Imitation Learning for Accelerating Iterative Computation of Fixed Points in Quantum Chemistry

Editor:

Abstract

Many computational methods in science and engineering work by fixed-point iteration—repeatedly applying a given update function until convergence. Unfortunately, this sort of iterative computation can be expensive: the update function itself may be costly to compute, and we may need many updates before we reach the desired fixed point. So, we explore a means to accelerate fixed-point iteration via imitation learning. Our approach is simple and appealing, and needs only black-box access to the original update function. We show in experiments that our approach successfully accelerates a central algorithm from quantum chemistry: the Hartree-Fock method for calculating the electronic structure and energy of a molecular system. Our results indicate that policies trained on one set of molecules transfer successfully to other molecules of the same general class. They also indicate that imitation learning leads to more-robust transfer compared to alternative methods that do not take into account the distribution of states induced by our learned policies.

1. INTRODUCTION

Many computational methods in science and engineering use fixed-point iteration ($x_{i+1} = f(x_i)$ for $i = 0, 1, 2, \ldots$) to attempt to find fixed points of f, i.e., points at which $f(x) = x$. These methods generate a trajectory of iterates, $x_0, x_1, x_2, \ldots$, that ideally converge to a fixed point x. This paper explores the application of imitation learning to fixed-point iteration to accelerate convergence and improve stability. Imitation learning, also called learning from demonstration, is designed to learn a policy that imitates an expert's method of performing a target task: we gather training data by posing task instances to the expert, and train a function approximator to mimic the expert's mapping from situations to actions.

The most straightforward way to apply imitation learning to fixed-point iteration would be to treat our original update function as the expert: train a cheaper function approximator to imitate the original, more expensive update. (There are two reasonable variants of this approach: in the first we need only a trajectory from our original fixed-point iteration, and we train our learner to produce the ith point on this trajectory at step i, no matter what the current point x_i is. In the second variant we need black-box access to the original update function f, and we train our learner to produce $f(x_i)$ when the current point is x_i.) Ross et al. (2011a) demonstrate that this type of imitation learning can be successful in calculating fixed points, specifically in the context of belief propagation for computing marginals of a posterior distribution.

However, scientific and engineering applications often have two important differences from typical uses of imitation learning, making the straightforward approach problematic.

First, to achieve a useful speedup, we do not need to completely eliminate calls to the original update function; it is enough to reduce their frequency. Second, considerable effort has often gone into developing and fine-tuning the original update function; simply replacing it with a naive function approximator would throw away much of this effort. In particular, the original update function often encodes physical constraints and intuitions that would be difficult to enforce with a naive function approximator; dropping these constraints and intuitions would lead to solutions that are not acceptable to domain experts, even if we were able to maintain or improve overall approximation error.

So instead of completely replacing the original update function, we propose to train our imitation learner to skip some of the steps from our original sequence of iterates: given the *output* of the update function at steps $1 \dots t$, the learner tries to predict a good *input* for the update function at step $t + \Delta$. By doing so, we replace two or more calls to the expensive update function with a single call, saving computation; but we still only task our learner with the easier job of producing a good input to the update function instead of mimicking its output. The difference is that, unlike the output, the input need not respect physical constraints or intuitions: we assume that the original update function itself is responsible for enforcing these, and is capable of repairing small violations.

As a test case for our methodology, we accelerate the Hartree-Fock method from computational quantum chemistry. Hartree-Fock is a fixed-point algorithm that approximates the electronic structure and energy of molecules. It is a specific instance of a broader class of mean-field-theory approaches to many-body problems. In such mean-field theories, the effect of all individuals on any given individual is approximated by a single averaged effect, thus reducing a many-body problem to a one-body problem. In quantum chemistry, the mean field arises from the averaged charge distribution of all electrons, as described by the electron density, $\rho(\mathbf{r})$. The Hartree-Fock iterations thereby generate a sequence of electron densities that ideally converge to a fixed point. Quantum chemical methods that go beyond mean-field theory typically begin with the results of a Hartree-Fock calculation, making the Hartree-Fock algorithm a pervasive component of quantum chemistry (Cook, 2005).

2. BACKGROUND

2.1 Hartree-Fock

The core problem of quantum chemistry is to compute the distribution of electrons in a molecule given the positions of its nuclei. The Hartree-Fock method efficiently approximates this distribution. From a computational standpoint, its relevant properties are:

- It is a fixed-point algorithm, whose iterates are single-electron density functions $\rho(\mathbf{r})$, represented as symmetric matrices, ρ, using a fixed set of basis functions $\{\chi_i(\mathbf{r})\}_{i=1}^{N_{\text{basis}}}$.
- The basis set size N_{basis} is typically at least linear in the number of nuclei in the molecule we are considering.
- Each iteration of Hartree-Fock is expensive: among other operations, we must iterate over all 4-tuples of basis functions to construct our mean-field approximation (nominally taking $O(N_{\text{basis}}^4)$ time, although scalings closer to $O(N_{\text{basis}}^3)$ are routinely achieved by taking advantage of sparsity (Cook, 2005)).

Hartree-Fock is shown as Algorithm 1, and a more detailed explanation is in Appendix A. The lines inside the **while** loop constitute the fixed point update, mapping ρ_i to ρ_{i+1}.

Data: Coordinates of atomic nuclei; basis set χ_i; number of electrons N; initial density matrix ρ_0; termination criterion $\delta > 0$
Result: Density matrix with approximately-minimum energy
Calculate overlap matrix: $S_{ij} \leftarrow \langle \chi_i, \chi_j \rangle$
Initialize $t \leftarrow 0$
while $t = 0$ *or* $|E_t - E_{t-1}| > \delta$ **do**
  Calculate the Fock matrix $F \leftarrow F(\rho_t)$
  Solve generalized eigenproblem $Fc = \epsilon Sc$ for eigenpairs c_a, ϵ_a
  Build matrix C: stack eigenvectors c_a side by side, two copies of each, starting from lowest energy ϵ_a, until we have added N columns
  Update density matrix $\rho_{t+1} = CC^\top$
  Update energy $E_{t+1} \leftarrow$ sum of ϵ_a for columns of C
  $t \leftarrow t + 1$
end

Algorithm 1: Hartree-Fock

A number of extensions to the base Hartree-Fock method have been proposed; most relevant to the current study is an approach called Direct Inversion in the Iterative Subspace, or DIIS (Pulay, 1980). In DIIS, we no longer take the output of iteration t directly as the input to iteration $t + 1$. Instead, at the beginning of iteration t, we build an input density matrix ρ'_t as a linear combination of the past several output density matrices $\rho_t, \rho_{t-1}, \rho_{t-2}, \ldots$, and use ρ'_t to calculate the Fock operator $\hat{F}$ in the first line of the **while** loop. The coefficients of the linear combination are chosen by minimizing some measure of the error of the resulting iterates. Various methods have been used to define the error measure and to find the weights that minimize it (Hu and Yang, 2010; Garza and Scuseria, 2012b,a). However, identifying weights that lead to the fastest and most stable convergence remains an open question (Kudin et al., 2010; Rohwedder and Schneider, 2011). The relevance of DIIS is that its overall form is the same as that of our proposed method: we train a function approximator (in this case linear regression) to produce a good input to our original fixed-point update step (in this case the Hartree-Fock update). In contrast to existing DIIS methods, however, we view the problem of training our function approximator as one of imitation learning: we explicitly take into account the effect of our learned weights, not just on the immediate error, but on the overall degree to which the fixed-point iteration tracks the behavior of our desired fixed-point iteration. By so doing, we hope to achieve faster and more stable convergence than existing methods.

2.2 Imitation learning

In imitation learning, we wish to discover how to solve a sequential decision problem by using expert demonstrations. We are given access to the expert's policy π^*, which is a (possibly randomized) mapping from states of the world s to actions $a = \pi^*(s)$. And, we are given access to a world model M: for any state s and action a, M is a (possibly randomized) mapping to the next state $s' = M(s, a)$. Using the policy and the world model

we can sample training trajectories: for each trajectory, we start in a designated start state $s_0 = \mathsf{start}$, execute action $a_0 = \pi^*(s_0)$, transition to state $s_1 = M(s_0, a_0)$, and repeat for T steps. We can then use these trajectories to train a predictor that, given the current state, predicts the expert's action. The key difficulty in imitation learning is that prediction errors early in a trajectory can cause us to deviate from the expert's distribution over states; so, we can easily drift into areas of the state space where we have little or no training data, causing a growing cascade of errors.

In our experiments, we use the DAgger algorithm for imitation learning (Ross et al., 2011b). DAgger (Algorithm 2) avoids the above difficulty by using its current hypothesized policy at each iteration to gather additional training trajectories, so that it gains experience on how to correct its own errors.

Data: Policy class Π, expert's policy π^*, horizon T, world model M
Result: Best $\hat{\pi}_j$ on validation
Initialize $D \leftarrow \emptyset$
Initialize $\hat{\pi}_1$ to any policy in Π
for *j=1 to N* **do**
  Let $\pi_j = \beta_j \pi^* + (1 - \beta_j)\hat{\pi}_j$
  Sample a trajectory $s_0, s_1, \ldots, s_T$ using π_j and M
  Construct dataset $D_j = \{(s_t, \pi^*(s_t))\}$ of visited states and their expert actions
  Aggregate datasets: $D \leftarrow D \cup D_j$
  Train policy $\hat{\pi}_{j+1}$ by supervised learning on D
end

Algorithm 2: DAgger

3. LEARNING THE POLICY

Table 1: Example: applying DAgger on expert demonstrations with step size = 2

			Hartree-Fock iterations (with step size = 2)				
DAgger iterations	iter		iter 1	iter 2	iter 3	...	iter x = $\frac{n}{2}$
	1	Objective	$(\rho_0) \to \rho_2$	$(\rho_0, \rho_2) \to \rho_4$	$(\rho_0, \rho_2, \rho_4) \to \rho_6$	...	$(\rho_{2i})_{i=0}^{x-1} \to \rho_n$
		Result	$(\rho_0) \to \rho'_2$	$(\rho_0, \rho_2) \to \rho'_4$	$(\rho_0, \rho_2, \rho_4) \to \rho'_6$	...	$(\rho_{2i})_{i=0}^{x-1} \to \rho'_n$
	2	New objective		$(\rho_0, \rho'_2) \to \rho_4$	$(\rho_0, \rho_2, \rho'_4) \to \rho_6$	...	$((\rho_{2i})_{i=0}^{x-2}, \rho'_{2(x-1)}) \to \rho_n$
		Result		$(\rho_0, \rho'_2) \to \rho''_4$	$(\rho_0, \rho_2, \rho'_4) \to \rho''_6$	...	$((\rho_{2i})_{i=0}^{x-2}, \rho'_{2(x-1)}) \to \rho''_n$
	3	New objective			$(\rho_0, \rho'_2, \rho''_4) \to \rho_6$	...	$((\rho_{2i})_{i=0}^{x-3}, (\rho_{2i}^{[i-(x-3)]})_{i=x-2}^{x-1}) \to \rho_n$
		Result			$(\rho_0, \rho'_2, \rho''_4) \to \rho'''_6$	...	$((\rho_{2i})_{i=0}^{n-3}, (\rho_{2i}^{[i-(x-3)]})_{i=x-2}^{x-1}) \to \rho'''_n$
	$\vdots$	$\vdots$				$\ddots$	$\vdots$
	x=$\frac{n}{2}$	New objective					$(\rho_{2i}^{[i]})_{i=0}^{x-1} \to \rho_n$
		Result					$(\rho_{2i}^{[i]})_{i=0}^{x-1} \to \rho_n^{[x]}$

To apply imitation learning methods like DAgger to speed up fixed-point iterations like Hartree-Fock, we need to specify three things: the world model M, the policy class Π, and

the expert policy π^*. For our target problem of accelerating fixed-point updates, the world model is simple: our state is the history of iterates (e.g., density matrices) we have visited so far, and our action is to append another iterate to this history.

We have already mentioned our policy class: at each iteration i of our fixed-point calculation, we apply a function approximator with parameter vector $c^{(i)}$ to the history of past iterates, and then we pass the output of the function approximator into the original fixed-point update (e.g., the base Hartree-Fock update). In our experiments, the function approximator is a linear combination: $c^{(i)}$ is a vector of weights on previous density matrices.

Finally, there are two reasonable choices for an expert policy. In both cases we pick a step size Δ; recall that the goal is to replace Δ calls to the original update function with just a single call.

The simpler choice of expert policy is to work from one or more expert demonstrations—in our case, each demonstration is a sequence of density matrices $\rho_0, \rho_1, \rho_2, \ldots$. At step i we train our function approximator to predict $\rho_{i+\Delta}$ of the expert, no matter what the current iterate is. In addition to simplicity, this choice of expert has the advantage that we can gather expert demonstrations however we like: we don't have to be able to make calls to the expert at training time. We use this choice of expert policy in our experiments below.

The second choice of expert policy is to use Δ calls to our original update function f: e.g., if the current iterate is ρ and $\Delta = 2$, then the expert predicts that we should move to $f(f(\rho))$. This choice of expert has the possible advantage that it could adapt better to mistakes by the learner. However, in our application it is substantially more expensive, since it requires calling the update function repeatedly during training; so, we defer experiments with this expert policy to future work.

We introduce the notation $(\rho_i, \rho_j, ...) \rightarrow \rho_k$ to represent: (i) creating a linear combination of the density matrices $\rho_i, \rho_j, ...$ and (ii) carrying out one iteration of the Hartree-Fock algorithm to generate the next iterate ρ_k. We use a different set of coefficients for each step of the trajectory; write $\hat{c}_i$ for the coefficients at step i. The training data is a set of molecular examples, and we train the coefficients to minimize a regularized sum of squared errors over these examples, subject to some constraints (details in Section 4).

The training process is visualized in Table 1, in which Hartree-Fock iterations are shown as columns and Dagger iterations are shown as rows. We begin by training a policy for the first iteration of Hartree-Fock. In this case DAgger has only one iteration, in which a policy is trained on the objective $(\rho_0) \rightarrow \rho_2$. The learned policy uses the coefficients $\hat{c}_1^{[1]}$, where the superscript indicates DAgger iteration and the subscript indicates Hartree-Fock iteration. The density matrices generated from this learned policy are referred to as $\rho_2^{'}$, where the number of primes indicates the DAgger iteration.

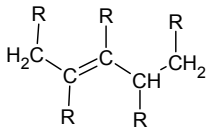
The training process then moves onto the second Hartree-Fock iteration, which has two DAgger iterations. In the first DAgger iteration, a policy is trained on $(\rho_0, \rho_2) \rightarrow \rho_4$. The learned policy uses coefficients $\hat{c}_2^{[1]}$ and generates induced states $\rho_4^{'}$. The second DAgger iteration uses an objective, symbolized $(\rho_0, \rho_2^{'}) \rightarrow \rho_4$, that includes states, $\rho_2^{'}$, induced from the learned policy of the previous Hartree-Fock iteration. For each molecular example, the objective selects between the expert density, ρ_2, and the induced one, $\rho_2^{'}$, with probability β_2. (β_j of Algorithm 2 is set to 0.5^{j-1}: sampling of expert states decreases with DAgger iteration.) The resulting learned policy uses $\hat{c}_2^{[2]}$ and generates induced states, $\rho_4^{''}$. At this

point, there are no additional induced states to include in the training and so the DAgger iteration terminates, leading to the upper triangular structure of Table 1.

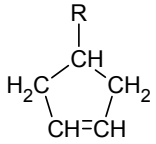
In the third Hartree-Fock iteration, the DAgger iterations train objectives $(\rho_0, \rho_2, \rho_4) \rightarrow \rho_6$, $(\rho_0, \rho_2, \rho_4^{'}) \rightarrow \rho_6$, and $(\rho_0, \rho_2^{'}, \rho_4^{''}) \rightarrow \rho_6$, respectively, such that each DAgger iteration samples induced states earlier in the trajectories.

4. EXPERIMENTAL DESIGN

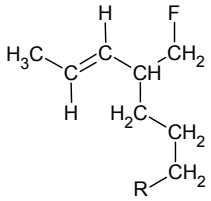
The computational experiments use the approach of Section 3 to train a policy for accelerating Hartree-Fock and compare the results to some baseline approaches. Of particular interest is the degree to which a policy trained on one class of molecules can transfer to a different class of molecules.



(a) Pent-3-ene



(b) Cyclo-Pentene



(c) Pent-3-ene-2-propyl-1-F

Figure 1: The chemical structure of molecules in the datasets, R = H, F, OH or NH_2

We generated the following three data sets, with the first being used for training. Each data set consists of a set of molecules, as defined by the bonding pattern between the atoms, and a set of distinct geometries of these molecules, as defined by distortions of the structure away from the equilibrium geometry. The geometric distortions are generated using a random uniform distribution of ± 0.5Å for bond lengths and $\pm 10^{o}$ for bond angles. This approach of uniform sampling generates highly distorted structures. To prevent inclusion of structures that are of little interest in chemical applications, structures are rejected if there are contacts closer than 3Å between non-bonded atoms. In addition, each molecular configuration is placed in 4 different electrical field environments (1 with no field + 3 different fields for X, Y and Z directions).

pent3ene 15 unique molecules corresponding to single substition of pent-3-ene: we place a single substituent at one of the positions indicated by R in Figure 1a, with all other R's being hydrogen (-H). The substituents are fluorine (-F), hydroxyl (-OH) and amine (-NH_2). The training set includes, for each molecule, the equilibrium geometry and three distorted geometries. Distortion includes a random free rotation about the leftmost carbon-carbon single bond of Figure 1a.

cycloPentene Includes cyclo-pentene and its three singly-substituted analogues (Figure 1b), each in the equilibrium geometry and 4 distorted geometries.

pentPropylF Includes the three singly-substituted species of pent-3-ene-2-propyl-1-fluorine (Figure 1c), each in its equilibrium geometry and 7 distorted geometries. Distortion

includes a random free rotation about the leftmost carbon-carbon single bond of Figure 1c.

For each molecular instance, we use a heuristic approach to generate a high-quality expert demonstration. We start from a steady-state density matrix, ρ_n, obtained from a standard fixed-point algorithm (Pulay, 1980). We first train a policy that takes the initial density matrix to the final density matrix directly, $(\rho_0) \rightarrow \rho_n$, and use this policy to generate ρ_1. We next train a policy on $(\rho_0, \rho_1) \rightarrow \rho_n$ and use it to generate ρ_2, and so on. Every instance in the training data converges in fewer than 12 iterations under this heuristic, which is considerably faster than DIIS (Pulay, 1980), and so provides a high-quality expert demonstration. To further expand the training data, expert demonstrations are constructed starting from two starting points, $\rho_0 = 0$ and $\rho_0 = \mathbf{I}$. In our experiments we compare the learned DAgger policy both to this "naive" policy and to the "DIIS" policy.

In training a policy to the objective, $(\rho_a, \rho_b, ...) \rightarrow \rho_c$, the error is defined as the sum of the distance of the density matrix, $\|\rho - \rho_c\|$, and the molecular energy, $|E(\rho) - E(\rho_c)|$ in Hartrees, from the target. We optimize the policy coefficients using the trust-region reflective algorithm (Coleman and Li, 1996) as implemented within Matlab (Mat, 2012). Since we are assuming only black-box access to the expert (and since the Hartree-Fock update step contains a number of operations, such as eigenvalue calculations, that are difficult to differentiate analytically), we computed gradients for the optimizer using finite differences.

To help ensure the stability of our learned policy, we constrain the coefficients c_i to lie in $[0, 1]$. As in DIIS methods, it is likely desirable to have the coefficients sum to one (Kudin et al., 2002), which we promote by adding regularizers to the objective, one for each step i of the Hartree-Fock iteration:

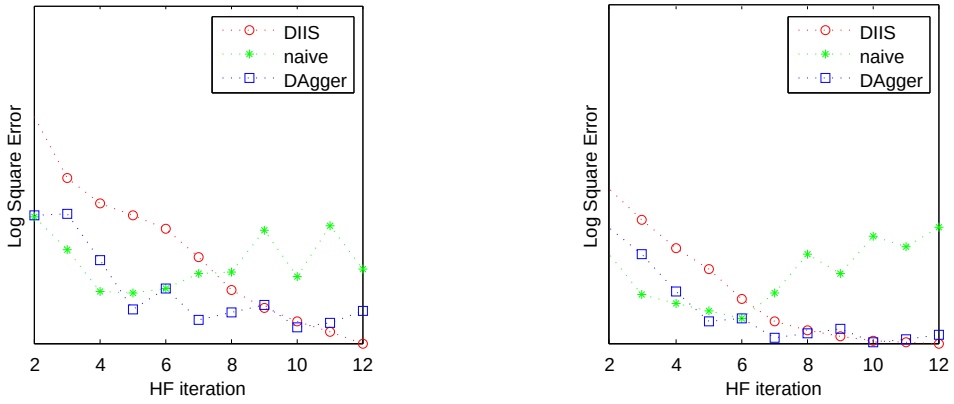
$$R^{(i)} = w(\mathbf{1}^\top \hat{c}^{(i)} - 1)^2 \tag{1}$$

where the weight w was empirically adjusted to a value of 30.

5. RESULTS

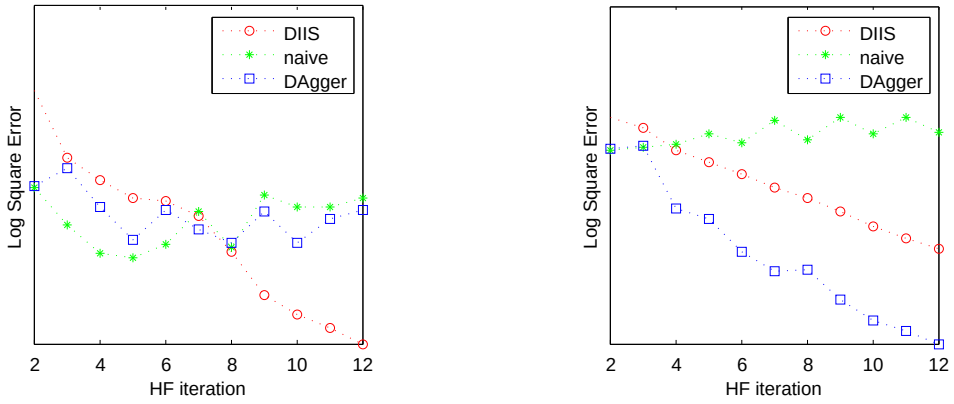
Both the "naive" and "DAgger" policies converge in the first few iterations on the *pent3ene* training dataset (data not shown). The behavior of the policies on the test data sets is shown in Figures 2 and 3. The DAgger algorithm was trained only up to the sixth iteration, beyond which, the conservative policy of taking a linear combination of the previous two density matrices with equal weight was employed. This is consistent with previous work on Hartree-Fock acceleration, which switch to a more conservative approach in the final stages of convergence (Hu and Yang, 2010).

The error from "DAgger" decreases more rapidly than the "DIIS" policy for the initial 6 iterations in all cases. This is not the case for the "naive" policy, which fails to converge on the substituted Pent-3-ene-2-propyl-1-F dataset with $\rho_0 = I$ as a starting point (Figure 3). These results demonstrate that imitation learning can accelerate fixed-point iteration in a manner that generalizes to situations not included in the training data and that the use of "DAgger" can substantially improve transfer.



(a) $\rho_0 = 0$ (b) $\rho_0 = I$

Figure 2: Log Error as a function of iteration tested on *cycloPentene* dataset



(a) $\rho_0 = 0$ (b) $\rho_0 = I$

Figure 3: Log Error as a function of iteration tested on *pentPropylF* dataset

6. CONCLUSION

The work explores a new application of imitation learning: accelerating the fixed-point iterations that are common in science and engineering. The mapping to imitation learning is quite general in that it requires only black-box access to the update function from an existing fixed-point algorithm. The intuition is simple: we accelerate convergence by attempting to skip $\Delta - 1$ out of every Δ calls to the original update function, replacing the skipped calls by a learned mapping. The specific case considered here, the Hartree-Fock algorithm from quantum chemistry, allowed us to test the ability of policies trained on one situation (chemical system) to transfer to different situations (chemical systems). Our results indicate successful transfer: imitation learning leads to policies that transfer better between systems than competing approaches.

References

Matlab version 7.14.0.739. Natick, Massachusetts, The MathWorks Inc., 2012.

T F Coleman and Y Li. An Interior Trust Region Approach for Nonlinear Minimization Subject to Bounds. *SIAM J. Optim.*, 6(2):418–445, 1996. ISSN 10526234. doi: 10.1137/0806023. URL http://link.aip.org/link/SJOPE8/v6/i2/p418/s1&Agg=doi.

D.B. Cook. *Handbook of Computational Quantum Chemistry*. Dover Books on Chemistry. Dover Publications, 2005. ISBN 9780486443072. URL http://books.google.com/books?id=Aa0qAwAAQBAJ.

Alejandro J. Garza and Gustavo E. Scuseria. Comparison of self-consistent field convergence acceleration techniques. *J. Chem. Phys.*, 137:054110, 2012a.

Alejandro J. Garza and Gustavo E. Scuseria. Comparison of self-consistent field convergence acceleration techniques. *J. Chem. Phys.*, 137(5):054110, 2012b. doi: http://dx.doi.org/10.1063/1.4740249. URL http://scitation.aip.org/content/aip/journal/jcp/137/5/10.1063/1.4740249.

Xiangqian Hu and Weitao Yang. Accelerating self-consistent field convergence with the augmented roothaanhall energy function. *J. Chem. Phys.*, 132(5):054109, 2010. doi: http://dx.doi.org/10.1063/1.3304922. URL http://scitation.aip.org/content/aip/journal/jcp/132/5/10.1063/1.3304922.

Konstantin N. Kudin, Gustavo E. Scuseria, and Eric Cancs. A black-box self-consistent field convergence algorithm: One step closer. *J. Chem. Phys.*, 116(19), 2002.

Konstantin N. Kudin, Gustavo E. Scuseria, and Eric Cancés. A black-box self-consistent field convergence algorithm: One step closer. *J. Chem. Phys.*, 116:8255, 2010.

Péter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73(2):393–398, July 1980. ISSN 00092614. URL http://www.sciencedirect.com/science/article/pii/0009261480803964.

Thorsten Rohwedder and Reinhold Schneider. An analysis for the diis acceleration method used in quantum chemistry calculations. *J. Math. Chem.*, 49(9):1889, 2011.

S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell. Learning message-passing inference machines for structured prediction. 2011a.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv*, 1011.0686, 2011b.

Appendix A. The Hartree-Fock algorithm

The electron distribution is in principle given by the time-independent Schrödinger equation,

$$\hat{H}\Psi = E\Psi \tag{2}$$

Here Ψ is the many-electron wave function, so that $\Psi^2(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$ is the probability of finding the molecule's N electrons at positions $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$. $\hat{H}$ is the Hamiltonian, a linear operator on the space of wave functions; $\hat{H}$ encodes the physical laws that govern the electrons. Eq. 2 is an eigenvalue problem: there are many solutions Ψ, each corresponding to a different energy level E. We are interested in particular in the ground state, the solution of lowest energy.

Unfortunately, it is computationally prohibitive to solve the Schrödinger equation directly for all but the smallest molecules. So, Hartree-Fock theory uses a mean-field approximation to replace the Schrödinger equation with the much simpler problem of finding the single-electron density $\rho(\mathbf{r})$, then solves for $\rho(\mathbf{r})$ by fixed-point iteration.

In more detail, given a guess $\rho(\mathbf{r})$ at the electron density, Hartree-Fock solves for *molecular orbitals* $\phi_a(\mathbf{r})$, each with associated energy ϵ_a, by analyzing the behavior of a single electron under the mean field generated by $\rho(\mathbf{r})$. It then forms an updated electron density ρ' by assuming that each of the molecule's N electrons occupies one of these orbitals: no more than two electrons per orbital (one each with spin up and spin down, consistent with the Pauli exclusion principle), and occupying orbitals starting from the lowest energy ones (since we are interested in the ground state).

The molecular orbitals are solutions of a single-electron eigenvalue problem:

$$\hat{F}\phi_a(\mathbf{r}) = \epsilon_a\phi_a(\mathbf{r}) \tag{3}$$

Here $\hat{F}$ is the Fock operator,

$$\hat{F}(\rho) = \hat{h}_1 + \hat{G}(\rho(\mathbf{r})) \tag{4}$$

where $\hat{h}_1$ is the one-electron Hamiltonian, containing operators that account for the kinetic energy of the electrons and the interaction of the electrons with the nuclei, and $\hat{G}$ is an operator that captures the interaction between a single electron and the mean field based on the input density $\rho(\mathbf{r})$.

For computational purposes, the molecular orbitals, ϕ_a of Eq. 3 are written as a linear combination of basis functions,

$$\phi_a(\mathbf{r}) = \sum_{i=1}^{N_{basis}} \chi_i(\mathbf{r})C_{i,a} \tag{5}$$

The operators $\hat{F}$, $\hat{h}_1$, and $\hat{G}$, along with the density $\rho(\mathbf{r})$, become symmetric matrices of dimension N_{basis}, and Eq. 3 becomes a generalised matrix eigenvalue problem:

$$F(\rho)c_a = Sc_a\epsilon_a \tag{6}$$

where F is the Fock matrix (with $F_{ij} = \langle\chi_i, \hat{F}\chi_j\rangle$), S is the matrix of inner products of the basis functions ($S_{ij} = \langle\chi_i, \chi_j\rangle$) and c_a is the a^{th} column of $C_{i,a}$. Assigning an electron to the orbital ϕ_a now corresponds to adding $c_a c_a^\top$ to the density matrix ρ.

The Fock matrix is constructed from the density matrix as

$$F_{i,j} = (i|\hat{h}_1|j) + \sum_{k,l=1}^{N_{basis}} \rho_{k,l} \left[2(ij|kl) - (il|kj)\right] \tag{7}$$

where $(i|\hat{h}_1|j)$ are matrix elements of the one-electron operator between basis functions i and j, and $(ij|kl)$ are two-electron matrix elements for the Coulomb interaction between electrons. The summation over two-electron matrix elements causes construction of the Fock matrix to scale naively as $O(N_{\text{basis}}^4)$, although (as described above) scalings closer to $O(N_{\text{basis}}^3)$ are routinely achieved by taking advantage of sparsity in the two-electron integrals (Cook, 2005).