set as an eigenvalue problem, and thus use the first definition. Once a density $n(\mathbf{r})$ is found such that a choice of N of the orbitals $\phi_j(\mathbf{r})$ reproduces the same density through Eq. (2.8), a stationary state has been determined. SCF is used to find ground states, where the N lowest-energy eigenstates are chosen, and ΔSCF[35], where a different selection of N orbitals is chosen, is used to find excited states. For small systems, insulating systems, and systems with low degeneracy of orbitals, after a few steps of SCF, the eigenstates rarely change order when sorted by energy from one step to the next. This means that occupied orbitals have similar character to those from the step before, so the density does not change drastically. In these cases SCF converges well so using the eigenstate definition of stationary states is sensible. However, in large systems and in metallic systems, or if an excited state is desired, the above conditions might not hold, leading to charge sloshing. In principle, the KS equations can still be used to verify a stationary state if the density is perfectly converged. In practice, this definition is inadequate in these difficult systems since a suitable approximate density could appear to be far from convergence if the wrong KS eigenstates are occupied, due to the next SCF step returning a very different density from the one given. In addition, this makes it challenging to determine the quality of a non-converging density. For example, in Section 2.4 we examine the performance of SCF on a ruthenium nanocluster, where we show that some non-converged densities give a reasonable energy estimate for the ground state, while others are incorrect.

To address convergence issues, DFT calculations of metallic systems and systems with high single-particle energy degeneracy are often performed with electronic smearing, where states near the Fermi level are given fractional occupations to simulate nonzero electronic temperature. This mitigates the problem by ensuring that states near each other in energy have similar fractional occupation. Smearing adds an entropic contribution to the energy, so a balance between obtaining an accurate energy and ease of convergence has to be struck. Electronic smearing is a computational tool and not intended to be an accurate representation of the effects of temperature, so it should be incrementally reduced until the solution with no smearing is achieved[49], a technique referred to as annealing.[60] In fact, cases have been found where even

small amounts of electronic smearing produce significantly different results from the same calculation performed with integer occupations, such as a HOMO-LUMO gap energy that differs by one order of magnitude[4]. As we show in the ruthenium nanocluster system in Section 2.4, achieving convergence while applying electronic smearing can still require finesse and guesswork.

In systems where SCF convergence is hard to attain, instead of using the KS equations to define a stationary state, we can use a state's invariance under imaginary-time evolution. If a KS wavefunction stays constant when propagated in imaginary time, then its orbitals span the same subspace as a set of eigenstates which solve the KS equations. The converse is true as well, namely that a set of N KS eigenstates satisfying the KS equations self-consistently will be invariant under imaginary-time evolution, ignoring the possibly changing norm which can be corrected (as discussed in Sec. 2.3.2). Thus, finding a KS many-body state $|\Phi(\tau_0)\rangle$ such that $|\Phi(\tau)\rangle = \hat{\mathcal{U}}(\tau,\tau_0)\,|\Phi(\tau_0)\rangle = |\Phi(\tau_0)\rangle$, where the Hamiltonian in the propagator $\hat{\mathcal{U}}$ contains orthonormality-preserving $\lambda(\tau)$, is equivalent to finding a set of N orbitals that satisfy the KS equations.

This definition has a few advantages. In systems where the orbitals are close in energy, occupation ambiguities and charge-sloshing issues are eliminated because it-TDDFT follows the occupied orbitals throughout their evolution. Additionally, it-TDDFT handles systems with degenerate states well since an initial state will converge to one of the states within the degenerate stationary-state subspace without being affected by the unoccupied states of identical energy.

2.3.5 Practical Advantages of it-TDDFT

One convenience afforded by it-TDDFT is that a user only needs to choose a single parameter, the time-step, when attempting to converge a system. Compare this to the various parameters usually required for SCF with a mixing scheme: the number of past states to mix, the mixing weight, and the amount of electronic smearing, to name a few. When encountering a set of nonconvergent parameters, it is often unclear which direction to change each parameter for a better chance at convergence. In addition, there are systems where different stationary states can be obtained for slight variations in the mixing parameters, as shown in the case of a Cu_{13} cluster in Section 2.4. In contrast, convergence in the it-TDDFT method is not very sensitive to the choice of time-step, and any choice smaller than a convergent time-step will lead to the same density trajectory in imaginary time given the same starting state. This property allows us to eliminate this parameter choice if desired through algorithms that automatically adjust the time-step on the fly. We found that the simple procedure of increasing the time-step while total energy decreases, and decreasing the time-step when it does not, can perform nearly as well as using a static convergent time-step that is as large

as possible.

Another practical advantage of using imaginary-time evolution is that not-yet-converged states still have physical meaning. The orbitals and the electronic density used in the KS Hamiltonian are self-consistent at all times, and in principle this density trajectory is equal to the imaginary-time evolving density of the interacting system by the van Leeuwen theorem. Through this connection, the partially-converged KS state corresponds to a superposition of a dominant ground state component and a few low-amplitude excited states. As such, even before the it-TDDFT ground state calculation has converged according to user-specified energy or density tolerance criteria, approximate ground state observables can be computed, and a sequence of these calculations along the imaginary-time trajectory will give an indication of their accuracy as they asymptote to their ground state values. This property allows for preliminary calculations of band structure, energies, optical properties, or atomic forces while the ground state calculation continues to be refined. In contrast, there are no guarantees of validity for observables calculated from intermediate states produced in a SCF loop since they are not self-consistent, not physically related from step to step, and their distances to the correct KS ground state are difficult to determine.

When an SCF loop ceases to make progress, not much is gained aside from the knowledge that the particular set of mixing parameters did not lead to convergence. It could take a subtantial amount of tweaking of these parameters before chancing upon a set that works, consuming time and computational resources. This highlights another strength of it-TDDFT: the calculation time used will always improve the quality of the state at hand. It is also straightforward to continue a calculation from the last saved state, enabling incremental improvement of an approximate stationary state over multiple runs.

In our discussion we have assumed that we are performing DFT with a Kohn-Sham system, which uses a single Slater determinant. It is possible to apply it-TDDFT for finding stationary states in other approaches which use linear combinations of Slater determinants, or ensemble DFT, since a DFT model that reproduces the same density trajectory as the true interacting system will evolve an arbitrary starting state into a stationary state when propagated in imaginary time.

2.4 Example Calculations

In order to compare two different densities, it is useful to have a measure of distance. We will use half the L^1 distance for its intuitive physical meaning:

$$D[n, n_0] \equiv \frac{1}{2} d_1(n, n_0) = \frac{1}{2} \int |n(\mathbf{r}) - n_0(\mathbf{r})| \, \mathrm{d}^3\mathbf{r}, \tag{2.25}$$

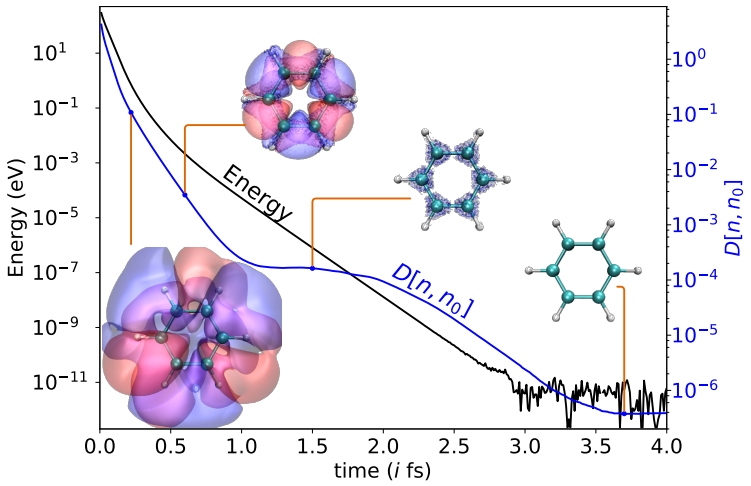

Figure 2.2: Determining the ground state energy E and density distance $D[n,n_0]$ of benzene using it-TDDFT, relative to SCF results. The time step used was 10.0 as. Positive and negative isosurfaces of the density difference $n(\mathbf{r}) - n_0(\mathbf{r})$ at fixed values are shown at various points in the propagation.

which can be interpreted as the number of electrons in the wrong place relative to the reference density n_0. This can be seen by using the fact that both densities integrate to the same value, the total number of electrons. The integral of the absolute value of the density difference over all space adds up the excess density and the negative of the lacking density, both contributing equally, so the $1/2$ factor is needed to obtain the number of electrons out of place.

As a demonstration of using it-TDDFT to determine a ground state, we apply the method to a benzene molecule and show that it produces the same density and energy as a standard SCF calculation. We use the PBE functional and the default double-zeta plus polarization (DZP) basis set generated by SIESTA. We initialize the KS wavefunction by drawing basis coefficients from a uniform distribution and orthonormalizing the orbitals. Propagating this initial state in imaginary time, we obtain the same ground state as that determined by an SCF approach with Pulay mixing. Given the same resources, the SCF calculation took 36 s to complete, 0.7 s per step, while it-TDDFT took 1316 s at 3.3 s per step. The latter calculation was performed with a constant time-step in order to accurately plot the trajectory through imaginary time; using the automatically-adjusting time-step described in Section 2.3.5 reduces the total calculation time to 130 s. The Kohn-Sham total energy E and the density distance $D[n,n_0]$ of the propagated state are plotted as a function of imaginary time in Fig. 2.2, both relative to the SCF-determined ground state. These quantities tend to zero, showing that it-TDDFT indeed produces a Kohn-Sham state that has the same energy and density as the ground state determined with an SCF approach. As an additional check, running SCF at the end of the imaginary-time propagation produces SCF convergence after the first step.

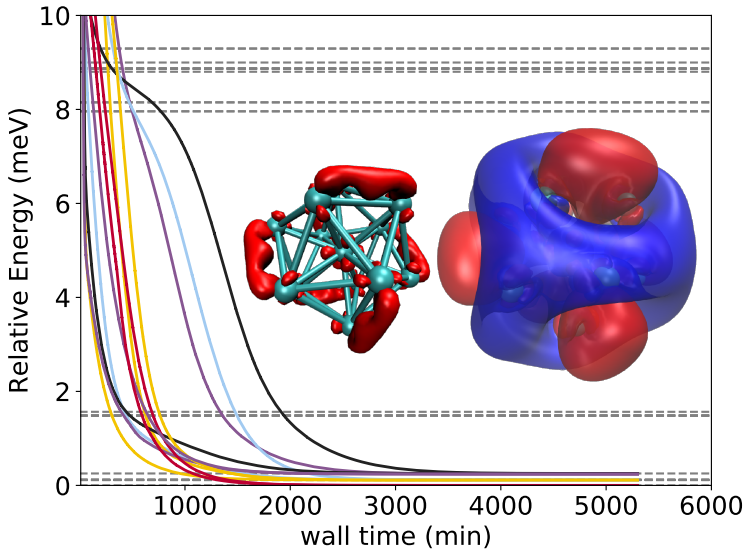

Figure 2.3: Electronic energy of Cu_{13} with fixed spin polarization $+1/2$, relative to the lowest energy obtained with this fixed polarization. Each curve is an energy trajectory produced by propagating a random initial state in imaginary time, plotted versus the wall time and colored according to the final state obtained. The right inset plot is the spin magnetization density of one such state with spin up and spin down designated by blue and red respectively. The left inset plot is a spin down isosurface to illustrate the five-fold symmetry of the lowest energy states. The horizontal dashed lines show the relative energies of converged states obtained using SCF and Pulay mixing with different parameters as detailed in Fig. 2.4 and Table 2.1.

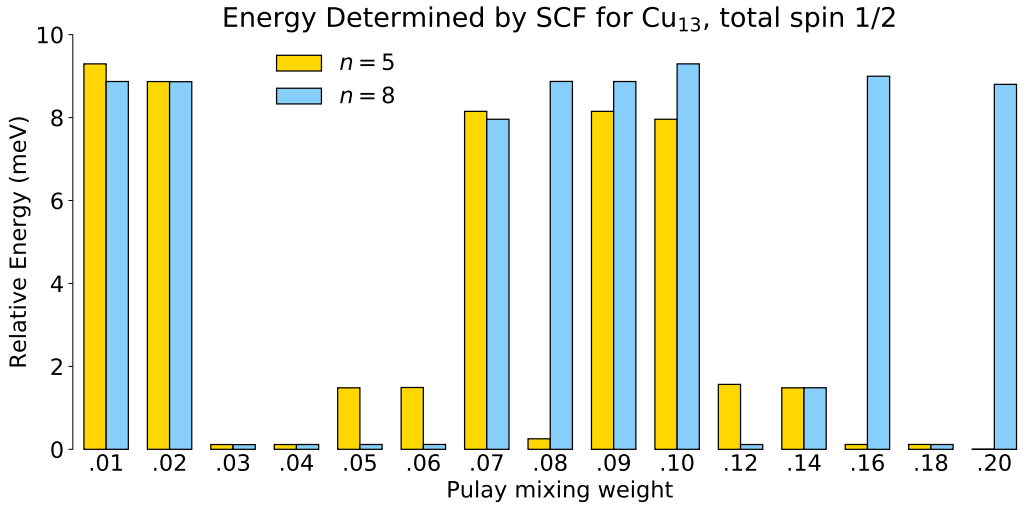

Figure 2.4: Relative total energy of Cu_{13} cluster with fixed total spin $1/2$, obtained by SCF with Pulay mixing involving $n = 5$ or 8 previous densities and different mixing weights. The reference value of the energy is that obtained with imaginary-time propagation. The energies here appear in Fig. 2.3 as horizontal dashed lines.

As our next example we consider the Cu_{13} nanocluster. Hoyt *et al.*[27] simulated this system in its ground state magnetization of $m = 5\mu_B$ and in an excited state with magnetization $m = 3\mu_B$, commenting that the $m = 1\mu_B$ excited state was tricky to converge, making it a good candidate for our it-TDDFT method. Specifics regarding the calculations such as the basis set and functional can be found in the Supporting Information. The mixing scheme used for the SCF runs is a modified version of Pulay mixing implemented in SIESTA where the number of past steps to use and a mixing weight can be specified as parameters.[37] In Fig. 2.3 and 2.4, we present the main results of our computations for the self-consistent KS states with $m = 1\mu_B$ magnetization, which has total spin $1/2$. The SCF calculations generally converged in 10 to 100 min, taking 2.3 s per step, though occasionally the total times were closer to 1000 min. The calculations performed with it-TDDFT converged in around 2000 min, taking 8.6 s per step. Additional information can be found in Table 2.1. SCF has trouble finding the minimum energy states in this fixed-spin system, due to the fact that there are five degenerate states[27]. Fig. 2.3 shows the energy trajectories in imaginary time of 12 different random starting configurations, and each of these converges to one of five lowest-energy states. To help identify the equality of final states, for both the it-TDDFT and SCF runs, we also computed the density distances between each combination of obtained states. States with energies within 10^{-2} meV and a density distance of less than $(1/100)e$ of each other were considered equal. The ground state of the system, with magnetization $m = 5\mu_B$, contains five degenerate valence electrons which have unpaired spins[27]. To obtain a magnetization of $m = 1\mu_B$, four of the electrons need to pair spins, leaving five possibilities of which electron remains unpaired. Fig. 2.3 shows the magnetization density, defined as the difference between spin up and spin down electron density, of one of these five lowest-energy states. There are five equivalent ways to place such a magnetization density on the icosahedral shape of the copper cluster, explaining the degeneracy. The visible differences of up to 0.1 meV in the energies of these degenerate states are due to the discretization effects of the real space grid breaking the icosahedral symmetry.

Our approach is better at finding the lowest-energy states compared to SCF, which for different mixing parameters often converges to other excited states of spin $+1/2$ (the energies of which are shown as dashed lines in the figure). In Fig. 2.4, we show the electronic energies of the states obtained using SCF with Pulay mixing, for various mixing parameter choices. Even small changes in the mixing parameters can result in a different final state. This happens in metallic systems where the gap between occupied and unoccupied states is small, causing SCF to find a low-lying excited state.

For our final example of applying our method we consider the Ru_{55} nanocluster. Montemore *et al.*[52] studied catalysis on the surface of this structure, and found that the spin-unpolarized ground state calculation was difficult to converge with SCF.

States Determined by SCF with Pulay Mixing for Cu_{13}, total spin $1/2$

	$n = 5$			$n = 8$		
Mixing Weight	Energy (meV)	Time (min)	Min. State	Energy (meV)	Time (min)	Min. State
0.01	9.2955	2021.7	No	8.8701	787.8	No
0.02	8.8690	2386.0	No	8.8656	750.6	No
0.03	0.1155	18.7	Yes	0.1135	26.8	Yes
0.04	0.1156	51.0	Yes	0.1162	14.9	Yes
0.05	1.4833	30.7	No	0.1164	15.4	Yes
0.06	1.4910	64.1	No	0.1176	13.9	Yes
0.07	8.1505	163.4	No	7.9599	29.3	No
0.08	0.2538	28.0	Yes	8.8722	119.4	No
0.09	8.1506	61.2	No	8.8698	207.5	No
0.10	7.9600	60.3	No	9.2955	137.4	No
0.12	1.5656	22.8	No	0.1159	17.9	Yes
0.14	1.4832	34.3	No	1.4844	15.3	No
0.16	0.1163	7.0	Yes	8.9980	32.0	No
0.18	0.1170	6.8	Yes	0.1162	10.7	Yes
0.20	0.0003	7.4	Yes	8.8015	54.8	No

Table 2.1: Electronic states of spin $1/2$ Cu_{13} obtained from SCF with no electronic smearing. For the Pulay mixing weight and number of past densities n used in the mixing scheme, we list the relative energy, wall time, and whether the converged state matches one of the five lowest-energy states determined with it-TDDFT. The wall times can be compared to those shown in Fig. 2.3.

In Table 2.2, we show the results of using SCF with Pulay mixing to find the ground state of the spin-unpolarized Ru_{55} cluster. Specifics regarding the calculations such as the basis set and functional can be found in the Supporting Information. The number of past densities to mix was kept at $n = 5$ for all trials and mixing weights ranging from 0.02 to 0.20 were tested. We used Fermi electronic smearing for half the trials with $T = 300\,\mathrm{K}$. For each run, we list the energy of the final step relative to the energy calculated with it-TDDFT, the density difference $\Delta\rho_{\max}$, and whether the run converged or not. The density difference $\Delta\rho_{\max}$ refers to the maximum elementwise difference in the density matrix between the final and penultimate step and is typically used to determine convergence. We used the criterion $\Delta\rho_{\max} < 10^{-6}$. Only a few mixing weights result in convergence, namely the smallest ones with $T = 300\,\mathrm{K}$ of smearing. In these runs, the entropic energy contribution is 78 meV relative to the ground state energy. None of the runs without electronic smearing converge, despite the fact that some parameter configurations obtain energies similar to the ground state energy. The states resulting from unconverged runs generally should not be trusted as they may not be acceptable approximations to actual solutions, which have to satisfy the KS equations self-consistently. For example, in Table 2.2, examining the row with mixing weight 0.10 and

SCF with Pulay Mixing for Ru_{55}, Spin Unpolarized

	$T = 0\,\mathrm{K}$			$T = 300\,\mathrm{K}$		
Weight	Energy (meV)	$\Delta\rho_{\max}$	Converged	Energy(meV)	$\Delta\rho_{\max}$	Converged
0.02	−0.012	0.086	No	78.388	9.9×10^{-7}	Yes
0.04	−0.004	0.052	No	78.380	9.8×10^{-7}	Yes
0.06	0.069	0.227	No	78.448	5.3×10^{-7}	Yes
0.08	0.395	0.185	No	78.408	9.0×10^{-7}	Yes
0.10	5.538	0.702	No	6.38×10^{3}	0.589	No
0.12	73.276	0.730	No	7.15×10^{4}	0.699	No
0.14	148.995	1.388	No	1.46×10^{5}	1.391	No
0.16	6.385	0.589	No	2.35×10^{5}	1.409	No
0.18	295.051	1.441	No	2.95×10^{5}	1.446	No
0.20	353.573	1.444	No	3.54×10^{5}	1.455	No

Table 2.2: Ground state electronic configurations of a Ru_{55} nanocluster using SCF with Pulay mixing, with a $n = 5$ density history length and mixing weights ranging from 0.02 to 0.2. $\Delta\rho_{\max}$ is the maximum elementwise difference in the density matrix between the final and penultimate step, with convergence criterion $\Delta\rho_{\max} < 10^{-6}$.

comparing the $T = 0\,\mathrm{K}$ and $T = 300\,\mathrm{K}$ cases, we find that even though $\Delta\rho_{\max} = 0.589$ in the latter run is smaller than $\Delta\rho_{\max} = 0.702$ in the former, the energy of the state obtained with $T = 300\,\mathrm{K}$ is more than 6 eV off from the correct ground state energy while the $T = 0\,\mathrm{K}$ run is only about 0.006 eV off. Applying our it-TDDFT method to the Ru_{55} cluster produces the ground state without issue, as illustrated in Fig. 2.5. The observed monotonically decreasing energy and density distance $D[n, n_0]$ show consistent progress, as we expect from the theory. In the SCF calculations that did converge, which all required Fermi smearing, the total computational time was around 10 min to reach a state within 78 meV of the ground state. While the it-TDDFT calculation took 300 h to reach numerical convergence, as seen in Fig. 2.5 the propagated state was less than 50 meV from the ground state by 10 h of wall time.

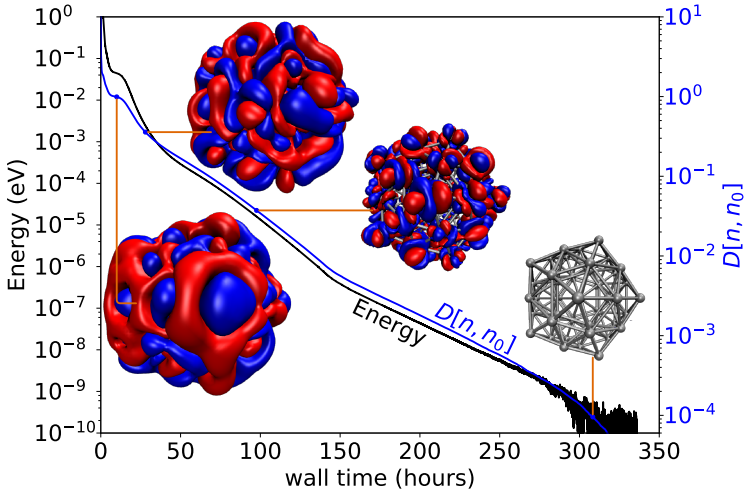

Figure 2.5: Electronic energy and density distance $D[n,n_0]$ trajectory of a spin unpolarized Ru_{55} cluster measured relative to the state it converges to, as obtained by it-TDDFT. Positive and negative isosurfaces of $n(\mathbf{r}) - n_0(\mathbf{r})$ are shown at various points in the propagation.

2.5 Conclusion

The first step of any Kohn-Sham DFT calculation is the determination of a self-consistent solution to the KS equations, resulting in a density corresponding to a stationary state of the many-body interacting system. While the standard method of using the iterative SCF procedure generally produces a solution efficiently, there are important classes of systems that pose problems for this approach due to their small band gaps or degenerate single-particle energies. We have proposed the it-TDDFT method as an alternative means for solving the KS equations in these difficult systems, and shown how it avoids the issues which affect SCF.

We established that the van Leeuwen theorem, a key theoretical foundation for TDDFT methods, can be extended to imaginary time, thereby ensuring convergence to a stationary state independent of the exchange-correlation potential and level of theory used in the model system. In addition, we discussed how it-TDDFT could be used in an alternative but equivalent definition of stationary states in DFT, better suited for metallic systems and systems with degenerate or nearly-degenerate states and based on the time-dependent Kohn-Sham equations. The it-TDDFT method also exhibits a number of practical advantages, such as justifying approximations to observables of interest before the ground state calculation is fully converged, requiring few input parameters, and allowing easy refinements of the results of previous runs by continuing from a saved state.

In the copper and ruthenium nanoclusters considered here, we demonstrated how SCF can struggle to find the electronic ground state, either converging to low-lying excited states or getting stuck in charge-sloshing

cycles. These systems were readily converged by it-TDDFT, showcasing its robustness through smooth trajectories with monotonically decreasing energy. For these systems we either ran the calculation as spin-unpolarized, or with a fixed total spin. This is not an inherent limitation of the method, as one could simply run the calculation with all possible spin polarizations and select the state with the lowest energy. The method can be adapted to non-collinear spin systems, since the operating principle depends only on the Hamiltonian being able to differentiate states by energy. Furthermore, while we used finite systems for our example calculations, our method can be extended to find ground states of periodic systems by simultaneously propagating Kohn-Sham states at multiple k-points.

Given an existing TDDFT code which evolves systems in real-time, it should be relatively straightforward to implement a prototype of the presented it-TDDFT approach, requiring only an imaginary time substitution in the propagation step and a method to orthonormalize the KS orbitals. For the systems considered here and others we tested while developing the method, the computation time for finding ground states with it-TDDFT was 10 to 100 times longer than for convergent SCF runs, depending on the complexity of the system and precision desired. While more efficient implementations could be examined in the future, the low barrier to utilizing it-TDDFT could make it an attractive alternative option for those dealing with particularly vexing systems.

2.6 Supporting Information

Descriptions of the Cu_{13} and Ru_{13} systems specifying the geometry, functional, and basis set can be found in Appendix B.

2.7 Acknowledgments

This work was supported by the Army Research Office Multidisciplinary University Research Initiative (MURI), Award No. W911NF-14-0247. We used computational resources on the Odyssey cluster (FAS Division of Science, Research Computing Group at Harvard University) and the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF Grant No. ACI-1053575.

Part II

Solving Differential Equations with Unsupervised Neural Networks

Page intentionally left blank

3

Introduction to Neural Networks

3.1 Introduction

Machine learning has become an incredibly popular topic in recent times, with both engineers in industry and researchers in academia pushing to find new ways to apply the tools of this rapidly developing field. Generally speaking, learning is the process of converting experience into expertise or knowledge. In the case of machine learning, the experience input to a learning algorithm comes in the form of a training dataset, and the output is some developed capability[68]. Machine learning shines when applied to tasks that are too complex to program by hand, either because of our own inability to introspectively identify the basic blocks that compose a complex task that we innately find easy to perform, like facial recognition or speech understanding, or because the task exceeds our human capabilities, such as in the analysis of very large datasets or very complex patterns. By developing a formalism for learning tasks in general, we can gain a deeper understanding across a vast range of disciplines.

Currently, machine learning can be separated into *supervised learning*, *unsupervised learning*, and *reinforcement learning*[72]. Supervised learning tasks utilize *training data* that consists of both inputs and outputs of the process or function to learn, where the outputs come from a knowledgeable external "supervisor" who has assigned the correct outputs to each input. For example, in an image classification problem, the training data would consist of images as inputs and labels that describe what class they belong in as outputs. In a regression task, training data inputs come from the domain of the underlying function to be learned, and outputs from the function evaluated at the input plus some irreducible noise. Unsupervised learning seeks to find structure in data, constructing a model that can describe the internal similarity and differences of the data in a compact manner. Clustering is an example of this class of learning tasks where points in a vector space are grouped into disjoint sets by relative proximity to each other. If the data consists of images, grouping the images by content without explicitly knowing what distinguishes the possible content classes is also an example of unsupervised learning. In Chapter 4 we will discuss an unsupervised

approach of training neural networks to approximate the solutions of differential equations, in contrast to a supervised approach where the solution would have to be known *a priori*, with inputs and outputs determined via evaluating the analytic solution if available, or by using conventional numerical methods. The final category of machine learning is reinforcement learning. This class of problems deal with capturing the most important aspects of a problem facing a learning agent interacting over time with its environment to achieve a goal[72]. Examples of these sorts of tasks are training a computer to drive a car, play a game, or control a complex industrial operation. In the following chapter we will focus on supervised learning, which tends to have simpler base concepts which can also be applied to the other two classes of learning.

There are many forms of training datasets which depend on the task we intend to solve. We will focus on regression tasks where the goal is to construct a function that best approximates an unknown underlying target function. Suppose that the target function is $\mathbf{f} : \mathcal{D} \to \mathbb{R}^m$, $\mathcal{D} \subset \mathbb{R}^n$. In a regression task, training data comes in the form of a set of pairs of inputs and outputs, $\{(\mathbf{x}_i, \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\epsilon}_i)\}$, where $\boldsymbol{\epsilon}_i$ are noise terms that add aleatoric error, *i.e.* inherent stochastic error. Regression seeks to construct an approximate function $\hat{\mathbf{f}}$ such that on arbitrary $\mathbf{x} \in \mathcal{D}$, the difference between $\hat{\mathbf{f}}(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$ is minimize based on a given metric. In parametric methods, $\hat{\mathbf{f}}(\mathbf{x}; \mathbf{w})$ depends on a finite number of parameters stored as elements within the vector $\mathbf{w}$ which tune the shape of $\hat{\mathbf{f}}$ according to the training data in order to accomplish the regression task.

The most well-known regression model is linear regression, where the approximating function is assumed to have a linear relationship to its inputs:

$$\hat{\mathbf{f}}(\mathbf{x}) = \boldsymbol{\beta}^\intercal \mathbf{x} + \beta_0. \tag{3.1}$$

The assumed linear model only works well if the true underlying function $\mathbf{f}$ is suspected to be very nearly linear, which is often not the case. To account for nonlinearity, the space of the approximating model has to be expanded, which generally adds more parameters. Polynomial regression is a logical step above linear regression, allowing for polynomial dependence on the input vector elements in the model. Increasingly complex models allow for better approximations to arbitrary functions if sufficient data is available.

Neural networks are a highly expressive family of functions that have captured great interest in the past decade. These artificial neural networks are graphs of connected computations that were first proposed in 1943 by McCulloch and Pitts[47], who created a computational model for nervous activity. One of the first neuro-inspired learning models was the perceptron invented by Rosenblatt in 1958[62], which intended to emulate a single neuron. While showing promise for use in a wide variety of learning tasks, the perceptron was eventually discounted for its mathematically-proven inability to represent certain important classes of

patterns[50]. However, it was eventually shown that these problems could be overcome by layering stacks of perceptrons, *i.e.* passing the outputs of perceptrons as inputs of later perceptrons, resulting in an object that could approximate any continuous function, as will be discussed in Section 3.3. In 1975 Werbos proposed *backpropagation*[87], a practical method to train these layered networks by employing automatic differentiation and gradient descent, poising this expressive class of functions to become the dominant thrust in machine learning and artificial intelligence research. Since the 2000s, *deep learning*, the training of multilayered perceptrons of more than two layers, has thrived through the advances in computational power and the cheap parallel computing power afforded by commercially-available graphics processing units (GPUs).

3.2 Artificial Neural Networks

Artificial neural networks are a general class of functions with significant expressive power, scalable complexity, and a form amenable to optimization against a desired objective function. In the past decade we have seen networks that have been trained to describe the content of images[29], classify text into predefined categories[90], drive a car[78 7], and generate sensible short texts on specified subjects[61]. Mathematically, a neural network is described by a directed graph, with nodes corresponding to neurons and edges corresponding to connections between them. This graph is commonly referred to as the neural network's architecture. Starting from the input and proceeding to the output, scalar values flow through the graph while nodes and edges perform mathematical operations on them. As will be discussed in Section 3.3, even with a restricted architecture and limited set of simple operations, neural networks can approximate any continuous function to a specified tolerance using a finite number of neurons, a result called the universal approximation theorem.

3.2.1 Multilayer Perceptron

Multilayer perceptron networks, often referred to as multilayer perceptrons or MLPs, form a common class of neural networks. MLPs are a subclass of feedforward network structures, the class of neural networks whose underlying graph does not contain cycles, *i.e.* a directed acyclic graph. In addition to this acyclic constraint, MLPs organize neurons into layers such that neurons in each layer only have directed edges feeding into neurons in the successive layer.

The following mathematical definition of MLPs is based on the textbook *Understanding Machine Learning* by Shalev-Shwartz and Ben-David[68]. An MLP is a directed acyclic graph $G = (V, E)$, with nodes

V and edges E, and a weight function over the edges, $w : E \to \mathbb{R}$. The graph is further constrained to be composed of a union of nonempty disjoint subsets, $V = \bigcup_{\ell=0}^{\mathcal{L}} V^\ell$, such that every edge in E connects some node in $V^{\ell-1}$ to some node in V^ℓ, for some $\ell \in [\mathcal{L}]$. Each V^ℓ is referred to as a *layer*, and V^0 is specifically called the *input layer* while $V^{\mathcal{L}}$ is the *output layer*. All other layers in between are called *hidden layers* because their intermediate outputs are typically not accessed. Such a layered network is referred to as having *depth-*$\mathcal{L}$, that is, the total number of layers $\mathcal{L}$ without counting the input layer. Let v_i^ℓ refer to the ith node in layer ℓ. Each node v_i^ℓ in the graph is called a *neuron*, which has an associated scalar function $\alpha_i^\ell : \mathbb{R} \to \mathbb{R}$, called an *activation function*, which it applies to values passing through it. A few common activation functions are plotted in Figure 3.1. Each edge in the graph, referred to as a *connection*, has a weight associated to it by the mapping w, which is used by neurons to perform a weighted sum of the values arriving from all incoming connections. In each layer, there are $n^\ell + 1$ neurons, where n^ℓ is the dimensionality of the space following layer V^ℓ. The extra neuron v_0^ℓ in each layer is a "constant" neuron, which always outputs 1 and has no incoming connection. These constant neurons and their associated weighted connections can be used to apply a bias, a constant offset, to the weighted sum computed in each neuron in the subsequent layer.

Let $o_i^\ell(\mathbf{x})$ be the output of neuron v_i^ℓ when the network is given input vector $\mathbf{x}$. The input layer V_0 simply outputs each element of the input $\mathbf{x}$, so $o_i^0(\mathbf{x}) = x_i$, and $o_0^0 = 1$ for the constant neuron. Somewhat confusingly, the output of a neuron is often called the neuron's activation, and the distinction between whether this refers the output or activation function is left to the context. Neurons in subsequent layers have outputs that depend on the weighted outputs of the layers coming before them via

$$o_i^\ell(\mathbf{x}) = \alpha_i^\ell\left(\sum_{j:\left(v_j^{\ell-1}, v_i^\ell\right)\in E} w\left(\left(v_j^{\ell-1}, v_i^\ell\right)\right) o_j^{\ell-1}(\mathbf{x})\right), \qquad \ell \in [\mathcal{L}], \tag{3.2}$$

$$o_0^\ell(\mathbf{x}) = 1, \qquad \ell \in 0, \ldots, \mathcal{L}-1. \tag{3.3}$$

Generally, MLPs consist solely of *dense* layers where every neuron in a layer connects to every other neuron (except the constant neuron) in the following layer, and every neuron within a layer uses the same activation. This allows us to rewrite the above equation as

$$o_i^\ell(\mathbf{x}) = \alpha^\ell\left(\sum_{j=0}^{n^{\ell-1}} w_{ij}^\ell o_j^{\ell-1}(\mathbf{x})\right), \qquad \ell \in [\mathcal{L}], \tag{3.4}$$

$$o_0^\ell(\mathbf{x}) = 1, \qquad \ell \in 0, \ldots, \mathcal{L}-1, \tag{3.5}$$

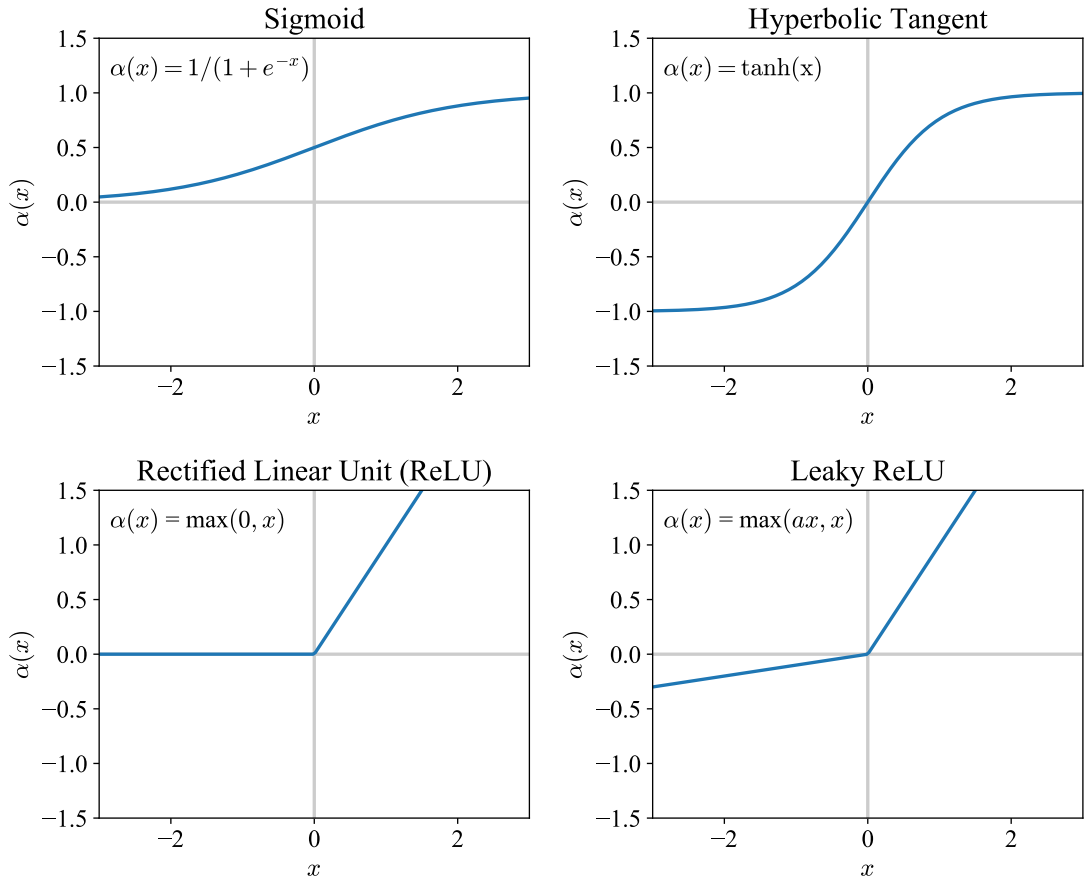

Figure 3.1: Some common activation functions.

where $w_{ij}^{\ell} \equiv w\left(\left(v_j^{\ell-1}, v_i^{\ell}\right)\right)$. In addition it is conventional to separate out the contributions of the constant neurons to explicitly express them as bias terms:

$$o_i^{\ell}(\mathbf{x}) = \alpha^{\ell}\left(\sum_{j=1}^{n^{\ell-1}} w_{ij}^{\ell} o_j^{\ell-1}(\mathbf{x}) + b_i^{\ell}\right), \qquad \ell \in [\mathcal{L}], \tag{3.6}$$

where $b_i^{\ell} \equiv w_{i0}^{\ell}$. When expressed in this form, the bias term is associated with each non-input neuron instead of an incoming connection from a constant neuron, and we no longer include the constant nodes v_0^{ℓ} in diagrams of the network. Figure 3.2 shows a schematic of a dense MLP, and Figure 3.3 shows the calculation happening at each non-input neuron.

In addition to dense layers, specialized layer architectures are widely used in the field, serving as modular bricks with which one can construct complex networks based on intuition and through trial and error based

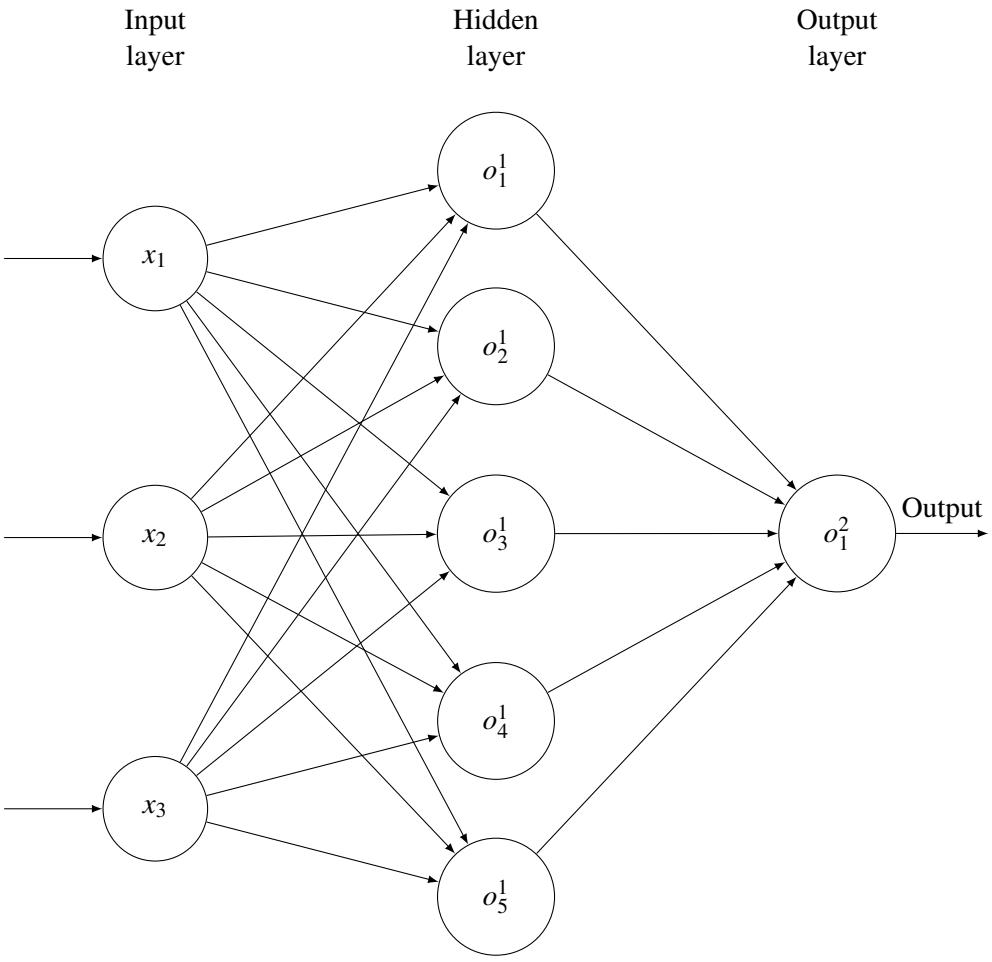

Figure 3.2: A diagram of a depth-2 dense multilayer perceptron with input dimension 3, output dimension 1, and width 5. Each neuron is labeled with the value it outputs according to Eq. (3.6).

on empirical performance. Many of these layers exhibit *weight sharing*, where the same neuron participates in multiple parts of the network ensuring that weight updates during training affect all these operations equally. When this feature is used appropriately it can greatly speed up training and generalization performance on unseen data. For example, *convolutional layers*[39], typically used for neural networks taking images as inputs, were inspired by biological processes, with neurons resembling the organization in the visual cortex[18]. These layers perform a discrete convolution of a block of weights, called a kernel, and an image represented as a matrix of pixel values, resulting in a *feature map* of activations. The feature map can itself be thought of as an image, but its pixel values correspond to the strength of the feature that the kernel was trained to pick out, effectively giving a heat map of the parts of the image exhibiting the feature. With

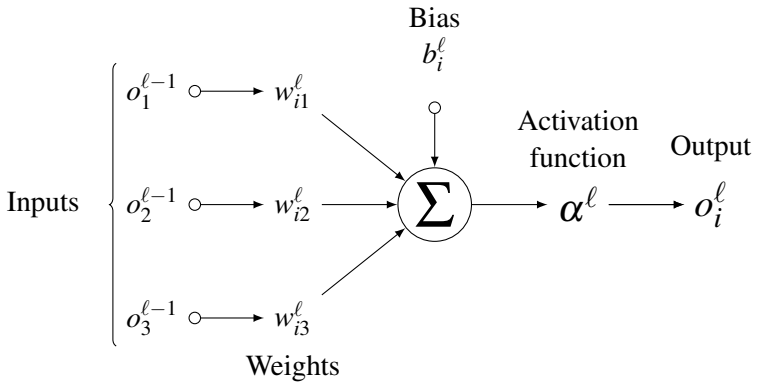

Figure 3.3: A diagram of neuron v_i^ℓ and its incoming connections. This is a visual representation of Eq. (3.6).

the aid of additional convolutional layers and possibly *pooling* layers that downsample the feature map, a neural network exhibiting translational invariance can be created, where the output of a network is independent of location of a feature it learns to detect in the image. For example, an appropriately designed convolutional neural network trained to detect faces could detect faces positioned anywhere in the image, even if the training data only had faces directly centered in the image. In essence, the translational symmetry of the task was used to constrain the network architecture, resulting in a model that better understands the structure of the data. *Recurrent neural networks*[42] are used on sequences of data. After taking in one datum in a sequence, a recurrent layer feeds portions of its output back as part of its input alongside the next datum in the sequence, so the weights in the layer can be thought of as being shared "in time" along the length of the data sequence. This structure makes it well-suited for processing data exhibiting temporal relationships, like text, measurements from dynamic systems, and speech. A specific class of recurrent neural networks fall under reservoir computing[44], which features recursive connections within a fixed collection of neurons called a reservoir. In these networks, only a simple readout layer of the reservoir is trained, greatly simplifying the training process compared to standard recurrent neural networks while still maintaining many of their advantages. Novel network architectures with general use cases continue to be discovered and applied in neural network applications.

3.3 Universal Approximation Theorem

One of the key features contributing to the power of neural networks is that they can approximate any continuous function, and with greater architecture complexity, refine this approximation to any desired tolerance. This flexibility is key to their use in machine learning for capturing underlying structure in data.

Mathematically, this property of neural networks is referred to as the *universal approximation theorem*.

3.3.1 SHALLOW NETWORKS

The distinction between shallow and deep neural networks is not fully agreed upon, but the general consensus is that a *credit assignment path* (CAP), the chain of transformations from input to output, of depth greater than 2 can be considered deep. For a layered network, a CAP depth of 2 corresponds to an input layer, a single hidden layer and an output layer, since only the last two layers are parameterized and perform transformations on their inputs. The universal approximation theorem was first proven for these depth-2 MLPs.

Universal approximation theorem. *Let $\alpha : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function (the activation function). Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $\mathbf{w}_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that the approximation*

$$F(\mathbf{x}) = \sum_{i=1}^{N} v_i \alpha\left(\mathbf{w}_i^{\mathsf{T}} \mathbf{x} + b_i\right) \tag{3.7}$$

satisfies

$$|F(\mathbf{x}) - f(\mathbf{x})| < \varepsilon \tag{3.8}$$

for all $\mathbf{x} \in I_m$.

A version of this result was first introduced by George Cybenko for sigmoidal activations in 1989 [10]. It has since been generalized to arbitrary bounded and nonconstant activation functions by Hornik [26], demonstrating that it is the architecture of the layered feedforward network itself that grants the universal approximation property and not the sigmoidal shape of the activation. More recently, Sonoda and Murata have shown that the universal approximation property also holds when employing non-polynomial unbounded activations like the widely-used rectified linear unit activation, $\mathrm{ReLU}(x) = \max(x, 0)$ [70]. Note that Eq. (3.7) has the form of a depth-2 MLP. It consists of a single dense hidden layer with biases and an activation function which feeds into an output layer containing a single neuron, no biases, and a linear activation. Complexity is added to the network architecture using additional neurons in the hidden state.

To build some intuition on how a neural network can represent an arbitrary continuous function, now we

prove a simpler restricted statement of the universal approximation theorem.

A restricted universal approximation theorem. *Let $f : [-1,1] \to \mathbb{R}$ be a ρ-Lipschitz function. For some $\varepsilon > 0$, a neural network $N : [-1,1] \to \mathbb{R}$ with the sigmoid activation function can be constructed such that for every $x \in [-1,1]$ it holds that $|f(x) - N(x)| < \varepsilon$.*

Proof. First, consider the following expression for a depth-2 multilayer perceptron with sigmoid activation $\sigma(t) = 1/(1+e^{-t})$:

$$N(x) = \phi + \sum_{i=1}^{n} c_i \sigma(w_i x + \theta_i). \tag{3.9}$$

The network consists of the input layer x, n biases θ_i, n weights w_i, a hidden layer of n neurons with sigmoid activation, n weights c_i, and a final bias ϕ to produce output $N(x)$. Let the interval $[-1,1]$ be split up into n equal-size subintervals, with the labeled x_i denoting the endpoints of these intervals:

$$-1 = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = 1. \tag{3.10}$$

Now, let $w_i = k$ for all $i \in [n]$, and $\theta_i = kt_{i-1}$ where $t_i = \frac{1}{2}(x_i + x_{i+1})$ are the midpoints of the intervals. Furthermore, let $\phi = f(x_0)$ and $c_i = [f(x_i) - f(x_{i-1})]$.

Consider the following:

$$|f(x) - N(x)| = \left| f(x) - f(x_0) - \sum_{i=1}^{n} [f(x_i) - f(x_{i-1})] \sigma[k(x - t_{i-1})] \right| \tag{3.11}$$

$$= \left| f(x) - f(x_j) + \sum_{i=1}^{j} [f(x_i) - f(x_{i-1})] + f(x_0) - f(x_0) - \sum_{i=1}^{n} [f(x_i) - f(x_{i-1})] \sigma[k(x - t_{i-1})] \right|, \tag{3.12}$$

where we introduce a telescoping sum and j is such that $x_j \leq x < x_{j+1}$, or $j = n$ if $x = x_n = 1$. As a notational convention, summations with starting index greater than ending index are to be treated as an

empty sum. Continuing,

$$
\begin{aligned}
&|f(x)-N(x)| \\
&\quad= \left| f(x)-f(x_j)+\sum_{i=1}^{j}[f(x_i)-f(x_{i-1})][1-\sigma(k(x-t_{i-1}))]-\sum_{i=j+1}^{n}[f(x_i)-f(x_{i-1})]\sigma[k(x-t_{i-1})] \right| \\
&\quad\leq \left| f(x)-f(x_j) \right| + \left| \sum_{i=1}^{j}[f(x_i)-f(x_{i-1})][1-\sigma(k(x-t_{i-1}))] \right| \\
&\qquad\qquad + \left| \sum_{i=j+1}^{n}[f(x_i)-f(x_{i-1})]\sigma[k(x-t_{i-1})] \right| \\
&\quad\leq \rho(x-x_j)+\sum_{i=1}^{j}\rho\frac{2}{n}[1-\sigma(k(x-t_{i-1}))]+\sum_{i=j+1}^{n}\rho\frac{2}{n}\sigma(k(x-t_{i-1})), \qquad (3.13)
\end{aligned}
$$

where to obtain the first term we apply the ρ-Lipschitzness of f, and to obtain the second and third terms we apply Lipschitzness and use the fact that $x_i - x_{i-1} = 2/n$. The absolute values have been dropped because the remaining terms are all positive. Note that $x - x_j < 2/n$ by definition of index j, so we might as well make the inequality strict.

$$
\begin{aligned}
|f(x)-N(x)| &< \frac{2\rho}{n}+\frac{2\rho}{n}\sum_{i=1}^{j}[1-\sigma(k(x-t_{i-1}))]+\frac{2\rho}{n}\sum_{i=j+1}^{n}\sigma(k(x-t_{i-1})) \qquad (3.14) \\
&< \frac{2\rho}{n}+\frac{2\rho}{n}\sum_{i=1}^{j}\left[1-\sigma\left(\frac{k}{n}\right)\right]+\frac{2\rho}{n}+\frac{2\rho}{n}\sum_{i=j+2}^{n}\sigma\left(-\frac{k}{n}\right), \qquad (3.15)
\end{aligned}
$$

where to obtain the first sum we use the fact that $1-\sigma\left(\frac{k}{n}\right)$ is larger than any of the original terms, since we recall that

$$
x_{j-1} < t_{j-1} < x_j \leq x < x_{j+1}, \qquad (3.16)
$$

so x is at least half a subinterval away from t_{j-1}, i.e. $x - t_{j-1} \geq 1/n$. To obtain the second sum, we split off the $i = j+1$ term since $\left|x-t_j\right| \leq 1/n$, requiring the use of a general bound $\sigma(k(x-t_j)) < 1$. The remaining terms are bounded by $\sigma(-k/n)$ since all t_i with $i > j$ are more than $1/n$ larger than x.

This is a good time to restrict k with $k \geq n\log(n-1)$, which is equivalent to

$1-\sigma(k/n)=\sigma(-k/n)\leq 1/n$. Using this inequality and bounding the number of terms in the sums by n,

$$|f(x)-N(x)|<\frac{2\rho}{n}+\frac{2\rho}{n}\left(n\cdot\frac{1}{n}\right)+\frac{2\rho}{n}+\frac{2\rho}{n}\left(n\cdot\frac{1}{n}\right) \tag{3.17}$$

$$=\frac{8\rho}{n}. \tag{3.18}$$

If we then choose $n\geq\frac{8\rho}{\varepsilon}$, we are thus left with

$$|f(x)-N(x)|<\varepsilon. \tag{3.19}$$

We have shown that for $k\geq n\log(n-1)$ and $n\geq\frac{8\rho}{\varepsilon}$, the neural network with remaining weights given by

$$N(x)=f(x_0)+\sum_{i=1}^{n}[f(x_i)-f(x_{i-1})]\sigma[k(x-t_{i-1})] \tag{3.20}$$

satisfies $|f(x)-N(x)|<\varepsilon$ for $x\in[-1,1]$. □

In Eq. (3.20) the intuition is that as a base the neural network outputs the leftmost value of the function to be approximated, $f(x_0)=f(-1)$. The sum smoothly updates this value to the next reference value at $f(x_1)$ by using the sigmoid function as a weighting of $f(x_1)-f(x_0)$, allowing for the corresponding update to be "turned on" from zero, up to a maximum of 1. For values of x more central in the interval $[-1,1]$, enough updates to increase the network's output value to the nearest reference value $f(x_i)$, where $x_i\approx x$ are enabled to match $f(x)$, while later updates are still weighted by values near zero.

The network constructed above is certainly not the only network nor the most efficient one to approximate the desired function $f(x)$, but the proof also serves to demonstrate that neural networks can indeed be designed "by hand" to accomplish certain tasks through appropriate choices of weights and biases. Indeed, many proofs of neural network capabilities and asymptotic bounds involve piece-by-piece assembly of subnetworks that each have specially-crafted characteristics to facilitate analysis.

The universal approximation theorem proves that neural networks have the possibility to approximate nearly any function of interest. However, there are other classes of functions that demonstrate desirable convergence properties, like Taylor and Fourier series. Classical harmonic analysis concerns Fourier series convergence under various criteria and classes of functions, so a natural question that arises is whether Fourier series would be as suitable for machine learning tasks, if not more suitable, than neural networks.

Consider a depth-2 feedforward neural network with one hidden layer of size n and input dimension d

and output dimension 1:

$$N(\mathbf{x}) = v_0 + \sum_{k=1}^{n} v_k \alpha\left(\mathbf{w}_k^\intercal \mathbf{x} + b_k\right), \tag{3.21}$$

$\mathbf{x}, \mathbf{w}_k \in \mathbb{R}^d$, $v_k, b_k \in \mathbb{R}$. Compare this to the form of a d-dimensional Fourier series approximating a function $f(\mathbf{x})$ integrable on $[-\pi, \pi]^d$:

$$F(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^d} \hat{f}_{\mathbf{k}} e^{i\mathbf{k}^\intercal \mathbf{x}}, \tag{3.22}$$

where the Fourier coefficients are defined by

$$\hat{f}_{\mathbf{k}} \equiv (2\pi)^{-d} \int_{[-\pi,\pi]^d} f(\mathbf{y}) e^{-i\mathbf{k}^\intercal \mathbf{y}} \, \mathrm{d}\mathbf{y}. \tag{3.23}$$

Comparing Eqs. (3.21) and (3.22), we can see a conceptual similarity: both involve linear combinations of non-linear transformations of the input. However, the Fourier series requires the function $f(\mathbf{x})$ to form its approximation, while the neural network learns from samples $\{\mathbf{x}_i, f(\mathbf{x_i}) + \varepsilon_i\}$, where ε_i allows for the possibility of noise.

Gallant and White pursued this avenue and attempted to gain the properties of Fourier series while maintaining the trainability of neural networks, producing the first Fourier neural network[19]. They showed that using a "cosine squasher" activation,

$$\alpha_{\mathrm{GW}}(x) \equiv \begin{cases} 0, & x \in (-\infty, -\frac{\pi}{2}), \\ \frac{1}{2}\left(\cos\left(x + \frac{3\pi}{2}\right) + 1\right), & x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \\ 1, & x \in \left(\frac{\pi}{2}, \infty\right), \end{cases} \tag{3.24}$$

in Eq. (3.21) and a specific choice of "hardwired" (non-learnable) hidden layer weights $\mathbf{w}_k$ and b_k, a Fourier series approximation to $f(\mathbf{x})$ is obtained. During training, only the connections linking the hidden state to the output, v_0 and v_k, would need to be learned using standard neural network training methods such as backpropagation. The paper proved that this depth-2 network would be at least as good or better than a Fourier series approximation.

However, more recently, Zhumekenov *et al.* have shown that simply using a sigmoid $\sigma(x) = 1/(1+e^{-x})$ activation in Eq. (3.21) empirically outperforms Gallant and White's Fourier neural network in minimizing approximation error of an arbitrary function for a given hidden layer size, suggesting that there may not be

much to gain from emulating truncated Fourier series approximations in general learning tasks[91].

3.3.2 Deep Networks

So far we have only discussed the representation capabilities of depth-2 networks, which we see can already approximate any function given a wide enough hidden layer. However, this only scratches the surface of possible neural network architectures, and it turns out that additional CAP depth adds exponentially more expressive power. That is, deep neural networks can achieve the same level of performance as shallow networks while employing exponentially fewer parameters.

Lu *et al.* showed that the universal approximation theorem holds in *width-bounded* networks as well[43]. In the previous section the classical results have been *depth-bounded* in the sense that the networks were restricted to depth-2 while the width of the hidden layer was used to decrease approximation error to the required tolerance. The result by Lu *et al.* instead demonstrates that a *width-bounded* network can approximate an arbitrary continuous function given a sufficient depth. In particular, they proved that width-$(n+4)$ ReLU networks, where n is the input dimension, are universal approximators[43].

A proof by Eldan and Shamir showed that a simple function expressible by a small 3-layer (2 hidden layers and one output) network cannot be approximated to better than constant accuracy by any 2-layer network, for virtually any activation function, unless the number of neurons in its hidden layer is exponentially more than the neurons per hidden layer in the depth-3 network[16]. Telgarsky presented a similar idea regarding deeper networks: for any positive integer k, there exist neural networks with $\Theta(k^3)$ layers, $\Theta(1)$ nodes per layer, and $\Theta(1)$ distinct parameters which can not be approximated by networks with $\mathcal{O}(k)$ layers unless they are exponentially large — they must possess $\Omega(2^k)$ nodes[76]. These results, as well as an overwhelming amount of empirical evidence from neural network applications, suggest that deep networks confer a substantial expressivity gain per parameter compared to their shallow counterparts. Considering that the shallow networks discussed in the previous section were already on par with standard parameterized families of functions like truncated Fourier series expansions, it appears that deep neural networks are amongst the most expressive known forms.

The intuition for why deep networks can be very effective for learning tasks lies in their ability to learn high-level abstraction. In a manner similar to how a change of coordinate system can substantially simplify the expression of the dynamics of a physical system, nonlinear transformations of the inputs to a neural network, often referred to as *features*, result in new features that can have a simpler relationship to the target output. In the past, feature extraction preprocessing was mostly crafted by hand, using intuitive or

empirically-determined transformations to capture what was expected to be the most essential information. Deep neural networks learn effective feature representations by optimizing feature extraction and the task performance simultaneously, with earlier layers serving as the built-in preprocessing step[67]. For example, in convolutional neural networks[39], an architecture frequently used for image classification and understanding tasks, the earliest layers learn to detect simple features like edges, colors, and gradients, while deeper layers detect more complex visual features like faces, wheels, and windows. This is achieved through a hierarchy of features; complex shapes consist of patterns of basic shapes, which themselves consist of edges, fills and gradients.

3.4 LEARNING

A neural network learns through an algorithm which processes training data, which in the case of regression takes the form $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, and adjusts the weights based on the information contained. Although the goal of regression is to minimize the *true error*, which is chosen to be some measure of the difference between the learned function $\hat{\mathbf{f}}(\mathbf{x})$ and true function $\mathbf{f}(\mathbf{x})$ across $\mathbf{x} \in \mathcal{D}$, unlike a Fourier series or Taylor expansion determination we do not have access to the true function $\mathbf{f}$, just a sampling of it at certain points as specified in the training set. The dominant approach for using the given information while striving to minimize true error is to minimize the *training error*, which is chosen to be a measure of the difference between $\hat{\mathbf{f}}(\mathbf{x}_i)$ and $\mathbf{y}_i$ for all the pairs $(\mathbf{x}_i, \mathbf{y}_i)$ in the training set. This process is called *empirical risk minimization*, or ERM for short[68].

There are potential drawbacks to fully minimizing the empirical risk. If the model has sufficient complexity, it is generally possible for the model to interpolate the data points, passing through them exactly. As the data points typically have some inherent error, this is generally not a desired outcome, potentially leading to large *generalization error* when the model is compared to additional test samples from $\hat{\mathbf{f}}(\mathbf{x}) + \boldsymbol{\epsilon}$. This sort of failure is referred to as *overfitting*, where the approximating function adapts too much to the unlearnable stochastic noise of the training data, inevitably leading to worse performance on test samples due to the extra deviations of $\hat{\mathbf{f}}$ compared to the true function $\mathbf{f}$. Overfitting can be avoided by reducing the complexity of a model, which typically means choosing a model with fewer tunable parameters. This smaller family of parameterized approximating functions will be limited in how well they can minimize the empirical risk, and less of the model's adaptability will be used to capture the unimportant fluctuations introduced by the aleatoric error. If the model is made too simple, it may not have the ability to emulate the features of the true function, like if a linear regression were used on data stemming from a

quadratic process. The choice of constraint on a family of hypothesis functions is referred to as inductive bias, and as the hypothesis class becomes "richer" in the types of functions it can emulate, the bias is reduced due to the decreased assumptions on what the true function should look like. Choosing the right model is a balance of simultaneously minimizing the error introduced by inductive bias and the error due to overfitting, typically referred to in this context as *variance*. Variance refers to the sensitivity of the trained model to the aleatoric errors in the training set. The idea is that if the same points $\mathbf{x}_i$ were sampled again, the corresponding outputs $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\epsilon}_i$ would have a different error, which would affect the fit. This balancing of errors is referred to as the *bias-variance* or *bias-complexity* tradeoff[20 68].

Conventional wisdom dictates that a model should have fewer free parameters than the number of data points; if this isn't the case, the model becomes overparameterized and runs the risk of overfitting[81]. For example, in polynomial regression, a degree p polynomial can exactly fit $p+1$ data points using its $p+1$ free parameters. As seen in Section 3.3, neural networks are very expressive and as such can be susceptible to overfitting. However, due to the way they are trained, as will be seen in Section 3.4.2, overfitting can be avoided by *early stopping*, the termination of the learning algorithm before full convergence on the training data is complete. In addition, recent studies have shown that, somewhat counterintuitively, both bias and variance can be decreased by increasing number of neurons in a network, contradicting the bias-variance tradeoff[54]. The unexpected high generalization performance of neural networks despite their very complex hypothesis class, which is often sufficiently parameterized to memorize the training set, is a welcome surprise, though not well explained by the field's current understanding of generalization[89].

3.4.1 Loss Function

To minimize empirical risk, we minimize a *loss function*, which is either the empirical risk itself or a *surrogate loss function* which acts as a proxy for the empirical risk but has nicer properties that aid in minimization. For example, if the task is classification into two classes, the empirical risk is the one minus the classification accuracy (classification risk) of the model. It makes intuitive sense that we would want to train our model to obtain the lowest risk possible on the training data, zero, which is when every input is correctly classified. However, with a complex model like a neural network, it is not easy to know how to tune the parameters such that this result is achieved. The most common method to train these models is by performing gradient descent, which requires derivative information from the function to be minimized. In the present classification task, it is not possible to perform gradient descent on the risk directly. When in a region where any small change in the weights does not cause a difference in classification, the gradient is

zero, and when a small change in weights will cause a training point to be classified differently, the gradient is undefined due to the discontinuity in the risk function. In addition, there are many regions of weight-space that can result in perfect classification on the training set (zero empirical risk), but these solutions are not all equivalent in terms of generalizability. For example, we usually want the network to be most certain about its classification if an unseen input is very close to one encountered in training. An arbitrary selection of weights that happens to classify all the training data correctly could potentially classify similar inputs differently just because a decision boundary lies too close to one of the training points. These issues rule out using empirical risk directly as our loss function. Instead, we use a surrogate loss function that can be differentiated, and does favor decision boundaries that maximize the likelihood of the training data. A commonly-used loss for binary classification is the binary cross-entropy loss:

$$L_{\text{cross-ent}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}(\mathbf{x}_n;\mathbf{w}) + (1-y_n)\log(1-\hat{y}(\mathbf{x}_n;\mathbf{w}))\right], \tag{3.25}$$

where the training data is given by $\{(\mathbf{x}_n, y_n)\}$, $y_n = 0, 1$, and the output of the neural network with weights $\mathbf{w}$ for input $\mathbf{x}_n$ is a probability $\hat{y}(\mathbf{x}_n;\mathbf{w}) \in [0,1]$. It is easy to show that this is the expression of the negative log-likelihood of observing the training data[77] given the probability $\hat{y}$ that the input $\mathbf{x}$ is in class 1. As a result, minimizing the cross-entropy loss gives the maximum likelihood estimate (MLE) for the network weights given the training data.

For regression tasks, which we focus on in this work, a common loss function is the mean squared error (MSE) loss,

$$L_{\text{MSE}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left[\mathbf{y}_n - \hat{\mathbf{y}}(\mathbf{x}_n;\mathbf{w})\right]^2. \tag{3.26}$$

This loss is differentiable in $\mathbf{w}$ provided that the neural network $\hat{y}$ uses differentiable activation functions.

The process of minimizing the empirical risk, whether doing so directly or through a surrogate loss function, is called *training* a neural network.

3.4.1.1 NON-CONVEXITY

In optimization tasks convexity is a very important property as it greatly simplifies the determination of global optima. Here we will briefly review the definition of convexity, some of its consequences, and its relevance to loss functions and neural networks.

A subset C of a vector space is *convex* if for all x and y in C, the line segment connecting x and y is

included in C. That is, $(1-t)x+ty \in C$ for all $x, y \in C$ and $t \in [0,1]$. Intuitively, the set cannot have any holes within it, nor any indents of its boundary. All of its boundaries must always be straight or curving inwards. A convex function is defined on a convex set, $f : C \to \mathbb{R}$, and it satisfies

$$\forall x_1, x_2 \in C, \quad \forall t \in [0,1] : \qquad f\big(tx_1 - (1-t)x_2\big) \leq tf(x_1) + (1-t)f(x_2). \tag{3.27}$$

In the inequality, the right-hand side of the equation describes the secant line passing through the points $\big(x_1, f(x_1)\big)$ and $\big(x_2, f(x_2)\big)$, while the left-hand side describes the segment of the curve lying between these two points. The condition of convexity simply requires that any segment of the function's curve not rise above the corresponding secant. This constrains the function to never curve downwards. Without downward curves, it is not possible for the function to have local minima that are different in value than the global minimum, and if the global minimum is attained for multiple points, they all form a convex set themselves (and hence are connected). Convexity is a very convenient property to have in a loss function since it guarantees the convergence of a gradient descent algorithm to a global minimum.

Unfortunately, loss functions on neural networks are generally non-convex. The universal approximation property of neural networks allow them to approximate any continuous function, which incidentally allows for a multitude of locally-optimal solutions in the loss function. Despite this limitation, many training convergence proofs in machine learning assume convexity[68]. In practice, it is common to obtain models with differing performance when weights are randomly initialized, indicating that the global minimum is not easy to obtain. However, the general consensus in the field that truly "bad" local minima are infrequently encountered, though they do exist[13,73].

The main concern when training neural networks is not the encountering of local minima, but rather the abundance of saddle points in high-dimensional spaces[11,55], the other feature that can appear in non-convex problems. As discussed in Pascanu *et al.*[55], an easy way to see this is by thinking about the Hessian matrix of a high-dimensional function. The Hessian, being a symmetric matrix, can always be diagonalized with real eigenvalues. A critical point is a maximum when all the eigenvalues are negative, a minimum when all the eigenvalues are positive, and a saddle point if there is a mix of positive and negative eigenvalues. For an arbitrary Hessian evaluated at a critical point, we expect the probability of an eigenvalue being positive to be $1/2$ (Wigner's semi-circular law[88]), so for a d dimensional function, the probability of the critical point being a maximum or a minimum is $2(1/2)^d$, which becomes vanishingly small as d gets large. In contrast, the probability that the critical point is a saddle point goes to 1. The large number of weights in neural networks make them very susceptible to saddle points, which can be problematic for gradient descent

methods which often crawl to unbearably slow speeds in the vicinity of these loss landscape features. As discussed in the next section, many modern gradient descent methods have been developed specifically to deal with the saddle point problem.

3.4.2 Stochastic Gradient Descent

The most common methods for minimizing loss functions are forms of *stochastic gradient descent* (SGD). SGD is closely related to the method of steepest descent except instead of using the gradient of the exact function to be minimized to take a step, we use a vector whose expectation value equals the gradient. This extra stochasticity results in a meandering path towards a local minimum, but it can be more practical for calculations involving a lot of data or very complex networks. In addition, the stochasticity of SGD can occasionally help it "jump" through barriers in the loss landscape that would otherwise keep the network from finding a better minimum.

Given a loss function $L(\mathbf{w})$ that depends on the weights $\mathbf{w}$ of the neural network, standard gradient descent (steepest descent) involves updating the weights according to

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla L\Big(\mathbf{w}^{(t)}\Big), \tag{3.28}$$

where $\mathbf{w}^{(t)}$ are the weights at training iteration t, and η is a positive number called the *learning rate*. This iterative process is typically continued until the decrease in the loss function is below a tolerance, or until the neural network performs sufficiently well on validation data. Figure 3.4 shows a simple visualization of gradient descent for a loss function dependent on a single weight.

Stochastic gradient descent is very similar to gradient descent. The update rule instead takes the form:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t, \tag{3.29}$$

where $\mathbf{v}_t$ satisfies the condition

$$\mathbb{E}[\mathbf{v}_t | \mathbf{w}^{(t)}] = \nabla L\Big(\mathbf{w}^{(t)}\Big). \tag{3.30}$$

Typically, this vector $\mathbf{v}_t$ is obtained by computing the gradient of a "partial" loss function on a random subset of the training data, called a *minibatch* (or often just batch). To be specific, suppose that the loss function is the average of loss functions $\ell(\mathbf{w}; \mathbf{x}_i, \mathbf{y}_i)$ defined for individual data points, which is often the

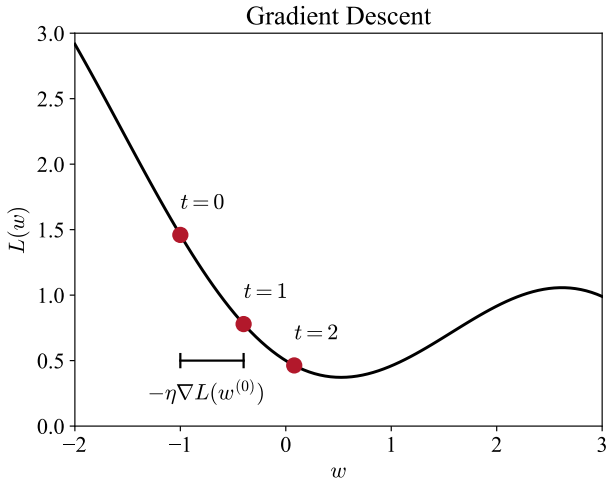

Figure 3.4: A schematic showing the weight updates of a one-dimensional loss function.

case. Then, the total loss function on all of the data $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ is given by

$$L(\mathbf{w}; \{(\mathbf{x}_i, \mathbf{y}_i)\}) = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{w}; \mathbf{x}_i, \mathbf{y}_i). \tag{3.31}$$

If we were to apply gradient descent directly on this loss function, before a single weight update can be made, the gradients of all the individual losses on each point would have to be computed. This can be very slow and it's not very efficient—perfect knowledge of the gradient at a particular $\mathbf{w}^{(t)}$ is not necessary for every step, especially if the network is still very far from minimizing the loss. Consider a minibatch B which contains a subset of the total training dataset, $B \subset \{(\mathbf{x}_i, \mathbf{y}_i)\}$. It has a corresponding batch loss

$$L(\mathbf{w}; B) = \frac{1}{|B|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in B} \ell(\mathbf{w}; \mathbf{x}_i, \mathbf{y}_i). \tag{3.32}$$

The batch size $|B|$ can range from 1 (a single data point) and N, the size of the entire training dataset. It is clear that the gradient of the batch loss satisfies the property

$$\mathbb{E}[\nabla L(\mathbf{w}^{(t)}; B) | \mathbf{w}^{(t)}] = \nabla L(\mathbf{w}; \{(\mathbf{x}_i, \mathbf{y}_i)\}), \tag{3.33}$$

since the gradient of the batch loss is simply a sample of the terms in the gradient of the total loss. There is some stochasticity introduced however, since at each step the batch loss is approximating the total loss function based on the subset of data in its batch. The batch loss is much faster to compute, especially if all

the individual contributions to the loss can be computed simultaneously on a parallel computing architecture, like a GPU.

The learning rate η appearing in the SGD update Eq. (3.29) is important to the training process as it is closely related to minimizing the loss function. As it controls the step size in the weight updates, when it is too small the training process takes a very long time, but when it is too large, SGD might diverge or be unable to descend into a local minimum due to repeatedly overstepping it. The learning rate also does not have to be a fixed quantity throughout training. It is common to shrink the learning rate as training progresses to allow the weights to better descend into the well of a minimum. In addition, the learning rate can be adjusted on the fly depending on how the minimization is going, which is the basis of many modern variations of the SGD method. The learning rate can also be changed on a per-weight basis depending on the size of the derivatives with respect to each weight. Adagrad[15] and Adam[30] use these concepts to allow for larger steps in flatter directions while taking smaller steps in directions with large derivatives. This can be very helpful for escaping saddle points in the appropriate direction when some directions are fairly flat while others form the steep sides of a valley.

3.5 Automatic Differentiation and Backpropagation

In order to make use of gradient descent for training neural networks, we will have to differentiate the network with respect to its weights. Intriguingly, differentiation of neural networks is fairly straightforward and it does not cost much more to perform than the evaluation of the network for a particular input (called the *forward pass*).

Automatic differentiation is the general set of techniques that allow for numerically evaluating the derivative of a function specified by a computer program. In composing the output of the function, if the final value is the result of a graph of elementary differentiable operations, then by employing the chain rule the value of the derivative at the input can be computed. Modern neural network packages like TensorFlow and PyTorch are built around automatic differentiation, building symbolic computation graphs from the steps used to construct each variable by default. In the *backward pass*, the graph of the forward pass can be used to construct *adjoint nodes* for each of the original (primal) nodes of the computation graph using the chain rule, and the output's derivative with respect to each input node is accumulated into each input node. The computation of the derivatives of the output is a reversed traversal of the computational graph that was used to assemble the output, as will be made clear shortly.

Backpropagation[87] is a specific case of automatic differentiation when it is applied to computing the

derivatives of neural networks. To demonstrate the mathematics of backpropagation, we will apply it to computing the gradient of a multilayer perceptron, which has the form explained in Section 3.2.1,

$$a_i^\ell(\mathbf{x}) = \sum_{j=1}^{n^{\ell-1}} w_{ij}^\ell o_j^{\ell-1}(\mathbf{x}) + b_i^\ell, \qquad \ell \in [\mathcal{L}], \tag{3.34}$$

$$o_i^\ell(\mathbf{x}) = \alpha^\ell\Big(a_i^\ell(\mathbf{x})\Big) = \alpha^\ell\left(\sum_{j=1}^{n^{\ell-1}} w_{ij}^\ell o_j^{\ell-1}(\mathbf{x}) + b_i^\ell\right), \tag{3.35}$$

where $a_i^\ell(\mathbf{x})$ is the output of neuron i in layer ℓ before the activation is applied, and a loss given by a function of the network outputs $L(\{o_i^\mathcal{L}\})$. First we will consider a few derivatives which will help us build up to the gradient of the loss. Letting $\partial\alpha^\ell$ represent the derivative of the activation function $\alpha^\ell(a)$ and using the chain rule,

$$\frac{\partial o_h^\ell}{\partial w_{ij}^\ell} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_h^\ell}\frac{\partial a_h^\ell}{\partial w_{ij}^\ell} = \begin{cases} \partial\alpha^\ell o_j^{\ell-1} & h = i \\ 0 & h \neq i \end{cases}, \qquad \ell \in \{1,\dots,\mathcal{L}\}. \tag{3.36}$$

We can use this result to iteratively calculate the following derivatives:

$$\frac{\partial o_h^\ell}{\partial w_{ij}^{\ell-1}} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_h^\ell}\frac{\partial a_h^\ell}{\partial w_{ij}^{\ell-1}} = \partial\alpha^\ell w_{hi}^\ell \frac{\partial o_i^{\ell-1}}{\partial w_{ij}^{\ell-1}} = \partial\alpha^\ell w_{hi}^\ell \partial\alpha^{\ell-1} o_j^{\ell-2} \qquad \ell \in \{2,\dots,\mathcal{L}\} \tag{3.37}$$

$$\frac{\partial o_h^\ell}{\partial w_{ij}^{\ell-2}} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_h^\ell}\frac{\partial a_h^\ell}{\partial w_{ij}^{\ell-2}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \frac{\partial o_k^{\ell-1}}{\partial w_{ij}^{\ell-2}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \partial\alpha^{\ell-1} w_{ki}^{\ell-1} \partial\alpha^{\ell-2} o_j^{\ell-3} \qquad \ell \in \{3,\dots,\mathcal{L}\} \tag{3.38}$$

$$\frac{\partial o_h^\ell}{\partial w_{ij}^{\ell-3}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \frac{\partial o_k^{\ell-1}}{\partial w_{ij}^{\ell-3}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \partial\alpha^{\ell-1} \sum_{m=1}^{n^{\ell-2}} w_{km}^{\ell-1} \partial\alpha^{\ell-2} w_{mi}^{\ell-2} \partial\alpha^{\ell-3} o_j^{\ell-4} \qquad \ell \in \{4,\dots,\mathcal{L}\}, \tag{3.39}$$

after which the pattern becomes clear. Similarly, for the derivatives with respect to the biases, we have

$$\frac{\partial o_h^\ell}{\partial b_i^\ell} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_i^\ell}\frac{\partial a_i^\ell}{\partial b_i^\ell} = \begin{cases} \partial\alpha^\ell & h = i \\ 0 & h \neq i \end{cases}, \qquad \ell \in \{1,\dots,\mathcal{L}\} \tag{3.40}$$

(3.41)

$$\frac{\partial o_h^\ell}{\partial b_i^{\ell-1}} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_h^\ell}\frac{\partial a_h^\ell}{\partial b_i^{\ell-1}} = \partial\alpha^\ell w_{hi}^\ell \partial\alpha^{\ell-1} \qquad \ell \in \{2,\dots,\mathcal{L}\} \tag{3.42}$$

$$\frac{\partial o_h^\ell}{\partial b_i^{\ell-2}} = \frac{\mathrm{d}\alpha^\ell}{\mathrm{d}a_h^\ell}\frac{\partial a_h^\ell}{\partial b_i^{\ell-2}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \frac{\partial o_k^{\ell-1}}{\partial b_i^{\ell-2}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \partial\alpha^{\ell-1} w_{ki}^{\ell-1} \partial\alpha^{\ell-2} \qquad \ell \in \{3,\dots,\mathcal{L}\} \tag{3.43}$$

$$\frac{\partial o_h^\ell}{\partial b_i^{\ell-3}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \frac{\partial o_k^{\ell-1}}{\partial b_i^{\ell-3}} = \partial\alpha^\ell \sum_{k=1}^{n^{\ell-1}} w_{hk}^\ell \partial\alpha^{\ell-1} \sum_{m=1}^{n^{\ell-2}} w_{km}^{\ell-1} \partial\alpha^{\ell-2} w_{mi}^{\ell-2} \partial\alpha^{\ell-3} \qquad \ell \in \{4,\dots,\mathcal{L}\}. \tag{3.44}$$

So, to compute the gradient of the loss L, we simply make use of the chain rule again,

$$\frac{\partial L}{\partial w_{ij}^\ell} = \sum_{h=1}^{n^\mathcal{L}} \frac{\partial L}{\partial o_h^\mathcal{L}} \frac{\partial o_h^\mathcal{L}}{\partial w_{ij}^\ell}, \qquad \ell \in [\mathcal{L}], \tag{3.45}$$

$$\frac{\partial L}{\partial b_i^\ell} = \sum_{h=1}^{n^\mathcal{L}} \frac{\partial L}{\partial o_h^\mathcal{L}} \frac{\partial o_h^\mathcal{L}}{\partial b_i^\ell}, \qquad \ell \in [\mathcal{L}]. \tag{3.46}$$

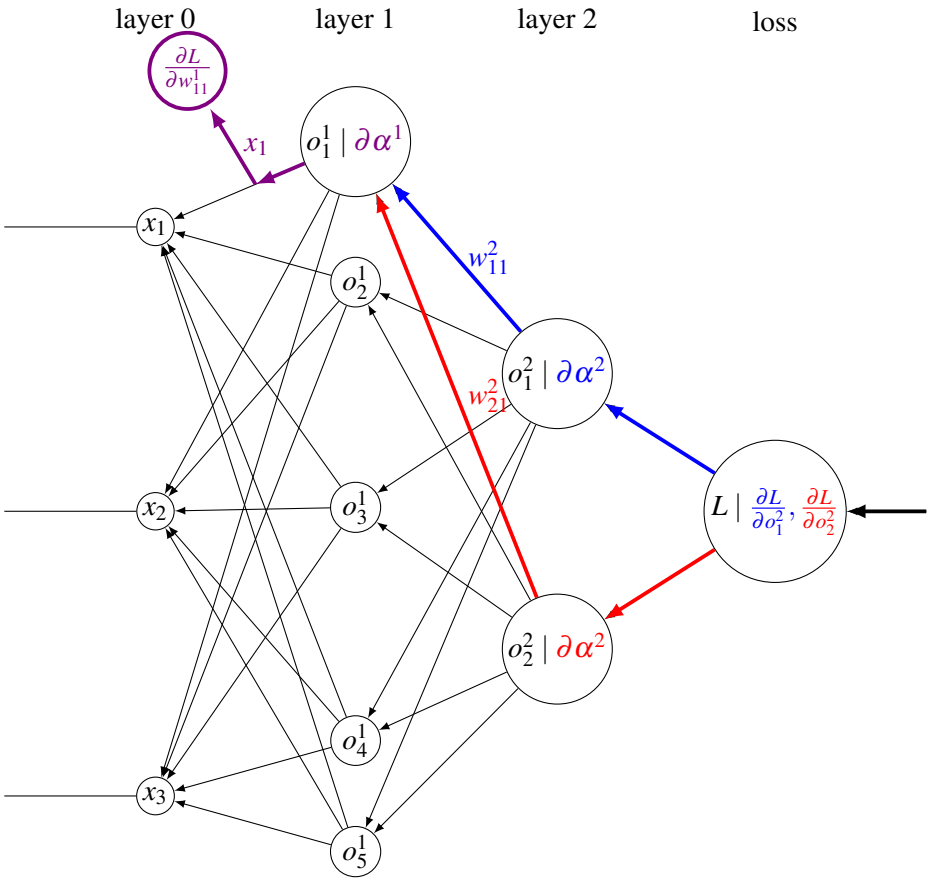

Figure 3.5: Backpropagation example: calculation of $\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial o_1^2}\partial\alpha^2 w_{11}^2 \partial\alpha^1 x_1 + \frac{\partial L}{\partial o_2^2}\partial\alpha^2 w_{21}^2 \partial\alpha^1 x_1$.

The above equations can be interpreted graphically, using the same graph that represents the neural network but with some of the operations modified. As a specific example, consider the network shown in Figure 3.5 and the calculation of $\frac{\partial L}{\partial w_{11}^1}$:

$$\frac{\partial L(\mathbf{x};\,\mathbf{w})}{\partial w_{11}^1} = \frac{\partial L}{\partial o_1^2}\frac{\partial \alpha^2(a_1^2)}{\partial a_1^2} w_{11}^2 \frac{\partial \alpha^1(a_1^1)}{\partial a_1^1} x_1 + \frac{\partial L}{\partial o_2^2}\frac{\partial \alpha^2(a_2^2)}{\partial a_2^2} w_{21}^2 \frac{\partial \alpha^1(a_1^1)}{\partial a_1^1} x_1, \tag{3.47}$$

where all the derivatives are evaluated at the present input $\mathbf{x}$ and weights $\mathbf{w}$. Each term in the sum represents a different path along the backward-directed graph shown in Figure 3.5 with each edge contributing a multiplicative value dictated by the corresponding weight, and each node multiplying its input by the value of the derivative of its activation function evaluated at the original pre-activation output $a_i^\ell(\mathbf{x})$ of the forward pass. When the paths converge to the leaf node $\frac{\partial L(\mathbf{x};\,\mathbf{w})}{\partial w_{11}^1}$, their values are summed. In other words, in this backward pass graph, the derivatives are *accumulated* at the leaf nodes, with each leaf corresponding to the derivative of the loss with respect to the trainable parameter located at the same position in the forward pass graph.

The training of deep neural networks was made possible by this convenient manner of evaluating the gradient of the loss without requiring finite difference calculations for every weight. However, automatic differentiation is not limited to helping with stochastic gradient descent. The same principles can be applied to any input of the neural network, allowing for exact evaluation of derivatives of the network with respect to $\mathbf{x}$, a trick we make use of in the next chapter for finding solutions to differential equations.

4

Solving Ordinary Differential Equations with Neural Networks

This chapter is part of a manuscript being prepared for publication:

> **Cedric Flamant**, David Sondak, Pavlos Protopapas. "Solving Ordinary Differential Equations Using Neural Network Solution Bundles." *in preparation*.

Abstract

The time evolution of dynamical systems is frequently described by ordinary differential equations (ODEs), which must be solved for given initial conditions. Most standard approaches numerically integrate the ODEs, producing a solution whose values are computed at discrete times. For every set of initial conditions and system parameters, the calculation has to be repeated from scratch, adding significant computational overhead to methods which require varied solutions to the ODE. We extend the Lagaris method of creating an approximating neural network solution to a set of differential equations, proposing that a neural network be used as a solution bundle, a collection of solutions to an ODE for various initial states and system parameters. The neural network solution bundle is trained with an unsupervised loss that does not require any prior knowledge of the sought solutions, and the resulting object is differentiable in initial conditions and system parameters. The solution bundle exhibits fast, parallelizable evaluation of the system state, facilitating the use of Bayesian inference for parameter or trajectory estimation in real dynamical systems.

4.1 Introduction

Many dynamical systems are described by ordinary differential equations (ODEs) which relate the rates and values of state variables and external driving functions. While some simple ODEs have closed form solutions to them, like the exponential function for radioactive decay, sines and cosines for harmonic oscillators, and logistic function for population growth, the vast majority have to be solved approximately

using discretization of the domain or by optimizing a parameterized trial solution. The former approximating methods are more common, with Runge-Kutta and multi-step methods as typical examples. These methods seek to numerically integrate the ODEs, starting from initial conditions and stepping forward until the desired final time is attained. While these conventional methods are typically efficient for determining the state of a system for a sequence of times, if we are only interested in the state at a specific later time, substantial computational effort must still be expended determining all the states at steps leading up to the state of interest. This causal order also limits parallelizability of the conventional single- and multi-step methods since the task cannot be parallelized in time—until the preceeding step is known, processors tasked with finding a segment of the system's evolution over a time interval cannot start calculating the correct piece of the trajectory. In addition, these discrete methods do not produce solutions that are directly differentiable since they return a sequence of points approximating the solution evaluated at each timeslice that was stepped through in the calculation.

In 1998 Lagaris *et al.* proposed a method of using neural networks as trial solutions, which could then be trained via a loss that approaches zero as the trial solution neared satisfaction of the differential equations at selected collocation points[38]. This approach is entirely unsupervised, requiring no prior knowledge of the solution sought. By simply constructing the loss from the differential equation of interest, training a neural network constrained to satisfy the boundary conditions would result in it becoming an approximation of the true solution, making use of the universal approximation property of neural networks. With the trained neural network, the approximate value of the solution at any point within the training range can be computed in constant time, without having to compute previous states first. This approach also has the ability to parallelize in time, allowing it to make use of the increasing parallelization of processors, the dominant direction of improvements in computational power in the present. The Lagaris method does have some limitations however. For different sets of initial conditions, or for different sets of parameters within a given differential equation, the network has to be retrained on the new task. This drawback entails a less-favorable consideration of the resources used during training the network since it has to be directly compared to the resources used for finding the solution using conventional means.

We propose an extension of the Lagaris method where the neural network is taught a variety of solutions to a parameterized differential equation. This increases the reusability of the trained network and also can speed up tasks that require knowing many solutions to a differential equation, such as in the Bayesian inference of parameters, or for propagating a distribution of uncertainty in a dynamical system. While it is straightforward to extend our approach to all the situations considered in the Lagaris paper, *i.e* for problems containing various types of boundary conditions, for partial differential equations, and higher derivatives,

here we will focus on initial value problems in first-order ordinary differential equations. We show that our method has promise when applied to a variety of tasks requiring the knowledge, and quick, parallel evaluation, of multiple solutions to an ODE, and where it could be useful to be able to differentiate the state at a particular time with respect to the initial condition or ODE parameters.

With the rapid advances in neural network development, as well as its supporting hardware, employing this method will become cheaper and more efficient, further extending its applicability in the future.

4.2 Solution Bundles

The Lagaris method of solving differential equations results in a single solution. A neural network is trained with a fixed set of boundary or initial conditions, and the entire capability of the network is used to best approximate the specific solution satisfying the differential equations. However, when working with a dynamical system it is common to require multiple solutions corresponding to different initial conditions in order to procure alternate trajectories. In addition, when an ODE is parameterized, say by a physical constant whose value has an associated uncertainty, it can be useful to have different solutions for various values of the parameters. These needs motivate an extension of the Lagaris method where we have the neural network adapt to a *solution bundle* where the "length" of the bundle extends in time and the "cross-section" of the bundle can extend in a subset of initial condition and parameter space, as will be made clear in the following sections.

4.2.1 Method Description

Consider the following general first-order differential equation parameterized by $\boldsymbol{\theta}$:

$$\mathbf{G}\left(\mathbf{x}, \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}, t\,;\boldsymbol{\theta}\right) = 0, \tag{4.1}$$

where in a dynamical system $\mathbf{x} \in \mathbb{R}^n$ is a vector of state variables, t is time, and $\boldsymbol{\theta} \in \mathbb{R}^p$ are physical parameters associated with the dynamics. We assume that the ODE describes a deterministic system where initial conditions $\mathbf{x}_0$ of the state variables uniquely determines a solution. We call the solutions to Eq. (4.1) over time range $[t_0, t_f]$, a subset $X_0 \subset \mathbb{R}^n$ of initial conditions $\mathbf{x}_0$, and a set $\Theta \subset \mathbb{R}^p$ of parameters $\boldsymbol{\theta}$, which together define a multivariate function, a *solution bundle* $\mathbf{x}(t; \mathbf{x}_0, \boldsymbol{\theta})$.

Let the approximating function to Eq. (4.1) for the solution bundle over (X_0, Θ) be given by

$$\hat{\mathbf{x}}(t\,;\mathbf{x}_0,\boldsymbol{\theta}) = \mathbf{x}_0 + a(t)\mathbf{N}(t\,;\mathbf{x}_0,\boldsymbol{\theta}\,;\mathbf{w}), \tag{4.2}$$

where $\mathbf{N} : \mathbb{R}^{n+p+1} \to \mathbb{R}^n$ is a neural network with weights $\mathbf{w}$, and $a : [t_0, t_f] \to \mathbb{R}$ satisfies $a(t_0) = 0$. This form explicitly constrains the trial solution to satisfy the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. The choice of $a(t)$ can affect the ease of training the network. While $a(t) = t - t_0$ is sufficient, Mattheakis *et al.* demonstrated that $a(t) = 1 - e^{-(t-t_0)}$ results in better convergence due to its upper bound fixing the scale for $\mathbf{N}(t\,;\mathbf{x}_0,\boldsymbol{\theta})$[46]. We primarily use multilayer fully-connected neural networks in our experiments, but improvements from minor deviations in the network structure make it clear that it is worth exploring other architectures in the future.

The unsupervised loss function used in training has the form

$$L = \frac{1}{|B|} \sum_{(t_i,\mathbf{x}_{0i},\boldsymbol{\theta}_i)\in B} b(t_i) \left| \mathbf{G}\left(\hat{\mathbf{x}}(t_i\,;\mathbf{x}_{0i},\boldsymbol{\theta}_i), \frac{\partial \hat{\mathbf{x}}(t_i\,;\mathbf{x}_{0i},\boldsymbol{\theta}_i)}{\partial t}, t_i\,;\boldsymbol{\theta}_i \right) \right|^2, \tag{4.3}$$

where the set $B = \{(t_i, \mathbf{x}_{0i}, \boldsymbol{\theta}_i)\}$ constitutes a training batch of size $|B|$, with $t_i \in [t_0, t_f]$, $\mathbf{x}_{0i} \in X_0$, and $\boldsymbol{\theta}_i \in \Theta$ drawn from some distribution over their respective spaces. The function $b : [t_0, t_f] \to \mathbb{R}$ appearing in Eq. (4.3) is used to weight data points based on their time, as will be discussed later. We typically use $b(t) = \exp(-\varepsilon(t - t_0))$. We found that uniform sampling over the spaces $[t_0, t_f]$, X_0, and Θ usually works well in practice, but there are situations where it is helpful to use a different distribution to generate batches. For example, for the FitzHugh-Nagumo model discussed in Section 4.4.3, a batch-number-dependent distribution over times $[t_0, t_f]$ was used for curriculum learning. In the loss function, the time derivative of $\hat{\mathbf{x}}(t)$ is computed using automatic differentiation, a common feature of deep learning libraries. Employing a gradient descent method for every batch and exactly computing the gradient of L with respect to the weights $\mathbf{w}$ with backpropagation, the trial function Eq. (4.2) will incrementally improve its approximation of the solution bundle.

There is no concept of an epoch in the training process of this method since every batch will be a unique sample of size $|B|$ from the distribution across times, initial conditions, and parameters. As such, the model cannot overfit as we are effectively operating in a regime of infinite data. The training ends when the approximation to the solution bundle is deemed acceptable, based on either the history of past losses or on some other metric, like its difference compared to a few solution curves computed via conventional finite difference methods. In principle, the training would naturally plateau when a minimum of the loss is attained which, loosely speaking, is when the capacity of the network is saturated and the majority of

weights are locally optimal. To obtain incrementally better solution bundles, one could increase the complexity of the network since the universal approximation theorem (Section 3.3) guarantees the existence of a better approximating network to the true solution bundle. In principle, arbitrarily high accuracy can be achieved this way.

At the end of training, the neural network solution bundle $\hat{\mathbf{x}}(t;\mathbf{x}_0,\boldsymbol{\theta})$ can be used to evaluate, in constant execution time, the approximate value of state $\mathbf{x}$ at any time $t \in [t_0,t_f]$, for any initial condition $\mathbf{x}_0 \in X_0$ given assumed differential equation parameters $\boldsymbol{\theta} \in \Theta$. This is to be compared with conventional finite-difference methods which require execution time linearly scaling with the desired t due to the necessary forward-stepping from the initial condition to time t. So, the evaluation time of the neural network can be fairly compared to the time it would take a numerical integrator to step from $t = 0$ to $t = t_f$; it does not have to beat the evaluation time of a single step of a Runge-Kutta or multi-step method, which often involves comparatively few operations. In addition, with the wide availability of parallel computing, more attention is paid to the degree of concurrency of a method. The neural network solution bundle approach is highly amenable to parallelization both during the training and inference stages. The Lagaris method of training these networks is easily parallelized in time[38], with processing units handling weight updates due to each disjoint subset of batch B. The inference stage exhibits trivial parallelization as each state within the solution bundle can be simultaneously evaluated by separate processors, unlike Runge-Kutta or multi-step approaches where states at late times cannot be known before the states that come earlier. The constant execution time and parallelization when evaluating states at different times for different initial conditions and ODE parameters make this method useful when the behavior of a distribution of solutions is desired, such as for propagating state uncertainty or for performing Bayesian inference.

The neural network solution bundle also has a closed analytic form and is differentiable in all of its inputs. This capability, which conventional methods lack, can be used for a variety of useful tasks. For example, in Bayesian inference, differentiability in the initial conditions and ODE parameters simplifies the calculation of maximum a posteriori (MAP) estimates of these quantities given observed data as gradient ascent or derivative-based optimization methods can be performed directly on the log of the posterior distribution over the possible initial conditions and parameters. The differentiability also simplifies the application of "shooting methods" where a condition at a later time is known but it is unclear what parameters and initial conditions are consistent with the constraint. This can be useful for solving nonunique inverse problems. For example, suppose we are interested in trajectories that result in a particular value of one of the state variables. Once we find a set of initial conditions and ODE parameters that leads to the given value at a certain time, which can be accomplished by optimization using the shooting method,

computing the Hessian with respect to time, initial conditions, and parameters will tell us which direction in that combined space we can move in order to find additional trajectories consistent with our constraint.

4.2.1.1 Discussion of the Weighting Function.

For the weighting function $b(t)$ in Eq. (4.3), we found that exponential decay weighting $b(t) = \exp(-\varepsilon(t-t_0))$ works better empirically than a uniform weighting, and ε can be treated as a hyperparameter. There are a couple intuitive reasons why this form of $b(t)$ makes sense for initial value problems: it conveys that earlier times are more important, and it serves to counteract the exponential dependence of global error on local error.

The exponential decay weighting fixes the relative importance of the local errors

$$\mathbf{T}(t_i\,;\mathbf{x}_{0i},\boldsymbol{\theta}_i) \equiv \mathbf{G}\left(\hat{\mathbf{x}}(t_i\,;\mathbf{x}_{0i},\boldsymbol{\theta}_i), \frac{\partial \hat{\mathbf{x}}(t_i\,;\mathbf{x}_{0i},\boldsymbol{\theta}_i)}{\partial t}, t_i\,;\boldsymbol{\theta}_i\right) \tag{4.4}$$

appearing in the loss Eq. (4.3), by their time difference. That is, for any t and t',

$$1 < \frac{b(t)}{b(t+\Delta t)} = \frac{b(t')}{b(t'+\Delta t)} \tag{4.5}$$

for a fixed time separation Δt. As we are trying to find solutions starting from initial conditions by local satisfaction of the differential equation, it is clear that earlier errors impact the absolute error of the states at later times. Even if the rest of the solution perfectly satisfies the ODE, it will be in the wrong region of state space compared to the true solution due to the deviation early on. The condition in Eq. (4.5) expresses this causal relationship well, satisfying the time-translation symmetry of the error dependence.

Another appeal for using a decaying exponential weighting comes from comparison to one-step methods like Runge-Kutta. Given a differential equation of the form

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(t,x), \tag{4.6}$$

a one-step method will have the form

$$x_{n+1} = x_n + h\Phi(t_n,x_n;h), \qquad h = t_{n+1} - t_n, \tag{4.7}$$

where for example the Euler method would have $\Phi(t_n,x_n;h) = f(t_n,x_n)$. The truncation (local) error is given

by

$$T_n = \frac{x(t_{n+1}) - x(t_n)}{h} - \Phi(t_n, x(t_n); h). \tag{4.8}$$

The global error in a one-step method, $\varepsilon_N = x(t_N) - x_N$, which is due to the accumulated truncation error, is bounded by[75]

$$|\varepsilon_N| \leq \frac{T}{L_\Phi}\left(e^{L_\Phi(t_N - t_0)} - 1\right) \tag{4.9}$$

where $T = \max_{0 \leq n \leq N-1} |T_n|$ and, Φ is Lipschitz continuous $|\Phi(t,u;h) - \Phi(t,v;h)| \leq L_\Phi |u - v|$.

We can derive an analogue of this bound on global error for the neural network solution bundle. The local error of Eq. (4.8) should be compared to Eq. (4.4), which for this differential equation, Eq. (4.6), would give

$$T(t) = \frac{\mathrm{d}\hat{x}(t)}{\mathrm{d}t} - f(t, \hat{x}). \tag{4.10}$$

In our approach, it is effectively a truncation error for an infinitesimal timestep. The global error $\varepsilon(t)$ is given by

$$\varepsilon(t) = \hat{x}(t) - x(t), \tag{4.11}$$

which we can substitute into Eq. (4.10) to obtain

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = T(t) - \frac{\mathrm{d}x}{\mathrm{d}t} + f\big(t, x(t) + \varepsilon(t)\big). \tag{4.12}$$

Assuming f is a Lipschitz-continuous function, $\Big|f\big(t, x(t) + \varepsilon(t)\big) - f\big(t, x(t)\big)\Big| \leq L_f |\varepsilon(t)|$, so the absolute value of the right-hand side can be written

$$\begin{aligned}
\left|T(t) - \frac{\mathrm{d}x}{\mathrm{d}t} + f\big(t, x(t) + \varepsilon(t)\big)\right| &= \Big|T(t) + f\big(t, x(t) + \varepsilon(t)\big) - f\big(t, x(t)\big)\Big| && (4.13)\\
&\leq |T(t)| + L_f |\varepsilon(t)| && (4.14)\\
&\leq T_{t'} + L_f |\varepsilon(t)|, && (4.15)
\end{aligned}$$

where we have made use of the differential equation $\mathrm{d}x/\mathrm{d}t = f\big(t, x(t)\big)$, and $T_{t'} = \max_{t_0 \leq t \leq t'} |T(t)|$. Since

$\varepsilon(t_0) = 0$, to find a bound on $|\varepsilon(t)|$ we can consider the solution to the ODE for upper bound $E(t) \geq 0$,

$$\frac{\mathrm{d}E}{\mathrm{d}t} = T_{t'} + L_f E(t) \tag{4.16}$$

where $E(t_0) = 0$, which is $E(t) = \frac{T_{t'}}{L_f}\big(\exp\big[L_f(t-t_0)\big] - 1\big)$. Thus,

$$|\varepsilon(t)| \leq \frac{T_{t'}}{L_f}\Big(e^{L_f(t-t_0)} - 1\Big), \tag{4.17}$$

which can be compared to the global error bound of the discrete case, Eq. (4.9). In addition, we can take $t' = t$ to have the tightest bound afforded by this proof since $T_t \leq T_{t'}$ for $t \leq t'$.

In light of Eq. (4.17), an early local error Eq. (4.10) can have up to an exponential impact on later times. This can be seen by noting that for time $t_1 < t_2$,

$$|\varepsilon(t)| \leq \frac{T_{t_1}}{L_f}\Big(e^{L_f(t-t_0)} - 1\Big) \leq \frac{T_{t_2}}{L_f}\Big(e^{L_f(t-t_0)} - 1\Big). \tag{4.18}$$

That is, the exponential cone of the error bound for times t_0 to t_1 is typically smaller than the one from t_1 to t_2, unless the largest local error over the entire time interval $[t_0, t_2]$ happened at a time prior to t_1, in which case the bounds are equal. Intuitively, if we must have a large local error, we want it to happen as late as possible to reduce the global error, which this upper bound shows can be exponential in the worst case and proportional to that maximum local error up to the time at which we evaluate. This motivates us to assign exponentially greater importance to earlier local errors with the weighting function $b(t)$. Given that the approximating function Eq. (4.2) will generally not be able to perfectly satsify the target ODE everywhere, there will always be some bias error

$$\text{Bias}(\mathbf{w}) = \int_\Theta \int_{X_0} \int_{t_0}^{t_f} \left| \mathbf{G}\left(\hat{\mathbf{x}}(t\,;\mathbf{x}_0,\boldsymbol{\theta}\,;\mathbf{w}), \frac{\partial \hat{\mathbf{x}}(t\,;\mathbf{x}_0,\boldsymbol{\theta}\,;\mathbf{w})}{\partial t}, t\,;\boldsymbol{\theta}\right)\right|^2 \mathrm{d}t\,\mathrm{d}\mathbf{x}_0\,\mathrm{d}\boldsymbol{\theta}_0 > 0, \tag{4.19}$$

regardless of the choice of $\mathbf{w}$. If we do not weight the loss function Eq. (4.3), *i.e.* $b(t) = 1$, we do not get to influence how the local error is distributed across the training region. However, by applying an exponentially decaying weight

$$b(t) = e^{-\varepsilon(t-t_0)} \tag{4.20}$$

we convey in the loss function the exponentially larger contribution of early-time local errors to our metric

of interest, the global error. This way, the minimization of the loss function during training will tend to exponentially suppress local errors at earlier times.

Note that in the above discussion, as well as in the derivation of Eq. (4.17), we assumed scarce knowledge about the behavior of the ODE of interest, using only its Lipschitz constant. It is evident that local errors in some regions of phase space are more important than others—relatively uniform, uneventful regions are unlikely to change the trajectory substantially in the presence of small errors, while others can be substantially less forgiving. It is likely that better choices for the weighting function $b(t)$ exist when prudently chosen for a given system.

4.3 Propagating a Distribution

A neural network solution bundle provides a mapping from initial conditions to the state at later times. This can be useful for time-evolving a distribution over initial conditions to obtain a probability distribution over states at later time t. Given a probability density over initial states $\rho_0(\mathbf{x}_0)$, we note that the solution bundle $\mathbf{x}(t\,;\mathbf{x}_0)$ at time t describes a coordinate transformation from $\mathbf{x}_0$, transforming the coordinates $\mathbf{x}_0$ to $\mathbf{x}_t$. If this transformation is a diffeomorphism from the subset of initial state space X_0 to the final state space X_t, which is the case in many systems in classical mechanics, we can write out the probability density of later states,

$$\rho_t(\mathbf{x}_t) = \rho_0\big(\mathbf{f}^{-1}(\mathbf{x}_t)\big)\,|\mathbf{J}_{\mathbf{f}^{-1}}|, \tag{4.21}$$

where $\mathbf{f}(\mathbf{x}_0) \equiv \mathbf{x}(t\,;\mathbf{x}_0)$, and $\mathbf{J}_{\mathbf{f}^{-1}} = \frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_t}$ is the Jacobian of $\mathbf{f}^{-1}$. In the case of a Hamiltonian dynamical system, Liouville's theorem guarantees the conservation of phase space density, *i.e.* $|\mathbf{J}_{\mathbf{f}^{-1}}| = 1$ and

$$\rho_t(\mathbf{x}_t) = \rho_0\big(\mathbf{f}^{-1}(\mathbf{x}_t)\big), \tag{4.22}$$

giving the probability density of the state being at $\mathbf{x}_t$ directly provided that the corresponding initial state $\mathbf{x}_0$ has been found.

The opposite task, *i.e* where a probability distribution over later states $\rho_t(\mathbf{x}_t)$ is known and the distribution over initial states is desired, takes on an even more convenient form where the neural network solution bundle does not have to be inverted:

$$\rho_0(\mathbf{x}_0) = \rho_t(\mathbf{f}(\mathbf{x}_0))|\mathbf{J}_{\mathbf{f}}|, \tag{4.23}$$

where $\mathbf{f}(\mathbf{x}_0) \equiv \mathbf{x}(t\,;\mathbf{x}_0)$, and $\mathbf{J_f} = \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_0}$ is the Jacobian of $\mathbf{f}$, which can be calculated exactly using automatic differentiation of the solution bundle. This gives a closed analytic form for the desired probability density. Note that if the dynamical system is time-reversible, Eq. (4.21) can be converted to the easier form Eq. (4.23) by simply treating the target space as the input space and training the solution bundle on the time-reversed equations of motion.

While Eqs. (4.21) and (4.22) can be useful, in practice we can simply sample the initial state space and construct a histogram of output $\mathbf{x}_t$ states using the solution bundle. This also removes the diffeomorphic requirement of $\mathbf{x}_t(\mathbf{x}_0) \equiv \mathbf{x}(t\,;\mathbf{x}_0)$, instead only requiring it to be a single-valued function, *i.e.* requiring that the ODE be deterministic. The sampling of initial states can be done according to $\rho_0(\mathbf{x}_0)$ using Markov chain Monte Carlo (MCMC) methods, or if the dimensionality of the state vector is low enough, by simply performing a uniform sampling over the initial states and constructing a weighted histogram of $\mathbf{x}_t$ weighting each sample by $\rho_0(\mathbf{x}_0)$ for the $\mathbf{x}_0$ that generated it.

4.3.1 Planar Circular Restricted Three-Body Problem

The planar circular restricted three-body problem describes a special case of the motion of three masses under Newton's law of universal gravitation. This special case describes the motion of the third body, which is assumed to have negligible mass, in the co-rotating frame of the first two bodies in circular orbits around their barycenter. All three bodies are also assumed to lie in the same plane (xy plane), with no velocity in the z direction. For clarity of discussion, let body 1 be the Earth, body 2 be the Moon, and body 3 be an asteroid. In the co-rotating frame, we can shift the coordinates such that the Earth is located at the origin, and the Moon is located at $x = 1$. The coordinates of the Earth and the Moon remain unchanged with time, while the asteroid has position $\mathbf{r}(t) = (x(t), y(t))^\intercal$ and velocity $\mathbf{u}(t) = (u(t), v(t))^\intercal$. We will call the full state vector $\mathbf{q} = (\mathbf{r}^\intercal, \mathbf{u}^\intercal)^\intercal = (x, y, u, v)^\intercal$. The nondimensionalized mass of the Earth is given by $m_1 = 1 - \mu$, and the mass of the Moon is $m_2 = \mu$, where μ is the ratio of the mass of the the Moon to the total mass of the pair. The

nondimensionalized equations of motion of the asteroid are given by:

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = u\,, \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = v\,, \\ \dfrac{\mathrm{d}u}{\mathrm{d}t} = x-\mu+2v-\left[\dfrac{\mu(x-1)}{\left((x-1)^2+y^2\right)^{3/2}}+\dfrac{(1-\mu)x}{(x^2+y^2)^{3/2}}\right], \\ \dfrac{\mathrm{d}v}{\mathrm{d}t} = y-2u-\left[\dfrac{\mu y}{\left((x-1)^2+y^2\right)^{3/2}}+\dfrac{(1-\mu)y}{(x^2+y^2)^{3/2}}\right]. \end{cases} \tag{4.24}$$

A derivation can be found in Szebehely's *Theory of Orbits. The Restricted Problem of Three Bodies*[74].

4.3.1.1 Training the Solution Bundle

For this system, we will use fixed parameters in the ODE, and train the neural network ansatz to approximate a solution bundle over a space X_0 of initial conditions. We will use $\mu = 0.01$, approximately equal to the true Moon/(Earth+Moon) mass ratio. The network we trained is a fully-connected MLP with 8 hidden layers consisting of 128 neurons each.

We trained uniformly over the x_0, y_0, u_0, v_0 initial condition space $X_0 = [1.05, 1.052] \times [0.999, 0.101] \times [-0.5, -0.4] \times [-0.3, -0.2]$ and times $[-0.01, 5]$. Even though we only intend to evaluate the solution bundle at times $t \in [0,5]$, we found that including times slightly earlier than t_0 in training helps improve accuracy. This makes the approximating function satisfy the ODE on both sides of t_0, resulting in a more accurate value of the derivative term in Eq. (4.3) around $t = t_0$. We used batchsize $|B| = 10{,}000$, the Adam optimizer[30], and learning rate $\eta = 0.001$, which we reduced on plateau. For the weighting function $b(t)$ in the loss, Eq. (4.3), we chose $b(t) = \exp(-\varepsilon t)$ where $\varepsilon = 2$.

In Figure 4.1 we show the loss versus training batch number, averaged over a moving window of 10,000 batches. Notice the presence of steps in the loss curve; these drops correspond to the moments when the learning rate was decreased by a factor of 2. This may be related to the weighting function $b(t) = \exp(-\varepsilon t)$ creating a hierarchy of importance in the loss landscape. The larger starting learning rate finds a large scale region of optimality in weight-space, and as the learning rate is decreased, smaller divots in the loss landscape can be better explored, corresponding to parts of the trajectory at later times which have their loss contribution exponentially suppressed. This can be observed in Figure 4.2, where a few solutions in the solution bundle are plotted for checkpoint models saved during the course of training. Notice that the

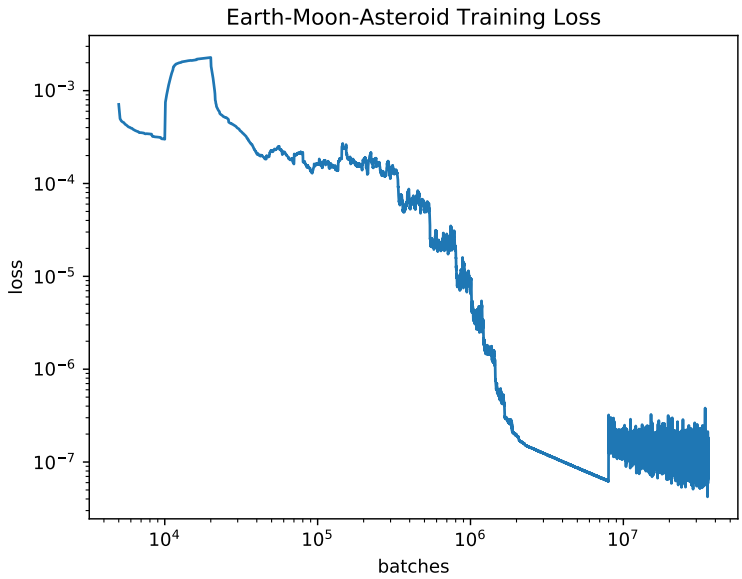

Figure 4.1: Loss as a function of batch number during training of the Earth-Moon-asteroid system. A moving average with a window of 10,000 batches is applied to smooth out the curve.

approximate trajectories at early times are the first to line up with the reference trajectories, and the rest of the trajectory slowly follows. The increase in loss at batch 8×10^6 is due to raising the learning rate from 10^{-6} to 10^{-5} after we noticed that the network became stuck in a local minimum. The local minimum can actually be seen in Figure 4.2, where at batch 8×10^6 the end of the trajectory bundle is stuck together and not fanning out. The increased learning rate allowed the network to jump out of the local minimum, and the trajectory bundle began to spread as required.

Figure 4.2 also shows the parallelization in time, with the curvature of later parts of the trajectory adjusting even before the earlier states have settled. This is reminiscent of the current dominant approach for parallelization in time, Parareal[41], which also involves computation of approximate later trajectories before their earlier paths are precisely known.

4.3.1.2 Propagation of Uncertainty

With the neural network solution bundle trained, we can now use it to propagate distributions in time. Suppose we have two measurements of the position of an asteroid at two different times, along with some

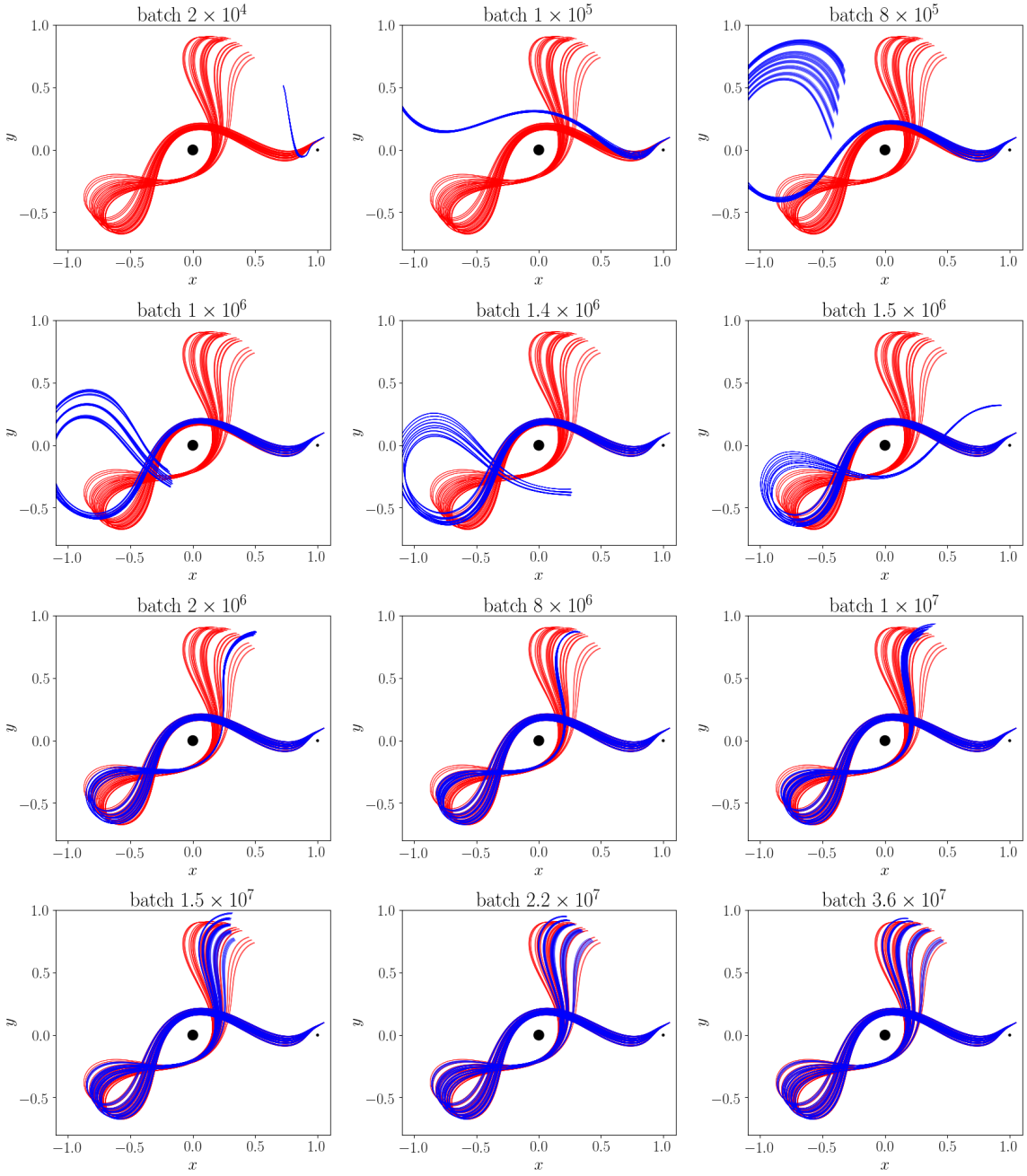

Figure 4.2: Plots of a few trajectories from the neural network solution bundle at various points in the training. Red trajectories are calculated with fourth-order Runge-Kutta, and the neural network solutions are shown in blue.

uncertainty:

$$(t_0, x_0, y_0) = (0.00, 1.0510 \pm 0.0003, 0.1000 \pm 0.0003) \tag{4.25a}$$

$$(t_1, x_1, y_1) = (0.05, 1.0276 \pm 0.0003, 0.0878 \pm 0.0003) \tag{4.25b}$$

If the majority of the probability mass of these uncertainty distributions falls within the solution bundle, it is easy to compute a probability distribution for the future position of the asteroid. Let $\rho\big(\mathbf{r}(t) = (x,y)^\intercal \mid \{\mathbf{r}_1, \mathbf{r}_0\}\big)$ be the probability density of the position at time t being $(x,y)^\intercal$ given the position measurements $\mathbf{r}_0$ and $\mathbf{r}_1$. By marginalizing over the final velocities, we obtain

$$\rho\big(\mathbf{r}(t) = (x,y)^\intercal \mid \{\mathbf{r}_1, \mathbf{r}_0\}\big) = \iint \rho\big(\mathbf{q}(t) = (x,y,u,v)^\intercal \mid \{\mathbf{r}_1, \mathbf{r}_0\}\big)\,\mathrm{d}u\,\mathrm{d}v\,. \tag{4.26}$$

To compute the integrand, we can use Bayes' theorem,

$$\begin{aligned}\rho\big(\mathbf{q}(t) = (x,y,u,v)^\intercal \mid \{\mathbf{r}_1, \mathbf{r}_0\}\big) &= \frac{\rho\big(\{\mathbf{r}_1, \mathbf{r}_0\} \mid \mathbf{q}(t) = (x,y,u,v)^\intercal\big)\rho\big(\mathbf{q}(t) = (x,y,u,v)^\intercal\big)}{\rho\big(\{\mathbf{r}_1, \mathbf{r}_0\}\big)} \\ &\propto \rho\big(\mathbf{r}_1 \mid \mathbf{q}(t) = (x,y,u,v)^\intercal\big)\rho\big(\mathbf{r}_0 \mid \mathbf{q}(t) = (x,y,u,v)^\intercal\big),\end{aligned} \tag{4.27}$$

where in the last step we have assumed a uniform prior and that the errors in the two position measurements $\mathbf{r}_0$ and $\mathbf{r}_1$ are independent. So, by simply iterating over a uniform grid of initial positions and velocities, evaluating the solution at time t_1 and t, weighting the samples with the probability densities given by Eq. (4.27), and forming a weighted histogram of positions, we have the approximate distribution of the asteroid's location at time t. Figure 4.3 shows the results at various final times t. When uniform sampling becomes infeasible due to high dimensionality of the inputs, it is straightforward to use MCMC instead.

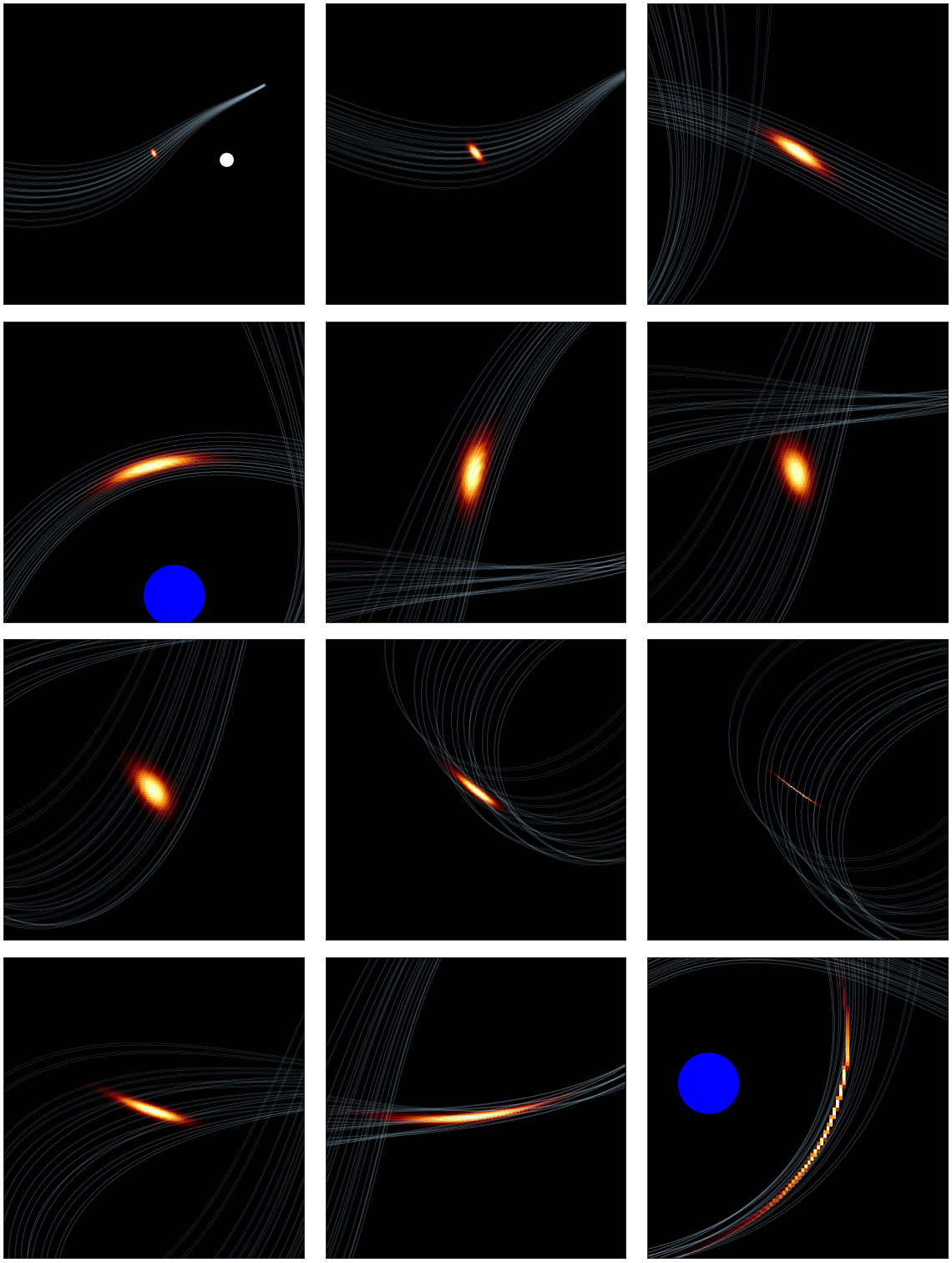

Figure 4.3: Probability distribution $\rho(\mathbf{x}(t) = (x, y)^\intercal)$ at various times. A few trajectories in the bundle are shown in white, and the distribution is shown as a heatmap. The full path is shown in Figure 4.2

4.4 Bayesian Parameter Inference

Another common task that requires computing many solutions to an ODE is Bayesian parameter inference in systems described by differential equations. In the physical sciences and other fields employing mathematical modeling of data, it is often necessary to estimate parameters of a system based on experimental measurements, as well as to determine their uncertainties or probability distributions. If the system is described by differential equations, these parameters modify terms in the equations, resulting in different families of solutions. The probability density of the initial conditions and parameters $\mathbf{x}_0, \boldsymbol{\theta}$ given a set of observed data $\{(t_i, \mathbf{x}_i)\}$ and prior $\rho(\mathbf{x}_0, \boldsymbol{\theta})$, the posterior distribution, can be computed from Bayes' theorem,

$$\rho\big(\mathbf{x}_0, \boldsymbol{\theta} \mid \{(t_i, \mathbf{x}_i)\}\big) = \frac{\rho\big(\{(t_i, \mathbf{x}_i)\} \mid \mathbf{x}_0, \boldsymbol{\theta}\big)\rho(\mathbf{x}_0, \boldsymbol{\theta})}{\rho\big(\{(t_i, \mathbf{x}_i)\}\big)} \propto \rho\big(\{(t_i, \mathbf{x}_i)\} \mid \mathbf{x}_0, \boldsymbol{\theta}\big)\rho(\mathbf{x}_0, \boldsymbol{\theta}). \tag{4.28}$$

Determination of the likelihood $\rho\big(\{(t_i, \mathbf{x}_i)\} \mid \mathbf{x}_0, \boldsymbol{\theta}\big)$ is typically the computationally intensive step as it requires computing $\big\{\big(t_i, \mathbf{x}(t_i\,; \mathbf{x}_0, \boldsymbol{\theta})\big)\big\}$ for parameters $\mathbf{x}_0, \boldsymbol{\theta}$ to compare to the data $\{(t_i, \mathbf{x}_i)\}$. Evaluating $\mathbf{x}(t_i\,; \mathbf{x}_0, \boldsymbol{\theta})$ with conventional methods would require forward stepping from the initial conditions all the way to time t_i, and this process would have to be repeated for every different set of initial states and parameters $\mathbf{x}_0, \boldsymbol{\theta}$. The greater the desired precision of the posterior distribution for the parameters, the more often the differential equation has to be solved. However, if a neural network solution bundle has been trained over X_0 and Θ containing the expected range of initial conditions and parameters, $\hat{\mathbf{x}}(t_i\,; \mathbf{x}_0, \boldsymbol{\theta})$ can be calculated in constant time for any $t_i \in [t_0, t_f]$, and the entire set of points $\big\{\big(t_i, \hat{\mathbf{x}}(t_i\,; \mathbf{x}_0, \boldsymbol{\theta})\big)\big\}$ can be computed in parallel. This allows for rapid likelihood evaluation and more efficient Bayesian inference. The training cost of the solution bundle can be further offset if it is used for many different sets of data, each with their own underlying parameters. In effect, a carefully-trained neural network solution bundle over a wide variety of parameters and initial conditions for an often-used ODE in a field could be shared amongst research groups, cutting back on the number of redundant calculations performed globally. As mentioned in Lagaris's original paper[38], neural networks can compactly represent high-dimensional functions such as these solution bundles, contributing to the feasibility of this use case.

4.4.1 Simple Harmonic Oscillator

One of the simplest dynamical systems is the simple harmonic oscillator, which can describe the motion of a mass on a spring. The equations of motion are given by

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v \tag{4.29a}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{k}{m}x, \tag{4.29b}$$

and we use state vector $\mathbf{x} = (x, v)^\intercal$. We choose $m = 1$ for simplicity, so the parameter in this system is $\theta = k$, the Hooke spring constant. We generate position data based on a selected ground truth trajectory and add Gaussian noise with standard deviation $\sigma = 0.1$, and assume that we do not have any velocity data.

To compute the posterior distribution, we take the logarithm of the Bayes formula Eq. (4.28) and make use of the independence of the data points:

$$\log \rho\left(\mathbf{x}_0, k \mid \{(t_i, x_i)\}\right) = \sum_i \log \rho\left((t_i, x_i) \mid \mathbf{x}_0, k\right) + \log \rho(\mathbf{x}_0, k) - C, \tag{4.30}$$

where $C = \log \rho(\{(t_i, x_i)\})$ is the log of the evidence, which is not important to know because it can be recovered by ensuring the unit norm of the posterior distribution. The terms in the likelihood can be computed using

$$\rho\left((t_i, x_i) \mid \mathbf{x}_0, k\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x_i - \hat{x}(t_i\,;\, \mathbf{x}_0, k)}{\sigma}\right)^2\right] \tag{4.31}$$

$$\log \rho\left((t_i, x_i) \mid \mathbf{x}_0, k\right) = -\frac{1}{2}\left(\frac{x_i - \hat{x}(t_i\,;\, \mathbf{x}_0, k)}{\sigma}\right)^2 + c, \tag{4.32}$$

where the resulting constants c for each likelihood term can be combined with the normalizing factor C in Eq. (4.30). In addition, since we assume a uniform prior we can put the log-prior into C as well. The log-posterior can be computed over a grid of x_0, v_0, and k in this case, since the low dimensionality of the space permits it.

We trained a solution bundle over $X_0 = [-1, 1] \times [-1, 1]$ and $\Theta = [0.5, 2]$, for times $[-0.01, 2\pi]$. The neural network in the ansatz Eq. (4.2) consisted of an input layer of size 4, 4 hidden layers of size 128, and an output layer of size 2. All activations were hyperbolic tangents except for a linear activation on the output layer. The fit of the data using the initial conditions and paramaters that maximize the posterior, the maximum a posteriori (MAP) estimate, is shown in Figure 4.4, and the marginal posterior distributions are

shown in Figure 4.5.

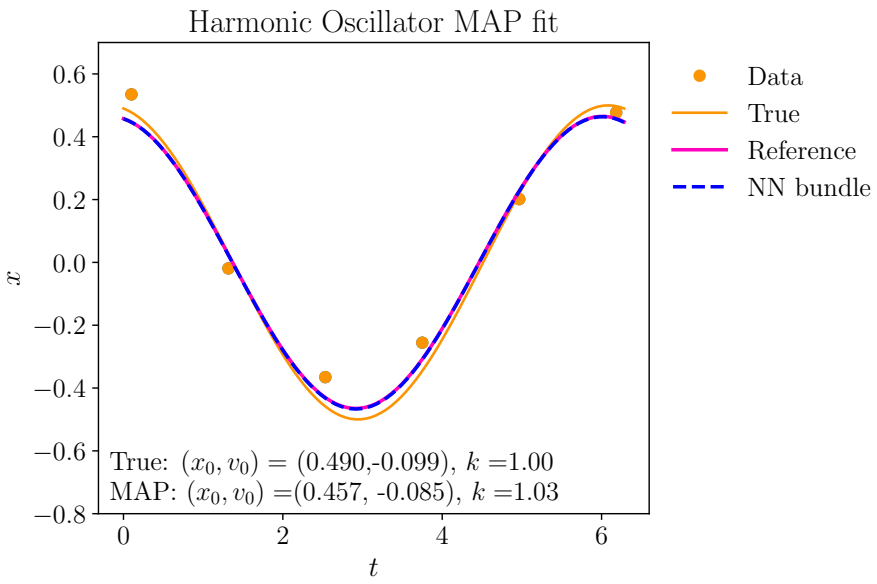

Figure 4.4: Simple harmonic oscillator fit corresponding to maximum a posteriori estimate, given the data and uniform prior.

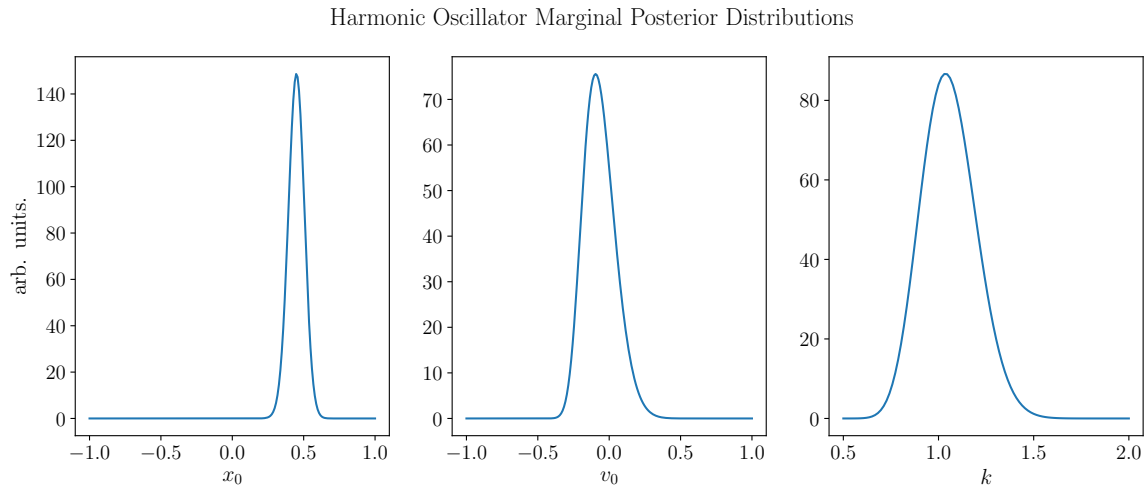

Figure 4.5: Simple harmonic oscillator initial conditions and spring constant k marginal distributions given the data shown in Figure 4.4.

In Figure 4.4, a reference solution computed with standard methods for the MAP estimate initial conditions and parameters is plotted to confirm that the best fitting trajectory from the neural network solution bundle is sufficiently converged.

A harmonic oscillator has a closed analytic solution, namely, all of the possible solutions to Eq. (4.29b)

are linear combinations of sine and cosine with frequency $\sqrt{k/m}$, so in this simple case there is not much benefit to using the neural network solution bundle. However, for the next two systems, an exact closed-form solution is not known, but the approximating solution bundle can grant the same advantages as an analytic solution.

4.4.2 Rebound Pendulum

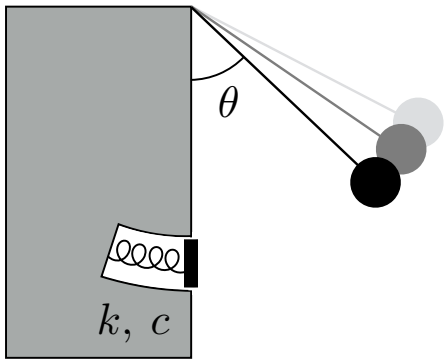

Figure 4.6: Diagram of a rebound pendulum. A pendulum with mass m swings under the influence of gravity and can rebound off a spring with Hooke constant k and damping coefficient c.

As a harder system, consider the rebound pendulum which consists of a simple pendulum that can collide with a damped spring at the bottom of its swing. A diagram of the setup is shown in Figure 4.6. Various forms of rebound pendula are used in the rubber industry for measuring rebound resilience, the ratio of the energy returned to the energy applied to a test piece of rubber as the result of a single impact of a striker. In conjunction with a contact time measurement, rebound pendula can also be used to measure the dynamic modulus of rubber[5]. The rubber pad is treated as a Kelvin-Voigt material, represented by a purely viscous damper and a purely elastic spring connected in parallel. The equations of motion for the state vector $\mathbf{x} = (\theta, \omega)^\intercal$ of the pendulum are given by

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \omega \tag{4.33a}$$

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = -\frac{g}{\ell}\sin\theta + H(-\theta)\mathrm{ReLU}\left(-\frac{k}{m}\theta - c\omega\right), \tag{4.33b}$$

where $\mathrm{ReLU}(x) = \max(x, 0)$, $H(x)$ is the Heaviside step function, g is the gravitational acceleration, ℓ is the

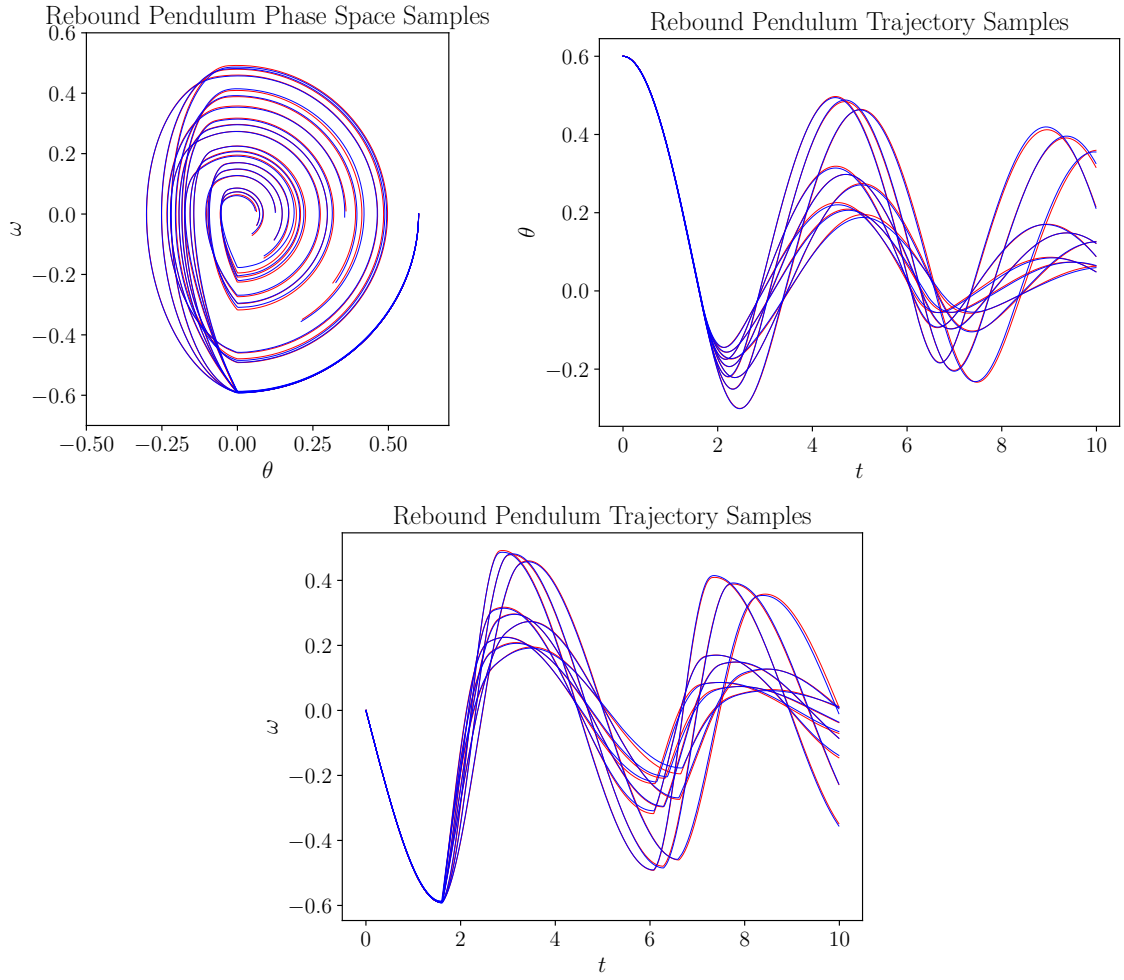

Figure 4.7: A selection of solutions within the rebound pendulum solution bundle. The initial state $(\theta_0, \omega_0) = (0.6, 0)$ is fixed while the solutions corresponding to different combinations of spring constant k and damping coefficient c are plotted. The red reference curves are computed with fourth-order Runge-Kutta and the trajectories from the solution bundle are overlayed in blue.

length of the pendulum, k is the spring constant, and c is the damping coefficient. The second term of Eq. (4.33b) describes how the pendulum interacts with the spring: the spring can only interact with the pendulum for negative angles, hence the step function, and the spring can only push the pendulum away, not "stick" to it and slow it down, hence the rectifier.

We trained a fully-connected network with 8 hidden layers of size 128 and hyperbolic tangent activation over $X_0 = [0,1] \times [-0.2, 0.2]$, $\Theta = [2,5] \times [0,2]$ and times $[-0.01, 10]$. We used $\varepsilon = 0.5$ in the weighting function $b(t)$, and a batch size of 10,000. A few example solutions are shown in Figure 4.7, where the convergence of the approximation can be compared to reference solutions computed with Runge-Kutta. Small deviations become apparent at later times, which can be reduced in part by longer training and further

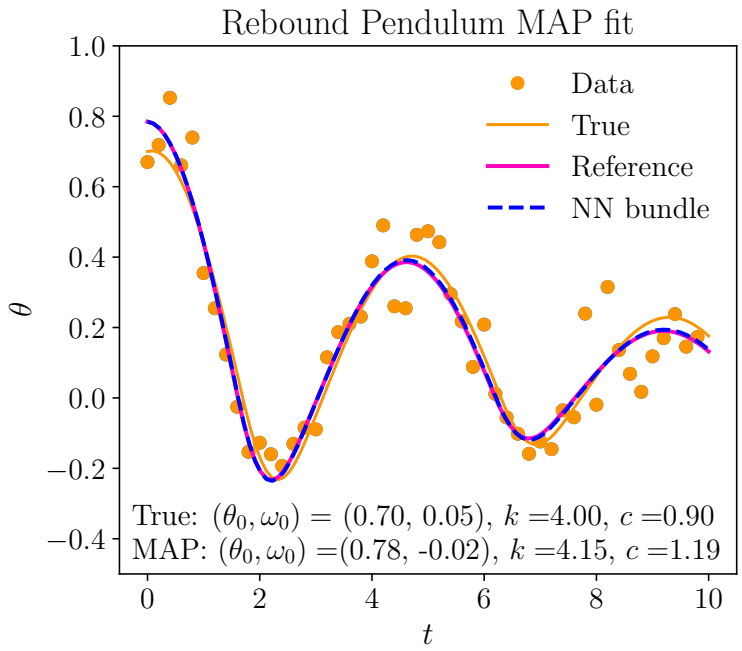

Figure 4.8: Rebound pendulum fit corresponding to maximum a posteriori estimate, given the data and uniform prior.

by increasing network complexity.

Using the trained solution bundle, we can fit simulated angle θ measurements which have Gaussian error with $\sigma = 0.1$. The result of Bayesian inference of the initial conditions and parameters are shown in Figure 4.8 and 4.9.

4.4.3 FitzHugh-Nagumo Model

The FitzHugh-Nagumo model[17,53] is a relaxation oscillator which can be used as a simple model of a biological neuron. Its state is described by a membrane voltage v, and a recovery variable w, and the ODE has parameters a, b, τ, and I. Certain combinations of the parameters can result in a single spike, where the voltage quickly rises and falls, subsequently relaxing to a constant value, while other combinations can lead to "spike trains" where the voltage continues to spike at periodic intervals. If the system is to model a biological neuron, it is useful to fit the system parameters to measured data, which we will simulate in this section. The FitzHugh-Nagumo model can be coupled with many copies of itself to simulate a biological neural network[51]. This suggests another possible use for neural network solution bundles: a solution bundle can be trained to capture the dynamics of a single base unit, making simulations of connected networks of these units cheaper to simulate. This is again leveraging the benefits of avoiding redundant "on-demand"

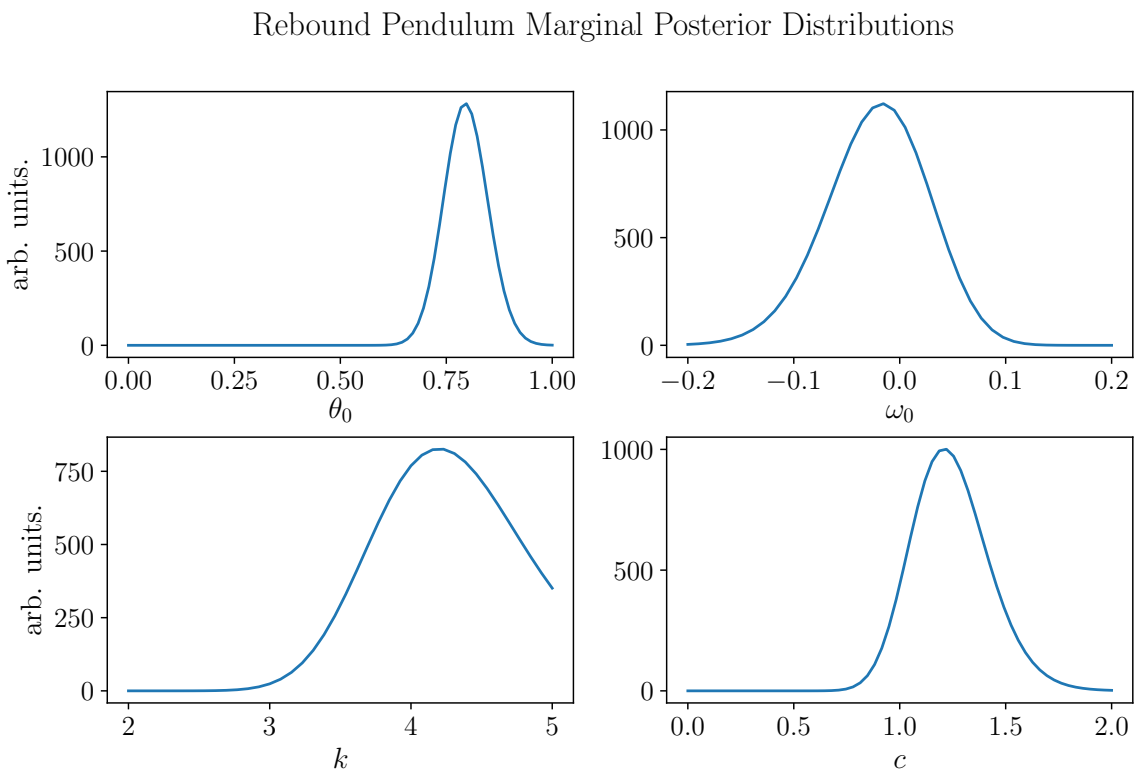

Figure 4.9: Rebound pendulum initial conditions, spring constant k, and damping coefficient c marginal distributions given the data shown in Figure 4.8.

solving of the ODE for a particular set of initial conditions and parameters through pre-training.

The differential equation for the FitzHugh-Nagumo model is given by

$$\frac{\mathrm{d}v}{\mathrm{d}t} = v - \frac{v^3}{3} - w + I, \tag{4.34a}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{1}{\tau}(v + a - bw). \tag{4.34b}$$

A typical spike train trajectory is plotted in red in Figure 4.10, forming a closed loop as the voltage continuously spikes and resets. The nullclines of Eqs. (4.34a) and (4.34b), *i.e.* where $\mathrm{d}v/\mathrm{d}t = 0$ and $\mathrm{d}w/\mathrm{d}t = 0$, help with understanding the dynamics of the FitzHugh-Nagumo model. The nullclines are given by the equations

$$w = v - \frac{v^3}{3} + I, \qquad \Rightarrow \frac{\mathrm{d}v}{\mathrm{d}t} = 0 \tag{4.35a}$$

$$w = \frac{v + a}{b}, \qquad \Rightarrow \frac{\mathrm{d}w}{\mathrm{d}t} = 0 \tag{4.35b}$$

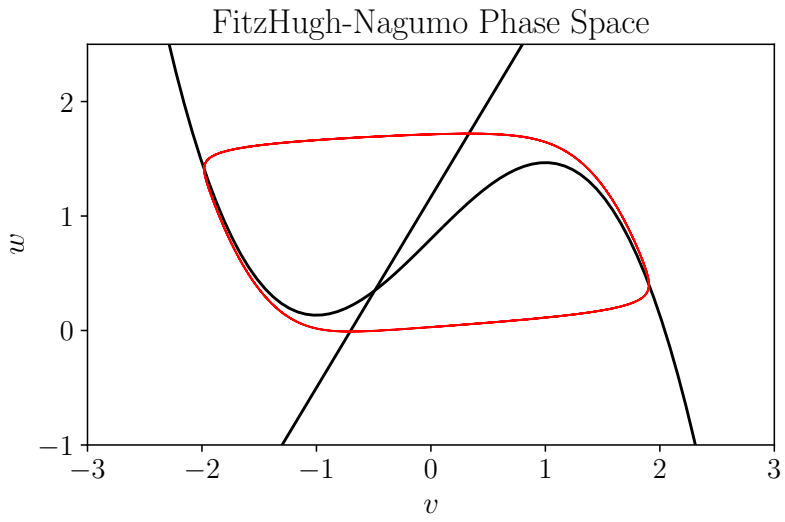

Figure 4.10: A phase space trajectory of the FitzHugh-Nagumo model shown in red. The black curves are nullclines, with the cubic nullcline indicating where $\mathrm{d}v/\mathrm{d}t = 0$, and the linear nullcline indicating where $\mathrm{d}w/\mathrm{d}t = 0$.

The nullclines are plotted in black in Figure 4.10. As the nullclines indicate zero crossings of the derivatives, they show where a trajectory will reverse in one of its coordinates. The parameters a, b, and I control the shapes of these nullclines while τ controls how quickly the system moves across trajectories in vw-space.

For the FitzHugh-Nagumo system, we found that it could be tricky to train the neural network solution bundle. During training, if for each batch we simply sampled times uniformly from the time interval $[t_0, t_f]$, we found that the solution bundle could get "stuck" to the nullclines, as seen in Figure 4.11.

This seems to happen because while the approximate solution bundle is dragged around phase space during early training, the tail end can encounter a nullcline by chance, where it is easier to satisfy the ODE. In the loss function Eq. (4.3), even though an error at an earlier time has to be made in the solution bundle for the tail to be on the nullcline as seen in Figure 4.11, if the later times have very low error due to the ease of simply predicting the same constant value whilst on the nullcline, the network weights can become stuck in this local minimum.

This pitfall can be avoided by applying curriculum learning[6] to the training process. If we want to avoid the influence of later times pinning the solution bundle before the earlier times are sufficiently converged, we can simply change how we sample the training set $[t_0, t_f]$ when we form the batches B in Eq. (4.3). In particular, when training starts, we can restrict the time samples to come from $[t_0, t_m]$, where $t_m < t_f$ and m is the batch number. As training progresses and the batch number m is increased, we can make the task a bit more difficult by increasing t_m, requiring the network to learn a greater portion of the solution bundle.

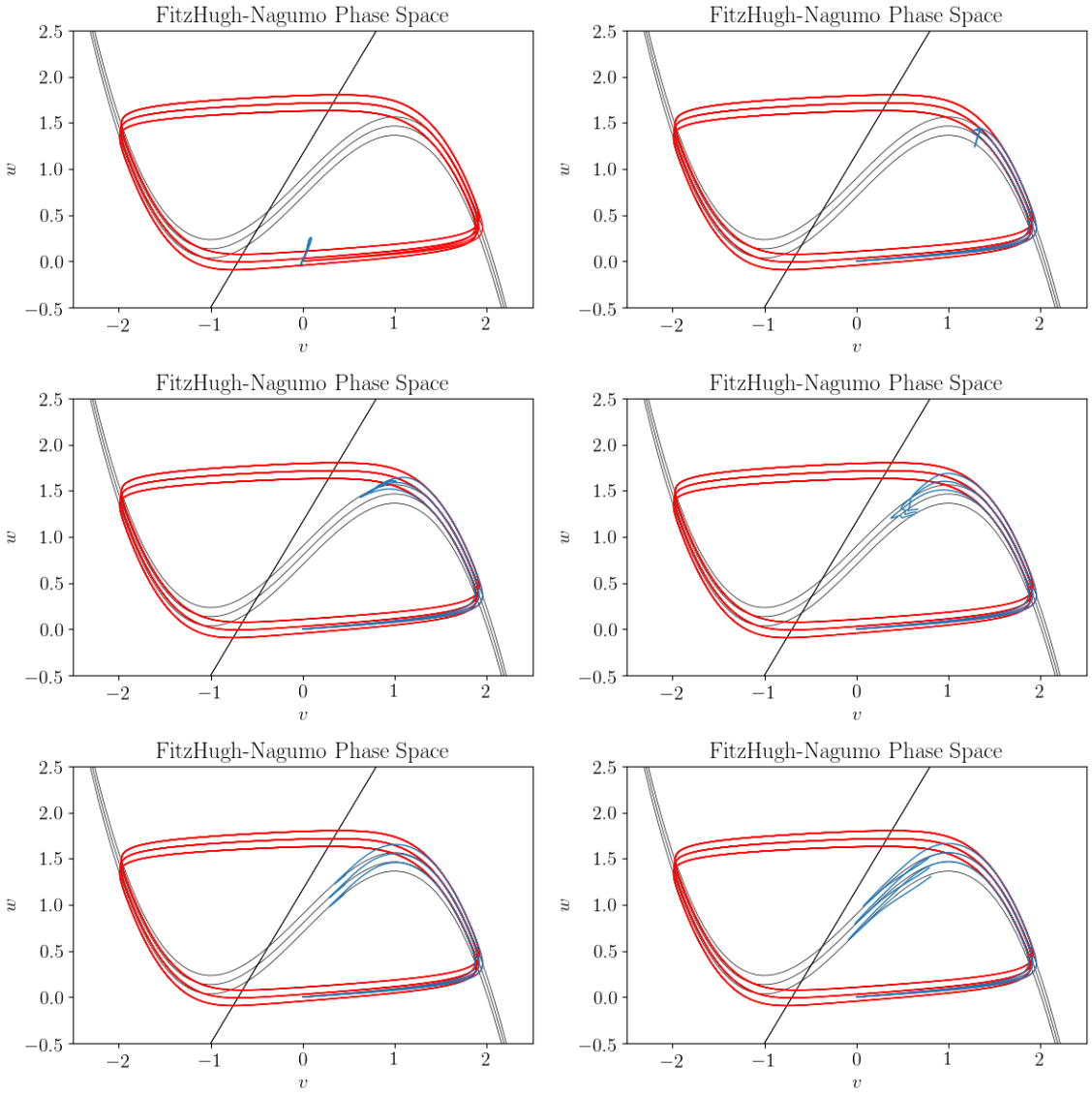

Figure 4.11: A few reference trajectories with three values of I are shown in red, the corresponding nullclines for those values are shown in black, and the approximate solution bundle at various points in training is shown in blue. Notice that the approximation gets stuck on the cubic nullcline.

The result of curriculum learning is shown in Figure 4.12. The entire solution bundle is plotted in each frame, but in the loss calculation only times up to t_m are sampled so the tail that flops around no longer has influence on the convergence of the network, and hence it no longer gets stuck on nullclines.

For this run we used a feedforward network with fully-connected layers and some additional connections from the input layer to each hidden layer. Specifically, in addition to the standard all-to-all connections in subsequent layers, each hidden layer also had incoming connections from each of the input neurons. We

found that these residual connections which remind the hidden layers what the input time, initial conditions, and parameters helped with the curriculum learning, possibly due to later hidden layers being able to start adapting earlier to the increasing time domain without having to wait for the activation distribution from preceeding layers to adjust to the new distribution of time inputs.

The network consisted of an input layer of size 7 (time, 2 initial conditions, 4 parameters), 8 hidden layers with 121 neurons each, and an output layer of size 2. Each hidden layer had incoming connections from the input layer. We used hyperbolic tangent activations on all neurons except the output neurons which had linear activations. Initial conditions v_0, w_0 came from the set $X_0 = [-0.1, 0.1] \times [-0.1, 0.1]$ and parameters a, b, τ, I from $\Theta = [0.6, 0.8] \times [0.5, 0.7] \times [11, 14] \times [0.7, 0.9]$. The target time interval was $[-0.1, 100]$, which was worked up to using curriculum learning. During training, the times were drawn from batch-number-dependent $[-0.1, t_m]$ with

$$t_m = \frac{100}{\log(11)} \log\left(\frac{11m}{M} + 1\right), \tag{4.36}$$

where m is the batch number and M is the total number of batches. This logarithmic function allows for the maximum time t_m to increase at a slower rate as the training progresses so the solution bundle has more training iterations to adjust to a larger time interval $[-0.1, t_m]$. In addition, we adjusted the weighting function as the training progressed, with

$$b_m(t) = \exp(-\varepsilon_m t), \qquad \varepsilon_m = \frac{4}{t_m + 5}. \tag{4.37}$$

Once the curriculum learning was done and the neural network had been exposed to the full training interval $[t_0, t_f] = [-0.1, t_m]$, we continued the training with the full interval and a fixed weighting function

$$b(t) = \exp\left(-\frac{4}{100} t\right) \tag{4.38}$$

to allow the network to settle in to the final task.

Figure 4.13 shows a sample of trajectories in v and w for fixed initial conditions $(v_0, w_0) = (0, 0)$ and various combinations of the ODE parameters. The red reference curves are computed with Runge-Kutta and the blue curves show the selected solutions from the training bundle. The agreement is decent overall, though the results could be improved with a greater network complexity or perhaps longer training. For the amount of training we have performed, the agreement is excellent up until $t = 60$, which is the maximum value we will use for the Bayesian inference task on simulated data.

Simulated (t_i, v_i, w_i) data is generated with Gaussian noise of $\sigma = 0.5$ in both coordinates v and w. We perform Bayesian inference using the trained neural network bundle, and the resulting MAP fit is shown in Figure 4.14. Figure 4.15 shows the marginal posterior distributions of the parameters. For simplicity the initial condition $(v_0, w_0) = (0, 0.0294)$ is assumed to be known exactly in this case.

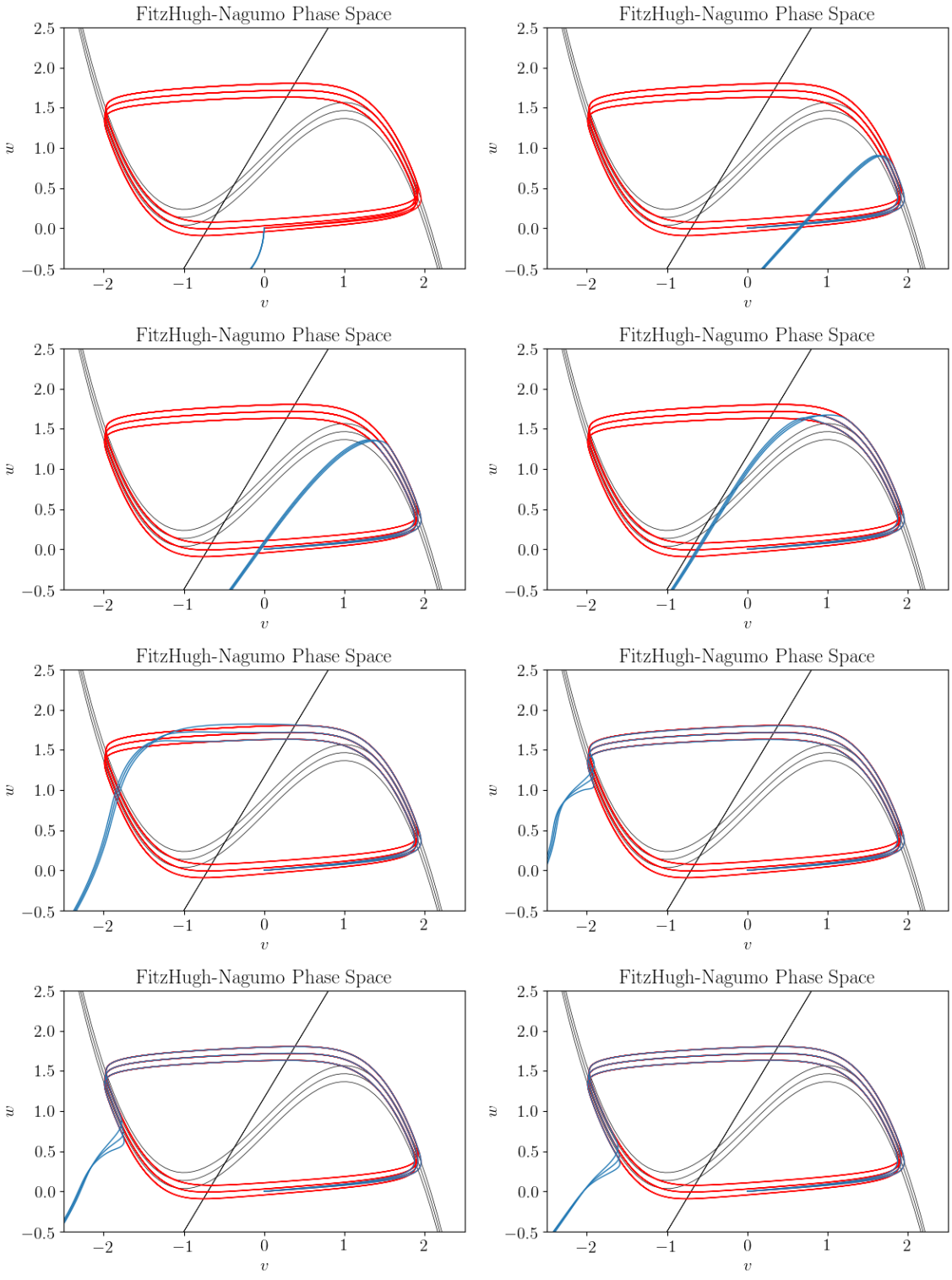

Figure 4.12: The same setup as in Figure 4.11, but we apply curriculum learning during training: only times up to an progressively increasing cutoff t_m are used in computing the loss for batch number m. The solution bundle traces out the correct trajectory as training progresses.

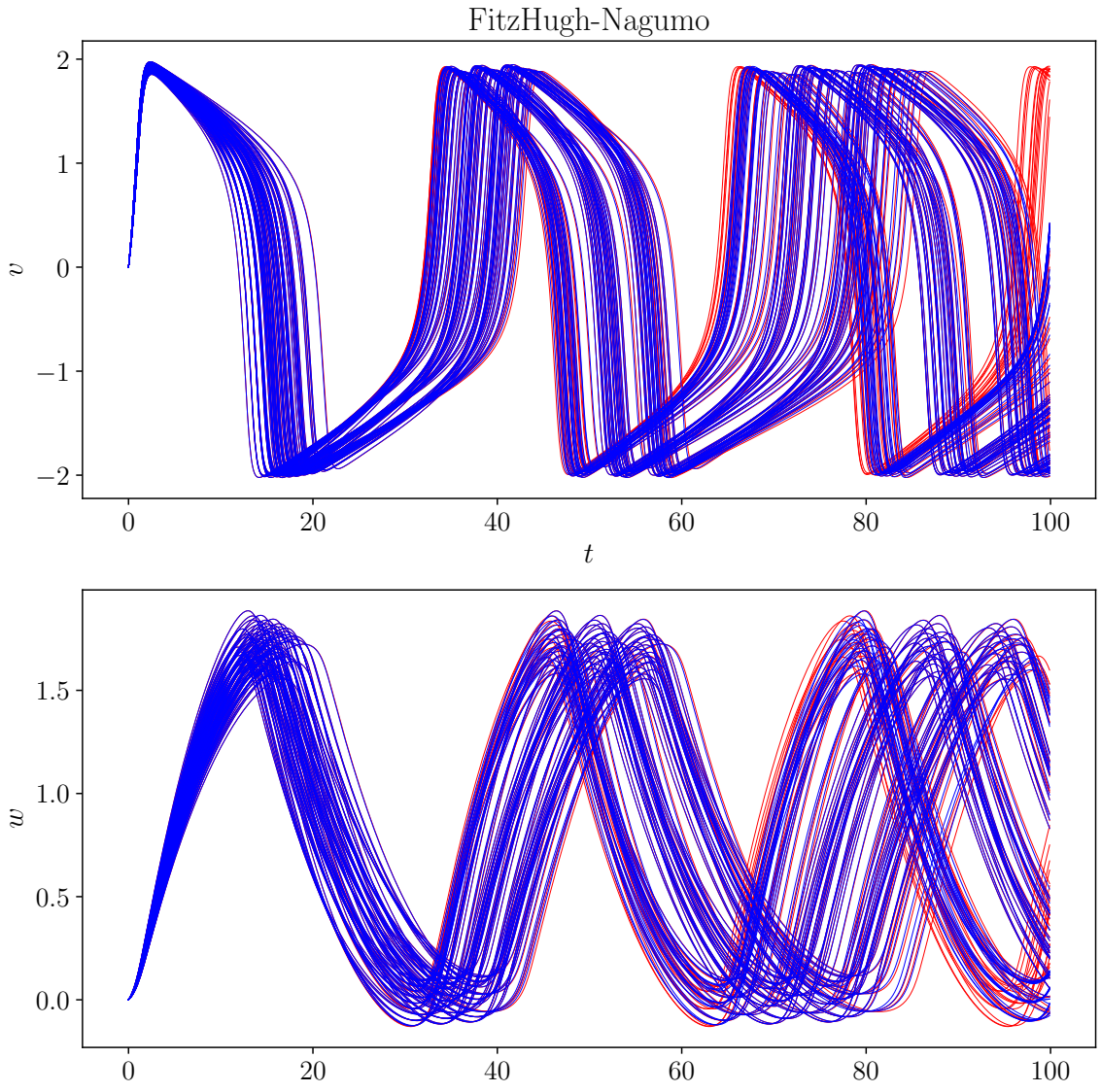

Figure 4.13: A plot showing the accuracy of the neural network solution bundle at the end of training. The red reference curves are plotted with fourth-order Runge-Kutta, and the solution bundle's trajectories are in blue.

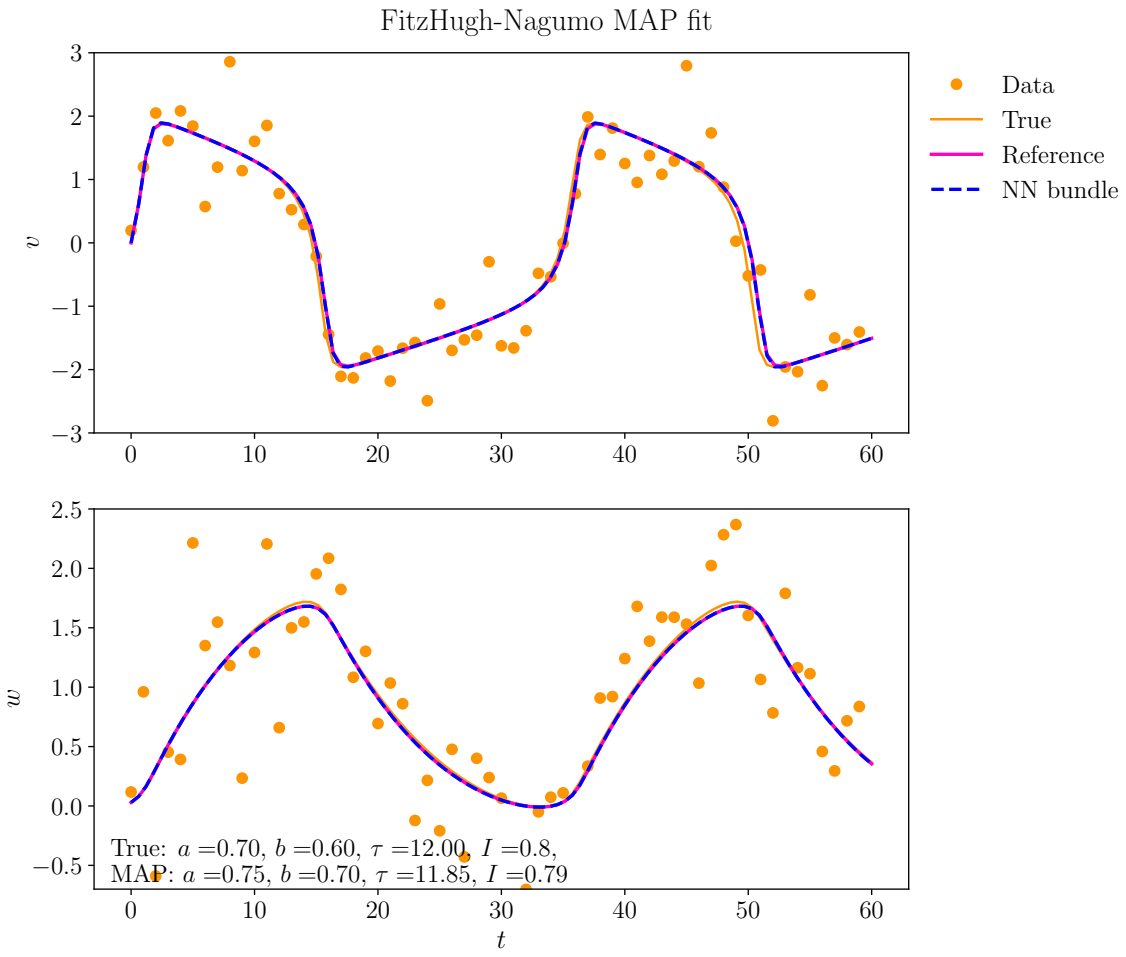

Figure 4.14: FitzHugh-Nagumo model fit corresponding to maximum a posteriori estimate, given the data and uniform prior.

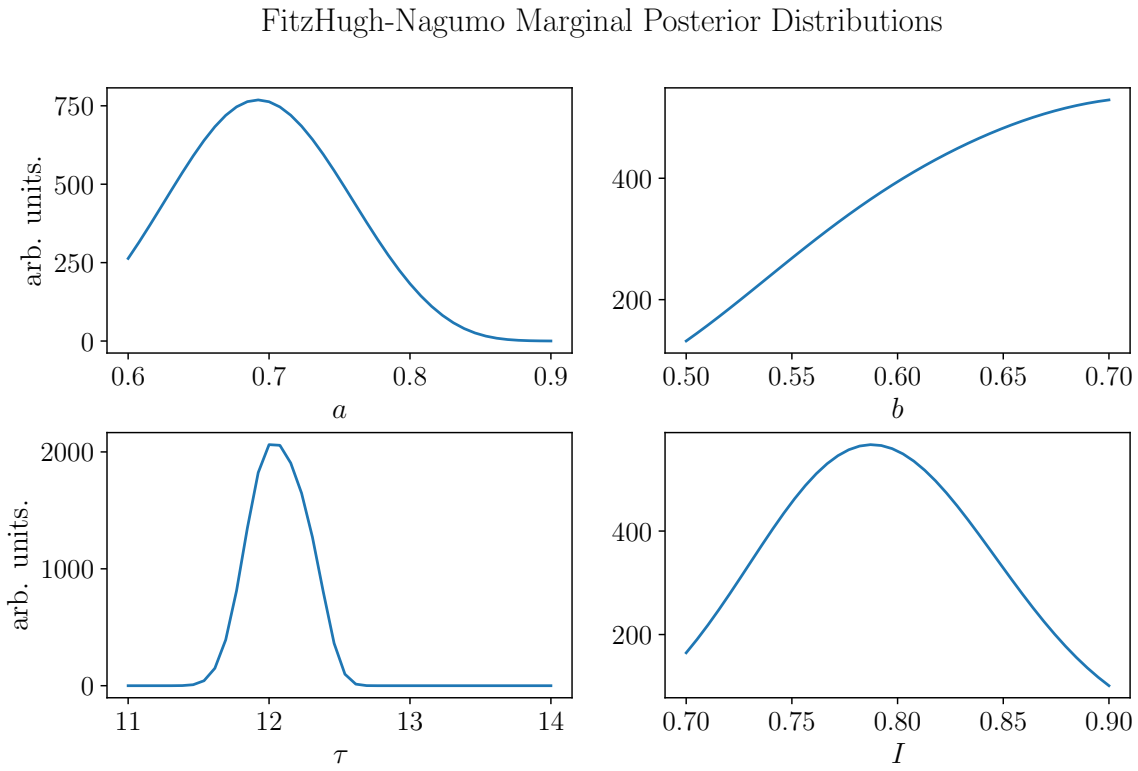

Figure 4.15: FitzHugh-Nagumo model parameters a, b, τ, and I marginal distributions given the data shown in Figure 4.14.

4.5 Conclusion

Backpropagation and the universal approximation theorem grants neural networks a unique ability to adapt and become any function that they need to be. Lagaris proposed an unsupervised method of training neural networks to approximate the solution to a set of differential equations for given boundary conditions, resulting in a memory-efficient representation of the solution with closed analytic form that is differentiable and parallelizable. We extend this method by introducing the concept of a neural network solution bundle, a group of solutions over a range of initial conditions and parameters. This allows for greater reuse of the trained network since it learns a variety of solutions. In addition, the solution bundle is differentiable in initial conditions and parameters, which can be useful for optimization tasks dependent on the value of the solution at given times. Other tasks that would require solving the differential equations repeatedly are also simplified, such as the propagation of uncertainty distributions across initial states, and for Bayesian inference in dynamical systems. In addition, this method may find some use in the study of chaotic systems where bundles of trajectories are often considered.

While the number of calculations involved in the training of neural network solution bundles is substantially higher than for computing a single solution using conventional methods, the cost can eventually be recouped if enough individual solutions are required, especially if the trained network is shared with other users. In addition, future advances in neural network training and evaluation, fueled by the general interest in these approaches across a wide range of disciplines, will directly benefit this method.

While we have investigated a few architectures, weighting functions, losses, and training approaches, the method still has ample room to grow. Specially tailored network architectures and losses are a clear way forward for improving the efficiency and performance of neural network solution bundles.

Bibliography

[1] M. Aichinger and E. Krotscheck. A fast configuration space method for solving local kohnsham equations. *Comput. Mater. Sci.*, **34**, 188 (2005).

[2] P.-M. Anglade and X. Gonze. Preconditioning of self-consistent-field cycles in density-functional theory: The extrapolar method. *Phys. Rev. B*, **78**, 045126 (2008).

[3] P. Bader, S. Blanes, and F. Casas. Solving the schrödinger eigenvalue problem by the imaginary time propagation technique using splitting methods with complex coefficients. *J. Chem. Phys.*, **139**, 124117 (2013).

[4] V. A. Basiuk. Electron smearing in dft calculations: A case study of doxorubicin interaction with single-walled carbon nanotubes. *Int. J. Quantum Chem.*, **111**, 4197 (2011).

[5] A. C. Bassi. Dynamic modulus of rubber by impact and rebound measurements. *Polymer Engineering and Science*, **18**, 750 (1978).

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars. arXiv:1604.07316 [cs] (2016). arXiv: 1604.07316.

[8] H. Bruus and K. Flensberg. *Many-body quantum theory in condensed matter physics: an introduction*. Oxford graduate texts. Oxford University Press, Oxford ; New York, 2004. OCLC: ocm56694794.

[9] D. M. Ceperley and B. J. Alder. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.*, **45**, 566 (1980).

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303 (1989).

[11] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc., 2014.

[12] K. T. R. Davies, H. Flocard, S. Krieger, and M. S. Weiss. Application of the imaginary time step method to the solution of the static Hartree-Fock problem. *Nucl. Phys. A*, **342**, 111 (1980).

[13] T. Ding, D. Li, and R. Sun. Sub-Optimal Local Minima Exist for Almost All Over-parameterized Neural Networks. arXiv:1911.01413 [cs, math, stat] (2020). arXiv: 1911.01413.

[14] R. M. Dreizler and E. K. Gross. *Density functional theory: an approach to the quantum many body problem*. Springer, Berlin, 1995. OCLC: 256760395.

[15] J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, **12**, 257 (2010).

[16] R. Eldan and O. Shamir. The Power of Depth for Feedforward Neural Networks. arXiv:1512.03965 [cs, stat] (2016). arXiv: 1512.03965.

[17] R. Fitzhugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal*, **1**, 445 (1961).

[18] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193 (1980).

[19] Gallant and White. There exists a neural network that does not make avoidable mistakes. In *IEEE 1988 International Conference on Neural Networks*, pages 657–664 vol.1, July 1988.

[20] S. Geman, E. Bienenstock, and R. Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, **4**, 1 (1992).

[21] E. Gross and W. Kohn. Time-Dependent Density-Functional Theory. In *Advances in Quantum Chemistry*, volume 21, pages 255–291. Elsevier, 1990.

[22] S. Hamel, P. Duffy, M. E. Casida, and D. R. Salahub. KohnSham orbitals and orbital energies: fictitious constructs but good approximations all the same. *Journal of Electron Spectroscopy and Related Phenomena*, **123**, 345 (2002).

[23] E. R. Hernández, S. Janecek, M. Kaczmarski, and E. Krotscheck. Evolution-operator method for density functional theory. *Phys. Rev. B*, **75**, 075108 (2007).

[24] P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Physical Review*, **136**, B864 (1964).

[25] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, **136**, B864 (1964).

[26] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251 (1991).

[27] R. A. Hoyt, M. M. Montemore, and E. Kaxiras. Nonadiabatic hydrogen dissociation on copper nanoclusters. *J. Phys. Chem. Lett.*, **9**, 5339 (2018). PMID: 30145896.

[28] J. F. Janak. Proof that E n i = in density-functional theory. *Physical Review B*, **18**, 7165 (1978).

[29] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. arXiv:1412.2306 [cs] (2015). arXiv: 1412.2306.

[30] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] (2017). arXiv: 1412.6980.

[31] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, **140**, A1133 (1965).

[32] G. Kolesov, O. Grånäs, R. Hoyt, D. Vinichenko, and E. Kaxiras. Real-time TD-DFT with classical ion dynamics: Methodology and applications. *J. Chem. Theory Comput.*, **12**, 466 (2016).

[33] G. Kolesov, E. Kaxiras, and E. Manousakis. Density functional theory beyond the born-oppenheimer approximation: Accurate treatment of the ionic zero-point motion. *Phys. Rev. B*, **98**, 195112 (2018).

[34] T. Koopmans. ber die Zuordnung von Wellenfunktionen und Eigenwerten zu den Einzelnen Elektronen Eines Atoms. *Physica*, **1**, 104 (1934).

[35] T. Kowalczyk, S. R. Yost, and T. Van Voorhis. Assessment of the ΔSCF density functional theory approach for electronic excitations in organic dyes. *J. Chem. Phys.*, **134**, 054128 (2011).

[36] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, **54**, 11169 (1996).

[37] G. Kresse and J. Furthmller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Comput. Mater. Sci.*, **6**, 15 (1996).

[38] I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, **9**, 987 (1998).

[39] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, **1**, 541 (1989).

[40] L. Lehtovaara, J. Toivanen, and J. Eloranta. Solution of time-independent schrödinger equation by the imaginary time propagation method. *J. Comput. Phys.*, **221**, 148 (2007).

[41] J.-L. Lions, Y. Maday, and G. Turinici. Rsolution d'EDP par un schma en temps pararel . *Comptes Rendus de l'Acadmie des Sciences - Series I - Mathematics*, **332**, 661 (2001).

[42] Z. C. Lipton, J. Berkowitz, and C. Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv:1506.00019 [cs] (2015). arXiv: 1506.00019.

[43] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The Expressive Power of Neural Networks: A View from the Width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.

[44] M. Lukoeviius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, **3**, 127 (2009).

[45] M. Marques, N. Maitra, F. Nogueira, E. Gross, and A. Rubio. *Fundamentals of Time-Dependent Density Functional Theory*, volume 837. 01 2012.

[46] M. Mattheakis, D. Sondak, A. S. Dogra, and P. Protopapas. Hamiltonian Neural Networks for solving differential equations. arXiv:2001.11107 [physics] (2020). arXiv: 2001.11107.

[47] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, **5**, 115 (1943).

[48] N. D. Mermin. Thermal properties of the inhomogeneous electron gas. *Phys. Rev.*, **137**, A1441 (1965).

[49] M. C. Michelini, R. Pis Diez, and A. H. Jubert. A density functional study of small nickel clusters. *Int. J. Quantum Chem.*, **70**, 693 (1998).

[50] M. Minsky and S. A. Papert. *Perceptrons: an introduction to computational geometry*. The MIT Press, Cambridge/Mass., 2. print. with corr edition, 1972. OCLC: 833070641.

[51] D. Mishra, A. Yadav, and P. K. Kalra. Chaotic Behavior in Neural Networks and FitzHugh-Nagumo Neuronal Model. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, N. R. Pal, N. Kasabov, R. K. Mudi, S. Pal, and S. K. Parui, editors, *Neural Information Processing*, volume 3316, pages 868–873. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[52] M. M. Montemore, R. Hoyt, G. Kolesov, and E. Kaxiras. Reaction-induced excitations and their effect on surface chemistry. *ACS Catal.*, **8**, 10358 (2018).

[53] J. Nagumo, S. Arimoto, and S. Yoshizawa. An Active Pulse Transmission Line Simulating Nerve Axon. *Proceedings of the IRE*, **50**, 2061 (1962).

[54] B. Neal. On the Bias-Variance Tradeoff: Textbooks Need an Update. arXiv:1912.08286 [cs, stat] (2019). arXiv: 1912.08286.

[55] R. Pascanu, Y. N. Dauphin, S. Ganguli, and Y. Bengio. On the saddle point problem for non-convex optimization. arXiv:1405.4604 [cs] (2014). arXiv: 1405.4604.

[56] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, **77**, 3865 (1996).

[57] V. Peuckert. A new approximation method for electron systems. *J. Phys. C*, **11**, 4945 (1978).

[58] P. Pulay. Convergence acceleration of iterative sequences. the case of SCF iteration. *Chem. Phys. Lett.*, **73**, 393 (1980).

[59] P. Pulay. Improved scf convergence acceleration. *J. Comput. Chem.*, **3**, 556 (1982).

[60] A. D. Rabuck and G. E. Scuseria. Improving self-consistent field convergence by varying occupation numbers. *J. Chem. Phys.*, **110**, 695 (1999).

[61] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. page 24 (2019).

[62] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386 (1958).

[63] E. Runge and E. K. U. Gross. Density-Functional Theory for Time-Dependent Systems. *Physical Review Letters*, **52**, 997 (1984).

[64] E. Runge and E. K. U. Gross. Density-Functional Theory for Time-Dependent Systems. *Phys. Rev. Lett.*, **52**, 997 (1984).

[65] W. Ryssens, V. Hellemans, M. Bender, and P.-H. Heenen. Solution of the Skyrme-HF+BCS equation on a 3D mesh, II: A new version of the Ev8 code. *Comput. Phys. Commun.*, **187**, 175 (2015).

[66] V. R. Saunders and I. H. Hillier. A "level-shifting" method for converging closed shell hartree-fock wave functions. *Int. J. Quantum Chem.*, **7**, 699 (1973).

[67] H. Schulz and S. Behnke. Deep Learning: Layer-Wise Learning of Feature Hierarchies. *KI - Knstliche Intelligenz*, **26**, 357 (2012).

[68] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: from theory to algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[69] J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal. The SIESTA method for *ab initio* order-N materials simulation. *J. Phys.: Condens. Matter*, **14**, 2745 (2002).

[70] S. Sonoda and N. Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, **43**, 233 (2017).

[71] I. Štich, R. Car, M. Parrinello, and S. Baroni. Conjugate gradient minimization of the energy functional: A new method for electronic structure calculation. *Phys. Rev. B*, **39**, 4997 (1989).

[72] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[73] G. Swirszcz, W. M. Czarnecki, and R. Pascanu. Local minima in training of neural networks. arXiv:1611.06310 [cs, stat] (2017). arXiv: 1611.06310.

[74] V. Szebehely. *Theory of orbits*. Acad. Pr, New York, 1967. OCLC: 164462279.

[75] E. Sli and D. F. Mayers. *An introduction to numerical analysis*. Cambridge University Press, Cambridge ; New York, 2003. OCLC: ocm50525488.

[76] M. Telgarsky. Benefits of depth in neural networks. arXiv:1602.04485 [cs, stat] (2016). arXiv: 1602.04485.

[77] S. Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Elsevier, AP, Amsterdam Boston Heidelberg London New York Oxford Paris San Diego San Francisco Singapore Sydney Tokyo, 2015. OCLC: 910913108.

[78] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, pages 303–314, Gothenburg Sweden, May 2018. ACM.

[79] C. A. Ullrich. *Time-Dependent Density-Functional Theory: Concepts and Applications*. Oxford University Press, 2012.

[80] C. A. Ullrich and Z.-h. Yang. A brief compendium of time-dependent density functional theory. *Brazilian Journal of Physics*, **44**, 154 (2014).

[81] G. J. G. Upton and I. Cook. *A dictionary of statistics*. Oxford paperback reference. Oxford University Press, Oxford ; New York, 2nd ed., rev edition, 2008. OCLC: ocn191929569.

[82] R. van Leeuwen. Mapping from Densities to Potentials in Time-Dependent Density-Functional Theory. *Physical Review Letters*, **82**, 3863 (1999).

[83] R. van Leeuwen. Key concepts in time-dependent density-functional theory. *Int. J. Mod. Phys. B*, **15**, 1969 (2001).

[84] M. Verstraete and X. Gonze. Smearing scheme for finite-temperature electronic-structure calculations. *Phys. Rev. B*, **65**, 035111 (2001).

[85] T. V. Voorhis and M. Head-Gordon. A geometric approach to direct minimization. *Mol. Phys.*, **100**, 1713 (2002).

[86] V. Weber, J. VandeVondele, J. Hutter, and A. M. N. Niklasson. Direct energy functional minimization under orthogonality constraints. *J. Chem. Phys.*, **128**, 084113 (2008).

[87] P. J. Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. Adaptive and learning systems for signal processing, communications, and control. Wiley, New York, 1994.

[88] E. P. Wigner. On the Distribution of the Roots of Certain Symmetric Matrices. *The Annals of Mathematics*, **67**, 325 (1958).

[89] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. arXiv:1611.03530 [cs] (2017). arXiv: 1611.03530.

[90] X. Zhang and Y. LeCun. Text Understanding from Scratch. arXiv:1502.01710 [cs] (2016). arXiv: 1502.01710.

[91] A. Zhumekenov, M. Uteuliyeva, O. Kabdolov, R. Takhanov, Z. Assylbekov, and A. J. Castro. Fourier Neural Networks: A Comparative Study. arXiv:1902.03011 [cs] (2019). arXiv: 1902.03011.

A

Janak's Theorem

We want to determine the meaning of the quantity ε_i in the Density-Functional Theory single-particle equations Eq. (1.34). To do this, we express the density as:

$$n(\mathbf{r}) = \sum_i f_i |\phi_i(\mathbf{r})|^2$$

where the f_i are real numbers between 0 and 1, called the "filling factors". By taking the partial derivative of the total energy with respect to f_i, we can find a relationship to ε_i.

The partial derivative is to be taken with the understanding that the orbitals relax under the influence of the change in the filling factor. In other words, the Kohn-Sham single-particle equations are to maintain self-consistency in the presence of the variation. This means that partial derivatives of the single-particle wavefunctions themselves have to be considered. Furthermore, since the density depends on the filling factor as well, we will have to apply the chain rule to the functionals too, given that they depend on the density.

The following proof is based on the paper by J. F. Janak, "Proof that $\partial E/\partial n_i = \varepsilon_i$ in density-functional theory"[28]. The resulting physical interpretation is generally referred to as "Janak's theorem" in the literature, and is a close analogue of Koopman's theorem[34] from Hartree-Fock theory.

First, the total energy in DFT can be written as

$$\tilde{E}^{\mathrm{KS}}[n] = \sum_i f_i t_i + \mathcal{E}^{\mathrm{ext}}[n] + \mathcal{E}^{\mathrm{H}}[n] + \mathcal{E}^{\mathrm{XC}}[n], \tag{A.1}$$

where

$$t_i \equiv \langle \phi_i | -\frac{\hbar^2 \nabla^2}{2m} | \phi_i \rangle$$

is defined to be the single particle kinetic energy, $\mathcal{V}(\mathbf{r})$ is the external potential, $\mathcal{E}^{\mathrm{H}}[n]$ is the Hartree

contribution,

$$\mathcal{E}^{\mathrm{H}}[n] \equiv \frac{e^2}{2}\int \mathrm{d}\mathbf{r}\int \mathrm{d}\mathbf{r}'\,\frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|},$$

$\mathcal{E}^{\mathrm{ext}}$ is the energy due to the density interacting with the external potential,

$$\mathcal{E}^{\mathrm{ext}}[n] \equiv \int \mathcal{V}(\mathbf{r})n(\mathbf{r})\,\mathrm{d}\mathbf{r},$$

and $\mathcal{E}^{\mathrm{XC}}[n]$ is the exchange-correlation energy. Notice that we put a tilde on top of $\tilde{E}^{\mathrm{KS}}$ since it is a generalization of the Hohenberg-Kohn total energy E^{KS} which is only defined for integral total number of electrons. This generalization allows for a continuous connection between the ground state energies af N and $N+1$ particle systems.

Let us begin by deriving a few expressions that will prove useful to us. First, we find that the functional derivative of the Hartree contribution to the energy with respect to the density is the Hartree potential:

$$\begin{aligned}
\frac{\delta\mathcal{E}^{\mathrm{H}}[n]}{\delta n(\mathbf{x})} &= \frac{e^2}{2}\int \mathrm{d}\mathbf{r}\int \mathrm{d}\mathbf{r}'\,\frac{\delta}{\delta n(\mathbf{x})}\left[\frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\right] \\
&= \frac{e^2}{2}\int \mathrm{d}\mathbf{r}\int \mathrm{d}\mathbf{r}'\,\frac{1}{|\mathbf{r}-\mathbf{r}'|}\left[\frac{\delta n(\mathbf{r})}{\delta n(\mathbf{x})}n(\mathbf{r}')+n(\mathbf{r})\frac{\delta n(\mathbf{r}')}{\delta n(\mathbf{x})}\right] \\
&= \frac{e^2}{2}\int \mathrm{d}\mathbf{r}\int \mathrm{d}\mathbf{r}'\,\frac{1}{|\mathbf{r}-\mathbf{r}'|}\left[\delta(\mathbf{r}-\mathbf{x})n(\mathbf{r}')+n(\mathbf{r})\delta(\mathbf{r}'-\mathbf{x})\right] \\
&= \frac{e^2}{2}\int \mathrm{d}\mathbf{r}'\,\frac{n(\mathbf{r}')}{|\mathbf{x}-\mathbf{r}'|}+\frac{e^2}{2}\int \mathrm{d}\mathbf{r}\,\frac{n(\mathbf{r})}{|\mathbf{r}-\mathbf{x}|} \\
&= e^2\int \frac{n(\mathbf{r}')}{|\mathbf{x}-\mathbf{r}'|} \\
&\equiv \mathcal{V}^{\mathrm{H}}[n](\mathbf{x}).
\end{aligned} \tag{A.2}$$

Similarly,

$$\frac{\delta\mathcal{E}^{\mathrm{ext}}[n]}{\delta n(\mathbf{r})} = \frac{\delta}{\delta n(\mathbf{r})}\left[\int \mathcal{V}(\mathbf{r}')n(\mathbf{r}')\,\mathrm{d}\mathbf{r}'\right] = \int \mathcal{V}(\mathbf{r}')\frac{\delta n(\mathbf{r}')}{\delta n(\mathbf{r})}\,\mathrm{d}\mathbf{r}' = \int \mathcal{V}(\mathbf{r}')\delta(\mathbf{r}'-\mathbf{r})\,\mathrm{d}\mathbf{r} = \mathcal{V}(\mathbf{r}). \tag{A.3}$$

Recall as well that the functional derivative of the exchange-correlation energy is defined to be the exchange correlation potential,

$$\frac{\delta\mathcal{E}^{\mathrm{XC}}}{\delta n(\mathbf{r})} \equiv \mathcal{V}^{\mathrm{XC}}[n](\mathbf{r}). \tag{A.4}$$

We can set the variation of total energy to zero while constraining the single-particle functions to be normalized in order to obtain the Kohn-Sham equations:

$$\delta\left[\tilde{E}^{\mathrm{KS}}-\sum_j f_j\varepsilon_j\left(\langle\phi_j|\phi_j\rangle-1\right)\right]=0$$

$$\sum_i\int\frac{\delta}{\delta\phi_i^*(\mathbf{r})}\left[\tilde{E}^{\mathrm{KS}}-\sum_j f_j\varepsilon_j\left(\langle\phi_j|\phi_j\rangle-1\right)\right]\delta\phi_i^*(\mathbf{r})\,\mathrm{d}\mathbf{r}=0,$$

where we have preemptively chosen our set of Lagrange multipliers to be $\{f_j\varepsilon_j\}$ in order to obtain generalized Kohn-Sham equations that agree with the standard approach when f_j are integral. Since the above variation has to hold for arbitrary $\delta\phi_i^*(\mathbf{r})$, we have that every one of these functional derivatives vanishes,

$$0=\frac{\delta}{\delta\phi_i^*(\mathbf{r})}\left[\tilde{E}^{\mathrm{KS}}-\sum_j f_j\varepsilon_j\left(\langle\phi_j|\phi_j\rangle-1\right)\right]$$

$$0=\sum_j f_j\frac{\delta}{\delta\phi_i^*(\mathbf{r})}\int\phi_j^*(\mathbf{r}')\frac{-\hbar^2\nabla^2}{2m}\phi_j(\mathbf{r}')\,\mathrm{d}\mathbf{r}'+\int\left[\frac{\delta\mathcal{E}^{\mathrm{ext}}}{\delta n(\mathbf{r}')}+\frac{\delta\mathcal{E}^{\mathrm{H}}}{\delta n(\mathbf{r}')}+\frac{\delta\mathcal{E}^{\mathrm{XC}}}{\delta n(\mathbf{r}')}\right]\frac{\delta n(\mathbf{r}')}{\delta\phi_i^*}\,\mathrm{d}\mathbf{r}'$$
$$-\sum_j f_j\varepsilon_j\frac{\delta}{\delta\phi_i^*(\mathbf{r})}\left[\int\phi_j^*(\mathbf{r}')\phi_j(\mathbf{r}')-1\right]$$

$$0=\sum_j f_j\int\delta_{ij}\delta(\mathbf{r}'-\mathbf{r})\frac{-\hbar^2\nabla^2}{2m}\phi_j(\mathbf{r}')\,\mathrm{d}\mathbf{r}'+\int\left[\mathcal{V}(\mathbf{r}')+\mathcal{V}^{\mathrm{H}}(\mathbf{r}')+\mathcal{V}^{\mathrm{XC}}(\mathbf{r}')\right]\frac{\delta}{\delta\phi_i^*(\mathbf{r})}\left[\sum_j f_j\phi_j^*(\mathbf{r}')\phi_j^*(\mathbf{r}')\right]\mathrm{d}\mathbf{r}'$$
$$-\sum_j f_j\varepsilon_j\int\delta_{ij}\delta(\mathbf{r}'-\mathbf{r})\phi_j(\mathbf{r}')$$

$$0=f_i\frac{-\hbar^2\nabla^2}{2m}\phi_i(\mathbf{r})+f_i\left[\mathcal{V}(\mathbf{r})+\mathcal{V}^{\mathrm{H}}(\mathbf{r})+\mathcal{V}^{\mathrm{XC}}(\mathbf{r})\right]\phi_i(\mathbf{r})-f_i\varepsilon_i\phi_i(\mathbf{r}),$$

giving us the Kohn-Sham equations, which we note are completely identical to the original case with integral filling factors:

$$\left[\frac{-\hbar^2\nabla^2}{2m}+\mathcal{V}^{\mathrm{eff}}(\mathbf{r})\right]\phi_i(\mathbf{r})=\varepsilon_i\phi_i(\mathbf{r}),\tag{A.5}$$

where we define the single-particle effective potential as $\mathcal{V}^{\mathrm{eff}}(\mathbf{r})=\mathcal{V}(\mathbf{r})+\mathcal{V}^{\mathrm{H}}(\mathbf{r})+\mathcal{V}^{\mathrm{XC}}(\mathbf{r})$.

Now, differentiating equation A.1 with respect to f_i, we find,

$$\begin{aligned}\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} &= t_i + \sum_j f_j \frac{\partial t_j}{\partial f_i} + \int \left[\frac{\delta \mathcal{E}^{\mathrm{ext}}}{\delta n(\mathbf{r}')} + \frac{\delta \mathcal{E}^{\mathrm{H}}}{\delta n(\mathbf{r}')} + \frac{\delta \mathcal{E}^{\mathrm{XC}}}{\delta n(\mathbf{r}')} \right] \frac{\partial n(\mathbf{r}')}{\partial f_i} \, \mathrm{d}\mathbf{r}' \\ &= t_i + \sum_j f_j \frac{\partial t_j}{\partial f_i} + \int \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \left[\left| \phi_i(\mathbf{r}') \right|^2 + \sum_j f_j \frac{\partial \left| \phi_j(\mathbf{r}') \right|^2}{\partial f_i} \right] \mathrm{d}\mathbf{r}' .\end{aligned}$$

Note that by multiplying by $\phi_i^*(\mathbf{r})$ and integrating the Kohn-Sham equation A.5, we obtain the relation

$$t_i = \varepsilon_i - \langle \phi_i | \mathcal{V}^{\mathrm{eff}} | \phi_i \rangle ,$$

which we can plug in to our above expression,

$$\begin{aligned}\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} &= \varepsilon_i - \langle \phi_i | \mathcal{V}^{\mathrm{eff}} | \phi_i \rangle + \sum_j f_j \frac{\partial t_j}{\partial f_i} + \langle \phi_i | \mathcal{V}^{\mathrm{eff}} | \phi_i \rangle + \sum_j f_j \int \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \frac{\partial \left| \phi_j(\mathbf{r}') \right|^2}{\partial f_i} \, \mathrm{d}\mathbf{r}' \\ &= \varepsilon_i + \sum_j f_j \frac{\partial}{\partial f_i} \left[\int \phi_j^*(\mathbf{r}') \frac{-\hbar^2 \nabla^2}{2m} \phi_j(\mathbf{r}') \, \mathrm{d}\mathbf{r}' \right] + \sum_j f_j \int \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \frac{\partial \left| \phi_j(\mathbf{r}') \right|^2}{\partial f_i} \, \mathrm{d}\mathbf{r}' \\ &= \varepsilon_i + \sum_j f_j \int \mathrm{d}\mathbf{r}' \left[\frac{\partial \phi_j^*(\mathbf{r}')}{\partial f_i} \frac{-\hbar^2 \nabla^2}{2m} \phi_j(\mathbf{r}') + \frac{\partial \phi_j^*(\mathbf{r}')}{\partial f_i} \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \phi_j(\mathbf{r}') \right. \\ &\qquad \left. + \phi_j^*(\mathbf{r}') \frac{-\hbar^2 \nabla^2}{2m} \frac{\partial \phi_j(\mathbf{r}')}{\partial f_i} + \phi_j^*(\mathbf{r}') \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \frac{\partial \phi_j(\mathbf{r}')}{\partial f_i} \right] .\end{aligned}$$

Consider the last two terms in the integral. Recall that $\langle \phi | M | \psi \rangle^* = \langle \psi | M^\dagger | \phi \rangle$, but if M is an observable, it's Hermitian and hence $M = M^\dagger$. So, regarding those terms, we see that

$$\left\langle \phi_j \middle| \left[\frac{-\hbar^2 \nabla^2}{2m} + \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \right] \middle| \frac{\partial \phi_j}{\partial f_i} \right\rangle = \left\langle \frac{\partial \phi_j}{\partial f_i} \middle| \left[\frac{-\hbar^2 \nabla^2}{2m} + \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \right] \middle| \phi_j \right\rangle^* .$$

This can also be shown without appealing to the fact that the operator is an observable by swapping which wavefunction the Laplacian operates on by using integration by parts twice. The resulting surface integrals vanish since the wavefunctions and their derivatives go to zero at infinity, and we again recover the above

equality. So, in our expression we now have

$$\begin{aligned}
\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} &= \varepsilon_i + \sum_j f_j \int \mathrm{d}\mathbf{r}' \left[\frac{\partial \phi_j^*(\mathbf{r}')}{\partial f_i} \left(\frac{-\hbar^2 \nabla^2}{2m} + \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \right) \phi_j(\mathbf{r}') + \mathrm{c.c.} \right] \\
&= \varepsilon_i + \sum_j f_j \int \mathrm{d}\mathbf{r}' \left[\frac{\partial \phi_j^*(\mathbf{r}')}{\partial f_i} \varepsilon_j \phi_j(\mathbf{r}') + \mathrm{c.c.} \right] \\
&= \varepsilon_i + \sum_j f_j \varepsilon_j \int \mathrm{d}\mathbf{r}' \left[\frac{\partial \phi_j^*(\mathbf{r}')}{\partial f_i} \phi_j(\mathbf{r}') + \phi_j^*(\mathbf{r}') \frac{\partial \phi_j(\mathbf{r}')}{\partial f_i} \right] \\
&= \varepsilon_i + \sum_j f_j \varepsilon_j \frac{\partial}{\partial f_i} \int \left| \phi_j(\mathbf{r}') \right|^2 \mathrm{d}\mathbf{r}' \\
&= \varepsilon_i,
\end{aligned}$$

where to get the second equality we have used the Kohn-Sham equations (eq. A.5), and in the last equation we have noticed that the derivative term vanishes since the normalization of the single-particle orbitals is held constant in the variation. This result is called Janak's theorem.

In principle, $\varepsilon_i(f_i)$ will depend on f_i. Integrating the derived relationship from $f_i = 0$ to $f_i = 1$, we obtain

$$E^{\mathrm{KS}}\big|_{f_i=1} - E^{\mathrm{KS}}\big|_{f_i=0} = \int_0^1 \frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} \, \mathrm{d}f_i = \int_0^1 \varepsilon_i(f_i) \, \mathrm{d}f_i \,, \tag{A.6}$$

that is, this integral gives you the difference in energy between the system with this i^{th} particle and without it. In an extended system with many electrons, we can assume that ε_i is weakly dependent on the presence of this one electron, and hence

$$E^{\mathrm{KS}}\big|_{f_i=1} - E^{\mathrm{KS}}\big|_{f_i=0} \approx \int_0^1 \varepsilon_i \, \mathrm{d}f_i = \varepsilon_i.$$

So, we can see that this theorem is the DFT analogue of Koopman's theorem from Hartree-Fock.

Alternative Derivation

Instead of applying the chain rule to everything, Janak's theorem can be proven in a different way that is a bit shorter. First, think of the Kohn-Sham total energy as an explicit function of f_i, and a functional of $\{\phi_j\}$, the set of single-particle wavefunctions, which implicitly depend on f_i through the density, through maintaining the self-consistency of the Kohn-Sham equations.

Then, when computing the partial derivative with respect to f_i,

$$\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} = \left.\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i}\right|_{\{\phi_j\}} + \sum_j \int \frac{\delta \tilde{E}^{\mathrm{KS}}}{\delta \phi_j^*(\mathbf{r})} \frac{\partial \phi_j^*(\mathbf{r})}{\partial f_i}\,\mathrm{d}\mathbf{r} + \sum_j \int \frac{\delta \tilde{E}^{\mathrm{KS}}}{\delta \phi_j(\mathbf{r})} \frac{\partial \phi_j(\mathbf{r})}{\partial f_i}\,\mathrm{d}\mathbf{r},$$

where the first term is a partial derivative taken with all ϕ_j held fixed (treated as constants) and the last two terms arise from the change in the single-particle wavefunctions due to the variation of f_i. However, recall that we are allowing the system to "relax" in the presence of the variation in f_i, i.e. self-consistency of the Kohn-Sham equations is always maintained. This implies that

$$\frac{\delta \tilde{E}^{\mathrm{KS}}}{\delta \phi_j^*(\mathbf{r})} = 0,$$

recalling that this condition is equivalent to the Kohn-Sham equations, and was used in their derivation. Same goes for the complex conjugate of this equation. Thus, the last two terms vanish, and we can just consider the first derivative.

$$\begin{aligned}
\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i} = \left.\frac{\partial \tilde{E}^{\mathrm{KS}}}{\partial f_i}\right|_{\{\phi_j\}} &= t_i + \int \left[\frac{\delta \mathcal{E}^{\mathrm{ext}}}{\delta n(\mathbf{r}')} + \frac{\delta \mathcal{E}^{\mathrm{H}}}{\delta n(\mathbf{r}')} + \frac{\delta \mathcal{E}^{\mathrm{XC}}}{\delta n(\mathbf{r}')}\right] \left.\frac{\partial n(\mathbf{r}')}{\partial f_i}\right|_{\{\phi_j\}} \mathrm{d}\mathbf{r}' \\
&= t_i + \int \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \left.\frac{\partial}{\partial f_i}\right|_{\{\phi_j\}} \left[\sum_j f_j \left|\phi_j(\mathbf{r}')\right|^2\right] \mathrm{d}\mathbf{r}' \\
&= t_i + \int \mathcal{V}^{\mathrm{eff}}(\mathbf{r}') \left|\phi_i(\mathbf{r}')\right|^2 \mathrm{d}\mathbf{r}' \\
&= \langle \phi_i | \frac{-\hbar^2 \nabla^2}{2m} + \mathcal{V}^{\mathrm{eff}} | \phi_i \rangle \\
&= \varepsilon_i.
\end{aligned}$$

B

Supporting Information for Chapter 2

This appendix is part of the supporting information accompanying the publication

> **Cedric Flamant**, Grigory Kolesov, Efstratios Manousakis, and Efthimios Kaxiras. "Imaginary-Time Time-Dependent Density Functional Theory and Its Application for Robust Convergence of Electronic States." *J. Chem. Theory Comput.* **15**, 11, 6036-6045 (2019).

B.1 Cu_{13} Example Calculation Details

The Cu_{13} cluster we considered has an icosahedral geometry. We used the PBE functional and the basis set optimized in Hoyt *et al.*[27]. The calculations were performed in TDAP 2.0[32], built on SIESTA. A portion of the FDF file specifying information about the geometry, functional, and basis set is included below:

```
MeshCutoff 250 Ry

SpinPolarized .true.
ElectronicTemperature 0 K
FixSpin .true.
TotalSpin 1.0

NumberOfSpecies 1
XC.Functional GGA
XC.Authors    PBE
%block ChemicalSpeciesLabel
  1 29 Cu
%endblock ChemicalSpeciesLabel

LatticeConstant 14 Ang
%block LatticeVectors
 1 0 0
 0 1 0
 0 0 1
%endblock LatticeVectors

%block PAO.Basis
```

```
Cu     3    .2418041
 n=4   0   2   E  18.7173392  8.5020294
   6.5921851    -.3403688
   1.000      1.000
 n=3   2   2   E  11.0301035  4.9740702
   5.9448110    -.5297646
   1.000      1.000
 n=4   1   1   E  24.8348938  10.3126965
   7.0470756
   1.00
%endblock PAO.Basis

NumberOfAtoms 13
AtomicCoordinatesFormat Fractional
%block AtomicCoordinatesAndATomicSpecies
 0.500007106       0.499993714       0.500000377   1    #   1  Cu
 0.498453720       0.499899399       0.672486528   1    #   2  Cu
 0.546961887       0.353314971       0.577419774   1    #   3  Cu
 0.374562040       0.409424763       0.575952213   1    #   4  Cu
 0.374556508       0.590479864       0.576064955   1    #   5  Cu
 0.546978482       0.646579577       0.577605062   1    #   6  Cu
 0.653471782       0.499932033       0.578578947   1    #   7  Cu
 0.453002558       0.353433152       0.422382514   1    #   8  Cu
 0.346550672       0.500082456       0.421392974   1    #   9  Cu
 0.453069420       0.646698210       0.422586236   1    #  10  Cu
 0.625465641       0.590554051       0.424049472   1    #  11  Cu
 0.625422636       0.409477775       0.423927127   1    #  12  Cu
 0.501585326       0.500116633       0.327506892   1    #  13  Cu
%endblock AtomicCoordinatesAndATomicSpecies
```

B.2 Ru_{55} Example Calculation Details

The Ru_{55} cluster we considered has an icosahedral geometry. The PBE functional and an optimized double-ζ with polarization (DZP) basis set were used, with a different basis set for the surface and interior atoms. The geometry and optimization was performed in Montemore *et al.*[52]. A portion of the FDF file specifying information about the geometry, functional, and basis set is included below:

```
NumberOfSpecies 2
NumberOfAtoms 55
%block Chemical_Species_Label
 1 44 Ru_surf
 2 44 Ru_bulk
%endblock Chemical_Species_Label
```

```
SpinPolarized .false.

xc.authors PBE
xc.functional GGA

%block PAO.Basis
Ru_surf    3
 n=5   0   2   E  5.1969765  6.4102033
   4.8150212    -.5408860
   1.000      1.000
 n=4   2   2   E  16.9399325  6.4404240
   5.2358603    -.0750774
   1.000      1.000
 n=5   1   2   E  16.7403551  6.7658275
   10         7.4372332
   1.00        1.00
Ru_bulk    3
 n=5   0   2   E  5.1276155  6.2372922
   4.6336805    -.5019498
   1.000      1.000
 n=4   2   2   E  16.8979148  6.7782168
   5.0620075    -.0825330
   1.000      1.000
 n=5   1   1   E  16.5510519  8.0000000
   5.2883443
   1.00
%endblock PAO.Basis

MeshCutoff 100 Ry

AtomicCoordinatesFormat ScaledByLatticeVectors
AtomicCoorFormatOut Ang

LatticeConstant 1 Ang

%block LatticeVectors
 20.0 0. 0.
 0. 20.0 0.
 0. 0. 20.0
%endblock LatticeVectors

%block AtomicCoordinatesAndAtomicSpecies
 0.494804 0.500007 0.754716 1 #1
 0.531818 0.387295 0.687004 1 #2
 0.410028 0.424973 0.686811 1 #3
 0.410017 0.575087 0.686814 1 #4
 0.531834 0.612736 0.687130 1 #5
 0.618857 0.499567 0.692272 1 #6
 0.652489 0.388034 0.610319 1 #7
 0.570147 0.286443 0.615078 1 #8
 0.440980 0.318366 0.612740 1 #9
```

```
 0.317355 0.369750 0.615482 1 #10
 0.311707 0.499994 0.612505 1 #11
 0.317313 0.630272 0.615475 1 #12
 0.440962 0.681649 0.612854 1 #13
 0.570131 0.713687 0.615145 1 #14
 0.652220 0.611898 0.610677 1 #15
 0.723708 0.500056 0.614382 1 #16
 0.629303 0.320015 0.499170 1 #17
 0.500627 0.281215 0.499393 1 #18
 0.370644 0.321716 0.500795 1 #19
 0.290064 0.433509 0.501231 1 #20
 0.290010 0.566327 0.501383 1 #21
 0.370653 0.678355 0.500837 1 #22
 0.500601 0.718849 0.499401 1 #23
 0.629188 0.680249 0.499604 1 #24
 0.708311 0.570130 0.501427 1 #25
 0.708412 0.430056 0.501389 1 #26
 0.558355 0.323353 0.385439 1 #27
 0.431148 0.287819 0.385868 1 #28
 0.349136 0.392022 0.382369 1 #29
 0.273526 0.500057 0.388859 1 #30
 0.349114 0.607907 0.382214 1 #31
 0.430962 0.712244 0.385913 1 #32
 0.558311 0.676863 0.385428 1 #33
 0.682410 0.630778 0.384892 1 #34
 0.686205 0.500505 0.382166 1 #35
 0.682472 0.369486 0.384932 1 #36
 0.460825 0.392201 0.312008 1 #37
 0.384054 0.499990 0.310737 1 #38
 0.460723 0.607863 0.312029 1 #39
 0.595084 0.565565 0.312618 1 #40
 0.595145 0.434669 0.312557 1 #41
 0.502608 0.500116 0.246900 1 #42
 0.499994 0.500063 0.500858 2 #43
 0.501694 0.500126 0.630767 2 #44
 0.535385 0.390702 0.558326 2 #45
 0.407489 0.431573 0.557917 2 #46
 0.407472 0.568482 0.557922 2 #47
 0.535333 0.609418 0.558390 2 #48
 0.614486 0.500035 0.557437 2 #49
 0.463687 0.391452 0.442669 2 #50
 0.385160 0.500018 0.442307 2 #51
 0.463633 0.608608 0.442664 2 #52
 0.593374 0.568167 0.443528 2 #53
 0.593500 0.432368 0.443422 2 #54
 0.499439 0.500041 0.371322 2 #55
%endblock AtomicCoordinatesAndAtomicSpecies
```