SAND2020-10797R

Surrogate Model Development of Spent Fuel Degradation for Repository Performance Assessment

Spent Fuel and Waste Disposition

Prepared for
U.S. Department of Energy
Spent Fuel and Waste Science Technology

P.E. Mariner,
T.M. Berg,
K.W. Chang,
B.J. Debusschere,
R.C. Leone,
D.T. Seidl

Sandia National Laboratories

September 18, 2020
M3SF-20SN010304044
SAND2020-xxxxx R

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

DISCLAIMER

This is a technical report that does not take into account contractual limitations or obligations under the Standard Contract for Disposal of Spent Nuclear Fuel and/or High-Level Radioactive Waste (Standard Contract) (10 CFR Part 961). For example, under the provisions of the Standard Contract, spent nuclear fuel in multi-assembly canisters is not an acceptable waste form, absent a mutually agreed to contract amendment.

To the extent discussions or recommendations in this report conflict with the provisions of the Standard Contract, the Standard Contract governs the obligations of the parties, and this report in no manner supersedes, overrides, or amends the Standard Contract.

This report reflects technical work which could support future decision making by DOE. No inferences should be drawn from this report regarding future actions by DOE, which are limited both by the terms of the Standard Contract and Congressional appropriations for the Department to fulfill its obligations under the Nuclear Waste Policy Act including licensing and construction of a spent nuclear fuel repository.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

Sandia National Laboratories

Revision 6
10/7/2019

APPENDIX E
NFCSC DOCUMENT COVER SHEET[1]

Name/Title of Deliverable/Milestone/Revision No. Surrogate Model Development of Spent Fuel Degradation for Repository Performance Assessment / M3SF-20SN010304044

Work Package Title and Number GDSA – Framework Development – SNL / SF-20SN01030404

Work Package WBS Number 1.08.01.03.04

Responsible Work Package Manager Paul Mariner
(Name/Signature)

Date Submitted

Quality Rigor Level for Deliverable/Milestone[2]	☐ QRL-1 ☐ Nuclear Data	☐ QRL-2	☒ QRL-3	☐ QRL-4 Lab QA Program[3]

This deliverable was prepared in accordance with Sandia National Laboratories
(Participant/National Laboratory Name)

QA program which meets the requirements of

☒ DOE Order 414.1 ☐ NQA-1 ☐ Other

This Deliverable was subjected to:

☒ Technical Review ☐ Peer Review

Technical Review (TR)
Review Documentation Provided

☐ Signed TR Report or,
☐ Signed TR Concurrence Sheet or,
☐ Signature of TR Reviewer(s) below

Peer Review (PR)
Review Documentation Provided

☐ Signed PR Report or,
☐ Signed PR Concurrence Sheet or,
☐ Signature of PR Reviewer(s) below

Name and Signature of Reviewers

Laura Swiler

NOTE 1: *Appendix E should be filled out and submitted with the deliverable. Or, if the PICS:NE system permits, completely enter all applicable information in the PICS:NE Deliverable Form. The requirement is to ensure that all applicable information is entered either in the PICS:NE system or by using the NFCSC Document Cover Sheet.*

- *In some cases there may be a milestone where an item is being fabricated, maintenance is being performed on a facility, or a document is being issued through a formal document control process where it specifically calls out a formal review of the document. In these cases, documentation (e.g., inspection report, maintenance request, work planning package documentation or the documented review of the issued document through the document control process) of the completion of the activity, along with the Document Cover Sheet, is sufficient to demonstrate achieving the milestone.*

NOTE 2: *If QRL 1, 2, or 3 is not assigned, then the QRL 4 box must be checked, and the work is understood to be performed using laboratory QA requirements. This includes any deliverable developed in conformance with the respective National Laboratory / Participant, DOE or NNSA-approved QA Program.*

NOTE 3: *If the lab has an NQA-1 program and the work to be conducted requires an NQA-1 program, then the QRL-1 box must be checked in the work Package and on the Appendix E cover sheet and the work must be performed in accordance with the Lab's NQA-1 program. The QRL-4 box should not be checked.*

ACKNOWLEDGEMENTS

This report incorporates principal contributions from the coauthors, as highlighted below:

- Tim Berg and Bert Debusschere (3.3 K Nearest Neighbors Regression (kNNr), 4.2 kNNr Surrogate, and 6 Future Work)
- K-Won Chang (5 Performance of Coupled FMD Surrogate Models)
- Rosie Leone (4 Coupling FMD Surrogates to PFLOTRAN)
- Tom Seidl (3.1 Source Data, 3.2 Artificial Neural Network (ANN), 4.1 ANN Surrogate, and verification analyses in 5 Performance of Coupled FMD Surrogate Models)

The authors are grateful to Laura Swiler for her thoughtful technical review, Jim Jerden for his informal review of Section 2, and the staff from U.S. Department of Energy Office of Nuclear Energy (DOE-NE), Prasad Nair (DOE NE-81) and Tim Gunter (DOE NE-81), for their discussions and support of this work.

EXECUTIVE SUMMARY

The Fuel Matrix Degradation (FMD) model calculates spent fuel degradation rates as a function of radiolysis, redox reactions, electrochemical reactions, alteration layer growth, and diffusion of reactants through the alteration layer (Jerden et al. 2015b). It is a complicated model requiring a large number of calculations and iterations at each time step. Because of this, repository simulations, which are already expensive, cannot directly include the FMD process model, especially when hundreds or thousands of waste packages breach.

The FMD surrogate modeling work in this report was initiated based on the hypothesis that surrogate models can be developed from FMD process model training data to inexpensively provide accurate fuel matrix degradation rates in a repository simulation for each individual breached waste package in its own evolving environment at each time step. This report confirms that such surrogate models can be developed. It shows that an artificial neural network (ANN) surrogate and a k-Nearest Neighbors regressor (kNNr) surrogate can emulate the FMD process model with reasonable accuracy. It also demonstrates that these surrogates can run inexpensively in repository simulations for each breached waste package when there are thousands of breached waste packages.

One of the key decisions made during development of the surrogates was to define the applicable input domain for the surrogates. For the FMD process model, there are seven inputs parameters: temperature, burnup, time out of reactor, and the local concentrations of four chemical species, CO_3^{2-}, O_2, Fe^{2+}, and H_2. Two ways to define the surrogate model input domain are (1) to target the ranges of the inputs for which the process model is valid and (2) to target the ranges of the inputs that will be used in the application of the surrogates. The first option provides a broad range of applicability, which can be useful when the actual ranges of applicability cannot be predicted. This benefit can come at a cost to accuracy because it can require more complicated surrogates and substantially more training data. The second option focuses on a subset of the process model input domain, which generally allows for improved surrogate accuracy within the smaller domain. The surrogate models developed in this report employed the first option.

Prior to coupling with PFLOTRAN, the ANN and kNNr surrogates were coded for standalone operation using Python. Training and testing data for the surrogate model input domain were generated using the MATLAB code of the FMD process model. Many of the MATLAB simulations were filtered out of the final surrogate model training and testing dataset due to excessive run time or exceedance of limits on output values. Scatterplots of the remaining training and testing data were then used to reduce the input domains of the surrogate models to ranges that were well-interrogated. From these training data, a two-layer ANN surrogate and a kNNr surrogate that uses 60 nearest neighbors (k=60) were developed.

The accuracies of the standalone surrogate models were evaluated using three standard error metrics: mean-squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). These errors were measured on the testing data to be 3.6×10^{-6} $(mol/m^2/yr)^2$, 9.3×10^{-4} $mol/m^2/yr$, and 31%, respectively, for the ANN surrogate and 5.9×10^{-6} $(mol/m^2/yr)^2$, 1.3×10^{-3} $mol/m^2/yr$, and 78% for the kNNr surrogate. Significant reduction in these errors are expected to be achievable by downsizing the sampling and training space to input ranges expected for a given application and also by generating training data that are more evenly spaced within the input domain. Currently, the FMD MATLAB process model generates time-history training data that are densely packed in time. This is particularly troublesome for the kNNr surrogate and is the main reason a high number of nearest neighbors (60) was needed. Future optimization of the kNNr surrogate might also be achieved by thinning the time histories in the training data set to a smaller number of points in time.

The ANN and kNNr FMD surrogates were successfully implemented in the master branch of PFLOTRAN for general use in repository simulations. Each is demonstrated in this report in two different example problems. The first example problem simulates 52 breached waste packages in a flow field. This

problem was examined because it is the same problem simulated in FY 2015 when a Fortran version of the FMD process model was coupled to PFLOTRAN. The time spent performing waste form calculations using the surrogate models was found to be extremely low compared to the time spent on flow and transport calculations in the same simulations. Time spent on waste form calculations using the coupled the FMD process model in FY 2015 was more than one thousand times greater.

The second demonstration is a full-scale shale repository reference case simulation. Times spent on waste form calculations relative to the total time spent on flow and transport calculations were 0.5% and 0.6% for ANN and kNNr, respectively.

Although there are upfront costs building surrogates, the demonstrations of the ANN and kNNr surrogate models coupled to PFLOTRAN and the standalone error analyses in this report confirm that the ANN and kNNr surrogates can inexpensively emulate the FMD process model in repository simulations for each individual breached waste package at each time step. Having the ability to emulate spent fuel degradation in probabilistic performance assessment simulations allows uncertainties in spent fuel dissolution to be propagated and sensitivities in FMD inputs to be quantified and ranked against other inputs. It is expected that the accuracy of the surrogates, especially for the kNNr surrogate, can be significantly improved in the future by targeting the ranges of application and by more evenly distributing the training data points within those ranges.

This report fulfills the GDSA Framework Development Work Package Level 3 Milestone – *Surrogate Model Development of Spent Fuel Degradation for Repository Performance Assessment*, M3SF-20SN010304044.

CONTENTS

FIGURES

Page

TABLES

NOMENCLATURE

1D	one-dimensional
ANN	artificial neural network
C	Celsius
CSV	comma separated value
CV	cross-validation
DOE	U.S. Department of Energy
DRZ	disturbed rock zone
Eq.	equation
FDR	fractional degradation rate
FMD	Fuel Matrix Degradation
FY	fiscal year
g	gram
GDSA	Geologic Disposal Safety Assessment
GWd	gigawatt day
H	height
HDF5	hierarchical data format, version 5
J	Joule
K	Kelvin
kg	kilogram
kNNr	k Nearest-Neighbors regressor
L	length
LHS	Latin hypercube sampling
M	mass
m	meter
MAE	mean absolute error
MAPE	mean absolute percentage error
mol	mole
MPa	megapascal
MPM	mixed potential model
MSE	mean-squared error
MTHM	metric tons of heavy metal
MWd	megawatt day
NA	not applicable
NMP	noble metal particle
OoR	out of reactor
Pa	Pascal
PA	performance assessment
PFLOTRAN	massively parallel reactive flow and transport model for describing subsurface processes (pflotran.org)
PWR	pressurized water reactor
QoI	quantity of interest
R&D	research and development
ReLU	rectified linear unit
s	second

SFWST	Spent Fuel and Waste Science and Technology
SNL	Sandia National Laboratories
T	time
UO_2	uranium dioxide
W	watt or width
WF	waste form
WP	waste package
wt%	percent by weight
yr	year

(This page is intentionally blank.)

1. INTRODUCTION

In model simulations of deep geologic repositories, UO_2 fuel matrix degradation typically begins as soon as the waste package breaches and groundwater contacts the fuel surface. The initial degradation rate depends on the timing of these events, burnup of the fuel, temperature, and concentrations of dissolved reactants.

Estimating the initial rate of degradation is fairly straightforward, but as UO_2 corrosion products precipitate on the fuel surface and the movement of dissolved species between the fuel surface and environment is impeded by the precipitated solids, the rate is more difficult to quantify. At that point, calculating the degradation rate becomes a reactive-transport problem in which a large number of equations must be solved by iteration for a large number of grid cells at each time step. The consequence is that repository simulations, which are already expensive, become much more expensive, especially when hundreds or thousands of waste packages breach.

The Fuel Matrix Degradation (FMD) model is the process model of the Spent Fuel and Waste Science and Technology (SFWST) campaign of the US Department of Energy (DOE). It calculates spent fuel degradation rates as a function of radiolysis, redox reactions, electrochemical reactions, alteration layer growth, and diffusion of reactants through the alteration layer (Jerden et al. 2015b). Like other similar fuel degradation process models, it is a complicated model requiring a large number of calculations and iterations at each time step.

One way to reduce the cost of repository simulations that include the FMD process model is to assume that all UO_2 in the repository (or certain sections of the repository) degrades at the same FMD process model rate [$M\ L^{-2}\ T^{-1}$], adjusting for the time of waste package breach. This way, the degradation process model only needs to be simulated once (or once for each section of the repository). A major drawback of this approach, however, is that temperature and environmental concentrations of reactants can vary widely across the repository in both space and time. Consequently, this approach will not propagate heterogeneity to fuel matrix degradation rates in the simulation. A more comprehensive discussion of the drawbacks of this homogenizing approach is provided in Mariner et al. (2018, Section 3.2.5).

To include the effects of spatial and temporal heterogeneity in variables influencing fuel matrix degradation rates in repository performance assessment (PA) simulations, a faster calculation of these rates is needed. The work in this report was initiated based on the hypothesis that surrogate models can be developed to inexpensively provide accurate degradation rates in a repository simulation for each individual breached waste package in its own evolving environment at each time step.

1.1 Surrogate Modeling

A surrogate model (sometimes called meta-model, emulator, or response surface model) is an input-to-output mapping that replaces a more complicated simulation code. Once constructed, this meta-model is relatively inexpensive to evaluate so it is often used as a surrogate for the physics model in uncertainty propagation, sensitivity analysis, or optimization problems that may require thousands to millions of function evaluations (Simpson et al. 2008).

There are many different types of surrogate models, including neural networks, regression models, radial basis functions, splines, etc. One popular approach in the literature is to develop an emulator that is a stationary smooth Gaussian process (Santner et al. 2003, Rasmussen and Williams 2006). There are many good overview articles that compare various meta-model strategies. Various smoothing predictors and nonparametric regression approaches are compared elsewhere (Santner et al. 2003, Simpson et al. 2008, Storlie et al. 2009). Simpson et al. (2008) provides an excellent overview not just of various statistical meta-model methods but also approaches that use low-fidelity models as surrogates for high-fidelity models.

Three different types of surrogate models were developed for the FMD process model: a polynomial regression surrogate, an artificial neural network (ANN) surrogate, and a k-Nearest-Neighbors regression (kNNr) surrogate (Mariner et al. 2019a, Appendix A). The polynomial and ANN surrogates are parametric models while the kNNr surrogate is nonparametric. In FY 2019, the ANN surrogate was found to be more accurate than the polynomial surrogate for this application; therefore, only the ANN and kNNr surrogates were pursued in FY 2020 and are presented in this report. The ANN surrogate utilizes a network of artificial neurons with nonlinear activation functions. The kNNr surrogate uses an advanced technique to interpolate between points in a multidimensional lookup table.

1.2 Objectives

This report is a major milestone of the FMD surrogate modeling work that began in FY 2018. Two surrogate models of the FMD process model, an artificial neural network (ANN) surrogate and a k-Nearest-Neighbors regression (kNNr) surrogate, were developed and tested for speed and accuracy and were coupled with PFLOTRAN for use in repository PA simulations.

The objectives of this report are to:

- Review the motivation for developing the FMD surrogate models,
- Describe the development and use of the surrogate models,
- Quantify the accuracy and speed of the surrogate models,
- Describe the coupling of the surrogate models to PFLOTRAN,
- Demonstrate the surrogate models in a repository reference case,
- Discuss lessons learned, and
- Propose future improvements to the surrogate models.

This report fulfills the GDSA Framework Development Work Package Level 3 Milestone – *Surrogate Model Development of Spent Fuel Degradation for Repository Performance Assessment*, M3SF-20SN010304044. This report incorporates information from Mariner et al. (2018) and Mariner et al. (2019a).

2. FMD PROCESS MODEL

The FMD process model calculates spent fuel degradation rates as a function of radiolysis, alteration layer growth, and diffusion of reactants through the alteration layer. This model was developed at Argonne National Laboratory and Pacific Northwest National Laboratory.

Coded in MATLAB, the FMD process model incorporates two general models, a mixed potential model (MPM) and an analytical radiolysis model. The MPM is based on a model developed by (King and Kolar 1999, King and Kolar 2003). It simulates interfacial electrochemical reactions and reactive transport processes between the fuel surface and bulk water.

The FMD process model used in this report (and used to develop the surrogate models) is the version that aligns with version 2 of the MPM (Jerden et al. 2014). Hereafter, this version is called FMD V2. Version 2 of the MPM includes NMP catalysis of redox reactions as an additional process. FMD V2 includes radiolysis. Radiolysis generates aqueous species, e.g., H_2O_2, that further augment fuel oxidation when the fuel surface dose rate is high.

Starting with version 3 of MPM and FMD, steel corrosion was added to provide a source of hydrogen (Jerden et al. 2018). This change is not included FMD V2. Steel corrosion is excluded from this report because the plan for GDSA Framework is to develop separate modules for waste package and waste form degradation and to couple them later via user options in PFLOTRAN.

This chapter describes the features and processes of FMD V2 (Sections 2.1), summarizes important assumptions and ranges of validity (Section 2.2), and identifies requirements for coupling to PFLOTRAN (Section 2.3).

2.1 Features and Processes

The features and processes of FMD V2 are illustrated in Figure 2-1. Features include the following:

- *UO_2 fuel surface*. The UO_2 fuel surface is mostly (≥99%) pure UO_2 (s) and partly (≤1%) regions of noble metal particles (NMPs) (Jerden et al. 2014).
- *Corrosion layer.* The corrosion layer is the accumulation of uranium minerals on the fuel surface resulting from chemical and electrochemical precipitation reactions. It is assumed to have 50% connected porosity and a tortuosity of 0.1 (Jerden et al. 2014).
- *Environmental water*. This nearby water introduces aqueous reactants from the environment and provides a boundary condition for the model. The environmental aqueous reactants in the model are H_2, O_2, CO_3^{2-}, and Fe^{2+}. Each of these species either reacts with radiolytic species or is itself radiolytically active (e.g., CO_3^{2-}). In addition, O_2 is an important chemical oxidant in the model.
- *Interstitial water*. This water fills the void between the fuel surface and the environmental water. Chemical reactions, radiolysis, and diffusion in both directions occur in this water.

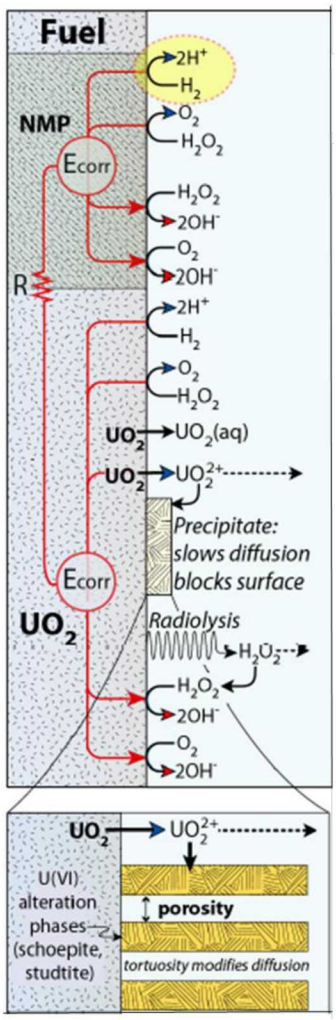

Figure 2-1 Schematic diagram of FMD features and processes implemented in PFLOTRAN and included in FMD surrogate models. Diagram adapted from Jerden et al. (2017).

The model domain depicted Figure 2-2 is one-dimensional (1D). It extends a total of 0.05 m from the fuel surface to the bulk water. This distance is divided into as many as 100 cells with increasing spatial resolution toward the cell boundaries.

The processes in FMD V2 include (Jerden et al. 2014):

- *Production of hydrogen peroxide*. Alpha radiolysis generates H_2O_2 (hydrogen peroxide) near the fuel surface. The amount of H_2O_2 generated each time step is a function of temperature, dose rate, radiolytic G-values, and the initial concentrations of aqueous species. H_2O_2 is the dominant fuel oxidant in anoxic repository environments.
- *Oxidative dissolution of the fuel matrix*. This type of dissolution oxidizes U(IV) at the surface of the UO_2(s) matrix to U(VI), releasing the U(VI) to solution. It is calculated as a function of interfacial redox reaction kinetics (based on corrosion potentials) for both pure UO_2(s) and for a fission product alloy phase referred to as the Noble Metal bearing Particles (NMP) or epsilon phase particles.
- *Chemical dissolution of the fuel matrix*. This type of dissolution releases reduced uranium as U(IV) based on solubility-driven rate calculations at the interface.

- *Other redox reactions at the fuel surface and the NMP surface*. H_2O_2, O_2, and H_2 are kinetically oxidized/reduced at the fuel surface and at NMP phase surfaces.
- *Aqueous redox reactions*. Reduction of H_2O_2, O_2, and U(VI) (aq) and oxidation of H_2 and Fe^{2+} are kinetically controlled.
- *Precipitation (and dissolution) of uranium phases*. Kinetically-driven precipitation (and dissolution) of solid U(IV) and U(VI) phases occur on the fuel surface generating a porous corrosion layer.
- *Complexation by carbonate*. Carbonate from bulk solution reacts kinetically with uranium at the fuel surface and with uranium in the corrosion layer.
- *Diffusion*. Aqueous reactants and products diffuse within the 1D domain. Diffusion is slower through the porosity of the corrosion layer.
- *Temperature dependence*. An Arrhenius temperature dependence applies to all rate constants, mineral saturation concentrations, and diffusion coefficients. A linear temperature dependence applies to standard electrochemical potentials (Jerden et al. 2012).

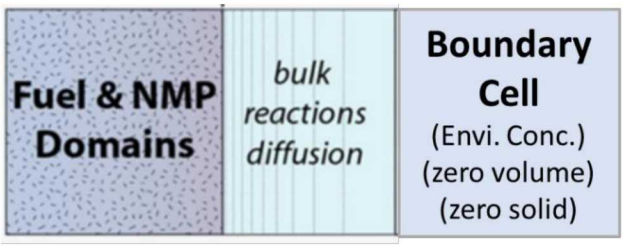

Figure 2-2 FMD process model domain

The user-selected FMD V2 inputs describing fuel and environmental characteristics include:

- UO_2 fuel properties
  - Burnup (GWd/MTHM)
  - Decay time, i.e., the time interval between the end of reactor use and emplacement in the repository
- Environmental concentrations of H_2, O_2, CO_3^{2-}, and Fe^{2+}
- Temperature

The model calculates a dose rate (J/kg) at the fuel surface induced by the radiation field. The dose rate is a function of the fuel properties and relationships presented in Radulescu (2011). The code runs from time zero to 100,000 years with a uniform log10 temporal discretization.

Outputs are calculated at each time step. They include:

- Concentrations of UO_2^{2+}, $UO_2(CO_3)_2^{2-}$, UO_2(aq), U(IV)(s), U(VI)(s), H_2, O_2, H_2O_2 , CO_3^{2-}, and Fe^{2+} for each cell in the 1D domain
- Corrosion layer thickness [L], calculated from the cumulative amount of uranium precipitation (50% porosity assumed)
- UO_2 fuel degradation rate [$M/L^2/T$], calculated from the amount of fuel that reacts during the time step

2.2 Assumptions and Limitations

The FMD process model relies on several assumptions. They include:

- Fuel degradation in this model is exclusively due to the release of UO_2 from the UO_2 fuel matrix.
- Radiolysis effects are dominated by α-dose. This assumption breaks down for out-of-reactor times less than 30 years prior to contact with environmental water (Buck et al. 2013).
- α-particles are assumed to have a penetration distance of 35 μm and a constant energy of 5.3 MeV over that distance (Jerden et al. 2012).
- H_2O_2 (generated by radiolysis) and O_2 (introduced at the environmental boundary and generated by the decomposition of H_2O_2) are the only oxidants in the system.
- The corrosion layer partially reduces the active surface area of the fuel, partially blocks α-particles from generating radiolytic oxidants, and slows diffusion of reactants in the pores of the corrosion layer (Buck et al. 2013).
- Cladding is not included in the model.
- Fuel surface degradation is uniform.

The ranges of input values over which the FMD model may be applied have not been fully explored. The ranges shown in Table 2-1 are theoretically valid. The ranges of concentrations of the environmental species are taken from documented example applications of the FMD process model or from personal communication from Jim Jerden (pers. email to Mariner, May 7, 2020 and June 16, 2020). Nearly all reactions in the FMD process model are kinetically controlled; therefore, combinations of aqueous species concentrations far from equilibrium are possible.

Table 2-1 Theoretical FMD process model input ranges

Parameter	Minimum	Maximum
Initial Temp. (K)	298	473
Burnup (GWd/MTHM)	20	90
Environmental CO_3^{2-} (mol/m^3)	10^{-4} (10^{-7} mol/liter)	10^{2} (10^{-1} mol/liter)
Environmental O_2 (mol/m^3)	10^{-7} (10^{-10} mol/liter)	10^{0} (10^{-3} mol/liter)
Environmental Fe^{2+} (mol/m^3)	10^{-3} (10^{-6} mol/liter)	10^{-2} (10^{-5} mol/liter)
Environmental H_2 (mol/m^3)	10^{-7} (10^{-10} mol/liter)	10^{0} (10^{-3} mol/liter)

2.3 Coupling Requirements

To couple the FMD process model with PFLOTRAN, a "coupled" FMD process model was coded in Fortran. At each time step, PFLOTRAN calls the coupled FMD process model to obtain a new degradation rate. Coupling requires PFLOTRAN to keep track of the 1D chemical profiles across the domain from the previous time step for each breached waste package. Other inputs include temperature, time, and several environmental concentrations in the boundary cell.

The specific requirements for implementation of the FMD process model in GDSA Framework are:

Inputs. The coupled FMD model must read the following for each waste form of each breached waste package at each time step:

- From the coupled FMD process model
    - Fuel burnup (GWd/MTHM)
    - Fuel decay time (time between the end of reactor use and repository emplacement)
    - Fuel specific surface area
    - Environmental concentrations of H_2, O_2, CO_3^{2-}, and SO_4^{2-} near the fuel surface
        - A future option will be to obtain these concentrations from PFLOTRAN when PFLOTRAN is used to simulate reactive transport of these species.
- From PFLOTRAN
    - Time of simulation
    - Time step length
    - Temperature near the fuel surface
    - Concentrations of $UO_2(s)$, $UO_3(s)$, $UO_4(s)$, H_2O_2, UO_2^{2+}, UCO_3^{2-}, UO_2, CO_3^{2-}, O_2, Fe^{2+}, and H_2 in each cell of the 1D process model domain between the fuel surface and the bulk water from the previous time step

Calculations. With these inputs, the coupled FMD process model must calculate at each time step:

- Dose rate for the time step as a function of burnup, decay time, and time of simulation
- Concentrations of $UO_2(s)$, $UO_3(s)$, $UO_4(s)$, H_2O_2, UO_2^{2+}, UCO_3^{2-}, UO_2, CO_3^{2-}, O_2, Fe^{2+}, and H_2 in each cell of the 1D domain at the end of the time step

Outputs. The coupled FMD process model must return to PFLOTRAN at each time step:

- Fuel degradation rate (kg m^{-2} s^{-1})
- Concentrations of $UO_2(s)$, $UO_3(s)$, $UO_4(s)$, H_2O_2, UO_2^{2+}, UCO_3^{2-}, UO_2, CO_3^{2-}, O_2, Fe^{2+}, and H_2 in each cell of the 1D domain so that PFLOTRAN can store them for the next time step

3. FMD SURROGATE MODELS

Three types of surrogate models were developed for the FMD process model – a polynomial regression surrogate, an artificial neural network (ANN) surrogate, and a k-Nearest Neighbors regressor (kNNr) surrogate (Mariner et al. 2019a). Of the two active learners (ANN and polynomial), the ANN surrogate performed better, so development of the polynomial surrogate was discontinued in FY 2020.

This section describes the development of the ANN and kNNr surrogates in standalone mode. Generation of training data for these surrogates is described in Section 3.1. Sections 3.2 and 3.3 describe the development and accuracy of the standalone ANN and kNNr surrogates, respectively. The relative accuracy of these surrogates is compared in Section 3.4.

3.1 Source Data

The first step of this study was to use the standalone MATLAB version of the FMD model from 2018 to generate surrogate build and test data. The temporal discretization in each FMD execution (referred to as a "simulation" or "run") consisted of 101 logarithmically spaced (base 10) points from 0 to 10^5 years. We ran two Latin Hypercube Sampling (LHS) studies of 50,000 and 5,000 runs to create training/validation (referred to as the "training set" for brevity) and test datasets, respectively. LHS is a stratified sampling technique that generates "well-spaced" samples; it typically gives lower variance statistical estimators than plain Monte Carlo sampling (Helton and Davis 2003). The six-dimensional sample space contained the following parameters: initial temperature, burnup, and the environmental concentrations of CO_3^{2-}, O_2, Fe^{2+}, and H_2. The probability distributions for each parameter are given in Table 3-1. The initial temperature (Kelvin) was a prescribed function of time (years) that exponentially decayed from an initial value to 298 K:

$$T(t) = \begin{cases} T_0 - 110e^{-40t^{-0.5}}, & T > 298, \\ 298, & T \leq 298. \end{cases} \quad (1)$$

There were three possible reasons to exclude a simulation run from the training or test sets. First, we removed simulations that failed to finish execution before a cutoff time of five minutes. Typical FMD model runs finish in less than a minute and runs that take much longer than that were often observed to "hang" (i.e., become stuck and never terminate). Second, some simulations produced a corrosion layer thickness that was greater than 5 cm, the size of the computational domain, and were deemed unphysical. The third and final reason for exclusion was "stagnation" or "flat-lining" at late times (see Figure 3-1). This behavior is unexpected and indicative of an issue in the process model (J. Jerden, personal communication). The specific exclusion criterion for stagnation was a change in the value of the quantity of interest (QoI) of less than ten percent between 10^4 and 10^5 years. Roughly half of the training and test sets simulations survived these filtering operations, which is a rather large amount of training data to discard. It is reasonable to assume that the accuracy of the surrogate will be degraded in regions were samples were removed. Quantifying this effect would be difficult, but it is a direction that warrants future investigation.

Figure 3-2 through Figure 3-5 depict pairwise scatter plots and histograms of the input parameter distributions from the training set LHS study that were removed due to excess corrosion layer thickness (Figure 3-2), stagnation (Figure 3-3), excess simulation runtime (Figure 3-4), and the kept runs that made it into the training set (Figure 3-5). There are a few trends that can be observed from these plots. Excess corrosion layer thickness appears to be correlated with the concentration of CO_3^{2-}. High concentrations of H_2 are more likely to produce stagnation but are also more likely to finish before the five-minute cutoff than low concentrations. Lastly, the plot of kept runs shows that there are some gaps in the training set that would likely degrade surrogate accuracy in those regions.

Each kept FMD simulation run contributed 101 points to a surrogate training or test set corresponding to the time trace for the FMD run. The feature space for the surrogate models consisted of temperature, environmental concentrations of CO_3^{2-}, O_2, Fe^{2+}, and H_2, and the dose rate at the fuel surface (a function of burnup and time), and the QoI or target vector was the UO_2 surface flux (also referred to as fuel dissolution rate).

There were a few simulations that stagnated within the last 5 time points which lead to duplicate entries in the training/test sets. These duplicates were dropped from the training set to avoid biasing the surrogate model towards preferentially matching these points, but they were retained in the test set so that entire simulation runs could be compared to surrogate predictions. These duplicate points only consisted of ~ 2% of the training set. The final sizes of the training and test sets were 2889913 and 292496 points, respectively.

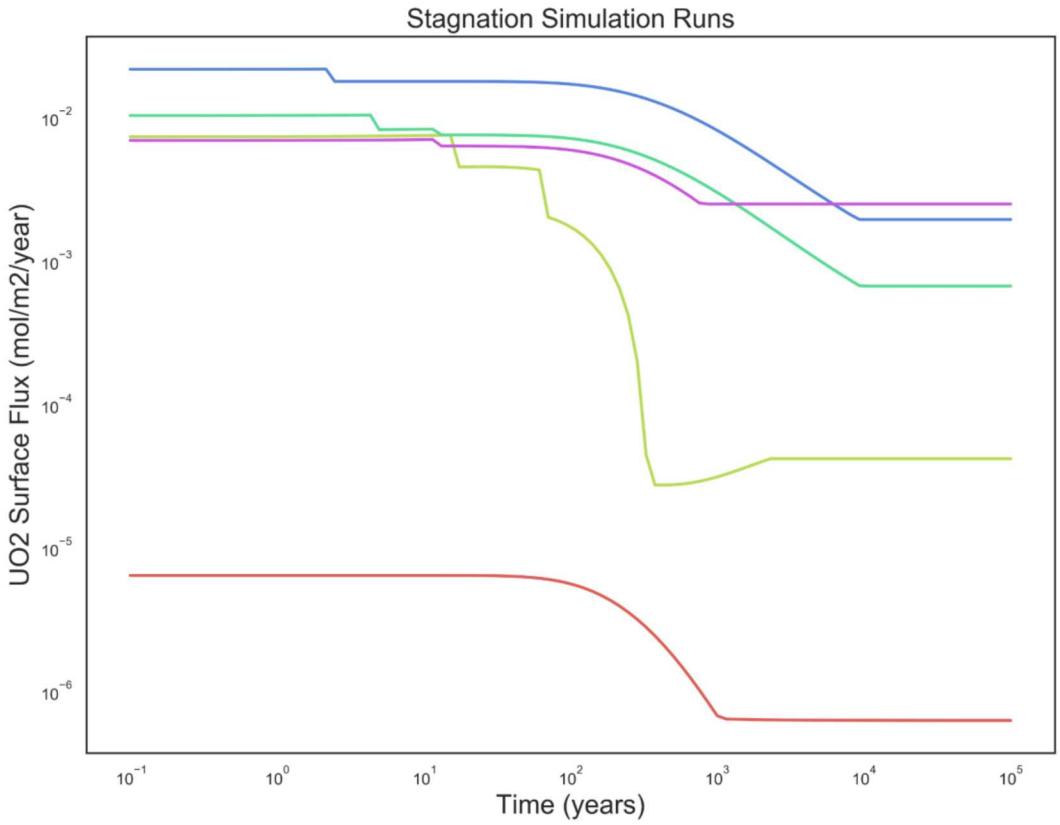

Figure 3-1 Five examples of FMD simulation that displayed late-time stagnation. Runs like these were removed from the surrogate training and test datasets.

We chose to focus on three error metrics in assessing the surrogate model predictions on the training, validation, and test sets: mean-squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). The formulas for these are given below for reference in terms of model predictions h_i and truth values y_i :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(h_i - y_i)^2 \tag{2}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|h_i - y_i| \quad (3)$$

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{h_i - y_i}{y_i}\right| \quad (4)$$

Table 3-1 FMD process model input parameter ranges for surrogate training and test data LHS

Parameter	Minimum	Maximum	Distribution
Initial Temp. (K)	298	393	Uniform
Burnup (GWd/MTHM)	20	70	Uniform
Environmental CO_3^{2-} (mol/m^3)	10^{-4} (10^{-7} mol/liter)	10^{-1} (10^{-4} mol/liter)	Log-uniform
Environmental O_2 (mol/m^3)	10^{-7} (10^{-10} mol/liter)	10^{-3} (10^{-6} mol/liter)	Log-uniform
Environmental Fe^{2+} (mol/m^3)	10^{-3} (10^{-6} mol/liter)	10^{-2} (10^{-5} mol/liter)	Log-uniform
Environmental H_2 (mol/m^3)	10^{-7} (10^{-10} mol/liter)	10^{-1} (10^{-4} mol/liter)	Log-uniform

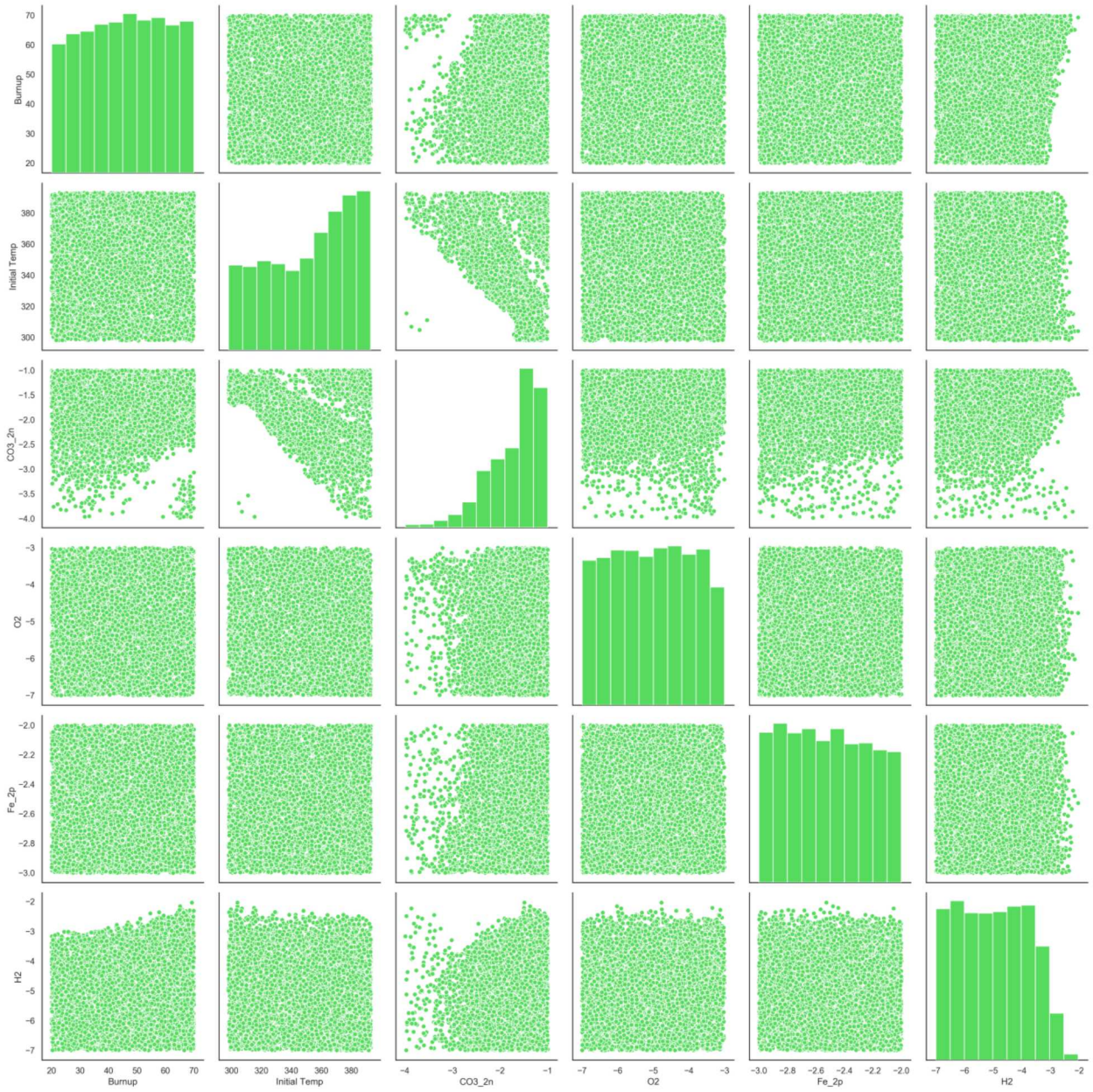

Figure 3-2 Parameter inputs that were removed due to corrosion layer thickness larger than the computational domain size of 5 cm. The concentration of CO_3^{2-} is correlated with the probability of exclusion.

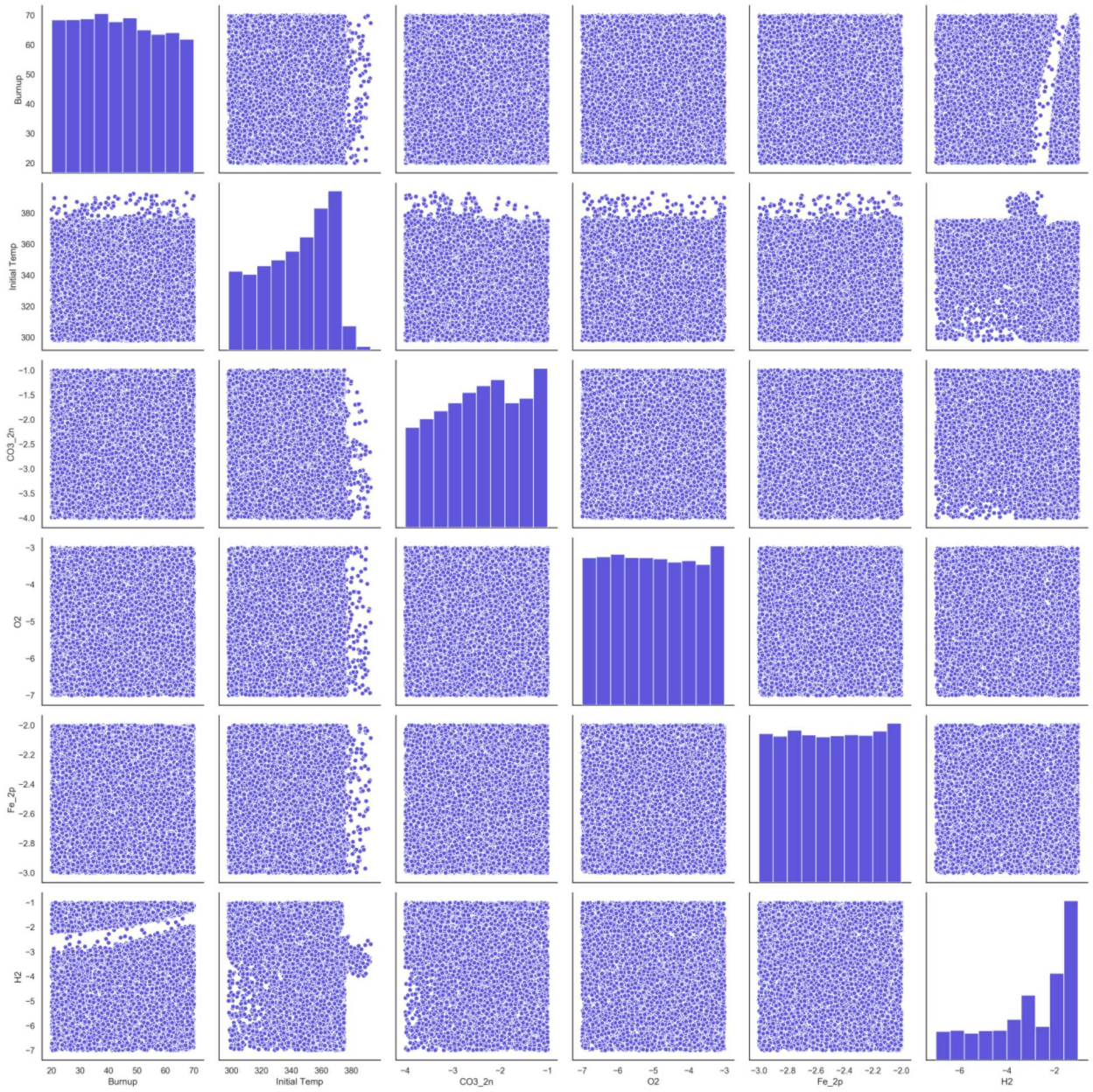

Figure 3-3 **Parameter inputs that were removed due stagnation at times greater than 10,000 years. There is a band in H_2-burnup space where few runs tend to stagnate. Concentrations of H_2 above 10^{-2} are more likely to result in stagnation than lower values.**

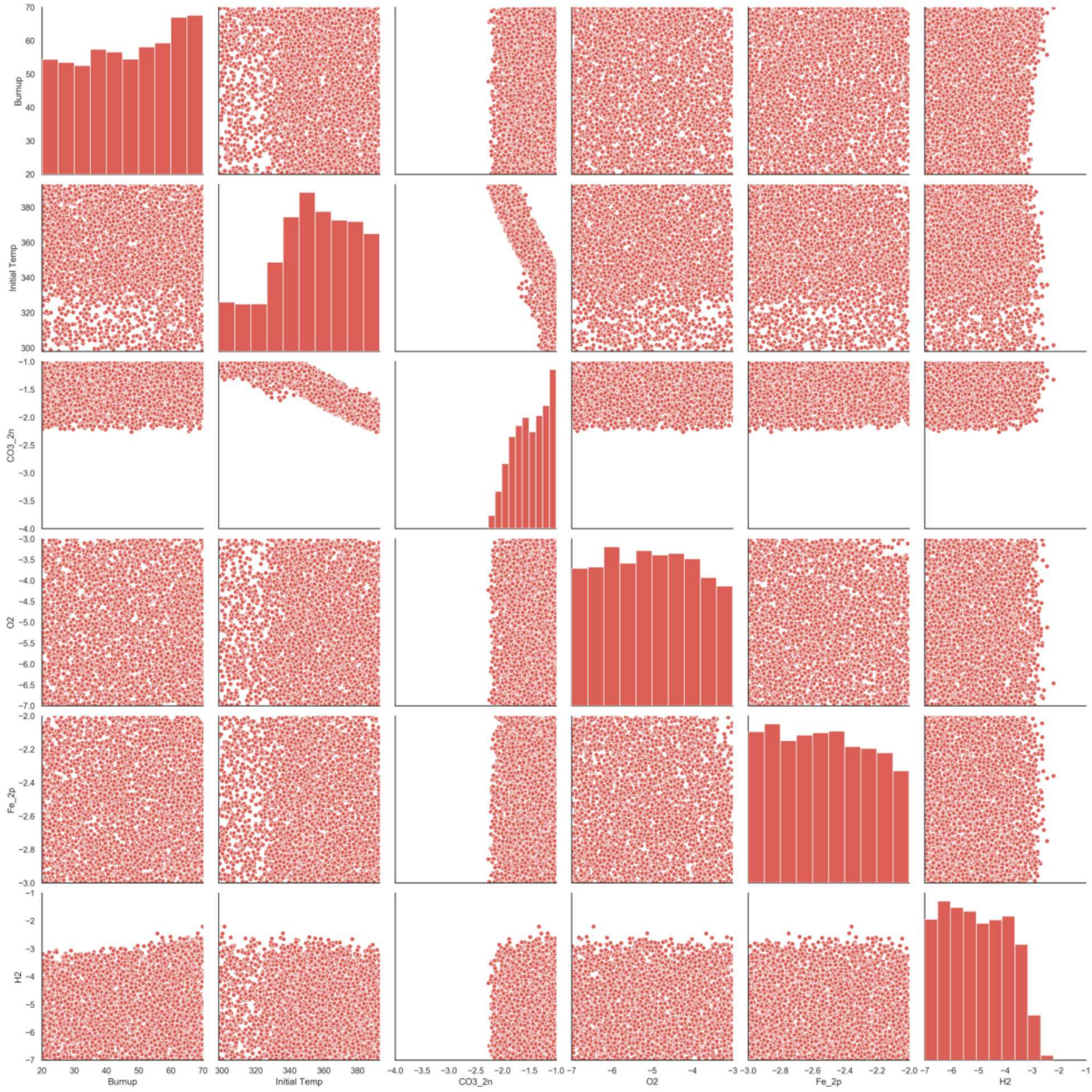

Figure 3-4 **Parameter inputs that were removed from the training set because the simulation failed to finish before a five-minute cutoff intended to filter out "hanging" runs. Simulations with CO_3^{2-} less than $10^{-2.5}$ and H_2, O_2, or Fe^{2+} greater than 10^{-2} all tended to make it past this filter.**

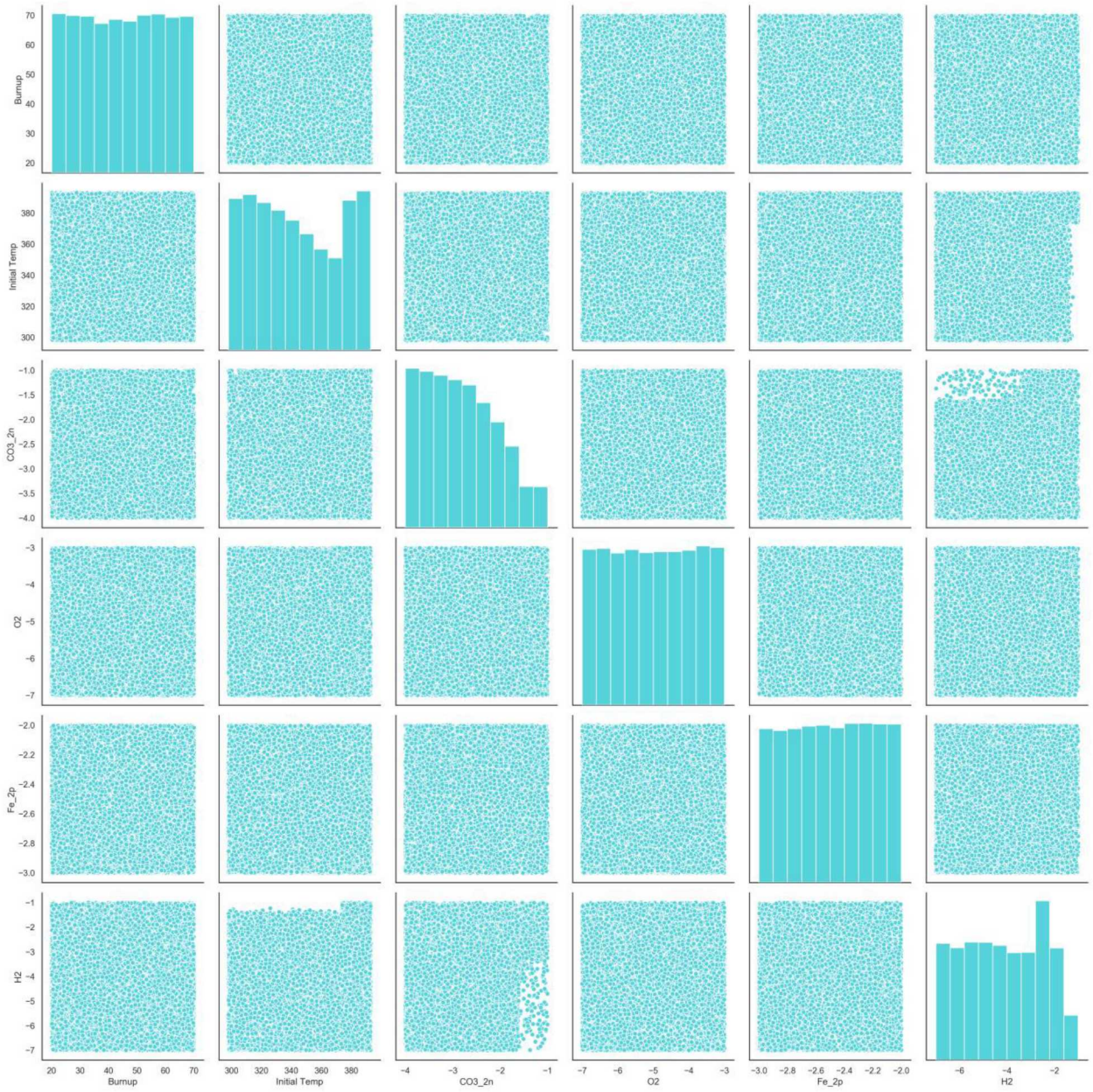

Figure 3-5 Parameter inputs for the training set after all three removal operations. Coverage appears mostly uniform with notable exceptions visible in H_2-CO_3^{2-} and H_2-initial temperature space.

3.2 Artificial Neural Network (ANN)

Neural network models are commonly employed by the machine learning community for regression and classification problems. They can be described as intricate networks of "artificial neurons" that are essentially weighted combinations of (usually simple) nonlinear functions. One motivation for the development of neural networks (Rasmussen and Williams 2006, Pedregosa et al. 2011, Ben-David and Shalev-Shwartz 2014) was to create a regression approach for complex functions that avoids the combinatorial growth of the feature space that occurs in polynomial regression models.

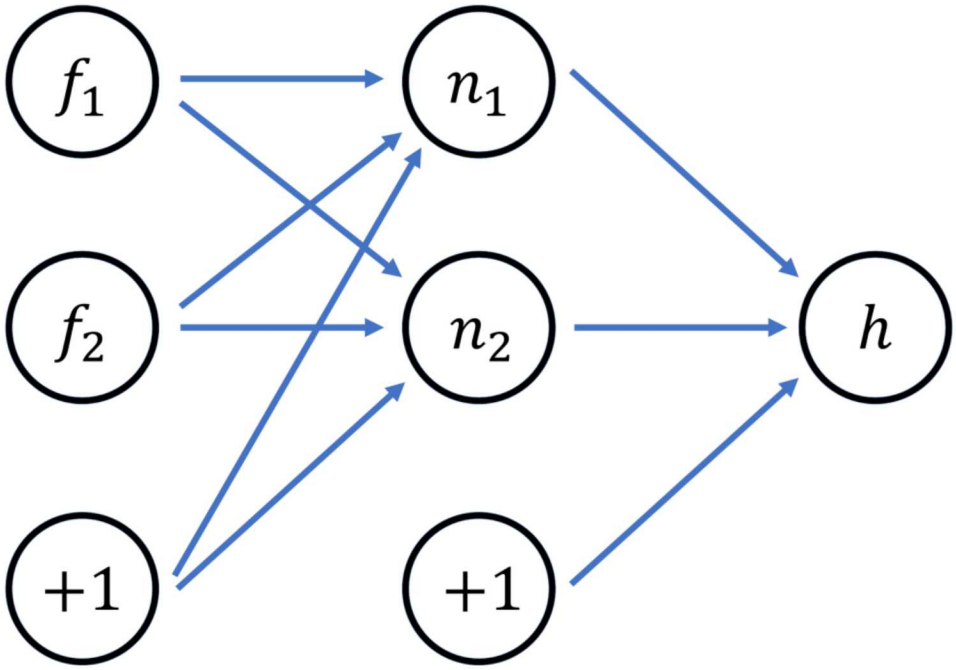

Figure 3-6 A schematic of a single layer feed-forward neural network with 2 input features and 2 neurons in the hidden layer

Figure 3-6 contains a depiction of a single layer feed-forward neural network with two features and a single output. It is called a single layer network because there is one "hidden" layer of neurons between the input and output layers, and the term feed-forward reflects that all the "weights", the directed connections between neurons, are pointing from the input layer to the output layer. The $+1$ nodes denote the "bias" or "offset" terms that are independent of the features f_i and hidden layer neurons. The "depth" of a network is defined as the number of hidden layers. So-called "deep" neural networks contain at least two hidden layers (and often more), while networks that contain only a few layers are described as "shallow."

The inputs to a neuron are scaled by their corresponding weights w_{ij} (which also includes the bias term for convenience), summed, and then fed into a nonlinear "activation function." The index i denotes the layer in the network and j the node or bias term in a given layer. In this work we use the popular rectified linear unit (ReLU) activation function, which is zero for an input less than zero and equal to the input otherwise. The output of each neuron in the final hidden layer is weighted and summed at the output node to produce the model prediction. In regression (as opposed to classification) neural networks there is typically no activation function applied at the output node.

The process of training a neural network involves minimizing a loss function that depends on the weights and biases w_{ij} (blue arrows in Figure 3-6). In this work we utilize a mean absolute error (MAE) loss function:

$$J(w_{ij}) = \frac{1}{N}\sum_{k=1}^{N}\left|h(w_{ij}, x_k) - y_k\right| \tag{5}$$

where N denotes the number of data points in the training set and $h(w_{ij}, x_k)$ is the neural network prediction for features x_k with corresponding QoI value y_k. All of features and QoI were $\log_{10}$-transformed prior to training save for temperature, as they are strictly positive and vary over orders of magnitude. The features were also standardized to have zero mean and unit variance prior to training. MAE was chosen as the loss metric for training because it less sensitive to outliers than mean squared error (MSE). We used the optimization algorithm RMSprop (Tieleman and Hinton 2012) to minimize (6) and terminated the algorithm after the improvement on the training set with additional iterations began to significantly diminish, which occurred around fifty epochs (an epoch is a single pass of RMSprop through the entire training set) as shown in Figure 3-7.

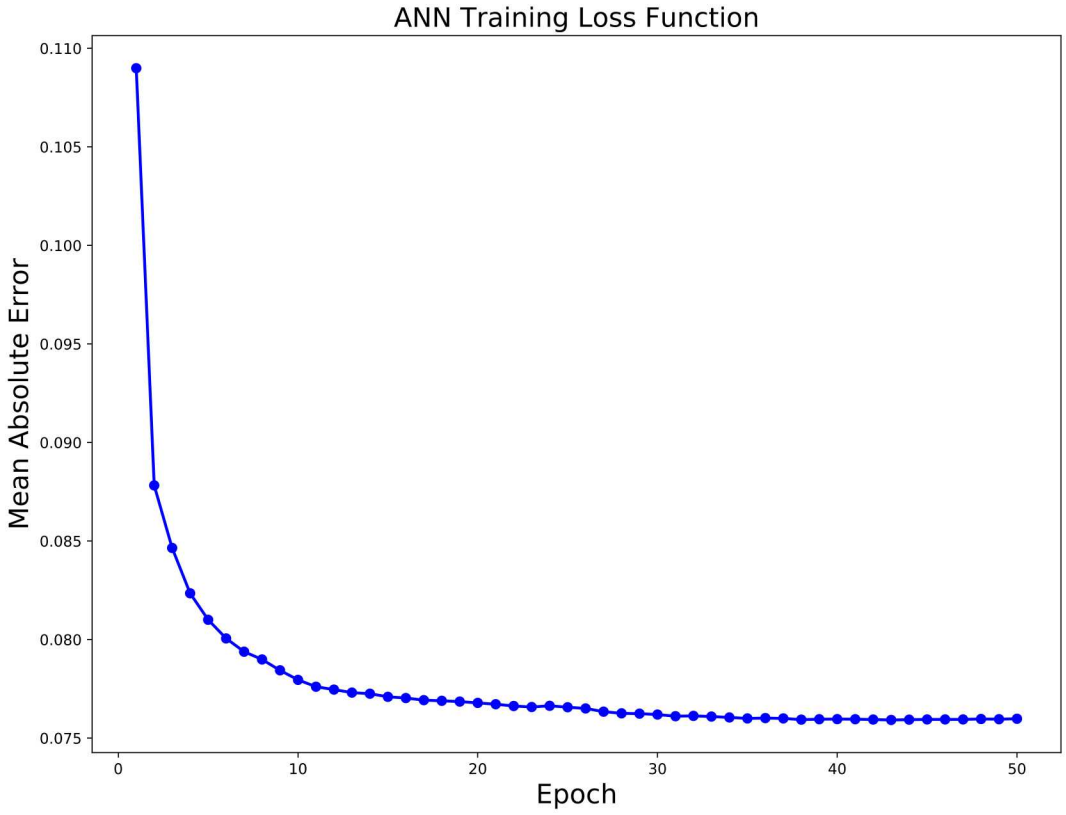

Figure 3-7 Example of convergence of the ANN training optimization algorithm RMSprop. The curve flattens out around 50 epochs.

3.2.1 Development

Although feed-forward neural networks are among the simplest neural network architectures, there are still a multitude of possible model designs. We decided to keep the number of hidden neurons in each layer constant (powers of 2 from 8 to 256) and considered a maximum of eight hidden layers during the model construction phase. Cross-validation (CV) learning curves were used to inform the selection of the network architecture and determine if the amount of training data was adequate for the neural network models.

K-fold cross-validation is a procedure used to assess surrogate accuracy or generalization error using a single dataset (Hastie et al. 2009). K non-overlapping validation data subsets of approximately equal size are randomly selected from a build dataset (i.e., the training set or a fixed subset of it). For each of these partitions, a surrogate model is built using the remaining fraction of the original dataset and its error on the validation set computed. The CV score or error is the average validation error over the K folds. In this work we chose K = 5 and used MAE in the $\log_{10}$-transformed space as the CV error metric (same as the ANN loss function).

Learning curves are plots of an error metric against the size of a training dataset (Murphy 2012). They are used to determine how the amount of training data affects surrogate accuracy. Figure 3-8 and Figure 3-9 depict learning curves for the CV error from neural networks with 64 nodes per layer and varying network depths (Figure 3-8), and two-layer networks with varying amounts of nodes per layer (Figure 3-9). In networks with more than one layer there was little variation in the CV score with the size of the cross-validation set. This observation suggests that the amount of training data was sufficient, and the chances of improving the surrogate by increasing the size of the training set are slim.

The results in Figure 3-8 and Figure 3-9 show that the single layer networks were considerably less accurate than deeper networks. Further, the CV learning curves suggest that a network with two hidden layers and 64 nodes per layer is the smallest network architecture with respect to number of parameters that achieves the minimum CV errors predicted by the best-performing models, so we selected this architecture for the ANN surrogate model. This model contains 4673 parameters (weights and bias terms).

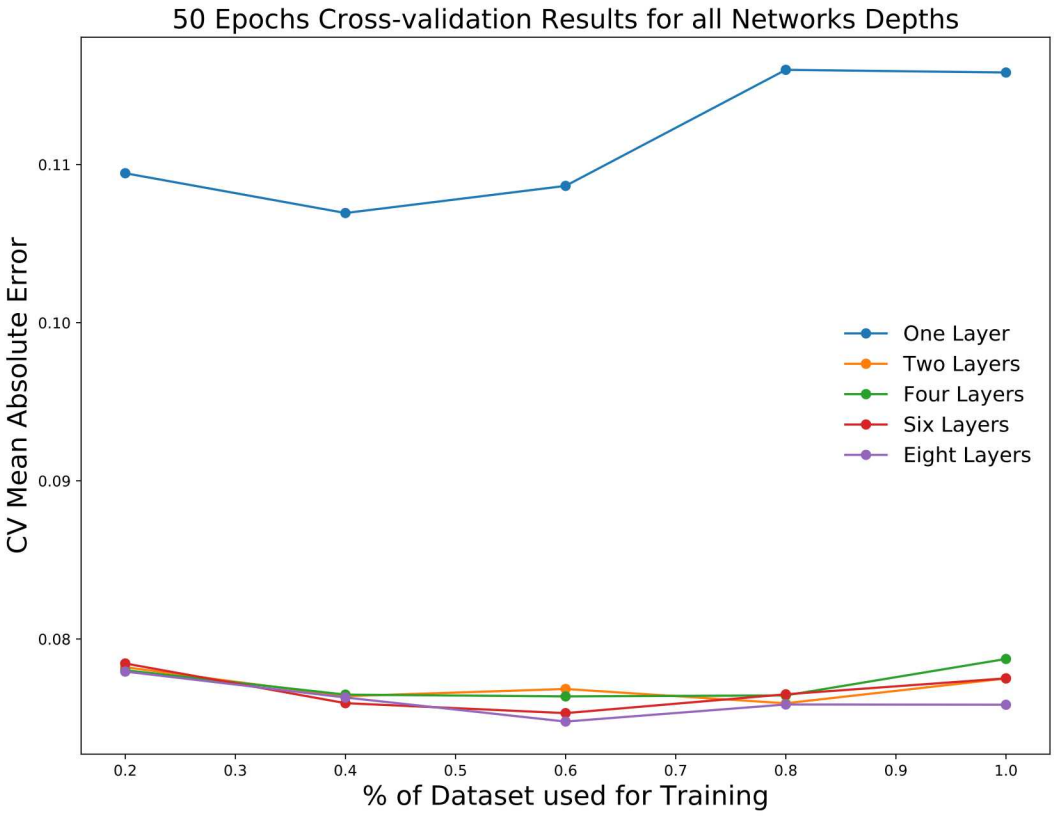

Figure 3-8 There is a significant difference in cross-validation results (mean CV scores) between single and multilayer networks with 64 nodes in each layer. The deep networks (networks with two or more hidden layers), produced similar errors that changed minimally as the size of the training set was increased.

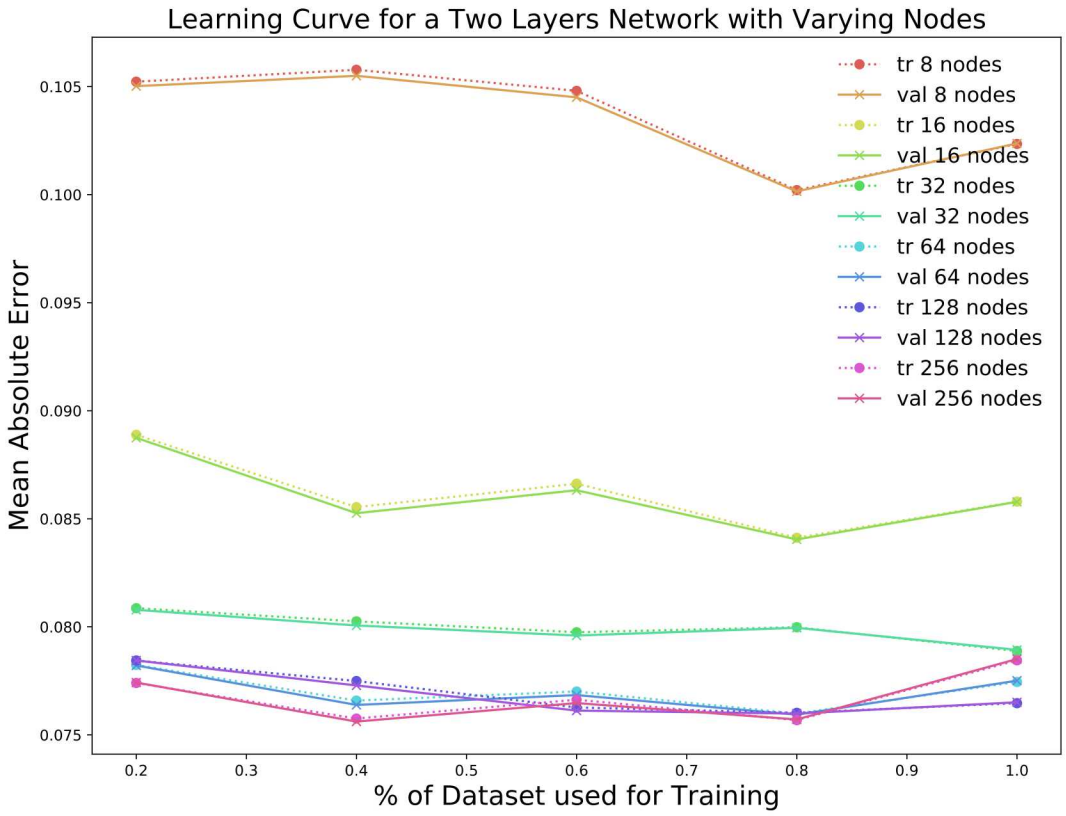

Figure 3-9 Mean CV scores generally drop as the number of nodes in each layer are increased, but returns are greatly diminished after 64 nodes per layer. Similar findings were observed for a network with four hidden layers and varying amounts of nodes per layer.

3.2.2 Accuracy

After determining the architecture of the network, we trained the ANN on the entire training set and computed its predictions on the test set. Figure 3-10 shows one hundred randomly selected surrogate prediction and test data traces from the FMD model, and Figure 3-11 displays the two best, worst, and average predictions with respect to MAPE for a given simulation trace from the test set (i.e., "per-run" MAPE). Figure 3-12 contains histograms of the per-run error metric. Most of the ANN predictions have a per-run MAPE less than 100% although there are some outliers.

One avenue for improvement of the ANN surrogate would be consideration of more complicated network architectures although there is no guarantee that this would be fruitful. Alternatively, diagnosing why the FMD model is stagnating, producing excess corrosion layer thickness, or not finishing in a timely manner, and addressing these issues would be helpful because it would remove the gaps caused by the filtering procedures in the training set which would likely improve prediction accuracy in those regions of feature space.

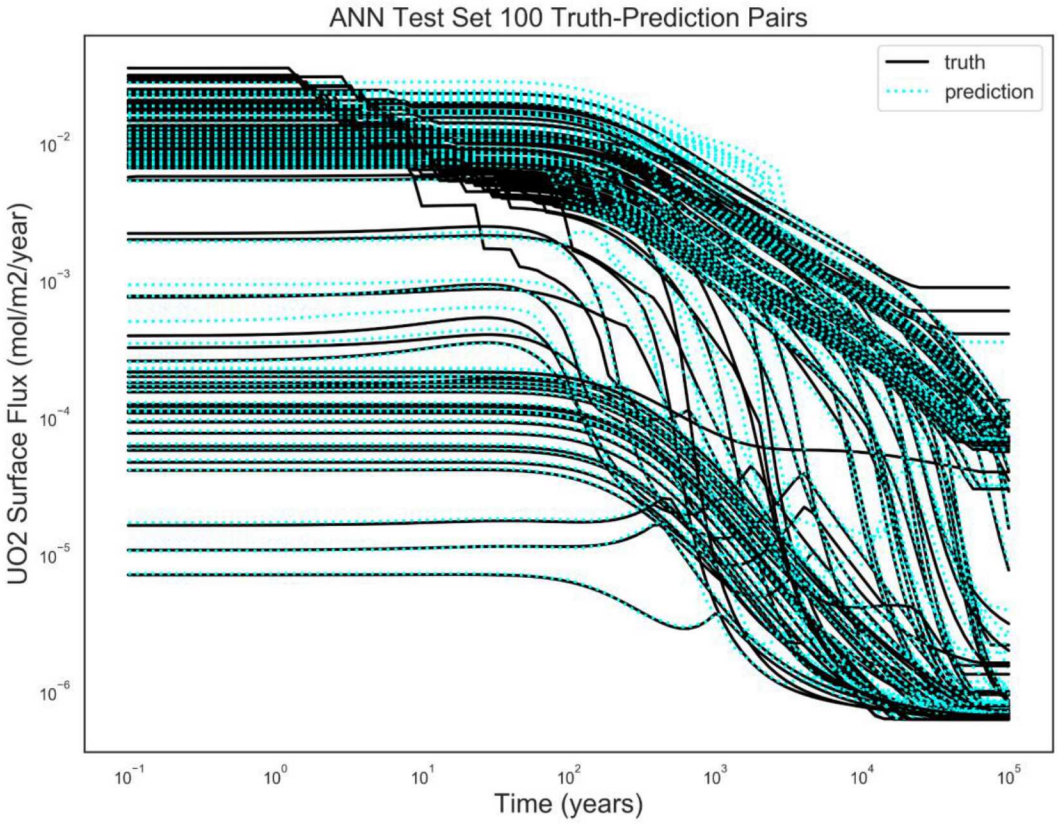

Figure 3-10 One hundred randomly-selected truth-prediction simulation trace pairs from the test set

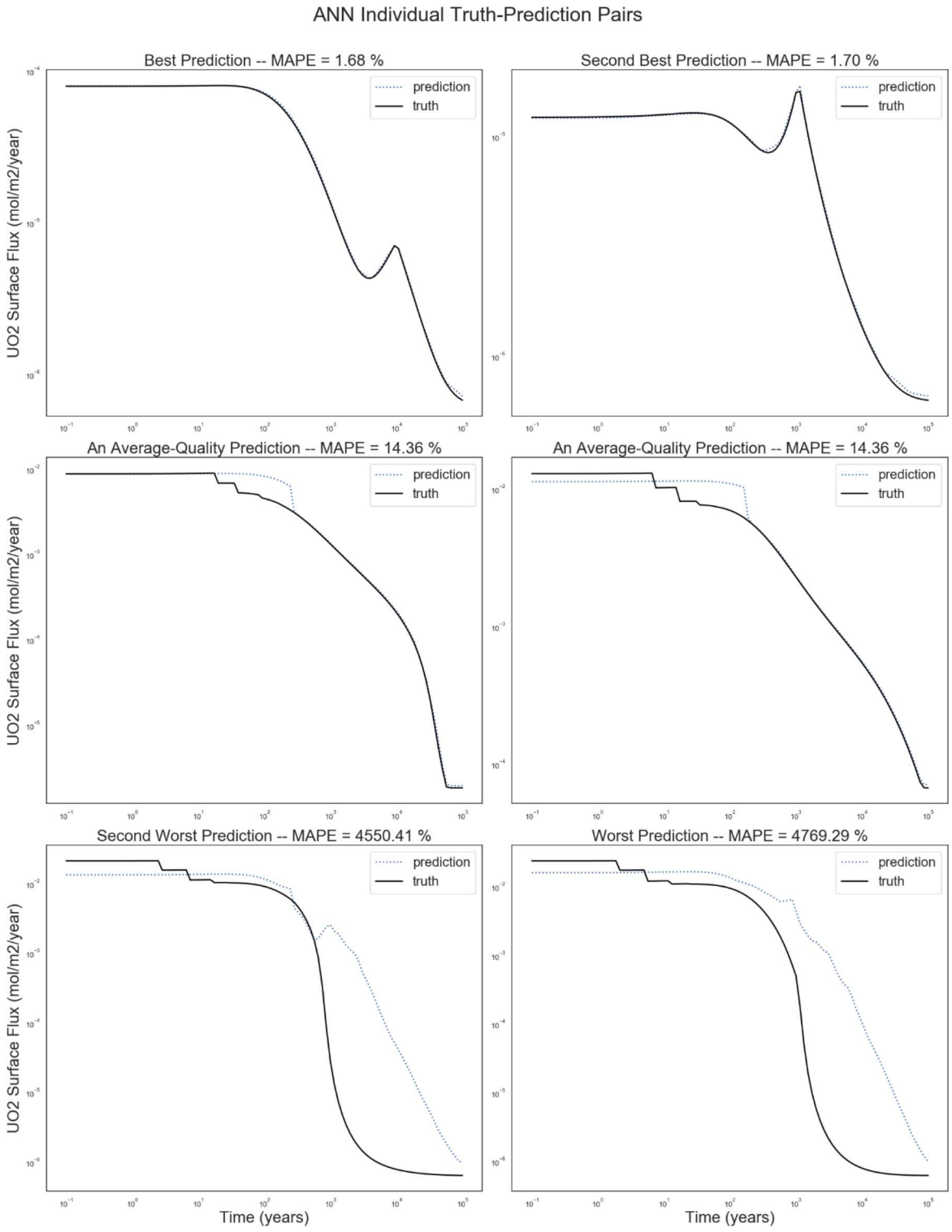

Figure 3-11 Individual truth-prediction pairs for the ANN surrogate

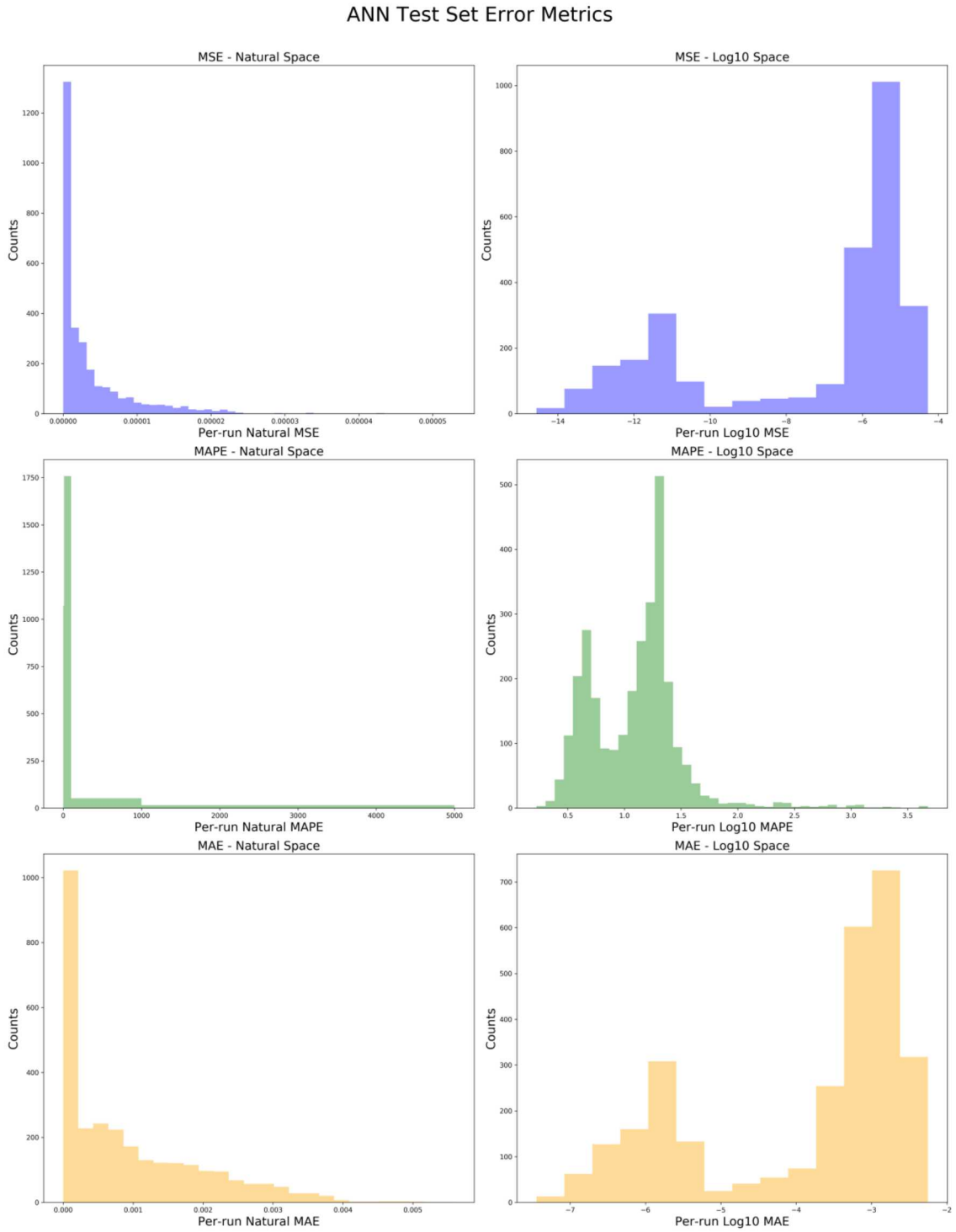

Figure 3-12 Distributions of the per-run (i.e., per FMD simulation trace) error metrics for the test set

3.3 K Nearest Neighbors Regression (kNNr)

This section describes the process of developing and tuning a surrogate model based on the k-nearest neighbor regression (kNNr) algorithm for coupling with the Fortran-based, state-of-the-art massively parallel code called PFLOTRAN in support of nuclear waste repository analysis. The process of coupling the model is described in a subsequent section. Recently, the feasibility of achieving significant speedup with limited loss of accuracy was demonstrated by precomputing a large number of states and implementing a fast surrogate model using a k-nearest neighbors approach (Mariner et al. 2019a). By identifying the nearest points in the precomputed data and interpolating and returning a corresponding uranium oxide degradation rate, the kNNr model functions as a type of high speed look up table and FMD surrogate model. To support production analysis, the kNNr FMD surrogate model must be coupled to the massively parallel subsurface flow and reactive transport code, PFLOTRAN, so that PFLOTRAN can repeatedly interrogate the kNNr surrogate UO_2 values. To effectively balance the need for both speed and accuracy, the kNNr model hyperparameters, such as the amount of training data and the number of nearest neighbors, must be properly tuned. This section addresses the selection and adaptation of a Fortran-based kNNr module, initial hyperparameter tuning studies, and the subsequent successful demonstration of coupling the tunable module with PFLOTRAN.

The kNNr (Ben-David and Shalev-Shwartz 2014) is a supervised, non-parametric machine learning method that, unlike polynomial regression or neural networks, does not re-express the data in any way in order to make predictions. In contrast to the Artificial Neural Network (ANN) approach, which employs active learners, the k-Nearest Neighbors regressor is a lazy learner that tabulates data points inside of a domain X with labels y to the end of using those values for predictions. This makes the kNNr highly interpretable, as no intermediate hypothesis selection process on the parameters is undertaken. Instead, the label for a point within the domain but not in the "table" is obtained as an average of the labels of the k nearest neighbors of this new point, where $k \geq 1$ is fixed (Figure 3-13). The definition of nearest depends on the metric function used, though a typical choice is the Minkowski metric $(\sum_{i=1}^{d}|x_i - y_i|^p)^{\frac{1}{p}}$, with $p \geq 1$. The case of $p = 2$ is the popular Euclidean metric, whereas $p = 1$ gives the Manhattan distance. The tabulation of data points can be implemented with a matrix representing entries in a table. However, this is less efficient than modern tabulation methods like the k-d Tree or the Ball Tree (Pedregosa, Varoquaux et al. 2011). The actual calculation of the predicted value need not be a uniform average. An inverse of the distance to each neighbor may be used to determine how influential that neighbor is in the final calculation of the weighted average.

One of the attractive features of kNNr is that it makes predictions based on local information only, and therefore does not require global smoothness over the input space. On the other hand, the approach requires a sufficiently dense table to get good predictive accuracy, and the cost of table look-ups increases as the table density increases.

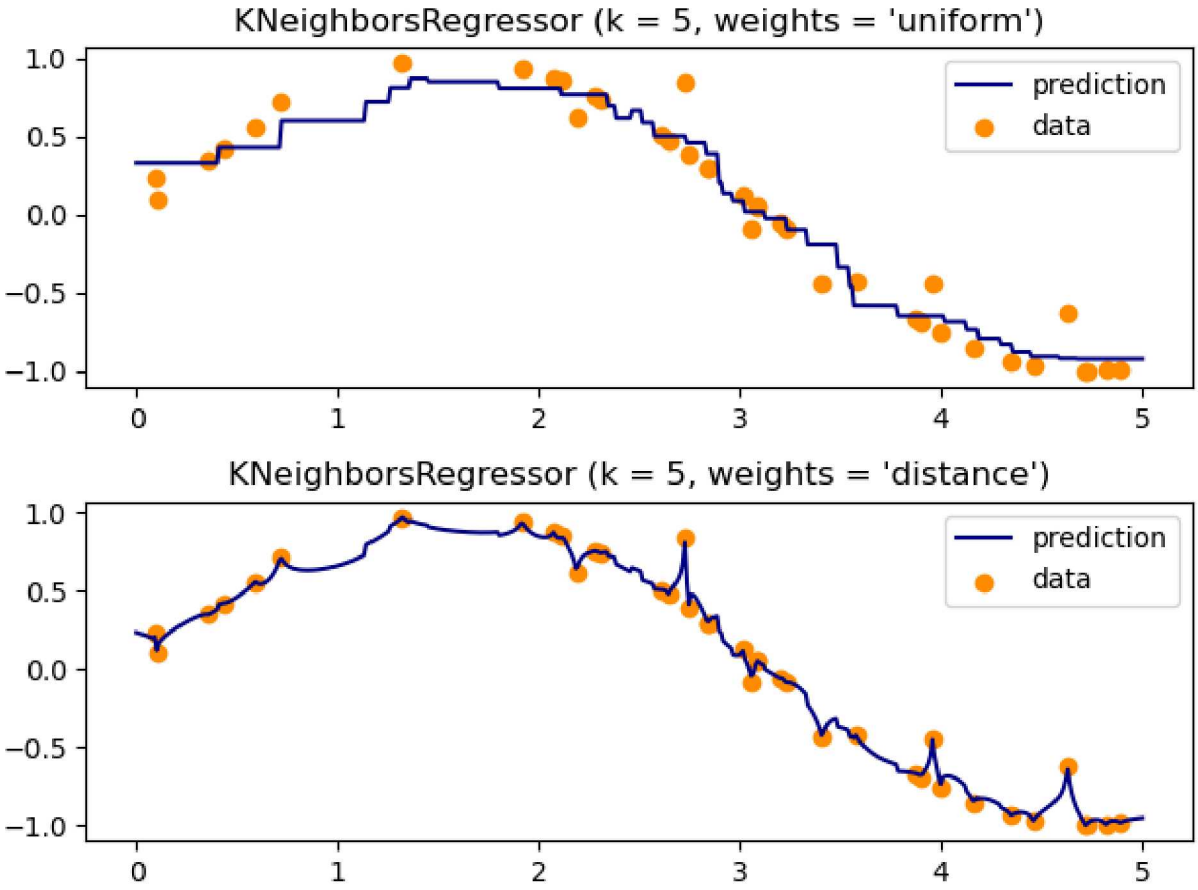

Figure 3-13 While k-nearest neighbors is often thought of as a classification algorithm, it is also used extensively for regression. Predictions of "missing" values are derived by interpolating with a selected number of nearest neighboring data points. In the one-dimensional examples above, the top chart shows the predictions obtained by averaging five nearest neighbors to interpolate missing values. In the lower plot, the interpolation weights the five neighbors based on the inverse of their distance to desired, missing value. In this case, parameter selections appear to overfit the predictions to the data. This image is from the scikit-learn documentation at https://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html

3.3.1 Development and Hyper-Parameter Tuning

To demonstrate the feasibility of coupling kNNr to PFLOTRAN for production computing, an open source, Fortran implementation of kNNr developed for high speed computing was taken through the initial stages of modification, tuning and coupling to PFLOTRAN for FMD surrogate modeling. As detailed below, we employed three versions of kNNr: a Python kNNr version for rapid prototyping of features and parameter tuning, a standalone version of the production Fortran kNNr for feature improvements and verification testing, and the production Fortran version of kNNr coupled to PFLOTRAN. The priority was demonstrating the feasibility of the surrogate model coupling rather than upgrading the Fortran code to include all desired features for optimal accuracy. The hyperparameter tuning effort focused on narrowing the parameter and feature selections to those yielding a balance of speed and accuracy while being implementable under the constraints provided by the production code development process and the limitations of the Fortran kNNr module. A standalone version of the production kNNr was split off to assess expected speed performance in production. With the KDTREE 2

limited in parameter options and the production version difficult to modify and analyze, changes to hyperparameters were made judiciously to respect the project timeline and retain focus on the priority of assessing the feasibility of coupling. The use of the three matching kNNr versions enabled the successful demonstration of coupling with PFLOTRAN for production and the offline development of hyperparameter tunings that balanced both expected production speed with regression accuracy. Verification of the agreement of the Python kNNr and Fortran kNNr was performed by running both with the same datasets and hyperparameters and comparing the resulting predictions. Further work to improve the production algorithm and hyperparameter tuning are discussed in the future work section.

3.3.1.1 Rapid Prototyping in Python with Scikit-Learn kNNr

The rapid prototyping in Python performed last year continued using a corresponding kNNr model implemented with the Python Scikit-Learn module (Pedregosa et al. 2011). Scikit-Learn offers many, built-in features not available in Fortran for tuning hyperparameters and calculating prediction accuracy. The standalone Fortran and Python versions were used in tandem offline with the Fortran version for informing computational and timing costs and the corresponding Python-based kNNr used for broader hyperparameter tuning. Verification testing confirmed that the Fortran and Python versions were functioning compatibly.

3.3.1.2 KDTREE 2 for Production Fortran kNNr

The central challenge was to find or build a Fortran kNNr module and couple it with PFLOTRAN with the lessons learned from the Python based prototype. An open source module called KDTREE 2 (Kennel 2004) offered a Fortran version built for speed with a limited but adequate set of kNNr features for testing the feasibility of the approach. The Fortran KDTREE 2 and corresponding Python Scikit-Learn implementations were made as similar as possible but were not identical. Both did use the same hyperparameters and datasets. The KDTREE 2 library was formulated explicitly for fast execution in eventual, Fortran-based high-performance computing and made available as open source code. More feature rich, k-nearest neighbor implementations exist in other programming languages, but a Fortran version was most desirable for coupling to the PFLOTRAN environment, so no wrapper or other potentially problematic joining code was needed. Developed at the University of California, San Diego Institute for Nonlinear Science, KDTREE 2 is a Fortran95 module written with "considerable care… in the implementation of the search methods, resulting in substantially higher computational efficiency (up to an order of magnitude faster) than the ... previous version." Specifically, "the k-d data structure and search algorithms are the generalization of classical binary search trees to higher dimensional spaces, so that one may locate near neighbors to an example vector in O(logN) time instead of the brute-force O(N) time, with N being the size of the data base" (Kennel 2004). The intended advantages of the code align well with the FMD use cases where the database is large, higher dimensional, and requires larger numbers of nearest neighbors. The author states that "the improvements are the most potent in the more difficult and slowest cases" which is essentially the FMD case. The algorithm is based on Cormen's popular text book (Cormen et al. 2009). The KDTREE 2 Github repository is forked by several developers with the master archived in 2009 here: https://github.com/jmhodges/kdtree2/tree/master/src-f90.

The KDTREE 2 library as available on GitHub does miss some key features. For example, it did not come with any interpolation capabilities. As such, a function to provide inverse distance weighted averaging between the retrieved nearest neighbors was added. The code was also expanded to accept HDF5 binary data files. Further, the KDTREE 2 library only has the Euclidean distance metric available. As will be shown below, kNNr with the Manhattan distance metric performs better than the Euclidean metric. Due to time limitations, the Fortran code has not been enhanced with the Manhattan metric, but this is under consideration for future work.

3.3.1.3 Data Conditioning, Hyperparameter Tuning and Timing Tests

Several, interdependent data conditioning, hyperparameter tuning and evaluation criteria can be considered:

Data conditioning:

- detection of erroneous data and outliers, in particular related to corrosion layer thickness and stalling computations,
- de-duplication of the training points, and
- normalization and/or standardization of the features and targets

Hyperparameter tuning:

- weighting of each nearest neighbor in the interpolation of the prediction
- metric used to determine nearness of a neighbor in multidimensional space, e.g., Manhattan versus Euclidean versus Mahalanobis
- number of nearest neighbors
- size of the kNNr training set (percentage of all data to use in production)
- choice of kNNr sub-algorithm, e.g., k-d Tree versus Ball Tree, as well as options such as tree sorting.

Evaluation criteria:

- accuracy of the resulting table on the test data set
- time required to load the training set and
- time taken to calculate a prediction.

Up front, we decided to use inverse-distance weighted interpolation between the retrieved nearest neighbors for a query vector. Contrary to uniformly-weighted interpolation, inverse-distance weighted interpolation has the nice option that the kNNr prediction error is zero for points that are already in the database. In other words, the error on the training data is zero. Further, having the influence of a point decrease if it is further away from the query vector guards against severe accuracy degradation if the table lookup returns points that are very far away. Also, the choice of data partitioning algorithm (k-d Tree versus Ball Tree) had no impact on the accuracy results using the Python code. As such, all results shown below used the k-d Tree algorithm, which is the one provided by the Fortran KDTREE 2 code.

3.3.1.4 Data Conditioning: Data pruning and Log10 Transformation

As discussed in section 3.1 concerning source data, one step in the data conditioning was the removal of runs with unphysical corrosion layer thicknesses. Beyond that, we also pruned duplicates from the feature set for three reasons: First, we discovered that duplicates in the feature set cause KDTREE 2 to crash, and second, from an information theoretical point of view, duplicates do not provide any new information, and can bias the dataset by giving more weight to some feature combinations. Third, if there are duplicates between the training, validation, and testing data set, it will skew the accuracy measurements because kNNr with the inverse-distance weighted interpolation has no prediction errors for validation or testing points that are also in the training set. As such, the kNNr starts with the same set of candidate 2,889,913 training points as the ANN, with each training point being the system state at a time point along the FMD MATLAB model trajectory.

In terms of data transformations, recall that the kNNr works by finding the nearest training points to the query vector in six-dimensional space. Multidimensional distance calculations must account for the features having different scales as discussed and also different distribution functions. Different scale features result in some features swamping the distance calculation, but this can be remedied by

transforming all features to a common scale. Tightly clustered features with non-normal distributions result in limited discrimination between neighbor distances. Mariner et al. (2019a) demonstrated that taking the log10 transform of the features improves the outcome. It is also helpful to log10 transform the quantities of interest and perform the prediction and interpolation in the transformed space as shown in Table 3-2:

Table 3-2 Mean Absolute Percentage Error (MAPE) is smaller if the QoI are log10 transformed

250K Training Points / 60 Nearest Neighbors	MAPE	MAE	MSE
Fortran - log10(QoI)	78%	1.26e-03	5.95e-06
Fortran - No log10	593%	1.68e-03	1.12e-05

3.3.1.5 Hyperparameter Tuning: Nearness Metric, Number of Neighbors and Training Dataset Size

To determine the best combination of hyperparameters and to average over the noise induced by selecting random subsets of data, ensembles of tests were run varying the neighbor nearness metric, number of nearest neighbors, and amount of training data. First, the available data was split into training and test data with the test data comprising approximately ten percent of all the data held in reserve for final accuracy testing of the selected hyperparameter combinations. The remaining 90% of the data was used for parameter studies. Note that it is not always a good idea to use all available training data as the training model in kNNr since using smaller training sets can result in faster performance. Balancing the potential accuracy of using large amounts of training data versus the speed advantages of smaller sets is part of the hyperparameter tuning. Since the size of the training data set impacts the accuracy of different numbers of nearest neighbors, they were addressed simultaneously. To balance the probabilistic effects of choosing training subsets at random (one might test really well and another poorly), ensembles were used and their statistics calculated. For each ensemble, a set of complete runs were set aside for validation and then randomly selected sets of training samples were selected without regard to which run they originated from. This approach should result in training sets best covering the dataspace rather than being confined to specific run data only. Each of these training sets constituted a model to which the features of the validation data were fed to make predictions. The predictions were then compared with the true values of the validation data QoI or target values to compute pointwise, run averaged, and ensemble errors along with their statistics. The results were plotted as function of the distance metric and number of nearest neighbors in Figure 3-14 and Figure 3-15. These plots then informed decisions about hyperparameter choices for the coupled production runs.

First, note that consistent with Mariner et al. (2019a), both Figure 3-14 and Figure 3-15 show smaller errors when using the Manhattan distance metric over the Euclidean metric, especially for the MAPE errors. However, this effect is not as large as others and can be offset somewhat by data conditioning. In multidimensional datasets, the relative sizes of the features and their distributions must be accounted for. If not, larger scale features will dominate distance calculations and these effects are amplified when the Euclidean metric is used because it squares the feature by feature distance elements. Many other potential distance metrics are possible and could be explored also. The effects of metric choices can also be inhibited or accentuated by the way the data is conditioned such as through the log10 transformation. KDTREE 2 currently only offers the Euclidean metric and adding the capability for Manhattan distance would be a welcome modification as potential future work.

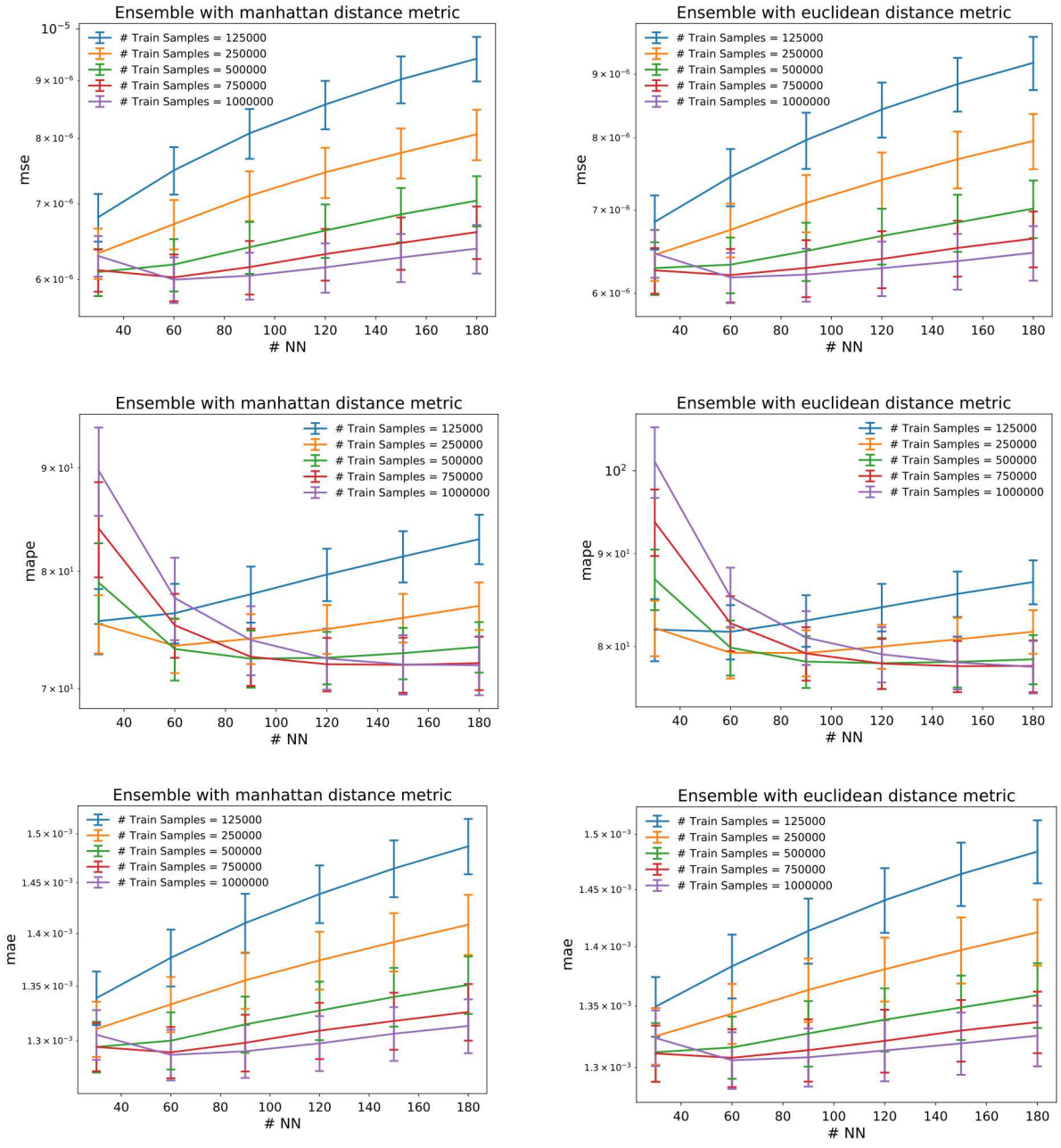

Figure 3-14 MSE, MAPE, and MAE errors plotted as a function of the number of nearest neighbors for ensembles of ten randomly selected training datasets varying in size. Plots show that errors are smaller with the use of the Manhattan distance metric over the Euclidean distance metric. The impact of the number of nearest neighbors and training size sets on errors is more complex and depends on the size of the training data set.

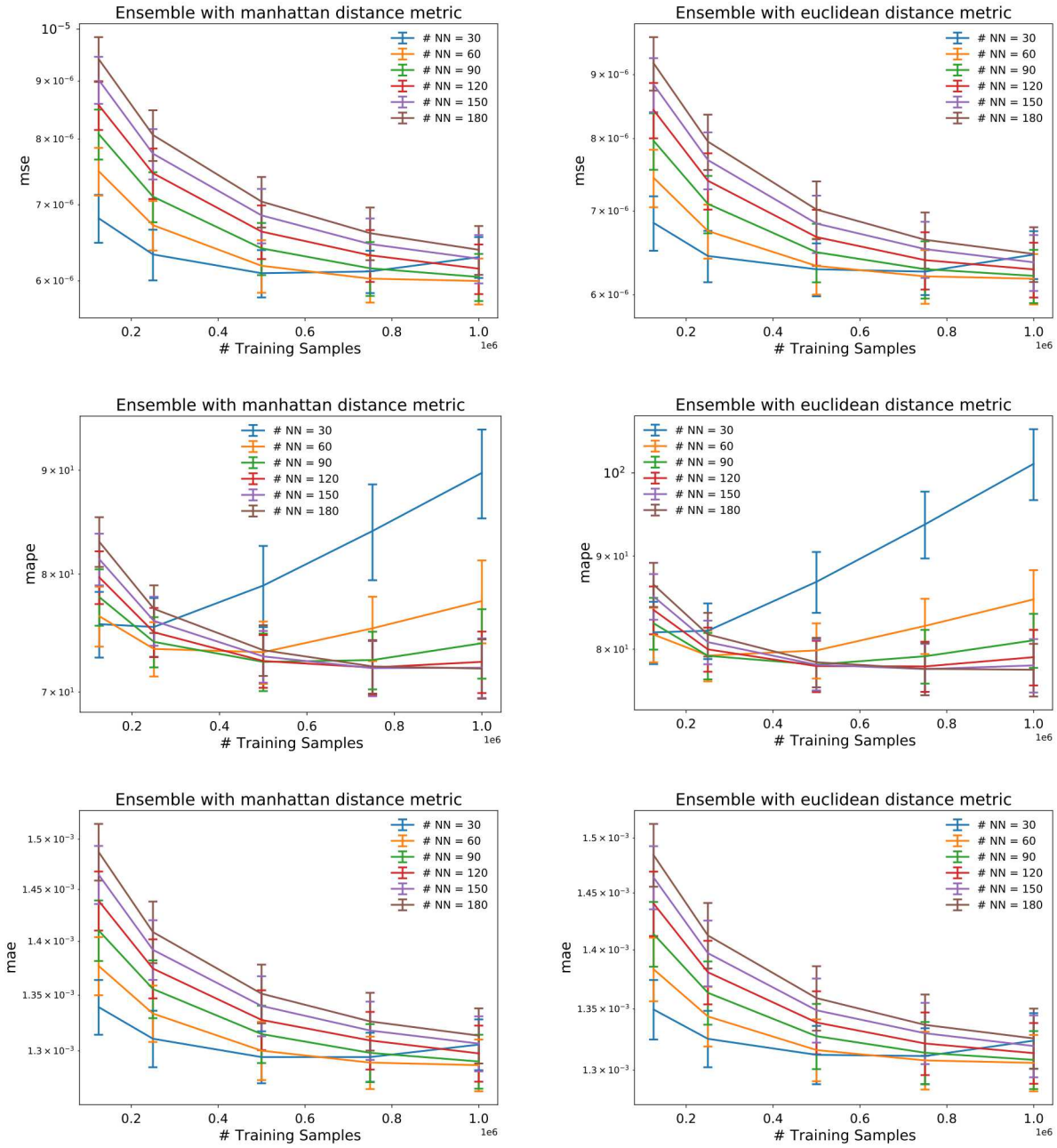

Figure 3-15 MSE, MAPE, and MAE errors plotted as a function of the number of training samples for select numbers of nearest neighbors used. Initially the errors decrease as more data points are added, but eventually the errors go back up, as is particularly evident in the MAPE metric.

The behavior of the kNNr prediction errors with respect to the number of training samples and the number of nearest neighbors used is somewhat counterintuitive. One would expect that increasing the number of training points would always reduce the error. However, as Figure 3-14 and Figure 3-15 show in the current case, more nearest neighbors are needed to get good accuracy as the number of training points increases. Our hypothesis is that this is connected to the nature of how the training data set was built. As explained in Section 3.1, the training data was sampled from time-trajectories of standalone fuel cask simulations for specified environmental conditions. As such, rather than having sample points that are fully randomly spread over the 6-dimensional feature space, we have clusters of data points, since the only features that vary within each of the standalone simulations are temperature and dose rate. As we increase the number of samples, more and more points from the same standalone simulations are retained in the training set, requiring more nearest neighbors to interpolate between in order to have sufficient amounts of data from runs with different environmental concentrations. This sampling approach will be revisited under future work.

Using the existing training data set, two configurations were chosen and tested in ensembles against the test dataset previously held in reserve and the standalone Fortran version was set up with timers to determine which configuration to use in production. Note that the optimal configuration (in terms of accuracy) depends on the error metric used. Here, we based the meta-parameter selection on the MAPE metric, as this metric weighs the error on the prediction of small fluxes more equally to the errors in large fluxes. Based on the MAPE plots in Figure 3-15, there appears to be a balanced set of parameter choices providing low errors at 250,000 training samples with 60 nearest neighbors. The smaller training set and relatively few nearest neighbors provide computational speed advantages in terms of being quicker to load, sort and interrogate while offering workably low prediction errors based on the ensemble testing. Similarly, the pairing of 500,000 training points with 120 nearest neighbors approaches the error asymptote seen with the larger training point and nearest neighbor combinations. The error statistics and timing data based on the reserve test data for these two configurations are shown in Table 3-3. Based on these results, the larger dataset delivers small improvements but at the cost of much more execution time, as illustrated in Figure 3-16. As such, a randomly chosen set of 250,000 training points, approximately ten percent of the original training set, with 60 nearest neighbors was used for the PFLOTRAN coupled demonstrations.

Note, as mentioned earlier, the production version of KDTREE 2 was modified to read in training data in HDF5 format. The binary training data file loads in 0.0243 seconds compared to the text-based comma separated value (CSV) file, which requires 128 times longer to load in 3.12 seconds. The production version was programmed to load the tree once, keep it in memory and query it repeatedly for each target value.

Table 3-3 Expected errors on the Test set of the two chosen kNNr configurations and their standard deviation over a set of 10 random replicas drawn from the training data. The larger dataset results in slightly better accuracy but with a significant time penalty when evaluated over the test set.

Configuration	Test MSE	Test MAE	Test MAPE [%]	Time (ms)
250K / 60NN	(6.11 ± 0.05) e-6	(1.269 ± 0.002) e-3	79.1 ± .5 %	64
500K / 120NN	(5.80 ± 0.02) e-6	(1.2487 ± 0.0009) e-3	79.2 ± .6 %	188
Percent Difference	-5.07%	-1.60%	0.126%	194%

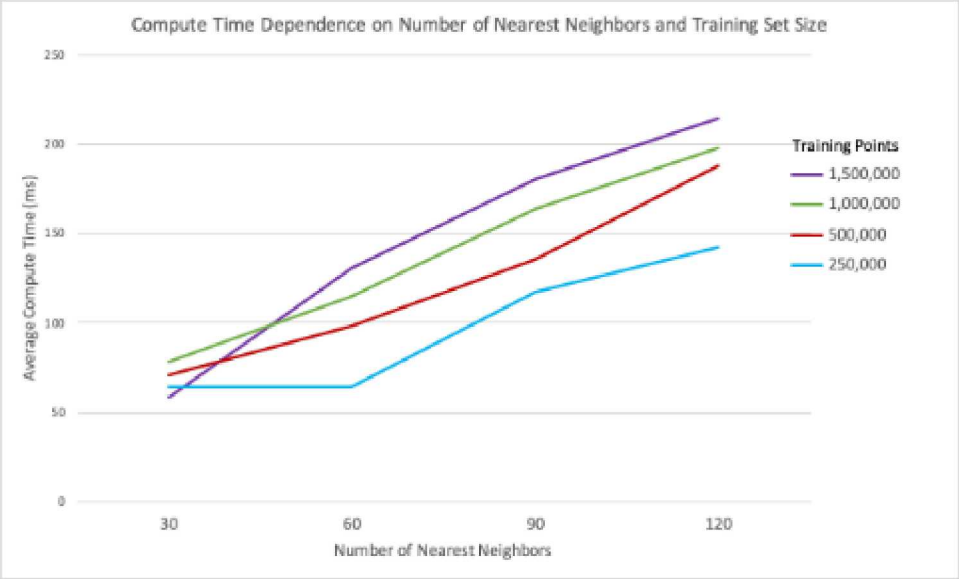

Figure 3-16 Compute time for evaluating the test set as a function of the number of nearest neighbors used and the amount of training data. The configuration with 500,000 training points and 120 nearest neighbors is almost 3 times as expensive as 250,000 training points and 60 nearest neighbors.

To verify that the Python and Fortran tabulation provided equivalent results, we used the same randomly selected training set to predict the test set with both the Python and the Fortran KDTREE 2 standalone codes. While the Python and Fortran results were not identical, they did match closely as shown in Table 3-4, with the differences between the two being much smaller than the errors with respect to the test set. Figure 3-17 gives six representative samples comparing the Python and Fortran results along with the true values. The two versions do not return identical predictions, but the differences are an order of magnitude or smaller than the errors with respect to the true trajectories.

Table 3-4 Verification testing to ensure the Fortran code is equivalent to the Python prototype

250K Training Points / 60 Nearest Neighbors	MAPE	MAE	MSE
Python	79%	1.27e-03	6.13e-06
Fortran	78%	1.26e-03	5.95e-06
Difference	-0.60%	-0.88%	-2.92%

The fact that the Python and Fortran kNNr implementations do not give identical results is not surprising given that they are totally independent implementations of the k-d Tree data partitioning and lookup approaches, as well as interpolation routines. However, as the differences between them is much smaller than their prediction errors, we feel confident that the Python code can be used as a flexible prototyping tool for hyperparameter tuning. Once the optimal configuration was determined, the Fortran code was used for all accuracy tests covered in the next section.

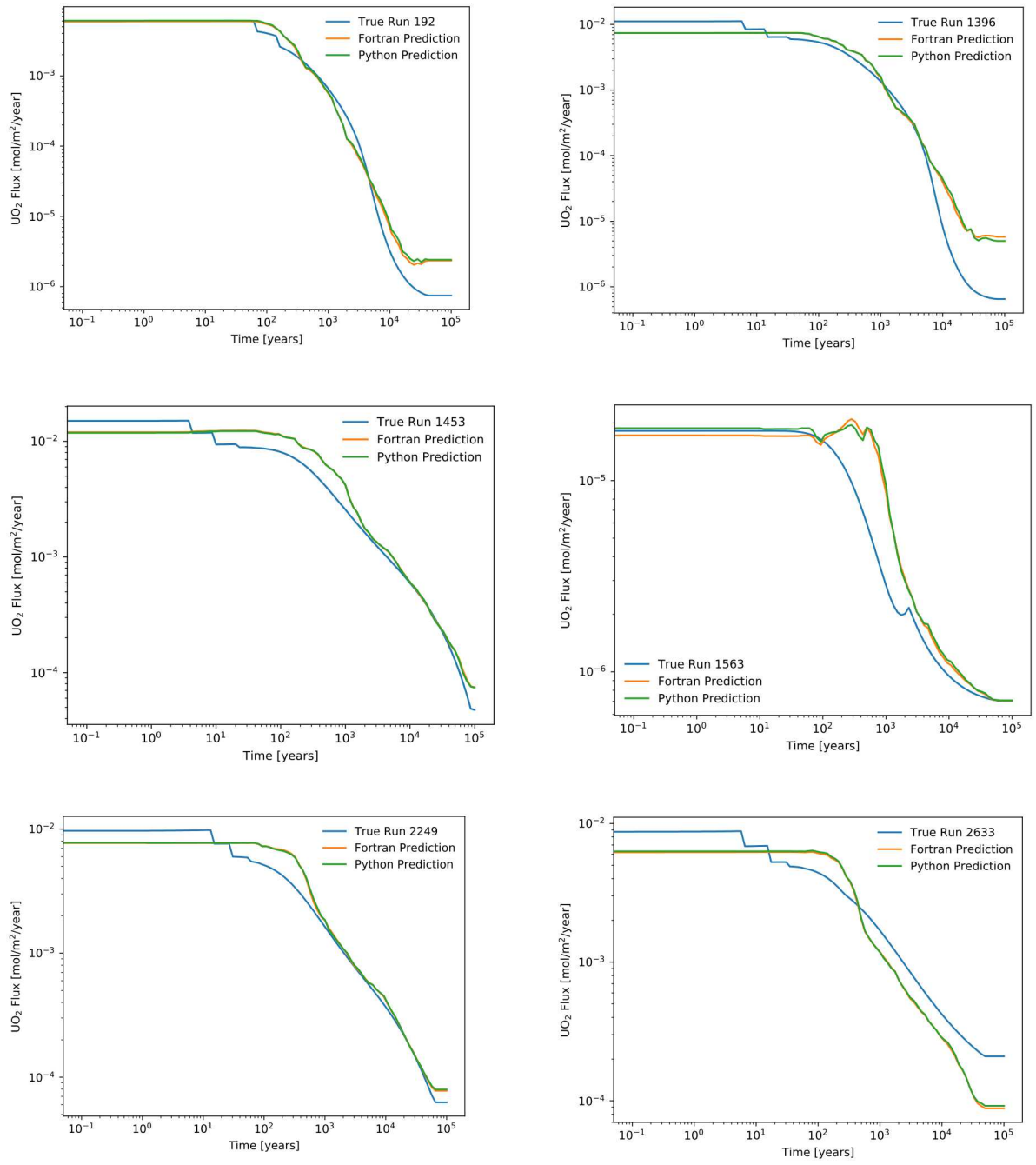

Figure 3-17 Samples of six runs of the Python and Fortran predictions along with the true values. The two versions do not return identical predictions, but the differences are an order of magnitude or smaller than the errors with respect to the true values.

3.3.2 Accuracy

To gain insights into the errors on the test set, one hundred random trajectories predicted with the 250K/60NN kNNr configuration are compared to the true trajectories in Figure 3-18. While some predictions appear to be working reasonably well, there are zones where performance is poor. Looking more closely at the best, worst, and median predictions (on a per-run averaged basis) in Figure 3-19 reveals runs with considerable errors. Since kNNr is essentially a look up table, for these cases the algorithm must be operating in areas where it lacks the density of training data required.

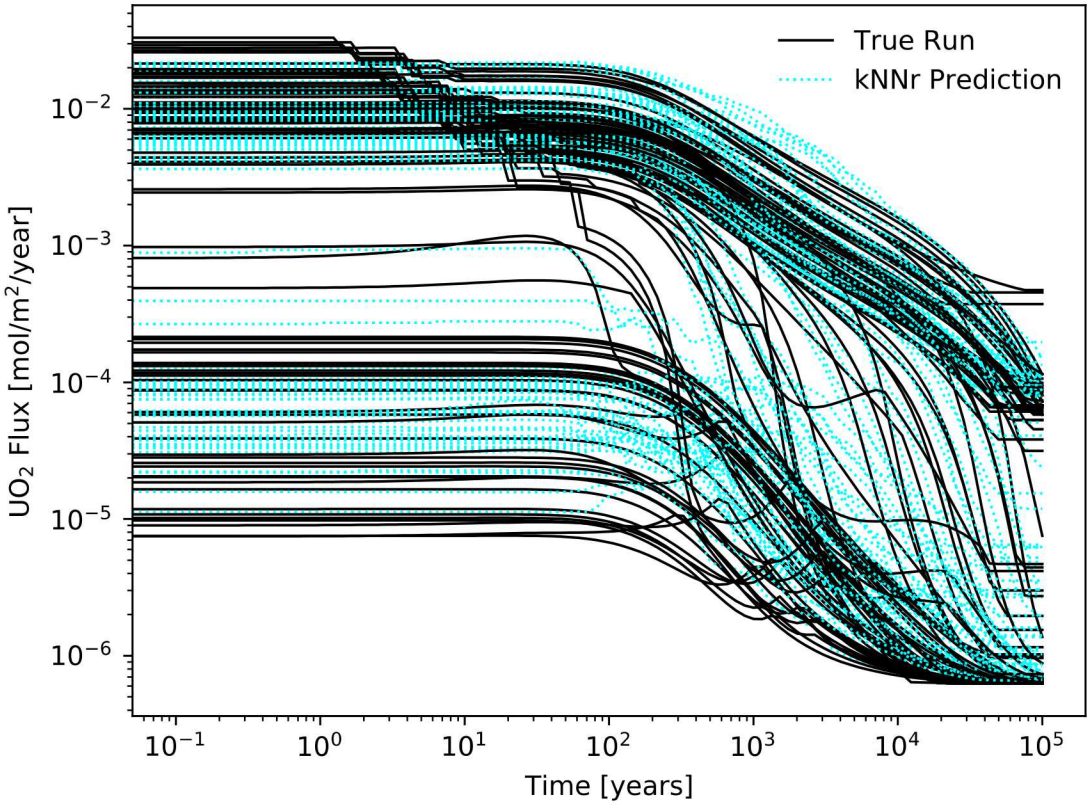

Figure 3-18 A random sampling of runs reveals that while some predictions are hitting the mark, others are not as accurate.

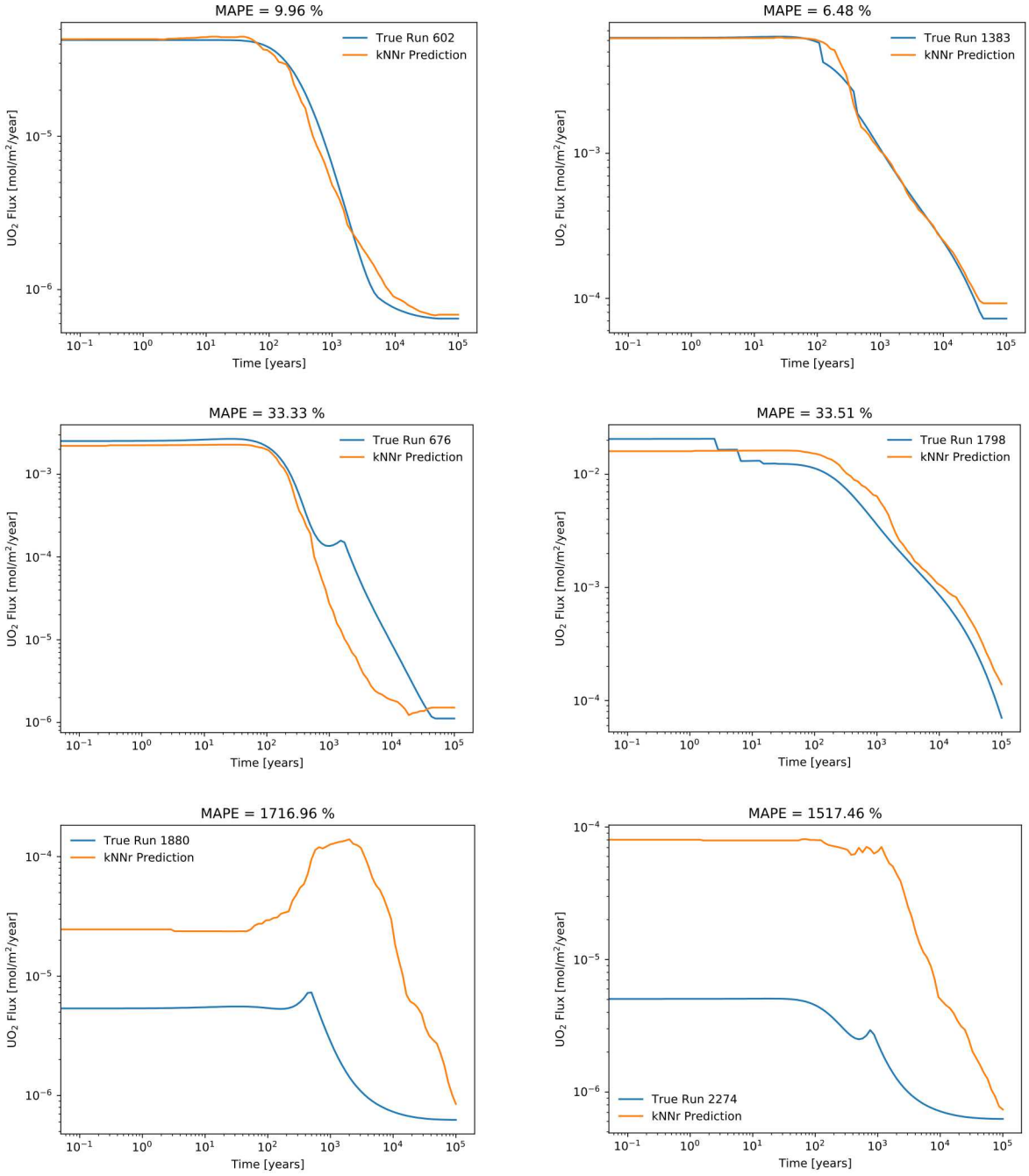

Figure 3-19 Plots of the best, median and worst kNNr runs by prediction mean absolute prediction error (MAPE)

To get a better feel for the error distributions, Figure 3-20 plots the histograms of the run-averaged MSE, MAPE, and MAE errors. When binning the errors themselves, as in the left column, the histogram shows a peak near zero, and then a long tail towards larger errors. By binning the 10-based log of the errors, we get a better insight into the behavior of the tails. Both the MSE and MAE error metrics show large secondary peaks in the error distributions. Future work will focus on these peaks and other outliers to see where the table does not perform well, and adjust the sampling strategy, distance metric, or data conditioning accordingly.

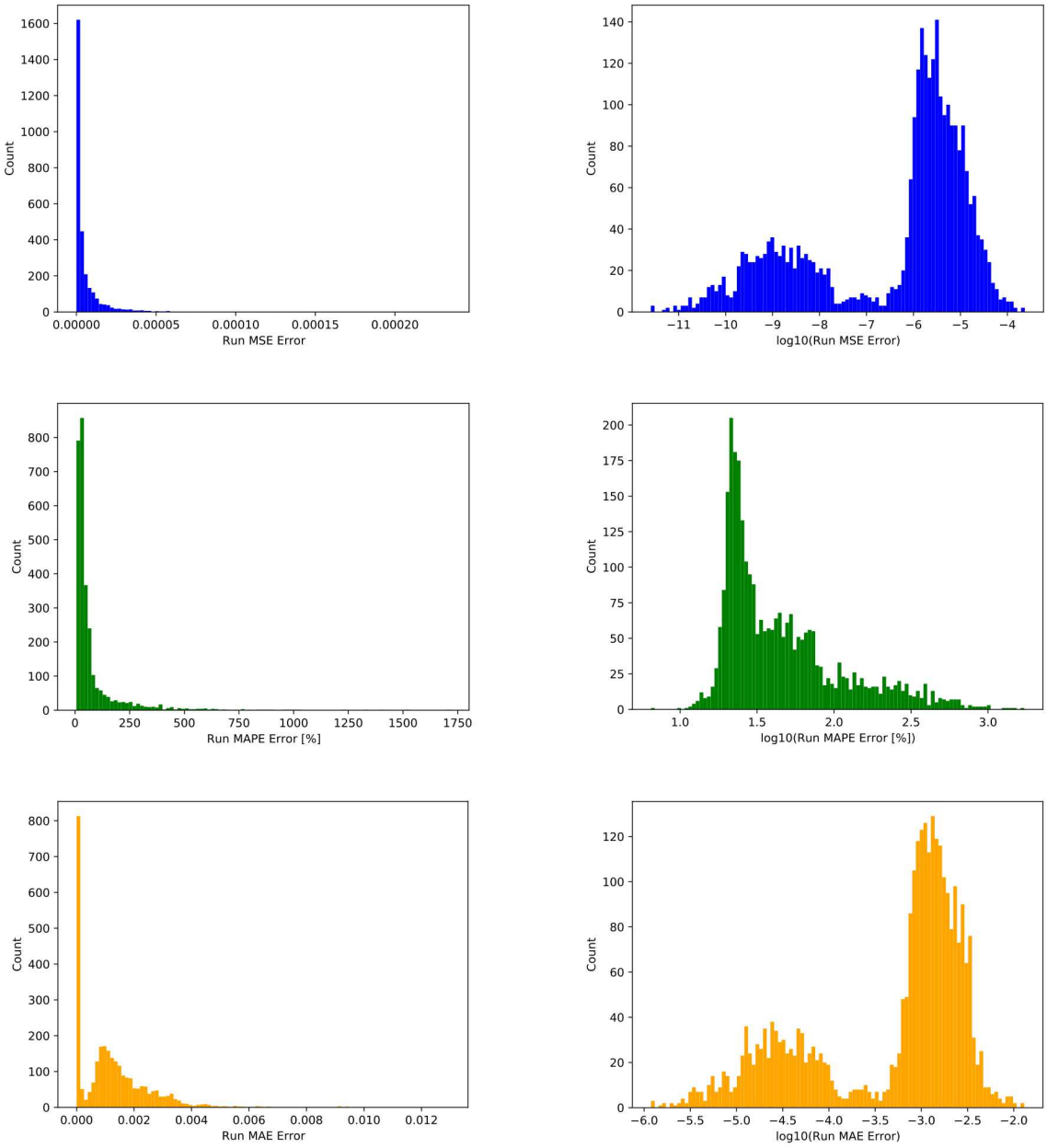

Figure 3-20 Histogram plots of the kNNr prediction errors reveals a large set of predictions that are much less accurate.

3.4 Summary Comparison

Table 3-5 compares the accuracy of ANN and kNNr surrogates on the test data set. For the data sets used here, the ANN method is consistently more accurate than the kNNr approach. Note also that the two kNNr configurations give nearly the same accuracy on the test set. As such, the 250K/60NN configuration is the one that will be used in the PFLOTRAN coupled production runs.

Table 3-5 Values of the error metrics computed on the test set (and training set for the ANN) for the UO_2 surface flux in the "natural" (i.e., non-log_{10} transformed) space. The accuracy of the two kNNr models is nearly the same. Note that the error on the training data is zero for kNNr as we use inverse-distance weighted averaging so the table prediction is the same as the tabulated value at the query point.

Surrogate	Train MSE $(mol/m^2/yr)^2$	Test MSE $(mol/m^2/yr)^2$	Train MAE $(mol/m^2/yr)$	Test MAE $(mol/m^2/yr)$	Train MAPE	Test MAPE
ANN	3.73e-6	3.56e-6	9.69e-4	9.30e-4	41.7%	31.2%
kNNr 250K / 60NN	N/A	5.95e-6	N/A	1.25e-3	N/A	78.4 %
kNNr 500K / 120NN	N/A	6.04e-6	N/A	1.26e-3	N/A	78.0 %

4. COUPLING FMD SURROGATES TO PFLOTRAN

There are a few details that must be accounted for to achieve proper coupling between PFLOTRAN and the surrogate models. First, the temperature input to the surrogate must be evaluated at the waste package spatial location. The local environmental concentrations of CO_3^{2-}, O_2, Fe^{2+}, and H_2 must also be evaluated, but currently they remain constant over the simulation domain. Second, two unit conversions between PFLOTRAN and those expected by the surrogate are necessary: mol/liter to mol/m^3 for concentrations and degrees Celsius to degrees Kelvin for temperature. Lastly, all of the surrogate features except dose rate are directly available from PFLOTRAN. Dose rate at the fuel surface is obtained by passing burnup, simulation time, and a decay time offset into a function that computes the dose rate at the fuel surface according to the formula in Radulescu (2011).

Unlike the actual FMD process model, there is no notion of a time step in the surrogate models. Both of surrogates are a functional representation (i.e., a mapping) between the feature space and model output. There is no history-dependence and none of the features are rate-like quantities. This is a particularly attractive property of the surrogate models given the restrictive time discretization ($\log_{10}$ time increments) required by the FMD process model, which PFLOTRAN must adhere to for the FMD model to work correctly. If a surrogate model is used instead then PFLOTRAN is free to adaptively form the time discretization.

Documentation for the PFLOTRAN implementation of the ANN surrogate can be found at https://doc-dev.pflotran.org/theory_guide/pm_waste_form.html and https://doc-dev.pflotran.org/user_guide/cards/gdsa/waste_form_general_card.html.

4.1 ANN Surrogate

There are two input parameters for the original FMD process model and the surrogates that are specific to these models (i.e., are not present in other, non-FMD process models). These include the burnup of the waste form in GWd/MTHM and the decay time that represents the age of the fuel prior to the beginning of the PFLOTRAN simulation. The decay time for the surrogate training and test sets was 100 years. This value was used in the demonstration problems in Section 5.

The neural network coefficients are stored in a HDF5 file that must be present in the directory where the simulation is executed. This file can be found in $PFLOTRAN_SRC/regression_tests/ufd and is named fmdm_ann_coeffs.h5.

The PFLOTRAN implementation of the ANN in pm_waste_form.F90 produces essentially the same input as its original Python form. Evidence of this is provided in the 52-waste package section where the output of the surrogate models is computed from PFLOTRAN's waste form output files and compared the results to the Python versions of each surrogate.

4.2 kNNr Surrogate

The input parameters to the kNNr coupled model in PFLOTRAN are similar to the ANN with a few additions. Under the MECHANISM card in PFLOTRAN, kNNr must be specified. Optional input parameters include number of nearest neighbors and epsilon value. The coupled model uses a default of seven nearest neighbors and the smallest positive real number that is non-zero for epsilon. If a query vector is found to be within epsilon of a value in the kNNr search tree, then the exact value is used rather than taking an inverse distance. The HDF5 input file must be named FMDM_knnr_data.h5 and located in the same folder as the input deck. An example can be found at $PFLOTRAN_SRC/regression_tests/ufd.

The coupled model reads in a HDF5 file of the FMD generated featured data and formats the data into a kNNr search tree. At each time step, PFLOTRAN will call the coupled kNNr and input the state of a cask represented as a query vector that includes temperature, concentrations, and dose rate. The query vector is

then log transformed. The kNNr search tree is then used to find the k nearest neighbors and extract the quantities of interest. An inverse distance is used to take a weighted average of the quantities of interest in order to determine the final dissolution rate. The model defines a zero distance between quantities of interest as distances less than epsilon. At the end of the simulation, PFLOTRAN prints the amount of time spent in the kNNr surrogate model. The output of the coupled kNNr was compared to the coupled ANN surrogate model in PFLOTRAN for validation.

A standalone Fortran version of the kNNr model was developed to test timing on nearest neighbors and compare to the PFLOTRAN coupled version of kNNr and standalone Python version. Values are hard coded for nearest neighbors, input file names, and epsilon but can be adjusted by the user. Output files for timing on the search tree and predicted versus true values for the dissolution rate are produced when the code is run. The standalone model can be found at: https://gitlab-ex.sandia.gov/bjdebus/parme/-/tree/master/code/rosie_knnr.

5. PERFORMANCE OF COUPLED FMD SURROGATE MODELS

The coupled surrogate models are demonstrated using two simulations. The first demonstration is a simple flow system involving 52 failed waste packages in a flow field (Section 5.1). The second is a full-scale shale repository reference case simulation.

The purpose of these simulations was to verify that the surrogates are coupled properly and to assess the computer time needed for the surrogate calculations relative to flow and transport calculations. Accuracy was also evaluated but was not the focus because the surrogates were not optimized for the narrow input ranges to which they were applied.

5.1 52-WP Demonstration

The 52-waste package (52-WP) problem was built in 2015 by Glenn Hammond. It was built to demonstrate his PFLOTRAN coupling of a new Fortran version of the FMD process model developed at Argonne National Laboratory (Jerden et al. 2015a, Mariner et al. 2015, Section 3.2.1). The purpose of the coupling was to include FMD processes in repository systems simulations.

The 52-WP problem was re-simulated using the newly coupled ANN and kNNr surrogates. Section 5.1.1 describes the model inputs, Section 5.1.2 presents and compares the model results and evaluates their speeds of computation, and Section 5.1.3 assesses whether the coupled surrogates are operating as expected.

5.1.1 Inputs

The 52-WP problem aims to demonstrate and verify the implementation of the ANN and kNNr surrogate models in PFLOTRAN as well as compare the numerical efficiency of each model. The model domain for this problem is 101 m (L) × 101 m (W) × 21 m (H) and contains 52 waste packages that are fully breached at the start of the simulation such that all fuel is exposed to groundwater. This simulation is isothermal at 25°C and has a simulation period of 100 years.

The RICHARDS mode solves the single-phase flow and isothermal system. Initially hydrostatic gradients of -7.21×10^{-4} Pa/m and -1×10^{-4} Pa/m are applied in the x- and y-axis directions, respectively. The inventory does not include actual radionuclides, but instead a pseudo free ion called “Tracer” with an initial concentration of 1×10^{-7} mol/liter and the following FMDM mechanism settings:

Table 5-1 Surrogate mechanism setting for 52-WP simulations

PFLOTRAN keyword	Values
MATRIX_DENSITY	19000 [kg/m^3]
SPECIFIC_SURFACE_AREA	0.001 [m^2/g]
BURNUP	60 [GWd/MTHM]
DECAY_TIME	100 [year]

For surrogate mechanisms, the concentrations of free ion are implemented as follows:

Table 5-2 Concentrations of free ion (environmental species)

Free ion	Free Concentration [mol/m^3]	Free Concentration [mol/liter]
O_2 (aq)	1e-6	1e-9
H_2 (aq)	1e-2	1e-5
Fe^{2+}	3e-3	3e-6
CO_3^{2-}	1.4e-2	1.4e-5

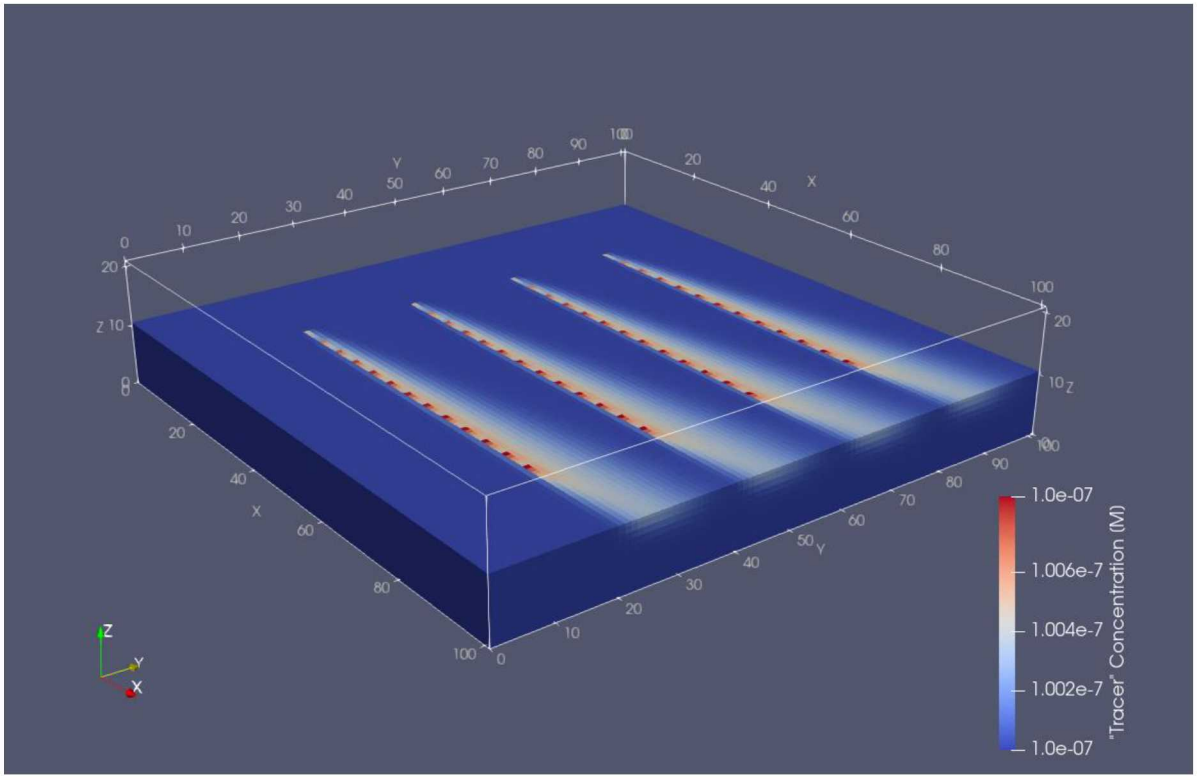

Figure 5-1 Concentration of Tracer at t = 100 years from 52-WP simulation with the ANN surrogate mechanism

5.1.2 Results

5.1.2.1 Degradation rate

From the PFLOTRAN surrogate simulation, we obtain the waste form dissolution rate [kg/s] and remaining volume [m^3] of each waste package at each time step when the waste form process model is called (with PFLOTRAN keyword "PRINT_MASS_BALANCE"). In the data spreadsheet with extension ".wf", the dissolution rate is calculated for the whole time step and used during that time step which started at the time shown in the previous row. Therefore, applying the dissolution rate at a specific time step (i^{th} time step) to the waste form volume from the previous time step ($i\text{-}1^{th}$ time step) gives the final volume at the specific time step (i^{th} time step).

To calculate the degradation rate of each waste package, the following equation is used:

$$Degradation\ rate\ \left[\frac{kg}{m^2 \cdot s}\right] = \frac{WF\ dissolution\ rate\ [kg/s]}{WF\ volume\ remaining\ [m^3] \times WF\ density\ [kg/m^3] \times Specific\ surface\ area\ [m^2/kg]} \quad (6)$$

To convert the unit of the degradation rate, the following equation is used:

$$Degradation\ rate\ \left[\frac{mol}{m^2 \cdot yr}\right] = \frac{Degradation\ rate\ [kg/(m^2 \cdot s)]}{Mole\ weight\ of\ UO_2\ [g/mol]} \times (Time\ conversion) \quad (7)$$

Calculated degradation rates for the two models are shown in Figure **5-2**. These rates are verified and compared to expected values in Section 5.1.3. The rates from the ANN surrogate are close to expected rates, but the kNNr rates are far below expected rates (Section 5.1.3). The 52-WP problem interrogates a very small region of the entire training domain and only at the minimum temperature (25°C) of the training domain. For these conditions, the training dataset is sparse (Section 5.1.3) and the kNNr surrogate is clearly not optimized. A temperature of 25°C was used to be consistent with the 52-WP simulation performed in 2015 using the coupled FMD process model. Future 52-WP simulations should use a higher temperature to be more consistent with the dose rate of spent fuel at early time.

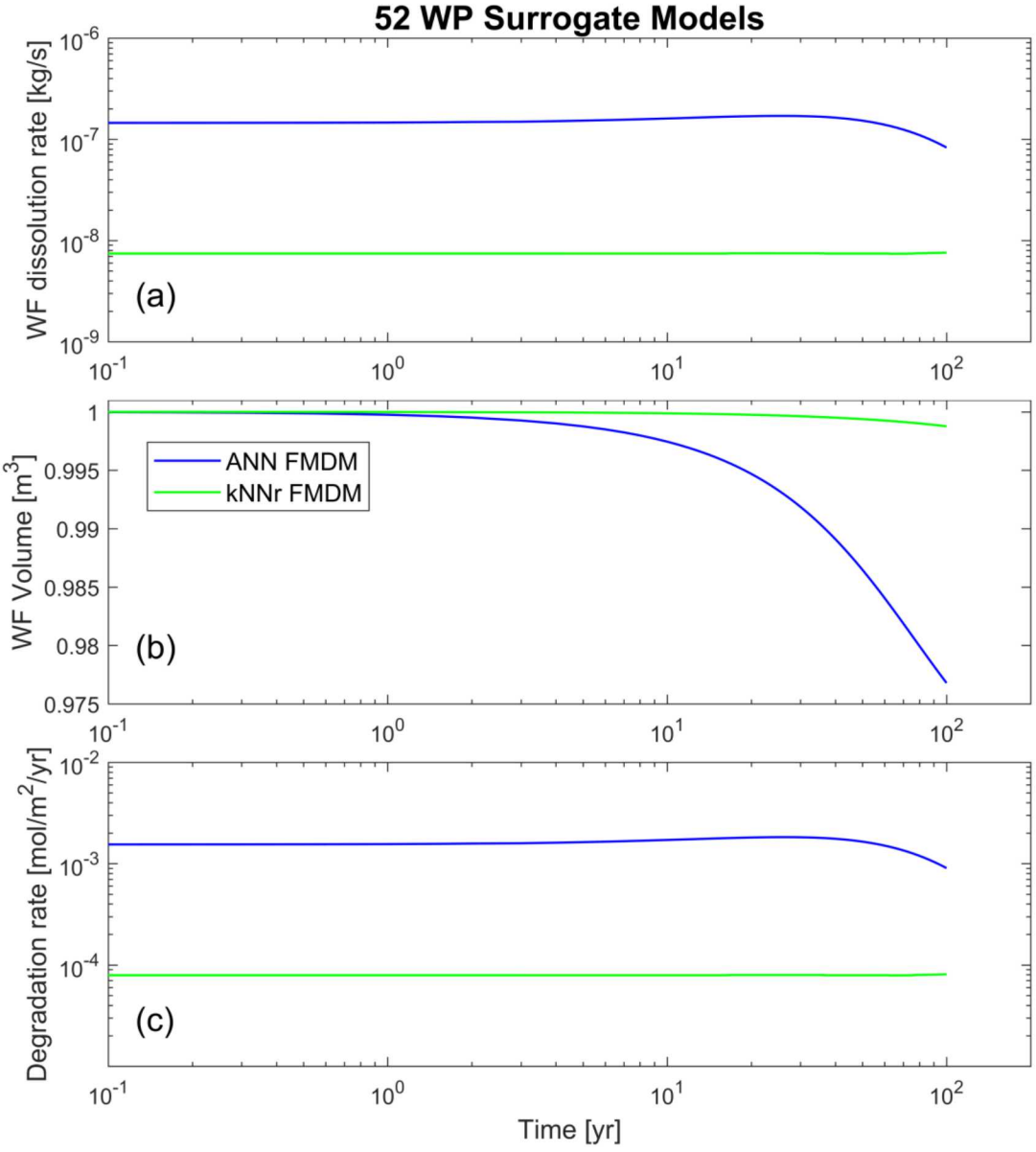

Figure 5-2 ANN and kNNr surrogate calculations of the waste form degradation rate expressed in two different units (a) and (c) and the resulting waste form volume (b) over time

5.1.2.2 Speed comparison

The surrogate model simulations of the 52-WP problem ran on 4 processors. Table 5-3 shows the speed comparison for the problem using the ANN surrogate, kNNr surrogate, and coupled FMD process model for time spent in the flow, transport, and waste form computations. The time spent in the waste form computations in the ANN and kNNr surrogate model simulations is only a small fraction of the time spent on flow and transport calculations. This result is very different from what happens using the directly-coupled FMD process model. Time spent on waste form calculations using the coupled FMD process model is more than three orders of magnitude higher than for the surrogate model simulations. The original 52-waste package problem, simulated in 2015 using the coupled FMD process model, is documented in Mariner et al. (2015, Section 3.2.1).

Table 5-3 Speed comparison of 52-waste package simulations with ANN and kNNr surrogate mechanisms

	Time consumed [s]		
	ANN FMD	kNNr FMD (250k training)	Coupled FMD Process Model (Mariner et al. 2019b)
Flow	61.0	60.3	128
Transport	147.8	117.6	244
Waste Form	0.05	0.54	1522

5.1.3 Surrogates Implementation Verification

The waste form output files with extension ".wf" produced by PFLOTRAN contain the effective dissolution rate (top panel), waste form volume (middle panel) shown in Figure **5-2**. The bottom panel, "degradation rate", can be computed according to equation (6), and is the output of the surrogate models. The correspondence between the PFLOTRAN implementation of the surrogates compared to their original Python versions can be quantified by comparing the back-calculated degradation rate to the output of the Python implementations given the inputs from the 52-WP problem.

One wrinkle in the calculation of degradation rate from the waste form file arises in temporal ordering of effective dissolution rate and waste form volume in the file. If "n" represents a time discretization vector index (or equivalently line number) in the waste form output file within a given line the following quantities are written as: simulation_time[n], effective_dissolution_rate[n-1], volume[n]. The effective_dissolution_rate[n] is computed using volume[n] and degradation_rate[n] (i.e., all from the same time step). The effective_dissolution_rate[n] quantity takes into account specific surface area, volume, etc. according to equation (6). These indexing differences must be accounted for to correctly compute the degradation rate for a given point in time.

Figure 5-3 shows the degradation rates computed from the waste form files and compares them to the output of the MATLAB FMD code generated using the same inputs. There is a noticeable discrepancy between the predictions of the surrogate models and FMD process model. The most likely reason for this is the iso-thermal nature (temperature equal to 298.15 K) of the 52-WP problem. The temperature function in the training data set typically began at temperatures up to 398 K and decayed over time to 298 K. It is probable that the training dataset does not contain many points with near-298 K temperatures at early (less than one hundred years) times.

Alternatively, the inputs to the 52-WP problem could be in a region of the input space where the surrogate models are not accurate. The training domain for the surrogates is fairly wide in terms of possible inputs (notably with respect to concentration ranges), and varying accuracy across the input space is a price paid for this range of applicability. Surrogates that are more accurate for the inputs in the 52-WP problem could be constructed by narrowing the concentration ranges (or fixing) to the values used by this demonstration problem, but then the surrogates would only be valid for this narrow case. In other words,

prediction accuracy would likely be degraded for inputs that our current surrogates provide accurate estimates for.

Lastly, Figure 5-4 and Figure 5-5 display the percent difference between the back-calculated degradation rate from the waste form files from PFLOTRAN and the standalone Python surrogate's outputs relative to the Python calculations. The differences between the versions for the ANN are within the range expected by the limited precision (around nine digits) of the waste form and MATLAB FMD (used in model training) file outputs. The kNNr implementations contain more variation, and this can be attributed to the different codes used in Python and Fortran versions for k-d Tree tabulation, as discussed in Section 3.3.1. While the correspondence is not exact, these results do provide a degree of confidence in the correctness of the PFLOTRAN implementations.

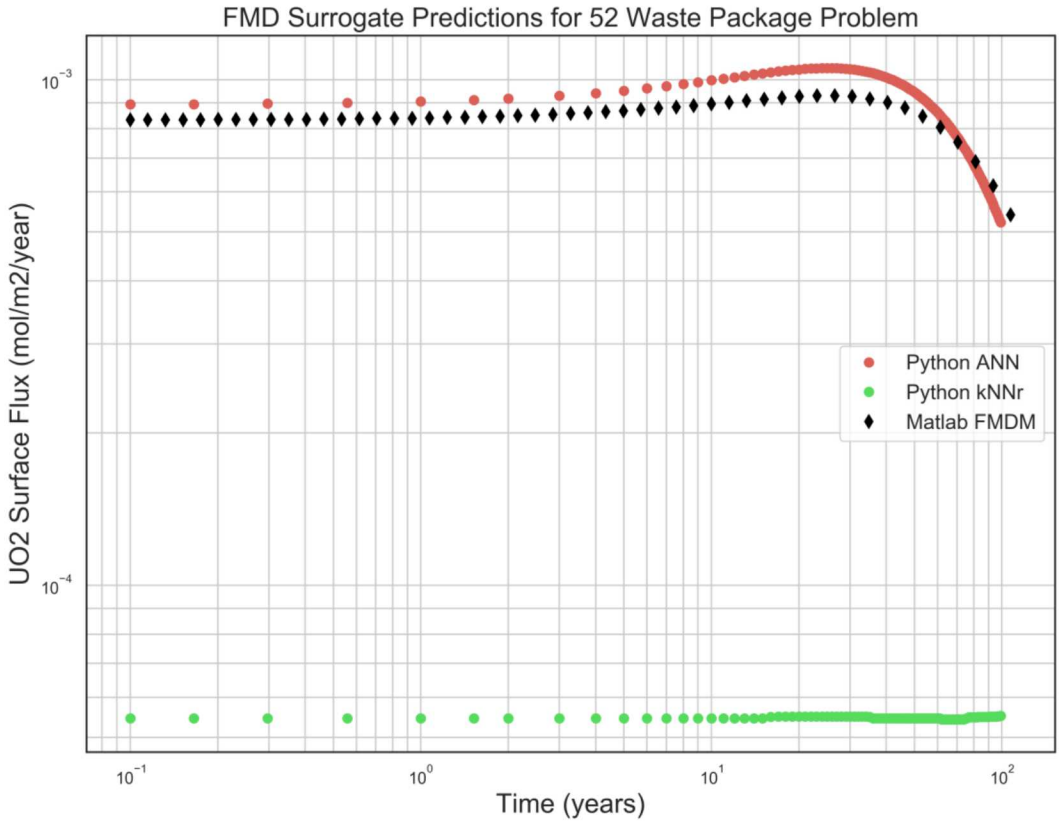

Figure 5-3 Comparison between the predictions from the surrogate models and original FMD process model for the inputs used in the 52-WP problem

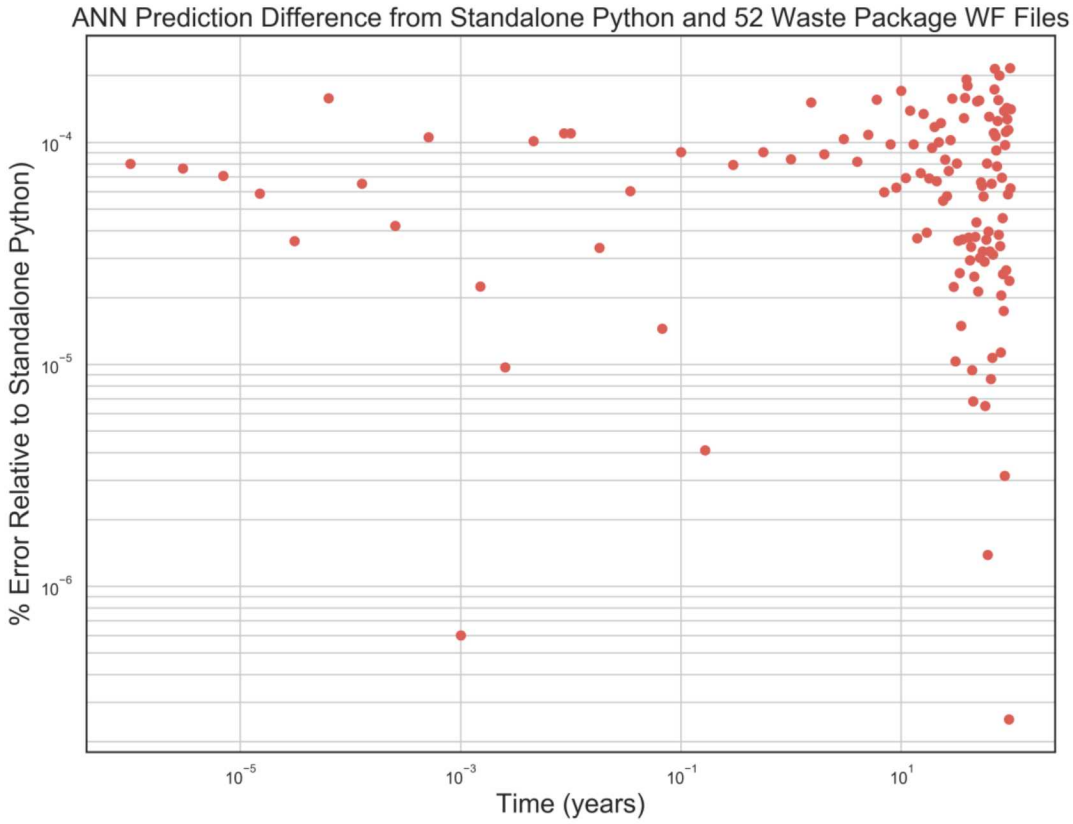

Figure 5-4 ANN prediction differences between standalone Python version and version used in the 52-WP simulation

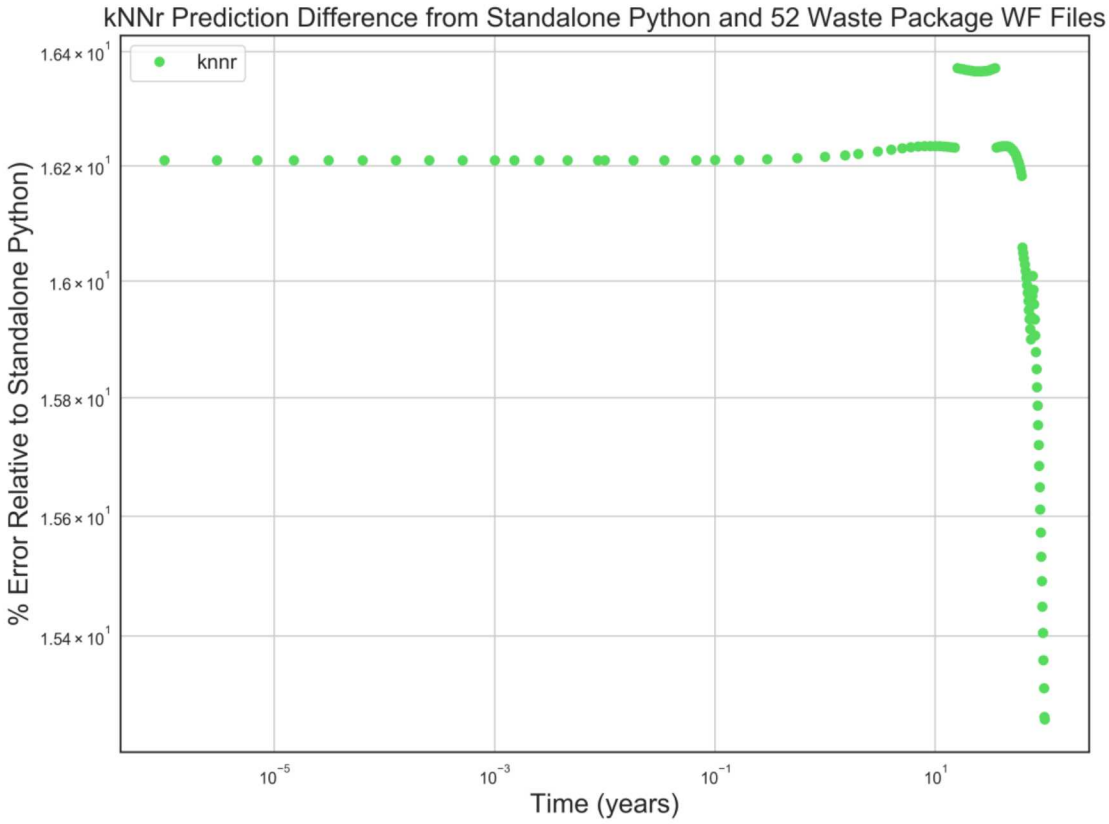

Figure 5-5 kNNr prediction differences between standalone Python version and version used in 52-WP simulation. The smoothness of this error trajectory suggests that points from only a few trajectories were used as the nearest neighbors to the actual condition.

5.2 Shale Repository Demonstration

This demonstration exercises the surrogate models in the shale repository system. The shale repository simulation includes (Sevougian et al. 2019):

- Natural and engineered barrier system in a shale formation
- 24-PWR and 37-PWR waste packages
- Inventory of radionuclides and waste forms used in the 2019 PA simulation model

5.2.1 Inputs

The model domain is 7215 m (L) × 2055 m (W) × 1200 m (H), long enough to implement observation points 5000 m down-gradient of the repository in the x-axis direction. The numerical grid contains 9.88 million cells, of which approximately 4.6 million are the finer cells in and around the repository. The repository is discretized into volumetric cells with size of 5 m (L) × 5 m (W) × 5 m (H) while most of other regions are discretized into 15 m (L) × 15 m (W) × 15 m (H) cells.

Initial pressure and temperature fields represent geothermal temperature and hydrostatic pressure gradients in a vertical direction (positive y-axis direction) by applying a liquid flux of 0 m/s and an energy

flux of 60 mW/m^2 to the bottom and atmospheric pressure and 10°C of temperature at the top boundary. Pressure at the top decreases from west to east (positive x-axis direction) with a head gradient of -0.0013 m/m. Initially, unsaturated conditions (0.7 of gas phase saturation) are applied in the disposal drifts, hallways, and shafts, and concentrations of all radionuclides in all cells are 10^{-21} mol/liter.

The symmetry boundary condition is applied to the south face (positive zx-face) whereas other boundaries have constant pressure and temperature. Radionuclide concentrations are 10^{-21} mol/liter at the upstream boundary and zero at the outflow boundaries. The simulation runs to 10^6 years.

This field-scale model simulates waste package degradation, waste form dissolution, equilibrium-controlled radionuclide sorption, precipitation/dissolution, radioactive decay and ingrowth in all phases, and multi-physical coupling of heat transfer, fluid flow and radionuclide transport.

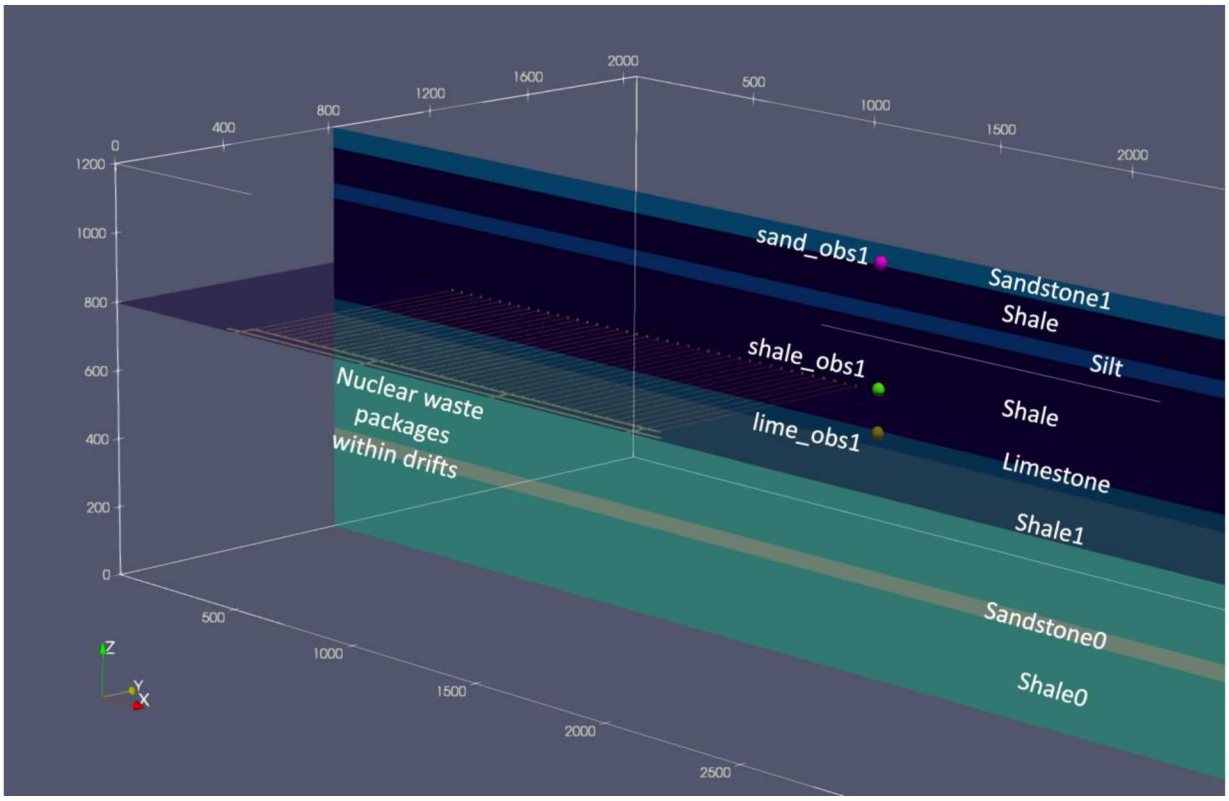

Figure 5-6 Model domain of the shale repository system

5.2.1.1 Formation Properties

The nuclear waste repository is located at depth of 405 m within the shale formation. The stratigraphic units of the layered system are shown in Figure 5-6 and the material properties are given in Table 5-4.

In this modeling study, the disturbed rock zone (DRZ) is defined manually by assuming 10 times larger permeability than that of the host rock to consider the changes in rock properties adjacent to the engineered barrier system.

Table 5-4 Material properties for the shale repository system model

	Host Shale	Upper sandstone (sandstone1)	Silty shale (silt)	Limestone (Limestone)	Lower shale (shale1)	Lower sandstone (sandstone0)	Bottom shale (shale0)
Location [m] (Domain top at 1200 m and bottom at 0 m)		1140-1200	990-1035	645-690	300-645	255-300	0-255
Permeability [m^2]	1e-19	1e-13	1e-17	1e-14	1e-20	1e-13	1e-20
Porosity [–]	0.2	0.2	0.2	0.1	0.1	0.2	0.1
Density[kg/m^3]	2700	2700	2700	2700	2700	2700	2700
Heat capacity	830	830	830	830	830	830	830
Saturated thermal conductivity [W/m/K]	1.2	3.1	1.4	2.6	1.2	3.1	1.2
Dry thermal conductivity [W/m/K]	0.6	1	0.8	1	0.6	1	0.6
Liquid residual saturation	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Gas residual saturation	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Saturation function	Van Genutchen function						
alpha [Pa^{-1}]	6.67e-7	1e-4	6.67e-7	1e-4	6.67e-7	1e-4	6.67e-7
m [–]	0.333	0.5	0.333	0.5	0.333	0.5	0.333

5.2.1.2 Waste Packages

The shale reference case implements 1575 24-PWR and 1000 37-PWR waste packages consisting of stainless-steel canisters and overpacks. Note that only half of the waste packages are modeled due to the symmetry-domain setting. The radionuclide inventory at the time of emplacement and heat of decay as function of time are calculated via decay and ingrowth from the 5-year out of reactor (OoR) radionuclide inventories (Sevougian et al. 2019). Calculated values for 24-PWR waste packages assume an initial enrichment of 3.72 wt% ^{235}U, 40 GWd/MTHM burn-up, and 100-year OoR surface storage prior to deep geologic disposal. Calculated values for 37-PWR waste packages assume an initial enrichment of 4.73 wt% ^{235}U, 60 GWd/MTHM burn-up, and 150-year OoR storage.

The waste packages are modeled as cuboids ($1.67 \times 1.67 \times 5$ m^3) to resolve gridding limits. The porosity of waste packages is set to 0.5, which is equal to the fraction of void space. Its permeability is set to 1×10^{-16} m^2, several orders of magnitude higher than that of the surrounding materials.

The temperature-dependent degradation rate per year with a truncated log normal distribution (a mean of -4.5, a standard deviation of 0.5, and an upper truncation of -3.5 in log units) is implemented to calculate normalized remaining canister thickness (fractional thickness) at each time step.

5.2.1.3 Bentonite Buffer

Compacted bentonite, filling access drifts and shafts, was engineered to have low permeability and high sorption capacity by mixing 70% bentonite and 30% quartz sand for 24-PWR disposal and adding 15% graphite for 37-PWR disposal.

Both buffer materials have a porosity of 0.35 and a permeability of 10^{-20} m^2. The bentonite/sand buffer has a saturated thermal conductivity of 1.5 W/m/K and a dry thermal conductivity of 0.6 W/m/K whereas the bentonite/graphite buffer has 3.0 and 1.9 W/m/K, respectively.

5.2.1.4 Geochemical Conditions

For the surrogate mechanism simulation, four environmental species are added to the chemical database and input file as follows:

Table 5-5 Additional primary species for surrogate mechanism simulations

	Ion size (a0)	Charge (Z)	Molar Weight (g/mol)	Free Conc. (mol/m^3)	Free Conc. (mol/liter)
O_2 (aq)	3.0	0.0	31.9988	1e-6	1e-9
H_2 (aq)	3.0	0.0	2.0159	1e-2	1e-5
Fe^{2+}	6.0	2.0	55.8470	3e-3	3e-6
CO_3^{2-}	4.5	-2.0	60.0092	1.4e-2	1.4e-5

The updated database is named as “ufd-decay_ann.dat” for PFLOTRAN simulations.

5.2.1.5 Surrogate Mechanism Inputs

Table 5-6 Surrogate mechanism settings for the shale repository system model

PFLOTRAN Keyword	24-PWR	37-PWR
MATRIX_DENSITY	10970 [kg/m^3]	10970 [kg/m^3]
SPECIFIC_SURFACE_AREA	0.001 [m^2/g]	0.001 [m^2/g]
BURNUP	40 [GWd/MTHM]	60 [GWd/MTHM]
DECAY_TIME	100 [year]	100 [year]

5.2.2 Results

5.2.2.1 Degradation Rate

The original shale reference case uses a simple fractional degradation rate (FDR) model for fuel matrix degradation. FDR specifies a rate of 10^{-7} yr^{-1} for the waste form matrix in this simulation. This model and this 10^{-7} yr^{-1} rate is used, for example, in the Swedish repository performance assessment model (SKB 2006, Table 10-3).

Figure 5-7 shows the time when each waste package breaches in the simulation and the waste form degradation rate history for each waste package. The yellow and orange lines are for the 37-PWR waste packages and the blue and green lines are for the cooler 24-PWR waste packages. The top and bottom graphs plot the fuel matrix degradation rate in units of kg/s and $mol/m^2/yr$, respectively. Because a constant specific surface area is assumed (0.001 m^2/g), the bottom graph shows that as soon as the waste package breaches, the fuel matrix degrades at a constant rate of approximately 3.7×10^{-7} $mol/m^2/yr$ for the remainder of the simulation. The middle graph tracks the remaining fuel matrix volume for each waste package.

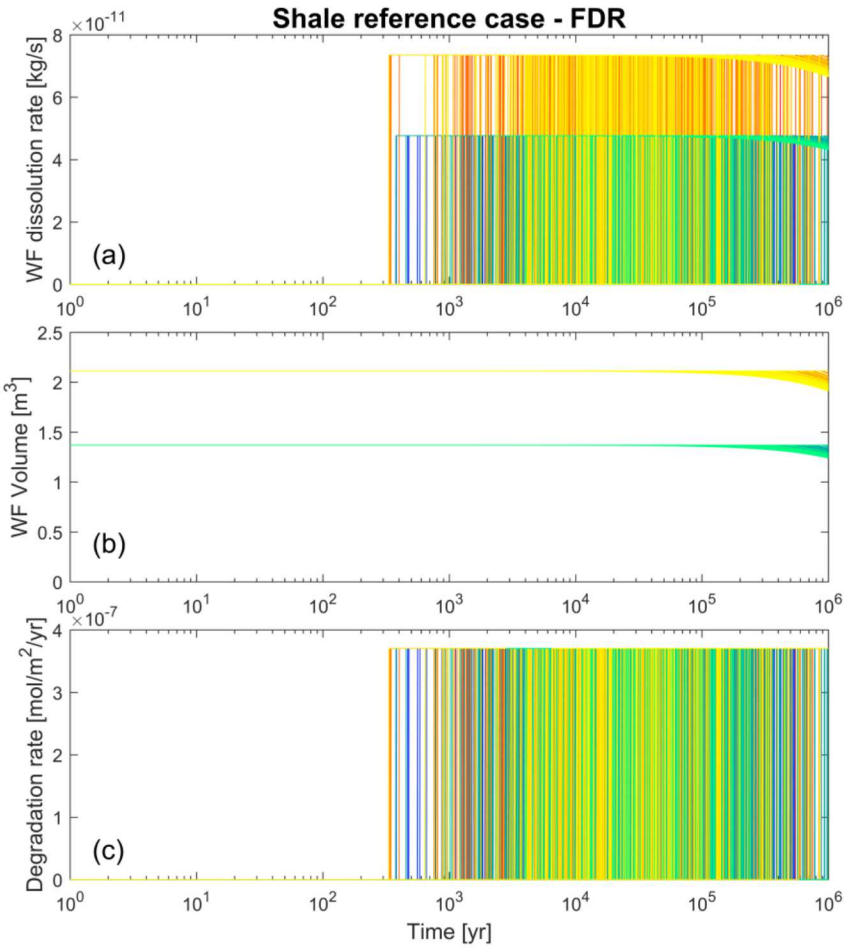

Figure 5-7 FDR model results: (a) fuel matrix degradation rate (kg/s), (b) volume of remaining waste form (m^3), and (c) specific degradation rate ($mol/m^2/yr$)

Figure 5-8 shows the same three graphs for the same shale reference case simulation, but instead of using the FDR fuel matrix degradation model, the ANN FMD surrogate model is used. The ANN surrogate calculates much higher fuel degradation rates than the FDR simulation, especially at early times. Because the surrogate emulates the FMD process model, the specific degradation rate ($mol/m^2/yr$) is high at early times when the temperature (>100°C) and dose rate are high, and it decreases over time as temperature and dose rate decrease. These trends are expected and are a major advantage over the FDR model. Also, the rates for the 24-PWR waste packages are generally lower than the rates for the 37-PWR waste packages because of differences in temperature. ANN rates for both 24-PWR and 37-PWR converge after 100,000 years to approximately 8×10^{-7} $mol/m^2/yr$.

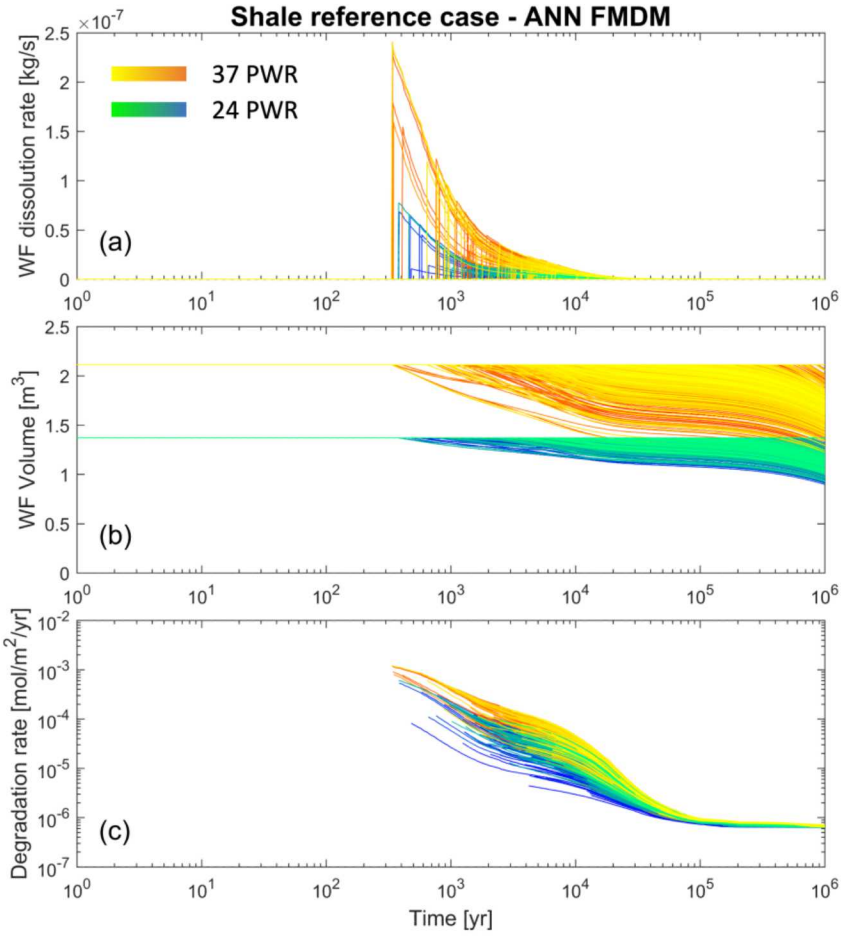

Figure 5-8 ANN FMD surrogate model results: (a) fuel matrix degradation rate (kg/s), (b) volume of remaining waste form (m^3), and (c) specific degradation rate ($mol/m^2/yr$)

Figure 5-9 shows the same plots for shale reference case when the kNNr surrogate model is used. The results show similar trends in degradation rates (high rates upon waste package breach and then gradual decreases). The maximum value of the specific degradation rate (~1×10^{-4} $mol/m^2/yr$) is considerably lower than the maximum specific degradation rate predicted by the ANN surrogate. The value over time converges to 8×10^{-7} $mol/m^2/yr$ as observed in the ANN surrogate model.

The lower degradation rates predicted by the kNNr surrogate relative to the ANN surrogate at early time is similar to what was observed in the 52-WP problem. Further work is needed to identify specific waste packages and their temperature histories to determine expected fuel matrix degradation rates for those waste packages. That information is needed to compare to the predictions by the ANN and kNNr surrogates for the waste packages.

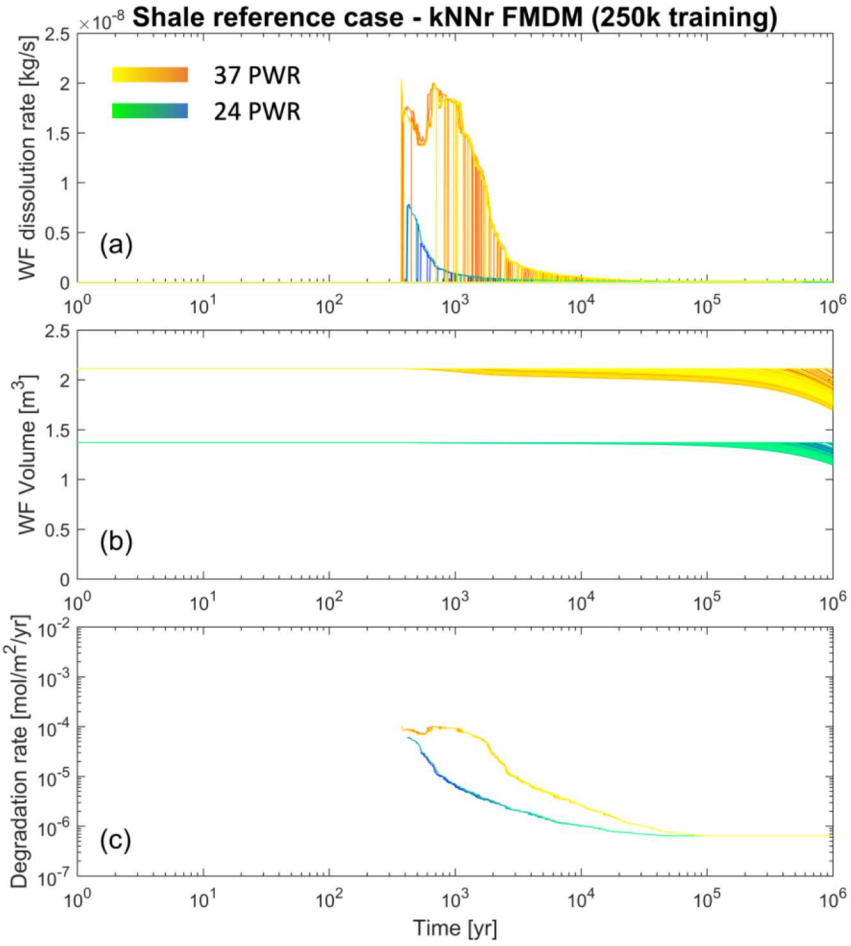

Figure 5-9 kNNr FMD surrogate model results using the 250k training input data: (a) fuel matrix degradation rate (kg/s), (b) volume of remaining waste form (m^3), and (c) specific degradation rate ($mol/m^2/yr$)

5.2.2.2 Speed Comparison

The shale repository system simulations ran on 1024 processors. Table 5-7 shows the speed comparison of surrogate model simulations.

Table 5-7 Speed comparison of the shale repository system simulations

	Time consumed [s]	
	ANN FMDM	kNNr FMDM (250k training)
Flow	26574	26275
Transport	27329	27224
Waste Form	281	334

As in the 52-WP problem, the ANN surrogate model consumes less time than the kNNr surrogate, but each surrogate consumes very little time relative to flow and transport. Times spent on waste form calculations relative to the total time spent on flow and transport calculations were 0.5% and 0.6% for ANN and kNNr, respectively. Based on the speed comparisons in the 52-WP problem, if the shale case

had been run using the coupled FMD process model, the simulation of the FMD process model would have required much more time than flow and transport.

6. FUTURE WORK

The results of the coupled PFLOTRAN simulations in Section 5 clearly show the potential of surrogate models to enable accounting for detailed Fuel Matrix Degradation dynamics while keeping the computational cost of reservoir simulations manageable. While this is a successful proof of concept, there is room for improvement in the surrogate accuracy.

Based on the analysis of the demonstration cases, the area with the most potential for improvement is the sampling of the training data, both in terms of sampling range and sampling approach. As the 52-waste package case clearly shows, if the training data does not contain enough samples from the regime where the simulation takes place, the accuracy will be insufficient. A first step to address this will be to examine the areas where the current surrogate models do not have enough accuracy to see if those areas indeed correspond to regions with insufficient density in the training data. As mentioned earlier, one mitigation approach is to widen the sampling range for the training data, to get a more generally applicable surrogate (ANN or kNNr alike). Alternatively, one could also create smaller training data sets that are targeting a specific application, to get problem-specific surrogates that might have high accuracy and speed, but lack generality.

In terms of the sampling scheme itself, the training data is currently generated from simulating the degradation of standalone fuel casks over time, under constant environmental conditions. As discussed in section 3.3.1, this results in a dataset of tightly clustered data points, one cluster per run, rather than a fully random sampling over the feature space. As the ANN uses a general non-linear functional representation, it is able to interpolate between those clusters. However, the kNNr approach would do better with a more random distribution of the training data, so that every query point is surrounded with a random cloud of training points. One approach is to generate standalone trajectories for a much larger set of input conditions than currently done, but only use a sparse subset of the available points in each run. This could also be generalized by only allowing new points to be added to the training data if they are sufficiently different from the existing points in the training data.

The sampling enhancements discussed so far are likely to benefit both the ANN and kNNr approach. Some other approaches that may primarily benefit the kNNr approach are listed below:

- **Manhattan distance metric:** As shown in section 3.3.1, the Manhattan distance tends to give better accuracy than the Euclidean metric for the current datasets, and could be added to the KDTREE 2 Fortran implementation.
- **Enhanced data conditioning:** Rather than simple log10 transformation of the training data, kNNr may benefit from additional normalization so that all features are weighted appropriately when selecting the nearest neighbors. In this context, it may be beneficial to emphasize some features over others if they are more predictive for the Quantity of Interest. Note that data conditioning has a similar effect as the choice of the distance metric used for selecting the nearest neighbors.
- **Appropriate metrics for hyperparameter tuning:** As discussed in section 3.3.1, the hyperparameters for kNNr were selected to minimize the MAPE error, as it weighs the accuracy on small fluxes more evenly than the MSE and MAE metrics. However, if it is most important to get large flux values predicted well, then selecting hyperparameters to minimize the MSE or MAE metrics may be better suited.
- **Online accuracy monitoring:** With kNNr, we know that the prediction accuracy will decline the further the nearest neighbors are from the query point. This could be used to assess during run-time whether the kNNr results can be trusted.

7. CONCLUSIONS

The Fuel Matrix Degradation (FMD) model calculates spent fuel degradation rates as a function of radiolysis, redox reactions, electrochemical reactions, alteration layer growth, and diffusion of reactants through the alteration layer. It is a complicated model requiring a large number of calculations and iterations at each time step. Because of this, repository simulations, which are already expensive, cannot directly include the FMD process model, especially when hundreds or thousands of waste packages breach.

The FMD surrogate modeling work in this report was initiated based on the hypothesis that surrogate models can be developed from FMD process model training data to inexpensively provide accurate fuel matrix degradation rates in a repository simulation for each individual breached waste package in its own evolving environment at each time step. This report confirms that such surrogate models can be developed. It shows that an artificial neural network (ANN) surrogate and a k-Nearest Neighbors regressor (kNNr) surrogate can emulate the FMD process model with reasonable accuracy. It also demonstrates that these surrogates can run inexpensively in repository simulations for each breached waste package when there are thousands of waste packages. Having the ability to emulate spent fuel degradation in probabilistic PA simulations allows uncertainties in spent fuel dissolution to be propagated and sensitivities in FMD inputs to be quantified and ranked against other inputs. It is expected that the accuracy of the surrogates can be significantly improved in the future by targeting the ranges of application and by more evenly distributing the training data points within those ranges.

8. REFERENCES

Ben-David, S. and S. Shalev-Shwartz (2014). Understanding Machine Learning: From Theory to Algorithmsed. Cambridge, United Kingdom, Cambridge University Press.

Buck, E., J. L. Jerden, Jr., W. L. Ebert and R. S. Wittmann (2013). Coupling the Mixed Potential and Radiolysis Models for Used Fuel Degradation. FCRD-UFD-2013-000290. Washington D.C., U.S. Department of Energy.

Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein (2009). Introduction to Algorithms. 3rd ed. Cambridge, Massachusetts, The MIT Press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, 2009.

Helton, J. C. and F. J. Davis (2003). "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems." Reliability Engineering & System Safety 81(1): 23-69.

Jerden, J. L., Jr., K. E. Frey, J. M. Copple and W. L. Ebert (2014). ANL Mixed Potential Model for Used Fuel Degradation: Application to Argillite and Crystalline Rock Environments. FCRD-UFD-2014-000490. Washington D.C., U.S. Department of Energy.

Jerden, J., G. Hammond, J. M. Copple, T. Cruse and W. Ebert (2015b). Fuel Matrix Degradation Model: Integration with Performance Assessment and Canister Corrosion Model Development. FCRD-UFD-2015- 000550. Washington, DC, US Department of Energy.

Jerden, J., J. M. Copple, K. E. Frey and W. Ebert (2015a). Mixed Potential Model for Used Fuel Dissolution - Fortran Code. O. o. U. N. F. Disposition. FCRD-UFD-2015-000159. Washington, DC, US Department of Energy.

Jerden, J., T. Cruse, J. Fortner, K. Frey and W. Ebert (2012). Used Fuel Degradation and Radionuclide Mobilization: Experimental Plan for Electrochemical Corrosion Studies. M4FT-12AN0806011. Illinois, Argonne National Laboratory.

Jerden, J., V. K. Gattu and W. Ebert (2017). Progress Report on Development of the Spent Fuel Degradation and Waste Package Degradation Models and Model Integration. SFWD-SFWST-2017-000091, SFWD-SFWST-2017-000095. Lemont, Illinois, Argonne National Laboratory.

Jerden, J., V. K. Gattu and W. Ebert (2018). Update on Validation and Incorporationof a New Steel Corrosion Module into Fuel Matrix Degradation Model. M4SF-18AN010301017, M4SF-18AN010302016. Illinois, Argonne National Laboratory.

Kennel, M. B. (2004). KDTREE 2: Fortran 95 and C++ software to efficiently search for near neighbors in a multi-dimensional Euclidean space. https://arxiv.org/pdf/physics/0408067.pdf, Institute for Nonlinear Science, University of California.

King, F. and M. Kolar (1999). Mathematical Implementation of the Mixed-Potential Model of Fuel Dissolution Model Version MPM-V1.0. Report No. 06819-REP-01200-10005 R00, Ontario Hydro, Nuclear Waste Management Division

King, F. and M. Kolar (2003). The Mixed-Potential Model for UO2 Dissolution MPM Versions V1.3 and V1.4. Report No. 06819-REP-01200-10104 R00, Ontario Hydro, Nuclear Waste Management Division.

Mariner, P. E., D. T. Seidl, B. J. debusschere, J. Vo, J. M. Frederick and L. P. Swiler (2019b). High Fidelity Surrogate Modeling of Fuel Dissolution for Probabilistic Assessment of Repository Performance. SAND2019-4151 C. International High-Level Radioactive Waste Conference, April 16, 2019, Knoxville, Tennessee.

Mariner, P. E., E. R. Stein, S. D. Sevougian, L. J. Cunningham, J. M. Frederick, G. E. Hammond, T. S. Lowry, S. Jordan and E. Basurto (2018). Advances in Geologic Disposal Safety Assessment and an Unsaturated Alluvium Reference Case. SFWD-SFWST-2018-000509, SAND2018-11858 R. Albuquerque, New Mexico, Sandia National Laboratories.

Mariner, P. E., L. A. Connolly, L. J. Cunningham, B. J. Debusschere, D. C. Dobson, J. M. Frederick, G. E. Hammond, S. H. Jordan, T. C. LaForce, M. A. Nole, H. D. Park, F. V. Perry, R. D. Rogers, D. T. Seidl, S. D. Sevougian, E. R. Stein, P. N. Swift, L. P. Swiler, J. Vo and M. G. Wallace (2019a).

Progress in Deep Geologic Disposal Safety Assessment in the U.S. since 2010. M2SF-19SN010304041, SAND2019-12001 R. Albuquerque, New Mexico, Sandia National Laboratories.

Mariner, P. E., W. P. Gardner, G. E. Hammond, S. D. Sevougian and E. R. Stein (2015). Application of Generic Disposal System Models. FCRD-UFD-2015-000126, SAND2015- 10037 R. Albuquerque, New Mexico, Sandia National Laboratories.

Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12: 2825-2830.

Radulescu, G. (2011). Radiation Transport Evaluations for Repository Science, Letter Report. ORNL/LTR-2011/294. Oak Ridge, Tennessee, Oak Ridge National Laboratory.

Rasmussen, C. E. and C. K. I. Williams (2006). Gaussian Processes for Machine Learninged., MIT Press.

Santner, T., B. Williams and W. Notz (2003). The Design and Analysis of Computer Experimentsed. New York, New York, Springer.

Sevougian, S. D., E. R. Stein, T. LaForce, F. V. Perry, T. S. Lowry, M. Nole and K. W. Chang (2019). GDSA Repository Systems Analysis FY19 Update. SAND2019-11942R. Albuquerque, New Mexico, Sandia National Laboratories.

Simpson, T. W., V. Toropov, V. Balabanov and V. F.A.C. (2008). Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come or not. Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada. AIAA Paper 2008-5802.

SKB (2006). Long-term safety for KBS-3 repositories at Forsmark and Laxemar – a first evaluation. SKB TR-06-09. Stockholm, Sweden, Svensk Kärnbränslehantering AB.

Storlie, C. B., L. P. Swiler, J. C. Helton and C. J. Sallaberry (2009). "Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models." Reliability Engineering & System Safety 94(11): 1735-1763.

Tieleman, Tijmen, and Geoffrey Hinton. "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning." University of Toronto, Technical Report (2012).