Surrogate and Iterative Machine Learning Methods for Modelling Chemical Phenomena

By

Karl Bernard Schaettle

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Chemical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Fall 2022

Abstract

Surrogate and Iterative Machine Learning Methods for Modelling Chemical Phenomena

by

Karl Bernard Schaettle

Doctor of Philosophy in Chemical Engineering

University of California, Berkeley

Professor Kranthi K. Mandadapu, Chair

Modelling soil properties has important implications both for soil remediation and preventing misapplication of fertilizer in large-scale farming settings. By better understanding the dynamics of radioactive cesium infiltration and binding in clays, remediation strategies can be designed to lessen the long-term impact of radioactive particles on the environment and society. Similarly, adverse environmental effects associated with fertilizer runoff such as toxic algal blooms can be mitigated by precisely modelling soil nutrient concentrations and quantitatively predicting the economic effect of fertilizer application. Roadblocks to modelling microscale ion diffusion in bulk clay include the relatively long timescale of cesium-potassium ion exchange as well as the excessive computational cost associated with modelling all-atom systems; in particular, explicitly modelling hydrogen atoms drastically reduces the minimum simulation timestep. Modelling crop yield as a function of the spatial distribution of soil nutrients is complicated by an inability to take a dense set of soil samples in large-scale farms. There is also a relative lack of traditional agronomic literature quantitatively describing crop yield as a function of the high-dimensional soil nutrient feature space.

Machine learning and surrogate modelling methods are becoming increasingly common in engineering and science. While "black box" methods such as random forest regression and neural network modelling have been very successful at fitting physical phenomena, there is an increasing need to qualitatively and quantitively improve model interpretability and computational efficiency. In addition, machine learning models can be quite computationally expensive to use in optimization and may not have a well-defined methodology for doing so. Methods for improving computational efficiency of a model include coarse-graining (in the case of all-atom simulations) or approximating a "black box" model with another model designed to have tractable optimization properties. In order to retain fidelity to the initial model while increasing interpretability, in both cases the dimensionality of the model is reduced either by introducing multi-atom coarse-grain centers or approximating the target function as a linear combination of low-dimensional components. To ensure that the coarse-grain or reduced order surrogate models accurately capture properties of the original model, information from the model being approximated is used in their construction. In the case of reduced order surrogate modelling of random forest regression, low-dimensional components are chosen on the basis of ranked feature interaction importance. Using iterative Boltzmann inversion (IBI) to coarse-grain an all-atom simulation, the radial distribution

functions of only a subset of atoms are used to reproduce structural and thermodynamic properties of the original system.

The goal of the study performed in Chapter 2 was twofold: to use a data-driven methodology to model soybean yield (Glycine max L. Merr.) as a function of soil nutrients in well-irrigated soil and to develop a reduced order surrogate model capable of gradient ascent optimization. Several datasets were used to approximate soil nutrient concentrations using a random forest model: discrete soil samples, dense multispectral images of the plants near midseason from an unmanned arial vehicle (UAV), and a dense map of soil electrical conductivity. An iterative random forest (iRF) model was then fitted to a dense set of soil features, and important feature interactions of dimension 2 to 4 were extracted. Each feature interaction was used to generate a Highly Adaptive Lasso (HAL) pseudo-response surface corresponding to a low-dimensional projection of the feature space. We used the HAL surfaces to develop a reduced order surrogate model (ROSM) of the random forest; this ROSM is a linear combination of HAL surfaces derived from the feature interactions identified by the random forest. The resulting ROSM essentially has low local dimension because each component has maximum dimension 4. In practice, order 5 and 6 interactions were identified, but retaining them greatly decreased the computational efficiency of the HAL modelling and did not improve the model fidelity. Because the ROSM is a linear combination of low-dimensional surfaces, its gradient can also be described as a linear combination of the gradients of each surface. The ROSM can therefore be used in gradient ascent optimization at the same computational cost of evaluating the ROSM itself and is well-defined over the entire feature space. Maps of fertilizer application are derived for optimizing the soil concentrations of phosphorus and potassium.

Chapter 3 is a study using iterative Boltzmann inversion to generate a coarse-grain model of an all-atom simulation of ion interstratification in illite clay. Experimental results indicate that cesium ions can exchange with potassium ions in bulk layered silicates, indicating that there is a mechanical or thermodynamic compensation for the incorporation of the larger cesium ion. Iterative Boltzmann inversion was used to incrementally update coarse-grain simulations of four clay layers by adjusting bonded and non-bonded interaction strength between coarse-grain centers, representing oxygen atoms in the clay layers and the ions themselves. The model was able to reproduce results from smaller all-atom simulations indicating that the barrier to ion exchange is a function of interlayer spacing, which in turn depends on the identity of the ion in the interlayer. By randomizing the position of ions in the interlayer between each coarse-grain simulation, the coarse-grain model was better able to sample the phase space and subsequently was not subjugated to overfitting based on the configuration of the ions. Most importantly, the coarse-grain model is able to run approximately 70 times faster than an all-atom simulation due to a roughly 2:1 reduction in the number of modelled particles. By eliminating explicit hydrogen atoms in the coarse-grain model, the time step could be increased by a factor of roughly 10.

Table of Contents

List of Figures

List of Tables

Dedication

This dissertation is dedicated to my wife Jessica, my advisor Dr. J. Ben Brown, and all my friends, family, and colleagues that have made it possible.

1 Introduction

1.1 Surrogate Modeling to Learn Ecosystem Control Points in Agriculture

In addition to the requirement of increased crop production to sustain population growth, global farming is subject to a variety of unpredictable socioeconomic and environmental pressures.[1] Weather uncertainty, crop diseases and pests, and market volatility are particularly important sources of risk. Managing risk through precision agriculture is especially important given the continuing increase in the number of hectares per farmer on modern large-scale farming operations.[2] By managing the risk associated with production in particular through the use of sustainable and high-precision farming techniques, growers can aim to boost yields and therefore counter market uncertainty.[3]

Both soil organic matter accumulation and soil microbe diversity are essential measures of soil health.[4] Sustainable farming practices are needed to maintain soil health, prevent depletion of mineral fertilizer sources, and prevent environmental consequences associated with overuse of fertilizers. These sustainable techniques include limited soil tillage to prevent soil erosion and nutrient runoff, using cover crops to naturally replenish minerals depleted by single species farming, and the use of organic amendments in lieu of mineral fertilizers.[5] However, the increased cost associated with using some of these sustainable techniques must be quantitatively weighed against their long-term advantages to facilitate widespread adoption.

Remote sensing technology includes not only direct measurements of properties such as soil electrical conductivity and pH level, but also the use of vegetation indices (VIs) computed from unmanned aerial vehicle (UAV) or satellite images of crops.[6] These computed indices such as the normalized difference vegetation index (NDVI), the soil adjusted vegetation index (SAVI), and various leaf area measurements[7] are typically used as an indicator of overall crop health, and together with other precision measurements can provide a wealth of data for determining the state of the crop.

1.1.1 Spatial Statistics and Interpolation: Kriging

Kriging[8,9] is a common method of creating a dense map of a feature from discrete or sparse measurements in agricultural and geospatial applications. For example, soil pH and micronutrient[10,11] concentrations are much more difficult to assess than quantities that can be measured continuously or on a dense grid such as electrical conductivity or crop yield. In ordinary kriging, a continuous function is generated from the discrete measurement data points in two or three dimensions by assigning weights to each of the known measurements. The weights are found by minimizing the variance of the error residual, which results in a system of linear equations with one additional Lagrange condition.[12] Generalizations of ordinary kriging include a variety of methods that do not assume the mean of the continuous function being generated stays constant in space.

In its original formulation, kriging does not account for the behavior of other variables that are correlated to the set of sparse measurements, for example the behavior of soil electrical conductivity as it relates to concentrations of individual micronutrient ions. Kriging has been extended to include information from other explanatory variables (usually continuous) in a formulation known as regression kriging.[13] In regression kriging, the ordinary kriging model of the sparse measurement of interest is paired with an arbitrarily chosen regression model of the additional explanatory variables. In many applications, multiple linear regression is a popular choice, but the formulation of regression kriging places no explicit functional limit on the paired regression model. A regression model of the sparse measurement of interest is fit using additional explanatory variables, and a kriging model with mean zero is subsequently applied to the residual values of the regression model. The sum of the regression model and the kriging model is called a regression kriging model. Regression kriging models tend to perform much better than ordinary kriging models when the covariates are strongly correlated to the sparse measurement of interest but can suffer from overfitting if they are weakly correlated or an inappropriate regression model is chosen.[13]

In precision agriculture, ordinary and regression kriging models are used to generate maps from sparse soil measurements of relevant quantities. Soil organic carbon content, total nitrogen, and pH maps were generated over a large area using multiple linear regression kriging and compared to the same maps generated through ordinary kriging in Pham et al.[14] They used three computed vegetation and topographic indices as regressor variables to compute a model of their target quantities. Their results indicated that the regression kriging model had significantly improved accuracy as compared to ordinary kriging alone, providing evidence that using a simple regression or regression kriging model for sparse measurements is appropriate if the covariates are mechanistically or otherwise strongly correlated.

1.1.2 Review of Machine Learning in Precision Ag

Machine learning and machine vision are currently used in a variety of settings in precision agriculture. The majority of industrial applications fall into the category of machine vision, including implementations for visual detection and discrimination of plants, sorting of fruits and other foodstuffs, and identification of diseased crops.[15,16] Automated detection of weeds is useful for precision targeting of herbicide, reducing the need for excessive, uniform application. In one study, the authors developed an algorithm to detect the difficult to identify Bidens pilosa L., a common agricultural weed similar in color to wheat. By assessing images using color segmentation and shape analysis, they were able to detect Bidens at a rate of approximately 80%.[17]

Machine vision has important applications in monitoring crop health. Hyperspectral imaging has been used to estimate density of vegetation from satellite imaging and UAV imaging. Doing so allows for a much larger number of observables and precise, localized estimation (sub field-scale) of crop health and yield.[18] Different crop coverage indices can be computed from reflectance in several wavelengths, especially in near infrared and red wavelengths due to high crop reflectivity. However, satellite images (as compared to UAV images) are especially prone to weather-related obfuscation due to their height above the atmosphere.[19] Incorporating information from the entire electromagnetic spectrum between near infrared and ultraviolet wavelengths has been associated with increases in yield predictive accuracy on the order of 20%.[20,21]

Hyperspectral imaging of crops with UAVs is becoming much more commonplace, despite the significantly higher cost compared to satellite imaging.[18] While satellite images have a resolution on the order of meters, high quality UAV images can distinguish features as small as a few centimeters. This higher resolution is essential for precisely estimating the development of individual plants during different stages of the growing season. In particular, UAV images can be used in the time series estimation of the leaf area index (LAI), which is critical in determining meter-scale fertilizer requirements.[22] Despite the successes of this technology, UAV imaging remains cost prohibitive compared to other analytical techniques, so correlating satellite imagery to UAV snapshots and/or improving prediction using other measurables is desirable.

Other important post-harvest applications of machine learning and machine vision include product quality assessment and sorting. Using a variety of machine learning techniques, grains can be efficiently graded according to industry specifications.[23] This extremely cost-effective evaluation is necessary to efficiently price products and deliver properly graded grains for different consumer applications, resulting in less industrial waste overall. Classification systems can also be coupled with automatic sorters, with applications especially in fruit to greatly increase efficiency and reduce the need for manual sorting.[24] Use cases also include the efficient identification of infested grains and/or otherwise defective products.[25] Different machine learning techniques such as artificial neural networks (ANN), support vector machines (SVM), and others have been used to classify varietals with accuracy exceeding 90%.[23]

Machine vision has applications in automated tractor guidance for precision harvesting, planting, and fertilizer application. In one study, a machine vision system was used to keep a tractor on the correct path to within 1 cm down a crop row by continuously monitoring position of plants in adjacent rows and adjusting steering accordingly.[26] Machine learning guidance systems allow for greater precision than GPS-based guidance at relatively little additional cost to the consumer.

Despite great advancements in machine vision, machine learning using high precision data is currently underapplied to yield prediction and variable rate application of fertilizer outside of small-scale studies. While neural network, gaussian process, random forest, and other machine learning methods have been applied to yield prediction,[27] most studies have focused on existing, readily-available low-precision measurements such as satellite multispectral imaging and GPS-based topographical analysis. Many of these studies found that certain computed vegetation indices (VIs) were significantly more important than others in predicting yield, indicating the need for a machine learning framework capable of discriminating between variable importances. It is predicted that growing adoption of higher precision measurements such as UAV image acquisition and meter-scale soil electrical conductivity measurements will be heavily incorporated into large-scale planting decisions and fertilizer management in the near future.[27]

1.1.3 Crop Nutrient Interactions

Several groups have used a variety of machine learning and statistical methods to capture higher order responses to macro- and micronutrients in staple crops such as soybeans and rice. One common technique is to create a polynomial yield response model of variables of interest in lieu of attempting to derive interactions from agronomic first principles, which naturally provides a framework for investigating interactions in a reduced view of the entire nutrient feature space.[28–30] These studies require optimized factorial design of differential fertilizer application or soil nutrient concentrations to create a model that spans the feature space in lieu of generating a dense map of nutrients across the field and partitioning after the fact. To generate a polynomial model of rice yield in response to nitrogen and phosphorus fertilizer application, Shen et al.[30] carried out 41 individual field trials using factorial design. They were able to use their model to estimate the optimal economic fertilizer rates for nitrogen and phosphorus application as a function of the soil cation exchange capacity. Their estimate implicitly accounted for the higher order interaction of the effect of nitrogen and phosphorus fertilizers by using a fully second-order polynomial model.

In two separate studies, a polynomial model was used to generate a yield response and herbivory model in soybeans as an implicit function of higher order between nutrients.[28,29] The authors generated Fe-B-Zn and S-N-P macronutrient interaction response surfaces by designing small-scale laboratory growing experiments using explicitly chosen nutrient ion solutions and using factorial design to optimally cover the designed feature space. In each of the studies, the authors indicated that the polynomial response surfaces (which are inherently smoothed due to the low order of the polynomial) were insufficient to capture the behavior of all of the observed datapoints, with some experimental observations lying far outside the generated surfaces of best fit. The results from both studies indicate that quantifying the effects of micro- and macro-nutrients individually on herbivory and soybean yield without investigating simultaneous, higher-order nutrient feature interactions may be a reason for contradictory predictions of plant response in the agronomic literature.

In another study, the feature response interactions between phosphorus, potassium, and sulfur were investigated in soybean plants.[31] The response interaction between phosphorus and potassium in particular has been understudied, with conventional agronomic wisdom indicating that the two nutrients can be applied and optimized separately without significant interaction in healthy soil regimes. Plant yield was modelled in greenhouse conditions in 27 separate experiments using 3x3x3 factorial design to span the feature space of healthy soil conditions for these three nutrients. The authors found that, at high concentrations of sulfur in the soil, the yield response peaked at high values of potassium and intermediate values of phosphorus, similar to a result that we report. This result indicates that under these sulfur concentrations there is a penalty associated with banking phosphorus (applying phosphorus in excess of the minimum necessary concentrations for optimal yield), a result which is not typically reported for healthy soil. Averaged over all sulfur concentration values considered in this study, the authors predict that the soybean yield response peaks at the maximum studied phosphorus value over the range of potassium values, not an intermediate concentration of phosphorus. That is, the two-dimensional response surface associated only with phosphorus and potassium generated from this study does not indicate the same banking penalty when considering soil with high concentrations of sulfur. This suggests that important macronutrient interactions are dependent on other explanatory variables and that feature interactions of order greater than two may provide additional information that would otherwise be overlooked.

It should be noted that although polynomial models are capable of providing some explanatory power in investigating higher order interactions between nutrient response, the higher order variables themselves (products and powers of nutrient concentrations and fertilizer application rates) are not physically meaningful quantities. Many data-driven, machine learning-based approaches do not require explicitly controlling soil ion concentrations, but instead creating a dense map of nutrients allows for partitioning the feature space into hundreds or thousands of individual "experiments" per field to generate a continuous response surface of physically meaningful variables. In this way, machine learning methods including ours are capable of identifying hard to detect or context-dependent (i.e., soil pH or texture) nutrient interactions.

Moreover, although factorial design experiments do attempt to cover the feature space in an optimal way, they can be fundamentally more difficult to perform due to issues with controlling soil ion concentrations, and they suffer from requiring a fundamentally smaller number of chosen features than post-experimental partitioning of the measured feature space. However, it should be noted that for investigating interactions one at a time for a small number of features, factorial design of laboratory-controlled experiments can be very useful. In addition, due to the parametric nature of their design, linear and polynomial regression models are capable of producing predictions on regions of the feature space outside of their training domain, unlike random forests,[32] although this property of parametric learning algorithms can produce spurious results if extrapolated far outside the training domain.

Some groups have used first principles from agronomy and plant biology as a starting point for investigating higher order interactions between micronutrients and macronutrients. In some cases, trace minerals such as molybdenum are mechanistically involved with enzymes required for oxidation reactions of different macronutrients.[33] Despite being generally understudied in the literature, and despite being important in high-order models of crop health and yield, macronutrient-macronutrient interactions oftentimes have mechanistic underpinnings. For example, in one hydroponic experiment modelling the effect of cadmium poisoning on soybeans,[34] it was observed that increasing the concentration of potassium in the soil led to an increase in overall plant growth, photosynthesis, and the uptake of other macronutrients, likely due to the effect of potassium-assisted enzyme activation during nodulation. Nitrate acts as a "signal molecule" to modulate plant uptake response of phosphate, and it has been shown mechanistically that phosphate starvation in plants is attenuated by nitrogen starvation.[35] Similarly, phosphorus starvation mechanistically modulates sulfur transport and uptake.

However, most studies that investigate the underlying biological mechanisms behind macro- and micro-nutrient interactions are not able to rigorously quantify the effects of nutrient starvation or overabundance, and to an even greater degree they are not able to predict fine changes to the yield response due to small changes in macronutrient concentration near optimal conditions. Most traditional agronomic studies consider the yield and other response variables such as plant overall height or leaf weight to be roughly constant over a range of so-called ideal conditions. Without a more precise formulation of plant response in healthy soils, it is difficult to precisely quantify optimal fertilizer application rates or the most economically effective way of modulating the soil for a given real-world field. In contrast, our study and other machine learning methods attempt to find insights non-mechanistically. By creating detailed maps of soil conditions across many fields, it is possible to find interactions that would otherwise have been overlooked, thus creating an opportunity for investigating the mechanistic underpinnings of discovered higher-order interactions from first biological principles.

1.1.4 Machine Learning Models of Yield

In recent years, many machine learning methods have been applied to crop yield prediction. Methods used to model crops include neural networks and deep learning, linear and polynomial regression, gaussian process methods, random forest, and support vector machines.[36] Deep learning techniques are by far the most popular current method of modelling crop yield. These models provide very high fidelity on large datasets and are capable of learning highly complex nonlinear functional forms with little oversight. Authors of various studies include a variety of soil, weather, and other features in their models to create a temporal model of yield throughout the growing season or at the time of harvest. Some studies have used regression models to generate dense soil maps (10 meter grids) from sparse soil measurements by correlating the sparse soil samples with dense observed measurables, thus allowing the machine learning model to be trained on a feature space without missing values.[37] Computing soil maps in this way can be more accurate than ordinary kriging because the feature maps of densely collected variables can contain far more information than just the position information derived from sparse soil samples. Of course, ensuring that these generated dense maps from sparse features are meaningful requires only using regression variables that would be physically expected to correlate to soil features such as soil texture, conductivity, and multi- or hyper-spectral data indicating overall plant health.

Unfortunately, many of these types of machine learning models can be considered "black boxes" in that interpreting the functional form between inputs and outputs is very difficult. The results of the learning algorithm may also depend heavily on the hyperparameters used to tune in the model instead of robust trends in the underlying dataset. Another drawback associated with gaussian process methods is that computing the inverse of the covariance matrix is difficult or infeasible for high dimensional feature spaces.[38] While random forests provide insight by allowing the user to generate a ranked list of feature importances, it can still be difficult to interpret the response of several features simultaneously or to accurately decide the meaning of the model response on a reduced version of the input space. Moreover, many of these models are unwieldy or inappropriate to use for optimization due to their complex or fundamentally nonparametric functional form. Therefore, one important goal of machine learning methods for crop modelling, and indeed modelling of any regression problem with a high dimensional feature space, should be to increase interpretability on an effectively reduced basis of observed variables.

Several machine learning methods have been employed to reduce the effective dimension of the feature space or increase the local interpretability of a machine learning model. Building a model directly as a function of feature interactions to probe low-dimensional behavior is one such technique. In one study, Ansarifar et al.[39] develop an interaction regression model to identify globally explanatory features for corn and soybean yield in several states. Their study compares the use of several kernel functions to define higher-order feature interactions which are automatically selected by minimizing the residual values for a model composed of a linear combination of these interactions. Although their model is able to outperform many traditional machine learning algorithms in modelling yield, the interaction regression model does not explicitly attempt to control the dimensionality of the interactions. The interaction regression model also requires first eliminating features from different partitions of the test set and subsequently identifying overlaps in the set of features in each partition. In doing so this method is likely to eliminate variables with high local importance but reduced global importance that may be vital to accurately modelling outliers or regions with sharp response boundaries in the feature space, which can be as important as capturing the overall structure of the response variable.

1.1.5 Dimensionality Reduction and Surrogate Modeling

Reducing the effective dimension of the feature space is an important goal of interpretable machine learning. Reducing the dimension allows for increases in computational speed and interpretability of the regression model while hopefully maintaining accuracy. While this can be accomplished by fitting a model that naturally considers only low-order subsets of the entire feature space in each component, such as in polynomial regression or interaction modelling, it can also be done by using methods that rank the importance of global variables such as random forest. Of course, all of these machine learning methods require fitting a regression model to the entire feature space to effectively reduce the dimension or eliminate features, which can be computationally expensive for methods that scale poorly with the number of model parameters or feature space dimension. In particular, random forest classification and regression models avoid linear computational cost scaling with dimension by optimizing over a reduced number of features at each split.

Another common approach is to reduce the feature space dimension before applying a regression model. One such approach is to use principal component analysis (PCA) to eliminate combinations of features that do not strongly correlate to the variance of the target variable.[40,41] In precision agriculture, PCA can be used to effectively reduce the dimensionality of highly correlated data with a large number of features, as is often the case with certain soil texture variables.[42] PCA decomposes the feature space into uncorrelated linear combinations of the features and thereby naturally lumps together variables that are strongly correlated with one another. Processed feature spaces with reduced dimension can subsequently be used to generate more interpretable regression models of the target variable. Alternatively, because PCA is an unsupervised learning technique, it can be used to discriminate between clusters of data that are distant from one another in the feature space or that have different sets of explanatory components, thus making it an effective tool for generating more localized or context-dependent models. In one study, authors used principal component analysis and a genetic algorithm to find a feature subset that can best be used for crop classification.[43] They found that for different genetic classifiers they were able to dramatically reduce the dimensionality of the feature space, with over 50% reduction in two classification tasks using decision trees.

In general, random forest regression model performance is not typically affected by PCA transformation of the feature space (without dimensionality reduction) because it is capable of learning complex decision boundaries.[44] While PCA transformation can make these decision boundaries more easily interpretable for humans, forests that are sufficiently deep will not suffer degradation in precision from training on untransformed data. Nevertheless, random forest regression models have been studied using PCA-transformed feature spaces.[45] Indeed, applying PCA or other variable transformation methods before fitting a random forest can oftentimes simply result in a less interpretable model because the forest is fit to a set of transformed variables without physical meaning. Because random forests do not suffer from computational complexity concerns from training on high dimensional feature spaces, and because they offer a natural structure for reducing the dataset dimension after fitting, it is natural to use them to develop a framework for increasing model interpretability by globally eliminating features or using them as a basis for creating an interaction regression model.

Another method of increasing machine learning model interpretability is to construct a simpler, surrogate model that recapitulates the behavior of the more complex model. Local Interpretable Model-agnostic Explanations (LIME) is a general surrogate modelling strategy that approximates a more complex, global machine learning model with a local kernel.[46] The authors investigate several applications using a locally linear formulation with a gaussian kernel for interpolation. They are able to extract subimages used in identifying discrete objects by a neural network in an image, thus providing an interpretable explanation of how the neural network discriminates between classes. Although this method does not explicitly reduce the dimension of the feature space, it is able to capture behavior of black box models such as neural networks using simple parametrized models whose coefficients or projections are easily interpretable.

1.1.6 Review of Soil Ecology in Ag

The soil microbiome is extremely sensitive to and can be negatively impacted by overapplication of mineral and organic fertilizers. The purpose of fertilizer application is to maximize crop productivity, but consistent overapplication in the United States and globally has resulted in soil acidification, unnecessary runoff, and other changes to soil conditions that damage the long-term health of the microbial community.[47] Applying organic fertilizers at industry recommended rates has been associated with elevated soil nitrogen levels after the growing season, potentially resulting in environmental leaching and reduced crop yields on the order of 20%.[48,49]

The soil organisms that interact with agricultural crops can be roughly divided into microflora (bacteria, fungi) and microfauna (protozoa, mites, nematodes, etc.).[47] These organisms perform important duties such as breaking down plant matter, fixing organic nitrogen, and recycling minerals. Since the microflora are essential for degradation of plant and other organic matter, they therefore serve mainly as a repository for plant nutrients.

Soil acidification is generally associated with poor soil health, while large concentrations of soil organic matter are typically associated with improved agricultural yields.[50] These two easily measurable observables are therefore important proxies for the overall health of the soil. However, fertilizers, when over-applied over long periods of time, have been shown to both acidify soil and reduce retention of organic matter on average. Mineral fertilizers show conflicting, context-dependent effects on overall microfauna counts and carbon and nitrogen concentrations depending on soil conditions.[47,51,52] In addition, typical surface fertilizer application can lead to extreme stratification of phosphorus in the soil due to low incorporation rates, potentially leading to overapplication, increased runoff rates and poisoning the deep soil. This excessive phosphorus runoff can lead to severe downstream negative impacts, including algal blooms.[53]

In contrast, organic fertilizers such as manure have a more complex interaction with the soil microbiome due to high concentrations of a variety of organic compounds. While higher levels of organic matter are generally important for soil health,[47] overapplication of certain organic molecules such as humins can hinder plant growth.[54] Therefore it is important to have a model that incorporates mineral composition and organic matter content of soil to accurately prescribe fertilizer application in different soil regimes within a field.

A variety of pest control compounds are used in modern agriculture including fungicides, insecticides, herbicides, and pesticides.[47] Their impact on the soil microbiome and organic matter content is generally considered to be understudied, partially because large effects may not be observed for several years after their initial application. Fungicides are particularly impactful due to the important role of beneficial fungi in the soil, which can be negatively affected by broadly targeted fungicides.[55] Therefore it is important to grow crops in soil regimes hostile to pathological fungi while promoting plant growth.

1.2 Structural Coarse-Graining

A "bottom-up" CG model is defined as a model of a target system by using an explicit modelling or mapping technique.[56] The so-called fine grain model may be itself an atomistic approximation of a first principles molecular model of a system of interest. Different iterative coarse graining techniques can be shown to exactly reproduce the structural (in the case of iterative Boltzmann inversion) or thermodynamic (in the case of force-matching and relative entropy minimization) properties of the given detailed model. However, this convergence is only exact in the limit that the finite size effects of real simulations are negligible and that a tabulated many-body potential is used instead of pairwise or parametrized interactions.[57] Bottom-up CG models are ideal for studying relatively large systems or the temporal dynamics of such systems due to the inherent reduction in computational cost associated with mapping several atoms or molecules to a single coarse grain particle. Since this type of CG model can be developed independently of experimental observations of thermodynamic properties, they are ideally suited for studying kinetic phenomena under conditions that would be difficult to replicate in a laboratory setting.

1.2.1 Iterative Boltzmann Inversion

Iterative Boltzmann inversion (IBI) is a popular methodology for structural coarse graining and reducing the degrees of freedom in biomolecules and polymers, especially in fluid phases.[58] The IBI algorithm creates coarse grain models in an attempt to recreate structural features of the all-atom molecular dynamics model as opposed to explicitly capturing all of the equilibrium thermodynamic properties of the system. Whereas force matching is often employed to coarse grain one interaction potential at a time (often by restraining certain degrees of freedom), IBI has been shown to effectively reproduce radial distribution functions of several interactions simultaneously. Moreover, because bonded interactions in a solid system are modelled with harmonic oscillators between only two particles, the coarse grain potential converges extremely quickly when fitting the inverted potential with a parabola near the equilibrium bond length. In practice, we found that the CG bond length and angle pairwise potentials for intrasheet interactions are very well approximated in our solid system after using only the initial inverted all-atom radial distribution function, implying that these modes are nearly independent of their linked neighboring interactions. However, modifying the force constants of these bending and stretching modes simultaneously with tabulated pairwise potentials between distant atoms using IBI was found to be necessary to achieve high fidelity in the radial distribution functions both in our study and in others.[59]

Although IBI does not rigorously reproduce the average statistical properties of the all-atom model like the force matching methodology, CG models produced by IBI do more faithfully model the local energy landscape and structural properties of a material. IBI coarse graining has been shown to reproduce physical properties such as the density of polymer systems,[58] which strongly suggests that it is an appropriate technique for reproducing measurable structural properties such as average clay interlayer spacing and kinetic transition barriers in our system. IBI has also been used to generate CG models to study phase transitions in phospholipid bilayers.[60,61] This model in particular is capable of self-assembly into a lipid bilayer from a random configuration at biologically relevant temperatures. Although the model was developed in the amorphous phase using a coarse graining method which is not guaranteed to reproduce thermodynamic properties,[60] the model is capable of reproducing temperature-induced phase transformation to a semi-solid gel phase at similar temperatures and pressures to all-atom simulations (after accounting for the finite size effect).[61] Importantly, the phase transition temperatures correspond closely to experimental values of similar systems. These results indicate that the iterative Boltzmann inversion algorithm is capable of producing a model that is kinetically relevant outside of its initial window of convergence.

Extensions have been proposed to IBI to extend the applicability of the coarse grained model to a range of states or physical environments including solid or crystalline structures.[59,62] These studies relied on a damping factor in the iterative equation for updating the coarse grain potential using the IBI algorithm, and also noted that it therefore required more iterations to converge to the all atom radial distribution function as compared to a non-solid phase. However, in this study we found it sufficient to smooth the CG potential after every update step using a moving average window. In addition, ion positions in our model were randomly permuted and perturbed within the binding sites in initial configurations, further preventing the CG model from adopting unphysical crystalline configurations or simply oscillating between distributions. Both of these techniques allowed us to use the standard IBI algorithm without damping, drastically reducing the total number of simulations needed to achieve convergence. These modifications to the standard IBI coarse graining methodology allowed us to extend the success of the algorithm in the fluid and crystalline phases to a bulk rigid system without fluid properties and thermodynamically far away from a phase transition.

In solid or crystalline systems, force constants for bonds and angle distributions have been found to be significantly larger in the coarse grained model compared to the corresponding values in all atom simulations.[59] Therefore it is expected that the coarse grain representation in the solid phase derived from iterative Boltzmann inversion may have higher energy barriers to particle transition (for nonbonded particles) despite an overall reduction in the number of bending and stretching modes. To create a CG model of lipid molecules in the crystalline phase, bond and angle force constants between coarse grain centers were first found using IBI in the amorphous fluid phase and subsequently used to derive tabulated pairwise potentials between distant CG centers. It was found that the force constants derived from the fluid phase were insufficient to retain the crystalline structure of the solid phase, indicating that isolating individual interactions for coarse graining may not be successful in solid phase systems.[59]

Damped IBI has also been used to study other crystalline biomolecules such as cellulose fibrils.[63] In Srinivas et al., entire glucose monomers are mapped to a single coarse grain center to form sheets of bonded CG fibrils. Because of the homogenous nature of the sheets, they were able to establish very close agreement between the all atom and coarse grain radial distribution functions. This study represents an improvement on previous CG simulations of cellulose fibrils, which suffer from stability issues. By using an iterative approach to faithfully reproduce the structure of the crystalline sheets, stability problems and phase transitions can be prevented by capturing discontinuities in the RDF. These discontinuities tend to correspond to large barriers in the potential energy surface of a tabulated potential, preventing the dissociation of the structure on the timescale of simulation without explicit constraints or the addition of extra nonphysical interactions.

1.2.2 Force Matching

Force matching is another coarse graining technique which is widely applied to a variety of physical systems.[64] In the force matching technique, pairwise or density-based potential parameters are fitted to a database of forces derived from ab initio or all-atom simulations. Formally, force matching is a least squares minimization of an objective function which is a difference of ab initio forces and predicted forces from the potential function. See Izvekov et al. for more details.[65] Force matching has been extended to develop a method of coarse graining.[65,66] This method uses force matching to define a potential of mean force between CG centers based on simulation results from a corresponding all atom model. Coarse graining using the force matching methodology is also referred to in the literature as the multiscale coarse-graining (MS-CG) method.

For large simulations, MS-CG and iterative MS-CG methodologies can be considered "consistent" in that the equilibrium distribution of the coarse grain configuration space is equal to that of an all atom model.[67] The authors derive the sufficient condition that a coarse grain simulation that maps sets of atoms to CG centers is thermodynamically consistent if the force on the CG centers is equal to the configuration-averaged atomistic force. For this reason, force matching is frequently used in coarse graining applications where statistical and thermodynamical properties must be reproduced at the expense of precise structural detail and kinetic insights.

Force matching is typically used to model solvated individual biomolecules and other systems in a non-rigid phase, but has been used in some applications to study phase transitions between solid and fluid phases.[68] Multi-scale coarse-graining in solid metals in this fashion can essentially be thought of as a 1:1 mapping of an ab initio simulation to a molecular dynamics simulation. Because force matching reproduces the thermodynamic properties of a fine system in the limit of an unconstrained CG potential, it is well suited to studying systems where phase transitions are to be investigated. However, like all CG systems trained on all atom or other fine systems at a single reference state, there can be problems with transferability and generalization to other states. Unlike CG systems developed using iterative Boltzmann inversion, systems created using force matching are more likely to locally smooth the energy landscape and therefore underestimate the size of kinetic barriers, especially in cases where a large number of atomic centers are mapped to a single CG bead.

1.2.3 Relative Entropy Minimization

The relative entropy of two systems, also known in certain fields as the Kullback–Leibler divergence,[69] has been applied as a measure of how faithfully the coarse grained model reproduces the features of an atomistic model.[70] The relative entropy is an information-theoretic measure of the loss induced by mapping one model to another and is essentially a metric for the fidelity of a CG model at equilibrium. Like other bottom-up coarse graining approaches, this methodology requires the construction of an explicit map of a fine model to CG centers. By minimizing the relative entropy, the CG model reproduces the same distribution of microstates as the fine model it approximates despite incurring an unavoidable increase in relative entropy due to the mapping operation itself.[57] It should be noted that this property is not strictly realizable if specific parametrized functional forms are imposed on all the pairwise interaction potentials between CG centers. Instead, tabulated potentials are often used for long-range interactions. However, using parametrized potentials provides a natural way to minimize the relative entropy in coarse grain systems by gradient descent of the parametric variables. If the CG potential does not have a functional form imposed on it, then it can be shown that the techniques of force matching and relative entropy minimization result in the same potential energy landscape in the limit of an infinite system. When the potential is constrained to be pairwise additive or otherwise restricted to subsets of the CG centers, the two approaches are not formally guaranteed to return the same results or strictly reproduce the equilibrium thermodynamic properties of the all-atom system, but oftentimes they produce similar models when using pairwise tabulated potentials.[57]

Relative entropy minimization coarse graining has been employed in a wide variety of liquid and fluid systems, and to a lesser degree in bulk semi-rigid system modelling. In one study, Carmichael and Shell use relative entropy coarse graining to parameterize a model of polyalanine and subsequently simulate several dozen of these molecules to investigate peptide self-assembly under physiologically relevant conditions.[71] They parametrized the model on an all atom system with helical secondary structures and observed that under crowded conditions that polyalanine molecules quickly formed into β-sheet-like structures, indicating that their model successfully replicated the potential energy landscape of the hydrogen bonding network. This work has been extended to study the structure of arbitrary globular proteins, which like clays feature strong non-bonded interactions between neighboring sheet-like structures.[72] Using a four-site coarse grained model of leucine and valine, the authors were able to predict the structure of globular proteins containing on the order of 250 residues to within 2.5 angstroms. These results indicate that despite not being an explicitly structure-based approach, and despite developing a pairwise, parametrized CG model, relative entropy coarse graining is capable of reproducing precise structural details. The converse has been observed for structural coarse graining as well; even though iterative Boltzmann inversion on pairwise potentials in finite systems often drastically changes the energy landscape in an attempt to preserve kinetic details, structural CG models are often capable of producing meaningful thermodynamic insights outside of their initial training conditions.

1.3 Statement of Innovation

Machine learning methods and coarse-grained models are broadly applicable to data-driven disciplines where an underlying mechanistic model is either not known or is too complex to be extrapolated from the microscopic level. Optimization of fertilizer application is important in preventing excess runoff and can potentially result in less destructive mining practices. Modelling crops such as soybeans as a function of soil minerals and other environmental variables allows data-driven identification of interactions and soil regimes that are beneficial for crop production. To better understand soil uptake of radioactive cesium, it is necessary to model the displacement of potassium ions in bulk clay systems. All-atom models are generally too small or computationally expensive to properly model bulk behavior, so coarse-grained models that reproduce their statistical or mechanical properties are useful for simulating bulk systems. By better understanding the microscopic incorporation of cesium into bulk clay, the environmental impact of radioactive materials can be more precisely modelled. Moreover, remediation strategies can be more precisely engineered to account for the exact behavior of cesium in clay soils.

Strong second-order interaction between soil potassium and phosphorus was identified by the SMiRF procedure. The effect of increased organic matter is dependent on soil pH and does not have a uniform effect on crop production in all soil regimes. Although soil pH is very important independent of other soil variables, significant economic optimization of farming practices is possible by addressing only the application rate of phosphorus and potassium fertilizers. The coarse-grained model of clay reproduced results of interlayer spacing in systems with mixtures of cesium and potassium ions. The coarse-grained modelling results indicate that the size of the energetic barrier to ion diffusion for each type of ion is purely a function of the local interlayer spacing, and therefore the rate of exchange in systems with a mixture of cesium and potassium can be effectively determined with a detailed mechanical model of the ion binding site.

We have developed a general surrogate model for random forest machine learning methods that retains the global importance of feature space variables without sacrificing context-dependent interactions or effects. Our surrogate model is a linear combination of HAL surfaces, which are themselves minimum loss estimators of the effect of the interactions separately. By using a combination of machine learning methods to identify and model interactions, we are able to retain the effects of independent interactions and quantitatively rank their importance in the overall model. Taking a linear combination of interaction surfaces results in a robust, potentially lightweight model that is far more human-interpretable than a black-box random forest. The reduced order surrogate models are capable of fast optimization by fitting smoothed gradient surfaces whose cost is not greater than evaluating the model itself. The ROSM represents an improvement over random intersection trees for general modelling in that the model linear coefficients more precisely represent a ranking of feature interaction importance because the linear combination of HAL surfaces is fitted to the original output values of the random forest. We have shown that the ROSM response values are tightly correlated with the original random forest, indicating that the surrogate model retains the random forest quantitative behavior while recapitulating the model in the context of higher order interactions that are not directly accounted for by the random forest. Thus, the reduced order surrogate model is able to provide a more accurate quantitative evaluation of the response of feature interactions than by simply considering either the response surface from a random forest model or the feature interaction importance alone. Although the model was applied in this work to a somewhat low-dimensional example, the SMiRF procedure is well suited to high dimensional problems such as quantifying the higher-order response of interactions within a genome.

We developed a coarse-grained model that allows for bulk modelling of dozens of layers while retaining a detailed mechanical model of the ion binding sites. In addition, we identified a linear relationship between the interlayer spacing and the energetic penalty to ion diffusion without directly training the coarse-grained model on heterogeneous systems. In this work we used iterative Boltzmann inversion in a purely bulk solid system which has been previously underreported in the literature. IBI was used in the solid phase without the use of a damping factor by randomizing the position of ions and smoothing the tabulated potentials to prevent unphysical conformations. Typically damping factors are needed for convergence when using iterative Boltzmann inversion in semirigid phases due to local minima in the energy landscape in non-physical conformations, but by randomizing the position of ions and slightly perturbing them within the active sites between each iteration we effectively increased the sampling of the phase space. This method should be broadly applicable to other bulk rigid systems with high symmetry or repeated interaction sites. Although IBI is based on approximations of systems in the gas phase, we have shown that it is possible to effectively reduce the number of iterations by eliminating the damping factor, thus allowing us to run the algorithm on larger systems for the same computational cost. We have demonstrated that the mechanical properties of the active site are retained in mixed ion configurations after converging the coarse-grained model in purely homogenous systems, indicating that using IBI coarse graining is a robust method for generating general mechanical models for bulk systems in the solid phase.

2 A Surrogate Modeling Strategy to Learn Ecosystem Control Points in Agriculture

Schaettle K, Falco N, Ulrich C, Dafflon B, Brodie E, McEntire J, Wainwright H, Brown JB

ABSTRACT

Ecosystem control points are localized processes that contribute substantially to particular ecosystem functions – e.g., biomass productivity, carbon cycling, water quality. Learning control points directly from data is a central pursuit in molecular ecosystems biology. Agricultural lands constitute exceptional "reduced order" model ecosystems – consisting of only a single plant species and its microbiome – and hence useful testbeds for new data science tools. Here, we take advantage of an extensive dataset compiled at the AR1K.org field site in Humphrey, Arkansas in 2017 for monoculture irrigated soybeans (Glycine max L. Merr.). We developed a three-stage machine learning algorithm for the discovery of control points for target ecosystem services – and here we focus on agricultural yield. In the first stage, an iterative Random Forest (iRF) is used to extract important interactions and processes that are predictive of soybean yield. In the second stage, we use Highly Adaptive Lasso (HAL) to model interactions using functions that are differentiable almost everywhere. Finally, we fit a Reduced Order Surrogate Model (ROSM) by performing forward-backward regression under an L2 loss where each term in the model is itself a HAL response surface. The resulting hybrid learning machine achieves comparable performance to the iRF from which it is derived and captures explicit relationships suitable for human exploration. We call our technique Surrogate Models through iRF (SMiRF), and here we describe its utility in obtaining a predictive understanding of agricultural yield in terms of ecosystem control points at the AR1K.org field lab. In future work, we will pursue the use of SMiRFs to construct mechanistic process models from data, and we describe some of these directions.

2.1 Introduction

The importance of ecosystem services – yield, water quality maintenance, soil health – are the principal drivers of value and sustainability in agriculture. The study of monoculture crop systems provides a unique opportunity for foundational ecosystems science: to discover ecosystem control points[73] in a highly simplified setting – a single strain of a single plant species in relation to its microbiome replicated thousands of times across soil contexts in each hectare. Toward this end, in 2017, we established the Arkansas 400-hectare field laboratory (AR1K), a highly instrumented site near Stuttgart in the river delta region. Here, we use over 300 layers of data to discover the emergent parameters that constitute the primary drivers – the control points – of soybean yield at AR1K.

Further, by the year 2050, the world's population is projected to hit 9.8 billion people.[74,75] If current trends in global economic development and urbanization hold, agricultural production will have to increase by up to 70% to meet demand.[76] To minimize the amount of land used for agriculture, crop yields must continue to grow at a steady pace over the next several decades. Sustainable agricultural practices such as crop rotation and general soil management will be necessary to reach production thresholds.[77] Although chemical fertilizers have been successful at increasing the yield of soybean and corn (Zea mays), there is a growing necessity to reduce fertilizer application to prevent degradation of soil health[48,49] and to prevent the down-stream ecological impacts of nutrient loading in surface waters.[78,79] Indeed, the chronic overuse of fertilizers can result in soil acidification and depletion of micronutrients vital to plant growth over time.[80]

Prior to the advent of precision agriculture methods and GPS-based advancements in fertilizer application and tractor guidance, fertilizers were either applied uniformly on individual fields or on a very coarse grid. By using GPS guided machinery and prescribing semi-localized fertilizer treatments, it has been estimated that crop yields can be improved between 15-30% dependent on local soil heterogeneity.[81] Soil tillage has been widely practiced in industrial and large-scale farming for decades to increase water penetration and nutrient uptake.[80] However, this practice has been associated with the long-term loss of soil organic matter and an overall increase in the rate of soil erosion, reducing organic matter content by approximately 50% compared to naturally occurring levels in commodity crop systems in North America.[82] The widespread adoption of sustainable and higher-precision farming practices is needed for the long-term viability of our croplands.[83]

Precision agriculture methods have been in widespread use for roughly 30 years.[84] Advances in and more widespread adoption of global positioning system (GPS) and remote sensing technologies in particular have enabled precise, localized application of fertilizer and soil treatments on the sub-hectare scale.[81] This increase in precision is motivated by highly heterogeneous soil characteristics including water availability, soil electrical conductivity, and other soil properties that can vary drastically on the order of tens of meters.[85] Measurements of soil electrical conductivity have been widely used in precision agriculture because they are closely tied to soil texture.[86,87] Precision agriculture technologies include remote sensing, equipment guidance systems, and variable rate technology (VRT) for the application of fertilizer and amendments.[88] Technological adoption rates are growing steadily, with over 40% of large-scale farms using VRT applications for farming corn as of 2010, while virtually all farms in the United States use some sort of GPS-based soil, yield, or weather mapping. These practices have already paid dividends for both farmers and the environment: studies have shown both a reduction in soil nitrogen leaching and overall improvement in crop yield by using a variable application rate of fertilizer compared to a uniform rate.[89,90]

Our approach to VRT, and, more generally, crop modeling, involves the fusion of high-precision datasets with localized crop yield measurements and fertilizer treatment information in a statistical machine learning framework. In this analysis, we use our iterative Random Forest (iRF)[91] to predict crop yield at high spatial resolution (1/10 hectare), using a diverse set of measurements at spatial resolutions that span five orders of magnitude. We use the iRF package implemented in the R[92] statistical programming language. In contrast to neural networks and many other machine learning techniques, random forests and iterative Random Forest provide a natural framework for isolating variable importances. The iRF algorithm uses random intersection trees (RIT)[93] to extract stable, multi-feature interactions of arbitrary order in a computationally efficient way. This methodology enables feature space decomposition into highly interpretable sub-components. Importantly, random forests and algorithmic extensions of random forest provide a natural framework for feature space dimensionality reduction,[94] which is critical for incorporating extremely high-dimensional –omics data sources. Our iRF approach enables feature space decomposition into agronomically interpretable, low-dimensional components – for each (1/10 ha) of the field, we aim to learn the set of covariates that drive or limit yield. While iRF is, in principle, capable of learning interactions of any form or order, we find that interactions with at most 4 factors constitute principal drivers.

We package our approach in a pipeline for the automated discovery of surrogate models of agricultural systems – surrogate models from iRF, or SMiRFs. The literature on surrogate models in agriculture is largely focused on the derivation of polynomial or generalized linear models.[95,96] A disadvantage of these approaches is that they generate, by design, smooth response surfaces relating predictors and their interactions to the dependent variable. Hence, sharp, ridge-like or multi-peaked responses are difficult to learn in this fashion. In addition, high dimensional data necessarily requires dimensionality reduction before attempting to fit second- or higher-order model coefficients to avoid exponential computational scaling with the number of predictors. Using iRF to generate important response surfaces at the same computational cost as the discovery of main effects provides a tractable method of solving both of these problems simultaneously.[91] Further, in this study, our SMiRFs are nearly lossless in terms of predictive accuracy in comparison to the iRF models they approximate.

We identify soil organic matter content as a key driver of soybean productivity and find unanticipated interactions between macro- and micro-nutrients that are rarely taken into account in fertilizer prescriptions. Trials of selectively reduced fertilizer application rates at high spatial resolution enabled the reduction of phosphate and potash application while simultaneously increasing soybean yield. Hence, the discovery of control points and the use of models to guide agricultural decisions benefit both growers and the environments for which they are stewards.

2.2 Results and Methods

Table 2.1. List of sparse and dense datatypes collected at our field site for analysis.

Measurement	Location	Collection Method	Density	Resolution
UAV wavelengths (NIR, RedEdge, Red, Green, Blue)	Surface	UAV[97]	Dense	4cm x 4cm
Yield (response)	Surface	Combine harvester	Dense	10m x 3m
Slope	Surface	Veris rig	Dense	1m x 1m
Soil electrical conductivity	Subsurface	Veris rig	Dense	1m x 1m
Boron	Subsurface	Soil sample	Sparse	~20 samples
Cation exchange capacity	Subsurface	Soil sample	Sparse	~20 samples
Copper	Subsurface	Soil sample	Sparse	~20 samples
Magnesium	Subsurface	Soil sample	Sparse	~20 samples
Manganese	Subsurface	Soil sample	Sparse	~20 samples
Organic matter content	Subsurface	Soil sample	Sparse	~20 samples
pH	Subsurface	Soil sample	Sparse	~20 samples
Phosphorus	Subsurface	Soil sample	Sparse	~20 samples
Potassium	Subsurface	Soil sample	Sparse	~20 samples
Sulfur	Subsurface	Soil sample	Sparse	~20 samples
Zinc	Subsurface	Soil sample	Sparse	~20 samples

2.2.1 **Field Site Description**

Soybeans were harvested from two fields (North and South fields) comprising approximately 20 hectares located in Humphrey, Arkansas (34° 24.458′ N, 91° 40.462′ W).[85] Fertilizer amendments triple superphosphate, muriate of potash, and lime were applied using an amendment map according to optimal prescription for soybeans. Two experimental biological amendments, AgPro and Nano, were applied in contiguous regions of the South field, but were not shown to influence soybean yield. Additional details of the site were previously reported in Falco et al.[85]

2.2.2 **Data Pre-Processing and Co-Registration**

Data was collected from two soybean fields on a farm in Humphrey, Arkansas.[85] Data points near field boundaries were removed to prevent the incorporation of data artifacts due to poor collection near the field edges. Furthermore, a small region in the South field that had undergone recent precision land leveling was excluded due to poor plant growth and yield that often results from exposing subsurface soil horizons.

Some data types exhibit incompatible spatial resolutions; yield is collected with a combine harvester with a 10m wide header – hence, yield data points form pixels that are approximately 10m by 3m. Soil electrical conductivity was recorded using a Veris rig, which is smaller and has 1m^2 resolution. Since yield is our response variable, we mapped all data points for each other datatype to the nearest yield voxel. We refer to this as the "yield grid". We then averaged multiple entries of each data-type when more than one entry mapped to a single 10x3m pixel – e.g., for Veris data, each pixel on the yield-grid is the average of approximately 21 measurements.

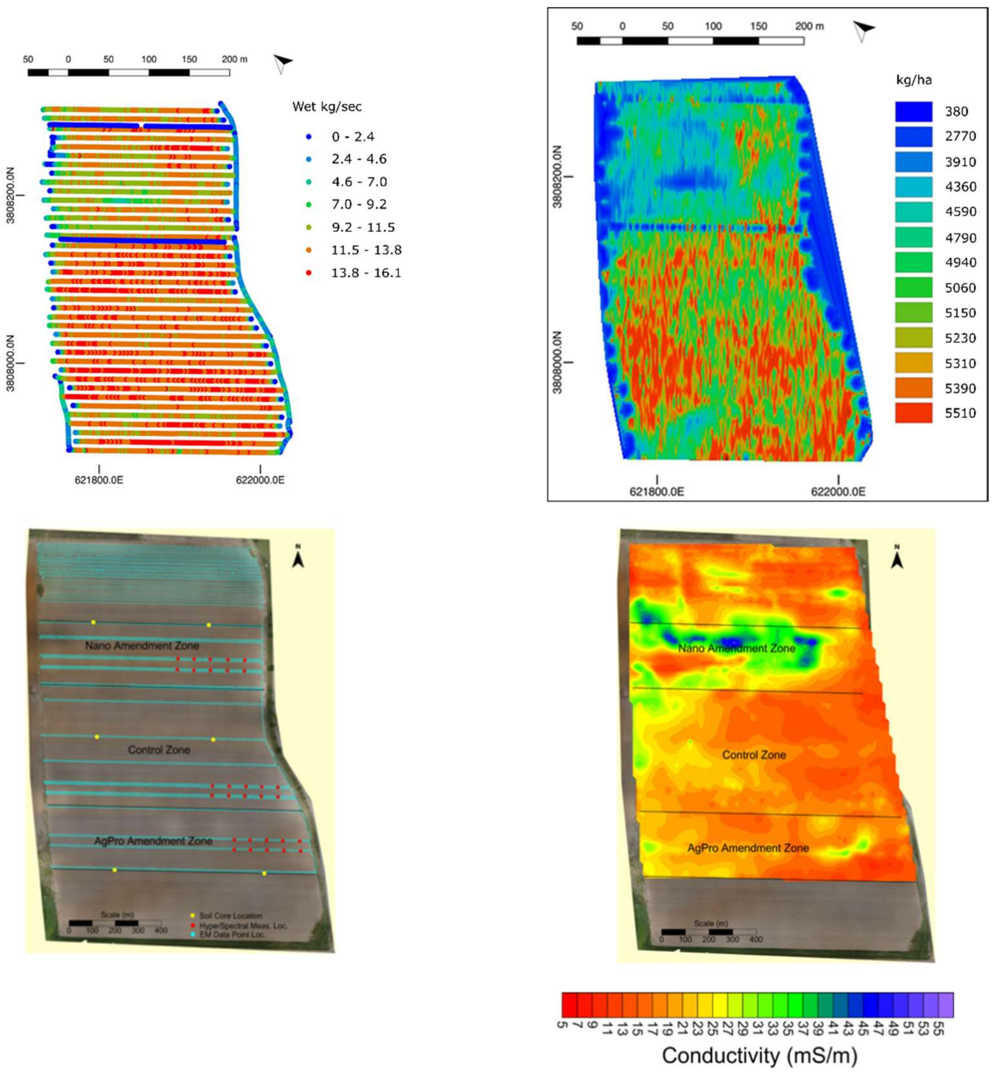


Figure 2.1. Field-scale data imputation. a) Individual measurements of yield data point collection. b) Data imputed to a 1 m by 1 m grid using bilinear interpolation. c) Collected conductivity data points. d) Data imputed to a 1 m by 1 m grid. Amendment was not found to have significant importance.

After co-registration, data were bilinearly interpolated across the fields with a 1 m by 1 m grid, yielding a total of roughly ~28,000 individual data points at the AR1K field site. Subsequently, all features other than slope were smoothed using a gaussian kernel[98] at a radius of 10 m, and the seed yield was smoothed with the same kernel at a radius of 20 m due to the large size of the yield collection grid. After smoothing the densely collected data, we processed sparse data features, including 11 soil features measured at ~20 locations in the two fields. The features included eight ion concentrations, soil pH, organic matter content, and cation exchange capacity. These sparse measurements include concentrations of macro and micronutrients (organic matter, P, K, and others), and hence are of particular value for modeling yield. However, their sparsity makes machine learning approaches intractable due to small sample sizes. To overcome this challenge, we imputed sparse features at field scale, at 1m resolution, using a random forest[99] regression model. Specifically, for each soil feature, field-scale features were sampled from a 50-m radius around each soil sample. Twenty points were selected randomly from around each soil sample, and a random forest regression was run to associate the two datasets using 500 trees. Subsequently, the entire dataset of field-scale observations was run through the random forest[99] regression model to generate field-scale predictions of the soil variables. This procedure was repeated 100 times and the predicted soil features were averaged together. The resulting model explained between 44% and 96% of variance for each soil feature (Supplementary Table 2.1). These procedures resulted in the generation of a dense map of 11 features at ~28k individual pixels across our field site. We used this dense feature map to develop an interpretable and explorable model of seed yield.

2.2.3 The Yield Model

We used an iterative Random Forest to model yield as a function of 13 subsurface variables (Table 2.1). We developed several models, some including multispectral images composed by red, blue, green and near-infrared spectral bands acquired by a UAV platform,[97] and some excluding these less interpretable spectral features. Here we focus on the model that includes only "controllable" parameters. See Supplementary Table 2.2 for results from the full model.

We used iRF with cross validation to assess model fidelity. Ensuring that training and testing sets are independent is a challenge with geospatial data due to spatial autocorrelation. We assessed spatial autocorrelation in our data by computing the variogram and observed a sharp drop-off near 20m. Hence, in each cross-validation fold, we excluded a 20m strip on either side of our test set, which was itself a 40m North-South strip in each CV fold. Test errors were on the order of 3% to 20% for different geospatially isolated test regions, with a median absolute error in seed yield of 75 kg/ha. This predictive accuracy is far greater than we would expect by chance (275 kg/ha) at a p-value of $< 10^{-1572}$, or with a linear model.

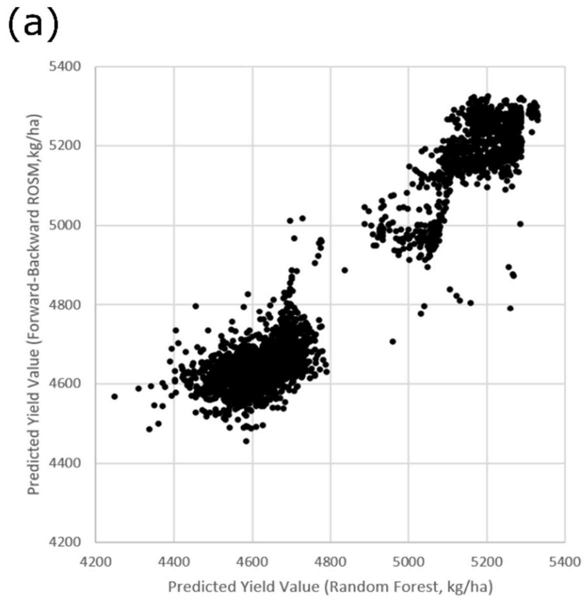
2.2.4 An Interpretable Yield Model

The iRF algorithm identifies nonlinear interactions between features and ranks these along with statistical main effects on the same scale using the weighted prevalence metric.[100] We develop a strategy for composing generalized additive models (GAMs) using individual interactions as individual terms. This procedure is similar in character to "locally interpretable model explanations" (LIME[46]) – however, our aim is to obtain a global predictor that is both locally and globally sparse. When interactions are composed of non-overlapping features, there is some theoretical justification for this framework.[101] However, there are many hyperparameters in this analysis that have yet to be optimized, which would benefit from future studies and larger datasets. Hence, the following models and analyses should be taken as proof of principle that simple surrogates can be developed to mimic the behavior and predictive power of a Random Forest in the setting of agricultural data. The full formulation of the reduced order surrogate model for a feature vector $\vec{v}$ can be expressed as:

$$f(\vec{v}) = \beta_0 + \sum_i \beta_i \sigma_i(\vec{v}) \tag{1}$$

where $\sigma_i(\vec{v})$ is the evaluation of the i-th interaction and the β are global coefficients derived from ordinary least squares (OLS) regression. The iRF model provides interactions, but not the $\sigma_i(\vec{v})$ values. To obtain these, we used the Highly Adaptive Lasso[102,103] to model yield as a function of the features in each interaction. While partial dependence plots could be used to more faithfully extract response surfaces from the Random Forest, these are optimized for fidelity, rather than predictive power. Some examples of fitted response surfaces are given in Figure 2.3. As expected, fitted response surfaces exhibit nonlinear behavior and complex dependencies between modeled features. In Figure 2.3a, the interaction between P and K is particularly interesting. We see that seed yield decreases with soil-test P over 30 mg kg^{-1} (ppm), consistent with reports in a number of agricultural systems,[104] and that optimal benefit from P requires sufficient K. While these findings have been previously reported elsewhere, we note that here they are recovered entirely from data-driven machine learning.

We fitted the reduced-order surrogate model in eq. 1 using OLS and forward-backward regression[105] (see Supplementary Table 2.3 for an exploration of different fitting strategies). Comparison of predicted values between the iRF model and the ROSM shown in Figure 2.2 reveal significant correlation between the two predictors ($R^2 \sim 0.78$), and overall predictive accuracies within 5% (77 kg/ha mean absolute error ROSM compared to 73 kg/ha iRF). We refer to the resulting ROSM as a “Surrogate Model through iRF”, or “SMiRF”.



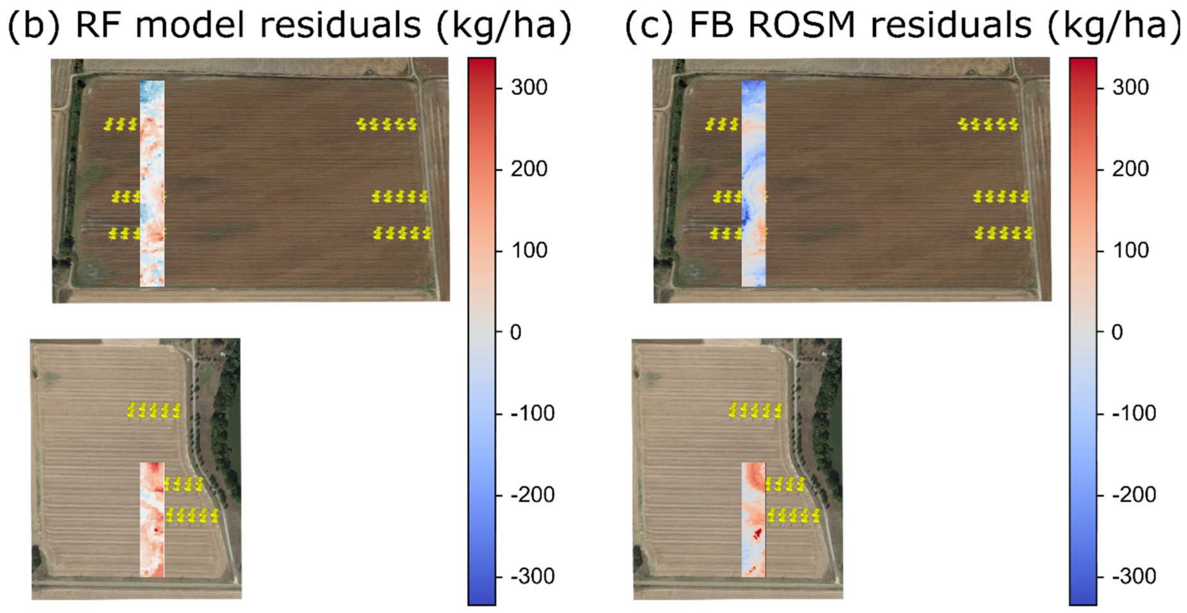
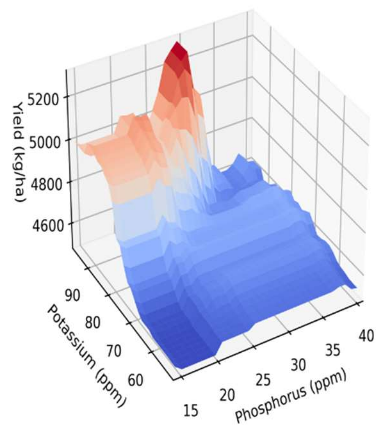
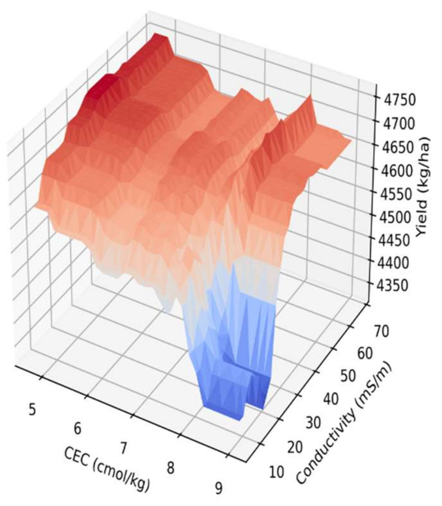


Figure 2.2. Comparison of random forest and reduced order surrogate model to predict soybean yield. a) Plot of ROSM yield residuals vs. random forest yield residuals. The two distinct clusters are due to a difference in mean yield between the North and South field. b) and c): Model residuals plotted over a held-out test strip. 20 meters of data was excluded from the training set on either side of the test strip.

(a)



(b)



(c)

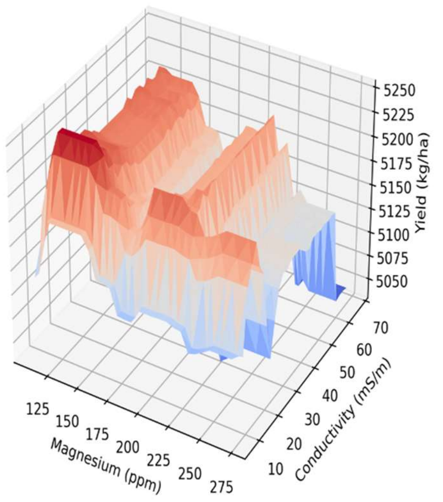


Figure 2.3. Example response surfaces from the iterative random forest. a) Sharply peaked and-like surface. b) Broad or-like surface. c) Complex multi-peaked surface.

2.2.5 The SMiRF Model as a Composition of Ecosystem Control Points

We view individual terms in the SMiRF model as detailing quantitative relationships between soil parameters and target ecosystem services – in this case soybean yield. Interrogation of individual response surfaces provides human insight into primary service drivers at each pixel on the field. One way of looking at the specific drivers of yield on a local, pixel-by-pixel basis is provided by local feature importance, or saliency maps (Supplementary Figure 2.4). These measures have the advantage that they can be computed on the parent iRF model, and do not require the surrogate. Another way to study the drivers of yield at a given pixel is to use the SMiRF model for decision support. In this setting, we posit that the SMiRF model is capturing causal information relating soybean yield to soil properties. Of course, establishing causality requires validation experiments; here we are interested in using the decision support framework as a tool for model exploration. Specifically, we optimize soil chemistry at each pixel, given the mapping between soil chemistry and soybean yield provided by the SMiRF. We can then study these optimal "policies" to ascertain the primary drivers of yield at each pixel in the field. These primary drivers correspond to ecosystem control points for soybean yield.

The following procedure is similar to feature importance measures that rely on the gradient, or smoothed versions thereof. The SMiRF model is of course amenable to gradient ascent optimization. To provide a realistic application of this procedure, we attempt to maximize Return on Investment (ROI) instead of raw yield data, which can (and does) lead to unrealistic or non-actionable policies. To optimize the ROI as a function of fertilizer amendment prescriptions, we begin with the formulation:

$$\mathrm{ROI} = \frac{\mathrm{profit}}{\mathrm{investment}} = \frac{\mathrm{crop\ price} * (\mathrm{yield\ increase}) - \mathrm{investment}}{\mathrm{investment}}$$

For a given initial feature vector $\vec{v}_0 \in V$, yield model function f defined in equation 1, and linear price function given by $p: V \to \mathbb{R} = \vec{\alpha} \cdot \vec{v}_0$ where $\vec{\alpha}$ is the per-unit price vector, maximizing the ROI is equivalent to finding:

$$\max_{\vec{v} \in V} \left(\frac{\mathrm{crop\ price} * [f(\vec{v} + \vec{v}_0) - f(\vec{v}_0)] - p(\vec{v})}{p(\vec{v})} \right)$$
$$= \mathrm{crop\ price} * \max_{\vec{v} \in V} \left(\frac{[f(\vec{v} + \vec{v}_0) - f(\vec{v}_0)]}{p(\vec{v})} \right) - 1$$

Climbing the ROI function is thus equivalent to climbing the function:

$$R(\vec{v}, \vec{v}_0) = \frac{[f(\vec{v} + \vec{v}_0) - f(\vec{v}_0)]}{p(\vec{v})}$$

which has gradient

$$\nabla R(\vec{v}, \vec{v}_0) = \frac{p(\vec{v}) * \nabla f(\vec{v} + \vec{v}_0) - [f(\vec{v} + \vec{v}_0) - f(\vec{v}_0)] * \nabla p(\vec{v})}{p(\vec{v})^2}$$

where both gradients are straightforward to compute based on the definitions of f and p.

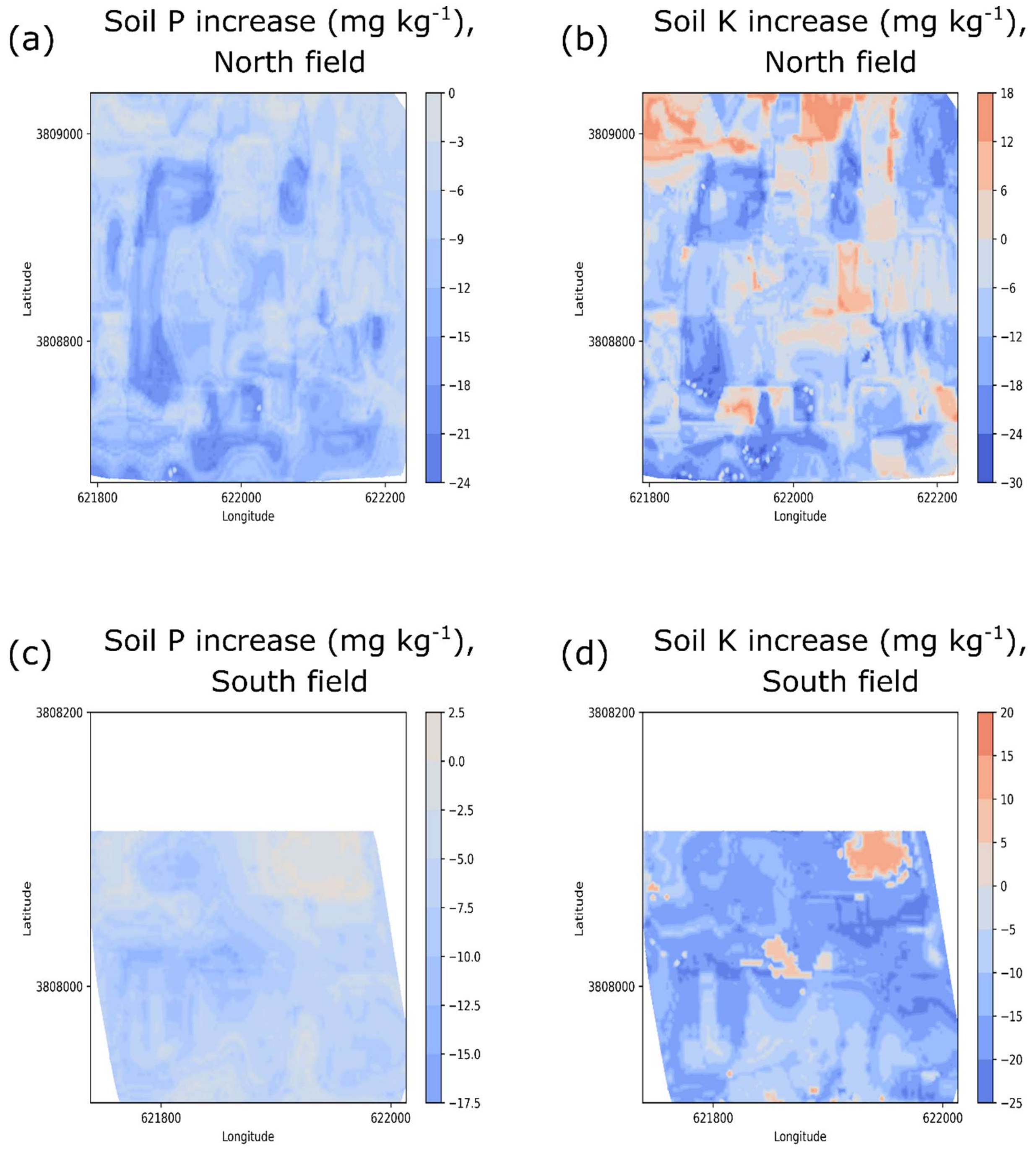

(a)
Soil P increase (mg kg^{-1}), North field
(b)
Soil K increase (mg kg^{-1}), North field
(c)
Soil P increase (mg kg^{-1}), South field
(d)
Soil K increase (mg kg^{-1}), South field
Latitude
Longitude
3809000
3808800
3808200
3808000
621800
622000
622200

Figure 2.4. Optimal amendment application maps for maximizing ROI. a) and b) show the mg kg^{-1} (ppm) increase in soil test P and K (respectively) for optimal ROI in the North field using the SMiRF. c) and d) show the same for the South field.

Triple superphosphate (TSP) and and muriate of potash (respectively) application on the North and South fields were simultaneously optimized to maximize ROI using a ROSM developed from 2017 yield data. Using gradient ascent to climb the ROI function, the iRF ROSM was used to construct maps of additional TSP and potash to be applied or removed from the field. The predicted increase in yield was from 4750 kg/ha to 4850 kg/ha using the optimized soil nutrient profile. The reduced order surrogate model offers the advantage of having an easily computable gradient everywhere in the feature space, making it a useful recapitulation of the original random forest. Although sharp boundaries in and- and or-like surfaces result in large gradients, the total differential fertilizer application in each step of the gradient ascent can be held to a maximum value to prevent numerical instability. Sharp boundaries in the ROI optimization are clearly preserved with only nominal memory overhead to store the response surfaces. This effect is due to the representation of sharper boundaries in the response in the SMiRF, thus allowing for convergence to closer local optima. In addition, the SMiRF-suggested application is relatively conservative for the increase in yield, in part due to relatively flat regions on the response surfaces corresponding to no additional optimization.

TSP and potash are predicted by our model to need to be applied or allowed to decrease due to runoff in relatively compact spatial regions in the North field; our ROSM is capable of using higher-order interactions to preserve simultaneous variable effects into both application and prediction. In order to both maximize ROI and limit potential runoff, it is important to accurately moderate fertilizer applications and avoid over-application. Further, our models indicate that over application of P, when soil-test P is above the optimal value of around 27 ppm, reduces soybean yields. Most of the south field requires no additional fertilizer K to maximize ROI. The regions where both nutrients appear over-applied in the North field correspond to very high soil-test concentrations of both P and K.

2.3 Discussion

Strong interactions between features have been identified using a spatially resolved dataset, specifically those related to soil electrical conductivity, pH, and macro nutrients. Although soil electrical conductivity is difficult to change without significant tillage or extreme application of fertilizer, this result strongly indicates the need for context-dependent amendment application.[106,107] Some of the response surfaces indicate significant and-like or or-like higher order interactions, neither of which can be neatly decomposed into single feature rules for fertilizer application or projected to polynomial-like responses. While typical farming practices seek to improve yields by adjusting several soil features including nutrient availability simultaneously, this approach is in stark contrast to true higher-order feature optimization.

Although machine learning is commonly applied to many problems related to food production, decisions related to fertilizer application and planting still rely on low-precision methods. Currently, fertilizers and amendments are generally applied to fields on the basis of sparse or non-representative soil sample analysis in an unsustainable fashion. However, recently there has been a growth in interest towards improving crop yields by applying predictive analytics in addition to using genetically modified crops. Agricultural machinery companies such as John Deere have projected a doubling of their revenue within a decade largely due to increasing demand for high precision equipment.[108]

The growing requirement for sustainable farming practices provides yet another economic driver for high-dimensional data analysis and prescription. Targeted farming techniques such as precise pesticide application have been shown to significantly reduce pesticide runoff,[109] but there is still room for optimization in the methodology of fertilizer application. Sustainability goals for fertilizer usage can only be met simultaneously with ever-increasing yield requirements by highly localized application. By identifying specific regions and multidimensional soil feature regimes most susceptible to fine tuning, the fertilizer requirements for a given field can be minimized by adjusting several features simultaneously. Ultimately, this means that the nonlinear effects of soil organic matter and nutrient concentrations can be maximally exploited without the need for costly grid soil sampling. The random forest regression model presented in this analysis provides a natural framework for partitioning the soil feature space on the basis of their suitability for improvement and is therefore ideal for economically optimizing inputs under sustainability and other constraints.

Reduced order surrogate models provide interpretable frameworks for exploring and understanding ecosystems and the control points that drive their productivity. Using response surfaces as the basis for the model allows for low-dimensional views of the data that show weighted interactions. In particular, extracted and- and or-like rules from the response surfaces demonstrate the importance of considering amendments simultaneously. The ROSM procedure provides a path to building a mechanistic understanding. Our models predict specific optima in nutrient profiles – future work is needed to test these models prospectively as decision support utilities. Toward this end, developing geospatial bootstrap and model perturbation strategies to develop confidence regions around learned response surfaces, and prediction intervals around SMiRFs presents an exciting frontier.

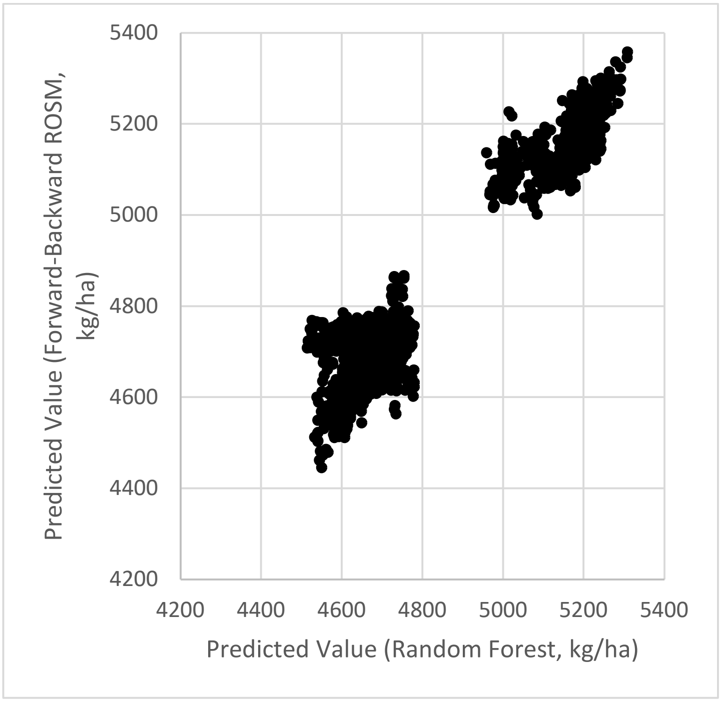
2.4 Supplement

Supplementary Table 2.1. Variance explained for each feature using a random forest model.

Feature	Proportion variance explained
Phosphorus	0.44
Potassium	0.54
pH	0.96
Zinc	0.53
Sulfur	0.61
Boron	0.62
Magnesium	0.63
Manganese	0.68
Copper	0.50
CEC	0.59
OrganicMatter	0.67

2.4.1 Results From Model with UAV Multispectral Bands Included as Parameters

Supplementary Figure 2.1. Parity plot of ROSM and random forest model using multispectral bands as parameters. The two clusters correspond to the North field (lower left) and South field (upper right).



Supplementary Table 2.2. Summary for SMiRFs with UAV bands included as features.

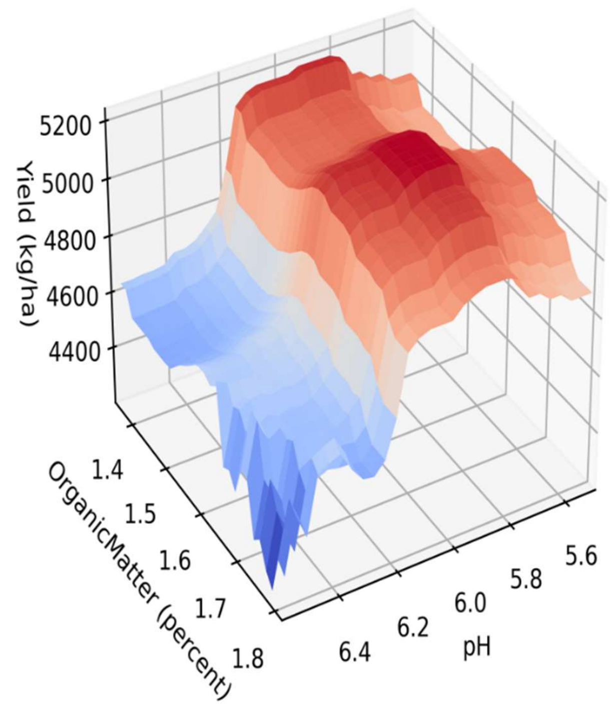
Model	Number of surfaces	Model mean absolute error (kg/ha)
Linear	36	94
Ridge	36	94
Lasso	6	77
Forward-backward	28	94

Supplementary Table 2.3. Summary for SMiRFs without UAV bands included as features.

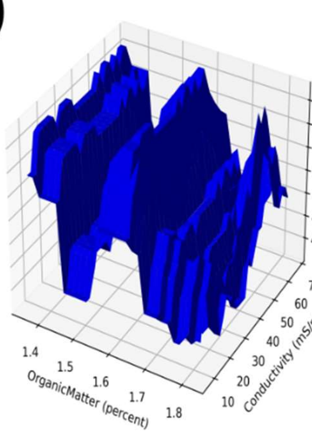
Model	Number of surfaces	Model mean absolute error (kg/ha)
Linear	90	77
Ridge	90	77
Lasso	9	86
Forward-backward	40	77

The table above shows a comparison of four linear iRF ROSMs that recapitulate a random forest regression over the North and South fields referenced throughout this study. The mean absolute error for the iRF model is 73 kilograms/hectare. The lasso regression ROSM is certainly less accurate than the other proposed surrogate models, but reproduces the random forest with a highly explainable, low-dimensional model. On the other hand, the forward-backward model has similar accuracy to both the linear and ridge regression models while eliminating a substantial fraction of the total number of response surfaces. The choice of linear coefficient model can be used as a tradeoff between complexity (and therefore computational efficiency) and accuracy.

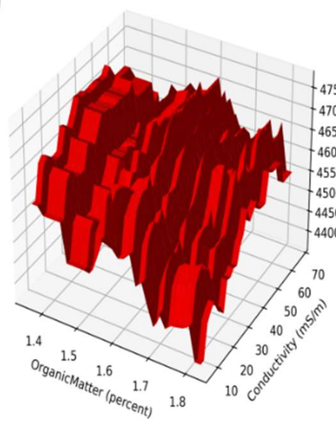
(a)



(b)



(c)



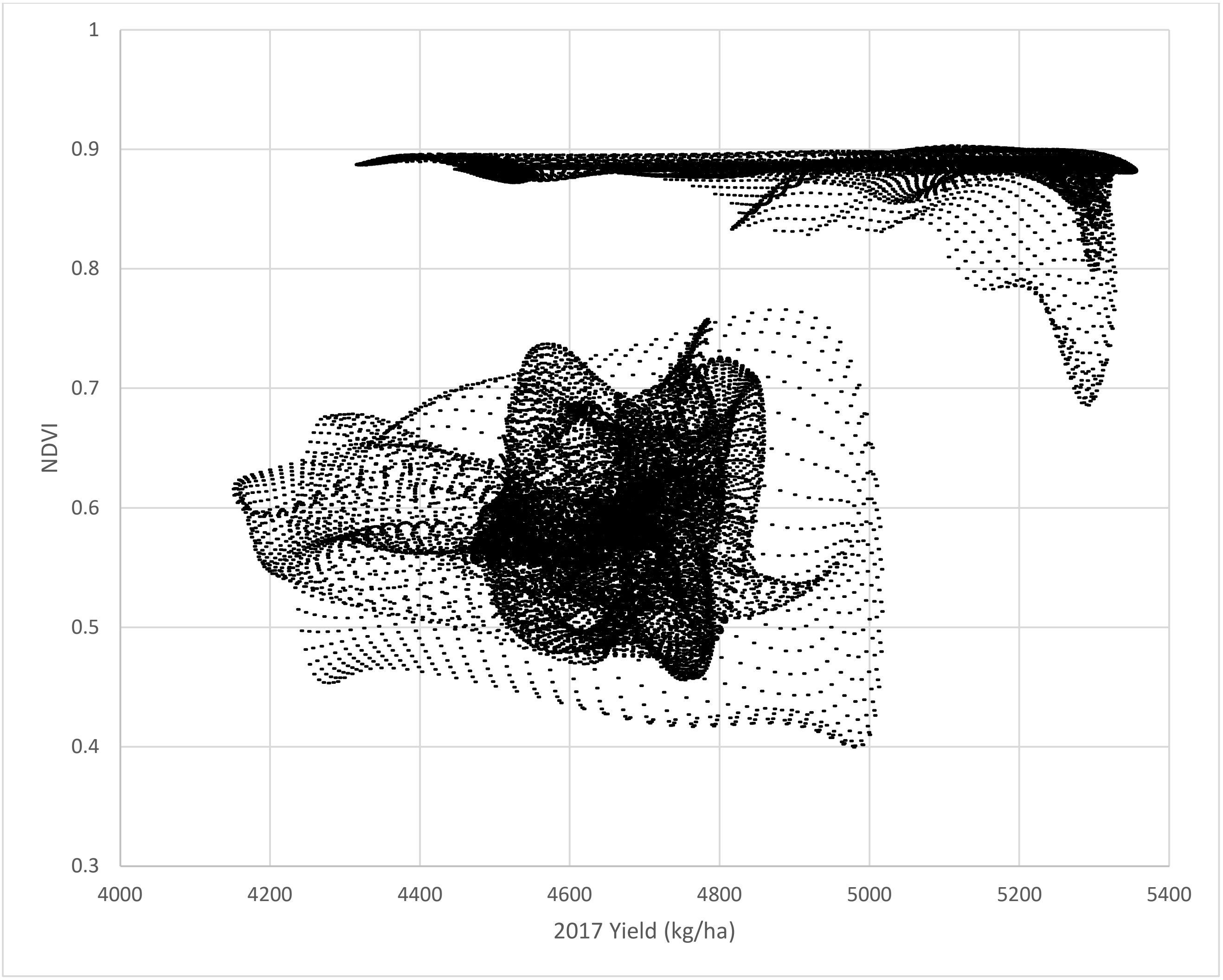
Supplementary Figure 2.2. Response surfaces highlighting the effect of organic matter on yield. b) and c) show the Conductivity-Organic Matter-pH response surface, with b) showing low pH and c) showing high pH.

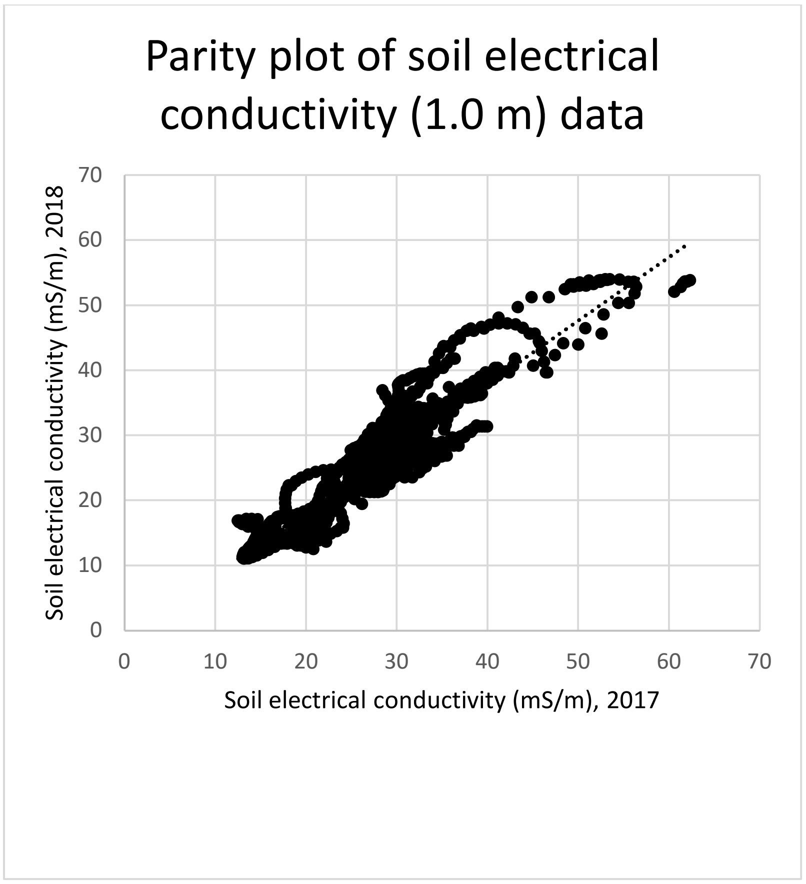
Our results indicate that soil organic matter content plays an important role in determining the soybean yield, as highlighted in Supplementary Figure 2.2, with significant higher-order dependence on both the soil pH and slope. Both response surfaces indicate that high soil organic matter (in non-clay regions) is associated with higher yield in moderate pH regimes, consistent with known agronomic science. Since tillage significantly reduces soil organic matter content, it is essential to adopt non- or low-tillage techniques to retain soil organic matter and therefore reduce the application of excess fertilizer.

2.4.2 NDVI

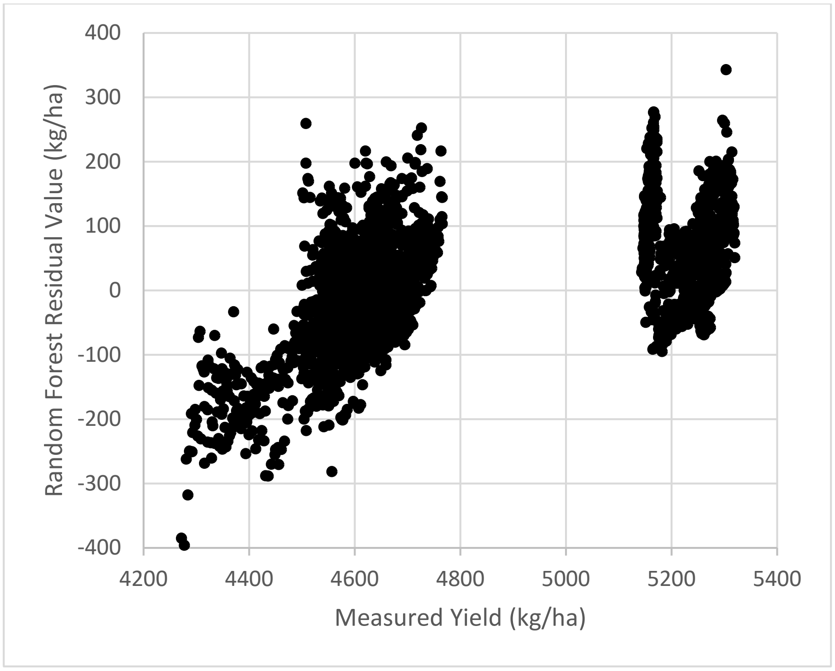
The test error for predicting NDVI with the 16 predictors was slightly lower than the test error for yield (4% vs. 12% testing error, respectively). In this study the NDVI was used implicitly to generate the field-scale soil variable maps, so this result is not surprising. While early-season NDVI assessment has been shown to be tightly coupled to yield,[110,111] our study used late-season NDVI that had essentially no correlation with the measured yield. NDVI measured in late season is generally more tightly correlated to total dry biomass, which has a noisy relationship with yield, partially explaining the lack of correlation observed in this study. In Supplementary Figure 2.3, the two clusters correspond to the North field (lower left cluster) and South field (upper right cluster).

Supplementary Figure 2.3. NDVI vs. yield.





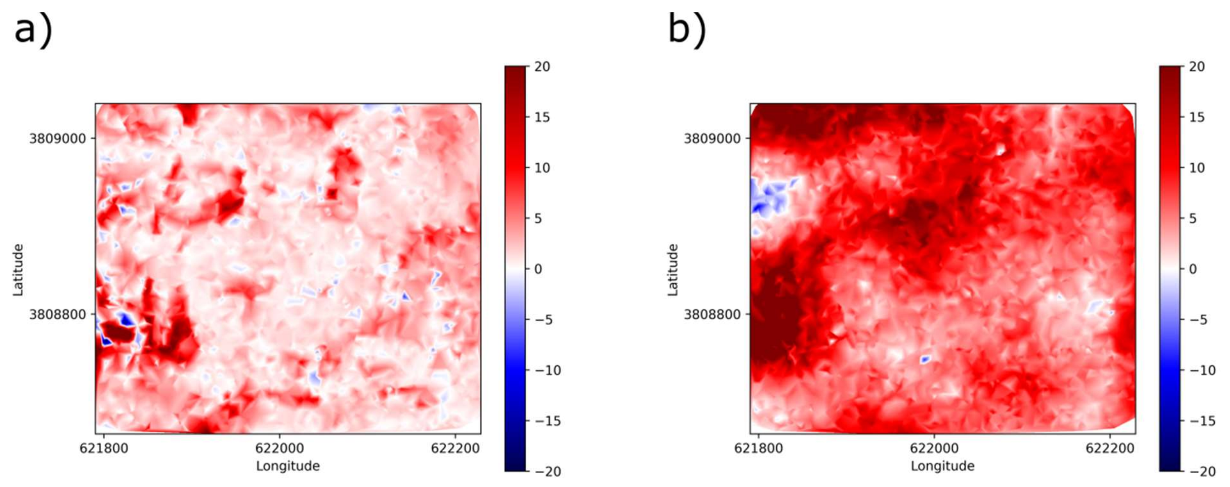
Supplementary Figure 2.4. Comparison of soil electrical conductivity in 2018 and 2017. Conductivity is largely stable over relevant timescales.



Supplementary Figure 2.5. Random forest prediction residual versus measured yield.

The yield data used to train the machine learning model was obtained from the 2017 season harvest, while high-precision soil electrical conductivity feature data used to predict it was gathered in early 2018. Soil electrical conductivity correlates strongly with soil features important to plant growth including clay content and soil water content and compared to other soil features is relatively easy to measure.[112,113] Soil electrical conductivity in the shallow subsurface (less than 0.5 meters) can vary drastically from year to year due to changes in soil moisture or topsoil ion concentrations between seasons as a consequence of runoff. However, deep soil (1.0 meter and below) measurements of soil electrical conductivity have been found to be very stable over the course of several years, indicating that these measurements provide an important context for plant growth potential in different soil regimes.

Supplementary Figure 2.4 shows a parity plot of deep (1.0 meter) soil electrical conductivity measured in 2017 and 2018, each smoothed with a Gaussian kernel at a characteristic distance of 10 meters. The nearest 90% of values to parity are shown. The data follow a linear trend with an intercept at -1.2 mS/m and an R^2 value of 0.82. In Supplementary Figure 2.5, the residual value is plotted against the measured yield for the iterative random forest regression model. The vast majority (nearly 90%) of the smoothed yield measurements are correctly predicted to within 275 kilograms per hectare. Considered together, these results indicate that the deep soil properties in these two fields are relatively stable across seasons, reducing the amount of data that must be gathered each year to accurately predict crop yield. Finally, when incorporated into the full random forest regression model with other field-scale observables, the conductivity features and higher order interactions ranked among the highest in overall importance, indicating that the predictive power of soil electrical conductivity is retained across seasons.



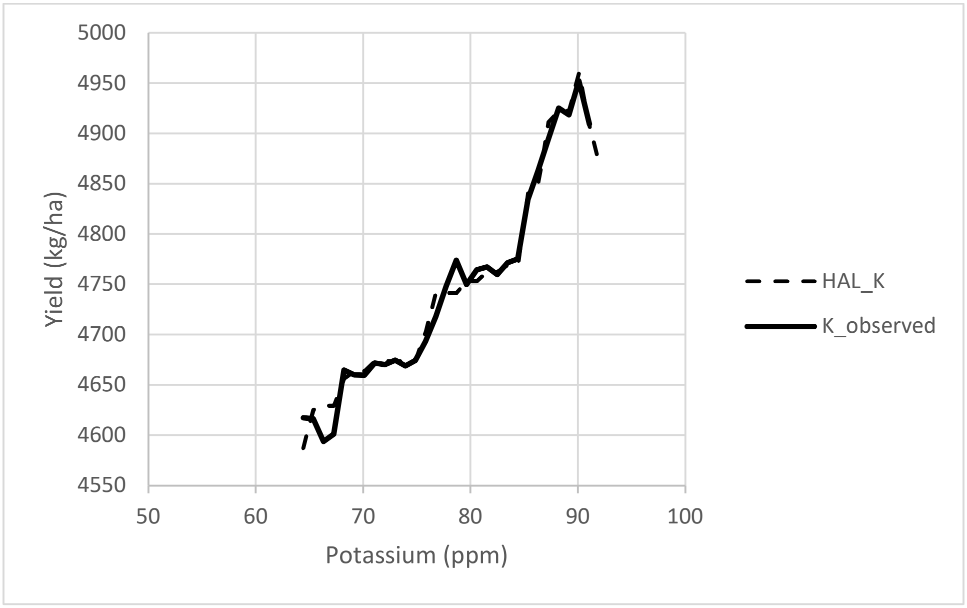
Supplementary Figure 2.6. Select local feature importances in the North field. a) Organic Matter. b) pH. Local importance is highly spatially correlated with regions of high and low spatial importance.

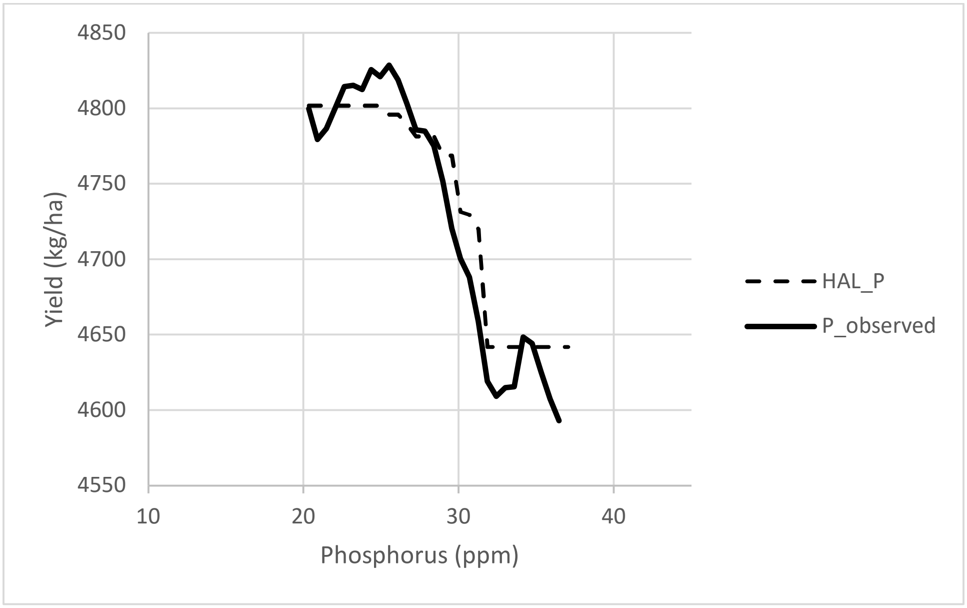
Feature local (or case) importance for regression is the average amount that the square of the residual value increases when that feature is permuted when it is out-of-bag.[99,114] The local importance was extracted from the iterative random forest regression model for each feature. Supplementary Figure 2.6 (a) and (b) shows examples of the local importance of two of the model features, the organic matter content and the pH, in one of the fields we investigated. In our analysis, pH was assigned a much higher global feature importance by the RF regression model than organic matter content, which is clear by inspection of the local importance map. Regions with large positive local importance correspond to a higher predictive power, while regions with large negative or nearly zero local importance indicate that that feature has essentially no predictive power on the observation and can safely be locally excluded from the model.

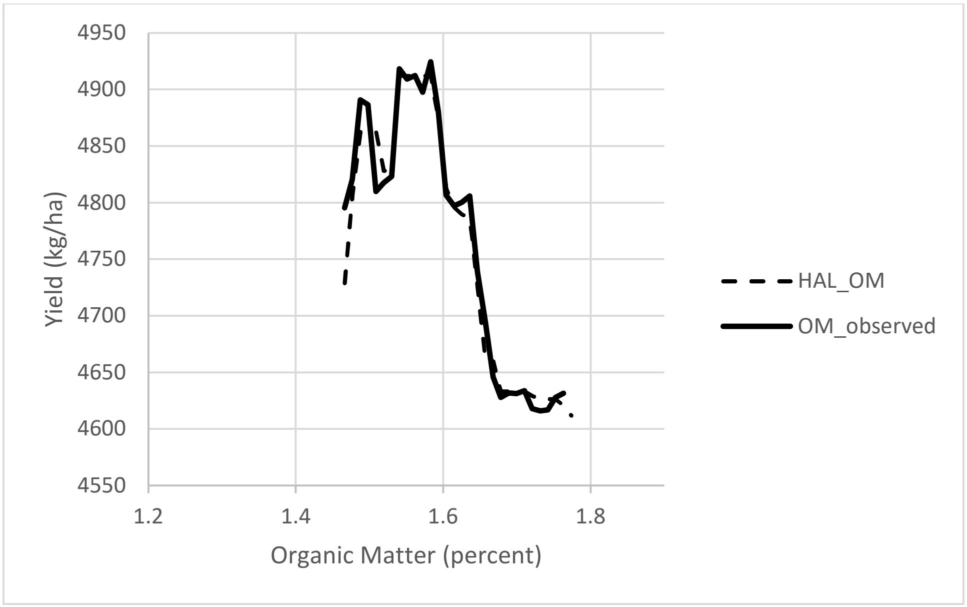
In Supplementary Figure 2.6 (a), there are distinct regions where organic matter has very high or very low local importance. Despite having a relatively low average global importance, it is clear that the model benefits from the consideration of the feature organic matter in certain locations. This is one important advantage that random forest regression has over continuous parametric models; because the random forest is constructed by repeatedly partitioning the data, even small feature space regimes where a globally unimportant feature is predictive of the observation are not lost to smoothing. As opposed to randomized or strip trial designs, the random forest is able to learn non-contiguous soil regimes well below the hectare scale that exhibit statistically different yield production. Additionally, treating the field as a continuum has the advantage of greatly increasing the effective number of observations without introducing significantly more complex methods of data collection.

By identifying regions where features have very little importance, we can potentially simplify the model of feature interactions in certain soil regimes and locally reduce the dimensionality of the model without sacrificing predictive power. Unsupervised learning techniques could be used to identify distinct soil regimes based on local importance, and separate reduced order surrogate models could be built from the most important interactions in each regime. This procedure has the potential to enable a more computationally feasible estimation of the globally optimum amendment application to most efficiently increase crop yield, which is essential both to maximize the return on investment and to sustainably minimize total environmental impact.

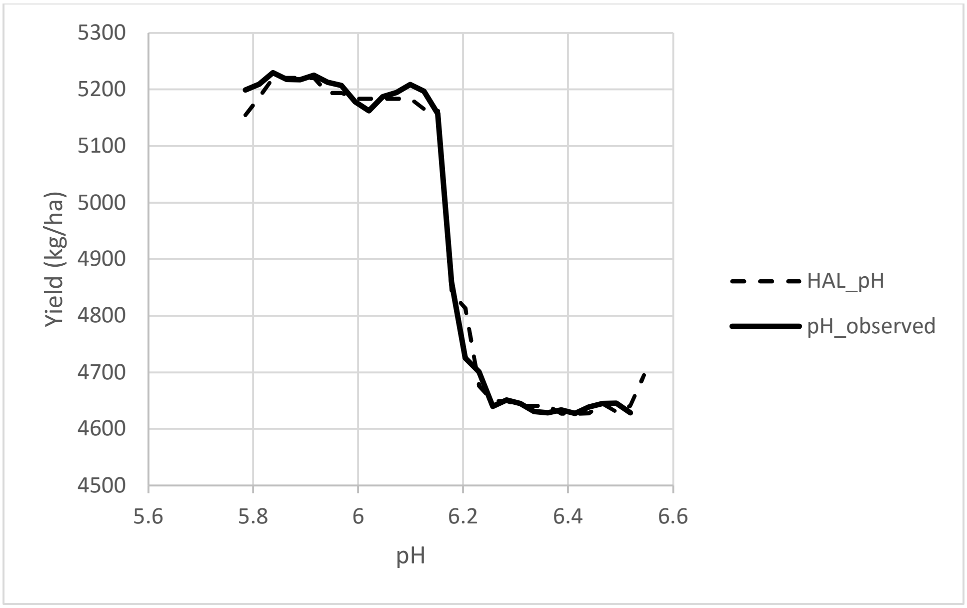
2.4.3 Yield Response Curves for Observed Variables



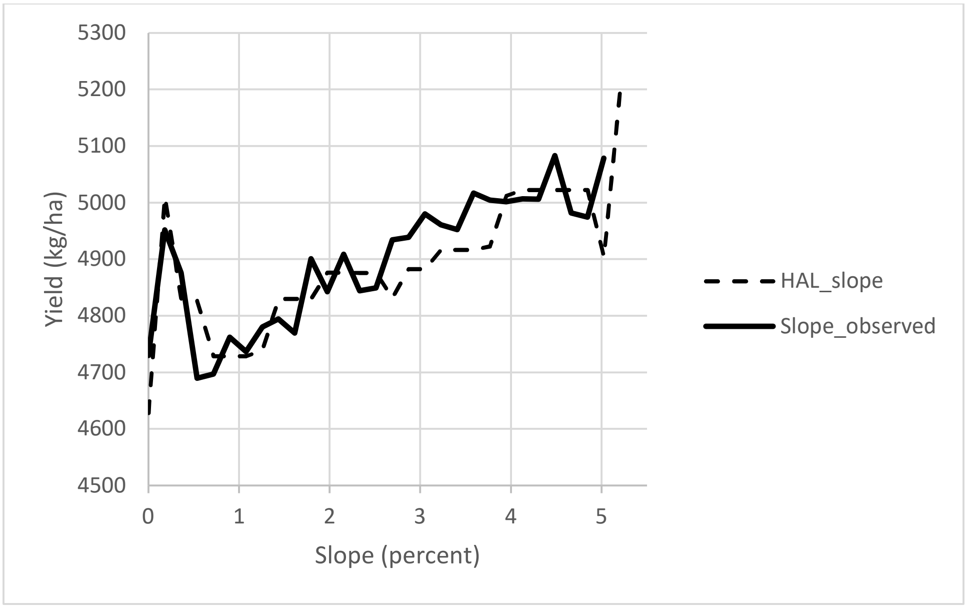


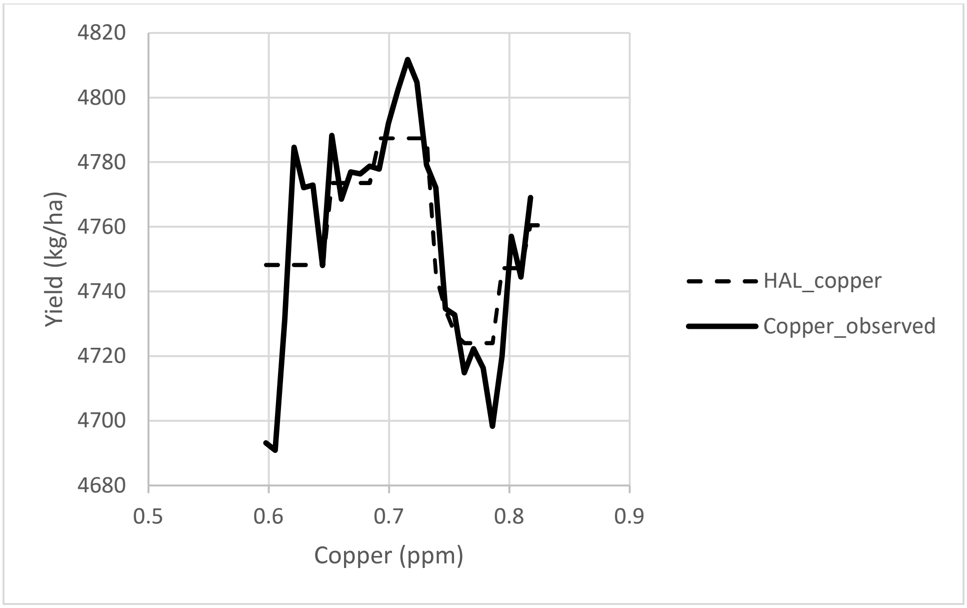


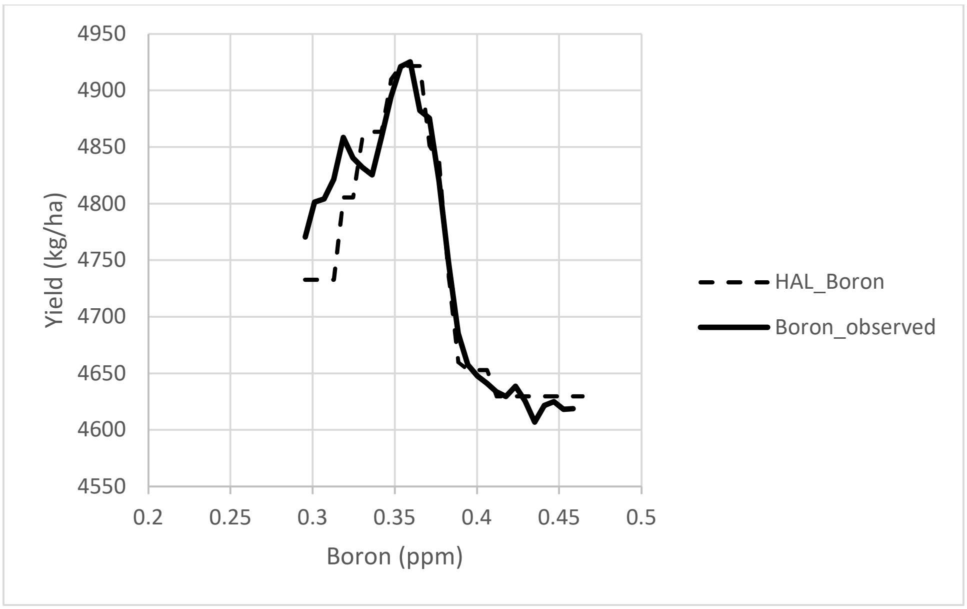
4950
4900
4850
4800
4750
4700
4650
4600
4550
Yield (kg/ha)
1.2
1.4
1.6
1.8
Organic Matter (percent)
HAL_OM
OM_observed

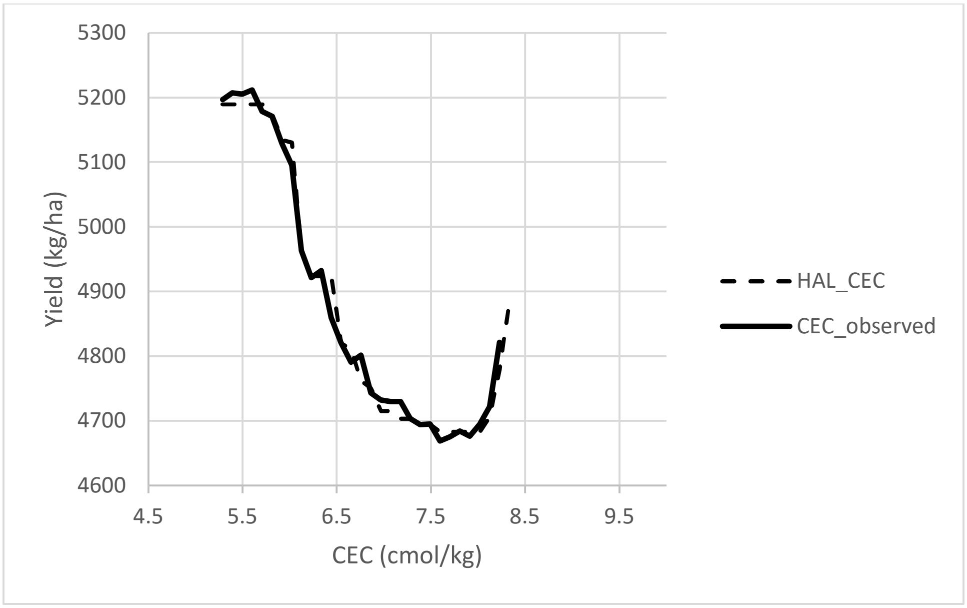


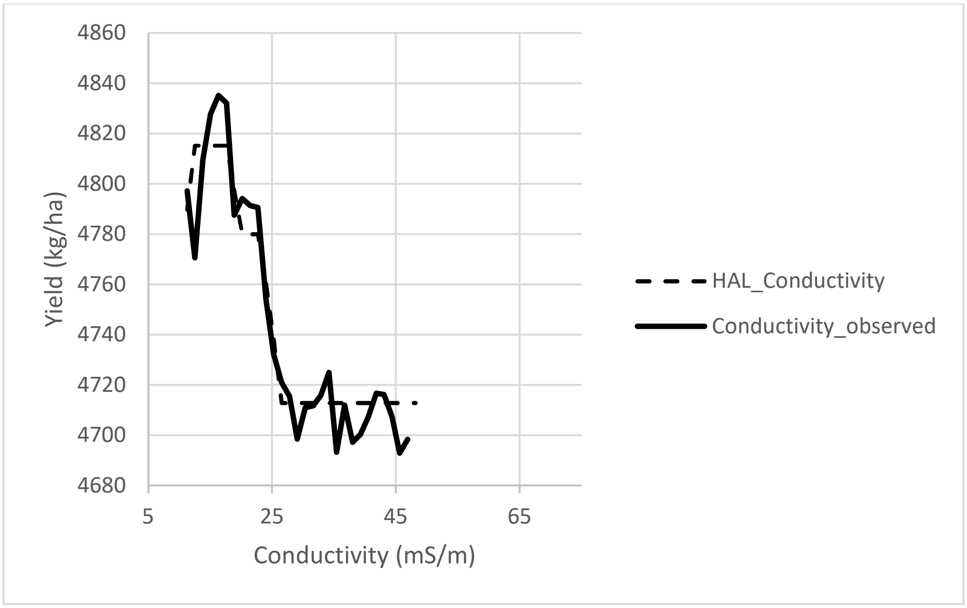
5300
5200
5100
5000
4900
4800
4700
4600
4500
Yield (kg/ha)
5.6
5.8
6
6.2
6.4
6.6
pH
HAL_pH
pH_observed


Yield (kg/ha)
Slope (percent)
HAL_slope
Slope_observed

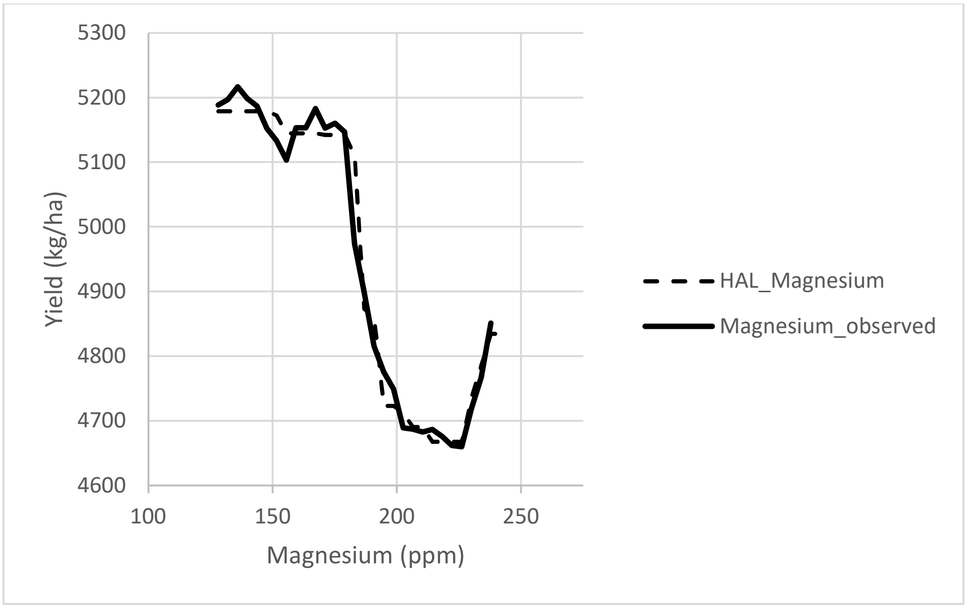

Yield (kg/ha)
Copper (ppm)
HAL_copper
Copper_observed


4950
4900
4850
4800
4750
4700
4650
4600
4550
Yield (kg/ha)
0.2
0.25
0.3
0.35
0.4
0.45
0.5
Boron (ppm)
HAL_Boron
Boron_observed

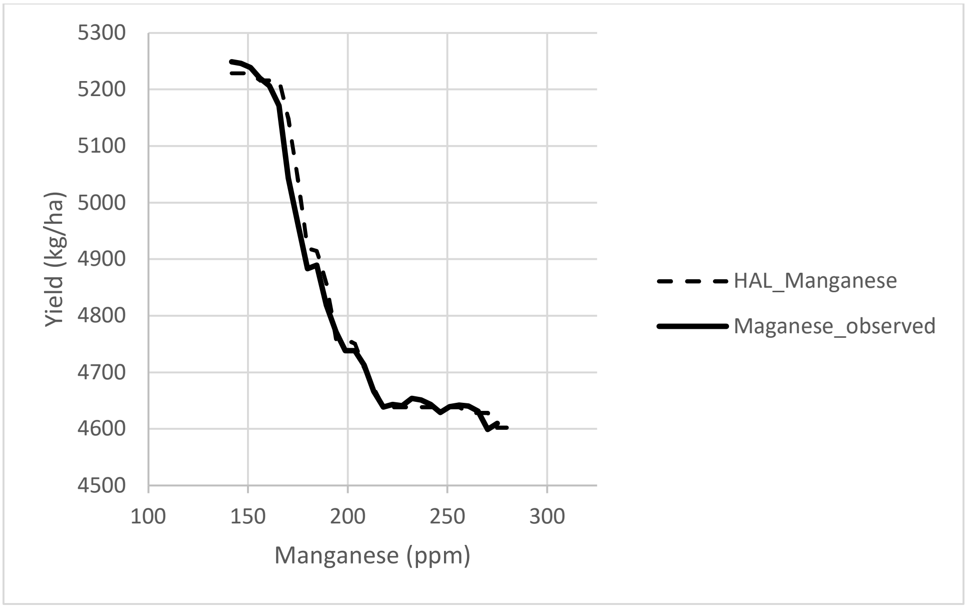

5300
5200
5100
5000
4900
4800
4700
4600
Yield (kg/ha)
4.5
5.5
6.5
7.5
8.5
9.5
CEC (cmol/kg)
HAL_CEC
CEC_observed



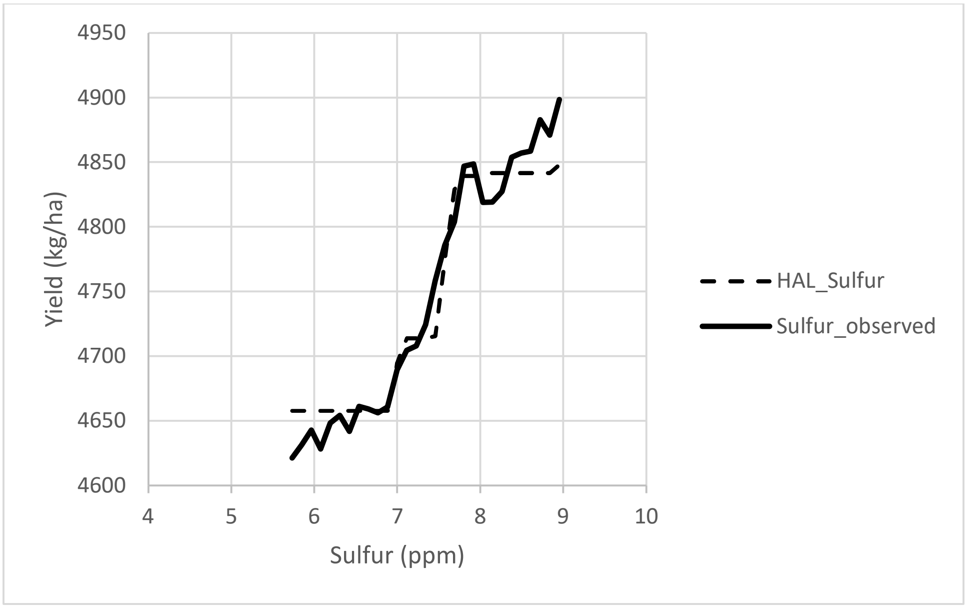
4860
4840
4820
4800
4780
4760
4740
4720
4700
4680
Yield (kg/ha)
5
25
45
65
Conductivity (mS/m)
HAL_Conductivity
Conductivity_observed



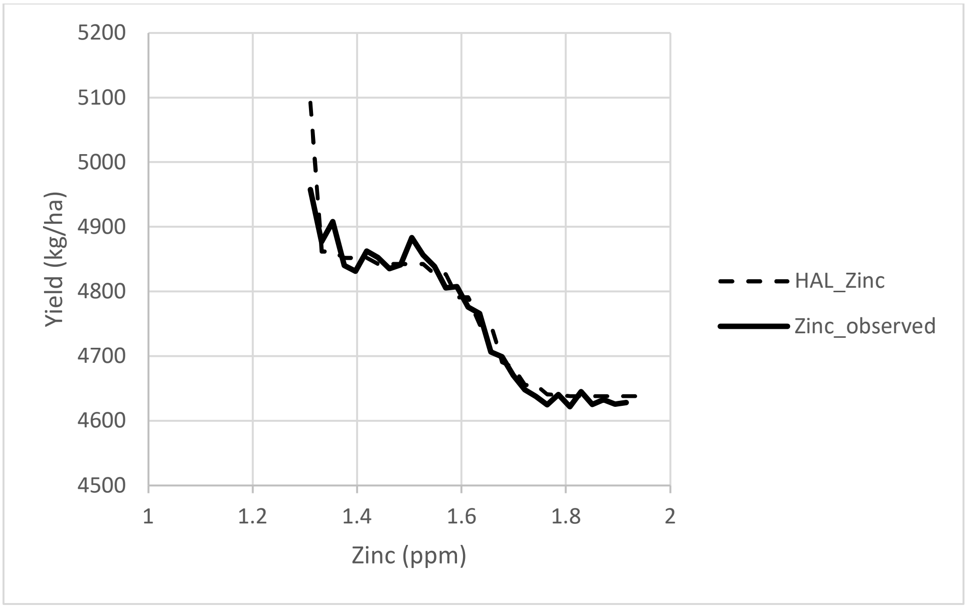
5300
5200
5100
5000
4900
4800
4700
4600
Yield (kg/ha)
100
150
200
250
Magnesium (ppm)
HAL_Magnesium
Magnesium_observed



5300
5200
5100
5000
4900
4800
4700
4600
4500
Yield (kg/ha)
100
150
200
250
300
Manganese (ppm)
HAL_Manganese
Maganese_observed



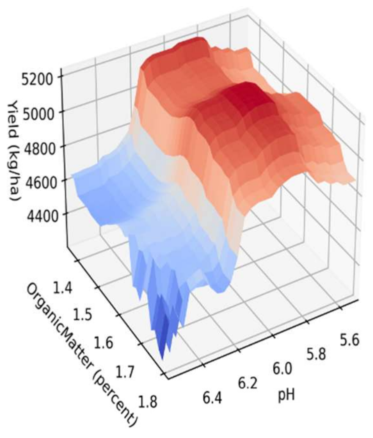
4950
4900
4850
4800
4750
4700
4650
4600
Yield (kg/ha)
4
5
6
7
8
9
10
Sulfur (ppm)
HAL_Sulfur
Sulfur_observed



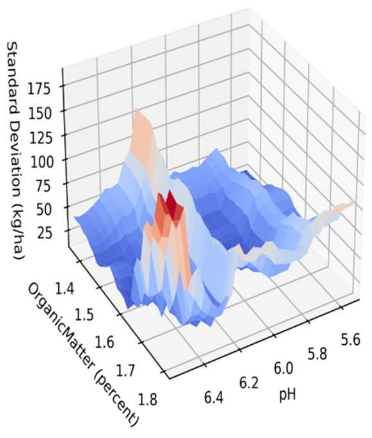
Our data indicate that the response curve for both phosphorus exhibits a noticeable drop-off at high concentration. This result indicates that part of the field had a higher than optimal amount of soil nutrient availability (since the support for each follows a roughly bell-shaped distribution centered near the median of the observed concentrations). Our overall estimated effect of fertilizer application is quite small, in part due to including both the soil concentration variables and application rates as observations to the model. It is also important to note that risk associated with using fertilizer conservatively is vastly outweighed due to risk associated with weather and market volatility. For example, significant confounding irrigation conditions such as flooding caused the mean yield to decrease in these two fields from 4750 to 4100 kg/ha in 2018, which is far outside of the predicted variability due to changes in soil chemistry alone.

Approximately 120 response surfaces were generated for each two- and three-feature interaction by training a HAL model[102,103] on random samples of 5% of the data. Examples of averaged surfaces and the pointwise standard deviation are presented in Supplementary Figure 2.7. Less important features had extremely tight uncertainty envelopes around the response surface. In contrast, some of the most significant interactions had local response uncertainties around 100 kg/ha.

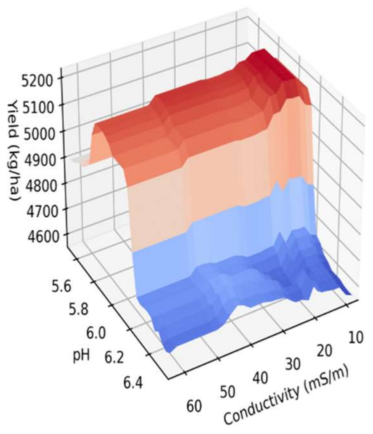
(a)



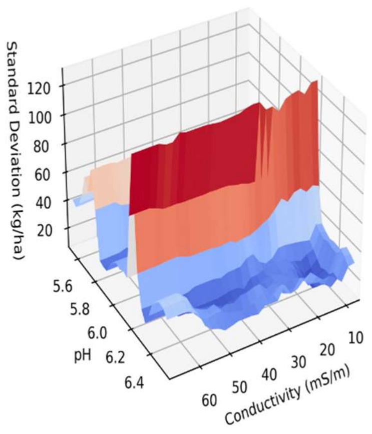
(b)



(c)

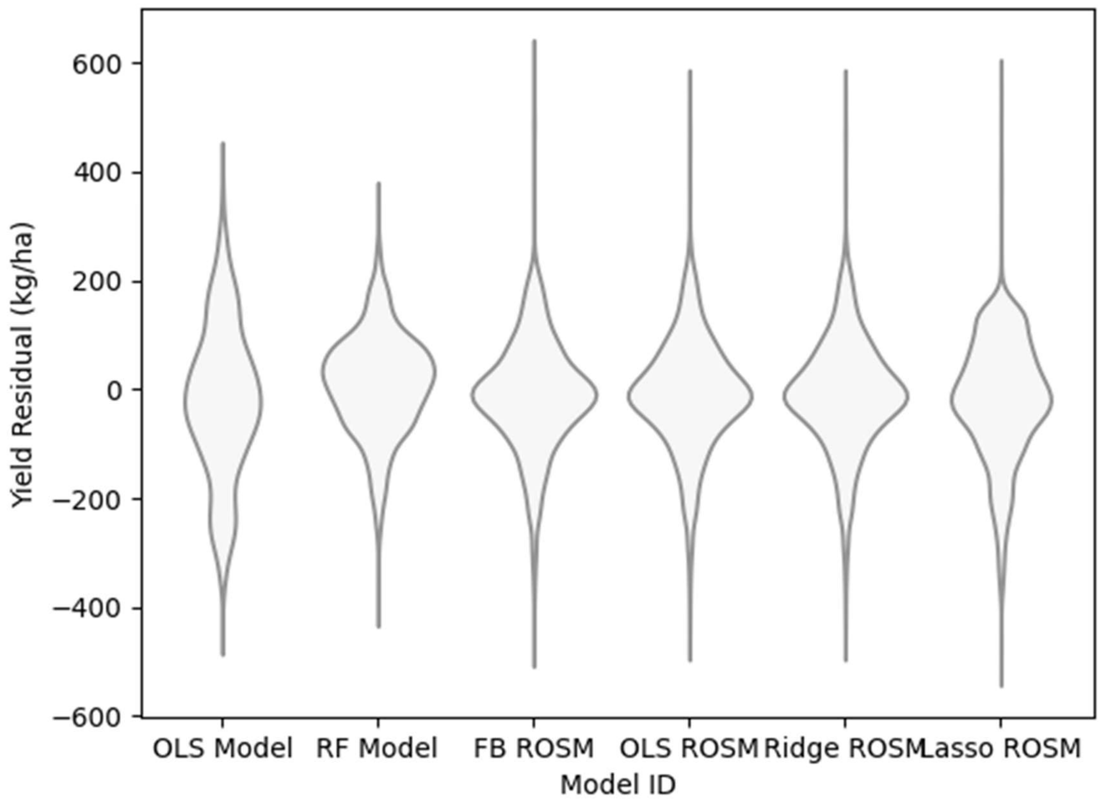


(d)



Supplementary Figure 2.7. Response surfaces and deviation surfaces. a) and c) are response surfaces, with corresponding pointwise deviation surfaces shown in b) and d).

Supplementary Figure 2.8. Yield residual distribution for various machine learning models.



The random forest model has a much more narrow yield residual distribution compared to the OLS model, but is centered further from zero due to soil regimes found in the test data that are not present in the training dataset. This is largely an artifact of the relatively large, geospatially correlated region chosen as the test set. The FB, OLS, and Ridge ROSM residuals have very similar distributions centered much closer to 0, while the Lasso ROSM distribution is more broad and exhibits systematic bias. However, the Lasso model uses far fewer surfaces, and can be very useful as a low-order parametric approximation of the random forest.

Supplementary Table 2.4. Random Forest feature interaction prevalences.

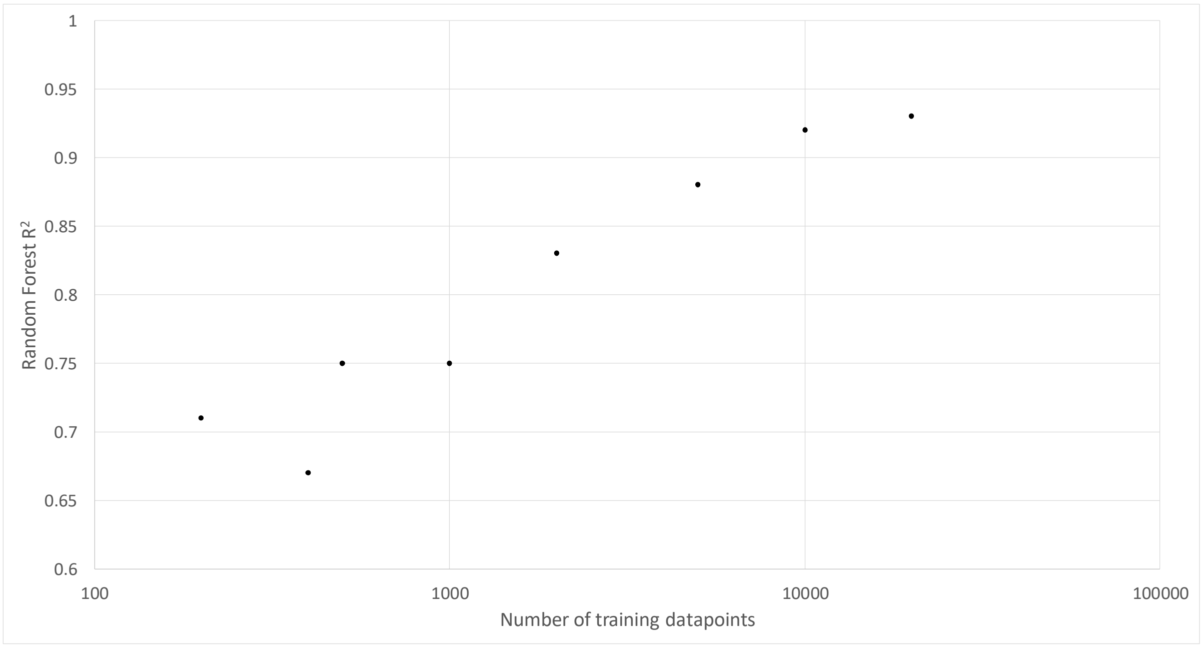
Interaction	Prevalence
Conductivity_pH	0.93
Slope_pH	0.91
Slope_Conductivity_pH	0.85
pH_OrganicMatter	0.76
pH_Manganese	0.74
Conductivity_pH_OrganicMatter	0.71
Slope_pH_OrganicMatter	0.70
Conductivity_pH_Manganese	0.69
Slope_pH_Manganese	0.67
Slope_Conductivity_pH_OrganicMatter	0.66
Slope_Conductivity_pH_Manganese	0.62
pH_CEC	0.59
pH_Manganese_OrganicMatter	0.56
Phosphorus_pH	0.56
Conductivity_pH_CEC	0.55
pH_Boron	0.55
Slope_pH_CEC	0.54
Conductivity_pH_Manganese_OrganicMatter	0.53
Conductivity_Phosphorus_pH	0.52
Conductivity_pH_Boron	0.51
Slope_pH_Manganese_OrganicMatter	0.51
Slope_Conductivity_pH_CEC	0.51
Slope_Phosphorus_pH	0.51
Slope_pH_Boron	0.50
pH_Magnesium	0.49
Slope_Conductivity_pH_Manganese_OrganicMatter	0.48
Slope_Conductivity_Phosphorus_pH	0.48
pH_Copper	0.48
Slope_Conductivity_pH_Boron	0.47
Conductivity_pH_Magnesium	0.47
pH_CEC_OrganicMatter	0.46
Slope_pH_Magnesium	0.46
Conductivity_pH_Copper	0.45
Slope_pH_Copper	0.44
Conductivity_pH_CEC_OrganicMatter	0.43
Slope_Conductivity_pH_Magnesium	0.43
Phosphorus_pH_OrganicMatter	0.43
pH_Manganese_CEC	0.43
Slope_pH_CEC_OrganicMatter	0.42

Slope_Conductivity_pH_Copper	0.42
Phosphorus_pH_Manganese	0.41
Conductivity_Phosphorus_pH_OrganicMatter	0.41
pH_Boron_OrganicMatter	0.41
pH_Boron_Manganese	0.41
Slope_Conductivity_pH_CEC_OrganicMatter	0.40
Conductivity_pH_Manganese_CEC	0.40
Slope_Phosphorus_pH_OrganicMatter	0.40
Slope_pH_Manganese_CEC	0.39
Conductivity_pH_Boron_OrganicMatter	0.39
Conductivity_Phosphorus_pH_Manganese	0.38
pH_Magnesium_OrganicMatter	0.38
Conductivity_pH_Boron_Manganese	0.38
Slope_Conductivity_Phosphorus_pH_OrganicMatter	0.38
Slope_Phosphorus_pH_Manganese	0.37
Slope_pH_Boron_OrganicMatter	0.37
Slope_Conductivity_pH_Manganese_CEC	0.37
Slope_pH_Boron_Manganese	0.37
Conductivity_pH_Magnesium_OrganicMatter	0.37
pH_Magnesium_Manganese	0.36
Slope_pH_Magnesium_OrganicMatter	0.36
Slope_Conductivity_pH_Boron_OrganicMatter	0.35
Slope_Conductivity_Phosphorus_pH_Manganese	0.35
pH_Zinc	0.35
Slope_pH_Copper_OrganicMatter	0.35
Slope_Conductivity_pH_Boron_Manganese	0.34
Conductivity_pH_Magnesium_Manganese	0.34
Slope_Conductivity_pH_Magnesium_OrganicMatter	0.34
Slope_pH_Magnesium_Manganese	0.33
pH_Manganese_CEC_OrganicMatter	0.33
Conductivity_pH_Manganese_Copper	0.33
pH_Boron_CEC	0.33
Slope_pH_Manganese_Copper	0.33
Conductivity_pH_Zinc	0.33
Slope_Conductivity_pH_Copper_OrganicMatter	0.33
Slope_pH_Zinc	0.32
Slope_Conductivity_pH_Magnesium_Manganese	0.32
Phosphorus_pH_Manganese_OrganicMatter	0.32
Conductivity_pH_Boron_CEC	0.31
Conductivity_Phosphorus_pH_CEC	0.31
Conductivity_pH_Manganese_CEC_OrganicMatter	0.31

Phosphorus_pH_Boron	0.31
Slope_Conductivity_pH_Manganese_Copper	0.31
Slope_pH_Manganese_CEC_OrganicMatter	0.31
Slope_Phosphorus_pH_CEC	0.31
Slope_pH_Boron_CEC	0.31
Slope_Conductivity_pH_Zinc	0.30
pH_Boron_Manganese_OrganicMatter	0.30
Conductivity_Phosphorus_pH_Manganese_OrganicMatter	0.30
Conductivity_Phosphorus_pH_Boron	0.29
Slope_Conductivity_Phosphorus_pH_CEC	0.29
Slope_Conductivity_pH_Manganese_CEC_OrganicMatter	0.29
Slope_Conductivity_pH_Boron_CEC	0.29
Slope_Phosphorus_pH_Manganese_OrganicMatter	0.29
Conductivity_pH_Boron_Manganese_OrganicMatter	0.28
Slope_Phosphorus_pH_Boron	0.28
pH_Magnesium_Manganese_OrganicMatter	0.28
Conductivity_pH_Magnesium_CEC	0.28
Conductivity_Phosphorus_pH_Magnesium	0.28
Slope_pH_Copper_CEC	0.27
Slope_Conductivity_Phosphorus_pH_Manganese_OrganicMatter	0.27
Slope_pH_Boron_Manganese_OrganicMatter	0.27
Slope_Conductivity_Phosphorus_pH_Boron	0.27
Conductivity_pH_Magnesium_Manganese_OrganicMatter	0.27
pH_Zinc_OrganicMatter	0.27
Slope_Conductivity_pH_Magnesium_CEC	0.26
Conductivity_pH_Boron_Magnesium	0.26
Phosphorus_pH_CEC_OrganicMatter	0.26
Slope_Conductivity_pH_Copper_CEC	0.26
Slope_pH_Magnesium_Manganese_OrganicMatter	0.26
Slope_Conductivity_Phosphorus_pH_Magnesium	0.26
Conductivity_pH_Manganese_Copper_OrganicMatter	0.26
Slope_Conductivity_pH_Boron_Manganese_OrganicMatter	0.26
Slope_pH_Manganese_Copper_OrganicMatter	0.26
pH_Zinc_Manganese	0.26
Slope_Phosphorus_pH_Copper	0.25
Slope_Conductivity_pH_Magnesium_Manganese_OrganicMatter	0.25
Slope_Conductivity_pH_Boron_Magnesium	0.24
Slope_pH_Zinc_OrganicMatter	0.24
Slope_Phosphorus_pH_CEC_OrganicMatter	0.24
Slope_Conductivity_pH_Manganese_Copper_OrganicMatter	0.24
Slope_Conductivity_Phosphorus_pH_CEC_OrganicMatter	0.23

Slope_pH_Zinc_Manganese	0.23
pH_Copper_CEC_OrganicMatter	0.23
Conductivity_pH_Boron_Manganese_CEC	0.23
Conductivity_Phosphorus_pH_Manganese_CEC	0.23
Slope_Conductivity_pH_Boron_CEC_OrganicMatter	0.22
Slope_pH_Boron_Manganese_CEC	0.22
Conductivity_Phosphorus_pH_Magnesium_OrganicMatter	0.22
Slope_Conductivity_pH_Zinc_Manganese	0.22
Slope_Conductivity_pH_Copper_CEC_OrganicMatter	0.21
Slope_Conductivity_Phosphorus_pH_Magnesium_OrganicMatter	0.21
Slope_Conductivity_Phosphorus_pH_Boron_OrganicMatter	0.21
Slope_Conductivity_Potassium_pH	0.21
Conductivity_Phosphorus_pH_Magnesium_Manganese	0.20
Conductivity_pH_Boron_Magnesium_OrganicMatter	0.20
Slope_Phosphorus_pH_Magnesium_Manganese	0.20
Slope_Conductivity_Phosphorus_pH_Copper_OrganicMatter	0.19
Slope_Conductivity_pH_Zinc_CEC	0.19
pH_Boron_Manganese_CEC_OrganicMatter	0.18
Slope_Conductivity_pH_Boron_Copper_OrganicMatter	0.18
Slope_Conductivity_pH_Magnesium_Copper_OrganicMatter	0.18
Conductivity_Phosphorus_pH_Manganese_CEC_OrganicMatter	0.18
Slope_Conductivity_Phosphorus_pH_Manganese_Copper	0.18
Slope_pH_Boron_Manganese_CEC_OrganicMatter	0.17
Slope_Conductivity_pH_Zinc_Manganese_OrganicMatter	0.17
Slope_Conductivity_pH_Magnesium_Manganese_Copper	0.16
Slope_Conductivity_pH_Boron_Manganese_CEC_OrganicMatter	0.16
Conductivity_pH_Magnesium_Manganese_CEC_OrganicMatter	0.16
Slope_Conductivity_Phosphorus_pH_Magnesium_Copper	0.14
Conductivity_pH_Zinc_Magnesium_OrganicMatter	0.14
Slope_Conductivity_Phosphorus_pH_Boron_CEC_OrganicMatter	0.13
Conductivity_Phosphorus_Potassium_pH	0.13
Slope_Conductivity_pH_Zinc_Manganese_CEC_OrganicMatter	0.11

Supplementary Figure 2.9. Model accuracy versus number of training points.



2.5 Code

2.5.1 Python Script to Match Soil Data to Measured Field-Scale Data (Randomly_Bin.Py)

```
import os
import random
import numpy as np
import sys

soil_pos_indices = [4,5]

def get_lines(num):
#simply reads the lines of a file name
    infile = open(sys.argv[num], 'r')
    inlines = infile.readlines()
    infile.close()
    return inlines

def get_point_lines(soil_lines, x_col, y_col):
#get positions
    point_lines = [line[:-1] for line in soil_lines if not line[:1] == '#']  #cuts out comments and
newlines
    point_lines = [line for line in point_lines if not line == '']
    point_lines = [line.split(',')[x_col:y_col+1] for line in point_lines  ]
    point_lines = [ np.array([float(entry[0]), float(entry[1]) ])  for entry in point_lines]
#    print(point_lines)
    return point_lines


def get_dist_mat(soil_lines, x_col, y_col):
#returns a distance matrix from points
    point_lines = get_point_lines(soil_lines, x_col, y_col)
    outmat = [[0 for y in range(len(point_lines))] for x in range(len(point_lines))]
    for x in range(len(point_lines)):
        for y in range(len(point_lines)):
            delta = point_lines[x] - point_lines[y]
            delta2 = delta**2
            dist = sum(delta2)**0.5
            outmat[x][y] = dist
    return outmat


def get_closest_points(soil_lines, field_lines, x_col, y_col, x_col2, y_col2):
#get an index array of the closest soil point for each non-soil point
```

```
#matches field-scale observable points to non-field scale (soil sample) points
    soil_point_lines = get_point_lines(soil_lines, x_col, y_col)
    field_point_lines = get_point_lines(field_lines, x_col2, y_col2)
    outarray = []

#    print(field_point_lines[:10])
#    print(soil_poin00t_lines[0])

    for fpi in range(len(field_point_lines)):
    #field point index
        field_point = field_point_lines[fpi]
        mindex = -1
        minval = 20         #minimum distance between the soil values and field-scale values
        #minval should be less than the distance between soil samples
        for spi in range(len(soil_point_lines)):
        #soil point index
            soil_point = soil_point_lines[spi]
            delt = soil_point - field_point
            delt2 = delt**2
            dist = sum(delt2)**0.5

            if dist < minval:
                minval = dist
                mindex = spi
        outarray.append(mindex)        #append the index of the field point value
#        print(mindex)
    return outarray

def get_dict(soil_lines, mapping_array):
#returns a dictionary assigning soil indices to field indices
    soil_point_lines = get_point_lines(soil_lines, soil_pos_indices[0], soil_pos_indices[1])
    outdict = {}
    for x in range(len(soil_point_lines)):
        outdict[x] = []
    outdict[-1] = []        #initialize as empty; need unassigned one for observables not near soil
samples

    for y in range(len(mapping_array)):
        index = mapping_array[y]
        outdict[index].append(y)
    return outdict


soil_lines = get_lines(1)
soil_lines = [line for line in soil_lines if len(line.split()) > 0]
soil_lines = [line for line in soil_lines if not line[:1] == '#']
```

```
yield_lines = get_lines(2)
random.shuffle(yield_lines)
#print('Shuffling complete')
#yield_lines = yield_lines[:10000]

random_points = int(sys.argv[3])        #number of points near each soil sample to take
yield_smoothing_index = sys.argv[4]


dist_mat = get_dist_mat(soil_lines, soil_pos_indices[0], soil_pos_indices[1])     #distance matrix
of points in soil lines
minmat = [min([subentry for subentry in entry if subentry > 0]) for entry in dist_mat]

mapping_array = get_closest_points(soil_lines, yield_lines, soil_pos_indices[0],
soil_pos_indices[1], 0, 1)
reduced_mapping_array = [entry for entry in mapping_array if not entry == -1]
#eliminates those points that are not near soil samples

soil_to_field_dict = get_dict(soil_lines, mapping_array)


outfile = open('randomly_matched_{0}.csv'.format(yield_smoothing_index), 'w')
for soil_index in soil_to_field_dict.keys():
#now we retain only soil sample matches with enough field-scale observations
    if len(soil_to_field_dict[soil_index]) > random_points and not (soil_index == -1):
        temp_array = soil_to_field_dict[soil_index]
        random.shuffle(temp_array)
        subarray = temp_array[:random_points]     #uncomment this line for NON fullscale...
#       subarray = temp_array[:]
        for subarray_index in subarray:
            outfile.write(yield_lines[subarray_index][:-1] + ',' + ','.join([entry for entry in
soil_lines[soil_index][:-1].split(',') if entry]) + ',' + str(soil_index+1) + '\n' )
outfile.close()
```

2.5.2 R Script to Run Regression Generating Imputed Field-Scale Soil Data (iRF_on_data_UAV_soil.R)

```
###This program runs regression of soil variables against UAV data locally, then uses field-scale
UAV to extrapolate
###regression on both the UAV AND the EC data
args = commandArgs()
args


library(httr)
httr::set_config(config(ssl_verifypeer = 0L))
set.seed(57)

field_1_mapped_reduced <- read.csv(args[3], header=FALSE)        #matched soil and yield file
argument
field_1_mapped_reduced <- field_1_mapped_reduced[sample(nrow(field_1_mapped_reduced)),]
#shuffle

EM_data <- read.csv(args[4], header=FALSE)     #yield data file argument
EM_data <- EM_data[1:( length(EM_data[,1]) ), ]
lemd <- length(EM_data[1,])   #number of cols
EM_mat_X <- data.matrix(EM_data[,3:(lemd-1)])    #yield data is last column
EM_mat_Y <- data.matrix(EM_data[,lemd])  #just a dummy; not the real test value...

#column 11 is the yield data

lf1mr <- length(field_1_mapped_reduced[1,])
X <- field_1_mapped_reduced[,3:(lemd-1)]    #use same value lemd because soil data is added on
end
X <- data.matrix(X, rownames.force=NA)

EM_mat_X[1,]
#X[1:10,]

tot_col_num <- length(field_1_mapped_reduced[1,])

Y_list <- list()
for (iter2 in (tot_col_num-11):(tot_col_num-1)){
#create vector for Y output
Y_list[[iter2]] <- data.matrix(field_1_mapped_reduced[,iter2])
}

library(devtools)
library(iRF)
```

```
sel_list <- list()
pred_list <- list()
test_error_list <- list()
for (iter1 in (tot_col_num-11):(tot_col_num-1)){
#do the following for each soil variable
Y <- Y_list[[iter1]]
n <- length(Y)
p <- length(X[1,])

train.id <- 1:(n*0.7)
test.id <- setdiff(1:n, train.id)  #train on this fraction of the data

rf <- list()
sel.prob <- rep(1/p, p)
rf_maxiter <- 10
for (iter in 1:rf_maxiter){
  cat(paste('iter = ', iter, ':: '))
  if (iter < rf_maxiter){
  rf[[iter]] <- randomForest(x=X[train.id,], y=Y[train.id],
                             xtest=X[test.id,], ytest=Y[test.id],
                             mtry.select.prob=sel.prob, ntree=500)
  }
  if (iter == rf_maxiter){
  #in last iteration, test with all data for field-scale imputation
  rf[[iter]] <- randomForest(x=X[train.id,], y=Y[train.id],
                             xtest=EM_mat_X, ytest=EM_mat_Y,
                             mtry.select.prob=sel.prob, ntree=500)
  }

  # performance on test set
  if (iter < rf_maxiter){
  test.error = mean((rf[[iter]]$test$predicted - Y[test.id]) ^ 2) / var(Y[test.id])
  cat(paste('test error: ', round(100*test.error, 2), '%\n', sep=''))
  sel.prob <- rf[[iter]]$importance/sum(rf[[iter]]$importance)
  }
}  #iter loop
test_error_list[[iter1]] <- test.error    #last testing error
sel_list[[iter1]] <- sel.prob
#pred_list[[iter1]] <- rf[[iter]]$test$predicted     #predicted soil data value
writeframe <- cbind(EM_data[,1:2], data.frame(rf[[iter]]$test$predicted))
write.csv(writeframe, file=sprintf("EM_soilpred_%d.csv", iter1))    #field-scale soil prediction is
written

}  #iter1 loop

###Now we write out the selection probabilities and testing errors for each variable
```

```
sel_list_outframe <- data.frame(sel_list[[(tot_col_num-11)]])
test_list_outframe <- data.frame(test_error_list[[(tot_col_num-11)]])
for (iter in (tot_col_num-10):(tot_col_num-1)){
   sel_list_outframe <- cbind(sel_list_outframe, data.frame(sel_list[[iter]]))
   test_list_outframe <- cbind(test_list_outframe, data.frame(test_error_list[[iter]]))
}

write.csv(data.frame(test_list_outframe),    file=sprintf('testing_errors_%s_%s.csv'       ,    args[4],
args[5] ))
write.csv(data.frame(sel_list_outframe),             file=sprintf('selection_probabilities_%s_%s.csv',
args[4], args[5]))
```

2.5.3 Code for Local Importance Regression

```
args = commandArgs()

library(httr)
httr::set_config(config(ssl_verifypeer = 0L))

infile <- read.csv(args[3], header=TRUE)
infile <- infile[sample(nrow(infile)), ]   #shuffle
infile_copy <- infile[,]   #copy for later

infile <- infile[1:(nrow(infile)*as.numeric(args[5])) , ]  #subfraction
column_names <- colnames(infile)

keep_cols                                                                                              <-
c('Slope','Lime','Potash','TSP','Conductivity','Phosphorus','Potassium','pH','Zinc','Sulfur','Boron','
Magnesium','Manganese','Copper','CEC','OrganicMatter')   #features for regression
yield_name <- c('Yield2018')


X <- data.matrix(infile[, keep_cols])
Y <- data.matrix(infile[, yield_name])

X_new <- data.matrix(infile_copy[, keep_cols])
Y_new <- data.matrix(infile_copy[, yield_name])

library(iRF)    #libraries needed to run iRF
library(AUC)
attach(mtcars)
library(ggplot2)

n.cores <- 4
n <- length(Y)
p <- length(X[1,])

train_fraction <- 0.7
itermax <- 10             #number of random forest iterations

train_rows <- 1:(train_fraction*nrow(infile))
test_rows  <- setdiff(1:nrow( infile ), train_rows)
train.id <- train_rows
test.id <-  test_rows

rf <- list()
sel.prob <- rep(1/p, p)   #begin with equally weighted features
```

```
Y_test <- infile[test.id, yield_name]
Y_train <- infile[train.id, yield_name]

###In the next section, we run random forest
for (iter in 1:itermax){
  cat(paste('iter = ', iter, ':: '))
  if (iter < itermax){
  rf[[iter]] <- randomForest(x=X[train.id,], y=Y_train,
                             xtest=X[test.id,], ytest=Y_test,
                             mtry.select.prob=sel.prob, ntree=500)
  }
  if (iter == itermax){
    rf[[iter]] <- randomForest(x=X[train.id,], y=Y_train,
                             xtest=X_new, ytest=Y_new,
                             mtry.select.prob=sel.prob, ntree=500, localImp=TRUE)
  }
  # update selection probabilities for next iteration

  if (iter < itermax){
  sel.prob <- rf[[iter]]$importance/sum(rf[[iter]]$importance)   #update selection probability

  # performance on test set
  test.error = mean((rf[[iter]]$test$predicted - Y[test.id]) ^ 2) / var(Y[test.id])
  print(test.error)
  #UE dissimilarity matrix
  }
}
###Now we write the local importances, observation locations, and global importances
write.csv(data.frame(rf[[itermax]]$localImp), file="local_importance_2018.csv")
write.csv(infile_copy[,1:2], file="local_locations_2018.csv")
write.csv(data.frame(sel.prob), file="global_importance_2018.csv")
```

2.5.4 Code for Response Surface Generation.

```
args = commandArgs()
'%ni%' <- Negate('%in%')

library(magrittr)
library(httr)
library(dplyr)
library(data.table)
httr::set_config(config(ssl_verifypeer = 0L))  # only need this if you're getting ssl errors from R
set.seed(58)        #for consistency

all_data <- read.csv(args[3], header=TRUE)
all_data <- all_data[sample(nrow(all_data)),]  #shuffle
all_data <- all_data[1:(nrow(all_data)*as.numeric(args[5])),]   #sub fraction of observations

ncols <- length(all_data[1, ])

yield_col_id <- c('Yield')    #this is the target column for regression
library(ggplot2)
library(iRF)
source('surfacePlot.R')        #from iRF package; must be in same directory

all_data_backup <- all_data

holdout_limits <- c(621880, 621920)  #longitudinal limits of test data
buffer_size <- 20    #buffer around strip to exclude from training data
holdout_limits2 <- c(holdout_limits[1]-buffer_size, holdout_limits[2]+buffer_size)

infile <- all_data
tr1 <- rownames(infile)[(infile[,1] > holdout_limits[1])]
all_temp <- data.frame(infile[tr1,])
test_rows <- rownames(all_temp)[(all_temp[,1] < holdout_limits[2])]

tr2 <- rownames(infile)[(infile[,1] < holdout_limits2[1])]
tr3 <- rownames(infile)[(infile[,1] > holdout_limits2[2])]
train_rows <- c(tr2, tr3)

#set up training matrix and testing matrix
train_mat  <- all_data[train_rows,]
test_mat   <- all_data[test_rows,]
all_data   <- rbind(train_mat, test_mat)
colnames(all_data) <- colnames(all_data_backup)
all_data[1,]
```

```
keep_columns <-
c('Slope','Lime','Potash','TSP','Conductivity','Phosphorus','Potassium','pH','Zinc','Sulfur','Boron','
Magnesium','Manganese','Copper','CEC','OrganicMatter')
X_train <- all_data[train_rows, keep_columns]
X_train <- data.matrix(X_train)

X_test  <- all_data[test_rows, keep_columns]
X_test  <- data.matrix(X_test)
Y_train <- data.matrix(all_data[train_rows, yield_col_id])
Y_test  <- data.matrix(all_data[test_rows , yield_col_id])

num_train <- length(X_train[,1])      #number of training points

x <- data.matrix(rbind(X_train, X_test))
y <- data.matrix(rbind(Y_train, Y_test))
y <- as.numeric(y)

varnames <- colnames(x)

n <- nrow(x)
p <- ncol(x)
train.id <- 1:num_train   #since we re-organized all_data
total_iterations <- 10

t1 <- Sys.time()
f <- iRF(x=x[train.id,], y=y[train.id], n.iter=total_iterations,
interactions.return=total_iterations,n.bootstrap=1, n.core=1, get.prevalence=TRUE,
int.sign=TRUE)
options(tibble.print_max=Inf)

f

write.csv(data.frame(f$prevalence[[total_iterations]]), file= sprintf("Prevalences_iRF_%s.csv",
args[5]))

###take interactions of order 2 and put them into a vector
interact_dict <- data.frame(f$prevalence[[total_iterations]])$int
interact_dict <- as.character(interact_dict)
length_dict  <- lengths(strsplit(interact_dict, '_'))  #breaks interaction names
keep_indices <- which(length_dict %in% c(2))    #only keeps those with length 2
interact_dict <- interact_dict[keep_indices]


# read RF paths and track selected thresholds
t2 <- Sys.time()
```

```
rd.forest  <-  readForest(f$rf.list[[total_iterations]],  x=x[train.id,],  n.core=16,  get.split=TRUE,
varnames.grp=varnames)

interact_dict <- sample(interact_dict)   #shuffle

for (interact_iter in 1:length(interact_dict)){
interact <- interact_dict[interact_iter]
print(interact)
int.id <- int2Id(interact, varnames, directed=TRUE)
print(int.id)

grid <- quantileGrid(x, 100, int.id)


xrange <- c(grid$g1[1], grid$g1[length(grid$g1)])
yrange <- c(grid$g2[1], grid$g2[length(grid$g2)])
rectangles <- forestHyperrectangle(rd.forest$tree.info, rd.forest$node.feature,
                              x=x[-train.id,], y=y[-train.id] ,
                              interact=interact, varnames.grp=varnames) #node.obs=rd.forest$node.obs)

library(rgl)

print.code<- paste(interact, args[4], sep="_")
plotInt2(rectangles, interact, x=x, y=y,
       varnames.grp=varnames, grids=grid, pred.prob=FALSE,
       xlab=interact[1], ylab=interact[2], zlab='Yield (bushel/acre)', print.code=print.code)

}
```

2.5.5 Code for ROSM Generation.

```
import os
import numpy as np
import scipy
import sys
from sklearn.decomposition import PCA
from sklearn.linear_model import Ridge
from sklearn import linear_model
from sklearn.linear_model import Lasso
import statsmodels.api as sm
import pandas
import math as m
import pandas as pd
import random

from scipy.interpolate import griddata
from numpy import ma
import copy

want_minmax =  True   #converts percentile to minmax format for gradient climbing
want_yieldpenalty = False  #uses arctan penalty against the desired yield min; balances ROI - yield
tradeoff
want_uncertainty = False

want_covariance = False     #use covariance mat
downsampling_factor = 101   #reduce ALL mats by this factor to work with cov. array dimensions

if want_yieldpenalty:
   yield_setpoint = 71.2  #in bushels/acre
   yield_charscale = 0.0001   #characteristic scale of arctan change; should be < 1 bushel/acre

def get_covariances(percentile_mat, column_titles, cov_array, interact_list, beta_hat,
cov_downsample=1):
#covariance downsample is essentially fraction of points in list to look at to reduce size
#need to evaluate covariance based on the surfaces
   beta_hat_noconst = beta_hat.tolist()[1:]
   beta_mat = np.array([[beta_hat_noconst[x] * beta_hat_noconst[y] for x in
range(len(beta_hat_noconst))] for y in range(len(beta_hat_noconst))])
   #this is the scaling between the surface covariance and model covariance
   num_downsampled_points = int(percentile_mat.shape[0]/cov_downsample)
   print(num_downsampled_points)


   length_limit = cov_array.shape[-1]   #this is the length of the matrix
```

```
    outmat = [[0 for x in range(num_downsampled_points)] for y in
range(num_downsampled_points)]
    for p1_index in range(num_downsampled_points):
        print(p1_index)
        true_p1_index = p1_index * cov_downsample
        for p2_index in range(p1_index, num_downsampled_points):
            true_p2_index = p2_index * cov_downsample
            covariance_grid = np.zeros(np.array(beta_mat).shape)   #store covariance values between
surfaces
            for interact_index_1 in range(len(interact_list)):
                interact_1_name = interact_list[interact_index_1]
                interact_1_split = ''.join(''.join(interact_1_name.split('-')).split('+'))  #cuts out signs
                interact_1_split = interact_1_split.split('_')   #
                interact_1_cols = [column_titles.index(entry) for entry in interact_1_split]  #indices of
cols
                interact_1_indices = [  min(int(percentile_mat[true_p1_index][entry]*length_limit),
length_limit-1)   for entry in interact_1_cols]  #indices to use on surface-surface covmati
                for interact_index_2 in range(interact_index_1, len(interact_list)):
                    interact_2_name = interact_list[interact_index_2]   #names of interactions
                    interact_2_split = ''.join(''.join(interact_2_name.split('-')).split('+'))
                    interact_2_split = interact_2_split.split('_')   #
                    interact_2_cols = [column_titles.index(entry) for entry in interact_2_split]
                    interact_2_indices = [ min(int(percentile_mat[true_p2_index][entry]*length_limit),
length_limit-1)   for entry in interact_2_cols]
                    covariance_grid[interact_index_1][interact_index_2] =
cov_array[interact_index_1][interact_index_2][interact_1_indices[0]][interact_1_indices[1]][inte
ract_2_indices[0]][interact_2_indices[1]]
                    covariance_grid[interact_index_2][interact_index_1] =
covariance_grid[interact_index_1][interact_index_2]   #repeat by symmetry...
            total_covariance = np.sum(np.array(covariance_grid) * np.array(beta_mat))  #by linearity
            outmat[p1_index][p2_index] = total_covariance
            outmat[p2_index][p1_index] = total_covariance
    return outmat

def shrink(data, shrink_array):
#shrinks each dimension according to array
    data = np.array(data)
    shrink_nparray = np.array(shrink_array)
    datashape = np.array(list(data.shape))    #need to subtract
    overages = np.mod(datashape, shrink_nparray)   #amount of extra entries in each dim.
    limits = (datashape - overages).tolist()
    for dimindex in range(len(data.shape)):
        data = np.take(data, [x for x in range(limits[dimindex])] , dimindex)   #truncates on each dim.

    rows, cols = shrink_array[0], shrink_array[1]
    return data.reshape(rows, int(data.shape[0]/rows), cols,
int(data.shape[1]/cols)).sum(axis=1).sum(axis=2)

def reshape(array, interact_list, sample_mat):
#reshapes array to be len(interact_list)^2 x length^4 for covariance tensor
#unwrap from 2D back to 6D; this will be MUCH easier to work with in numpy later on
    num_interactions = len(interact_list)
    side_length = np.array(sample_mat).shape[0]    #assumes same in each dim.
    newmat = np.zeros((num_interactions, num_interactions, side_length, side_length, side_length,
side_length))

    s_l_2 = side_length ** 2   #this will get used a few times

    for x in range(num_interactions):
        print('On interaction {0} out of {1}'.format(x+1, num_interactions))
        first_index_x = x * s_l_2    #first index of matrix
        for y in range(num_interactions):
            surface_surface_array = np.zeros((side_length, side_length, side_length, side_length))
            first_index_y = y * s_l_2   #use this + s_l**2 to get cutoffs
            submat = array[first_index_x:(first_index_x+s_l_2), first_index_y:(first_index_y+s_l_2)]
#this is the (still unwrapped) surface-surface interaction
            for x2 in range(side_length):
                x2sl = x2 * side_length        #need this for coordinate
                for y2 in range(side_length):
                #Take subpart of matrix, then upwrap AGAIN
                    surface_2_vec = submat[x2sl + y2][:]    #now we unwrap down to 2d
                    surface_2_vec = np.reshape(surface_2_vec, surface_2_vec.size)   #make sure this is
row
                    inner_mat = np.zeros((side_length, side_length))
                    for inner_index in range(surface_2_vec.size):   #get innermost mat
                        inner_mat[int(inner_index/side_length)][inner_index % side_length] =
surface_2_vec[inner_index]   #innermost index
                    newmat[x][y][x2][y2][:][:] = copy.deepcopy(inner_mat)

    print(newmat.shape)
    return newmat


def dumpout(array, filename='zzzDumped_array.csv'):
#creates textfile from a 2D array
    listarray = array.tolist()
    listarray = [[str(subentry) for subentry in entry] for entry in listarray]
    listarray = [','.join(entry) + '\n' for entry in listarray]
    outfile = open(filename, 'w')
    for entry in listarray:
        outfile.write(entry)
    outfile.close()

def array_print(nparray, filename):
#similar to function above; creates textfile from np array
    outfile = open(filename, 'w')
    outarray = nparray.tolist()
    for x in range(len(outarray)):
        entry = outarray[x]   #should ALSO be a list...
        entry = [str(subentry) for subentry in entry]
        outfile.write(','.join(entry)  + '\n')
    outfile.close()

def sort_fb_beta(val_array, new_name_array, old_name_array):
#sort the values between the old and new name arrays
    out_val_array = []
    for x in range(len(old_name_array)):
        old_name = old_name_array[x]
        try:
            new_index = new_name_array.index(old_name)
            value = val_array[new_index]
            out_val_array.append(value)
        except:
            out_val_array.append(0)
    return np.array(out_val_array)

def stepwise_selection(X, y,
                       initial_list=[],
                       threshold_in=0.01,
                       threshold_out = 0.05,
                       verbose=True):
    """ Perform a forward-backward feature selection
    based on p-value from statsmodels.api.OLS
    Arguments:
        X - pandas.DataFrame with candidate features
        y - list-like with the target
        initial_list - list of features to start with (column names of X)
        threshold_in - include a feature if its p-value < threshold_in
        threshold_out - exclude a feature if its p-value > threshold_out
        verbose - whether to print the sequence of inclusions and exclusions
    Returns: list of selected features
    Always set threshold_in < threshold_out to avoid infinite looping.
    See https://en.wikipedia.org/wiki/Stepwise_regression for the details
    """
    included = list(initial_list)
    while True:
        changed=False
        # forward step
        excluded = list(set(X.columns)-set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included+[new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.argmin()
            included.append(best_feature)
            changed=True
            if verbose:
                print('Add  {:30} with p-value {:.6}'.format(best_feature, best_pval))

        # backward step
        model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        # use all coefs except intercept
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max() # null if pvalues is empty
        if worst_pval > threshold_out:
            changed=True
            worst_feature = pvalues.argmax()
            included.remove(worst_feature)
            if verbose:
                print('Drop {:30} with p-value {:.6}'.format(worst_feature, worst_pval))
        if not changed:
            break
    return included, model


def datprint(X, Y, colnames):
#creates a textfile from a matrix X and a vector Y
    outfile = open('zzzXARRAY.dat', 'w')
    Xlist = X.tolist()
    Ylist = Y.tolist()
    outfile.write('\t'.join(colnames) + '\t' + 'yield' + '\n')
    for x in range(len(X)):
        outstring = ''
        for y in range(len(X[0])):
            outstring = outstring + str(Xlist[x][y]) + '\t'
        outstring = outstring + str(Ylist[x]) + '\n'
        outfile.write(outstring)
    outfile.close()


def sort_print(valarray, namearray, outname):
    #sorts value array & name array, assuming they correspond, and then puts into textfile
    tuple_array = [tuple([abs(valarray[x]), namearray[x], valarray[x]]) for x in range(len(valarray))]
    tuple_array.sort()
    tuple_array = tuple_array[::-1]
    outfile = open(outname, 'w')
    for entry in tuple_array:
        outstring = str(entry[1]) + '\t' + str(entry[2]) + '\n'
        outfile.write(outstring)
    outfile.close()

def pad(data):
#interpolates over convex hull of data in a matrix despite the presence of nan... useful for spotty
data
    bad_indexes = np.isnan(data)
    good_indexes = np.logical_not(bad_indexes)
    good_data = data[good_indexes]
    interpolated = np.interp(bad_indexes.nonzero()[0], good_indexes.nonzero()[0], good_data)
    data[bad_indexes] = interpolated
    return data


def get_lines(string, sample_fraction=1.0):
#returns column titles and floated, split inlines from a csv
    infile = open(string, 'r')
    inlines = infile.readlines()
    infile.close()



    splitlines = [line[:-1].split('#')[-1] for line in inlines]   #for first line
    splitlines = [line.split(',') for line in splitlines]

    column_titles, splitlines = splitlines[0], splitlines[1:]   #separates
    splitlines = [[float(subentry) for subentry in entry] for entry in splitlines]
    random.shuffle(splitlines)
    splitlines = splitlines[:int(sample_fraction*len(splitlines))]
    return column_titles, splitlines   #returns these portions separately

def get_yield_vec(lines, column):
#just returns a vector of the yield values given a column ID
    return np.array([entry[column] for entry in lines])

def remove_columns(titles, lines, cols=False):
#columns to remove from the data after storing elsewhere
#should NOT be negative or otherwise outside the limits
    if not cols:
        return titles, lines   #just returns original input
    cols = [entry % len(titles) for entry in cols]   #modulo the length
    titles = [titles[x] for x in range(len(titles)) if not x in cols]
    lines = [[entry[x] for x in range(len(entry)) if not x in cols] for entry in lines]
    return titles, lines

def percentile_format(Xmat, Y):
#uses percentile format from R output for X; sorts Y on same basis
#can use MINMAX version instead of percentile version... linear interpolation between points

    xy_mat = np.zeros((Xmat.shape[0], Xmat.shape[1]+1))
    percentile_vec = np.array([x/(Xmat.shape[0]-1) for x in range(Xmat.shape[0])])
    for col in range(xy_mat.shape[1]-1 ):   #for each col except last
        xy_mat[:, col] = Xmat[:, col]
    xy_mat[:,-1] = Y[:]    #last col is Y
    for col in range(xy_mat.shape[1]-1 ):   #now we sort on these cols and then reassess vals
        xy_mat = xy_mat[xy_mat[:, col].argsort()]   #sorts on this column
        xy_mat[:, col] = percentile_vec[:]          #dumps these vals into the column...
    out_Y = xy_mat[:,-1]
    out_X = xy_mat[:, [dummyvar for dummyvar in range(xy_mat.shape[-1]-1)]]
    if not want_minmax:
        return out_Y, out_X

    xy_mat = np.zeros((Xmat.shape[0], Xmat.shape[1]+1))   #don't sort values
    for col in range(xy_mat.shape[1]-1):
        xy_mat[:, col] = Xmat[:, col]   #dummy values
    xy_mat[:, -1] = Y[:]    #last col is Y
    for col in range(xy_mat.shape[1]-1):
        minval = np.min(xy_mat[:, col])
        maxval = np.max(xy_mat[:, col])
        coldelta = maxval - minval
        xy_mat[:, col] = (xy_mat[:, col] - minval)/coldelta     #puts between zero and 1; can't be
constant
    out_Y = xy_mat[:, -1]
    out_X = xy_mat[:, [dummyvar for dummyvar in range(xy_mat.shape[-1]-1)]]
    return out_Y, out_X     #these are unshuffled


def get_interpolated_mat(string, substring='average'):
#opens the file from the string... right now version is for two features
#This function returns an interpolated matrix
    #downsmaple by global variable
    def cast_to_n_dims(array, newdims, charlen):
    #turns array of one dimension into MORE dimensions
        zeros_array = np.zeros(tuple(charlen for x in range(newdims))) #zero array of right size
        for index in range(len(array)):   #need to unwrap
            index_backup = index
            pos_tuple = []
            pos_tuple = tuple(int(index/charlen**(newdims-dim_index-1)) % charlen for dim_index
in range(newdims))
            zeros_array[pos_tuple] = array[index]   #add this entry here...   ###karl
        return zeros_array.tolist()


    ###Get floated, split lines
    print(string)
    inname = [f for f in os.listdir('.') if string in f and substring in f and ((string+'_' + substring) in f)
and not (('_' + string) in f)][0]
    infile = open(inname, 'r')
    inlines = infile.readlines()
    infile.close()
    splitlines = [line.split() for line in inlines]
    splitlines = [[float(subentry) for subentry in entry] for entry in splitlines]

    total_dims = len(string.split('_'))   #need special protocol for unwrapping
    char_len = len(splitlines)      #lenght of each dimension...
    splitlines = [cast_to_n_dims(entry, total_dims-1, char_len) for entry in splitlines]
    return_array = np.array(splitlines)
    ###DOWNsample the return array by downsampling_factor
    return_array = shrink(return_array, [downsampling_factor, downsampling_factor])
    return_array = np.array(return_array)

    print(return_array.shape)
    return return_array


def get_responses(submat, interact_mat):
#interpolates between responses of interaction given an array of values (submat)
    lower_ints = [[int(subentry) for subentry in entry] for entry in submat]
    upper_ints = [[int(np.ceil(subentry))  for subentry in entry] for entry in submat]
    li_np = np.array(lower_ints)
    ui_np = np.array(upper_ints)
    lambdas = (ui_np - np.array(submat))    #this is fraction of lower to take...
    om_lambdas = 1-lambdas                #one minus lambda... for interpolation

    super_int_mat = [[ [lower_ints[x][y], upper_ints[x][y]] for y in range(len(lower_ints[0]))] for x
in range(len(lower_ints))]     #for interpolation
    super_lambda_mat = [[ [lambdas[x][y], om_lambdas[x][y]] for y in range(len(lambdas[0]))] for
x in range(len(lambdas))]

###Find corners given the dimension of the array
    out_array = np.zeros(lambdas.shape[0])   #to be added to for sum
    for corner_index in range(2**lambdas.shape[-1]):   #num of cols...
```

```
        corner_tuple = tuple(int(corner_index/2**(lambdas.shape[-1]-1-dim_index) ) % 2 for
dim_index in range(lambdas.shape[-1])) #just binarizes
      # res_corner = np.array([interact_mat
        corner_indices = [[super_int_mat[x][y][corner_tuple[y]] for y in
range(len(super_int_mat[x]))] for x in range(len(submat))]
        corner_indices = [tuple(entry) for entry in corner_indices]
        res_corner = np.array([interact_mat[entry] for entry in corner_indices])
        lambda_list = [[super_lambda_mat[x][y][corner_tuple[y]] for y in
range(len(super_lambda_mat[x]))] for x in range(len(submat))]  #list of lambdas
        lambda_list = np.array([np.prod(np.array(entry)) for entry in lambda_list])  #product for
weighting
        weighted_response = res_corner * lambda_list
        out_array += weighted_response  #add weighed response using array...

   # print(out_array)

    return out_array  #linear interpolation

def get_row_subset(array, rows):
#index of rows to choose
    list_array = array.tolist()
    list_array = np.array([list_array[x] for x in range(len(list_array)) if x in rows])
    return list_array

def get_colvec(array):
#turns 1D numpy array in to true colvec
#This could be done in one line but cuts down on syntax later on
    listvec = array.tolist()
    return np.array([[entry] for entry in listvec])

def get_overall_derivative(beta, pd_name, varnames, interaction_mat_dict, X_percentile,
column_names_signed, uncertainty_switch=False):
    #beta is the set of global coefficients, pd_name is the name of the variable
    #vector is the vector of values where we are evaluating the derivative
    #this is one of the main functions for calculating the gradient
    #vector should be scaled from 0 to 1...
    pd_index = varnames.index(pd_name)  #index where we evaluate the derivative
    outsum = 0

    uncertainty_power = 1
    if uncertainty_switch:
        uncertainty_power = 2  #for form of variance calculation

###Collect the partial derivative matrices and multiply by the global coefficients
    for key in interaction_mat_dict.keys():
        if pd_name in key:     #then it will also be in the lower directories
      #print(key)
      deriv_mats = interaction_mat_dict[key][-1]
      if pd_name in deriv_mats.keys():
        deriv_mat = deriv_mats[pd_name]
        #now get the value of the derivative..
        interaction_name = key
        interaction_name_split = [entry[:-1] for entry in interaction_name.split('_')]
        colnames = interaction_name_split      #names of columns that belong in interaction
        col_indices = [varnames.index(entry) for entry in colnames]     #col indices for each
variable
        submatrix = X_percentile[:, col_indices]   #list form
        submatrix = submatrix * (interaction_mat.shape[0]-1)
        responses = get_responses(submatrix, deriv_mat)     #derivative mat...
        outsum += responses *
beta[column_names_signed.index(interaction_name)]**uncertainty_power  #multiplied by linear
factor for interaction
        #go over the interactions
  return outsum


def evaluate_response(interaction_lines, X_array, Y_vec, column_names_signed,
interaction_mat_dict, column_titles):
#record the response of the ROSM over a percentile matrix
  X_signed = np.ones((X_array.shape[0],1+len(interaction_lines) ))
  column_names_signed = ['CONSTANT'] + interaction_lines     #this will be the order we use...

  Y_vec, X_percentile =  Y_vec, X_array    #re-assigns names
  Y_average = np.mean(Y_vec)

  for x in range(1, X_signed.shape[1]):
    interaction_name = column_names_signed[x]
    interaction_mat = interaction_mat_dict[interaction_name][0]

    interaction_name_split = [entry[:-1] for entry in interaction_name.split('_')]  #take out sign
    colnames = interaction_name_split      #names of columns that belong in interaction
    col_indices = [column_titles.index(entry) for entry in colnames]    #col indices
    submatrix = X_percentile[:, col_indices]   #list form
    submatrix = submatrix * (interaction_mat.shape[0]-1)

    responses = get_responses(submatrix, interaction_mat)
    X_signed[:, x] = responses[:]   #adds the response...
  return X_signed


def get_minmax_version(interaction_mat, interaction_name, X_array, column_titles):
#returns a minmax version of the array instead of percentile-based version
```

```
  def get_percentile_from_minmax(minmax_val, X_array, col_index):
  #get percentile... of each entry based on the minmax value
    colvals = X_array
    true_val = colvals[0] + (colvals[-1] - colvals[0]) * minmax_val    #convert from 0 to 1 back
to ordinary units

    index_below = 0
    while colvals[index_below] < true_val:
      index_below = index_below + 1
    if colvals[index_below] == true_val:
      return index_below/(colvals.shape[0] - 1)    #goes between 0 and 1 (if exact)

    index_below = index_below - 1
    overage = true_val - colvals[index_below]
    delta = colvals[index_below+1] - colvals[index_below]
    if delta == 0:
      return index_below

    return  (index_below + overage/delta)/(colvals.shape[0] - 1)    #between 0 and 1...

  #remove signs from names
  interaction_name_unsigned = ''.join(interaction_name.split('+'))
  interaction_name_unsigned = ''.join(interaction_name_unsigned.split('-'))
  interaction_name_split = interaction_name_unsigned.split('_')     #split with no signs
  col_indices = [column_titles.index(entry) for entry in interaction_name_split]  #indices of cols

  #need to convert minmax limits to a percentile format first
  col_mins  = [np.min(X_array[entry]) for entry in col_indices]
  col_maxes = [np.max(X_array[entry]) for entry in col_indices]
  col_deltas = (np.array(col_maxes) - np.array(col_mins)).tolist()    #

  numdims = len(col_mins)   #number of dimensions
  dim_size = interaction_mat.shape[0]   #number of entries (observations)
  out_mat = np.zeros(interaction_mat.shape)    #copy but with zeros

  #sort the X_array columns in the copy X_array_sorted2
  X_array_sorted = [X_array[:, col_indices[x]].tolist() for x in range(numdims)]
  X_array_sorted2 = []
  for entry in X_array_sorted:
    entry2 = copy.deepcopy(entry)
    entry2.sort()
    X_array_sorted2.append(np.array(entry2))
  X_array_sorted = X_array_sorted2
```

```
   for num_index in range(out_mat.size):    #total entries in the numpy mat...
      minmax_tuple = []
      num_index_copy = num_index    #backup
      for dim_index in range(numdims):    #go over the dims
         new_num = (num_index_copy % (dim_size-1))/(dim_size-1)   #between 0 and 1
         num_index_copy = num_index_copy - (num_index_copy % (dim_size-1))
         num_index_copy = num_index_copy/dim_size   #(dimension size)
         minmax_tuple.append(new_num)
      minmax_tuple = tuple(minmax_tuple)    #tuple converting to minmax format; has proper
dimensions for array shape
      percentile_format            =            tuple([get_percentile_from_minmax(minmax_tuple[x],
X_array_sorted[x], col_indices[x]) for x in range(len(minmax_tuple)) ])

###evaluate the percentile formatted values
      percentile_format = list(percentile_format)
      submat = [[entry*(dim_size-1) for entry in percentile_format]]
      matrix_response = get_responses(submat, interaction_mat)  #evaluate using the old format
      minmax_list = [int(entry*(dim_size)) for entry in minmax_tuple]
      out_mat[minmax_list] = matrix_response
   print(out_mat)
   return out_mat

def get_unique_sign_names(inlist):
#returns a list w/ unique elements ONLY, ignoring signs in names
   unsigned_list = [ ( ''.join( (''.join(entry.split('-'))).split('+')) , entry) for entry in inlist]
   unsigned_list.sort()
#   print(unsigned_list)
   outlist = [inlist[0]]
   for x in range(1, len(inlist)):
      if not unsigned_list[x][0] == unsigned_list[x-1][0]:
         outlist.append(unsigned_list[x][1])    #original entry
   return outlist



#####################################################FUNCTIONS###############
###############
holdout_limits = [621880, 621920]   #lower and upper longitude values for strip trial

remove_cols = [3,4,5,6,7,8, 25]   #removes lat, lon, and UAV columns from textfile
target_col = 13  #BEFORE removing remove_cols; this is the column id of the yield
sample_fraction = 1.0   #fraction of rows to keep (global)
sample_fraction2 = 0.3  #fraction of observations to keep for training

inname = sys.argv[1]
column_titles, splitlines = get_lines(inname, sample_fraction)
```

```
train_rows = [x for x in range(len(splitlines)) if splitlines[x][0] < holdout_limits[0] or
splitlines[x][0] > holdout_limits[1]]
train_rows = random.sample(train_rows, int(sample_fraction2 * len(train_rows)))    #subsample
for speed
test_rows  = [x for x in range(len(splitlines)) if splitlines[x][0] >= holdout_limits[0] and
splitlines[x][0] <= holdout_limits[1]]
print(len(train_rows))
print(len(test_rows))


###get yield vector and testing lines for model fitting
Y_vec = get_yield_vec(splitlines, target_col)   #get yield vector
yield_mean = np.mean(Y_vec)
test_lines = get_row_subset(np.array(splitlines), test_rows)
test_positions = np.array([entry[:2] for entry in test_lines.tolist()])
print(test_positions[0])

column_titles, splitlines = remove_columns(column_titles, splitlines, remove_cols + [target_col])
X_array = np.array(splitlines)

###covariance must be pre-computed between surfaces
if want_covariance:
    cov_name = sys.argv[2]
    cov_file = open(cov_name, 'r')
    cov_lines = cov_file.readlines()
    cov_file.close()
    cov_array = [line[:-1].split(',') for line in cov_lines]
    cov_array = [[float(subentry) for subentry in entry] for entry in cov_array]
    cov_array = np.array(cov_array)    #covariance point-point array  #this needs to be reshaped



###This is all pre-processing; now we add MORE dimensions to the array for the linear nth order
models
####Model 1: ALL 2nd order interactions are fitted and evaluated...
num_vars = X_array.shape[-1]    #number of variables...
ones_array = np.ones((X_array.shape[0], 1))  #constant value for constant in "linear" model
second_order_array = np.ones((X_array.shape[0], int(num_vars*(num_vars+1)/2 ) )) #second
order var
second_order_titles = ['CONSTANT'] + column_titles
new_titles = []
col_counter = 0
for x in range(len(column_titles)):
    for y in range(x , len(column_titles)):
        new_title = column_titles[x] + '_' + column_titles[y]
        new_titles.append(new_title)
        second_order_array[:,col_counter] = X_array[:,x] * X_array[:,y]   #duplicates data in col
        col_counter += 1

second_order_titles = second_order_titles + new_titles
X2_array = np.concatenate((ones_array, X_array, second_order_array), axis=1)

  #training and testing rows
X2_train = get_row_subset(X2_array, train_rows)
Y_train = get_row_subset(Y_vec, train_rows)
X2_test = get_row_subset(X2_array, test_rows)
Y_test = get_row_subset(Y_vec, test_rows)

beta_hat = np.linalg.lstsq(X2_train, Y_train)[0]   #train on train rows...
predicted = np.dot(X2_test, beta_hat)
residuals = Y_test - predicted
ave_residual = np.mean(np.abs(residuals))
print(X2_array.shape)
print(ave_residual)
print(beta_hat)
residuals = get_colvec(residuals)
            #This is the second-order model.



####Model 2: similar to above, but ONLY use interactions that are IMPORTANT from RF
interaction_files = [f for f in os.listdir('.') if 'average.txt' in f]
interaction_files = [f for f in interaction_files if len(f.split('_')) == 3]
interaction_files.sort()
interaction_files = get_unique_sign_names(interaction_files)
interaction_files = interaction_files[:]
interaction_lines = ['_'.join(f.split('_')[:-1]) for f in interaction_files]
interaction_lines.sort()   #sorts alphabetically

###Get standard deviation surfaces
uncertainty_files =  [f for f in os.listdir('.') if 'deviations.txt' in f]   #undo this later
uncertainty_files = [f for f in uncertainty_files if len(f.split('_')) == 3]
uncertainty_files.sort()
#uncertainty_files = get_unique_sign_names(uncertainty_files)
uncertainty_files = uncertainty_files[:]                                #check for truncation
uncertainty_lines = ['_'.join(f.split('_')[:-1]) for f in uncertainty_files]
print(interaction_lines)
print(uncertainty_lines)



X_signed = np.ones((X_array.shape[0],1+len(interaction_lines) ))
```

```
column_names_signed = ['CONSTANT'] + interaction_lines     #this will be the order we use...


Y_vec, X_percentile = percentile_format(X_array, Y_vec)   #re-casts values as from 0 to 1; easiest
if Y goes along too...
Y_average = np.mean(Y_vec)

#X_signed[:,0] = yield_mean


###uncertainty and derivative matrix lookup dictionaries are set up here
interaction_mat_dict = {}   #{name:[interaction_mat, derivs], ...}; derivs == {name:object, ...}
uncertainty_mat_dict = {}   #these dictionaries store the derivatives of the interactions
X_signed_uncertainty = copy.deepcopy(X_signed)
for x in range(1, X_signed.shape[1]):
   interaction_name = column_names_signed[x]
   interaction_mat = get_interpolated_mat(interaction_name)     #gets matrix version of the
feature...

   print(np.array(interaction_mat).shape)   # test

   if want_minmax:
      interaction_mat = get_minmax_version(interaction_mat, interaction_name, X_array,
column_titles)
      uncertainty_mat = get_minmax_version(uncertainty_mat, interaction_name, X_array,
column_titles)
   interaction_name_split = [entry[:-1] for entry in interaction_name.split('_')]  #take out sign
   colnames = interaction_name_split      #names of columns that belong in interaction
   colnames = [''.join( ''.join(subentry.split('-')).split('+')) for subentry in colnames]
   print(colnames)
   print(column_titles)
   col_indices = [column_titles.index(entry) for entry in colnames]    #col indices
   submatrix = X_percentile[:, col_indices]   #list form
   submatrix = submatrix * (interaction_mat.shape[0]-1)

   #Now we will compute and organize the derivative matrices...
   interaction_mat_dict[interaction_name] = [interaction_mat, {}]

   gradient_mat_array = np.gradient(interaction_mat)
   gradient_mat_array = [np.array(entry) for entry in gradient_mat_array]  #list of 1D gradients
   print([entry.shape for entry in gradient_mat_array])

   for y in range(len(colnames)):
      colname = colnames[y]      #name and index
      interaction_mat_dict[interaction_name][-1][colname] = gradient_mat_array[y]    #set up the
structure first...
```

```
    responses = get_responses(submatrix, interaction_mat)
    X_signed[:, x] = responses[:]   #adds the response...

    #we must compute the gradient of the uncertainty for each response surface as well
    if want_uncertainty:
        uncertainty_mat = get_interpolated_mat(interaction_name, 'deviations')
        uncertainty_mat = uncertainty_mat**2    #squares the read-in plot
        if want_minmax:
            uncertainty_mat  =  get_minmax_version(uncertainty_mat,  interaction_name,  X_array,
column_titles)
        print(uncertainty_mat)
        uncertainty_mat_dict[interaction_name] = [uncertainty_mat, {}]
        gradient_var_array = np.gradient(uncertainty_mat)
        gradient_var_array = [np.array(entry) for entry in gradient_var_array]
        ###store the gradients in the dictionary uncertainty_mat_dict
        for y in range(len(colnames)):
            uncertainty_mat_dict[interaction_name][-1][colname] = gradient_var_array[y]
        var_responses = get_responses(submatrix, uncertainty_mat)
        X_signed_uncertainty[:, x] = var_responses[:]   #variance responses
        X_signed_uncertainty[:, 0] = 0      #constant column uncertainty set to 0


if want_covariance:
#now we reshape the array in this case...
    cov_array = reshape(cov_array, column_names_signed[1:], interaction_mat)  #cuts out constant


#use variance mats


X_signed_train = get_row_subset(X_signed, train_rows)
Y_train = get_row_subset(Y_vec, train_rows)
X_signed_test = get_row_subset(X_signed, test_rows)
Y_test = get_row_subset(Y_vec, test_rows)

###Compute the Linear ROSM
beta_hat = np.linalg.lstsq(X_signed_train, Y_train)[0]
signed_predicted = np.dot(X_signed_test, beta_hat)
signed_residuals = Y_test - signed_predicted
print(np.mean(np.abs(signed_residuals) ))
print(X_signed.shape)
print(beta_hat)
sort_print(beta_hat, column_names_signed, 'linear_ROSM.dat')
signed_residuals = get_colvec(signed_residuals)
array_print(np.concatenate((test_positions ,signed_residuals), axis=1), 'Linear_important.txt')
```

```
print('Above: linear regression over interactions')


###Ridge and Lasso regression for building ROSMs
yield_mean = np.mean(Y_vec)    #mean of the yield; put in place of constant for penalized regr.

alpha_value = 0.5

clf = Ridge(alpha=alpha_value)

clf.fit(X_signed_train, Y_train)
Y_pred = clf.predict(X_signed_test)
print(np.mean(np.abs(Y_pred-Y_test)))
print(clf.coef_)
sort_print(clf.coef_, column_names_signed, 'Ridge_ROSM_{0}.dat'.format(alpha_value))
array_print(np.concatenate((test_positions,      get_colvec(Y_test      -      Y_pred)),      axis=1),
'Ridge_residuals.txt')
print('Above: Ridge regression')


clf2 = Lasso(alpha=alpha_value)
clf2.fit(X_signed_train, Y_train)
Y_pred2 = clf2.predict(X_signed_test)
print(np.mean(np.abs(Y_pred2-Y_test)))
print(clf2.coef_)
sort_print(clf2.coef_, column_names_signed, 'Lasso_ROSM_{0}.dat'.format(alpha_value))
print('Above: Lasso regression')
array_print(np.concatenate((test_positions,      get_colvec(Y_test      -      Y_pred2)),      axis=1)      ,
'Lasso_residuals.txt')


X_signed_pd = pandas.DataFrame(data=X_signed_train, columns=column_names_signed)
y_pd = pandas.DataFrame(data=Y_train, columns = ['yield'])
datprint(X_signed, Y_vec, column_names_signed)


result, stepwise_model = stepwise_selection(X_signed_pd, y_pd)
print(stepwise_model.params.tolist())
fb_beta_hat = sort_fb_beta(stepwise_model.params.tolist(), result, column_names_signed)
predicted = np.dot(X_signed_test, fb_beta_hat)
residual = Y_test - predicted
print('forward-backward residual:')
print(np.mean(np.abs(residual)))

print('resulting features:')
result.sort()
```

```
print(result)
print(len(result))

sort_print(stepwise_model.params.tolist(), result, 'ForwardBackward.txt')
array_print(np.concatenate((test_positions,                    get_colvec(residual)),                   axis=1),
'stepwise_residuals.txt')

###############################################################################
##################This next section is the gradient solver for maximizing ROI###########
beta_hat = fb_beta_hat
beta_hat_squared = (np.array(beta_hat)**2).tolist()
tolerance = 0.00000000001    #hill climbing tolerance...
integrating_factor = 1000    #another hyperparameter for the inner gradient loop; optimizing ROI
analogue
current_delta = tolerance * 10000
old_ROI = 0.0
portfolio_lambda = 0    #lambda for effective utility function
new_X_percentile = X_percentile.tolist()
new_X_percentile = [[min(subentry*1.0000001, 1.0) for subentry in entry] for entry in
new_X_percentile]
new_X_percentile = np.array(new_X_percentile)
dumpout(new_X_percentile, 'orig_values.csv')

unadjustables = [0, 1]    #columns to not change; alternatively only optimization columns will be
changed
old_model_eval = np.dot(X_signed, beta_hat)   #clf.predict(X_signed)    #this is f(v0)
print(np.mean(old_model_eval))
prices = [1/X_signed.shape[0] for entry in column_titles]  #this will have to be adjusted... price
per unit per area will be important. just use 1 as price "per unit" for now
#for entry in unadjustables:
#    prices[entry] = 1


###prices must be in standardized units, for example $/lb
###Use hard-coded column ids for now
prices[3] = 30/X_signed.shape[0]  * (np.max(X_array[3]) - np.min(X_array[3])) * 0.00001
prices[4] = 318/X_signed.shape[0] * (np.max(X_array[4]) - np.min(X_array[4])) * 0.00001
prices[5] = 425/X_signed.shape[0] * (np.max(X_array[5]) - np.min(X_array[5])) * 0.00001


ones_mat =  np.ones(new_X_percentile.shape)
zeros_mat = np.zeros(new_X_percentile.shape)
optimization_columns = [3,4,5]     #just these columns; adjust by 1 for the constant
print(column_titles)

##establish lower and upper bounds
```

```
lower_bounds = np.array([0 for x in range(ones_mat.shape[-1])])
upper_bounds = np.array([100000000 for x in range(ones_mat.shape[-1])])    #these must be
converted to 0 - 1 range format
if want_minmax:
   mins = np.min(X_array, axis=0)
   maxs = np.max(X_array, axis=0)
   deltas = maxs - mins
   lower_bounds = (lower_bounds - mins)/deltas
   upper_bounds = (upper_bounds - mins)/deltas    #set to min and max
new_lower_bounds = np.zeros(ones_mat.shape)
new_upper_bounds = np.zeros(ones_mat.shape)
for column_index in range(ones_mat.shape[-1]):  #set these new matrix col vals to old ones...
   new_lower_bounds[:, column_index] = lower_bounds[column_index]
   new_upper_bounds[:, column_index] = upper_bounds[column_index]


loop_counter = 0
while current_delta > tolerance and loop_counter < 30000:
   if want_covariance:
#we will use this to calculate field-scale uncertainty, and other features...
      all_covariances = get_covariances(new_X_percentile, column_titles, cov_array,
column_names_signed[1:], beta_hat, cov_downsample=50)
      all_covariances = np.array(all_covariances)
      all_covariances = all_covariances.tolist()
      all_covariances = [[str(subentry) for subentry in entry] for entry in all_covariances]
      all_covariances = [','.join(entry)+'\n' for entry in all_covariances]



   loop_counter += 1
   partial_derivs = [get_overall_derivative(beta_hat, entry, column_titles, interaction_mat_dict,
new_X_percentile, column_names_signed)
                  for entry in column_titles]   #derivative for each adjustable variable... might need to
change which columns to use

   #print(partial_derivs)

   response_vector = evaluate_response(interaction_lines, new_X_percentile, Y_vec,
column_names_signed, interaction_mat_dict, column_titles)
   new_model_eval = np.dot(response_vector, beta_hat)  #This is f(v+v0)

   if want_uncertainty:
      for x in range(1, len(column_names_signed)):
      ###get uncertainty at each location
         interaction_name = column_names_signed[x]
         uncertainty_mat = uncertainty_mat_dict[interaction_name][0]
```

```
            interaction_name_split = [entry[:-1] for entry in interaction_name.split('_')]  #
            colnames = interaction_name_split      #names of columns that belong in interaction
            col_indices = [column_titles.index(entry) for entry in colnames]    #col indices
            submatrix = new_X_percentile[:, col_indices]   #list form
            submatrix = submatrix * (uncertainty_mat.shape[0]-1)
            X_signed_uncertainty[:, x] =  get_responses(submatrix, uncertainty_mat)
        new_model_variance = np.dot(X_signed_uncertainty, beta_hat_squared)

    print('Current              yield:              '+str(np.mean(new_model_eval))+'              Orig.:
'+str(np.mean(old_model_eval)))
    model_delta = new_model_eval - old_model_eval  #this is mean f(v0+v) - f(v0)
    price_change = np.sum(np.dot((new_X_percentile - X_percentile), np.array(prices)))    #total
price of new treatment
    gradient = [0 for x in range(len(partial_derivs))]
    for x in optimization_columns:   #these are the ONLY ones we change
        if not want_yieldpenalty or np.mean(new_model_eval) > yield_setpoint:
         #prevent model from getting "stuck" at very low changes in yield; yield_setpoint should be
slightly greater than original yield for proper performance
            gradient[x]              =              (partial_derivs[x]*price_change              -
model_delta*prices[x])/(price_change**2+0.0000001)
        else:
            gradient[x] = partial_derivs[x]*1/(abs(price_change)+0.00000001)    #just use the yield
itself as the target funciton in this case... make sure derivatives are scaled accordingly

    for x in range(len(gradient)):
        if not x in optimization_columns:
            gradient[x] = np.zeros(gradient[optimization_columns[0]].shape)    #not all columns will
be adjusted.
    gradient = np.array([entry.tolist() for entry in gradient])
    gradient = np.transpose(gradient)    #transposition


    if want_uncertainty:   #portfolio optimization; first compute uncertainty, then apply lambda
        partial_derivs        =        [get_overall_derivative(beta_hat,        entry,        column_titles,
uncertainty_mat_dict, new_X_percentile, column_names_signed, want_uncertainty) for entry in
column_titles]
        print(partial_derivs)
        gradient2 = [0 for x in range(len(partial_derivs))]
        for x in optimization_columns:
            gradient2[x]  = partial_derivs[x]
        for x in range(len(gradient2)):
            if not x in optimization_columns:
                gradient2[x] = np.zeros(gradient2[optimization_columns[0]].shape)
        for x in range(len(gradient2)):
            if np.sum(np.abs(np.array(gradient2[x]))) == 0:
                gradient2[x] = np.zeros(gradient2[optimization_columns[0]].shape)
        gradient2 = np.array([gradient2])     #see above
        gradient2 = np.transpose(gradient2)
        #print(gradient2)
        print('Gradients1 and 2: ' + str(np.sum(gradient)) + ' ' + str(np.sum(gradient2)))
        gradient2 = np.squeeze(gradient2)
        print('Portfolio lambda: ' + str(portfolio_lambda))
        gradient = gradient - portfolio_lambda * gradient2    #climbs combined optimization function


    current_delta = np.mean(np.abs(gradient))

    dumpout(new_X_percentile)

    #update the observations using the newly computed gradient
    new_X_percentile = new_X_percentile + integrating_factor*gradient
    new_X_percentile = np.minimum(new_X_percentile, ones_mat)
    new_X_percentile = np.maximum(new_X_percentile, zeros_mat)     #keep between 0 and 1
    new_X_percentile = np.maximum(new_X_percentile, X_percentile)  #positive prescription
    new_X_percentile = np.minimum(new_X_percentile, new_upper_bounds)
    new_X_percentile = np.maximum(new_X_percentile, new_lower_bounds)   #upper and lower
bounds
    ROI = np.mean(model_delta)/(price_change+0.00000000001)
    current_delta = abs(ROI - old_ROI)
    old_ROI = ROI

#    new_X_percentile = apply_physical_equations(new_X_percentile, X_percentile)

    print('ROI: ' + str(ROI))
    print('Avg. of gradient: ' + str(current_delta))
    print('Total price: ' + str(price_change/model_delta.shape[0]))
    if want_uncertainty:
        print('Average yield variance: {0} (bushel/acre)^2'.format(np.mean(new_model_variance)))
    print()
```

3 A Structural Coarse-Grained Model for Clays Using Simple Iterative Boltzmann Inversion[115]

Karl Schaettle, Luis Ruiz Pestana, Laura Nielsen Lammers, Teresa Head-Gordon

ABSTRACT

Cesium-137 is a major byproduct of nuclear energy generation and is environmentally threatening due to its long half-life and affinity for naturally occurring micaceous clays. Prolonged exposure to low levels of Cs-137 in the environment can increase the risk for certain cancers due to gamma radiation. Recent experimental observations of illite and phlogopite mica indicate that Cs^+ is capable of exchanging with K^+ bound in the anhydrous interlayers of layered silicates, forming sharp exchange fronts leading to interstratification of Cs- and K-illite. We present here a coarse-grain (CG) model of the anhydrous illite interlayer developed using iterative Boltzmann inversion (IBI) that qualitatively and quantitatively reproduces features of a previously proposed feedback mechanism of ion exchange.[116] The CG model represents a 70-fold speedup over all-atom (AA) models of clay systems and predicts interlayer expansion for K-illite near ion exchange fronts. Contrary to the longstanding theory that ion exchange in a neighboring layer increases the binding of K in lattice counterion sites leading to interstratification,[117] we find that the presence of neighboring exchanged layers leads to short-range structural relaxations that increase basal spacing and decrease cohesion of the neighboring K-illite layers. We also provide evidence that the formation of alternating Cs- and K-illite interlayers (i.e. ordered interstratification) is both thermodynamically and mechanically favorable compared to exchange in adjacent interlayers.

3.1 Introduction

The environmental impact of cesium adsorption and diffusion into various types of naturally occurring layered silicates has received renewed interest in recent years, especially in the aftermath of the Fukushima Daiichi Nuclear disaster.[118–120] One of the most environmentally threatening products of nuclear fission is Cesium-137, both because of its relatively long half-life (30.2 years) and its affinity for mineral surfaces, which prevents it from leaching from surface soils.[121,122] Cesium is strongly and irreversibly adsorbed to various clay surfaces in the presence of other ions, and can slowly diffuse into the bulk volume of both anhydrous and hydrated layered silicates.[123–125] Due to its intermediate half-life and its relative abundance as a nuclear decay product, Cesium-137 can contaminate environmental sites with dangerous levels of radiation for decades, while most other fallout isotopes may only present a threat on the order of several months or years.[126]

Cesium diffuses deep into naturally occurring clays and displaces other types of ions normally found in the interlayer, such as potassium, sodium, and calcium. Because of its ability to selectively exchange radiocesium, illite and similar clays have been investigated for the possibility of remediating radioactive plumes of cesium.[127] Despite extensive experimental study of ion adsorption at the frayed clay edge and exchange of ions in the clay interlayer,[124,125,128–130] the exact mechanism of cesium uptake remains elusive.[131] It is especially unclear how cesium displaces potassium within the interlayer far from the edge in anhydrous interlayers for clays such as illite,[128,132–134] despite very large barriers to ion diffusion. Specifically, these large energy barriers make explicit all-atom simulation of exchange difficult for bulk clays. Some groups have suggested that only hydrated ions within the interlayer or near a clay edge should be capable of exchange;[135] however, this exchange mechanism is very thermodynamically unfavorable in bulk illite, since both potassium and cesium strongly favor anhydrous interlayers.[136] Moreover, cesium is found to have a strong affinity for the weathered clay edge, further suggesting that interlayer hydration is likely not a necessary step either for cesium binding or penetration into the interlayer.[131,137–141]

Recent experimental[124,125] and computational work[116] supports the direct exchange by diffusion for large ions in the interlayer of several clay types without repeated hydration and dehydration at the edge sites. Instead, cesium has been hypothesized to first bind to frayed clay edges, and then more slowly exchange with the potassium naturally found in the interlayer through a simple diffusive mechanism.[134,142] Once potassium begins exchanging for cesium at the weathered edge, it has been proposed that there is a thermodynamic and kinetic driving force to displace additional potassium ions, resulting in an accelerated replacement of potassium ions for cesium.[143] Despite this proposed feed-forward mechanism, it has been observed that the rates of ion exchange over moderate timescales in neighboring interlayers can vary drastically.[124,125] Although some thermodynamic arguments have been proposed to explain the stability of interstratified clay particles in other clay types,[144–146] the physical reason for the disparity in neighboring interlayer exchange rates in illite is still not entirely clear.

In order to adequately predict the ability of micaceous minerals both to uptake radioactive cesium and its susceptibility to remobilization, it is necessary to understand and model the mechanisms that drive the adsorption and exchange of various ionic species in, for example, anhydrous illite. Recent computational work from our group using the classical ClayFF force field and density functional theory (DFT) has indicated that the presence of cesium ions within potassium interlayers creates mechanical forces that significantly increase the interlayer spacing.[116] This expansion in turn results in lower rate coefficients for ion exchange by 6 to 10 orders of magnitude which accelerates the diffusion of potassium ions.[116] However, the problem of characterizing interstratified particles by definition requires the evaluation of diffusion barriers involving multiple interlayers. Probing the energy landscape of large atomistic simulations of this type is computationally expensive. Therefore it motivates the development of a coarse-grain (CG) model that is capable of reproducing the energy barriers to ion diffusion near an exchange front and in interstratified clay particles.

Previous CG simulations of clay include both continuous and discrete models of the clay interlayer.[147–150] In one study, Marry et al.[147] investigated a montmorillonite system with a hydrated interlayer. Their CG model consisted of two uniformly charged plates representing the clay layers held a fixed distance apart. Water was modeled implicitly, and both cations and anions in solution were modeled continuously as an effective density that was allowed to vary over space. This model accounted for the excluded volume of the ions using the mean spherical approximation, an advance on previous continuous models of hydrated clay systems. Using DFT, the grand potential was minimized to derive the density of ions as a function of distance from the clay surface for sodium ions. These types of CG models can be highly accurate for the calculation of continuous properties such as tensile strength or average interlayer spacing. Several groups have also successfully modelled clay and clay-polymer systems by coarse-graining entire clay layers into one single particle.[151–153] However, continuous, ultra coarse-grained models are fundamentally incapable of capturing site-specific effects such as ion binding, and due to mean-field approximations would not be expected to precisely reproduce the observed variation in the ion binding energy landscape in anhydrous clay.[116]

Other groups have developed more detailed coarse-grain clay models that map multiple neighboring atoms to single pseudo-atoms, a common coarse-graining technique.[154] In the model of clay-polymer composites developed by the Coveney group, each montmorillonite clay layer is represented by bonded pseudo-atoms corresponding to ion binding sites.[155,156] The pseudo-atoms in the Suter et al. model are bonded with harmonic potentials derived from an iterative Boltzmann inversion (IBI) of the corresponding all-atom RDFs, and represents the three sheets of the clay layer as a single layer of coarse-grain sites.[28] Many of the coarse-grain potentials of ion-sheet interactions in this model were derived using PMF[157] matching with fixed sheets, and the various coarse-grain potentials were converged sequentially.[28] When deriving the potential between free ions and the bulk sites on the clay sheet, Suter and co-workers used the layer-averaged z-coordinate. This method faithfully reproduced the behavior of ions in the fluidized system, but would likely not be sufficiently accurate for reproducing the behavior of confined ions, which is strongly influenced by charge localization.

Simulations with both DFT and ClayFF indicate that ions in the anhydrous illite interlayer approach interlayer-facing oxygen atoms extremely closely, suggesting that compensatory compression of the neighboring sheets could play an important role in determining the energy landscape.[116] In addition, potassium and cesium ions at equilibrium within the interlayer are tightly confined within ditrigonal coordination cavities with oxygen, with interatomic distances of roughly 3.0 – 3.4 Å.[125,136] Due to this extreme confinement and the detailed structure of ion binding sites, representing an anhydrous clay layer by a single sheet of coarse-grain sites would likely eliminate important stiff stretching and bending degrees of freedom for determining the energetics of ion diffusion.

We present a coarse-grain model of an anhydrous illite clay system with different coarse grain types for representing both the tetrahedral and octahedral sheets. This model runs approximately 70 times faster than the all-atom implementation in ClayFF within the LAMMPS[158] simulation environment due to both reduced number of particles as well as permitting much larger timesteps of up to 10 fs for molecular dynamics simulations. While ClayFF is typically used to study only a few layers, our CG model can easily simulate dozens of relatively large clay layers in the same amount of simulation time. We show that studying large systems is necessary to completely eliminate finite size effects in the determination of converged diffusion barriers and provide evidence of a thermodynamic compensation for the interstratification of potassium and cesium ion distributions in anhydrous illite clay interlayers. Our CG model is capable of investigating relatively large systems (10^{-6} m) on simulation timescales of microseconds and the model is available to others through the LAMMPS simulation package.

3.2 Methods

3.2.1 Simulation Cell Setup

Classical molecular dynamics (MD) simulations of an atomistic 2-layer illite clay under periodic boundary conditions were used as a reference for generating a coarse-grain (CG) model using iterative Boltzmann inversion. The all-atom MD simulations were performed using the ClayFF[159] forcefield within the LAMMPS[158] simulation package. The ClayFF forcefield is a generalized, nonbonded model for hydrated clays, and consists mainly of Lennard-Jones and electrostatic interactions between atomic centers for bulk clay. It has been fitted to multiple multi-sheet, aluminosilicate clay types including kaolinite ($Al_2Si_2O_5(OH)_4$) and pyrophyllite ($AlSi_2O_5(OH)$), and has been shown to reproduce the swelling behavior of montmorillonite ($Na_3(Si_{31}Al)(Al_{14}Mg_2)O_{80}(OH)_{16}$) very accurately.[159] In addition, ClayFF has been used extensively to study the dynamics of ion adsorption in hydrated interlayers.[160–162] Because of its ability to model multi-layered clay systems under a variety of physical conditions, ClayFF was chosen as the reference atomistic forcefield for our coarse-grain model.

All-atom molecular dynamics simulations were run at 300 Kelvin for 100 picoseconds with a timestep of 1 fs after an initial equilibration period of 120 picoseconds. Coordinates of all atoms were sampled every 250 fs to build an ensemble for using the IBI algorithm. Each coarse-grain simulation was sampled in the NVT ensemble for 40 picoseconds after an equilibration of 100 picoseconds with a timestep of 3 fs to compromise between fast turnaround time and sufficient sampling for IBI. To ensure faithful reproduction of the all-atom data, the coarse-grained systems used during the IBI procedure were mapped directly from the corresponding all-atom systems.

3.2.2 CINEB Calculations

Climbing-Image Nudged Elastic Band (CINEB) calculations were performed using the “neb” command within LAMMPS[163–166] to obtain energy barriers for interlayer counterion (K^+ and Cs^+) migration, using 25 images integrated with a 5 femtosecond timestep. Before beginning each NEB calculation, both ion position and substitution sites were randomized. The first image for each NEB calculation was generated by equilibrating the randomized structure, and the final image had one ion from the first image displaced to an empty binding site. The remaining images were generated by linear interpolation of the ion position, so that the ion’s initial trajectory was a linear path between an occupied and unoccupied site. Oxygen atoms surrounding the initial and final ion binding sites were included in the reaction coordinate due to their displacement during ion diffusion. By including these atoms in the reaction coordinate, the distribution of energy barriers was reduced by around 30% without significantly altering the mean diffusion barrier value, indicating that this accurately captures important physical effects.

To determine system enthalpies, five-interlayer clay systems with different patterns of ion interstratification, periodic in the x- and y-directions with dimensions of 93 Å and 60 Å, respectively, were relaxed in the non-periodic z-direction over a period of 150 ps using a 3 fs timestep. The z-direction was non-periodic, and was initialized at 50 Å. The simulations were integrated using the multilevel summation method (MSM) real-space electrostatics.[167,168] This electrostatic integration method computes short-range interactions exactly and computes long-range interactions by decomposing the potential into a sum of smooth potentials which are integrated with a series of progressively coarser meshes. MSM has a competitive level of accuracy as PME for calculating long-range electrostatic interactions, and unlike PME can be used for non-periodic systems. Using the “shrink-wrap” feature in LAMMPS, the z-dimension of the simulation box was allowed to dynamically change over the course of each simulation until convergence. This procedure resulted in the system reaching the interlayer spacing that minimized the system enthalpy.

3.2.3 CG Model Parametrization

The iterative Boltzmann inversion (IBI) algorithm for coarse-graining attempts to reproduce all-atom pair correlation functions by constructing a pairwise interaction potential, a consequence of Henderson’s uniqueness theorem.[169,170] Although Henderson’s uniqueness theorem cannot be shown to rigorously hold for multiple pairwise interactions, in practice coarse graining strategies built on the methodology of reproducing the radial pair correlation function have been successful. CG simulations are run iteratively, and the new pair correlation function $g_{CG}(r)$ is used to update the old interaction potential. Assuming only pairwise effects, the all-atom pair correlation function $g_A(r)$ of gases can be roughly approximated:

$$g_A(r) \approx Ae^{-\beta u(r)} \tag{2}$$

where A is an arbitrary constant, β is the thermodynamic beta, and $u(r)$ is the potential as a function of the radial separation. The iterative Boltzmann inversion algorithm updates the coarse-grain potential based on the all-atom and CG pair correlation functions:

$$u_{CG,n+1}(r) = u_{CG,n}(r) + \frac{1}{\beta}\left(\ln\left(\frac{g_{CG}(r)}{g_A(r)}\right)\right) \tag{3}$$

Henderson’s uniqueness theorem is only precisely true for a homogenous fluid, but IBI is a robust method of coarse-graining that works for heterogeneous fluids and other phases as well. In particular, good performance of the IBI algorithm can be expected if run simultaneously for “orthogonal” degrees of freedom, such as the inter- and intra-sheet forces in our CG model of illite. The iterative Boltzmann inversion procedure was interfaced with LAMMPS by updating non-bonded and bonded coefficients as well as tabulated potentials after each iteration.

The CG model of anhydrous clay consists of 5 different coarse-grain centers. This mapping reduces the number of tracked centers in ClayFF by approximately 2:1 for the bulk clay. The five clay CG centers are as follows: 1 CG type for the octahedral sheet (Type Al) corresponding to the structural Al^{3+} cations; 2 CG types for the tetrahedral sheet, representing O^{2-} anions directly adjacent to the interlayer, but are separated into CG types corresponding to oxygen anions near sites with and without isomorphic substitutions of Al^{3+} for Si^{4+} (Type Os and O), so that their interactions with other CG centers are computed separately in order to capture differences in the binding site characteristics. In addition, there are two types of ions (Type K and Cs). Silicon and aluminum are not explicitly tracked in the tetrahedral sheet.

The CG sites experience five types of forces based on the following interaction potential: harmonic bonds and angles between neighboring centers in the octahedral and tetrahedral sheets, Lennard-Jones interactions between CG centers, electrostatics between ion and substituted-oxygen CG centers, and finally tabulated forces (Eq. (4)).

$$U = \sum_{bonds} k_b(r - r_0)^2 + \sum_{angles} k_a(\theta - \theta_0)^2 + \sum_{LJ} 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right) + \sum_i U_{tab,i.}(r) + U_{electrostatic} \tag{4}$$

The Lennard-Jones interactions are defined between each pair of CG centers and are not based on mixing rules; this approach was chosen because the CG centers represent different numbers of atoms from the all-atom model, and therefore the pairwise interactions are unlikely to be characterized by effective radii. Lennard-Jones interactions are slightly less computationally expensive than tabulated forces and are sufficient for the purposes of this study for characterizing the forces between sheets. Tabulated potentials are used between the ion and tetrahedral CG centers to more accurately capture the nature of the binding sites. Lennard-Jones interactions and electrostatics are excluded for 2nd, 3rd, and 4th neighbors based on the bond and angle topology. After each round of IBI, the updated CG potential is fitted to either a harmonic or Lennard-Jones function for most of the degrees of freedom, and this fit is used in the next round of MD simulation.

In some cases, the same parameters were used to characterize multiple interactions due to the corresponding all-atom model centers having very similar pair correlation functions. For example, one single Lennard-Jones potential was used to govern the nonbonded interaction between the octahedral and tetrahedral CG sites. This approach further simplifies the model, although it does not affect its computational cost. To run our CG clay model, all that is needed is a properly configured input script and data file with the clay system coordinates, charges, and topology. No modification to the LAMMPS software itself is needed to run the model, and the software used to run the IBI algorithm is entirely separate from the core codebase of any simulation software. The CG model is highly portable, and should be able to run on any other molecular simulation package that allows for the implementation of tabulated potentials.

3.3 Results

3.3.1 Clay CG Performance

Figure 3.1 and Supplementary Figure 3.1 show the distribution of bond distances and angles for the all-atom and CG models for the degrees of freedom fit with harmonic bonds, in which we observe overall excellent agreement. For the non-bonded degrees of freedom, Figure 3.2 and Supplementary Figure 3.2 present the radial distribution function (RDF) of the all-atom model in comparison to the most converged iteration for the CG model. Since the fit for the nonbonded degrees of freedom to the Lennard-Jones functions used only the first peak of each rdf, there is a relatively good agreement for all pair correlations with the exception of the O-O and O-Al CG types. However, even in these cases there is relatively good reproduction of the positions of secondary peaks in the RDFs, indicating that the geometry of the all-atom and CG systems are quite similar. The model is well converged based on the similarity of the oxygen-ion CG center RDFs to the corresponding distribution in the all-atom model, since these degrees of freedom are the most important for fully characterizing the ion binding site.

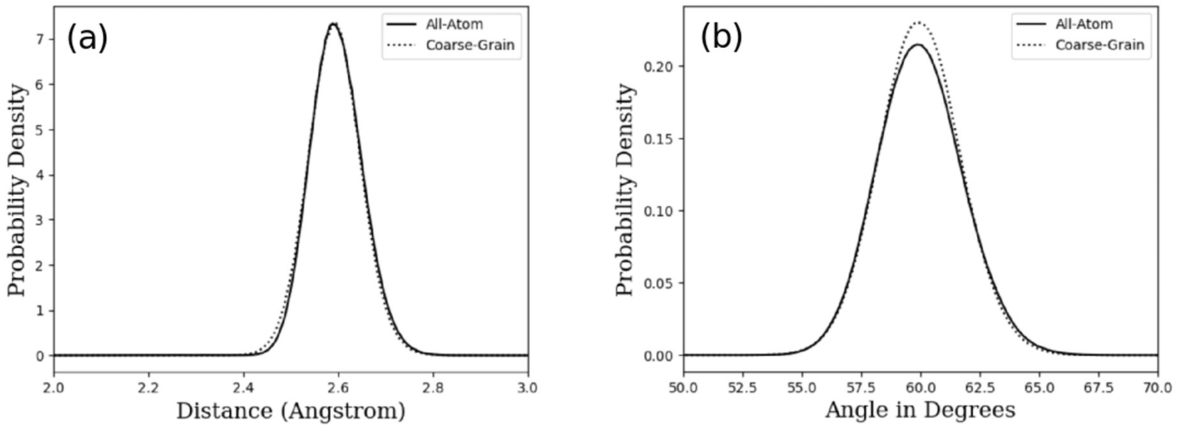


Figure 3.1. Comparison of the probability distribution of bond distances and bond angles for the all-atom (solid) and coarse-grained model (dotted). (a) Distribution of bond distances for the O-O CG bond; (b) Distribution of bond angles for the O-O-O 60 degree angle type bond.

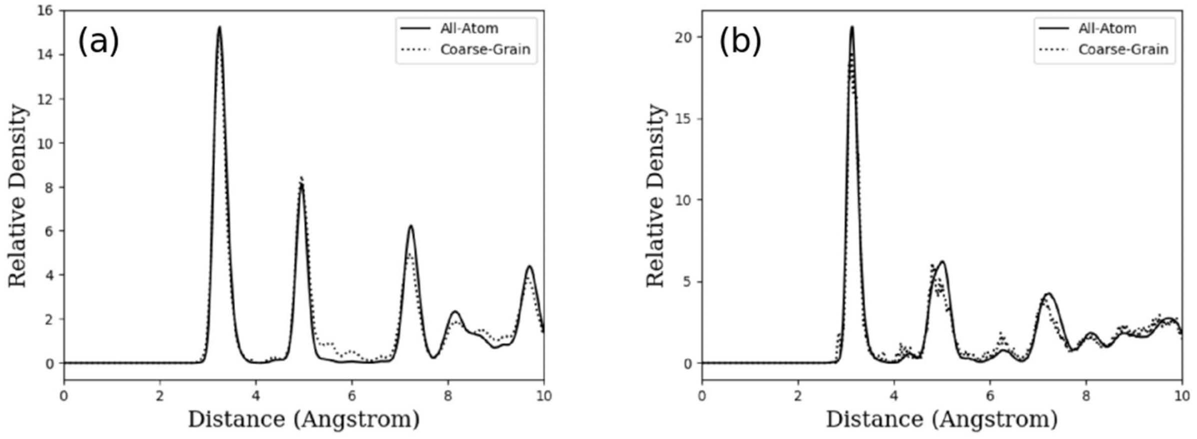


Figure 3.2. Comparison of the radial distribution function (RDF) for the all-atom (solid) and coarse-grained model (dotted). RDFs for (a) O-Cs CG types and (b) Os-Cs CG types. RDFs were sampled every 250 fs over a time period of 40 ps in equilibrated systems.

In order to quantify the increase in speed for our coarse-grain model, an all-atom model and its corresponding coarse-grain model were run for 3 nanoseconds on 32 cores, and the real time needed to simulate each 100 fs was recorded. The all-atom system consisted of 2 clay layers with periodic boundary conditions and a total of 9099 atoms and was simulated using the ClayFF force field. The coarse-grain model was simulated using the force field developed in this paper. The coarse-grain model ran roughly 6.9 times faster than the corresponding all-atom model with the same timestep. The shortest vibrational period in the all-atom model is on the order of 10 fs due to the explicit modelling of hydrogen, and in contrast the fastest vibrational mode in the coarse-grain model presented here is between bonded oxygen-oxygen centers, which is on the order of 100 fs.[171] Because of this, the coarse-grain model is stable with a timestep of up to 10 fs, while ClayFF must use an integration timestep on the order of 1 fs. Thus our coarse-grain model represents a roughly 70-fold speedup compared to the corresponding all-atom forcefield, which is comparable to the speedup obtained by other groups using similar coarse-graining techniques.[155,156]

As shown by Johnson et al., the Henderson uniqueness proof implies that there is always a representability problem as a general feature of a CG potential, i.e. a CG procedure cannot simultaneously resolve all the properties at a given state point.[172] For example, reproducing the energetics of a system when the coarse-graining approach is based on reproducing structural or geometric features of the more complex reference system is not formally guaranteed.[137,173] However, the structural coarse-graining approach is likely to reproduce qualitative trends in properties such as energetic barriers for ion diffusion in anhydrous clays, since these barriers will be primarily determined by mechanical forces.

3.3.2 Energy Barriers to Ion Migration in Pure Phases

We sought to further validate our CG model by performing NEB calculations to determine diffusion barriers for different ions in the presence of the same or different ions in the clay interlayers. While the energy barriers derived from the coarse-grain model were consistently higher and had a broader distribution than the corresponding barriers in the all-atom model, Figure 3.3 confirms that the CG model qualitatively reproduces the difference in diffusion energy barriers for K^+ and Cs^+ found previously in the all-atom clay model.[116]

The mean energy barrier for the migration of potassium in pure K-illite was found to be 300 ± 94 kJ/mol (Figure 3.3a) on average, compared to 226 ± 51 kJ/mol in the corresponding ClayFF model (Figure 3.3b). We also found for both K^+ and Cs^+, the energy barrier for diffusion was found to be much lower in systems with a higher fraction of Cs^+ atoms in the interlayer. This result is in agreement with the trend derived from the all-atom forcefield, which has led to our determination of a mechanism for interlayer exchange[116], whereby as more Cs^+ enters an interlayer, both ion species become much more mobile, effectively increasing the rate at which the exchange front will advance into the interlayer.[124,125,143]

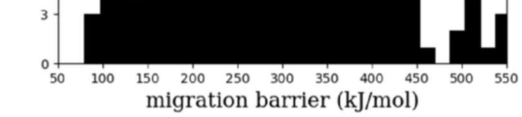


Figure 3.3. Energy barrier distribution for potassium ion diffusion in a 4-layer periodic clay system with 100% potassium interlayers. (a) CG model and (b) all atom Clay-FF model[116] (reproduced with permission). The distribution of energy barriers in the CG model was consistently found to be about 70 kJ/mol higher and 30% more broad than the corresponding all atom barriers. This effect is likely due to the inherent undersampling of high-energy paths during the IBI algorithm.

Both the CG and all-atom energy barriers correspond to timescales that are inaccessible for direct observation of diffusion events in MD simulations,[116] but the ability of the CG model to approximate the energy barriers and their trends with respect to all atom ClayFF is promising for using this coarse-grain model as a probe for changes in diffusion barriers to infer the kinetics of ion exchange. Properly modelling ion diffusion near and far from the exchange front necessarily requires a model capable of simulating a large, heterogeneous interlayer, as well as overcoming finite size effects by modeling many interlayers (i.e. greater than 2), which is extremely computationally expensive in all-atom ClayFF.

For sufficiently small periodic systems, the finite size effect can dramatically impact the compressibility, and would be expected to increase the calculated barrier to ion diffusion artificially.[174] Therefore, the magnitude of the NEB barriers was studied as a function of the number of simulated layers using periodic 2-interlayer, 4-interlayer, and 12-interlayer systems. Table 3.1 shows the average energy barrier for K^+ diffusion in K-illite as a function of the number of clay layers and the energy barrier distributions determined by NEB are presented in Supplementary Figure 3.3. Since there was essentially no observed change in the average energy barrier and variance between 4- and 12-interlayer systems, it was determined that simulating at least 4 interlayers would be sufficient to approximate an effectively infinite clay for the purposes of this study.

Table 3.1. Average Energy Barrier (and variance) for K^+ ion diffusion in periodic K-illite as a function of the number of interlayers. Very small systems greatly overestimate the barrier due to finite size effects. Both the CG model and the all-atom model in ClayFF feature very broad energy barrier distributions.

# of Interlayers	Average Energy Barrier and variance (kJ/mol)
2	332 +/- 131
4	300 +/- 94
12	298 +/- 95

3.3.3 Interlayer Energetics and Ion Migration Barriers in Interstratified and Homostructured Illite

Mixing of unlike ions in layered silicate interlayers can adopt different structures depending on ion distributions in the final structure. When interlayer ions form random mixtures the phase is homostructured, and when ions are separated into distinct phase-separated layers the phase is interstratified (Figure 3.4 and Figure 3.5a, respectively). In the following section we analyze the impact of structure on the barriers to ion migration, and consequently on the kinetics of ion exchange.

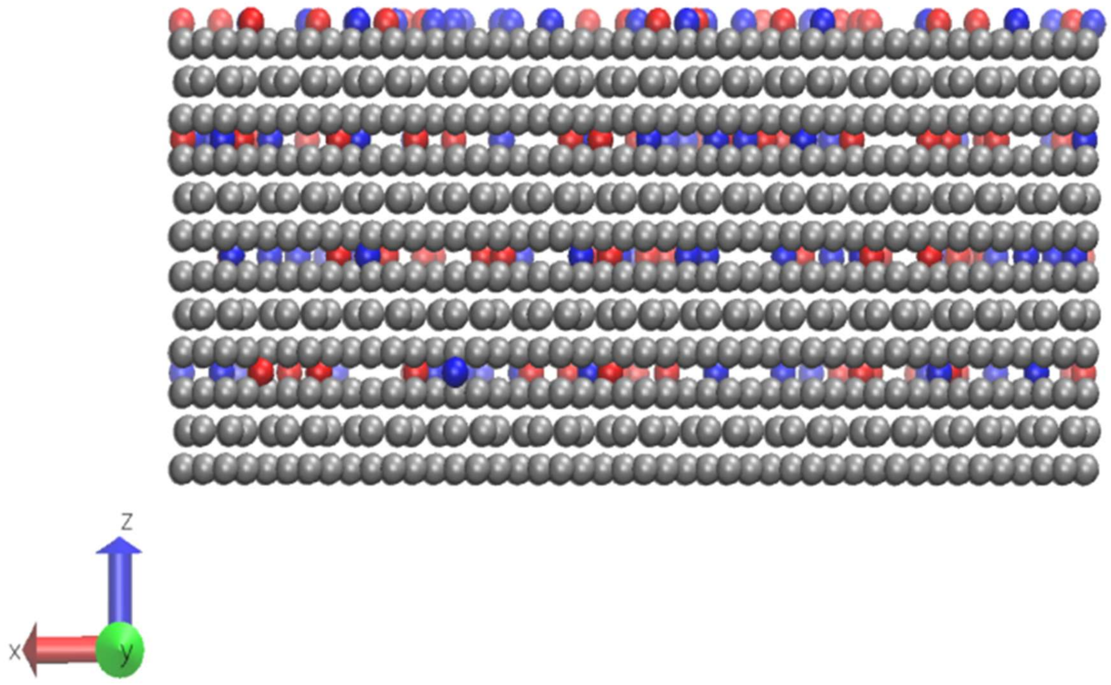


Figure 3.4. Visualization of a periodic four layer homostructured clay particle in VMD.[175] Orthographic representation showing bulk clay (gray), cesium (red), and potassium (blue) coarse-grain types. The structure is periodic in all three dimensions, with ions at the top in contact with the clay layer at the bottom of the image.

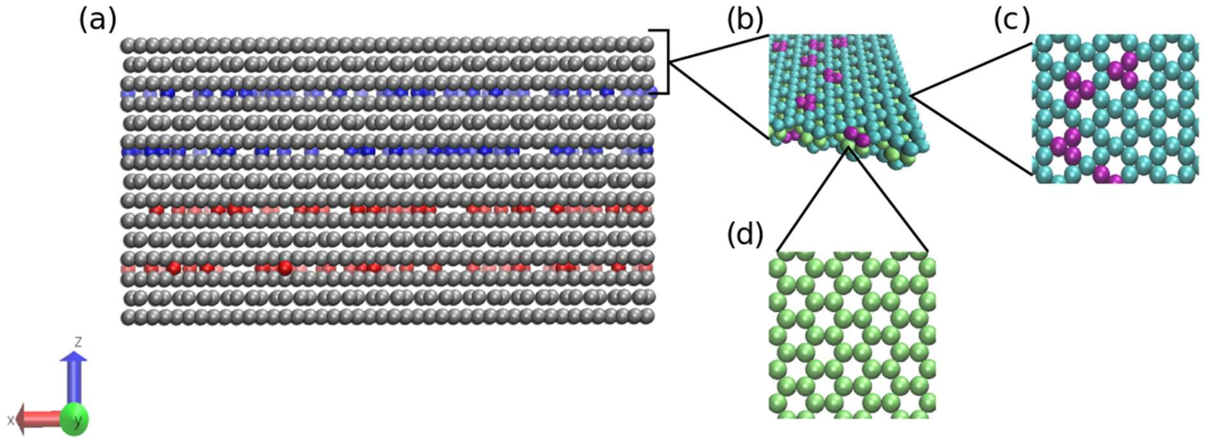


Figure 3.5. Visualization of a non-periodic five layer interstratified clay system in VMD.[175] (a) Orthographic representation showing bulk clay (gray), cesium (red), and potassium (blue) coarse-grain types; (b) top-down visualization of the three sheets in each clay layer; (c) CG centers of the tetrahedral sheet with substitutions shown in purple; (d) CG centers of the octahedral sheet.

The results in Table 3.2 summarize the energy barriers to ion migration as a function of layer Cs^+ content for homostructured K/Cs-illites (Figure 3.4), in which each interlayer consisted of both Cs^+ and K^+ ions positioned randomly at counterion binding sites. As expected, both ions experience dramatically reduced energy barriers in Cs-illite compared to K-illite due to interlayer expansion, and there is a nearly linear trend in the change in barrier height as a function of composition, consistent with prior results using an all-atom forcefield.[116] As a function of the change in equilibrium interlayer spacing, these results correspond to a decrease in the average barrier energy of approximately -71 kJ/mol per Å for Cs^+ and -78 kJ/mol per Å for K^+. This result is similar in magnitude to the change in energy barrier found in the all-atom ClayFF model of -92 kJ/mol per Å for K^+.[116] The information in Table 3.2 is presented graphically in Figure 3.7.

Table 3.2. Average energy barrier (and variance) for Cs^+ and K^+ ion diffusion in a 4-layer periodic clay system as a function of interlayer composition. Each system featured a homogenous mixture of bound Cs^+ and K^+ ions in all four interlayers. Interlayer expansion as a function of composition was very nearly linear, with an interlayer expansion of 0.071 Å under complete exchange for Cs^+.

Fraction Cs^+ in the interlayer	**K^+ NEB Barrier and Variance (kJ/mol)**	**Cs^+ NEB Barrier and Variance (kJ/mol)**	**Interlayer Spacing (nm)**
0	300 ± 94	321 ± 83	0.984
0.25	286 ± 83	309 ± 75	0.997
0.5	279 ± 94	295 ± 68	1.015
0.75	262 ± 84	283 ± 88	1.032
1	243 ± 74	272 ± 83	1.055

One advantage of our ClayCG model is the ability to model numerous clay structures with significantly reduced computational cost. Small clay systems were used to investigate the mixing enthalpy of interstratification due to ion exchange. These systems consisted of four periodic anhydrous interlayer regions between five illite clay layers stacked in a vertical configuration. The outer layers lack counterions on the exterior basal surfaces, which is necessary to allow convergence of the simulation cell size during the shrinkwrap procedure in LAMMPS. Although redistributing these ions in the interlayers is not physically realistic, it is not expected to significantly alter the equilibrium interlayer spacing or NEB energies presented in Table 3.3, because basal spacing is controlled primarily by the counterion size. The excess mixing enthalpy was computed as:

$$\Delta \mathrm{H}_{\mathrm{mix}} = \mathrm{H}_{\mathrm{X}} - \mathrm{f}_{\mathrm{Cs}}\mathrm{H}_{\mathrm{Cs}} - \mathrm{f}_{\mathrm{K}}\mathrm{H}_{\mathrm{K}} \qquad (5)$$

where $\Delta \mathrm{H}_{\mathrm{mix}}$ is the excess enthalpy of mixing (i.e., the difference between the real and ideal enthalpy values), H_{X} is the computed minimum enthalpy of the clay system being simulated, f_{i} is the fraction of ion i in the clay system, and H_{Cs} and H_{K} are the enthalpies of five-layer clay containing only cesium and potassium in their interlayers, respectively (see Methods). For all of the five-layer systems investigated, each interlayer was occupied exclusively by one type of ion and did not have an exchange front. In the following tables, each five-layer system is abbreviated using the identity of the ions in its interlayers from the bottom to the top as a code. For example, the system corresponding to "Cs Cs K K" had two adjacent interlayers filled by cesium ions below two interlayers filled by potassium ions (see Figure 3.5).

Table 3.3 summarizes the trends in the layer basal spacing, energy barriers to migration, and normal stress in the z-direction on both types of ions within different interstratification arrangements. For both K^+ and Cs^+ ions, the overall trend is towards increasing normal stress (and therefore cohesion) with decreasing interlayer spacing. The presence of Cs-illite decreases z-axial stress on K-illite layers, while the presence of K^+ in the structure tends to increase z-axial stress on Cs^+ interlayers. This finding suggests that the longstanding supposition[117] that exchange on a layer increases cohesive energy in neighboring K^+ interlayers is incorrect, at least when the adjacent layer also contains anhydrous counterions. Instead, we find that exchange on one layer, regardless of its proximity to the clay interior, alters the basal spacing and cohesive energies of ions in immediately adjacent layers. As shown in Figure 3.6, the magnitude of this effect decreases rapidly as a function of distance from the exchanged layers and is nearly undetectable only a few layers away from the Cs-illite/K-illite interface.

Table 3.3. Trends in interlayer spacing, ion diffusion energy barrier, z-axial stress, and excess mixing enthalpy versus interstratification. In general, both z-axial stress and diffusion energy barrier increase dramatically with increasing interlayer confinement. Adjacent Cs^+ interlayers are associated with a higher excess enthalpy of mixing, suggesting a thermodynamic compensation for alternation of Cs-illite and K-illite interlayers interstratified particles. For interstratified particles, NEB barriers are computed for interlayers adjacent to the Cs-illite/K-illite interface.

Interstratification Type	**Interlayer spacing (nm)**	**NEB energy (kJ/mol)**	**z-stress (atm)**	**ΔH_{mix} (kJ/mol)**
Cs in Cs Cs Cs Cs	1.055	273	1.25	
Cs in K Cs Cs Cs	1.046	284	1.27	3.33
Cs in Cs Cs K K	1.043	289	1.29	-5.89
Cs in Cs K Cs K	1.038	291	1.35	-9.93
Cs in Cs K K K	1.041	288	1.41	-7.09
Cs in K Cs K K	1.037	286	1.38	-7.32
Cs in Cs K Cs Cs	1.049	279	1.30	0.70
K in K K K K	0.984	299	1.64	
K in Cs K K K	0.994	289	1.60	-7.09
K in Cs Cs K K	0.999	278	1.57	-5.89
K in Cs K Cs K	1.005	275	1.48	-9.93
K in K Cs Cs Cs	1.001	273	1.47	3.33
K in K Cs K K	0.990	294	1.60	-7.32
K in Cs K Cs Cs	1.001	274	1.50	0.70

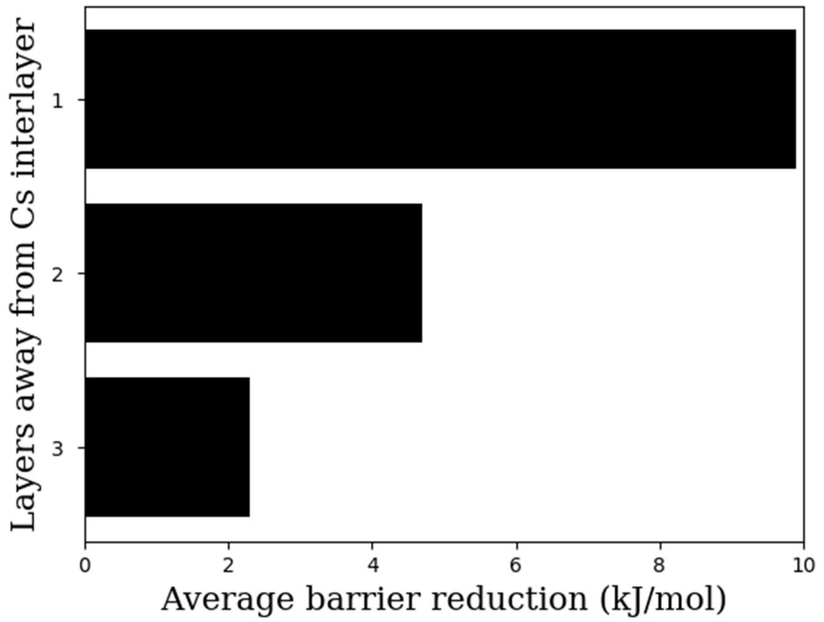


Figure 3.6. Reduction in K^+ diffusion barrier in a four layer interstratified particle. The barrier reduction to K^+ diffusion in a "Cs K K K" particle relative to bulk K-illite is shown as a function of the distance from the Cs-illite/K-illite interface.

For both types of ions, there is a clear trend of increasing energy barrier under increased confinement. In the case of only one Cs-illite layer present in bulk K-illite, it is clear from the interlayer spacing and normal stress that the cesium interlayer experiences maximal compression. To confirm this, NEB calculations were run on the displacement of a Cs^+ ion in Cs-illite in the middle of a 12-layer K-illite particle. The barrier in this case was found to be 288 kJ/mol, very similar to the 286 kJ/mol barrier for Cs^+ diffusion in the "K Cs K K" particle (Table 3.3), supporting the conclusion that interlayer compression and expansion in interstratified particles is a localized effect. Similarly, a K-illite layer isolated in bulk cesium relaxes to a much larger spacing.

The compensatory expansion and compression of neighboring layers may explain why the change in Cs^+ and K^+ diffusion barriers in interstratified particles is greater than the corresponding change in homostructured particles for a given interlayer spacing (Figure 3.7). In homostructured clays, the local interlayer spacing near a Cs^+ counterion is greater than the average interlayer spacing due to its larger atomic radius, resulting in a lower migration barrier for a given spacing. The opposite is true for K^+ counterions, which experience greater local confinement in homostructured clays than would be expected from measuring the average interlayer spacing alone. In contrast, the compression and expansion of interlayers in interstratified particles results in a more uniform interlayer spacing, and therefore a more dramatic change in the diffusion barrier. This effect may be an artifact of using ClayFF as the all-atom model, as ClayFF is known to overpredict flexibility in large clay systems.

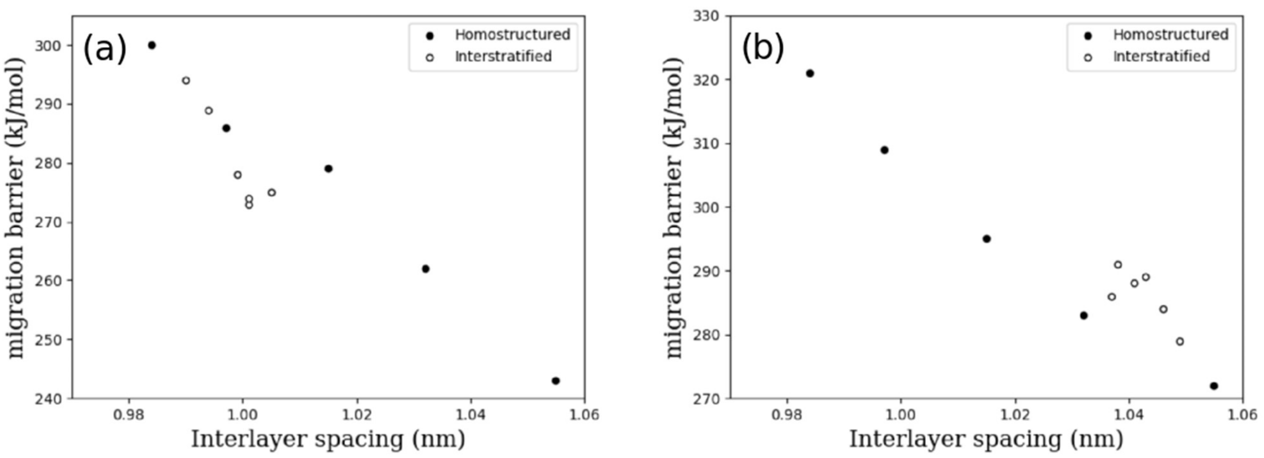


Figure 3.7. Migration barriers to counterion diffusion in homostructured and interstratified clay particles. (a) Barriers for K^+ migration; (b) barriers for Cs^+ migration. These plots summarize the NEB data from Table 3.2 and Table 3.3.

From the trends in the energy barriers to ion diffusion in K-illite interlayers adjacent to Cs-illite interlayers (Table 3.3), one would assume that exchange would be enhanced near cesium-dominated interlayers instead of being inhibited. That is, diffusion energy barriers in neighboring K-illite interlayers are lowered in the immediate vicinity of a Cs-illite interlayer. Experimental evidence for the Cs/K system is insufficient to confirm or refute this hypothesis, but there is some visual evidence that Cs exchanged layers occur in clumps in a K-phlogopite and are not randomly distributed, as expected from these results.[124]

3.3.4 Impact of the Exchange Front on Barriers

Exchange of ions of different size leads to bending deformation of the layer structure locally, which may alter the coordination of K^+ in the vicinity of the exchange front.[176] In this case, the selectivity for and mobility of K^+ is expected to vary with the sharpness and uniformity of the front. In order to capture the effects of the sharp exchange front in anhydrous illite clay systems,[116,124,125] a series of simulations were performed in which all ions on one half of the exchanging layer were assigned to be Cs^+ and all ions on the other half were assigned to be K^+. The exchange front was modelled in a periodic, four-layer clay system that featured one completely exchanged interlayer below the exchange front and two fully unexchanged interlayers above the exchange front. Ions far from the exchange front did not have significantly altered energy barriers to diffusion compared to barriers in an interstratified particle, but K^+ ions characterized at the interface showed a slightly reduced average energy barrier compared to an interstratified clay with no exchange front barrier (273 kJ/mol vs. 278 kJ/mol). In comparison with the results of the energy barrier distribution presented in Table 3.2, K^+ ions near the exchange front experience approximately 20% additional barrier lowering (27 kJ/mol) with respect to the barrier in pure K-illite as compared to ions far from the front (22 kJ/mol), and ions more than approximately 2 to 3 nm from the exchange front are essentially unaffected by it.

3.3.5 Thermodynamic Compensation for Ordered Interstratification

Amongst all of the non-periodic four interlayer systems studied, the system "K Cs K Cs" exhibited the greatest thermodynamic compensation for interstratification. Overall, the ΔH_{mix} trends indicate that there is a thermodynamic driver during the exchange process to form alternating K-illite and Cs-illite layers (i.e. ordered interstratified structures).[128,144,177] Exchange will tend to disrupt bulk K-illite, and thermodynamic feedback will favor exchange that leads to the formation of ordered interstratification instead of regions of bulk Cs-illite. However, the thermodynamic compensation for forming ordered interstratification is quite small (at most around 10 kJ/mol), indicating that it is unlikely to be the primary cause of experimentally observed differences in the exchange rate of adjacent interlayers.[124,125]

3.4 Conclusions

We have presented a coarse-grain model of anhydrous K- and Cs-illite that represents a 70-fold speedup over the corresponding all-atom ClayFF forcefield. Although using tabulated potentials between the oxygen coarse-grain centers may slightly improve the fidelity of modelling the ion binding sites, there is a non-negligible computational cost savings associated with using Lennard-Jones potentials as compared to splined tabulated potentials.[178] While other CG clay models have represented ion binding sites as single CG centers,[155,156] the model presented here is capable of capturing physical degrees of freedom important during ion diffusion in the confined interlayer.[125,136] By modelling all the atoms within and adjacent to the interlayer, we were able to accurately reproduce the structure of ion binding sites without significant computational overhead.

The reduction in particle density due to the coarse-grain procedure is especially helpful for nudged elastic band simulations, since the number of molecular dynamics steps necessary to reach convergence of the energy pathway is heavily dependent on system size and the number of particles surrounding the transition path. We were able to converge NEB pathways on relatively large systems, which is promising for probing the energy barriers in physical scenarios that would necessitate investigating bulk effects. Our CG model qualitatively reproduced the ion diffusion energy trends as a function of interlayer separation found in ClayFF[116] and demonstrates interlayer expansion near sharp exchange fronts and near fully exchanged layers. Our model indicates a significant enthalpy of mixing associated with adjacent K- and Cs-illite interlayers in interstratified particles, which may impact exchange front propagation in adjacent interlayers. The coarse-graining strategy used here is expected to generalize well to other anhydrous and swelling clay types, and the intrasheet CG bond potentials will be directly portable to other clay systems.

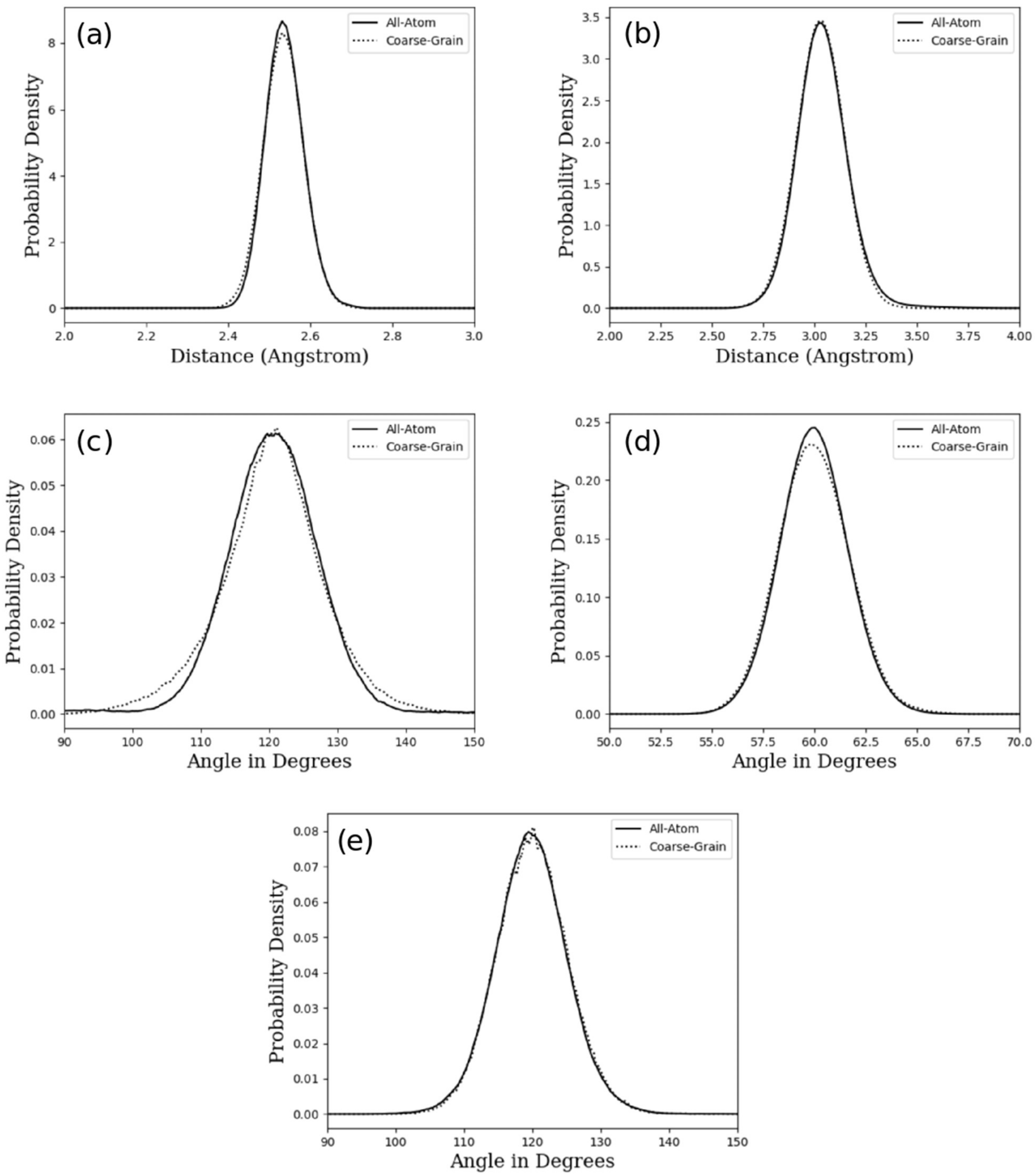
3.5 Acknowledgements

The Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory supported KS and LNL under U.S. Department of Energy Contract No. DE-AC02-05CH11231. THG and LRP acknowledge support by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

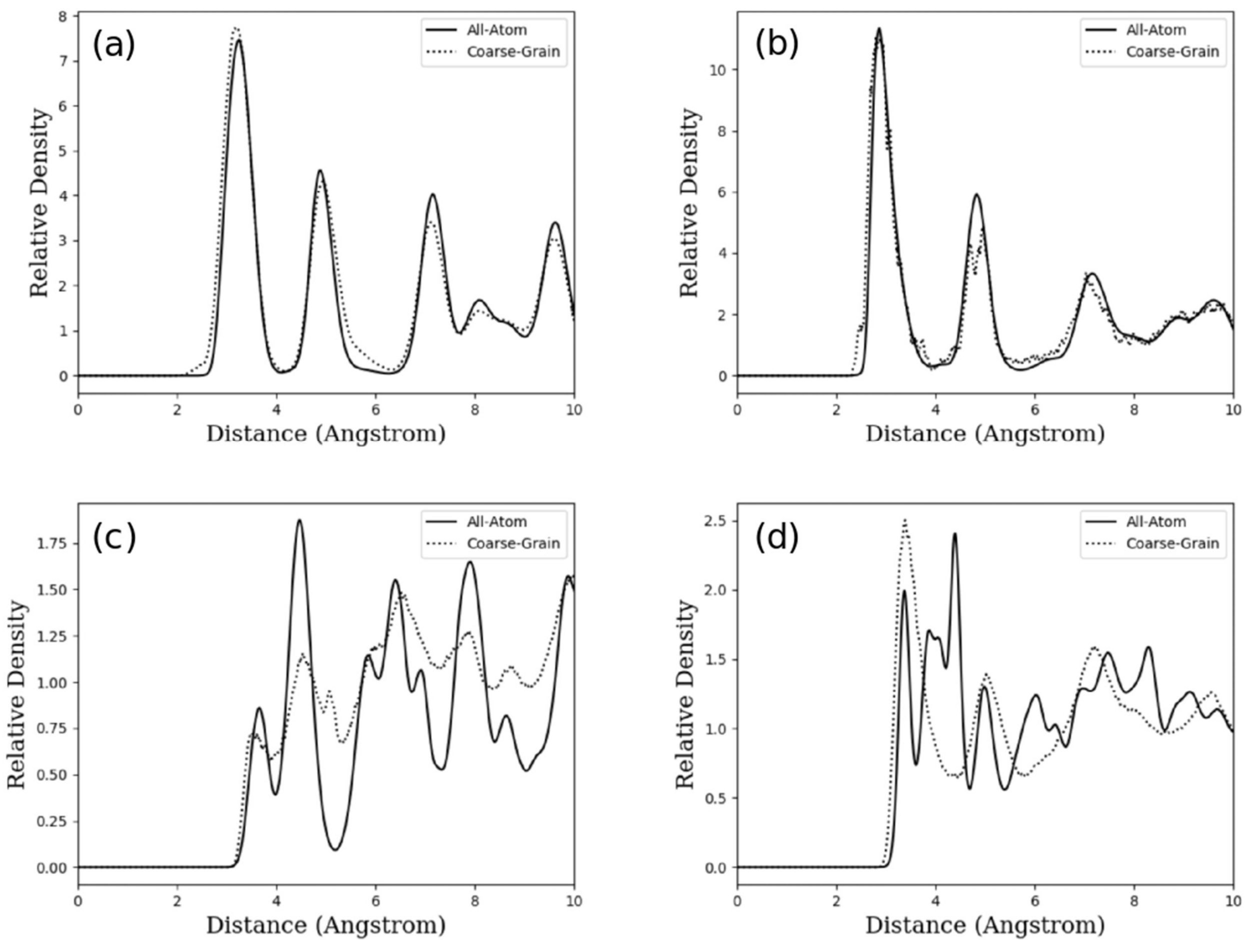
3.6 Supplement

Coarse-grain and all-atom simulations were run in LAMMPS on the Edison and Cori supercomputers at the National Energy Research Scientific Computing Center (NERSC) as well as our own in-house supercomputing cluster, Armada. All-atom simulation data was generated from 2-layer K-illite and Cs-illite clay systems with randomized substitution sites. For each ion X, the composition of the simulated illite clay was $X_{0.7}[Si_{3.3}Al_{0.7}]$-$Al_2O_{10}(OH)_2$. All-atom systems were equilibrated with a 1 fs timestep in the NVE ensemble for 120 picoseconds, and subsequently sampled every 250 fs over a period of 40 ps. Coarse-grain simulations were run in the NVT ensemble for 40 picoseconds after an equilibration time of 100 picoseconds with a timestep of 3 fs. The Coulombic interactions for the coarse-grain model were computed using a particle-particle particle mesh with a cutoff of 10 Å. The cutoff distance was the same for both Lennard-Jones and tabulated interactions between the coarse-grain centers. These simulations were run in the NVT ensemble at 300 Kelvin using a Nosé-Hoover thermostat and barostat. The all-atom reference model was simulated under the same conditions to ensure transferability.

The all-atom radial distribution functions in Supplementary Figure 2.1 were fitted with harmonic potentials to derive an initial guess for the corresponding coarse-grain potentials. These initial potentials were used along with non-bonded interaction potentials derived from the all-atom RDFs in Supplementary Figure 2.2. However, this first guess very poorly represented the clay structure and failed to reproduce flat clay sheets. Better initial parameters for the harmonic potentials were found by running the iterative Boltzmann inversion (IBI) algorithm[169,170] on isolated, uncharged sheets, and then subsequently re-introducing non-bonded interactions. We then used IBI to converge the potentials associated with each of the degrees of freedom in Supplementary Figure 2.1 and Supplementary Figure 2.2 simultaneously, which required around 100 iterations to achieve high fidelity in reproducing the oxygen-ion RDFs. Although IBI is much more difficult to converge by changing multiple potentials after each round, the final CG model was able to more accurately reproduce the degrees of freedom near ions in the interlayer in the presence of the full force field. The potentials in the coarse-grain model were derived by iteratively simulating an anhydrous two-layer periodic clay system, with dimensions of 93 Å x 60 Å x 20 Å.

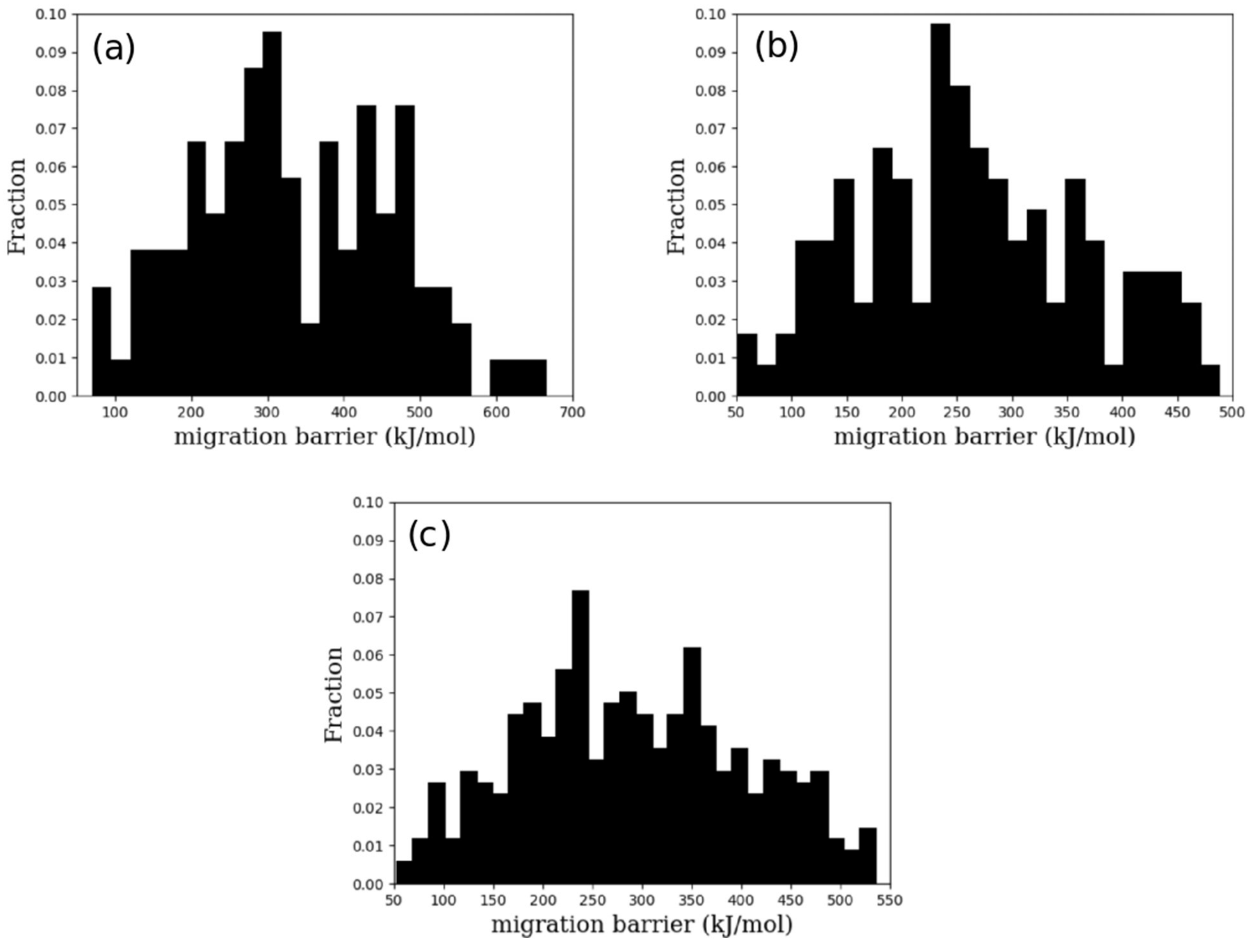


Supplementary Figure 3.1. Comparison of the probability distribution of bond distances and bond angles for the all-atom (solid) and coarse-grained model (dotted). (a) Os-Os CG bond; (b) Al-Al CG bond; (c) O-O-O CG type 120 degree angle; (d) Os-Os-Os CG type 60 degree angle; (e) Al-Al-Al CG type 120 degree angle.



Supplementary Figure 3.2. Comparison of the radial distribution function (RDF) for the all-atom (solid) and coarse-grained model (dotted). RDFs for (a) O-K CG types; (b) Os-K CG types; (c) O-O CG types; (d) O-Al CG types.

The data in Supplementary Figure 3.3 was generated using the climbing image nudged elastic band (CINEB) method implemented in LAMMPS. In CINEB, the highest-energy system replica is driven to a true saddle point to better approximate the exact minimum energy path for a given transition.[163–166] Fictitious spring forces are introduced between images, which are initialized as a linear interpolation between the equilibrium bound state and proposed final state. Since the initial and final states are also driven to an energy minimum during CINEB, randomization of ion positions can be implemented efficiently by displacing ions according to the underlying geometry of the clay interlayer. Substitutions and ion configurations in the interlayer were randomized during the simulation of both periodic and non-periodic clay particles for all CINEB results presented in this work. 25 images of the clay system were run in parallel on 25 cores for each energy barrier calculation. The simulations were run with a 5 fs timestep to enhance accurate convergence of the diffusion pathway.



Supplementary Figure 3.3. Migration barrier distributions for K^+ diffusion in K^+-illite as a function of the number of interlayers. Each system is periodic in all three dimensions. (a) 2 interlayers, (b) 4 interlayers, (c) 12 interlayers.

4 Summary

The goal of this dissertation is to present interpretable reduced order models of existing machine learning methods and all-atom simulations that faithfully retain the properties of the original model. A reduced order surrogate model was developed with the intention of being used in high-dimensional optimization of return on investment or other target functions. We developed a coarse-grain model of a clay system to better understand ion behavior near an exchange interface in the bulk phase. By randomizing ion position, we were able to implement iterative Boltzmann inversion in a crystalline system without a damping factor and better sample the entire phase space. Both of these models are more computationally tractable than the models they replace, and in the case of SMiRFs are much more quantitatively interpretable in low-dimensional representations of the feature space.

Chapter 2 presents a methodology for dense imputation of discrete soil data derived from soil samples using random forest regression against dense features such as multispectral images and electrical conductivity. By generating a dense set of soil nutrient concentrations, we were able to use them as features to build a quantitative model of crop yield as a function of soil properties. Generating a fundamentally localized model of sparse data as a function of dense data results in a model that is much more general than by using kriging alone, especially when the dense datasets are expected to be correlated to the sparse soil data. We found that we were able to predict a map of fertilizer application that would boost yield by approximately 100 kg/ha with essentially no additional cost by optimizing the ROI of a reduced order surrogate model of a random forest model of yield. The SMiRF methodology is particularly well suited to datasets with a high-dimensional feature space, because the iterative random forest procedure extracts low-dimensional, high-importance feature interactions in a way that scales very slowly with dimension. We were able to reproduce and expand on existing agronomic trends in relatively healthy soil by using a purely data-driven approach, attesting to the efficacy of treating each 100 m^2 section of a field as a roughly independent test sample, thereby drastically increasing the effective number of samples that can be used in crop modelling.

Chapter 3 focuses on a coarse-grain representation of illite clay with cesium and potassium ions in the interlayer. Through iterative Boltzmann inversion we were able to efficiently derive a roughly 2:1 model that in practice runs approximately 70 times faster than a corresponding all-atom simulation. The CG model is also capable of simulating dozens of clay layers simultaneously. Using CINEB simulations, we found that the barrier to ion site translocation depends significantly on the interlayer spacing, and moreover that there is a thermodynamic compensation associated with alternating layers filled with cesium and potassium ions, consistent with experimental results. By simulating multiple arrangements of alternating interlayers, we were able to show that ions in a layer far from an exchanged interlayer do not have a significant associated enthalpy of mixing, while those in neighboring layers have a greatly altered thermodynamic landscape. By using structural coarse-graining in a rigid phase, we were able to very accurately reproduce the mechanical behavior of the binding sites as indicated by the extremely close match between the radial distribution functions of the all-atom and coarse-grain centers.

5 References

1. Musser, W. N. & Patrick, G. F. How Much does Risk Really Matter to Farmers? in *A Comprehensive Assessment of the Role of Risk in U.S. Agriculture* (2002). doi:10.1007/978-1-4757-3583-3_24.
2. MacDonald, J. M., Korb, P. & Hoppe, R. A. *Farm Size and the Organization of U.S. Crop Farming, ERR-152. Economic Research Report* (2013).
3. Finger, R., Swinton, S. M., El Benni, N. & Walter, A. Precision Farming at the Nexus of Agricultural Production and the Environment. *Annual Review of Resource Economics* vol. 11 Preprint at https://doi.org/10.1146/annurev-resource-100518-093929 (2019).
4. Nascente, A. S., Li, Y. C. & Crusciol, C. A. C. Cover crops and no-till effects on physical fractions of soil organic matter. *Soil Tillage Res* (2013) doi:10.1016/j.still.2013.02.008.
5. Ashworth, A. J., Allen, F. L., Saxton, A. M. & Tyler, D. D. Impact of crop rotations and soil amendments on long-term no-tilled soybean yield. *Agron J* (2017) doi:10.2134/agronj2016.04.0224.
6. Wojtowicz, M., Wojtowics, A. & Piekarczyk, J. Application of remote sensing methods in Agriculture. *Communication in Biometry and Crop Science* (2015) doi:10.2478/gein-2014-0007.
7. Das, J. *et al.* Devices, systems, and methods for automated monitoring enabling precision agriculture. *Automation Science and Engineering (CASE), 2015 IEEE International Conference on* (2015) doi:10.1109/CoASE.2015.7294123.
8. Krige, D. G. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of Southern African Institute of Mining and Metallurgy* **52**, (1951).
9. Krige, D. G. A Statistical Analysis of Some of the Borehole Values in the Orange Free State Goldfield. *Journal of Chemical, Metallurgical and Mining Society of South Africa* 47–64 (1952).
10. Shukla, G., Mishra, G. C. & Singh, S. K. Kriging Approach for Estimating Deficient Micronutrients in the Soil: A Case Study. *International Journal of Agriculture, Environment and Biotechnology* **8**, (2015).
11. Dafonte, J. D., Guitián, M. U., Paz-Ferreiro, J., Siqueira, G. M. & Vázquez, E. V. Mapping of soil micronutrients in an european atlantic agricultural landscape using ordinary kriging and indicator approach. *Bragantia* **69**, (2010).
12. Chilès, J. P. & Desassis, N. Fifty years of kriging. in *Handbook of Mathematical Geosciences: Fifty Years of IAMG* (2018). doi:10.1007/978-3-319-78999-6_29.
13. Hengl, T., Heuvelink, G. B. M. & Rossiter, D. G. About regression-kriging: From equations to case studies. *Comput Geosci* **33**, (2007).
14. Pham, T. G., Kappas, M., Huynh, C. Van & Nguyen, L. H. K. Application of ordinary kriging and regression kriging method for soil properties mapping in hilly region of central Vietnam. *ISPRS Int J Geoinf* **8**, (2019).
15. Chen, Y. R., Chao, K. & Kim, M. S. Machine vision technology for agricultural applications. in *Computers and Electronics in Agriculture* (2002). doi:10.1016/S0168-1699(02)00100-X.
16. Billingsley, J. Machine Vision in Agriculture. *Encyclopedia of Agrophysics* 433–436 (2011) doi:10.1007/978-90-481-3585-1.

17. Zhang, Z., Kodagoda, S., Ruiz, D., Katupitiya, J. & Dissanayake, G. Classification of Bidens in wheat farms. in *15th International Conference on Mechatronics and Machine Vision in Practice, M2VIP'08* (2008). doi:10.1109/MMVIP.2008.4749584.
18. Kasampalis, D. *et al.* Contribution of Remote Sensing on Crop Models: A Review. *J Imaging* (2018) doi:10.3390/jimaging4040052.
19. Stafford, J. V. Implementing precision agriculture in the 21st century. *Journal of Agricultural and Engineering Research* (2000) doi:10.1006/jaer.2000.0577.
20. Oppelt, N. M. Use of remote sensing data to assist crop modeling. *J Appl Remote Sens* (2010) doi:10.1117/1.3491191.
21. Mariotto, I., Thenkabail, P. S., Huete, A., Slonecker, E. T. & Platonov, A. Hyperspectral versus multispectral crop-productivity modeling and type discrimination for the HyspIRI mission. *Remote Sens Environ* (2013) doi:10.1016/j.rse.2013.08.002.
22. Yao, X. *et al.* Estimation of wheat LAI at middle to high levels using unmanned aerial vehicle narrowband multispectral imagery. *Remote Sensing* Preprint at https://doi.org/10.3390/rs9121304 (2017).
23. Vithu, P. & Moses, J. A. Machine vision system for food grain quality evaluation: A review. *Trends Food Sci Technol* (2016) doi:10.1016/j.tifs.2016.07.011.
24. Nandi, C. S., Tudu, B. & Koley, C. Machine vision based techniques for automatic mango fruit sorting and grading based on maturity level and size. in *Smart Sensors, Measurement and Instrumentation* (2014). doi:10.1007/978-3-319-02315-1_2.
25. Ebrahimi, E., Mollazade, K. & Babaei, S. Toward an automatic wheat purity measuring device: A machine vision-based neural networks-assisted imperialist competitive algorithm approach. *Measurement (Lond)* (2014) doi:10.1016/j.measurement.2014.05.003.
26. Gottschalk, R., Burgos-Artizzu, X. P., Ribeiro, Á., Pajares, G. & Sánchez-Miralles, Á. Real-time image processing for the guidance of a small agricultural field inspection vehicle. in *15th International Conference on Mechatronics and Machine Vision in Practice, M2VIP'08* (2008). doi:10.1109/MMVIP.2008.4749582.
27. Chlingaryan, A., Sukkarieh, S. & Whelan, B. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. *Comput Electron Agric* **151**, 61–69 (2018).
28. Busch, J. W. & Phelan, P. L. Mixture models of soybean growth and herbivore performance in response to nitrogen-sulphur-phosphorous nutrient interactions. *Ecol Entomol* **24**, (1999).
29. Beanland, L., Phelan, P. L. & Salminen, S. Micronutrient interactions on soybean growth and the developmental performance of three insect herbivores. *Environ Entomol* **32**, (2003).
30. Shen, J., Li, R., Zhang, F., Rengel, Z. & Tang, C. Orthogonal polynomial models to describe yield response of rice to nitrogen and phosphorus at different levels of soil fertility. *Nutr Cycl Agroecosyst* **65**, (2003).
31. Moreira, A., Moraes, L. A. C., Moretti, L. G. & Aquino, G. S. Phosphorus, Potassium and Sulfur Interactions in Soybean Plants on a Typic Hapludox. *Commun Soil Sci Plant Anal* **49**, (2018).
32. Jeong, J. H. *et al.* Random forests for global and regional crop yield predictions. *PLoS One* **11**, (2016).
33. Bagale, S. Nutrient Management for Soybean Crops. *International Journal of Agronomy* vol. 2021 Preprint at https://doi.org/10.1155/2021/3304634 (2021).

34. Shamsi, I. H. *et al.* Alleviation of cadmium toxicity in soybean by potassium supplementation. *J Plant Nutr* **33**, (2010).
35. Kumar, S., Kumar, S. & Mohapatra, T. Interaction Between Macro- and Micro-Nutrients in Plants. *Frontiers in Plant Science* vol. 12 Preprint at https://doi.org/10.3389/fpls.2021.665583 (2021).
36. van Klompenburg, T., Kassahun, A. & Catal, C. Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture* vol. 177 Preprint at https://doi.org/10.1016/j.compag.2020.105709 (2020).
37. Filippi, P. *et al.* An approach to forecast grain crop yield using multi-layered, multi-farm data sets and machine learning. *Precis Agric* **20**, 1015–1029 (2019).
38. Vohra, M., Nath, P., Mahadevan, S. & Tina Lee, Y. T. Fast surrogate modeling using dimensionality reduction in model inputs and field output: Application to additive manufacturing. *Reliab Eng Syst Saf* **201**, (2020).
39. Ansarifar, J., Wang, L. & Archontoulis, S. V. An interaction regression model for crop yield prediction. *Sci Rep* **11**, (2021).
40. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space . *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**, (1901).
41. Jolliffe, I. T. Principal Component Analysis, Second Edition. *Encyclopedia of Statistics in Behavioral Science* **30**, (2002).
42. Zhang, T. & Yang, B. Big Data Dimension Reduction Using PCA. in *Proceedings - 2016 IEEE International Conference on Smart Cloud, SmartCloud 2016* (2016). doi:10.1109/SmartCloud.2016.33.
43. Dela Cruz, G. B., Gerardo, B. D. & T. Tanguilig III, B. Agricultural Crops Classification Models Based on PCA-GA Implementation in Data Mining. *International Journal of Modeling and Optimization* **4**, (2014).
44. Reddy, G. T. *et al.* Analysis of Dimensionality Reduction Techniques on Big Data. *IEEE Access* **8**, (2020).
45. Geetha, R., Sivasubramanian, S., Kaliappan, M., Vimal, S. & Annamalai, S. Cervical Cancer Identification with Synthetic Minority Oversampling Technique and PCA Analysis using Random Forest Classifier. *J Med Syst* **43**, (2019).
46. Ribeiro, M. T., Singh, S. & Guestrin, C. 'Why should i trust you?' Explaining the predictions of any classifier. in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* vols 13-17-August-2016 (2016).
47. Bunemann, E. K., Schwenke, G. D. & Van Zwieten, L. AUSTRALIAN JOURNAL OF SOIL RESEARCH. *Impact of agricultural inputs on soil organisms - a review* Preprint at (2006).
48. Sadeghpour, A., Ketterings, Q. M., Godwin, G. S. & Czymmek, K. J. Under- or over-application of nitrogen impact corn yield, quality, soil, and environment. *Agron J* (2017) doi:10.2134/agronj2016.06.0355.
49. Albornoz, F. Crop responses to nitrogen overfertilization: A review. *Scientia Horticulturae* Preprint at https://doi.org/10.1016/j.scienta.2016.04.026 (2016).
50. Graham, M. H. & Haynes, R. J. Organic matter accumulation and fertilizer-induced acidification interact to affect soil microbial and enzyme activity on a long-term sugarcane management experiment. *Biol Fertil Soils* (2005) doi:10.1007/s00374-005-0830-2.

51. Moore, J. M., Klose, S. & Tabatabai, M. A. Soil microbial biomass carbon and nitrogen as affected by cropping systems. *Biol Fertil Soils* **31**, 200–210 (2000).
52. Sarathchandra, S. U., Ghani, A., Yeates, G. W., Burch, G. & Cox, N. R. Effect of nitrogen and phosphate fertilisers on microbial and nematode diversity in pasture soils. *Soil Biol Biochem* (2001) doi:10.1016/S0038-0717(00)00245-5.
53. Smith, D. R., Huang, C. & Haney, R. L. Phosphorus fertilization, soil stratification, and potential water quality impacts. *J Soil Water Conserv* (2017) doi:10.2489/jswc.72.5.417.
54. Chen, Y., De Nobili, M. & Aviad, T. Stimulatory effect of humic substances on plant growth. in *Soil organic matter in sustainable agriculture* (eds. Magdoff, F. & Weil, R. R.) 103–130 (2004).
55. Merrington, G., Rogers, S. L. & Van Zwieten, L. The potential impact of long-term copper fungicide usage on soil microbial biomass and microbial activity in an avocado orchard. *Australian Journal of Soil Research* (2002) doi:10.1071/SR01084.
56. Noid, W. G. Perspective: Coarse-grained models for biomolecular systems. *Journal of Chemical Physics* vol. 139 Preprint at https://doi.org/10.1063/1.4818908 (2013).
57. Shell, M. S. Coarse-Graining With The Relative Entropy. in *Advances in Chemical Physics* vol. 161 (2016).
58. Agrawal, V., Arya, G. & Oswald, J. Simultaneous iterative boltzmann inversion for coarse-graining of polyurea. *Macromolecules* **47**, (2014).
59. Hadley, K. R. & McCabe, C. A coarse-grained model for amorphous and crystalline fatty acids. *Journal of Chemical Physics* **132**, (2010).
60. Wang, Z. J. & Deserno, M. A systematically coarse-grained solvent-free model for quantitative phospholipid bilayer simulations. *Journal of Physical Chemistry B* **114**, (2010).
61. Wang, Z. J. & Deserno, M. Systematic implicit solvent coarse-graining of bilayer membranes: Lipid and phase transferability of the force field. *New J Phys* **12**, (2010).
62. Moore, T. C., Iacovella, C. R. & McCabe, C. Derivation of coarse-grained potentials via multistate iterative Boltzmann inversion. *Journal of Chemical Physics* **140**, (2014).
63. Srinivas, G., Cheng, X. & Smith, J. C. A solvent-free coarse grain model for crystalline and amorphous cellulose fibrils. *J Chem Theory Comput* **7**, (2011).
64. Ercolesi, F. & Adams, J. B. Interatomic potentials from first-principles calculations: The force-matching method. *EPL* **26**, (1994).
65. Izvekov, S., Parrinello, M., Bumham, C. J. & Voth, G. A. Effective force fields for condensed phase systems from ab initio molecular dynamics simulation: A new method for force-matching. *Journal of Chemical Physics* **120**, (2004).
66. Izvekov, S. & Voth, G. A. A multiscale coarse-graining method for biomolecular systems. *Journal of Physical Chemistry B* **109**, (2005).
67. Noid, W. G. *et al.* The multiscale coarse-graining method. I. A rigorous bridge between atomistic and coarse-grained models. *Journal of Chemical Physics* **128**, (2008).
68. Izvekov, S. & Rice, B. M. Free-energy based pair-additive potentials for bulk Ni-Al systems: Application to study Ni-Al reactive alloying. *Journal of Chemical Physics* **137**, (2012).
69. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics* **22**, (1951).
70. Shell, M. S. The relative entropy is fundamental to multiscale and inverse thermodynamic problems. *Journal of Chemical Physics* **129**, (2008).

71. Carmichael, S. P. & Shell, M. S. A new multiscale algorithm and its application to coarse-grained peptide models for self-assembly. *Journal of Physical Chemistry B* **116**, (2012).
72. Sanyal, T., Mittal, J. & Shell, M. S. A hybrid, bottom-up, structurally accurate, G o -like coarse-grained protein model. *Journal of Chemical Physics* **151**, (2019).
73. Bernhardt, E. S. *et al.* Control Points in Ecosystems: Moving Beyond the Hot Spot Hot Moment Concept. *Ecosystems* **20**, 665–682 (2017).
74. Michel, L. & Makowski, D. Comparison of statistical models for analyzing wheat yield time series. *PLoS One* **8**, (2013).
75. United Nations Department of Economic and Social Affairs Population Division. *World Population Prospects The 2017 Revision Key Findings and Advance Tables*. *World Population Prospects The 2017* (2017) doi:10.1017/CBO9781107415324.004.
76. FAO. How to Feed the World in 2050. *Insights from an expert meeting at FAO* (2009) doi:10.1111/j.1728-4457.2009.00312.x.
77. How to feed a hungry world. *Nature* (2010) doi:10.1038/466531a.
78. Beman, J. M., Arrigo, K. R. & Matson, P. A. Agricultural runoff fuels large phytoplankton blooms in vulnerable areas of the ocean. *Nature* (2005) doi:10.1038/nature03370.
79. Leslie, J. E., Weersink, A., Yang, W. & Fox, G. Actual versus environmentally recommended fertilizer application rates: Implications for water quality and policy. *Agric Ecosyst Environ* (2017) doi:10.1016/j.agee.2017.02.009.
80. Baumhardt, R. L., Stewart, B. A. & Sainju, U. M. North American soil degradation: Processes, practices, and mitigating strategies. *Sustainability (Switzerland)* (2015) doi:10.3390/su7032936.
81. Hedley, C. The role of precision agriculture for improved nutrient management on farms. *Journal of the Science of Food and Agriculture* Preprint at https://doi.org/10.1002/jsfa.6734 (2015).
82. Peterson, G. A. *et al.* Reduced tillage and increasing cropping intensity in the Great Plains conserves soil C. *Soil Tillage Res* (1998) doi:10.1016/S0167-1987(98)00107-X.
83. Heinemann, J. A., Massaro, M., Coray, D. S., Agapito-Tenfen, S. Z. & Wen, J. D. Sustainability and innovation in staple crop production in the US Midwest. *Int J Agric Sustain* **12**, 71–88 (2014).
84. Gebbers, R. & Adamchuk, V. I. Precision agriculture and food security. *Science* (2010) doi:10.1126/science.1183899.
85. Falco, N. *et al.* Influence of soil heterogeneity on soybean plant development and crop yield evaluated using time-series of UAV and ground-based geophysical imagery. *Sci Rep* **11**, (2021).
86. Anderson-Cook, C. M. *et al.* Differentiating Soil Types Using Electromagnetic Conductivity and Crop Yield Maps. *Soil Science Society of America Journal* (2002) doi:10.2136/sssaj2002.1562.
87. Hubbard, S. S. *et al.* Estimation of soil classes and their relationship to grapevine vigor in a Bordeaux vineyard: advancing the practical joint use of electromagnetic induction (EMI) and NDVI datasets for precision viticulture. *Precis Agric* (2021) doi:10.1007/s11119-021-09788-w.
88. Schimmelpfennig, D. *Farm Profits and Adoption of Precision Agriculture*. (2016).
89. Yost, M. A. *et al.* Long-term impact of a precision agriculture system on grain crop production. *Precis Agric* (2017) doi:10.1007/s11119-016-9490-5.

90. Bongiovanni, R. & Lowenberg-Deboer, J. Precision agriculture and sustainability. *Precision Agriculture* Preprint at https://doi.org/10.1023/B:PRAG.0000040806.39604.aa (2004).
91. Basu, S. & Kumbier, K. iRF: iterative Random Forests. Preprint at (2017).
92. R core team. R: A language and environment for statistical computing. *R Foundation for Statistical Computing, Vienna, Austria.* (2017) doi:http://www.R-project.org/.
93. Shah, R. D. & Meinshausen, N. Random Intersection Trees. *The Journal of Machine Learning Research* (2013).
94. Cano, G. *et al.* Automatic selection of molecular descriptors using random forest: Application to drug discovery. *Expert Syst Appl* (2017) doi:10.1016/j.eswa.2016.12.008.
95. Borkowski, John. Response Surface Methodology: Process and Product Optimization Using Designed Experiments (3rd ed.). by Raymond H. Myers, Douglas C. Montgomery, and Christine M. Anderson-Cook. *J Am Stat Assoc* (2010).
96. Andre I. Khuri. *Response Surface Methodology and Related Topics*. *World Scientific Publishing Co. Pte. Ltd.* (2006). doi:10.2139/ssrn.2395836.
97. Falco, N. *et al.* Remote sensing to UAV-based digital farmland. in *International Geoscience and Remote Sensing Symposium (IGARSS)* vols 2018-July (2018).
98. Oliphant, T. E. SciPy: Open source scientific tools for Python. *Comput Sci Eng* (2007) doi:10.1109/MCSE.2007.58.
99. Breiman, L. Random forests. *Mach Learn* (2001) doi:10.1023/A:1010933404324.
100. Kumbier, K., Basu, S., Brown, J. B., Celniker, S. & Yu, B. Refining interaction search through signed iterative Random Forests. *arxiv.org* 1–26 (2018).
101. Wainwright, M. J. *High-Dimensional Statistics*. (2019). doi:https://doi.org/10.1017/9781108627771.
102. Benkeser, D. & Van Der Laan, M. The highly adaptive lasso estimator. in *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016* (2016). doi:10.1109/DSAA.2016.93.
103. Hejazi, N., Coyle, J. & van der Laan, M. hal9001: Scalable highly adaptive lasso regression in R. *J Open Source Softw* **5**, (2020).
104. Hopkins, B. G. & Hansen, N. C. Phosphorus Management in High-Yield Systems. *J Environ Qual* **48**, (2019).
105. Efroymson, M. A. Multiple Regression Analysis. *Mathematical Methods for Digital Computers* (1960).
106. Loper, S. *et al.* Organic soil amendment and tillage affect soil quality and plant performance in simulated residential landscapes. *HortScience* (2010).
107. Anderson-Cook, C. M. *et al.* Differentiating Soil Types Using Electromagnetic Conductivity and Crop Yield Maps. *Soil Science Society of America Journal* (2002) doi:10.2136/sssaj2002.1562.
108. Pham, X. & Stack, M. How data analytics is transforming agriculture. *Bus Horiz* (2018) doi:10.1016/j.bushor.2017.09.011.
109. Zilberman, D., Khanna, M. & Lipper, L. Economics of new technologies for sustainable agriculture. *Aust J Agric Resour Econ* (1997) doi:10.1111/1467-8489.00004.
110. Gao, F., Anderson, M., Daughtry, C. & Johnson, D. Assessing the variability of corn and soybean yields in central Iowa using high spatiotemporal resolution multi-satellite imagery. *Remote Sens (Basel)* (2018) doi:10.3390/rs10091489.

111. Daughtry, C. S. T., Gallo, K. P., Goward, S. N., Prince, S. D. & Kustas, W. P. Spectral estimates of absorbed radiation and phytomass production in corn and soybean canopies. *Remote Sens Environ* (1992) doi:10.1016/0034-4257(92)90132-4.
112. Sheets, K. R. & Hendrickx, J. M. H. Noninvasive Soil Water Content Measurement Using Electromagnetic Induction. *Water Resour Res* (1995) doi:10.1029/95WR01949.
113. Farahani, H. J. & Buchleiter, G. W. Temporal stability of soil electrical conductivity in irrigated sandy fields in Colorado. *Transactions of the American Society of Agricultural Engineers* (2004).
114. Breiman, L., Cutler, A., Liaw, A. & Wiener, M. *The randomForest package*. *R Core Team* (2015).
115. Schaettle, K., Ruiz Pestana, L., Head-Gordon, T. & Lammers, L. N. A structural coarse-grained model for clays using simple iterative Boltzmann inversion. *Journal of Chemical Physics* **148**, (2018).
116. Ruiz Pestana, L., Kolluri, K., Head-Gordon, T. & Lammers, L. N. Direct Exchange Mechanism for Interlayer Ions in Non-Swelling Clays. *Environ Sci Technol* **51**, 393–400 (2017).
117. Bassett, W. A. The origin of the vermiculite deposit at Libby, Montana. *The American Minerologist* **44**, 282–299 (1959).
118. Mukai, H. *et al.* Cesium adsorption/desorption behavior of clay minerals considering actual contamination conditions in Fukushima. *Sci Rep* **6**, 21543 (2016).
119. Endo, S. *et al.* Measurement of soil contamination by radionuclides due to the Fukushima Dai-ichi Nuclear Power Plant accident and associated estimated cumulative external dose estimation. *J Environ Radioact* **111**, 18–27 (2012).
120. Koarashi, J. *et al.* Retention of potentially mobile radiocesium in forest surface soils affected by the Fukushima nuclear accident. *Sci Rep* **2**, 1005 (2012).
121. Shiozawa, S. Vertical migration of radiocesium fallout in soil in fukushima. in *Agricultural Implications of the Fukushima Nuclear Accident* vol. 9784431543 49–60 (2013).
122. Hashimoto, S. *et al.* Predicted spatio-temporal dynamics of radiocesium deposited onto forests following the Fukushima nuclear accident. *Sci Rep* **3**, 2564 (2013).
123. Mukai, H., Motai, S., Yaita, T. & Kogure, T. Identification of the actual cesium-adsorbing materials in the contaminated Fukushima soil. *Appl Clay Sci* **121–122**, 188–193 (2016).
124. Okumura, T., Tamura, K., Fujii, E., Yamada, H. & Kogure, T. Direct observation of cesium at the interlayer region in phlogopite mica. *Microscopy* **63**, 65–72 (2014).
125. Fuller, A. J. *et al.* Caesium incorporation and retention in illite interlayers. *Appl Clay Sci* **108**, 128–134 (2015).
126. Unterweger, M. P. Half-life measurements at the National Institute of Standards and Technology. *Applied Radiation and Isotopes* **56**, 125–130 (2002).
127. Ishidera, T., Kurosawa, S., Hayashi, M., Uchikoshi, K. & Beppu, H. Diffusion and retention behaviour of Cs in illite-added compacted montmorillonite. *Clay Miner* **51**, 161–172 (2016).
128. Kogure, T., Morimoto, K., Tamura, K., Sato, H. & Yamagishi, A. XRD and HRTEM Evidence for Fixation of Cesium Ions in Vermiculite Clay. *Chem Lett* **41**, 380–382 (2012).
129. Tamura, K., Kogure, T., Watanabe, Y., Nagai, C. & Yamada, H. Uptake of cesium and strontium ions by artificially altered phlogopite. *Environ Sci Technol* **48**, 5808–5815 (2014).

130. KIKUCHI, R., MUKAI, H., KURAMATA, C. & KOGURE, T. Cs–sorption in weathered biotite from Fukushima granitic soil. *Journal of Mineralogical and Petrological Sciences* **110**, 126–134 (2015).
131. Okumura, M., Nakamura, H. & Machida, M. Mechanism of strong affinity of clay minerals to radioactive cesium: First-principles calculation study for adsorption of cesium at frayed edge sites in muscovite. *J Physical Soc Japan* **82**, (2013).
132. Sawhney, B. L. Selective sorption and fixation of cations by clay minerals. A review. *Clays and Clay Minerals* vol. 20 93–100 Preprint at https://doi.org/10.1346/CCMN.1972.0200208 (1972).
133. Francis, C. W. & Brinkley, F. S. Preferential adsorption of cs-137 to micaceous minerals in contaminated freshwater sediment. *Nature* **260**, 511–513 (1976).
134. N.J. Comans, R., Haller, M. & De Preter, P. Sorption of cesium on illite: Non-equilibrium behaviour and reversibility. *Geochim Cosmochim Acta* **55**, 433–440 (1991).
135. Lai, T. & Mortland, M. Diffusion of ions in bentonite and vermiculite. *Soil Science Society of America* **25**, 353–357 (1961).
136. Liu, X. D. & Lu, X. C. A thermodynamic understanding of clay-swelling inhibition by potassium ions. *Angewandte Chemie - International Edition* **45**, 6300–6303 (2006).
137. Poinssot, C., Baeyens, B. & Bradbury, M. H. Experimental and modelling studies of caesium sorption on illite. *Geochim Cosmochim Acta* **63**, 3217–3227 (1999).
138. Tsai, S. C., Wang, T. H., Li, M. H., Wei, Y. Y. & Teng, S. P. Cesium adsorption and distribution onto crushed granite under different physicochemical conditions. *J Hazard Mater* **161**, 854–861 (2009).
139. Zaunbrecher, L. K., Cygan, R. T. & Elliott, W. C. Molecular models of cesium and rubidium adsorption on weathered micaceous minerals. *Journal of Physical Chemistry A* **119**, 5691–5700 (2015).
140. Lee, J., Park, S. M., Jeon, E. K. & Baek, K. Selective and irreversible adsorption mechanism of cesium on illite. *Applied Geochemistry* (2016) doi:10.1016/j.apgeochem.2017.05.019.
141. Nakao, A., Thiry, Y., Funakawa, S. & Kosaki, T. Characterization of the frayed edge site of micaceous minerals in soil clays influenced by different pedogenetic conditions in Japan and northern Thailand. *Soil Sci Plant Nutr* **54**, 479–489 (2008).
142. Comans, R. N. J. & Hockley, D. E. Kinetics of cesium sorption on illite. *Geochim Cosmochim Acta* **56**, 1157–1164 (1992).
143. Rosso, K. M., Rustad, J. R. & Bylaska, E. J. The Cs/K exchange in muscovite interlayers: An AB initio treatment. *Clays Clay Miner* **49**, 500–513 (2001).
144. Stixrude, L. & Peacor, D. R. First-principles study of illite – smectite and implications for clay mineral systems. *Nature* **420**, 165–168 (2002).
145. Inoue, A. Thermodynamic Study of Na-K-Ca Exchange Reactions in vermiculite. *Clay and Clay Minerals* **32**, 311–319 (1984).
146. Inoue, A. & Minato, H. Ca-K EXCHANGE REACTION AND INTERSTRATIFICATION IN MONTMORILLONITE. *Clays Clay Miner* **27**, 393–401 (1979).
147. Marry, V. *et al.* Salt exclusion in charged porous media: a coarse-graining strategy in the case of montmorillonite clays. *Physical Chemistry Chemical Physics* **11**, 1869 (2009).
148. Liu, C. An ion diffusion model in semi-permeable clay materials. *Environ Sci Technol* **41**, 5403–5409 (2007).

149. Sheng, N. *et al.* Multiscale micromechanical modeling of polymer/clay nanocomposites and the effective clay particle. *Polymer (Guildf)* **45**, 487–506 (2004).
150. Gelineau, P., Stepień, M., Weigand, S., Cauvin, L. & Bédoui, F. Elastic properties prediction of nano-clay reinforced polymer using multi-scale modeling based on a multi-scale characterization. *Mechanics of Materials* **89**, 12–22 (2015).
151. Ebrahimi, D., Pellenq, R. J. M. & Whittle, A. J. Mesoscale simulation of clay aggregate formation and mechanical properties. *Granul Matter* **18**, (2016).
152. Ebrahimi, D., Whittle, A. J. & Pellenq, R. J.-M. Mesoscale properties of clay aggregates from potential of mean force representation of interactions between nanoplatelets. *J. Chem. Phys.* **140**, 154309 (2014).
153. Katti, D. R., Matar, M. I., Katti, K. S. & Amarasinghe, P. M. Multiscale modeling of swelling clays: A computational and experimental approach. *KSCE Journal of Civil Engineering* **13**, 243–255 (2009).
154. Delhorme, M., Labbez, C., Caillet, C. & Thomas, F. Acid-base properties of 2:1 clays. I. modeling the role of electrostatics. *Langmuir* **26**, 9240–9249 (2010).
155. Suter, J. L., Groen, D. & Coveney, P. V. Mechanism of Exfoliation and Prediction of Materials Properties of Clay-Polymer Nanocomposites from Multiscale Modeling. *Nano Lett* **15**, 8108–8113 (2015).
156. Suter, J. L., Groen, D. & Coveney, P. V. Chemically specifi C multiscale modeling of clay-polymer nanocomposites reveals intercalation dynamics, tactoid self-assembly and emergent materials properties. *Advanced Materials* **27**, 966–984 (2015).
157. Kirkwood, J. G. Statistical mechanics of fluid mixtures. *J Chem Phys* **3**, (1935).
158. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* vol. 117 1–19 Preprint at https://doi.org/10.1006/jcph.1995.1039 (1995).
159. Cygan, R. T., Liang, J.-J. & Kalinichev, A. G. Molecular Models of Hydroxide, Oxyhydroxide, and Clay Phases and the Development of a General Force Field. *J Phys Chem B* **108**, 1255–1266 (2004).
160. Bourg, I. C. & Sposito, G. Connecting the molecular scale to the continuum scale for diffusion processes in smectite-rich porous media. *Environ Sci Technol* **44**, 2085–2091 (2010).
161. Ferrage, E. *et al.* Hydration properties and interlayer organization of water and ions in synthetic na-smectite with tetrahedral layer charge. Part 2. Toward a precise coupling between molecular simulations and diffraction data. *Journal of Physical Chemistry C* **115**, 1867–1881 (2011).
162. Marry, V. *et al.* Water dynamics in hectorite clays: Infuence of temperature studied by coupling neutron spin echo and molecular dynamics. *Environ Sci Technol* **45**, 2850–2855 (2011).
163. Henkelman, G. & Jónsson, H. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *Journal of Chemical Physics* **113**, 9978–9985 (2000).
164. Henkelman, G., Uberuaga, B. P. & Jónsson, H. Climbing image nudged elastic band method for finding saddle points and minimum energy paths. *Journal of Chemical Physics* **113**, 9901–9904 (2000).
165. Nakano, A. A space-time-ensemble parallel nudged elastic band algorithm for molecular kinetics simulation. *Comput Phys Commun* **178**, 280–289 (2008).

166. Maras, E., Trushin, O., Stukowski, A., Ala-Nissila, T. & Jónsson, H. Global transition path search for dislocation formation in Ge on Si(001). *Comput Phys Commun* **205**, 13–21 (2016).
167. Hardy, D. J., Stone, J. E. & Schulten, K. Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Comput* **35**, 164–177 (2009).
168. Hardy, D. J. *et al.* Multilevel summation method for electrostatic force evaluation. *J Chem Theory Comput* **11**, 766–779 (2015).
169. Henderson, R. L. A uniqueness theorem for fluid pair correlation functions. *Phys Lett A* **49**, 197–198 (1974).
170. Schommers, W. A pair potential for liquid rubidium from the pair correlation function. *Phys Lett A* **432**, 157–158 (1973).
171. Perakis, F. *et al.* Vibrational Spectroscopy and Dynamics of Water. *Chemical Reviews* vol. 116 7590–7607 Preprint at https://doi.org/10.1021/acs.chemrev.5b00640 (2016).
172. Johnson, M. E., Head-Gordon, T. & Louis, A. A. Representability problems for coarse-grained water potentials. *Journal of Chemical Physics* **126**, (2007).
173. Louis, A. A. Beware of density dependent pair potentials. *Journal Of Physics-Condensed Matter* **14**, 9187–9206 (2002).
174. Salacuse, J., Denton, a. & Egelstaff, P. Finite-size effects in molecular dynamics simulations: Static structure factor and compressibility. I. Theoretical method. *Phys Rev E* **53**, 2382–2389 (1996).
175. Humphrey, W., Dalke, A. & Schulten, K. VMD: Visual molecular dynamics. *J Mol Graph* **14**, 33–38 (1996).
176. Von Reichenbach, H. G. & Rich, C. I. Potassium release from muscovite as influenced by particle size. *Clays Clay Miner* **17**, 23–29 (1969).
177. Nadeau, P. H., Wilson, M. J., McHardy, W. J. & Tait, J. M. Interstratified clays as fundamental particles. *Science* **225**, 923–925 (1984).
178. Wolff, D. & Rudd, W. G. Tabulated potentials in molecular dynamics simulations. *Comput Phys Commun* **120**, 20–32 (1999).