Advanced Parameter Estimation, Bayesian Uncertainty Quantification, and Surrogate Modeling for Chromatography Processes

Fortgeschrittene Parameterschätzung, Bayesianische Fehleranalyse und Surrogat Modellierung für Chromatographieprozesse

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

William Heymann

Berichter/in: Univ. Prof. Dr. rer. nat. Wolfgang Wiechert
Univ. Prof. Dr.-Ing. Andreas Jupke

Tag der mündlichen Prüfung: 20.12.2023

i. Abstract

Manufacturing medicine is hard and expensive. Manufacturing protein-based medicines can be especially difficult due to the size and complexity of the molecules. Packed bed liquid chromatography is used during the manufacturing process to remove impurities. The problem with liquid phase chromatography is that it is an expensive and difficult process to develop. The goal of chromatography modeling is to make it faster and easier to develop a chromatography process that purifies that target protein.

Chromatography modeling is difficult and the process of designing a model, designing experiments, calibrating the model, and determining the uncertainty of the model parameters is a complex and computationally intensive process. This thesis covers model calibration, parameter uncertainty, and surrogate modeling.

Model calibration is done with parameter estimation where the parameters of a model are estimated based on chromatogram data. While least squares estimation of unknown parameters is a well-established standard it can also suffer from critical disadvantages. The description of real-world systems is generally prone to unaccounted mechanisms in the models that are customarily applied, such as dispersion in external holdup volumes, and systematic measurement errors, such as caused by pump delays. In this scenario, matching the shape between simulated and measured chromatograms has been found to be more important than the exact peak positions. A new score system is demonstrated that separately accounts for the shape, position, and height of individual peaks. A genetic algorithm is used for optimizing these multiple objectives. Even for non-conflicting objectives, this approach shows superior convergence in comparison to single-objective gradient search, while conflicting objectives indicate incomplete models or inconsistent data. In the latter case, Pareto optima provide important information for understanding the system and improving experiments.

Once a model is calibrated the next step is to determine how well defined the model is. With increased dependence on modeling, it is not enough to calibrate a model to experimental data. No model is perfect, and no model can perfectly explain experimental data. What is increasingly needed is the uncertainty in the calibrated model’s parameters along with the uncertainty in the resulting chromatogram. A method is presented to determine the probability distribution of parameters for a calibrated model using Bayesian uncertainty quantification. This method incorporates experimental errors such as pump delays, pump flow rates, loading concentration, and UV measurement error. The method presented here is based on Bayes’ theorem and uses Markov Chain Monte Carlo with an ensemble sampler and covers how to build and evaluate the error model.

While the model calibration and parameter uncertainty analysis presented here work well for synthetic and industrial data, they also require a lot of computing resources. A surrogate model approximates the original model and can stand in for the real model under restricted conditions. The advantage of surrogate models is they can be tens of thousands of times faster than the original model. Construction of a surrogate model using an artificial neural network is demonstrated along with the entire network design process.

All software created for this project is freely available as open-source code on GitHub (https://github.com/modsim/CADET-Match).

ii. Zusammenfassung

Die Herstellung von Medikamenten ist schwierig und teuer, und die Herstellung proteinbasierter Arzneimittel kann aufgrund der Größe und Komplexität der Moleküle besonders schwierig sein. Festbett-Flüssigkeitschromatographie wird in Herstellungsprozessen verwendet, um Verunreinigungen zu entfernen. Problematisch ist, dass es sich bei der Flüssigphasen-Chromatographie um ein teures und schwierig zu entwickelndes Verfahren handelt. Daher besteht das Ziel der Chromatographie-Modellierung darin, die Entwicklung von Chromatographieverfahren zur Reinigung von Zielproteinen schneller und einfacher zu gestalten. Chromatographie-Modellierung ist schwierig und das Entwerfen eines Modells, das Entwerfen von Experimenten, das Kalibrieren des Modells und das Bestimmen der Unsicherheit der Modellparameter ist ein komplexer und rechenintensiver Prozess. In diesem Kontext befasst sich diese Arbeit mit Modellkalibrierung, Parameterunsicherheit und Ersatzmodellierung.

Die Modellkalibrierung erfolgt, indem die unbekannten Modellparameter auf der Grundlage von Chromatogrammdaten geschätzt werden. Die Kleinste-Quadrate-Schätzung unbekannter Parameter ist etablierter Standard, hat aber auch kritische Nachteile. Die Beschreibung realer Systeme ist im Allgemeinen anfällig für nicht berücksichtigte Mechanismen in den üblicherweise verwendeten Modellen, wie z. B. Streuung in externen Totvolumina und systematischen Messfehlern, wie sie z. B. durch Pumpenverzögerungen verursacht werden. In diesem Szenario hat sich herausgestellt, dass die Übereinstimmung der Form von simulierten und gemessenen Chromatogrammen wichtiger ist, als die exakten Peakpositionen. Deshalb wird ein neues Bewertungssystem vorgestellt, welches die Form, Position und Höhe einzelner Peaks separat berücksichtigt. Ein genetischer Algorithmus wird verwendet, um mehrere resultierende Ziele zu optimieren. Selbst für nicht widersprüchliche Ziele zeigt dieser Ansatz eine überlegene Konvergenz im Vergleich zur Einzelziel-Gradientensuche, während widersprüchliche Ziele auf unvollständige Modelle oder inkonsistente Daten hindeuten. Im letzteren Fall liefern Pareto-Optima wichtige Informationen zum Verständnis des Systems und zur Verbesserung von Experimenten.

Sobald ein Modell kalibriert ist, muss als Nächstes bestimmt werden, wie gut das Modell definiert ist. Mit zunehmender Abhängigkeit von der Modellierung reicht es nicht aus, ein Modell auf experimentelle Daten zu kalibrieren. Kein Modell ist perfekt, und kein Modell kann experimentelle Daten perfekt erklären. Deshalb muss zunehmend auch die Unsicherheit in den Parametern des kalibrierten Modells betrachtet werden, zusammen mit der Unsicherheit im resultierenden Chromatogramm. Es wird eine Methode vorgestellt, um die Wahrscheinlichkeitsverteilung der Parameter eines kalibrierten Modells unter Verwendung der Bayesianischen Unsicherheitsquantifizierung zu bestimmen. Dieses Verfahren berücksichtigt experimentelle Fehler wie Pumpenverzögerungen, Pumpendurchflussraten, Beladungskonzentration und UV-Messfehler. Die hier vorgestellte Methode basiert auf dem Satz von Bayes und verwendet Markov Chain Monte Carlo mit einem Ensemble-Sampler und beinhaltet, wie das Fehlermodell erstellt und ausgewertet wird.

Während die hier vorgestellte Modellkalibrierung und Parameterunsicherheitsanalyse für synthetische und industrielle Daten gut funktionieren, erfordern sie auch viele Rechenressourcen. Ein Surrogatmodell approximiert das Originalmodell und kann unter eingeschränkten Bedingungen das reale Modell ersetzen. Der Vorteil solcher Ersatzmodelle besteht darin, dass sie 10.000-mal schneller sein können, als das Originalmodell. Die Konstruktion eines Ersatzmodells unter Verwendung eines künstlichen neuronalen Netzwerks wird zusammen mit dem gesamten Netzwerkdesignprozess vorgestellt.

Die gesamte für dieses Projekt erstellte Software ist als Open-Source-Code auf GitHub (https://github.com/modsim/CADET-Match) frei verfügbar.

iii. Publications

Beck, J., Heymann, W., Lieres, E. von, & Hahn, R. (2020). Compartment Model of Mixing in a Bubble Trap and Its Impact on Chromatographic Separations. Processes, 8(7), 780. https://doi.org/10.3390/pr8070780

Diedrich, J., Heymann, W., Leweke, S., Hunt, S., Todd, R., Kunert, C., Johnson, W., & von Lieres, E. (2017a). Multi-state steric mass action model and case study on complex high loading behavior of mAb on ion exchange tentacle resin. Journal of Chromatography, 1525, 60–70. https://doi.org/10.1016/j.chroma.2017.09.039

Heymann, W. (2016). Extending the CADET chromatography simulator towards process chains and networks (p. 79 p.) [MS, RWTH Aachen]. https://juser.fz-juelich.de/record/824776

Heymann, W., Glaser, J., Schlegel, F., Johnson, W., Rolandi, P., & von Lieres, E. (2022). Advanced score system and automated search strategies for parameter estimation in mechanistic chromatography modeling. Journal of Chromatography A, 1661, 462693. https://doi.org/10.1016/j.chroma.2021.462693

Kumar, V., Leweke, S., Heymann, W., von Lieres, E., Schlegel, F., Westerberg, K., & Lenhoff, A. M. (2021). Robust mechanistic modeling of protein ion-exchange chromatography. Journal of Chromatography A, 1660, 462669. https://doi.org/10.1016/j.chroma.2021.462669

Heymann, W., Glaser, J., Schlegel, F., Johnson, W., Rolandi, P., & von Lieres, E. (Not Yet Published) Advanced Error Modeling and Bayesian Uncertainty Quantification in Mechanistic Chromatography Modeling

iv. Thesis Content

Most of Chapter 2 comes from my published paper “Advanced score system and automated search strategies for parameter estimation in mechanistic chromatography modeling.” Most of Chapter 3 comes from my “Advanced Error Modeling and Bayesian Uncertainty Quantification in Mechanistic Chromatography Modeling “ paper which is not yet published. Chapter 1 contains information from both papers. The information in chapter 4 does not come from any previously published paper or in-progress paper.

v. Contents

vi. List of Figures

vii. List of Tables

1. Introduction

In 2021 the Biotechnology market was worth about $1 trillion USD and by 2028 it is expected to grow to about $2.4 trillion USD [1]. Agencies like the FDA have also been pushing for Quality by Design (QbD) which focuses on designing a quality into a product by understanding the product and process along with the risks involved in manufacturing the product and the mitigation of those risks. This has led to the development of digital twins which are essentially computer models of a physical system designed to match how a real physical system works in order to better understand and control the physical system. The FDA and other regulatory agencies have been moving in the direction of digital twins to support regulatory filings and operation [2]. For digital twins to work they must be calibrated to experimental data, allow modeling of process errors, and run quickly. This is effectively the scope of this thesis with parameter estimation, uncertainty quantification, and surrogate modeling. While this thesis focuses exclusively on packed bed liquid chromatography the same technology developed in this thesis is already successfully being used to help build digital twins for other types of applications.

All the code for this thesis is publicly available on GitHub, https://github.com/modsim/CADET-Match, and is free for anyone to use. Documentation is available at https://cadet.github.io/CADET-Match for all the features of the software.

The core reason for why all of this is done comes down to money and time to market. Biological drug experiment is extremely expensive with the typical R&D cost of bringing a new drug to market ranges from $314 million USD to $2.8 billion USD [3]. This is also why many medicines are extremely expensive. The experiments are expensive to run, and it is not feasible to run enough experiments to cover every possibility. We don't use optimal manufacturing processes because there are too many possible parameters to check and instead, we find regions that are good enough and can be shown to operate safely and reliably. The goal of modeling is to bring these costs down and allow more focused experiments. Once a computer model is calibrated billions of experiments can be run under a wider range of conditions than it will ever be possible to test in a lab and find promising areas for lab experiments. With increasing computer power more advanced modeling has become feasible which has made the development of digital twins practical. To keep costs under control and increase safety we need accurate computer models as part of the development process.

Digital twins do have a problem and it is not simple to solve. Digital twins are based on the same underlying physics, to the best of our knowledge, as their real counterparts and reality does not provide an easy way to just look the parameters up. For a digital twin to work it must first be calibrated with parameter estimation to fit the model to experimental data. These experiments often must be carefully chosen to identify all the parameters of interest. The parameter estimation process needed new methods that could handle more complex models and better fit with modern computer hardware. An emphasis was placed on methods that can run in parallel given that chips are now available with hundreds of cores, while single core performance has mostly stagnated for more than a decade.

Once the model has been calibrated the uncertainty of the parameters and the model predictions must be quantified before the model can be relied upon. Quantifying the uncertainty in the model is not a simple process. The models used are usually very complex and non-linear. This makes determining the uncertainty of the parameters a difficult process and, for reasons that will be discussed later, while existing methods are suitable existing error models are unsuitable. An error model was needed that could easily incorporate experimental errors such as variations in the concentration of different components or variations in the fluid flow rate. The error model also needed to be extensible to handle new types of models and new types of errors.

The final piece is surrogate modeling. Even with all the computing power that we have available these newer generation of models can be very computationally expensive to evaluate. A surrogate model is designed to approximate the full model but at a reduced computational cost. A model is a function that given a set of parameters it generates some output. Artificial Neural Networks (ANN) can be universal function approximators, depending on the configuration. So long as a reasonable operating range can be determined an ANN can be trained as a surrogate and it can run hundreds of thousands of times faster than the full model while also retaining sufficient accuracy.

1.1. Thesis Scope

The scope of this thesis is to create open-source software for parameter estimation and uncertainty quantification for packed bed liquid chromatography. The software needs to be robust and largely automatic. It must work on a range of synthetic and experimental data sets. Research needs to be done on how to accomplish these objectives. For parameter estimation this involves research into search algorithms, loss functions, variable transformations, and supporting methods. For uncertainty quantification a model is needed that can incorporate physical sources of error and determine parameter uncertainty along with the resulting uncertainty in the predicted chromatogram. Surrogate modeling needs to be investigated to determine which method of surrogate modeling is appropriate and to create a working surrogate model if possible.

1.2. Literature Review

1.2.1. Parameter estimation and uncertainty quantification

Parameter estimation is not an end to itself. A calibrated model only matters to the extent it can add value to its defined purpose. Once a model has been calibrated it can be used to design an optimal separation process [4], [5]. A model can also be used to see the tradeoffs between productivity and yield to design processes optimized for previously defined goals [6]. Additionally, models can be used to understand the underlying physics and the impact of effects, such as surface diffusion [7].

Early methods for calibrating models used algebraic approximations and frontal experiments in 1983 for the mass transfer and binding [8], [9]. As more complex binding models like steric mass-action (SMA) [10] where published the calibration approaches became more complex and a combination of breakthrough and pulse experiments were integrated into the calibration process [11]. Early methods still tended to assume systems are always in equilibrium for computational reasons but did start to integrate non-pore penetrating pulses and pore-penetrating but non-binding pulses to find the interstitial and particle porosity [12]. However, due to the coarse nature of the models used, rate models were mostly insensitive to diffusion parameters, which can be explained by high numerical diffusion.

The next stage of calibrating models used rate-based models with decreased numerical diffusion more frequently and mostly manual fitting with some computer assistance. Early rate-based models for the mass transfer were implemented with equilibrium binding models and while most parameters where still manually calculated some numerical fitting started to be used [13]. As systems became more complicated the fitting process started to become more complex with it, such as handling multi-component competitive binding using Lagrange multipliers [14]. Other methods continued to try to efficiently solve problems given the computer resources of the time, such as using the perturbation method [15].

As computing power further increased fewer assumptions had to be made and it was possible to fit multiple experiments simultaneously with overlapping components [16]. This marked a major change because it showed that directly fitting to experimental data gave superior results to previous methods using breakthrough curves. Direct fitting allowed removing assumptions, for example that the binding capacity for each component was the same. It became more common to use a combination of breakthrough and pulse experiments with rate-based binding and transport models [5], [17]. While it appears methods get more effective and efficient in terms of fitting models to data, the fact remains that they require good starting points for the gradient descent algorithms used in the fitting and the models are still quite coarsely discretized due to limited computing resources available.

As more computing power became available it drove the development of other methods to find suitable model parameters. A data-based approach using a support vector machine (SVM) and quantitative-structure-property-relationship (QSPR) is used to predict binding properties [18]. Other methods use batch experiments to directly measure film diffusion and binding parameters instead of having to rely on parameter estimation [19].

Computing power continued to rapidly advance to the point that finer grained models can be used for single column systems. One of the drawbacks of gradient descent is that accurate gradients are needed which can be computationally expensive to approximate or the entire model needs to have analytical derivatives. An iterative approach was used with finite difference using complex numbers. This approach requires modifying a model such that complex numbers can be used [20]. As solvers get more accurate and computers get more efficient, fitting diffusion parameters is used to better understand diffusion inside beads with fitted pore diffusion in the range we measure today [21]. Even as solvers get more accurate for single column systems, for multi-column systems with multiple components older methods of model calibration using algebraic assumptions again are used [22].

With increasing compute power, activities in development of the parameter estimation methods increased. While inverse fitting is difficult and gets more complex as model complexity increases, the results are superior to older methods that use empirical correlations and are also shown to be insensitive to UV noise but extremely sensitive to time offsets [23]. Often sum of squared errors are used as an estimation objective. The drawback of sum of squared errors is that it is extremely sensitive to time offsets of peaks and not as sensitive to the shape of the chromatograms. One way to deal with this is to use a weighted sum of mean, standard deviation, and skewness of the chromatograms which provides more sensitivity to the overall shape and more tolerance of non-overlapping starting points and is an important stepping point towards the methods presented in this thesis [24].

Variable transforms for parameter estimation are missing from earlier papers. Parameters of different scales slow optimizer performance. A major step forward was combining variable transforms, fitting simultaneous experiments using a combination of breakthrough and pulses, modeling the external column effects using a combination of PFR and CSTR, and using gradient descent [25]. However, even with all these advances fitting of many models remains difficult [26], [27].

Finally, we move towards more advanced and robust but computationally expensive methods. With more computing power available using a global optimization algorithm like a genetic algorithm and using gradients for local refinement further improves parameter estimation by removing the need to have a good initial starting point [28], [29]. An alternative way to solve parameter estimation problems that is still in its infancy is using neural networks [30]. The basic idea is to use a model to sample the space and then train a neural network using the model output chromatogram as the input and the output of the network as the parameters. An unknown chromatogram based on the same model can then be provided as input for the neural network and the parameters directly obtained as output without any estimation. Depending on the data used to train the network and the quality of the network this approach has the potential to dramatically shorten estimation times or provide good starting points for refinement.

A common thread that weaves through all the history here is that creating calibrated models is complex but necessary. Many of the parameters are correlated and not intuitively disentangled from each other and from extra column effects and necessitated not only multiple experiments but also a stagewise estimation process [31]. This is what sets the stage for a robust and automated parameter estimation process.

1.2.2. Neural Networks

In 1958 the first artificial neural network (ANN) was created by psychologist Frank Rosenblatt [32]. This first network was called a perceptron and was a simple binary classifier. It wasn't until 1965 that the first multilayer network was created [33]. Today the layers of a multilayer perceptron are called dense layers. A dense layer connects the output of every neuron in a layer to input of all neurons in the next layer. In 1973 backpropagation was first used to train a time-lag controller [34]. Backpropagation adapts the parameters of a system based on applying the chain rule to the derivative of the loss function with respect to each of the parameters, with the math explained in chapter 4.2. In 1980 the first convolutional layer was created [35] and patterned after how an eye works. Convolutional layers do not fully connect the output of all neurons to the inputs of every neuron in the next layer. This reduces the computational burden substantially because every connection also increases the size of the matrixes used in the calculation of the network. It wasn't until 1982 that backpropagation was used to train a neural network [36], and this is how neural networks today are still trained. In 1989 it was shown that a multi-layer fully connected dense network is a universal function approximator [37] which can approximate any continuous function.

Unfortunately, ANN where still not very useful at this point. The networks designed where small by today's standards and training was difficult due to the computing time required. ANN are mostly matrix operations and CPUs are just not very fast at those operations even today. In the mid-90s 3D accelerator cards started to become available and today we call those Graphics Processing Units (GPU). At their core almost all video games are built up from triangles that a stretched, rotated and shaded and these operations can be written as matrix operations. In 1998 a neural network was trained on a GPU for the first time [38]. The calculations were performed by converting the matrixes and operations needed for an ANN into graphics textures and commands for the video card to perform using the OpenGL graphics library standard. The ANN ran about 5 times faster on the GPU than on a CPU.

It wasn't long after that before libraries started to appear for video cards to make running neural networks easier. Nvidia released the Cg library in 2003 which was designed to compile more complex shader programs for video games, but it was also used to compile neural network functions for a video card [39]. Compute Unified Device Architecture (CUDA) was released in 2007 and changed neural networks on GPUs. CUDA was designed to work with a graphics card as a matrix processing device without having to convert the math needed for ANN to Graphical User Interface library calls.

With CUDA research into neural networks quickly accelerated with many more applications and network designs. GPUs were shown to work for training dense neural networks with 10 to 2500 neurons per layer and 3 to 10 layers using backpropagation in 2010 [40]. Up to this point most networks consisted of three layers with an input layer, output layer, and one hidden layer i.e., a layer that is not an input or output. Deep learning is characterized by more than one hidden layer and was introduced in 2016 [41]. Before this most networks where had one hidden layer with many neurons. The more neurons a layer has the larger the matrixes are which can be a computational and memory problem. Deep learning uses smaller matrixes and more of them. In 2020 it was shown that convolutional neural networks are also universal function approximators [42], [43].

ANN also have been used in chromatography. The usage of neural networks for model calibration in chromatography was first published in 2017 [30]. Normally the input of a neural network is thought of as your model parameters and the output as dependent variable. However, this is not a requirement. In 2021 a neural network for chromatography was trained with chromatograms as input and the output as the model parameters [44]. This allows extremely rapid parameter estimation once the network is trained.

1.3. Nomenclature

All the nomenclature used in this thesis is contained in this chapter for easy reference. Table 1 contains all the nomenclature needed for chromatography modeling. Table 2 contains the nomenclature for the parameter estimation and uncertainty quantification software, CADET-Match, created for this thesis. Table 3 contains the nomenclature for the neural network.

Table 1: Chromatography model parameters.

Symbol	*Description*	*Unit*
z	Axial column position	m
r	Radial particle position	m
t	Time	s
N_c	Number of chemical components	-
N_d	Number of discrete time points	-
N_p	Number of estimated parameters	-
Q	Volumetric flow rate	m^3/s
A_Ω	Cross-section area	m^2
u_Ω	Linear velocity	m/s
L_Ω	Length	m
r_p	Particle radius	m
c_i^κ	Solute concentration	mol/m^3
c_i^{in}	Inlet concentration	mol/m^3
$\bar{c}_0^s$	Available counter ion concentration	mol/m^3
$c_{r,i}^p$	Pore phase reference concentration	mol/m^3
$c_{r,i}^s$	Stationary phase reference concentration	mol/m^3
ε_c	Column porosity	-
ε_p	Particle porosity	-
D_Ω	Axial dispersion	m/s^2
D_p	Pore diffusion	m/s^2
k_f	Film diffusion	m/s
$k_{a,i}$	Adsorption rate	$(m^3/mol)^{\nu_i}/s$
$k_{d,i}$	Desorption rate	$(m^3/mol)^{\nu_i}/s$
$\tilde{k}_{a,i}$	Scaled adsorption rate	$1/s$
$\tilde{k}_{d,i}$	Scaled desorption rate	$1/s$
$k_{eq,i}$	Equilibrium constant	-
ν_i	Characteristic charge	-
σ_i	Shielding coefficient	-

Λ	Ionic capacity	mol/m^3

In the above table, the index $\kappa \in \{b, p, s, t\}$ refers to the interstitial column or bulk liquid (b), particle pores (p), stationary phase (s) or tubing (t) of the system. The index $\Omega \in \{c, t\}$ refers to the chromatography column (c) or tubing (t). The linear flow rate is the ratio of the volumetric flow rate and the cross-section area ($u_\Omega = Q/A_\Omega$).

Table 2: Arrays, vectors, and variables in CADET-Match.

Symbol	*Description*	*Unit*
$X_{i,j}$	Measured chromatogram	mol/m^3
$\dot{X}_{i,j}$	Time derivative of X	$mol/m^3/s$
$S_{i,j}$	Smoothed chromatogram	mol/m^3
$Y_{i,j}$	Simulated chromatogram	mol/m^3
$Y_{i,j}^{\tau}$	Simulated chromatogram (time-shifted and interpolated)	mol/m^3
$\dot{Y}_{i,j}$	Time derivative of Y	$mol/m^3/s$
$\bar{X}_i$	Mean of X	mol/m^3
$\bar{Y}_i$	Mean of Y	mol/m^3
σ_{X_i}	Standard deviation of X_i	mol/m^3
σ_{Y_i}	Standard deviation of Y_i	mol/m^3
$cov(X_i, Y_i)$	Covariance between X_i and Y_i	mol^2/m^6
t_s	Time offset in shape sensitive metrics	s
t_r	Non-binding retention time (one column volume)	s

The arrays and vectors in the above tables depend on indices $i = 1, \dots, N_c$ for the component and $j = 1, \dots, N_d$ for the discrete time points. Several metrics are defined on subsets $J \subset \{1; \dots; N_d\}$ of the data points, which usually belong to a coherent time interval where the chromatogram displays a specific feature. The mean, variance and covariance are defined as follows, with $|J|$ denoting the number of elements in the index set J:

$$\bar{X}_{i,J} = \frac{1}{|J|}\sum_{j\in J} X_{i,j} \qquad \bar{Y}_{i,J} = \frac{1}{|J|}\sum_{j\in J} Y_{i,j}$$

$$\sigma_{X_{i},J} = \sqrt{\frac{1}{|J|}\sum_{j\in J}\left(X_{i,j} - \bar{X}_{i,J}\right)^2} \quad \sigma_{Y_{i},J} = \sqrt{\frac{1}{|J|}\sum_{j\in J}\left(Y_{i,j} - \bar{Y}_{i,J}\right)^2}$$

$$cov(X_i, Y_i)_J = \frac{1}{|J|}\sum_{j\in J}\left(X_{i,j} - \bar{X}_{i,J}\right)\left(Y_{i,j} - \bar{Y}_{i,J}\right)$$

Table 3: Neural network nomenclature.

Symbol	*Description*
n	neuron index at current layer
n^*	neuron index at next layer
l	layer index
f	activation function
x_n	input value
x	input array
$a^{(l)}$	activation layer
$W^{(l)}$	weight layer

$W_{n^*,n}^{(l)}$	weight value
$b^{(l)}$	bias layer
$b_n^{(l)}$	bias value
$z^{(l)}$	dense layer
y_{pred}	output layer prediction
y_{true}	output layer ground truth
y_n	output layer length
r	learning rate

1.4. Model Introduction

This thesis addresses packed bed liquid chromatography at preparative scale. Such systems can be described in-silico by combining different models of the governing transport and binding processes. The general rate model (GRM), eq. 1-2, with suitable boundary conditions, eq. 3-6, and non-equilibrium SMA binding, eq. 7-10, is a common choice. The GRM describes convection and dispersion in the interstitial column volume, diffusion in the porous particles, and binding to the inner surfaces of these particles.

$$\frac{\partial c_i^b}{\partial t} = -u_c \frac{\partial c_i^b}{\partial z} + D_c \frac{\partial^2 c_i^b}{\partial z^2} - \frac{1-\varepsilon_c}{\varepsilon_c} \frac{3}{r_p} k_f \left(c_i^l - c_i^p(r = r_p)\right) \tag{1}$$

$$\frac{\partial c_i^p}{\partial t} = D_p \left(\frac{\partial^2 c_i^p}{\partial r^2} + \frac{2}{r} \frac{\partial c_i^p}{\partial r}\right) - \frac{1-\varepsilon_p}{\varepsilon_p} \frac{\partial c_i^s}{\partial t} \tag{2}$$

$$D_c \frac{\partial c_i^b}{\partial z}(z = 0) = u_c \left(c_i^b(z = 0) - c_i^{in}\right) \tag{3}$$

$$\frac{\partial c_i^b}{\partial z}(z = L_c) = 0 \tag{4}$$

$$\frac{\partial c_i^p}{\partial r}(r = r_p) = \frac{1}{\varepsilon_p D_p} k_f \left(c_i^b - c_i^p(r = r_p)\right) \tag{5}$$

$$\frac{\partial c_i^p}{\partial r}(r = 0) = 0 \tag{6}$$

$$\frac{dc_i^s}{dt} = \tilde{k}_{a,i} c_i^p \left(\frac{\bar{c}_0^s}{c_{r,i}^s}\right)^{\nu_i} - \tilde{k}_{d,i} c_i^s \left(\frac{c_0^p}{c_{r,i}^p}\right)^{\nu_i} \tag{7}$$

$$k_{a,i} = \tilde{k}_{a,i} \left(c_{r,i}^s\right)^{-\nu_i} \tag{8}$$

$$k_{d,i} = \tilde{k}_{d,i}\left(c_{r,i}^{p}\right)^{-\nu_i} \tag{9}$$

$$\bar{c}_0^s = \Lambda - \sum_{i=1}^{N_c}(\nu_i + \sigma_i)c_i^s \tag{10}$$

The concentrations c_i^κ with $\kappa \in \{b, p, s\}$ refer to the interstitial column or bulk liquid (b), particle pores (p) and stationary phase (s) of the system. Chromatography models depend on many parameters, several of which need to be estimated by fitting simulated chromatograms to experimental data. In eq. 1-6 the column porosity, ε_c, particle porosity, ε_p, axial dispersion, D_c, film diffusion, k_f, pore diffusion, D_p, adsorption rate, $k_{a,i}$, desorption rate, $k_{d,i}$, shielding coefficient, σ_i and characteristic charge, ν_i, of component $i = 1, \dots, N_c$ are typically estimated from measured chromatograms. Film and pore diffusion can generally differ between components but are assumed to be identical for the specific molecules used in this study. Other parameters such as the column length, L_c, particle radius, r_p, interstitial velocity, u_c, and ionic capacity, Λ, can be controlled or measured in advance of the simulation. The same is true for the initial concentrations, $c_i^\kappa(t = 0)$, and inlet concentration profiles, c_i^{in}.

The SMA model, as originally introduced by Brooks and Cramer [10], becomes numerically unstable for molecules with high characteristic charge, ν, such as monoclonal antibodies on high capacity resins. This critically important problem is effectively avoided by scaling the rate constants by the νth power of reference concentrations c_r^p and c_r^s [45]. Recommended values are the highest salt concentration in the feed during elution for c_r^p, and the ionic binding capacity of the resin for c_r^s. This ensures both terms raised to the νth power in eq. 7 range between 0 and 1, while without scaling they can become extremely large and consequently cause accuracy loss and instability of the numerical solver. Moreover, the units of the scaled rate constants are independent of the characteristic charge, which is beneficial for the physical interpretation of these values and for the performance of search algorithms during parameter estimation.

The tubing is modeled using a dispersive plug flow reactor (DPFR), eq. 11. The tubing model is solved with the same inlet and outlet boundary conditions, as the column model, eq. 3-4, with the tubing length, L_t, in place of the column length, L_c.

$$\frac{\partial c_i^t}{\partial t} = -u_t \frac{\partial c_i^t}{\partial z} + D_t \frac{\partial^2 c_i^t}{\partial z^2} \tag{11}$$

In the tubing, c_i^t denotes concentration, u_t velocity and D_t axial dispersion. A continuously stirred tank reactor (CSTR) is used to model mixers and mixing effects. In addition to the model equations presented here, CADET-Match works with any combination of transport and binding models that is covered by the CADET solver, which is an independent and continuously extended open-source project (https://github.com/modsim/CADET).

1.5. Data smoothing

Experimental data is often noisy and even synthetic data can be noisy depending on the simulator tolerances. The methods discussed in this thesis are shape sensitive and often require not only a smooth signal but a smooth first derivative. Hence, the data needs to be smoothed to reduce the noise. For routine application in industrial workflows, the smoothing needs to be automatic and work robustly, i.e., not attenuate relevant features, without human involvement. This is complicated by the situation that an experiment with detached column typically results in a signal length on the order of seconds, a non-binding but pore penetrating tracer pulse on the order of minutes, and a gradient elution experiment on the order of hours. Thus, the automated smoothing method must work on data of very different time scales and still reliably isolate a signal from the noise that retains all relevant features and is smooth enough to obtain accurate first derivatives.

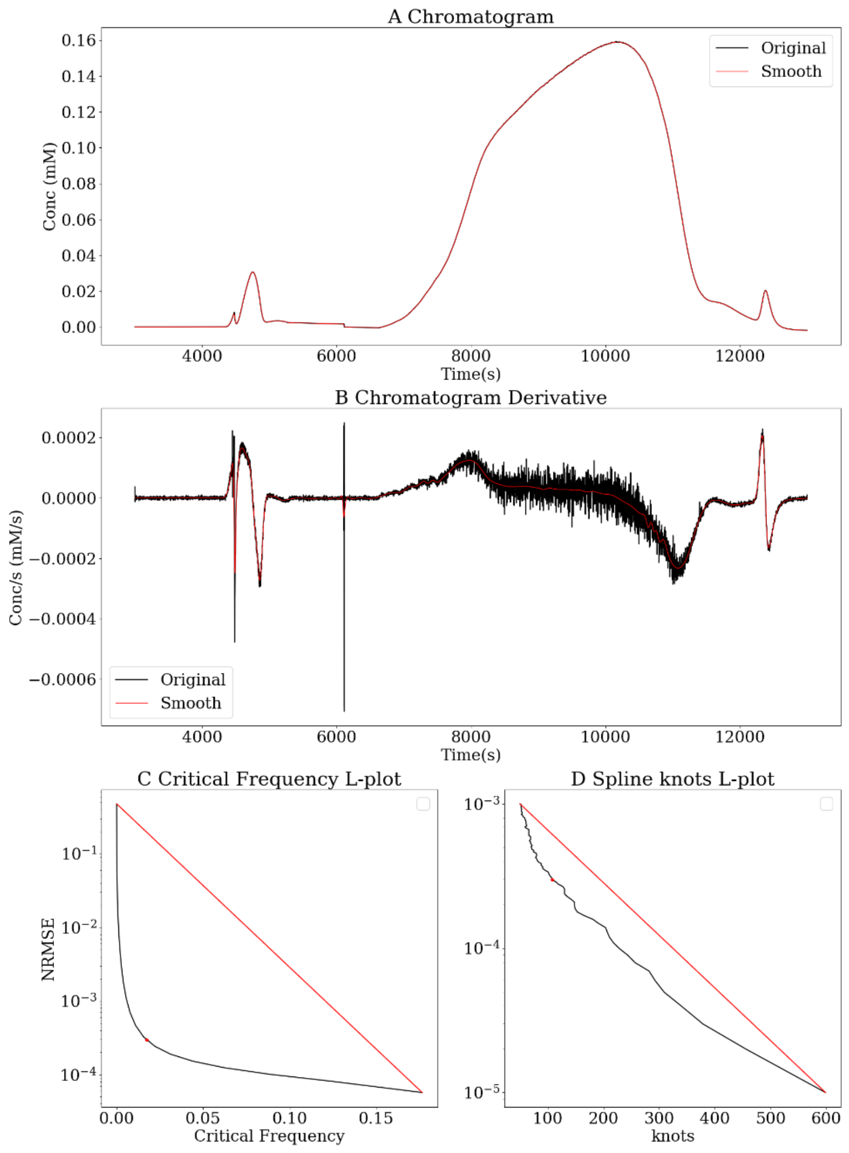

Figure 1: Original and smoothed signal (A) and first derivative (B) of an example chromatogram taken from (Diedrich, et al., 2017), critical frequency (C) and spline knots (D).

Most high frequency noise removal strategies fall into a few general categories. The Fast Fourier Transform (FFT) [46] converts a signal to frequency space where high frequency ranges can be removed before the signal is converted back to the original space. FFT filters are fast and suitable for removing high frequency noise. They can be robustly automated and include a simple method for computing the first derivative. Moving average filters and windowed polynomial regression, such as the Savitzky-Golay filter [47], can also locally approximate the signal with reduced noise. These methods depend on several parameters for controlling the window width and smoothing factors. Such filters can work quite well with humans choosing these parameters but are generally more difficult to robustly automate as compared to splines with only one parameter. Splines [48] are another alternative for reconstructing signals from noisy observations, and they can also provide first derivatives. However, splines can become computationally expensive due to an increased number of required knots for signals with high frequency noise. The required knot number can be automatically determined as will be detailed in the next paragraph.

The following procedure is a robust-automated smoothing procedure for noisy chromatograms and implemented in CADET-Match. The procedure is illustrated using a previously published chromatogram [45] shown in Figure 1A. This example is typical for industrial data and the signal contains particularly small and large features. Extensive testing revealed that no single choice of the above-described methods can automatically and robustly remove measurement noise from chromatograms without filtering out relevant features over a wide range of time scales. However, satisfactory results were obtained by combining FFT filters with a spline.

To measure the impact of smoothing on the signal Normalized Root Mean Squared Difference (NRMSD) shown in eq. 12 is used. RMSD has the advantage of giving an error in the same units as the original signal and it independent of the number of points. By normalizing the RMSE the measurement becomes scale independent. For example, an NRMSD of 1e-2 indicates an error on the order of 1% of the peak maximum regardless of the actual value of the peak maximum. Measuring the error this way greatly simplifies the process of targeting an acceptable level of error to smooth the signal.

$$NRMSD_j = \sqrt{\frac{\sum_{i=1}^{N}\left(S_{i,j} - X_{i,j}\right)^2}{N * \max\limits_{i=1,\dots,N}\left(X_{i,j}\right)^2}} \tag{12}$$

The choice of what kind of FFT filter to use is also important. There are two options in scipy.signal [49] that are appropriate, the Butterworth filter [50] and the Bessel filter [51]. The Butterworth filter is smoother while the Bessel filter preserves the shape of filtered signals with little oscillation on a sharp transition. Both filters, when used as a low pass filter, take a single parameter, which is the critical frequency. In most situations in chromatography the behavior of the two filters is almost identical. In the case of a sharp transition a ringing effect can occur where oscillations are seen on the transition with a butter filter. This effect is shown in Figure 2 and is a synthetic example with no noise added and it comes from a large library of testing data composed of synthetic and experimental chromatograms used for testing smoothing. The time and concentration in the plot are both normalized to 1 since this method is independent of both. As can be seen in cell A using a Butter filter with 1000 samples has large oscillations and increasing to 5000 samples as shown in cell C resolves the problem. Cells B and C use a Bessel filter instead and with 1000 and 5000 samples no ringing is seen or degradation of the signal. This makes a Bessel filter more appropriate in chromatography due to working with fewer samples which also speeds up processing and reduces memory requirements. The ringing effect can be difficult to automatically detect and can impair the performance of any algorithm using the smoothed signal. As a result, a Bessel filter is used for smoothing in this procedure.

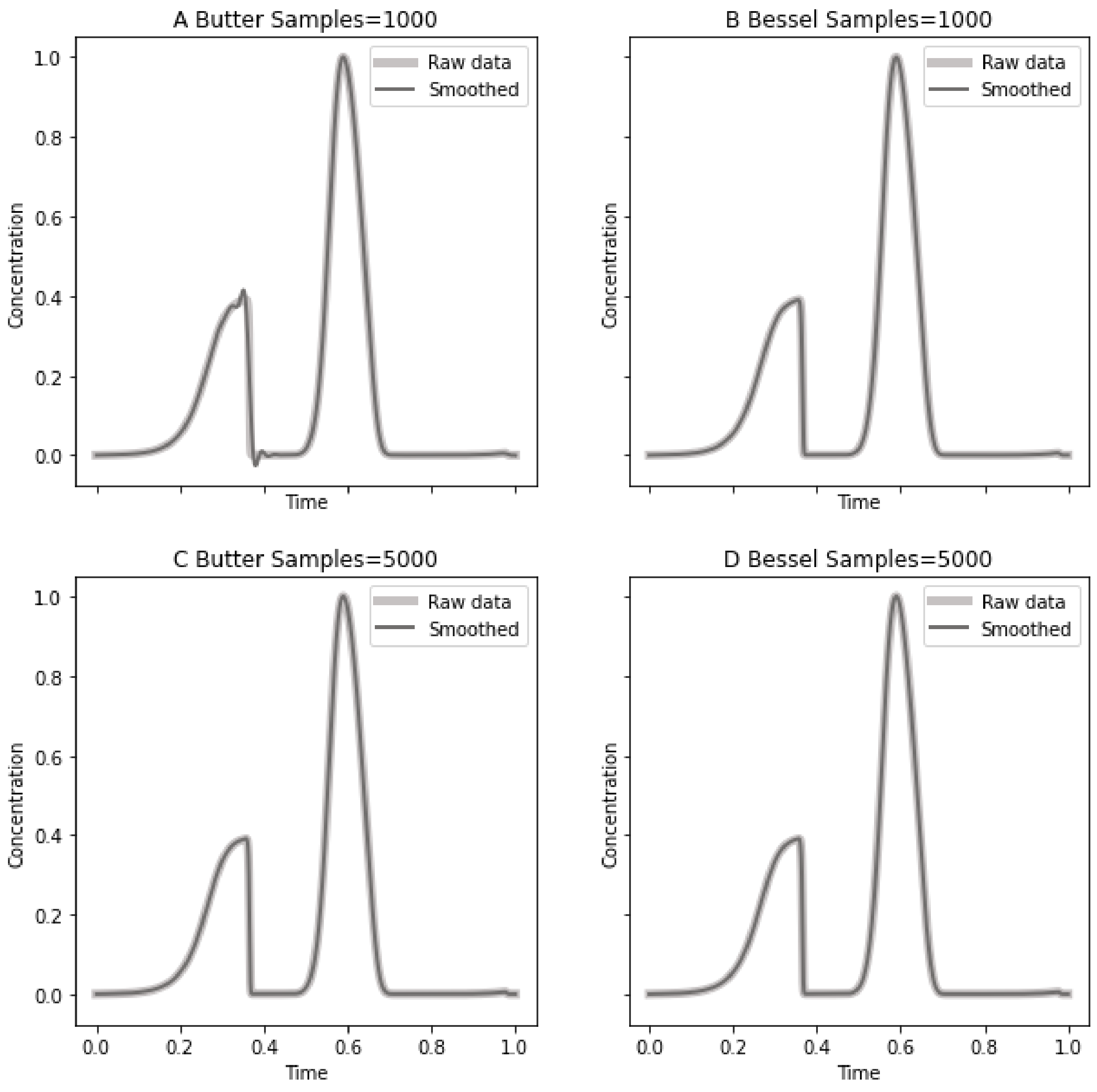

Figure 2: Ringing example plot showing Bessel and Butter filters at different sampling number of samples.

Prior to the smoothing procedure some data transformations are used which are reversed at the end. The chromatograms are normalized by dividing the concentrations of each component by the maximum concentration of that component. The data is also resampled to a uniformly spaced time grid due to FFT requirements. Chromatograms are normalized so that NRMSD can be used as a target in the smoothing process.

First, a third-order Bessel low pass filter is created for the normalized chromatogram using the scipy.signal.bessel function and applied using scipy.signal.sosfiltfilt to create a smooth non-phase-shifted signal. A suitable critical frequency for smoothing a given chromatogram is automatically determined using the so-called elbow point method. Figure 1C shows the logarithm of the normalized root mean square difference (NRMSD) between the filtered and original signals over the critical frequency of the filter. The elbow point maximizes the distance between that curve and a straight line between the extreme points. It indicates the best compromise between removing as much noise as possible while approximating the signal as accurately as possible.

Second, a 5th order spline with non-equidistant knots is applied to the low pass filtered chromatogram using the scipy.interpolate.UnivariateSpline function [49]. Cubic splines would be sufficient for approximating the original signal, but higher order splines are beneficial for computing derivatives. A smoothness factor is passed to this function, which is the upper bound for a sum of squared errors difference between the smoothed and original signal, and the minimal number of knots are used. Figure 1D shows the logarithm of the NRMSD between the approximated and original signals over the number of knots. This elbow point indicates the best compromise between using as few knots as possible while approximating the signal as accurately as possible. To the left of the elbow point, the NRMSD drops very quickly with an increasing number of knots. Then, the spline switches from smoothing to interpolating and the NRMSD decreases very slowly. Technically, a smoothness factor is passed to the SciPy function instead of the number of knots. The function internally computes the minimal number of knots for which the NRMSD falls below the specified smoothness factor.

Third, if a derivative is required, an additional pass of the Bessel filter is performed on the derivative following the same procedure as the first step. In some data sets the previous two smoothing steps are insufficient to provide a clean first derivative and this additional smoothing step is needed. If the data is exceptionally noisy even this step is not enough but in practice it is good enough for optimization and uncertainty quantification to work.

The presented smoothing procedure is routinely applied to every chromatogram, measured, or simulated. The latter allows to save compute time by using rather coarse simulator tolerances in early stages of the parameter estimation procedure that could otherwise cause numerical problems with some of the applied search algorithms. The smoothing procedure is applied once to each measured chromatogram. Iterative search strategies can involve numerous similar simulations with hardly varying elbow points for the resulting chromatograms. Hence, these points are pre-determined for a representative set of initial values and reused further on. Very smooth input data might not show distinct elbow points for the Bessel filter and/ or the spline approximation. In these cases, the respective methods are simply disabled.

1.6. Experimental Data

All the experimental data in this thesis has been provided by Amgen. This dataset has already been published[52] and is used for parameter estimation and uncertainty quantification case studies in this thesis.

Chromatographic cation exchange runs were conducted on Äkta Avant (GE Healthcare). Blue Dextran 2000 (GE Healthcare) was used for non-pore penetrating pulse experiments. A volume of 1 mL Dextran solution with a concentration of 0.002 mM was injected at a flow rate of 5 mL/min with and without column attached. These experiments were carried out in duplicates. Dextran chromatogram data was measured at UV 280nm.

Pore-penetrating as well as load, wash and elution steps were performed using a Fractogel SO_3^- (EMD Millipore) resin in a packed bed column with inner diameter 16 cm and length 25 cm at a flow rate of 5 mL/min. The column was pre-equilibrated for 3 CV with 200 mM sodium acetate buffer containing 1 M NaCl at pH 5.0. The column was equilibrated for 3 CV with 100 mM sodium acetate buffer at pH 5.

Pulse injections under non-binding conditions were run using previously purified monomeric antibody as tracer. The antibody was prepared in a 100 mM sodium acetate buffer with 500 mM NaCl at pH 5 to prevent adsorption.

For elution experiments, the column was loaded with 152 mL or 540 mL, respectively, of Filtered Virus Inactivated Pool (FVIP) of monoclonal antibody product. The material was obtained from a previous capturing step. The antibody material was produced by CHO cell culture. The FVIP material was conditioned with arginine, targeting 50 mM arginine concentration in the FVIP material. A wash step was conducted with 100 mM sodium acetate buffer at pH 5. The elution step was conducted with a linear 1 mM/CV gradient between two buffers with 100 mM sodium acetate and 100 mM sodium acetate plus 1 mM NaCl, respectively, at pH 5.0. After elution the column was cleaned using 1 M NaOH solution. Chromatogram data of the load, wash and elution experiments were measured at UV 300 nm and UV 280 nm. For the 152 mL load volume run, fractionation samples were taken at 8 mL fraction volume starting at 0.1 AU of UV 280 signal.

2. Parameter Estimation

Many parameters in chromatography models are highly correlated in the sense that small changes in different parameters impact on the simulated chromatogram in similar ways. This could technically be neglected when accurate model predictions are required only for the exact same column dimensions and operating conditions that were used for parameter estimation. However, this is not the case when the calibrated model is to be applied for guiding rational process design and scale-up.

In these applications it is crucially important that the estimated parameter values correctly represent and accurately quantify the impact of the respective underlying physical mechanism described by the model. Parameter correlations can be avoided by a staged estimation procedure that isolates these parameters using specific experiments whose design and order depends on the mathematical structure of the model equations [13]. Typically, four types of experiments are required with 1) detached column to determine the band broadening effect of extra-column volumes, 2) a non-binding tracer that does not penetrate the particle pores to determine column porosity and axial dispersion, 3) a non-binding but pore-penetrating tracer to determine particle porosity, film diffusion and pore diffusion, and 4) the target molecules to determine the parameters of the binding model.

The first stage is increasingly recognized to be crucial for determining unbiased values of all other model parameters that can be transferred across operating conditions, system configurations and scales. Neglecting extra column effects or even small errors in accounting for them can have a large impact on the binding parameters that are estimated at a later stage [53]. Extra column effects have been previously accounted for by shifting the time scale [26]. More comprehensive models comprise a series or network of dispersive plug flow reactors (DPFR) and continuously stirred tank reactors (CSTR) [25]. The respective model parameters are isolated by removing the column from the simulated system.

In the second stage, model parameters for characterizing the packed bed are isolated by setting the film diffusion coefficient to zero, which effectively eliminates eq. 2-6 from the system. Dextran is normally used as non-binding and non-pore penetrating tracer. However, this tracer often behaves non-ideally, which causes specific challenges in the parameter estimation procedure as will be discussed in section 2.3.

In the third stage, model parameters for characterizing the porous particles are isolated by setting the adsorption constant to zero, which effectively eliminates eq. 3-6 from the system. Ideally, the target molecule is used as pore-penetrating tracer under non-binding conditions, such as high salt in ion-exchange chromatography. Smaller tracers, such as acetone, are prone to overestimate the film and pore diffusion coefficients. Occasionally, the film and pore diffusion coefficients remain too correlated to be estimated independent of each other. In these cases, the pore diffusion can be determined from some other type of experiment, such as using confocal laser scanning microscopy [21]. Alternatively, a simpler transport model can be used, such as the lumped rate model with pores.

In the fourth stage, the parameters of the binding model are estimated. Due to inherent non-linearity of the more complex binding models that are required for describing preparative chromatography, this is usually by far the most complicated and time-consuming stage, in particular for complex multi-component systems with competitive binding. Base-line separation is typically not achieved in experimental data available for calibrating such models. Hence, the binding model parameters of multiple chemical components cannot be completely isolated from each other. However, correlations between the binding parameters can be reduced by estimating them from a set of several chromatograms that are measured at specifically designed operating conditions. For large molecules, such as monoclonal antibodies, this typically includes a breakthrough and two or three gradient elution experiments with varying slopes. The design of such experiments can be optimized by evaluating different estimation strategies on synthetic data that are generated using an initial guess of plausible values for the sought model parameters. Similar components, such as charge variants or high/ low molecular weight impurities, are often lumped in groups to reduce model complexity and the number of estimated parameters.

Measurement data are generally prone to random and systematic errors that are in turn propagated to the estimated parameters. In the proposed staged procedure, parameters estimated in one stage are fixed in the next and, consequently, parameter errors are carried over to subsequent stages. This can be avoided in a Bayesian approach where the posterior parameter distribution of one stage is used as prior information in the next stage, this is covered in section 3.

2.1. Parameter transformations

Most search strategies struggle when the sought parameters spread over orders of magnitude or are correlated with each other. Parameter transformations can help to soften these challenges. CADET-Match provides several transformation rules, i.e., biunique maps between model parameters, p, that are passed to the chromatography simulator and estimated parameters, p', that are passed to the search algorithm. These transformations are based on upper bounds, $\hat{p}$, and lower bounds, $\check{p}$, of the model parameters.

The linear transformation, eq. 13, maps the original range $[\check{p}, \hat{p}]$ to $[0,1]$. This is usually sufficient when the upper and lower parameter bounds are less than three orders of magnitude apart from each other. For wider ranges, a nonlinear transformation, eq. 14, is advisable. The latter automatically adapts the step width of the search algorithm to the magnitude of the respective model parameter. Otherwise, the same step could be huge for one parameter but tiny for another.

$$p = (\hat{p} - \check{p}) \cdot p' + \check{p} \tag{13}$$

$$p = \exp\big((log(\hat{p}) - log(\check{p})) \cdot p' + log(\check{p})\big) \tag{14}$$

Nonlinear parameter correlations are hard to detect and need to be specifically addressed. For instance, the adsorption and equilibrium constants, k_a and k_{eq}, are usually much less correlated than the adsorption and desorption constants, k_a and k_d. The relation $k_{eq} = k_a/k_d$ allows to pass k_a and k_d to the simulator while the search algorithm operates on k_a and k_{eq}. The corresponding transformation, eq. 15-16, also accounts for large parameter ranges. This decouples the binding rate from the concentration equilibrium.

$$k_a = \exp\Big(\big(log(\hat{k}_a) - log(\check{k}_a)\big) \cdot k_a' + log(\check{k}_a)\Big) \tag{15}$$

$$k_d = \frac{\exp\Big(\big(\log(\hat{k}_a) - \log(\check{k}_a)\big) \cdot k_a' + \log(\check{k}_a)\Big)}{k_{eq}'} \tag{16}$$

A common issue that occurs during parameter estimation on a multicomponent system with a large proteins and charge variants of that protein is that that the binding parameters are highly correlated with each other. Instead of estimating all the binding parameters independently it simplifies and speeds up the estimation process by estimating the parameters for one charge variants and having the other charge variants based on an additive or multiplier offset. A parameter, p^*, is introduced which can be any other parameter in the model. An additive offset is shown in eq. 17 and a multiplicative one is shown in eq. 18.

$$p = p^* + (\hat{p} - \check{p}) \cdot p' + \check{p} \tag{17}$$

$$p = p^* * \left((\hat{p} - \check{p}) \cdot p' + \check{p}\right) \tag{18}$$

Other parameter transformations are designed to solve specific problems that are encountered in experiments. CADET-Match uses a simple plugin system for parameter transformations where a new transformation is written in Python and then put in a specific directory. This ease of creating transformations has led to many transformations being created to solve problems as they were encountered. Parameter transformations are defined in the configuration file, and they are run in the order they are defined which allows transformations to be based on previously transformed values.

The reciprocal transform is an example of a transform that was created to solve a very particular problem related to the coupling between parameters. As will be shown later in chapter 3.3.1.3 film diffusion and particle diffusion are non-linearly coupled with each other and while this slows down parameter estimation a bit it has a much larger impact on uncertainty quantification performance. For a difference of less than three orders of magnitude eq. 19 is used otherwise eq. 20 is used. The difference can be seen in Figure 3 which shows the joint probability distribution, explained in chapter 3, between particle diffusion and film diffusion using the normal transform (A) and the reciprocal transform (B). The important thing to notice here is the non-linear relationship in A is much harder for uncertainty quantification than the linear one in B.

$$p = \left(\frac{1}{\hat{p}} - \frac{1}{\check{p}}\right) \cdot \frac{1}{p'} + \frac{1}{\check{p}} \tag{19}$$

$$p = \exp\left(\left(\frac{1}{\log(\hat{p})} - cc\right) \cdot \frac{1}{log(p')} + \frac{1}{log(\check{p})}\right) \tag{20}$$

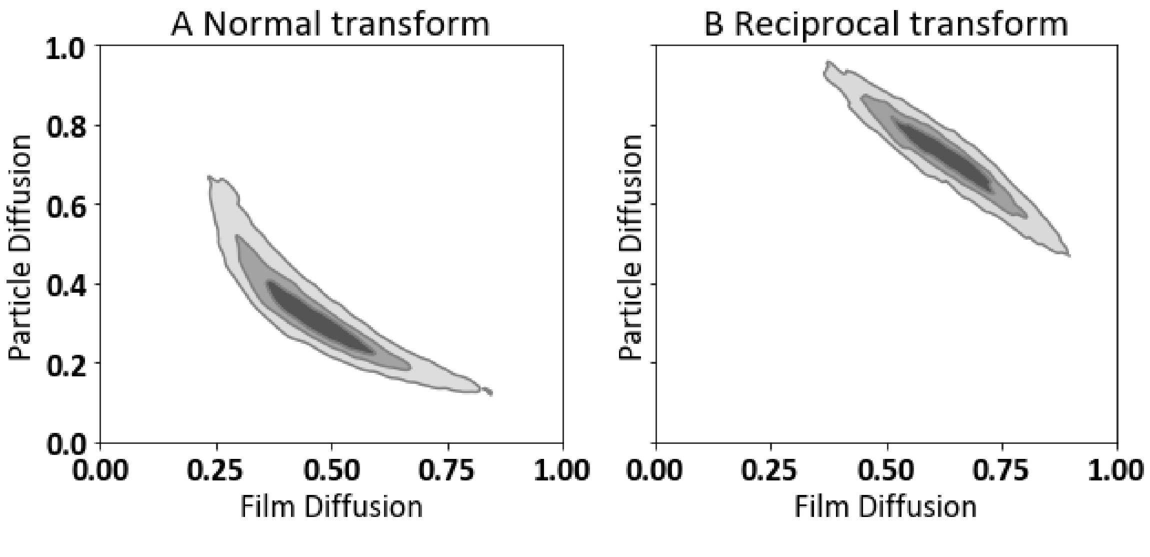

Figure 3: A shows a joint probability distribution between particle diffusion and film diffusion using the normal transform while B shows the same thing with a reciprocal transform.

2.2. Goals

We introduce a new goal system for estimating chromatography model parameters. Here, goal means a set of shape-sensitive metrics. Each metric is a single scalar value such as the time difference between simulation and measurement at peak maximum, the height difference at peak maximum, etc. Metrics are defined based on specific knowledge of the modeled process and of typical errors in the measurement data. The metrics in a goal can be passed to a multi-objective search algorithm or they can be combined into one objective and passed to a single-objective search algorithm.

Multiple metrics can guide (multi-objective) search strategies much better to the desired optimum than the commonly applied sum of squared differences (SSD) can guide (single objective) search strategies, as will be demonstrated in the results section. The metrics are grouped into scores to organize the specification of goals for different parameter estimation procedures in CADET-Match. A suitable goal must have the property that as the fit quality improves the value of at least one metric must decrease and as the fit quality worsens, the value of at least one metric must increase. A goal that does not have this property can guide search algorithms in the wrong direction. This might appear trivial but is critically important and at the core of why new goals were designed. Due to competitive binding and other complex mechanisms, many model parameters influence the simulated chromatograms in non-linearly coupled ways. Therefore, some customarily applied metrics such as SSD can increase while the model parameters move closer to their correct values.

In addition, measured chromatograms from industrial large-scale applications are often affected by systematic errors such as pump delays that can cause a time offset between the measured and simulated signals, unless the model captures the cause of the delay which is often not possible in practice. Pump delays occur when there is a difference between when a pump is given the signal to start and when it starts. This data is usually not available and thus can't be modeled. To further complicate matters pump delays may not be consistent between runs or within a run. A good goal needs to account for this, since otherwise the simulated peaks end up in the correct location but with the wrong shape. Wrong shapes generally indicate errors in the underlying physics of the model. Hence, good metrics should prefer peaks with nearly fitting shape but small offsets rather than peaks without offset but with wrong shapes.

2.2.1. Sum of Square Difference

For the SSD, the squared differences between simulated and measured chromatograms are summed up over the time points, eq. 21. For the NRMSD, the SSD is divided by the number of time points before taking the square root and dividing the result by the maximum of the measurement data, eq. 22. Due to the monotonicity of this transformation, SSD and NRMSD have the same minima. These metrics can be applied to the entire chromatogram, $J = \{1; \dots; N_d\}$, or a subset of the data, $J \subset \{1; \dots; N_d\}$. The SSD is most commonly applied with gradient descent search algorithms. Hence, it is included here for comparison. The theory is well established in the framework of maximum likelihood estimation for independent and identically distributed random measurement errors. However, these preconditions are generally not valid for modeling large-scale preparative chromatography where systematic errors such as feed variations, pump delays and flow rate variations typically dominate the detector noise. The NRMSD is better suited than the SSD for interpreting the results, because the numerator has the same unit as the data and is related to the maximum concentration by the denominator.

$$SSD(X_i, Y_i)_J = \sum_{j \in J} \left(X_{i,j} - Y_{i,j}\right)^2 \tag{21}$$

$$NRMSD(X_i, Y_i)_J = \frac{\sqrt{\frac{1}{|J|} \sum_{j \in J} \left(X_{i,j} - Y_{i,j}\right)^2}}{\max\limits_{j \in J} \left|X_{i,j}\right|} \tag{22}$$

The SSD requires a sufficient overlap between the simulated and measured chromatograms to be sensitive to parameter changes and guide the search algorithm towards the optimum. This can complicate the choice of suitable starting points, in particular for sharp and/ or small peaks. A further disadvantage of the SSD is illustrated in Figure *4* using a synthetic example with parameters shown in Table 4. The parameters of scenario 2 are much closer to the ground truth, with only a relatively small deviation in the characteristic charge, ν, even though Scenario 1 has a smaller SSD and would hence normally be considered a better fit. In addition, the peak shape of Scenario 2 is more similar to the ground truth but out of alignment.

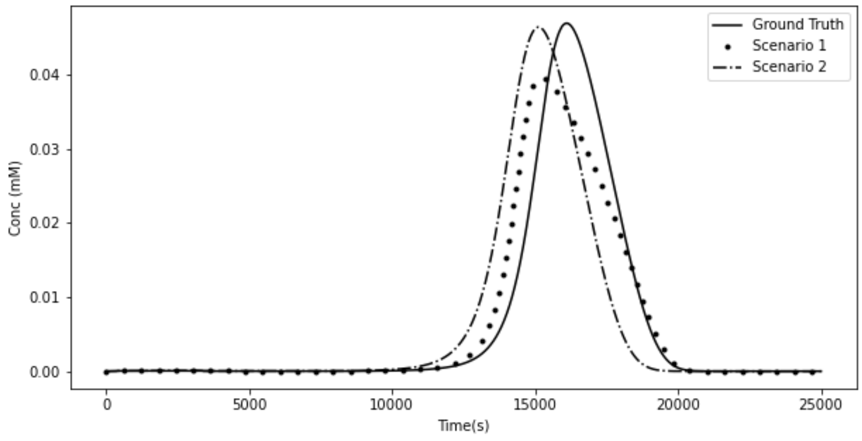

Figure 4: SSD chromatogram alignment issue illustrated by synthetic data.

Table 4: Synthetic data with resulting NRMSD for illustrating alignment issue.

	Ground Truth	*Scenario 1*	*Scenario 2*
k_a	2.00	2.9e02	2.00
k_d	10.0	3.7e03	10.0
ν	7.00	9.60	6.00
σ	50.0	99.0	50.0
SSD		4.3e+00	1.5e+01
NRMSD		4.2e-03	7.7e-03

In real experiments, such time offsets are often caused by pump delays that cannot be explained by the mechanistic model. In this case, the SSD favors peaks that are in the right position even though it is obvious to the human eye that the peak shape is wrong. The model can also reproduce the correct peak shape but not in the right position with a much larger SSD. As the peak shape is predominantly determined by the binding model parameters, the SSD would lead to unphysical parameter values. Hence, we will now introduce alternative metrics that favor peak shape over position and are less demanding on the choice of suitable starting points.

2.2.2. Alternative Metrics

The shape and position of a chromatogram are determined by mass transport through the entire system, including the column and external volumes, and binding to the functionalized resin. The disadvantages of the SSD are avoided by separately measuring the shape, position, and height of individual peaks without requiring base line separation. Metrics for peak position are sensitive to changes of the respective model parameters, independent of peak overlaps between simulation and measured data. This provides flexibility and robustness with respect to the choice of starting points for the search algorithms, which is critically important for automation in industrial applications. Focusing on individual peaks allows to reduce the impact of process variations and further components that are not fully included in the model. For example, pump washes or pressure alarms can cause spurious peaks, and industrial feeds usually contain large numbers of more or less uncharacterized impurities. In such cases, separate metrics can be assigned to distinct but partially separated peaks of target components and impurities of high and low molecular weight. Separate metrics also help to provide (multi-objective) search algorithms with more precise information on which component impacts which peak, as will be detailed in section 2.5. All metrics are designed for minimization and yield zero for a perfect match between simulation and experiment.

2.2.2.1. Peak Shape

The shape metric is the most innovative of the new metrics and a core component of nearly all scores. It is the difference between one and the maximum of the Pearson correlation [54] between measured and simulated chromatograms over a continuous range of time offsets, eq. 23. For evaluating this metric, the simulated chromatogram is shifted in time, $Y_i^\tau(t) = Y_i(t-\tau)$. The maximum in eq. 23 is determined by an initial grid search followed by Powell's method. While it is advisable for the SSD to simulate the chromatogram on the same grid as the measurement data, continuous offsets require interpolating the simulated data. This is implemented in CADET-Match using the 5th order spline from the smoothing procedure described in section 1.5. Allowing for continuous time offsets that are independent of the discrete measurement grid is crucial for creating a smooth metric. The shape metric is typically applied to individual peaks that are sliced out of the chromatogram. By design, this metric only accounts for shape similarity and requires two other metrics to measure the time offset and height difference between simulation and measurement data.

$$Shape(X_i, Y_i)_J = 1 - \max_\tau \left(\frac{cov(X_i, Y_i^\tau)_J}{\sigma_{X_i,J}\sigma_{Y_i^\tau,J}} \right) \tag{23}$$

2.2.2.2. Peak Position

The position metric can be more complex than it might first appear. It is based on the time offset, t_s, obtained from maximizing eq. 24.

$$t_s(X_i, Y_i)_J = \arg\max_\tau \left(\frac{cov(X_i, Y_i^\tau)_J}{\sigma_{X_i,J}\sigma_{Y_i^\tau,J}} \right) \tag{24}$$

The standard position metric gives an immediate penalty for a time offset with a linear ascent to one when out of alignment by t_r, eq. 25. Here, t_r is the length of the measurement time interval. It can be replaced by the retention time of a non-binding tracer if sufficient starting points are provided to the search algorithm. As will be shown in the results section, this metric it a good choice for estimating column and particle porosity. However, it requires great care in running experiments to ensure there are as few delays as possible and alarms are immediately cancelled. Such delays affect the chromatogram almost exactly like changes in the column and particle porosities. As previously discussed, in the presence of such delays it can be advantageous for the parameter estimation procedure to compromise on the alignment of simulated and measured peaks while matching their shape and height. Hence, an alternative position metric is introduced that initially reduces the penalty by $1/2$ in a range of less than $1/10\ t_r$ and then linearly ascends to one when out of alignment by t_r, eq. 26. Figure 5 illustrates the difference between the standard and initially reduced position penalty metrics. The initial reduction, $1/2$, and range, $1/10$, are chosen by experience and can be changed by the user.

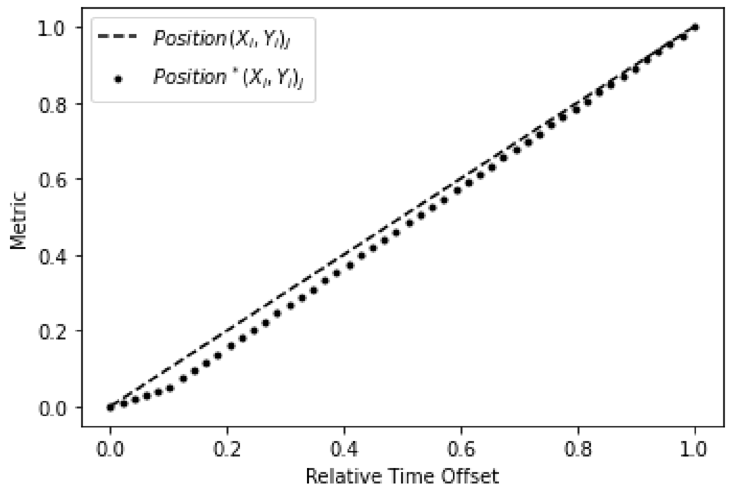

Figure 5: Comparison of standard position penalty and initially reduced position penalty metrics.

$$Position(X_i, Y_i)_J = \frac{t_s(X_i, Y_i)_J}{t_r} \tag{25}$$

$$Position^*(X_i, Y_i)_J = \begin{cases} \frac{1}{2}\frac{t_s(X_i, Y_i)_J}{t_r}, \frac{t_s(X_i, Y_i)_J}{t_r} \leq \frac{1}{10} \\ \frac{19}{18}\frac{t_s(X_i, Y_i)_J}{t_r} - \frac{1}{18}, \frac{t_s(X_i, Y_i)_J}{t_r} > \frac{1}{10} \end{cases} \tag{26}$$

2.2.3. Peak Height

The peak height metric relates the maximal concentration of the simulated chromatogram to that of the measured data, eq. 27. This metric ascends to one when the difference in either direction is larger than 100%.

$$Height(X_i, Y_i)_J = \left| 1 - \frac{\max\limits_{j \in J} \mathrm{Y}_{i,j}}{\max\limits_{j \in J} \mathrm{X}_{i,j}} \right| \tag{27}$$

2.3. Combined Scores

The previously introduced metrics serve as building blocks for creating scores that quantify the difference between simulated chromatograms and measurement data. Scores are defined for individual components and can target the full chromatogram, individual peaks, or parts thereof, such as only the front of a peak. Each score is a set of metrics that depend on the index of the component, i, and on the set of considered time points, J. Goals will be composed of one or several scores that can then be combined into one objective or passed to a multi-objective search algorithm. The following scores have been defined per component this may include using a virtual component created by summing any combination of components.

2.3.1. Sum of Square Difference

The SSD score is the set of differences between simulated and measured chromatogram data, eq. 28. For technical reasons, each difference is interpreted as separate metric. In section 8, a goal will be defined as sum of squares of these metrics.

$$S_{SSD} = \{X_{i,j} - Y_{i,j} | j \in J\} \tag{28}$$

2.3.2. Full Peak

The most straightforward of the new scores is the combination of peak shape, position, and height, S_{Gauss}. This score is usually applied to a time interval, specified by the index set J, that contains a single peak of nearly Gaussian shape. This time interval is not automatically detected but needs to be specified by the user. A more elaborated score, S_{Peak}, additionally accounts for the shape, minimum and maximum of the time derivative and is better suited for fitting non-Gaussian peaks. By combining the peak shape with the shape of its derivative, this score is highly sensitive to the curvature of the chromatogram. The time offsets in the peak and in the slope are technically not constrained to be equal. In practice they hardly differ, except for the very first iterations of search algorithms with poor starting points. The scores S^*_{Gauss} and S^*_{Peak} are analogously defined with $Position^*(X_i, Y_i)_J$ in place of $Position(X_i, Y_i)_J$. As will be demonstrated in the results section, the score S_{Peak} with standard position penalty is particularly useful for estimating transport parameters, while the score S^*_{Peak} with initially reduced position penalty is more suitable for estimating binding parameters.

$$S_{Gauss} = \{Shape(X_i, Y_i)_J; Position(X_i, Y_i)_J; Height(X_i, Y_i)_J\} \tag{29}$$

$$S_{Slope} = \left\{Shape\left(\dot{X}_i, \dot{Y}_i\right)_J; Height\left(-\dot{X}_i, -\dot{Y}_i\right)_J; Height\left(\dot{X}_i, \dot{Y}_i\right)_J\right\} \tag{30}$$

$$S_{Peak} = S_{Gauss} \cup S_{Slope} \tag{31}$$

2.3.3. Peak Front

In some cases, only the front of a peak can be used for parameter estimation while other parts of the peak are deteriorated by unspecific interactions of a tracer molecule with the column or tubing. Dextran is a prominent example for such non-ideal behavior that leads to strong tailing and a reduced peak height. On the other hand, Dextran is commonly applied as tracer that does not penetrate the particle pores. Errors in the execution of an experiment can also render the back of a peak unusable for parameter estimation. These situations are addressed by a score, S_{Front}, that considers shape and position but not height of the peak. This score is typically used with rather short time intervals and few data points.

$$S_{Front} = \left\{Shape(X_i, Y_i)_J; Position(X_i, Y_i)_J\right\} \tag{32}$$

The peak front score is designed to extract as much usable information as possible from the chromatogram. Unsupervised application of this score requires to automatically determine the usable time interval while robustly removing the non-ideal parts with high precision on the cut points. For dextran data, the back end of this interval is chosen at the first inflection point of the measured chromatogram, i.e., the upper cut point is at the first maximum of the time derivative. By experience, this is a good choice as non-ideal interactions mainly impact on the height and tailing of the peak. The lower cut point is chosen where the measured chromatogram starts to differ from the baseline by more than 0.1% of the concentration at the upper cut point. By experience, 0.1% is a robust choice for this threshold. The exact positions of these cut points are determined using Powell's method on the continuous spline approximation from section 5. The nearest time points of the discrete measurement data are then used as boundaries of the time interval specified by J.

2.3.4. Fractionation Data

Optical detectors that are typically applied for measuring chromatograms can usually not distinguish between different chemical components. Instead, they deliver a single sum signal where the contributions of the individual components are weighted by their extinction coefficients. Such signals alone cannot be used for parameter estimation unless the peaks of the relevant components are sufficiently separated. For instance, the acidic, main, and basic components of a monoclonal antibody often completely overlap in a single peak. This situation is normally addressed by fractionation, i.e., pooling the efflux of the column into a series of vials. Each of these vials is then analyzed offline to quantify the components of interest, which provides additional information for setting up a dedicated parameter estimation score.

The previously introduced metrics can generally be applied to the concentrations in each vial using the centers of the corresponding collection intervals as time points. For precise comparison, the corresponding per component simulations are averaged over the same collection intervals when a metric is applied to fractionation data. The resulting information is often sparse, with 5 to 10 fractions per peak, and can be afflicted with additional errors in the fractionation times and volumes. Small shifts in the collection intervals can cause major changes in the distribution of components between the analyzed fractions, particularly for sharp peaks. A spline is applied to the original simulated data before it is shifted and virtually fractionated to determine the time offset, t_s, to maintain sub-grid accuracy. Based on this offset, the scores S^*_{Gauss} and S^*_{Peak} as well as their immediately penalized versions can be computed. Analogously, the SSD score can be applied to fractionation data by averaging the simulations over the collection intervals.

2.4. Search algorithms

CADET-Match uses two alternative search strategies, gradient descent, and a multi-objective genetic algorithm. For gradient descent, all metrics need to be combined into a single scalar value, while the genetic algorithm can operate on multiple metrics.

2.4.1. Gradient Descent

Gradient descent algorithms search for a local optimum of the goal function using derivative information with respect to the sought parameters[55]. Gradient descent has long been used for parameter estimation in chromatography. It is very efficient near the sought optimum but can fail if the goal function is not smooth or the Jacobian becomes singular. Moreover, this algorithm is prone to becoming trapped in local optima, which can be remote from the starting point. This can be avoided by basin hopping or multi-start strategies. The latter is often applied when refining the results of population bases search strategies.

2.4.2. Genetic Algorithm

Genetic algorithms (GA) where first published by Holland in 1975 [56] and are an example of biomimicry. At their core they work like a colony of bacteria adapting to an outside environment and share many of the same features. GAs are embarrassingly parallel. An initial population is created, often using quasi-random methods such as Latin hypercube sampling [57] or Sobol sequences [58]. Each member of the population is then evaluated based on one or more objectives. At the end of each generation the fittest members survive and reproduce to form the next generation. The next generation is created by a combination of breeding and mutation on the survived members. There are variations in this procedure that maintain population diversity, choose members to be included in the next generation and change how breeding and mutation are implemented. These variations result in different algorithms such as NSGA2 [59], NSGA3 [60] and SPEA2 [61]. In view of the no free lunch theorem [62], different GA variants were tested and optimized on a variety of problems before settling on NSGA2 for single-objective problems and NSGA3 for multi-objective problems.

2.4.3. Progress Monitoring

Building complex models correctly, properly processing experimental data, and determining suitable starting points can be a difficult and tedious task. Based on experience, new models or concepts are unlikely to be correctly implemented on the first attempt. However, such issues can only be tested by attempting to fit model to data. Since errors can often be identified in early phases of the parameter estimation process, CADET-Match provides functionality to monitor the progress of specific indicators such as peak height, shape, mass, etc. This allows observing if the starting points yield reasonable results and if the search algorithm continuously improves the goal. Progress monitoring enables early aborting if progress is poor or if results are already good enough. This is essential for rapid testing of models, goals, starting points and stopping criteria. Since suitable starting points can be hard to determine, a GA with rather large population size is generally a good choice for initial testing. Multi-start gradient search is not a good alternative, as parallel iterative processes are more difficult to monitor.

2.5. Practical Application

The goal system and search strategy are first verified on synthetic examples of increasing complexity (section 2.6) and then validated on experimental data (section 2.7). The parameter estimation procedure is highly automated and the same for all case studies.

Depending on the current stage in the parameter estimation procedure (section 2) and on the quality of the data, the goals are based on one of the S_{Front}, S_{Peak}, or S^{*}_{Peak} scores and separately on the S_{SSD} score for comparison. While SSD is used for search, the results of different scores and search algorithms are compared using NRMSD. The S_{SSD} scores are usually taken over the whole chromatogram, $J = \{1, \ldots, N_d\}$. For single-objective optimization, the scores of multiple components are simply merged into one set. The corresponding goal, G_{SSD}, is created by adding the squared metrics in this set, eq. 33. Single-objective goals for the alternative scores are similarly created by merging the scores of different components and adding the squared metrics in the resulting set. CADET-Match offers other single objective goals, such as the average or the maximum of the metrics, but they are not used in this study.

$$G_{SSD} = \sum_{i=1}^{N_c} \sum_{j=1}^{N_d} \left(X_{i,j} - Y_{i,j}\right)^2 \tag{33}$$

Multi-objective optimization is more demanding on the search strategy but can benefit from much richer information on the impact of individual parameters or parameter groups on the path from the starting points to the sought optimum. For example, column porosity affects the peak position metric more strongly than the peak shape metric and column dispersion affects the peak shape metric more strongly than the peak position metric. Providing these metrics to a multi-objective search algorithm practically decouples the parameters, i.e., progress can be made in the objectives independent of each other. If these objectives are not in conflict, which is most often the case, the search algorithm still converges to a unique optimum. In more complex settings with numerous parameters and metrics, progress can be made in several objectives while other objectives are temporarily sacrificed even when they are not conflicting in the global optimum. The multi-objective approach improves the performance of genetic algorithms, as these are population-based by nature, while gradient search is more efficient on single objectives.

In addition, multi-objective search automatically detects conflicting metrics, in case they exist, and provides detailed information on the resulting Pareto front. This usually indicates a tradeoff, e.g., between matching shape and position as illustrated in Figure *4*. In such cases, the user can manually select the preferred optimum. To fully automate this process, the Pareto optimal parameter set with lowest mean of the involved metrics is selected. Eq. 34 defines the mean, $\bar{S}$, of a given set, S, of metrics, M, with $|S|$ elements. The metrics range from 0 to 1 with 0 indicating a perfect match and 1 a very poor match. This scale is reversed before and after taking the geometric mean, as otherwise one very low metric could potentially dominate the entire mean. For comparison, the mean, $\bar{S}$, is reported on the final result even when the optimization was based on the S_{SSD} score.

$$\bar{S} = 1 - \left(\prod_{M \in S} (1 - M)\right)^{\frac{1}{|S|}} \tag{34}$$

The search space is confined by box constraints that are specified in the case studies. A Sobol sequence is used to create a starting population with 100 N_p individuals, where N_p is the number of concurrently estimated parameters. When gradient descent is used with S_{SSD}, starting points with less than 5% peak overlap between simulation and experiment are removed from the population without replacement to guarantee sufficient sensitivity of the goal with respect to changes in the estimated parameters. This is not required for the new scores S_{Front}, S_{Peak} and S^{*}_{Peak} that include the peak position metric. This metric is naturally sensitive to parameter changes, independent of the peak overlap.

For gradient descent, the scores are always combined into a single objective. The trust-region reflective algorithm [63] from the scipy.optimize.least_squares function [49] is started at each point of the population. These searches are independent of each other and run in parallel to save time. They are stopped with a tolerance of $x_{tol} = 10^{-10}$ or if the current simulation fails. Looser tolerances were tested on synthetic examples and found to terminate too soon in some cases.

The non-dominated sorting genetic algorithm (NSGA) is used with a single objective (NSGA2) for S_{SSD} and multiple objectives (NSGA3) for S_{Front}, S_{Peak} and S^{*}_{Peak}. Distributed Evolutionary Algorithms in Python (DEAP) [64] is used for both algorithms. Crossover and mutation rates of 1.0 are applied to ensure that in every generation each member will mutate at least one parameter and swap parameters with at least one other member. The GA terminates after 30 generations without a new point being added to the Pareto front. Each entry on the final Pareto front is locally optimized in parallel using the trust-region reflective algorithm from above. This is a single-objective search, but the results are again analyzed with respect to Pareto optimality based on the individual metrics.

The reported wall clock times should be understood as approximate due to the non-deterministic nature of the GA, and small changes in stopping criteria can cause substantial differences in runtime. Moreover, the GA utilizes the available compute cores more efficiently than gradient search. All simulations where run on a dual socket Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz with a total of 32 cores and 64 threads with Ubuntu Linux 18.04.1, Python 3.7.7, Intel MKL 2019.3.199, CADET 4.0.1 and CADET-Match 0.6.13. Version information on other used software packages is shown in Table 5. CADET-Match can be installed from the Python Package Index (PyPI). The full code, including the scripts for the following case studies, is freely available on GitHub (https://github.com/modsim/CADET-Match).

Table 5: Software versions used for parameter estimation.

Software	*Version*	*Source*
CADET	4.0.1	https://cadet.github.io
CADET-Match	0.6.13	https://github.com/modsim/CADET-Match
CADET-Python	0.6	https://github.com/modsim/CADET-Python
Python	3.7.7	https://www.python.org
Joblib	0.15.1	https://joblib.readthedocs.io
addict	2.2.1	https://github.com/mewwts/addict
corner	2.0.1	https://corner.readthedocs.io
emcee	3.0.2	https://emcee.readthedocs.io
SALib	1.3.11	https://salib.readthedocs.io
DEAP	1.3.1	https://deap.readthedocs.io
psutil	5.7.0	https://psutil.readthedocs.io
NumPy	1.18.5	https://numpy.org
OpenPyXL	3.0.3	https://openpyxl.readthedocs.io
SciPy	1.5.0	https://www.scipy.org
Matplotlib	3.2.2	https://matplotlib.org
pandas	1.0.5	https://pandas.pydata.org
H5py	2.10.0	https://www.h5py.org
Seaborn	0.10.1	https://seaborn.pydata.org
scikit-learn	0.23.1	https://scikit-learn.org
Jstyleson	0.0.2	https://github.com/linjackson78/jstyleson

2.6. Synthetic Case Study

The synthetic case study was designed using the general rate model, based on experience with experimental data so that the parameters are within plausible ranges. They have not been tuned to make the output particularly easy or hard to match. All parameters are reported to two decimal places. The fixed parameters for simulating the synthetic ground truth data are shown in Table 2. In practice, these parameters are specified or separately determined by other means. The ground truth of the estimated parameters is reported in the following subsections. All synthetic data has a time spacing of 1 value per second. The column and tubing start off with a concentration of 0 except in the case of the SMA binding model where the column begins with bound salt equal to the ionic capacity (eq. 10) and a liquid phase salt concentration equal to the loading salt concentration. Independent and identically distributed Gaussian noise with mean 0.0 and standard deviation 0.1% of peak maximum is added to the simulated chromatogram of each component.

Table 6: Fixed parameters for simulating synthetic ground truth.

Parameter	*Value*	*Unit*
Q	2.88e-08	m^3/s
A_c	1.04e-04	m^2
L_c	2.50e-01	m
r_p	4.50e-05	m
Λ	2.25e+00	mol/m^3

The synthetic cases are designed to verify the functioning of the goal system and search strategies. They are presented in the same staged order that the parameters are normally estimated in (section 2). However, the results of one stage are not carried over to the next stage, but previously estimated parameters are fixed at their nominal values, to test all goal and search algorithm combinations with a defined ground truth. In the synthetic case study, multiple objectives of the new scores are generally not in conflict. The mean of the involved metrics, $\bar{S}$, is shown for comparison with the experimental case study, where it is used to select a Pareto optimal parameter set, but not further discussed for the synthetic case study. A match is considered successful if the sought parameters are correctly estimated to two digits.

The parameter transformations from section 2.1 are automatically applied. If the upper and lower bounds are less than 3 orders of magnitude apart, the linear transformation is used, eq. 13. Otherwise, the logarithmic transformation is used, eq. 14, except for the custom transformation for k_a and k_d, eq. 15-16. The reference concentrations of the SMA model are $c_r^s = 225\ mol/m^3$ and $c_r^p = 450\ mol/m^3$.

2.6.1. Non-Binding and Non-Pore Penetrating Tracer Pulse

The first parameter estimation stage is omitted here, as the synthetic data is generated without tubing and other holdup volumes that are external to the column. In the second stage, column porosity, ε_c, and axial dispersion, D_{ax}, are estimated from a pulse experiment with a non-binding and non-pore penetrating tracer molecule. In theory, these model parameters do hardly depend on the chosen tracer molecule. In real experiments, dextran is typically used for this purpose. This can be problematic due to non-ideal behavior of this tracer that often leads to strong tailing and a reduced peak height, as has been introduced in section 2.3. Only the peak front is used because the rest of the peak is often deteriorated by unspecific interactions of dextran with the column. Since this non-ideal behavior is not covered by the applied chromatography model, it cannot be reproduced in the peak tailing. However, an idealized setting is used for verifying the respective score, S_{Front}. The time interval J of this score is determined as described in section 2.3. It ranges from 269 to 359 s (Figure 6). For comparison, the S_{SSD} score is applied to the same slice of the chromatogram. Table 7 shows the parameter bounds that were specified in the search algorithms. The parameter estimation results are shown in Table 7 and Figure 7. Any combination of tested search algorithm and score were able to recover the estimated parameters to prescribed accuracy in less than five minutes. The NRMSD is about 0.01% of the peak maximum for GA and S_{Front} and lower in the other cases. For the human eye, the fitted chromatograms in Figure 7 are practically indistinguishable from the synthetic ground truth, even 100x magnified (Figure 6).

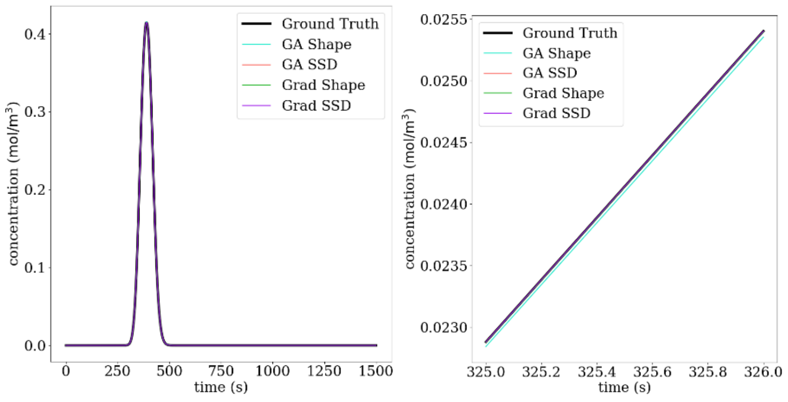

Figure 6: Non-binding and non-pore penetrating tracer pulse experiment. Synthetic ground truth and estimated chromatograms. Full chromatogram (left) and 100x magnified detail (right).

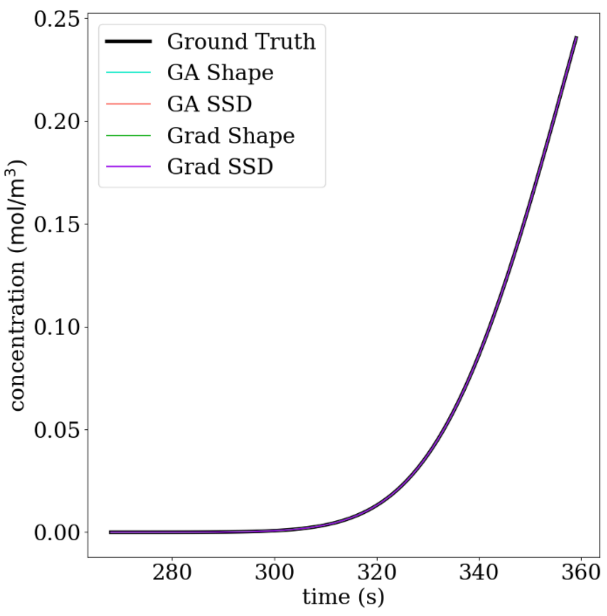

Figure 7: Non-binding and non-pore penetrating tracer pulse experiment. Synthetic ground truth and estimated chromatograms.

Table 7: Non-binding and non-pore penetrating tracer pulse experiment. Synthetic ground truth, parameter bounds, estimated parameters, and performance indicators.

	Ground Truth	*Bounds*		*GA*		*Gradient*	
		Lower	Upper	S_{Front}	S_{SSD}	S_{Front}	S_{SSD}
ε_c	0.40	0.20	0.50	0.40	0.40	0.40	0.40
D_c	3.0e-07	1.0e-12	1.0e-5	3.0e-07	3.0e-07	3.0e-07	3.0e-07
NRMSD				9.2e-05	1.4e-07	7.6e-06	8.5e-08
$\bar{S}$				3.5e-07	3.3e-06	3.3e-06	3.3e-06
Wall Time				0:03:10	0:03:37	0:04:44	0:02:24

2.6.2. Non-Binding but Pore-Penetrating Tracer Pulse

In the third stage, particle porosity, ε_p, film diffusion, k_f, and pore diffusion, D_p, are estimated from a non-binding but pore penetrating tracer pulse. These parameters depend on the size and shape of the chosen tracer molecule more strongly than porosity and axial dispersion in the column. Hence, it is advisable to use the target protein under non-binding conditions, such as high salt in ion-exchange chromatography. This setting is used for verifying the score S_{Peak} in comparison with S_{SSE}. The time interval J covers the whole peak (Figure 8). Results are shown in Table 8 and Figure 9. Also in this case, any combination of tested search algorithm and score were able to recover the sought parameters to prescribed accuracy, now in under eight minutes. The NRMSD is 0.05% of the peak maximum for gradient search and S_{Peak} and lower in the other cases. For the human eye, the fitted chromatograms in Figure 9 are practically indistinguishable from the synthetic ground truth. When 100x magnified, a slight difference can be observed for gradient search with S_{Peak} (Figure 8).

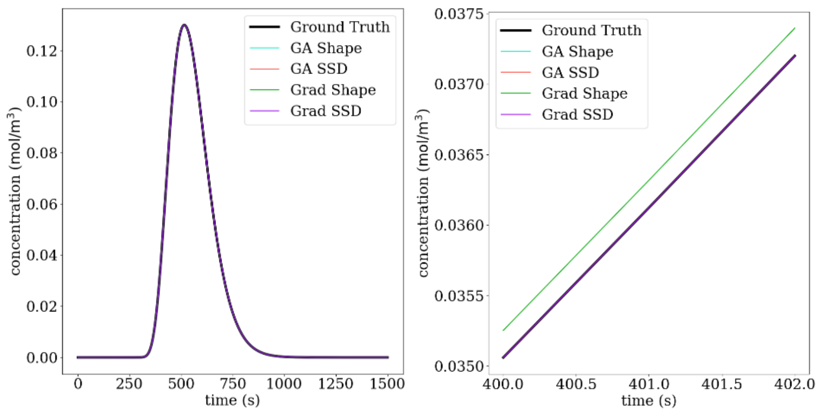

Figure 8: Non-binding but pore penetrating tracer pulse experiment. Synthetic ground truth and estimated chromatograms. Full chromatogram (left) and 100x magnified detail (right).

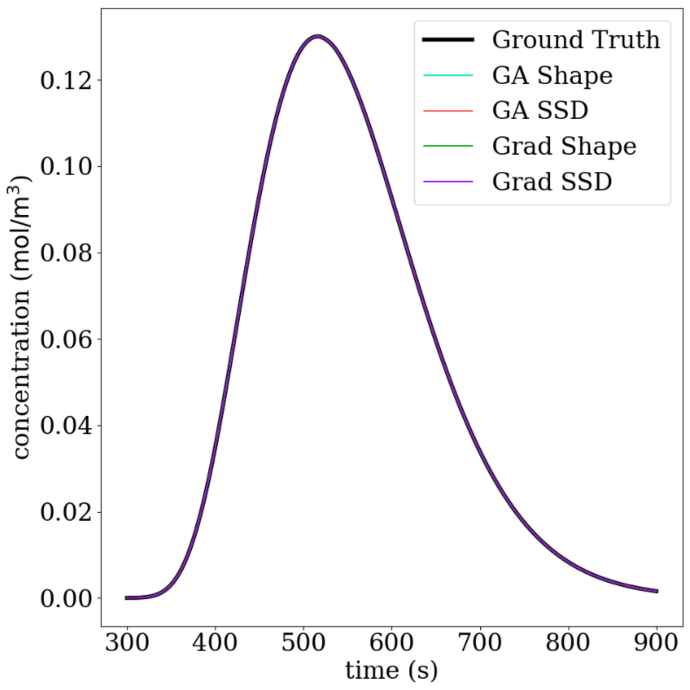

Figure 9: Non-binding but pore penetrating tracer pulse experiment. Synthetic ground truth and estimated chromatograms.

Table 8: Non-binding but pore penetrating tracer pulse experiment. Synthetic ground truth, parameter bounds, estimated parameters, and performance indicators.

	Ground Truth	*Bounds*		*GA*		*Gradient*	
		Lower	Upper	S_{Peak}	S_{SSD}	S_{Peak}	S_{SSD}
ε_p	0.30	0.2	0.5	0.30	0.30	0.30	0.30
k_f	2.0e-07	1.0e-09	1.0e-05	2.0e-07	2.0e-07	2.0e-07	2.0e-07
D_p	5.0e-11	1.0e-14	1.0e-06	5.0e-11	5.0e-11	6.0e-11	5.0e-11
NRMSD				7.5e-07	7.5e-07	4.7e-04	7.5e-07
$\bar{S}$				2.2e-08	1.8e-08	2.3e-05	1.6e-08
Wall Time				0:04:08	0:04:16	0:05:06	0:07:48

2.6.3. Single Component Gradient Elution

In the fourth stage, the parameters of the binding model are estimated from gradient elution data. The parameters of the non-equilibrium SMA binding model, i.e., scaled adsorption and desorption rates, $\tilde{k}_a$ and $\tilde{k}_d$, characteristic charge, ν, and shielding coefficient, σ, are estimated from gradient elution data. Only the ionic capacity, Λ, is set to the ground truth, as this parameter is usually determined separately by titration. Estimating the binding parameters is particularly difficult due to strong non-linearity of the SMA model. Hence, synthetic data of three linear gradient elution experiments with different gradient slopes are used, and the loading phase of the first experiment is extended to a full breakthrough, as shown in Figure 10. The minor peaks in the other two experiments indicate complete loading.

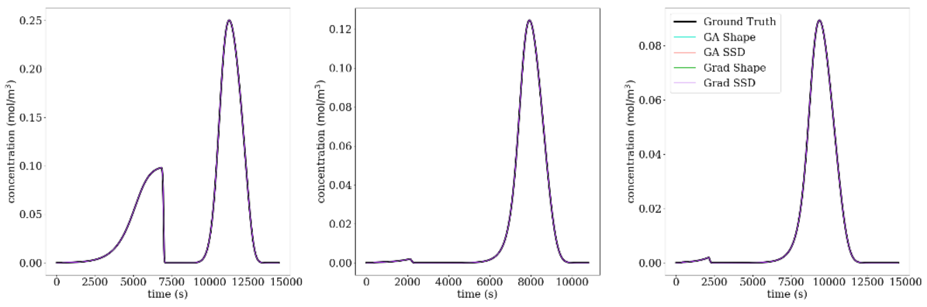

Figure 10: Single component gradient elution experiments with different loading times and gradient slopes. Synthetic ground truth and estimated chromatograms.

The column is loaded at inlet salt and protein concentrations of $c_0^{in} = 180\ mol/m^3$ and $c_1^{in} = 0.10\ mol/m^3$ for $6500\ s$ (Figure 11) in the first experiment and for $1800\ s$ in the second (Figure 12) and third experiment (Figure 13). It is washed at an inlet salt concentration of $c_0^{in} = 70\ mol/m^3$ for $2000\ s$ in all three experiments. The elution gradients start at an inlet salt concentration of $c_0^{in} = 70\ mol/m^3$ and have slopes of $0.08\ mol/(m^3 \cdot s)$, $0.06\ mol/(m^3 \cdot s)$ and $0.04\ mol/(m^3 \cdot s)$. They are applied for $6000\ s$, $7000\ s$ and $11000\ s$. The time interval of the S_{Peak}^* score on the full breakthrough in the first experiment was chosen from $1000\ s$ to $7300\ s$. The minor peaks in the second and third experiments indicate complete loading but are not utilized for parameter estimation. The time intervals of the S_{Peak}^* score on the elution peaks were chosen from $8500\ s$ to $14000\ s$, from $5000\ s$ to $10000\ s$ and from $6000\ s$ to $12000\ s$.

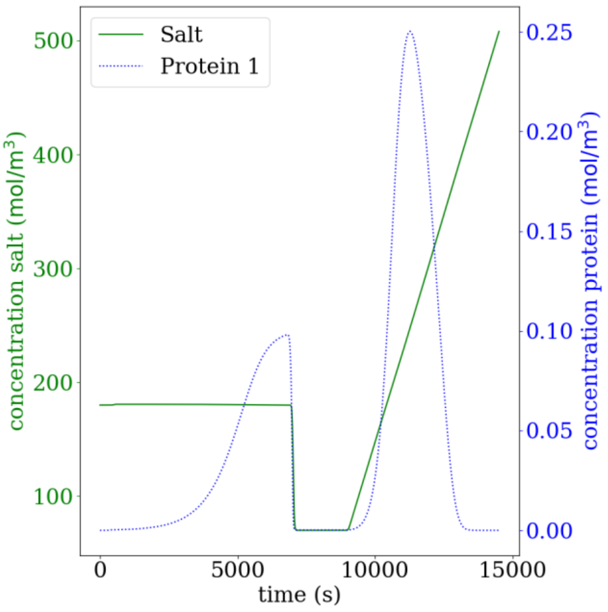

Figure 11: Synthetic Single Component Gradient Elution Experiment 1: Salt profile overlaid with ground truth.

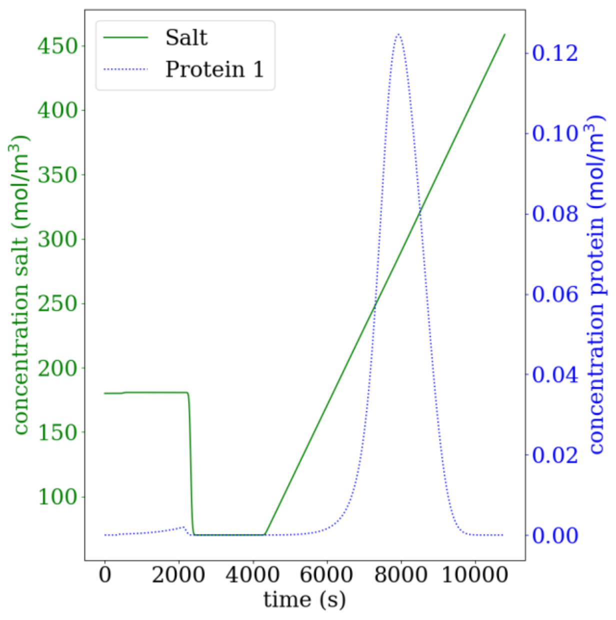

Figure 12: Synthetic Single Component Gradient Elution Experiment 2: Salt profile overlaid with ground truth.

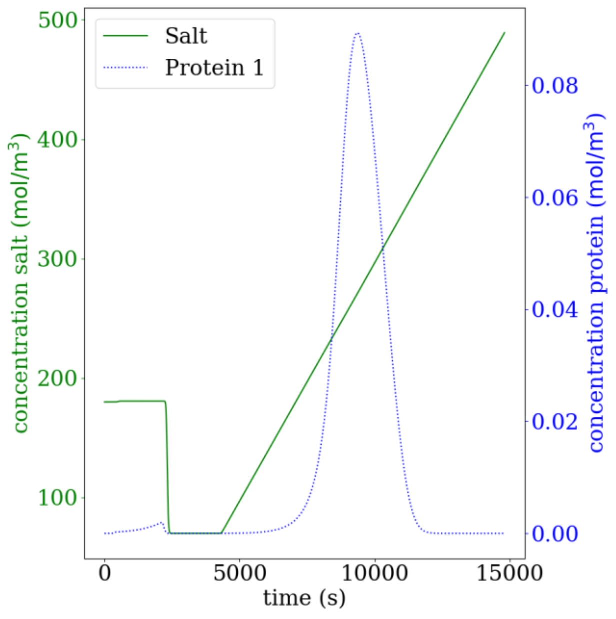

Figure 13: Synthetic Single Component Gradient Elution Experiment 3: Salt profile overlaid with ground truth.

The sought parameters are estimated by fitting separate model instances with respective boundary conditions simultaneously to all synthetic experiments. The S^*_{Peak} score is applied to the four major peaks and is composed of six metrics, resulting in $4 \cdot 6 = 24$ objectives for the genetic algorithm. A single-objective goal for gradient search is created by adding the squared metrics. The S_{SSE} score is applied to the same slices of the chromatograms. The parameter transformation in eq. 15-16 is applied with $\tilde{k}_{eq} = \tilde{k}_a/\tilde{k}_d$. That is, the search algorithms operate on $\tilde{k}'_a$ and $\tilde{k}'_{eq}$ instead of $\tilde{k}_a$ and $\tilde{k}_d$. Results are shown in Table 9 and Figure 10. The values of $\tilde{k}_d$ are reported for completeness. Search bounds are not needed for this parameter. Any combination of search algorithm and score were able to recover the sought parameters in less than 10 hours. For the S^*_{Peak} score, the GA was about two times faster than gradient search and about three times faster for the S_{SSD} score. The NRMSD is less than 0.01% of the peak maximum for gradient search and S^*_{Peak} and lower in the other cases.

Table 9: Single component gradient elution experiment. Synthetic ground truth, parameter bounds, estimated parameters, and performance indicators.

	Ground Truth	*Bounds*		*GA*		*Gradient*	
		Lower	Upper	S^*_{Peak}	S_{SSD}	S^*_{Peak}	S_{SSD}
$\tilde{k}_a$	0.30	1.0e-02	1.0e+02	0.30	0.30	0.30	0.30
$\tilde{k}_d$	1.50	-	-	1.50	1.50	1.50	1.50
$\tilde{k}_{eq}$	0.20	1.0e-02	1.0e+02	0.20	0.20	0.20	0.20
ν	7.00	1.00	50.0	7.00	7.00	7.00	7.00
σ	50.0	1.00	100	50.0	50.0	50.0	50.0
NRMSD				1.8e-06	1.8e-06	6.5e-05	1.8e-06
$\bar{S}$				1.4e-04	1.4e-04	1.7e-05	1.4e-04
Wall Time				5:58:09	3:29:55	9:29:12	9:00:45

2.6.4. Multi-Component Gradient Elution

This case study is analogous to the previous one but with two components. The synthetic ground truth mimics charge variants of a protein without baseline separation. The transport parameters of the first three stages are the same for both components. SMA binding parameters of both components are simultaneously estimated from three gradient elution experiments, including one full breakthrough. Full chromatogram data is assumed to be known for each component. This is not normally given in practice but an important intermediate step in verifying the goal system and search strategies. The assumption will be dropped in the next case study. In this case, the breakthrough peaks at the end of all three loading phases are large enough to provide useful information. Hence, the S^*_{Peak} score is applied to two components and six peaks, resulting in $2 \cdot 6 \cdot 6 = 72$ objectives.

The inlet concentrations and durations of the load, wash and elution phases are the same as in the single component case (Figure 14,Figure 15,Figure 16), with the same inlet concentrations for both proteins. The time interval of the S^*_{Peak} score on the full and partial breakthroughs were chosen from $0\ s$ to $7300\ s$, from $0\ s$ to $2500\ s$ and from $0\ s$ to $2500\ s$. The time intervals of the S^*_{Peak} score on the elution peaks were chosen from $9000\ s$ to $14000\ s$, from $5000\ s$ to $10000\ s$ and from $5000\ s$ to $12000\ s$ for the first protein and from $9000\ s$ to $13000\ s$, from $4000\ s$ to $9000\ s$ and from $4000\ s$ to $11000\ s$ for the second protein.

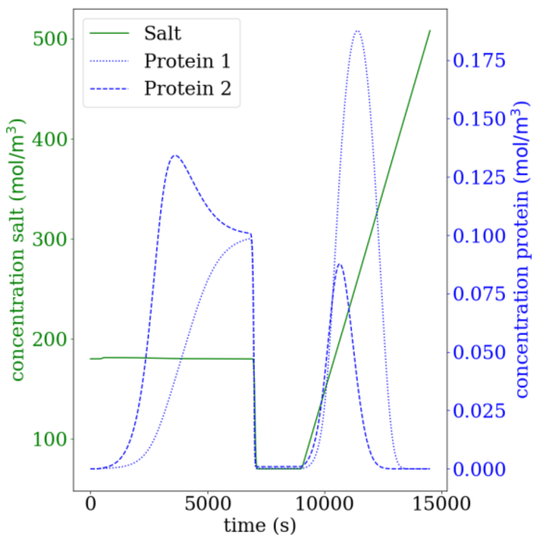

Figure 14: Synthetic Multi Component Gradient Elution Experiment 1: Salt profile overlaid with ground truth.

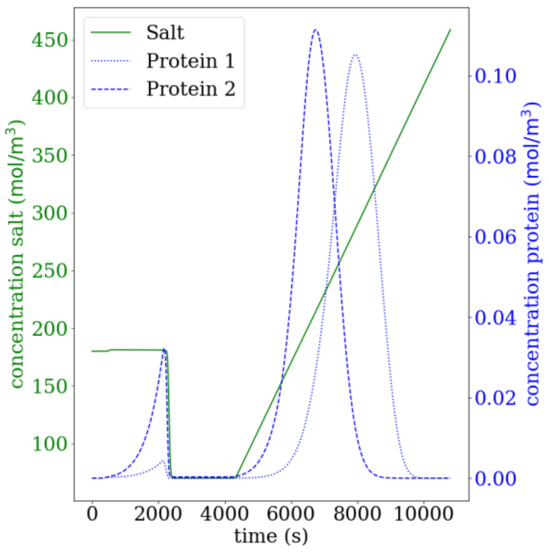

Figure 15: Synthetic Multi Component Gradient Elution Experiment 2: Salt profile overlaid with ground truth.

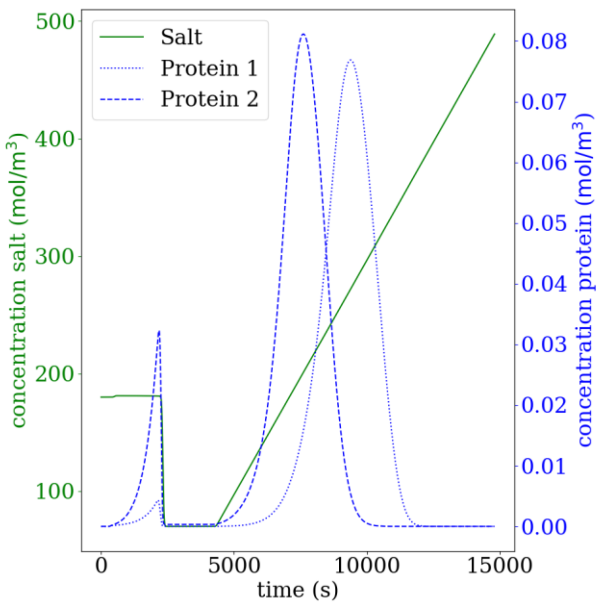

Figure 16: Synthetic Multi Component Gradient Elution Experiment 3: Salt profile overlaid with ground truth.

Results are shown in Table *10* and Figure 17. Any combination of search algorithm and score were able to recover the estimated parameters, except for the scaled adsorption rates, $\tilde{k}_a$, and shielding coefficients, σ, when estimated using S^*_{Peak} and gradient search. The shielding coefficients deviate from the ground truth by less than 0.5%. Even though the scaled adsorption rates are off by 5% and 20% for the first and second component, the corresponding NRMSD is below 0.2% of the peak maximum, which is still hard to observe by the human eye. This exemplifies that parameters with little impact on the model prediction are harder to estimate, which can be tolerated if accurate model predictions are sufficient for the application. However, it becomes problematic when accurate parameter values are required, e.g., for tabulation and application in scenarios with higher impact on model predictions.

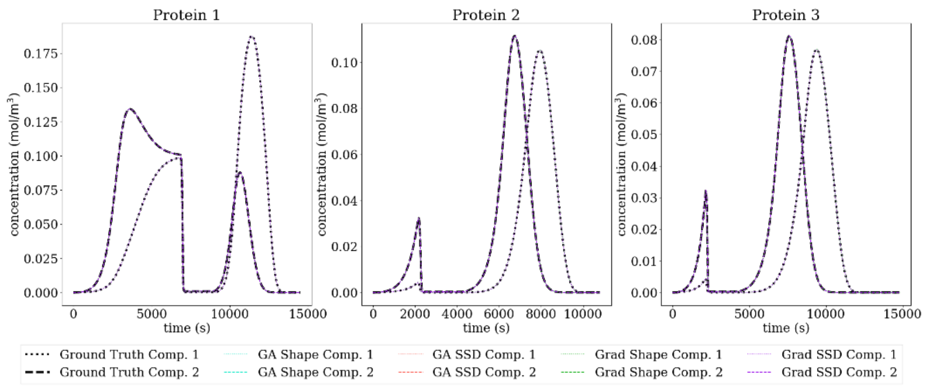

Figure 17: Two component gradient elution experiment with different loading times and gradient slopes. Synthetic ground truth and estimated chromatograms.

Table 10: Two component gradient elution experiment. Synthetic ground truth, parameter bounds, estimated parameters, and performance indicators.

	Ground	*Bounds*	*GA*	*Gradient*

	Truth	Lower	Upper	S^*_{Peak}	S_{SSD}	S^*_{Peak}	S_{SSD}
$\tilde{k}_{a,1}$	2.00	1.0e-02	1.0e+02	2.00	2.00	1.90	2.00
$\tilde{k}_{d,1}$	10.0	-	-	10.0	10.0	9.70	10.0
$\tilde{k}_{eq,1}$	0.20	1.0e-02	1.0e+02	0.20	0.20	0.20	0.20
ν_1	7.00	1.00	50.0	7.00	7.00	7.00	7.00
σ_1	50.0	1.00	100	50.0	50.0	50.1	50.0
$\tilde{k}_{a,2}$	2.00	1.0e-02	1.0e+02	2.00	2.00	2.40	2.00
$\tilde{k}_{d,2}$	10.0	-	-	10.0	10.0	12.0	10.0
$\tilde{k}_{eq,2}$	0.20	1.0e-02	1.0e+02	0.20	0.20	0.20	0.20
ν_2	5.00	1.00	50.0	5.00	5.00	5.00	5.00
σ_2	50.0	1.00	100	50.0	50.0	49.8	50.0
NRMSD				3.3e-05	3.3e-05	1.8e-03	3.3e-05
$\bar{S}$				9.9e-06	9.9e-06	6.1e-04	9.9e-06
Wall Time				2 days: 21:23:58	1 day: 20:08:52	5 days: 12:20:55	3 days: 23:09:31

The scaled desorption constants, $\tilde{k}_d$, which are not directly estimated but deduced from the respective recovered equilibrium constants, $\tilde{k}_{eq}$, are off by 3% and 20%. The increased compute time of more than 5 days also indicates a flat optimum which is more challenging for gradient search than for the GA. We speculate that the combination of S^*_{Peak} and gradient search is particularly sensitive to the smoothing procedure, and this issue is subject of ongoing research. The other combinations improved the NRMSD by about 2 orders of magnitude, and the same is true for the mean, $\bar{S}$, of the 72 metrics in the optimum. The other combinations ran between 2 and 4 days with GA on S_{SSD} being the fastest.

2.6.5. Multi-Component Gradient Elution with Fractionation

The last synthetic case is close to real experimental data, as the assumption that the full chromatogram of each component can be separately observed is dropped. Instead, only the sum signal and fractionation data are available. The basic setup is parallel to the previous section but with additional fractionation data. Fractionation is equivalent to an extremely coarse discretization of the signal and can yield less than 10 data points compared to thousands in a normal chromatogram. In practice, the column efflux is partitioned into a series of samples that are analyzed offline to determine the concentration of each component. In industrial settings, parameter estimation is further complicated by additional sources of error from the fractionation process. The impact of such errors is currently studied but not in the scope of this publication.

Here, each fraction is approximately 1000 seconds in length. Precise collection times and pool concentrations are shown in Tables 10-12. The parameters of the SMA binding model are estimated using the S^*_{Peak} score on the sum chromatogram and the S^*_{Gauss} score on the fractionation data. S^*_{Peak} with six metrics on six peaks and S^*_{Gauss} with three metrics to three data sets and two components results in $1 \cdot 6 \cdot 6 + 2 \cdot 3 \cdot 3 = 54$ objectives. The S_{SSE} score is applied to the sum chromatogram and to the fractionation data of each component.

Table 11: Fractionation times and resulting pool concentrations of first synthetic gradient elution experiment.

Collect times (s)		*Pool concentrations (mol/m^3)*	
Start	Stop	Protein 1	Protein 2

2000	3000	9.66e-03	6.24e-02
3000	4000	3.49e-02	1.29e-01
4000	5000	6.68e-02	1.21e-01
5000	6000	8.91e-02	1.07e-01
6000	7000	9.39e-02	9.84e-02
7000	9000	2.60e-04	1.14e-03
9000	9800	3.38e-03	8.15e-03
9800	10600	5.66e-02	5.85e-02
10600	11400	1.65e-01	6.84e-02
11400	12200	1.64e-01	1.49e-02
12200	13000	5.26e-02	5.27e-04

Table 12: Fractionation times and resulting pool concentrations of second synthetic gradient elution experiment.

Collect times (s)		*Pool concentrations (mol/m^3)*	
Start	Stop	Protein 1	Protein 2
2000	3000	9.40e-04	8.08e-03
3000	4000	6.40e-06	3.23e-04
4000	5000	5.13e-05	1.21e-03
5000	6000	1.51e-03	1.77e-02
6000	7000	1.86e-02	9.05e-02
7000	9000	7.58e-02	2.61e-02
9000	9800	7.08e-03	6.19e-06

Table 13: Fractionation times and resulting pool concentrations of third synthetic gradient elution experiment.

Collect times (s)		*Pool concentrations (mol/m^3)*	
Start	Stop	Protein 1	Protein 2
2000	3000	9.40e-04	8.08e-03
3000	4000	6.40e-06	3.23e-04
4000	5000	2.49e-05	7.70e-04
5000	6000	3.48e-04	5.86e-03
6000	7000	2.74e-03	3.14e-02
7000	9000	3.10e-02	5.92e-02
9000	9800	7.43e-02	6.33e-03
9800	10600	5.10e-02	3.34e-04
10600	11400	1.43e-02	5.60e-06
11400	12200	7.81e-04	1.53e-08

Results are shown in Table 14 and Figure 18. Similar to the previous case, any combination of search algorithm and score were able to recover the estimated parameters, except the combination of S^*_{Peak} and gradient search. The shielding coefficients again deviate from the ground truth by less than 0.5%, and the scaled adsorption rates by 10% and 20% for the first and second component. Potential reasons for this have been explained in the previous section. The NRMSD is below 0.1% of the peak maximum. The NRMSD is more than two orders of magnitude smaller for the other algorithm and sore combinations. Differences are hardly visible to the human eye. This is remarkable since this case is based on much sparser experimental data than the previous one. However, systematic errors that typically occur in real experiments are not considered here. They will be present in the next case. The runtimes mainly differ between the algorithms and are similar between the scores, with the GA needing less than 2.5 days and gradient search about 5 days.

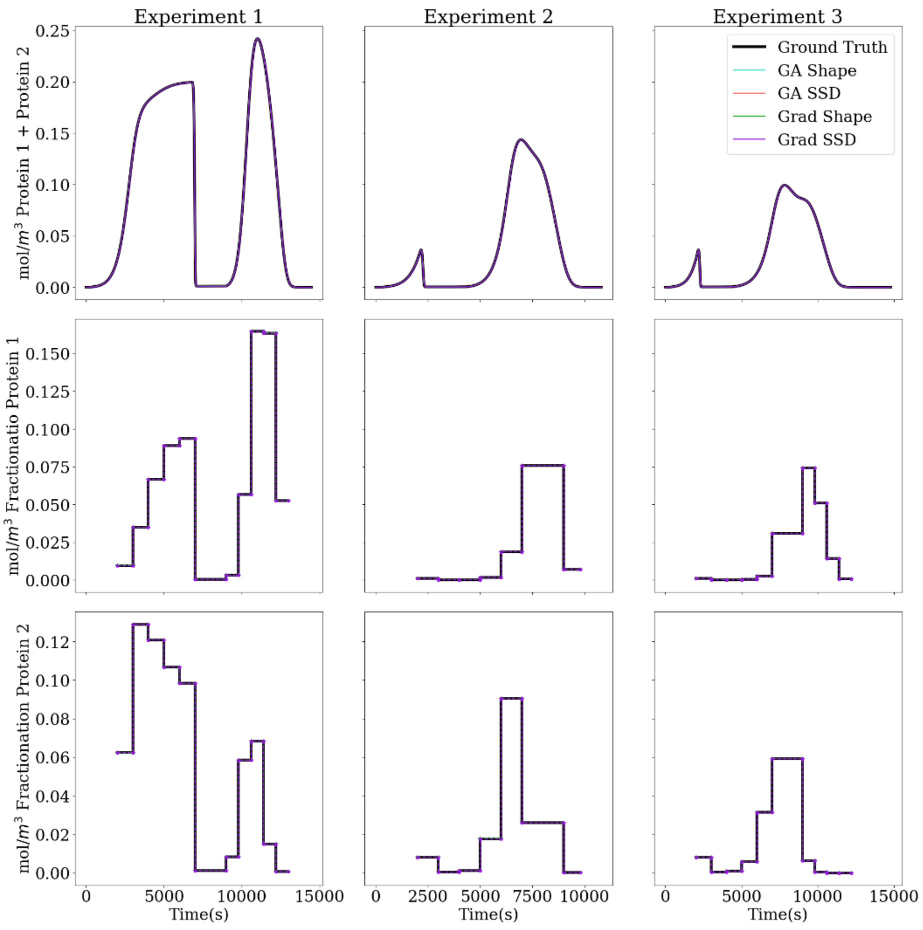

Figure 18: Two component gradient elution experiment with different loading times and gradient slopes. Synthetic ground truth and estimated chromatograms: sum signal (top), fractionation data of component 1 (mid) and 2 (bottom).

Table 14: Two component gradient elution experiment with fractionation. Synthetic ground truth, parameter bounds, estimated parameters and performance indicators.

	Ground Truth	*Bounds*		*GA*		*Gradient*	
		Lower	Upper	S^*_{Peak}	S_{SSD}	S^*_{Peak}	S_{SSD}
$\tilde{k}_{a,1}$	2.00	1.0e-02	1.0e+02	2.00	2.00	1.80	2.00
$\tilde{k}_{d,1}$	10.0	-	-	10.0	10.0	8.80	10.0
$\tilde{k}_{eq,1}$	0.20	1.0e-02	1.0e+02	0.20	0.20	0.20	0.20
ν_1	7.00	1.00	50.0	7.00	7.00	7.00	7.00
σ_1	50.0	1.00	100	50.0	50.0	50.0	50.0

$\tilde{k}_{a,2}$	2.00	1.0e-02	1.0e+02	2.00	2.00	2.40	2.00
$\tilde{k}_{d,2}$	10.0	-	-	10.0	10.0	12.0	10.0
$\tilde{k}_{eq,2}$	0.20	1.0e-02	1.0e+02	0.20	0.20	0.20	0.20
ν_2	5.00	1.00	50.0	5.00	5.00	5.00	5.00
σ_2	50.0	1.00	100	50.0	50.0	49.8	50.0
NRMSD				4.2e-06	4.2e-06	8.7e-04	4.2e-06
$\bar{S}$				4.5e-05	4.5e-05	4.6e-04	4.5e-05
Wall Time				2 days: 9:38:44	2 days: 3:40:50	4 days: 15:23:18	5 days: 2:19:39

2.7. Experimental Case Study

An experimental dataset for two charge variants of a monoclonal antibody has been measured at Amgen. Available are two dextran pulses with detached column, two dextran pulses with attached column, two protein pulses under non-binding conditions, a gradient elution with fractionation and a gradient elution with extended loading phase but without fractionation. Evaluating industrial data is more complicated than synthetic, since there is no ground truth available. Moreover, the detector noise is typically dominated by systematic errors such as feed variations, pump delays and flow rate variations. These issues are addressed by our new score system. The experimental case study comprises the four parameter estimation stages described in section 2. Parameters estimated in one stage are fixed in the next, with the results separately passed on for each search algorithm and score combination using the selected result shown in each stage's table. Orthogonality in the applied models and procedures avoids lumping of fundamental mechanisms and minimizes parameter correlations. This greatly improves predictivity of the calibrated model across operating conditions and scales. Modeling and propagation of errors is covered in section 3.

In the experimental case study, multiple objectives of the new scores happen to be in conflict due to imperfections in the model and data. For a new optimum to be added to the Pareto front, at least one parameter and at least one metric need to differ from the existing optima by at least 1%. The mean of the involved metrics, $\bar{S}$, is used to select the final result. The model parameters in Table 15 are determined independently of the staged parameter estimation procedure. Parameter transformations are applied analogously to the synthetic case study. Reference concentrations of the SMA model are $c_r^s = 225\ mol/m^3$ and $c_r^p = 450\ mol/m^3$.

Table 15: Fixed parameters for experimental case study.

Parameter	*Value*	*Unit*
Q	8.33e-08	m^3/s
A_c	2.01e-04	m^2
L_c	2.50e-01	m
r_p	4.50e-05	m
L_t	1.46e-01	m
Λ	2.23e+00	mol/m^3

2.7.1. Dextran Pulse with Detached Column

A DPFR is used to describe the impact of the tubing and other holdup volumes that are external to the chromatography column. For this dataset, more complex models with multiple CSTR and DPFR units were unable to better reproduce the observed behavior but suffered from high parameter correlations and poor identifiability (data not shown). Hence, the external holdup volumes are lumped, and the DPFR model parameters are not meant to reflect the real dimensions of the tubing. It was further observed that the given tubing length, L_t, can be used for the DPFR without deteriorating the match between model and data. In the first parameter estimation stage, the cross-section area, A_t, and dispersion, D_t, of the DPFR are estimated from a dextran pulse experiment with detached column, i.e., with the column replaced by a zero-volume connector. The S_{Front} score accounts for the non-ideal behavior of dextran tracer. The model is simultaneously fitted to two experimental replicates by combining the scores as described in section 2.5. The time intervals J are separately determined for both chromatograms. They range from 19 to 32 s for the first experiment and from 15 to 32 s for the second (Figure 19). As in the synthetic case study, the S_{SSD} score is applied to the same time intervals for comparison.

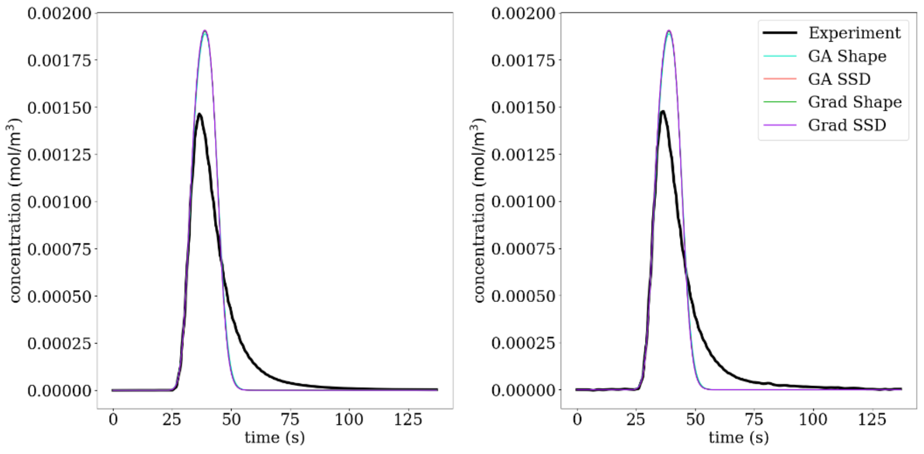

Figure 19: Two replicates of dextran pulse experiment with detached column. Full experimental data and estimated chromatograms.

Results are shown in Table 16 and Figure 20. Obviously, the quality of this real-world data is much worse than in the synthetic case study. This is reflected in larger NRMSD values of about 4% of the highest concentration in the selected time interval. Runtimes vary between one and three minutes. The algorithm and score combinations reach the same NRMSD and terminate at the same estimated areas, A_t, with two digits precision. The dispersion estimates deviate 8% or less from the mean of the compared values, with the largest deviation for S_{Front} with the GA. It is important to understand that these differences do not allow assessing the certainty of the estimates. Measurement errors clearly exist and are propagated to the estimated parameters. However, different methods are required for quantifying their nature and impact.

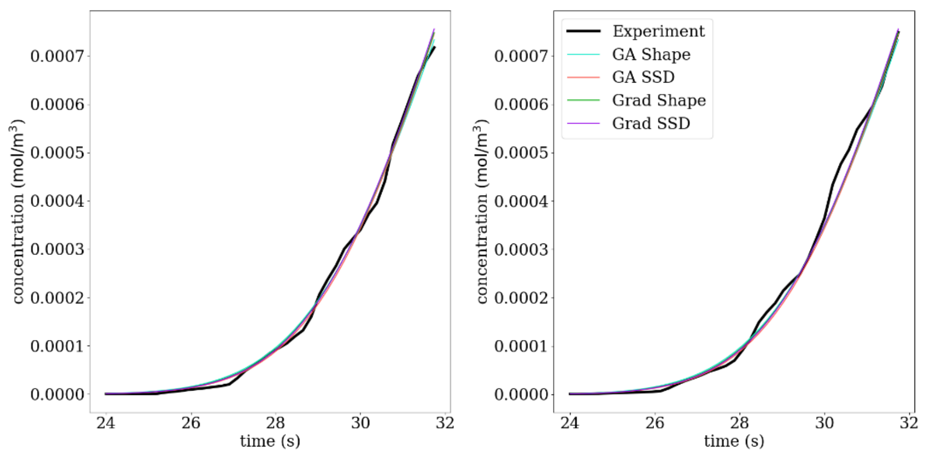

Figure 20: Two replicates of dextran pulse experiment with detached column. Experimental data and estimated chromatograms.

Table 16: Dextran pulse experiment with detached column. Parameter bounds, estimated parameters, and performance indicators.

	Bounds		*GA*		*Gradient*	
	Lower	Upper	S_{Front}	S_{SSD}	S_{Front}	S_{SSD}
A_t	1.5e-05	2.5e-05	1.9e-05	1.9e-05	1.9e-05	1.9e-05
D_t	1.0e-09	1.0e-05	2.7e-06	2.4e-06	2.5e-06	2.4e-06
NRMSD			3.8e-02	3.8e-02	3.8e-02	3.8e-02
$\bar{S}$			4.6e-04	4.2e-04	4.1e-04	4.2e-04
Wall Time			0:02:43	0:01:07	0:01:10	0:00:55
Results			2		1	

Interestingly, the GA has found conflicts between the metrics in the S_{Front} score, even though they are designed to be complementary to each other. Two different parameter sets are optimal in the Pareto sense, i.e., one metric can only be improved at the cost of deteriorating another. In this example, multiple optima are likely caused by relatively large errors in the experimental data. However, Pareto optima should always be carefully analyzed, as they indicate inconsistencies in the data that would remain undetected with the S_{SSD} score. In Table 16, the selection of a final result is based on the mean, $\bar{S}$, of the involved metrics. The parameters, metrics and chromatograms of both Pareto optima are shown in Table 17. The parameter estimates are identical within two digits precision, and differences in the corresponding chromatograms are hardly visible to the human eye without magnification. The mean of the metrics is about 3x smaller for the selected optimum, while the NRMSD about 8 times larger. However, the NRMSD can be misleading as exemplified in section 2.2.1. The selected Pareto optimum of GA and S_{Front} is used in the next stages.

Table 17: Pareto optima obtained by GA with S_{Front} for experimental dextran pulses with detached column.

Parameter	*Optimum 1*	*Optimum 2*
A_t	1.9e-05	1.9e-05
D_t	2.7e-06	2.7e-06
NRMSD	3.8e-02	4.8e-03

$\bar{S}$	4.6e-04	1.5e-03

2.7.2. Dextran Pulse with Attached Column

In the second stage, column porosity, ε_c, and column dispersion, D_{ax}, are estimated from a dextran pulse experiment with attached column. The model is fitted to two experimental replicates following the same procedure as above. The time intervals range from 52 to 218 s for the first experiment and from 24 to 218 s for the second (Figure 22). Results are shown in Table 18 and Figure 21. The NRMSD is similar and well below 1% for all algorithm and score combinations. Runtimes vary between about two and nine minutes with an advantage for the GA. The NRMSD values are slightly better for gradient search with both scores. The porosity estimates are almost identical, while the Dispersion estimates deviate 10% or less from the mean of the compared values, with larger deviations for the GA but in opposite directions for S_{Front} and S_{SSD}. Figure 21 shows a systematic mismatch between the slopes of model and data towards the right end of the considered time intervals, i.e., the mechanistic model does not fully capture the observed process. Uncertainty analysis and model extensions are covered in 2.8 and this particular example is looked at in 3.3.2.2. The selected optimum of gradient search and S_{Front} is used in the next stages.

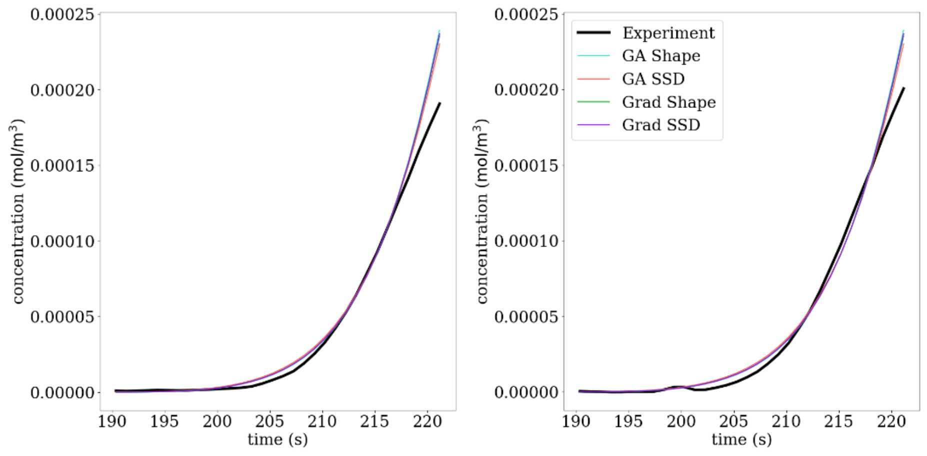

Figure 21: Two replicates of dextran pulse experiment with attached column. Experimental data and estimated chromatograms.

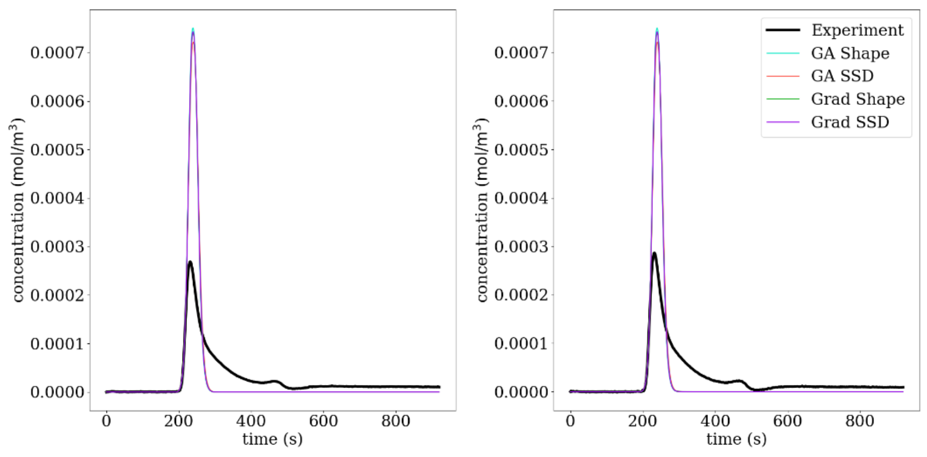

Figure 22: Two replicates of dextran pulse experiment with attached column. Full experimental data and estimated chromatograms.

Table 18: Dextran pulse experiment with attached column. Parameter bounds, estimated parameters, and performance indicators.

	Bounds		*GA*		*Gradient*	
	Lower	Upper	S_{Front}	S_{SSD}	S_{Front}	S_{SSD}
ε_c	2.0e-01	4.0e-01	3.3e-01	3.4e-01	3.3e-01	3.3e-01
D_c	1.0e-12	1.0e-05	3.2e-07	3.8e-07	3.4e-07	3.4e-07
NRMSD			7.0e-03	7.0e-03	6.6e-03	6.6e-03
$\bar{S}$			1.6e-03	1.6e-03	1.4e-03	1.5e-03
Wall Time			0:04:29	0:01:46	0:06:02	0:08:34
Results			1		2	

For this data set, gradient search has found two Pareto optima for S_{Front} with details shown in Table 19 and Figure 23. Here, NRMSD and $\bar{S}$ agree on the best choice. Note that the applied multi-start strategy for gradient search is not specifically designed for finding Pareto optima, as the metrics are combined into one objective for each optimizer run. However, Pareto optima can be found as a side effect when the results of several runs are compared at the level of individual metrics.

Table 19: Pareto optima obtained by gradient search with S_{Front} for experimental dextran pulses with attached column.

Parameter	*Optimum 1*	*Optimum 2*
ε_c	3.3e-01	3.4e-01
D_c	3.4e-07	2.5e-07
NRMSD	6.6e-03	5.0e-02
$\bar{S}$	1.4e-03	1.5e-03

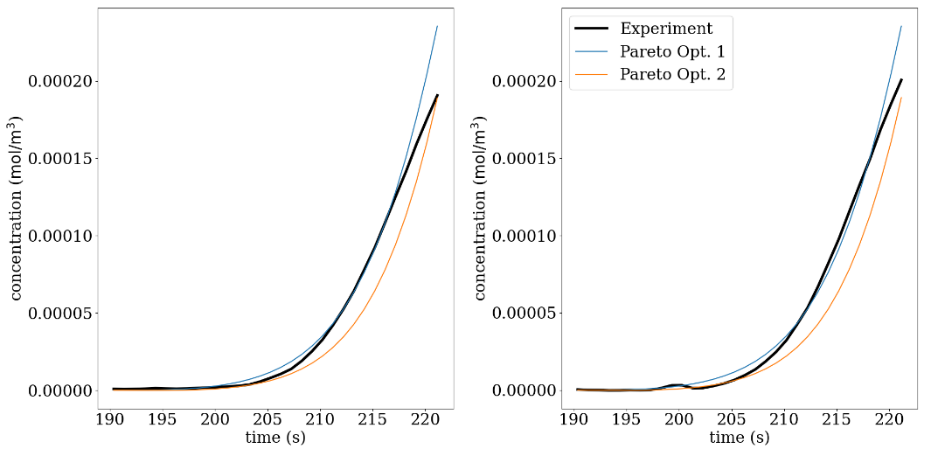

Figure 23: Two replicates of dextran pulse experiment with attached column. Experimental data and estimated chromatograms. Pareto optima obtained by GA with S_{Front}.

2.7.3. Non-Binding Protein Pulses

In the third stage, particle porosity, ε_p, film diffusion, k_f, and pore diffusion, D_p, are estimated from a protein pulse under non-binding conditions, i.e., high salt. Using the target protein instead of salt as non-binding but pore penetrating tracer is more reliable, as salt diffuses faster and has a better pore accessibility. Parameter estimation results are shown in Table 20 and Figure 24. Runtimes are between six and 22 minutes for GA and are about 15 minutes for gradient search. The NRMSD is very similar for the compared methods. The fitted chromatograms are also similar with the largest variations around the peak maximum. The porosity estimates are identical.

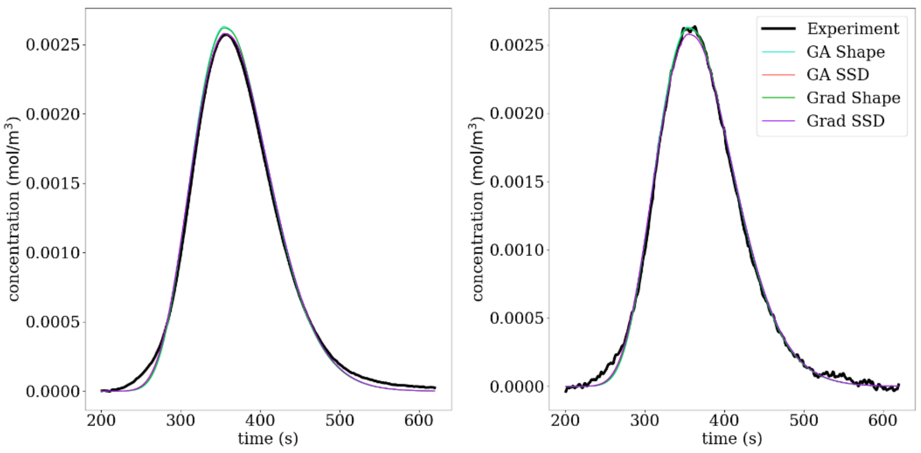

Figure 24: Two replicates of protein pulse experiment under non-binding conditions. Experimental data and estimated chromatograms.

Table 20: Protein pulse experiment under non-binding conditions. Parameter bounds, estimated parameters, and performance indicators.

	Bounds		*GA*		*Gradient*	
	Lower	Upper	S_{Peak}	S_{SSD}	S_{Peak}	S_{SSD}
ε_p	2.0e-01	5.0e-01	3.3e-01	3.3e-01	3.3e-01	3.3e-01
k_f	1.0e-12	1.0e-05	2.6e-06	1.3e-06	2.0e-06	1.3e-06
D_p	1.0e-12	1.0e-05	2.2e-11	3.1e-11	2.5e-11	3.0e-11
NRMSD			2.1e-02	1.9e-02	2.1e-02	1.9e-02
$\bar{S}$			1.1e-02	1.6e-02	1.1e-02	1.5e-02
Wall Time			0:21:16	0:06:16	0:16:02	0:14:08
Results			4		2	

The estimates for particle porosity and pore diffusion show relatively large differences between the scores, partly above 100%, while similar results are obtained by both algorithms. Moreover, both algorithms found multiple Pareto optima for the S_{Peak} score. Table 21 shows the Pareto optima found with GA and S_{Peak}. They differ by more than 100% in some parameters, and the lowest mean conflicts with the lowest NRMSD. Table 22 shows the Pareto optima found with gradient search and S_{Peak}. They are closer to each other, and the lowest mean is not in conflict with the lowest NRMSD. Figure 25 compares the simulated chromatograms of the four Pareto optima of GA and S_{Peak} and the single optimum of gradient search and S_{SSD}, which is the standard method for parameter estimation in chromatography. One Pareto optimum matches the peak position in one experiment better and one Pareto optimum matches the peak position in the other experiment better, while the other two Pareto optima have time offsets but describe the peak shape in both experiments better. This inherent conflict is caused by a 0.4 second signal offset between the two experiments which is likely caused by slightly different pump delays in each experiment. As introduced and comprehensively discussed in section 2.2, the S_{SSD} cannot handle such inconsistencies in the data, which was a major motivation for designing the new score system. In Table 21, the Pareto optimum with the lowest NRMSD has the largest mean, $\bar{S}$. This Pareto optimum is very similar to single optima found by both algorithms with S_{SSD}. Hence, the results of the new score system include the result of the standard method, but the standard method is not able to reveal deficiencies in the model or data. Even when the final optimum is automatically selected, the existence of several Pareto optima indicates potential issues and should generally trigger manual inspection of the model and data. The selected Pareto optima of GA and gradient search with S_{Peak} are used in the next stage.

Table 21: Pareto optima obtained by GA with S_{Peak} for experimental non-binding protein pulses.

Parameter	*Optimum 1*	*Optimum 2*	*Optimum 3*	*Optimum 4*
ε_p	3.3e-01	3.1e-01	3.0e-01	3.3e-01
k_f	2.6e-06	1.3e-06	7.7e-07	1.3e-06
D_p	2.2e-11	2.8e-11	4.4e-11	3.0e-11
NRMSD	2.1e-02	4.7e-02	8.2e-02	1.9e-02
$\bar{S}$	1.1e-02	1.2e-02	1.3e-02	1.6e-02

Table 22: Pareto optima obtained by gradient search with S_{Peak} for experimental non-binding protein pulses.

Parameter	*Optimum 1*	*Optimum 2*
ε_p	3.3e-01	3.2e-01
k_f	2.0e-06	1.3e-06
D_p	2.5e-11	2.9e-11
NRMSD	2.1e-02	2.8e-02
$\bar{S}$	1.1e-02	1.2e-02

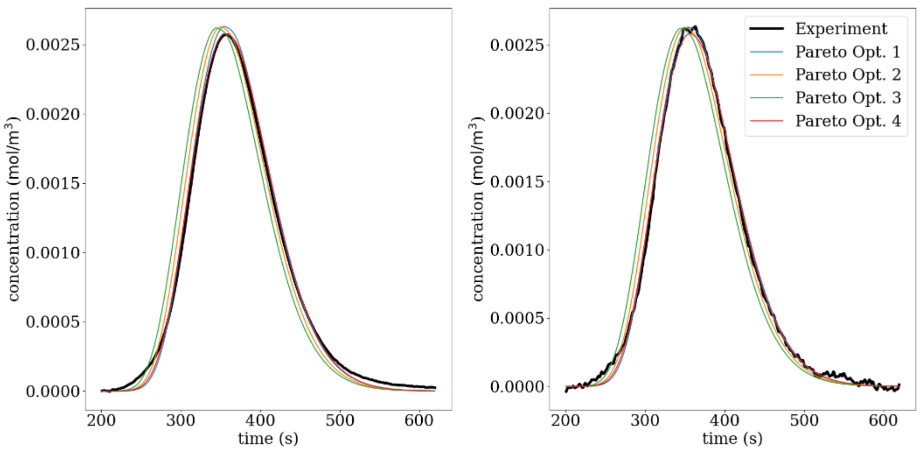

Figure 25: Two replicates of non-binding protein pulse. Experimental data and estimated chromatograms. Pareto optima obtained by GA with S_{Peak}.

2.7.4. Gradient Elution with Partial Fractionation

In the fourth and last stage, the parameters of a two component SMA model are estimated from two gradient elution experiments with different loading times and gradient slopes. One experiment has fractionation data available (Figure 26) while the second features an extended loading time (Figure 27). Experimental details are described in section 1.6. Each fraction is 96 seconds in length. Precise collection times and pool concentrations are shown in Table 23. The scaled adsorption rates, $\tilde{k}'_a$, the scaled equilibrium constant, $\tilde{k}'_{eq}$, characteristic charge, ν, and shielding coefficient, σ, are estimated. The adsorption and desorption rates, $\tilde{k}_a$ and $\tilde{k}_d$, are determined from the parameter transformation in eq. 15-16. The ionic capacity, Λ, was separately determined by titration. The parameters are estimated using the S^*_{Peak} score on the sum chromatogram and the S^*_{Gauss} score on the fractionation data. S^*_{Peak} with six metrics on three peaks and S^*_{Gauss} with three metrics on two peaks and two components results in $1 \cdot 3 \cdot 6 + 2 \cdot 2 \cdot 3 = 30$ objectives. The S_{SSD} score is applied to the sum chromatogram and to the fractionation data of each component.

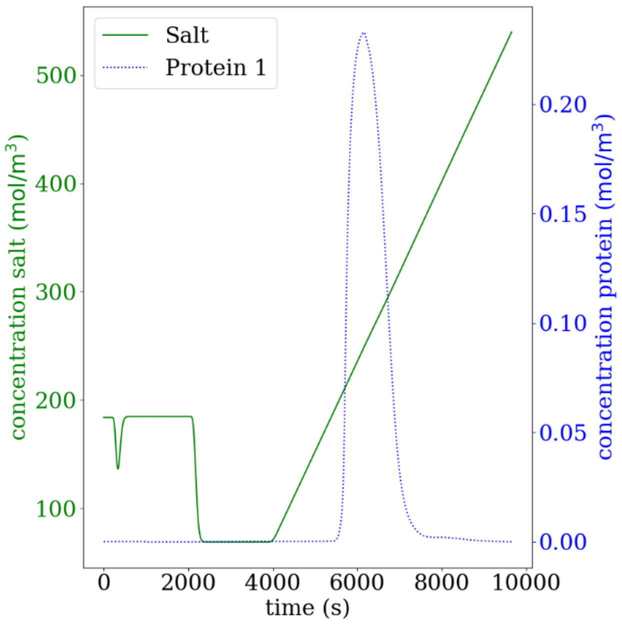

Figure 26: Experimental Gradient Elution Experiment 1: Salt profile overlaid with ground experimental data.

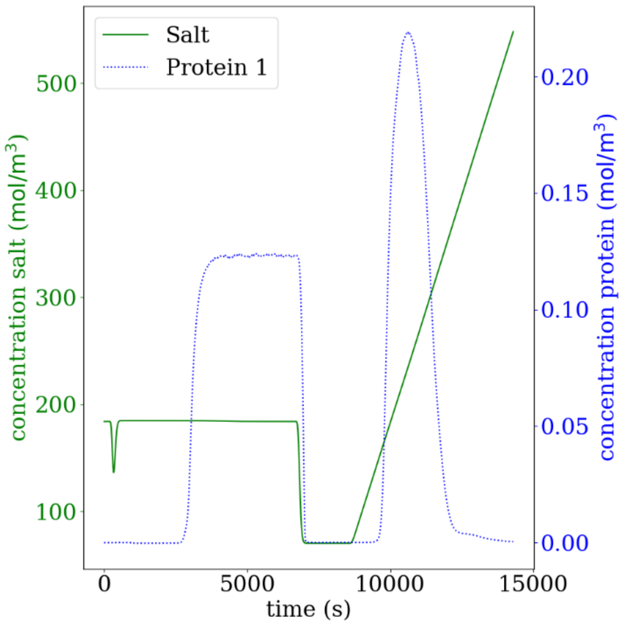

Figure 27: Experimental Gradient Elution Experiment 2: Salt profile overlaid with ground experimental data.

Table 23: Fractionation times and measured pool concentrations of gradient elution experiment.

Collect times (s)		*Pool concentrations (mol/m^3)*	
Start	Stop	Protein 1	Protein 2
5520.12	5616.12	2.21e-03	1.11e-06
5616.12	5712.12	1.70e-02	4.28e-05
5712.12	5808.12	1.12e-01	1.12e-04
5808.12	5904.12	2.03e-01	2.03e-04
5904.12	6000.12	2.34e-01	2.35e-04
6000.12	6096.12	2.49e-01	2.50e-04
6096.12	6192.12	2.56e-01	1.28e-04
6192.12	6288.12	2.54e-01	2.55e-04
6288.12	6384.12	2.44e-01	2.45e-04
6384.12	6480.12	2.29e-01	2.30e-04
6480.12	6576.12	2.03e-01	3.06e-04
6576.12	6672.12	1.73e-01	4.34e-04
6672.12	6768.12	1.30e-01	7.24e-04
6768.12	6864.12	9.43e-02	1.41e-03
6864.12	6960.12	5.59e-02	2.53e-03
6960.12	7056.12	3.16e-02	3.78e-03
7056.12	7152.12	1.57e-02	3.71e-03
7152.12	7248.12	9.13e-03	3.01e-03
7248.12	7344.12	6.01e-03	2.05e-03
7344.12	7440.12	3.99e-03	1.25e-03
7440.12	7536.12	2.89e-03	7.88e-04
7536.12	7632.12	2.08e-03	4.90e-04
7632.12	7728.12	1.76e-03	3.81e-04
7728.12	7824.12	1.55e-03	3.37e-04
7824.12	7920.12	1.44e-03	3.32e-04
7920.12	8016.12	1.46e-03	3.62e-04
8016.12	8112.12	1.39e-03	3.77e-04
8112.12	8208.12	1.29e-03	3.70e-04

Results are shown in Table 24 and Figure 28. The compute times of the different algorithm and score combinations range from ca. 1 to 18 days. The GA appears slower than gradient search, but the reported GA runtimes include local refinement by gradient search. For the $S^*_{Peak} \cup S^*_{Gauss}$ score, 3 days of GA runtime were followed by 5 days of local refinement, and for the S_{SSD} score, 5 hours of GA runtime were followed by 17 days of local refinement. In both cases, local refinement hardly improved the final result, and the same has been observed in previous stages. With synthetic data local refinement can improve the solution up to the limits of numerical precision due to the model being able to perfectly explain the synthetic data. With experimental data the model can't perfectly explain the data and many small steps are taken to make marginal improvements to the results. The GA can quickly get close enough to an optimal solution that local refinement is left with making small refinements to match imperfect data to an imperfect model. In conclusion, the total runtime can be substantially reduced by skipping the local refinement.

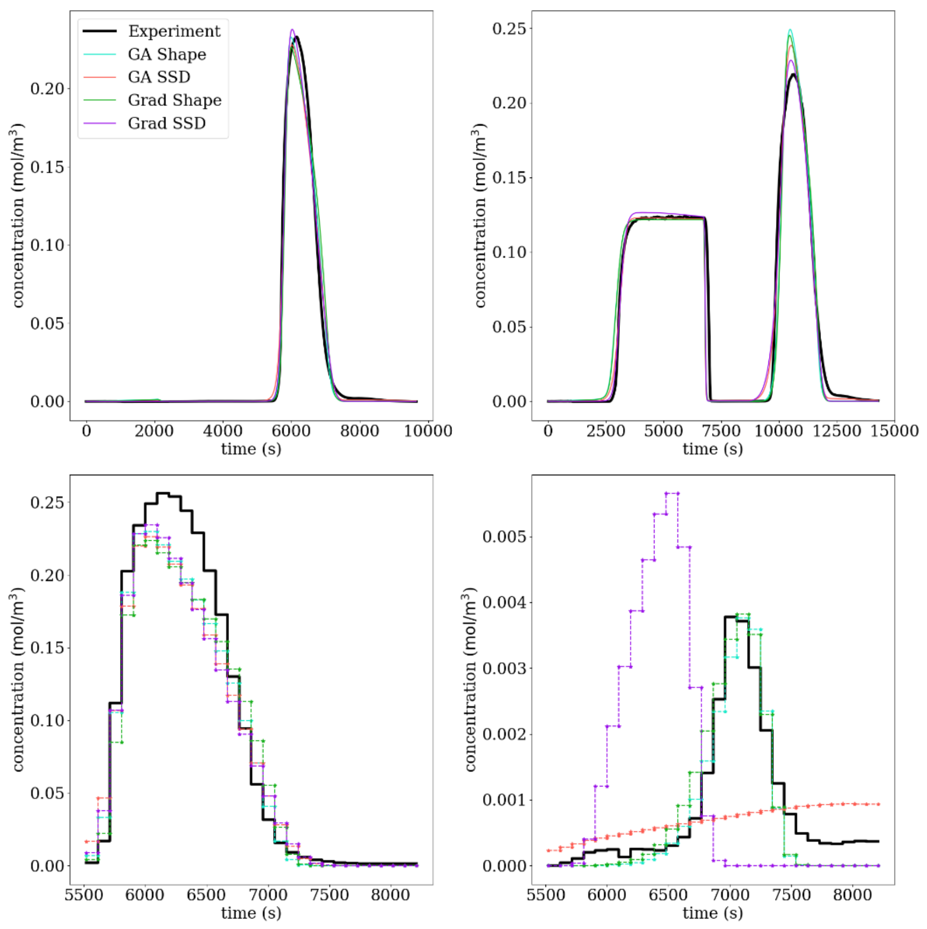

Figure 28: Gradient elution experiment with different loading times and gradient slopes. Experimental data and estimated chromatograms. Sum signal of first experiment (top left) and second experiment (top right). Fractionation data of component 1 (bottom left) and component 2 (bottom right) in first experiment.

Table 24: Gradient elution experiment with partial fractionation. Parameter bounds, estimated parameters and performance indicators.

	Bounds		*GA*		*Gradient*	
	Lower	Upper	$S^*_{Peak} \cup S^*_{Gauss}$	S_{SSD}	$S^*_{Peak} \cup S^*_{Gauss}$	S_{SSD}
$\tilde{k}_{a,1}$	1.0e-06	1.0e+06	5.1e+00	9.8e+06	1.2e+01	7.1e-01
$\tilde{k}_{d,1}$	-	-	3.4e+04	7.6e+08	2.3e+05	1.7e+01
$\tilde{k}_{eq,1}$	1.0e-06	1.0e+06	1.5e-04	1.3e-02	5.4e-05	4.1e-02
ν_1	1.0e+00	2.0e+02	1.5e+01	8.2e+00	1.7e+01	6.7e+00
σ_1	1.0e+00	2.0e+02	5.0e+01	4.4e+01	5.1e+01	4.1e+01
$\tilde{k}_{a,2}$	1.0e-06	1.0e+06	7.4e-03	1.5e+04	8.9e-03	1.0e+06
$\tilde{k}_{d,2}$	-	-	2.6e+03	1.6e+03	1.0e+03	1.0e+12
$\tilde{k}_{eq,2}$	1.0e-06	1.0e+06	2.9e-06	9.6e+00	8.7e-06	1.0e-06
ν_2	1.0e+00	2.0e+02	2.3e+01	1.0e+00	2.1e+01	2.1e+01
σ_2	1.0e+00	2.0e+02	9.5e+01	1.0e+02	3.9e+01	2.0e+02
NRMSD			3.9e-01	3.7e-01	3.9e-01	3.4e-01
$\bar{S}$			9.0e-02	1.9e-01	9.1e-02	1.0e+00
Wall Time			8 days	17 days	0 days	6 days

	1:33:00	9:42:53	19:14:13	12:06:01
Results	13		3	

The algorithm and score combinations differ not only in their runtimes but also in the achieved parameter estimates. The NRMSD is relatively low for all algorithm and score combinations. Even though it is slightly smaller for S_{SSD} than for $S^*_{Peak} \cup S^*_{Gauss}$, for both algorithms, some alarming issues are observed for S_{SSD}. For GA, the characteristic charge of the second component, ν_2, is at the lower bound, and for gradient search, the shielding factor of the second component, σ_2, is at the upper bound. These values indicate unrealistic parameter values, as wide search intervals were chosen around typically observed values for monoclonal antibodies. Despite the similar NRMSD, the scaled adsorption constants of the first component, $\tilde{k}_{a,1}$, differ by more than 5 orders of magnitude between the search algorithms, and the same is true for the scaled equilibrium constant of the second component, $\tilde{k}_{eq,2}$. This indicates poor identifiability of the estimated parameters with the S_{SSD} score. Moreover, the corresponding chromatograms do not at all match the fractionation data of the second component, as can be seen in Figure *28*.

In stark contrast, the $S^*_{Peak} \cup S^*_{Gauss}$ score was able to guide both search algorithms towards satisfying matches between simulation and experiment for all peaks of both components in both experiments. The corresponding parameter estimates of both search algorithms are relatively similar. Again, rigorous uncertainty analysis is not in the scope of this publication. Moreover, the GA with $S^*_{Peak} \cup S^*_{Gauss}$ result even features an initial breakthrough peak in the first experiment, which is tiny but important, as it indicates saturation of the column.

For the $S^*_{Peak} \cup S^*_{Gauss}$ score, 13 Pareto optima were found by the GA, Table 25, and three by gradient search, Table 26. For both search algorithms, the NRMSD of the Pareto optima conflict with the respective mean, $\bar{S}$. The GA results are compared in Figure 29. All Pareto optima, except one, match all peaks of both components in both experiments. These twelve Pareto optima indicate a tradeoff between the two experiments. However, none of them perfectly matches one experiment or the other. This indicates that the mechanistic model does not fully capture all features of the process. The observed deviations are likely caused by a combination of non-ideal hydrodynamics, complex binding processes and experimental errors. Figure 29 provides rich information on the Pareto optima that can be related to expert knowledge for selecting the most suitable parameters for specific applications of the calibrated model. Specific applications of the model described here can entail process robustness investigation. Simulations of scenarios with varying process input parameters that are known to vary between lots, such as total protein concentration in the feed solution or ionic strength of elution buffers, can inform decision making in case of process performance deviations. Further applications can range from screening process parameters, such as stop collect criteria, to inform process optimization experiments, to risk-analyses for technical transfer activities.

Table 25: Pareto optima obtained by GA with $S^*_{Peak} \cup S^*_{Gauss}$ for experimental gradient elution case study.

Parameter	*Optimum 1*	*Optimum 2*	*Optimum 3*	*Optimum 4*	*Optimum 5*	*Optimum 6*	*Optimum 7*	*Optimum 8*	*Optimum 9*	*Optimum 10*	*Optimum 11*	*Optimum 12*	*Optimum 13*
$\tilde{k}_{a,1}$	5.1e+00	2.3e+01	6.1e+01	2.2e+02	1.8e+03	1.4e+03	9.3e+01	3.8e+02	4.9e+02	3.1e+02	2.2e+03	5.0e+05	5.5e+05

$\tilde{k}_{d,1}$	3.4e+04	1.5e+05	4.5e+05	1.5e+06	1.4e+07	1.1e+07	3.0e+06	6.1e+08	1.0e+09	6.0e+08	5.5e+09	1.0e+07	3.2e+07
$\tilde{k}_{eq,1}$	1.5e-04	1.5e-04	1.4e-04	1.5e-04	1.3e-04	1.3e-04	3.1e-05	6.2e-07	4.8e-07	5.1e-07	3.9e-07	5.0e-02	1.8e-02
ν_1	1.5e+01	1.5e+01	1.5e+01	1.5e+01	1.5e+01	1.5e+01	1.7e+01	2.4e+01	2.5e+01	2.5e+01	2.6e+01	6.2e+00	7.8e+00
σ_1	5.0e+01	5.0e+01	5.0e+01	5.0e+01	5.0e+01	5.0e+01	5.1e+01	5.7e+01	5.4e+01	5.6e+01	5.3e+01	3.9e+01	4.5e+01
$\tilde{k}_{a,2}$	7.4e-03	7.7e-03	6.8e-03	7.4e-03	6.8e-03	6.9e-03	6.6e-03	7.7e-03	1.0e-02	7.7e-03	2.1e-02	1.1e-02	3.9e-02
$\tilde{k}_{d,2}$	2.6e+03	2.6e+03	2.9e+03	2.6e+03	2.8e+03	2.8e+03	2.5e+03	2.8e+03	4.2e+03	1.9e+03	5.5e+03	1.1e+03	3.9e+05
$\tilde{k}_{eq,2}$	2.9e-06	2.9e-06	2.4e-06	2.9e-06	2.5e-06	2.5e-06	2.7e-06	2.8e-06	2.5e-06	4.1e-06	3.8e-06	1.1e-05	1.0e-07
ν_2	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.3e+01	2.4e+01	2.1e+01	2.3e+01
σ_2	9.5e+01	9.6e+01	7.5e+01	9.5e+01	6.6e+01	7.7e+01	4.0e+01	4.0e+00	9.8e+01	9.6e+01	9.8e+01	9.8e+01	1.0e+00
NRMSE	3.9e-01	4.1e-01	4.2e-01	4.2e-01	4.2e-01	4.3e-01	4.5e-01	4.8e-01	4.7e-01	4.9e-01	4.4e-01	3.6e-01	3.7e-01
$\bar{S}$	9.0e-02	9.1e-02	9.2e-02	9.2e-02	9.2e-02	9.3e-02	9.5e-02	9.8e-02	1.0e-01	1.1e-01	1.1e-01	1.3e-01	1.0e+00

Table 26: Pareto optima obtained by gradient search with $S^*_{Peak} \cup S^*_{Gauss}$ for experimental gradient elution case study.

Parameter	*Optimum 1*	*Optimum 2*	*Optimum 3*
$\tilde{k}_{a,1}$	1.2e+01	4.5e+01	8.5e+01
$\tilde{k}_{d,1}$	2.3e+05	4.0e+06	1.2e+04
$\tilde{k}_{eq,1}$	5.4e-05	1.1e-05	7.2e-03
ν_1	1.7e+01	2.0e+01	9.1e+00
σ_1	5.1e+01	5.7e+01	4.2e+01
$\tilde{k}_{a,2}$	8.9e-03	7.8e-03	5.3e-01
$\tilde{k}_{d,2}$	1.0e+03	7.9e+02	1.0e-02
$\tilde{k}_{eq,2}$	8.7e-06	1.0e-05	5.1e+01
ν_2	2.1e+01	2.1e+01	1.3e+02
σ_2	3.9e+01	1.4e+02	6.0e+01
NRMSE	3.9e-01	4.5e-01	3.8e-01
$\bar{S}$	9.1e-02	9.6e-02	1.0e+00

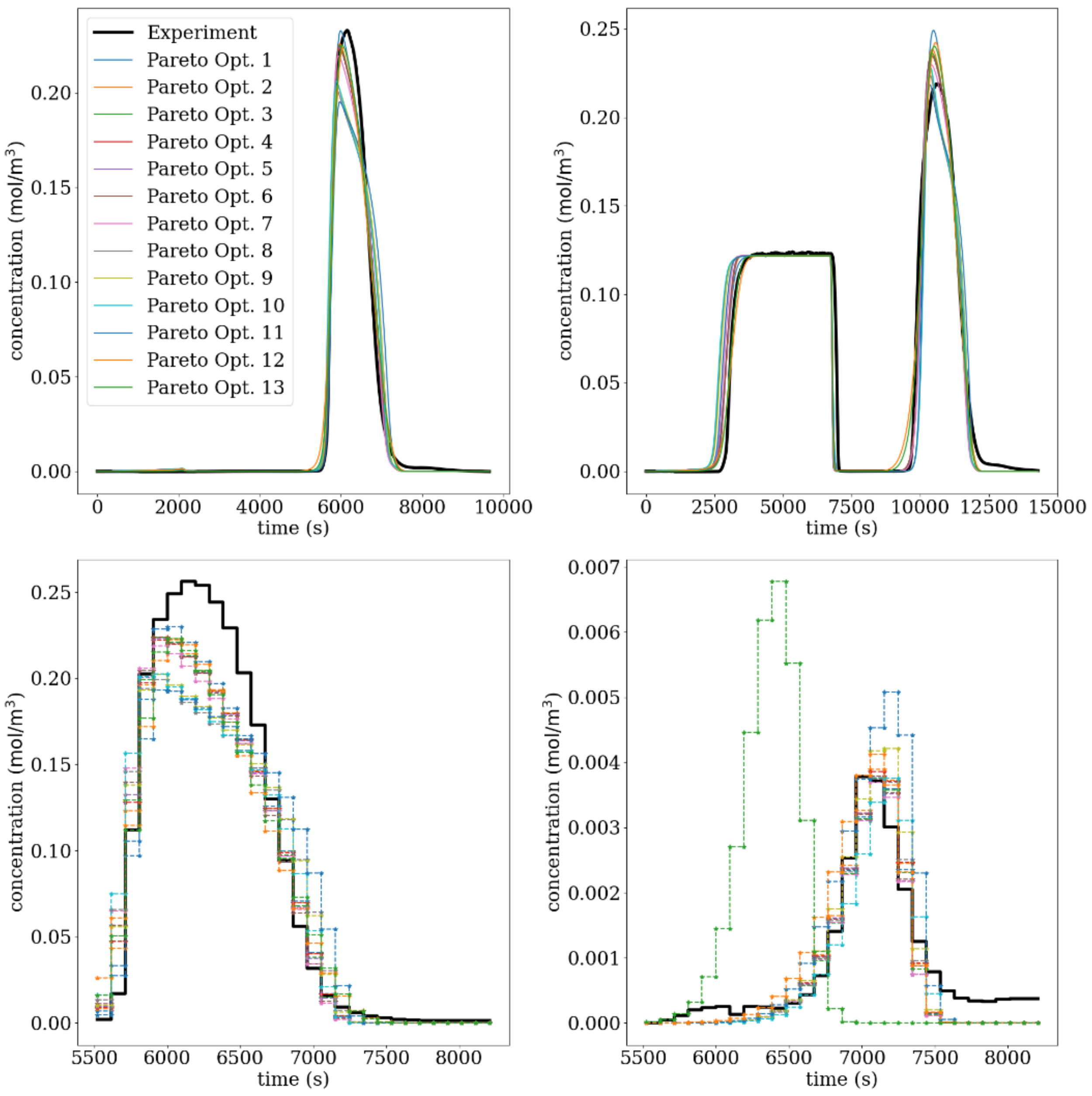

Figure 29: Gradient elution experiment with different loading times and gradient slopes. Experimental data and estimated chromatograms. Pareto optima obtained by GA with $S_{Peak}^{*} \cup S_{Gauss}^{*}$. Sum signal of first experiment (top left) and second experiment (top right). Fractionation data of component 1 (bottom left) and component 2 (bottom right) in first experiment.

2.8. Summary

A novel score system for estimating chromatography model parameters has been introduced and demonstrated using both synthetic as well as experimental case studies. In contrast to least squares estimation, which is the de facto standard approach and mostly combined with single-objective gradient search, the new score system provides multiple objectives (metrics) that are simultaneously optimized. Typical objectives are the shape, position, and height of individual peaks. Even when these objectives are not in conflict, which is typically the case for synthetic data, they can help to improve convergence of the search algorithm, particularly for poor start values and in initial phases of the optimization, by allowing progress in one objective while sacrificing another. It has also been observed that a multi-objective GA is more robust for complex problems. The GA is also parallelized with progress monitoring for computational efficiency on compute clusters but also on multi-core processors in personal computers.

With a robust method for parameter estimation of complex problems the logical next question is the uncertainty on the parameters. It is important to know how well defined the parameters are and what the impact of the uncertainty is on the resulting chromatogram. The next chapter continues with this problem.

3. Uncertainty Quantification

No model is perfect, and no model can perfectly explain experimental data. Fundamentally the question we need to answer is how accurate a chosen combination of model and parameters is. While it is important to compare the probability of different models correctly explaining the data, this paper is focused on the probability distribution of parameters in a single model. In the case of chromatography, we are normally looking at the accuracy of predicting chromatograms or derived metrics like yield and purity. The overall goal is to determine the probability distribution of each parameter and to show the impact of this uncertainty on the chromatogram. The parameter distributions can then be visualized by sampling from these distributions and overplotting many chromatograms and coloring them according to the parameter distribution probability. This results in a confidence tube with higher and lower probability regions shown. The tube can then be checked against experimental data to see if the prediction is reasonable.

The parameters have a multivariate probability distribution associated with them based on the systemic and random errors in the experiments and the errors in the model. No model perfectly explains the data and no experiment is performed without any errors and both types of errors show up in the parameter probability distribution. As a concrete example assume the goal is to determine the column porosity and axial dispersion by running a non-pore penetrating and non-interacting pulse through a column. An obvious error would be concentration errors in the loading solution. However, there are also other errors such as how accurately the pump flow rate is and how long it takes the pump to start. The question can then be refined to what is the probability distribution of the column porosity and axial dispersion given errors in loading concentration, pump flow rates and pump delays which can be written as P(column porosity, axial dispersion | loading concentration, pump flow rate, pump delay). This is a conditional probability and in general is written as P(A|B) which is the probability of event A given event B.

Uncertainty of parameters is deceptively simple. An obvious approach would be to run a single experiment with multiple replicates, fit the model to each one and then determine the mean and standard deviation of each parameter. With one experiment and 3 replicates the mean and standard deviation for each parameter can be calculated from a parameter estimation on each replicate. If the parameter distribution is not a normal distribution the mean and standard deviation can be misleading and more replicates will be needed to reveal the true distribution and coupling between parameters. In practice different experiments are needed to estimate different parameters and each needs multiple replicates.

An alternative to performing many experiments is to use a stage-wise estimation procedure [52]. In chromatography a typical system would include external tubing and mixing valves along with a column containing porous resin. A stage-wise approach breaks such a system into stages so that parameters can be estimated independently. For instance, the column can be disconnected, and the volume and dispersion can be estimated for the external tubing and mixing valves. The next stage would then connect the column but use a non-pore penetrating tracer to estimate the column porosity and axial dispersion in the column. However, instead of just using the best value from the previous stage the parameters' distributions can be used as prior information. This approach allows a complex system to be broken up into smaller pieces and error information carried forward. This approach allows incorporating manufacturer information into the estimation as additional prior information. Manufacturers often provide information about the accuracy of pumps, sensors, tolerances on tubing etc. and this information is valuable prior information to incorporate.

Mathematically this incremental refinement approach is based on Bayes' theorem eq. 33. For events A and B there is a probability, P(A), of A occurring and an independent probability, P(B), of B occurring. There are also the conditional probability of A happening given B is true, P(A|B), and the probability of B happening given A is true, P(B|A). For this application A represents our parameters and B represents the data. P(A) is called the prior and it is our initial probability distribution for A. In the case where we have no prior knowledge of A, a bounded uniform distribution is normally used. P(A|B) is the posterior, and it represents the belief in A given B. P(B) is called the evidence and represents the probability distribution of the data and importantly it is a constant. In the context of uncertainty quantification, P(B|A) is the error model and represents the probability of the data (B) being observed given the parameters (A). The posterior from one stage can be used as the prior in the next stage.

With Bayes' theorem the stage-wise example can directly implemented. The posterior of one stage is used as a prior next stage. The first stage can then be written more precisely as determining the posterior P(volume, tubing dispersion | chromatogram). For the next stage, where the column is connected and a non-pore penetrating tracer is used, the posterior is P(volume, tubing dispersion, axial dispersion, column porosity | chromatogram) with the previous stage's posterior P(volume, tubing dispersion) used as a prior. At each stage the number of variables increase but the previous variable distributions are included. For example, uncertainty in the column porosity will also result in increased uncertainty in the particle porosity due to having similar impacts on the chromatogram. This make Bayes' theorem extremely powerful by allowing stages to be chained together. If a parameter or set of parameters shows a uniform distribution or too wide of a distribution, additional stages or experiments within a stage can be added to attempt to improve the parameter(s) distribution.

$$P(A|B) = \frac{P(B|A)\ \cdot P(A)}{P(B)} \tag{35}$$

3.1. Markov Chain Monte Carlo

In general, there is no analytical solution for eq. 33 and instead Markov Chain Monte Carlo (MCMC) is used. MCMC allows the creation of a chain of dependent samples whose distribution approximates the posterior, P(A|B). A simple MCMC algorithm takes a single starting point and proposes a step to a new point and then accepts or rejects it based on the relative probability of the two points. In this way it will walk around, and the elements of the resulting chain approximate the sought posterior distribution. There are various algorithms for proposing the next point in the chain and the better the algorithm is at proposing steps the fewer steps are needed in the chain to converge to the posterior distribution. In this paper, we use emcee [65] which is an a pure python implementation of an affine-invariant MCMC ensemble sampler [66]. It uses an ensemble of walkers working together to sample more efficiently, and this also allows the walkers to be run in parallel which is a large benefit on modern computers. These parallel walkers create parallel chains. Emcee uses a log probability function shown in eq. 34. Only the relative probability of two sets of parameters matters in MCMC and since P(B) is a constant it can be left out which results in eq. 37. This leaves the error model and the prior as the only required terms.

$$\log\big(P(A|B)\big) = \log\big(P(B|A)\big) + \log\big(P(A)\big) - \log\big(P(B)\big) \tag{36}$$

$$\log\big(P(A|B)\big) \propto \log\big(P(B|A)\big) + \log\big(P(A)\big) \tag{37}$$

The default error model most often used in MCMC is to assume that all errors are random, independent, and normally distributed. Such an error model is inappropriate to use in general for chromatography model parameters although it may apply in some special cases. A simple error is a pump delay and when performing experiments, a delay can be heard between when the pump is instructed to start and when it starts. These delays can be quite significant, especially when estimating porosity using a narrow pulse of only a few seconds. Pump delays mathematically have similar impacts on the chromatogram as a different porosity. The impact on the chromatogram of a single pump delay is to shift the entire chromatogram and it can only be shifted later in time, the error is a systematic error and clearly not random or independent for each concentration measurement. The same analysis applies to pump flow rate errors. A pump flowing 1% faster does not just result in a peak appearing earlier, the peak also appears sharper due to less time for diffusion. Pumps also tend to be consistent so if the pump is 1% high at the beginning it will tend to stay that way for the entire run which is also clearly not independent. Due to such errors a different error model is needed. We propose here a different type of extensible error model able to deal with errors in chromatography that are not random, independent, and normally distributed.

In MCMC there are many ways the algorithm can propose new points for the walkers to evaluate and this is called a move. There are many types of moves available [65]–[68] with newer ones designed to take advantage of ensembles of walkers. It is important to remember that these are proposals for new points and acceptance, or rejection is controlled by eq. 3. The purpose of more advanced moves is to choose points that get accepted more often and result in less correlation in the chain. The simplest moves are the walk and stretch moves [65], [66]. The walk move allows each walker to move randomly in the area around their current point. The walk move is the simplest move but also the least efficient. The stretch move is slightly more complicated and involves walking in the direction of the current position of the other walkers in the ensemble and thus allowing the entire ensemble to walk more efficiently. The differential evolution move [67] proposes a new location by selecting two walkers at random and making a step in the same direction as the vector connecting the two walkers' current location. The differential evolution move does have a parameter, gamma, which controls the size of the step taken of $2.38/\sqrt{2 \cdot \#parameters}$ and this was left at the default value. A further adaptation of the differential algorithm added a so-called snooker update [68] which increases walker diversity and is designed to be used with the differential evolution move. The same paper proposes a mixture of moves that is also used here with 81% differential evolution, 9% differential evolution with gamma set to 1.0 and 10% differential evolution snooker moves. This mixture was found to improve sampling efficiency compared to using stretch or walk moves in testing which reduces the number of steps that must be taken before the chain converges.

A major goal of this project is for the entire uncertainty quantification process to be robust, automatic, and consistent posterior distributions from run to run, and mitigate problems associated with MCMC in high dimensions. To understand the latter a small digression is needed and that is to look at the curse of dimensionality. Consider a unit N-sphere and the containing N-cube and look at the ratio of the volume of the sphere to the volume of the cube. As can be seen in Figure 30, in 3-dimensions the ratio is about 0.5. However, as dimensionality increases the sphere makes up an ever-decreasing percentage of the volume, and by 20-dimensions the ratio is down to 2.5e-8. In this example, the N-cube represents the boundaries of the MCMC search space, and the N-sphere represents the high probability region. In practice the constraints in each dimension are often far larger than the high probability region and the problem is correspondingly worse. Consequently, any walkers started in low probability space tend to stay trapped in low probability space and degrade the performance of the MCMC process. These issues can be mitigated by narrowing the boundaries automatically and by adjusting the starting point of walkers with the details explained later. Another major problem of MCMC is gathering sufficiently many samples. To make the system robust and automatic the chains are monitored for convergence and sampling stops when enough samples are taken with the details covered in section 3.1.1.

$$\frac{V_{sphere}}{V_{cube}}(n) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right) * 2^{n}} \tag{38}$$

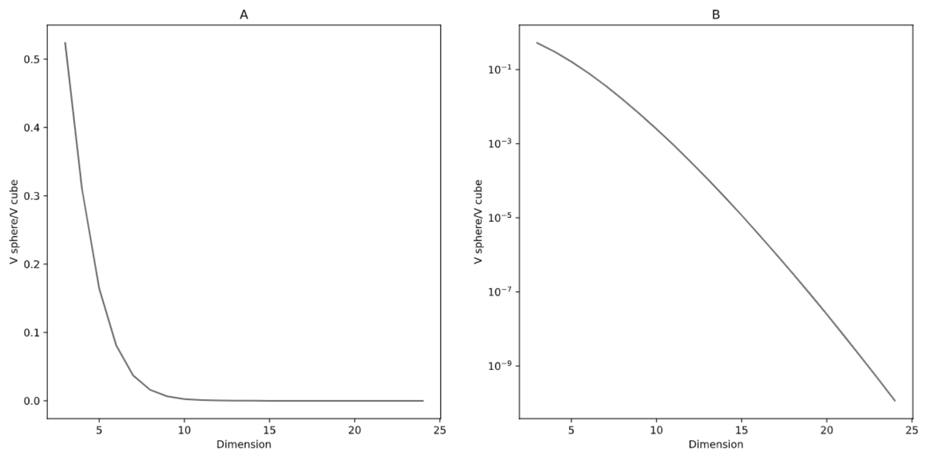

Figure 30: Volume ratio of unit N-sphere to containing N-cube on linear (A) and log (B) ordinates.

3.1.1. Convergence and Accuracy Metrics

The first question to arise after MCMC has been run and the posterior obtained is how to analyze it. It is important to note that the chains approximate the true posterior, and this approximation is not of equal quality along the entire posterior. The tails of the posterior converge slower than the rest of the posterior, partially due to fewer samples in the tails. After the posterior is obtained, a credible interval is normally calculated to provide an easy metric for the width of the bulk of the posterior distribution. Typically, an equal tailed interval (ETI) is determined, for example selecting from 5% to 95% of the posterior distribution. An ETI works well if the posterior is approximately symmetric. If the distribution is asymmetric a situation can occur where areas outside the selected interval are higher probability than areas included in the interval. An alternative selection metric is the highest density interval (HDI) [69]. All points in the HDI have higher probability than points outside the HDI. The HDI can contain a user defined portion of the posterior distribution. The equivalent of a 5% to 95% ETI interval would be an 90% HDI. For symmetric distributions the HDI is the same as ETI. HDI is more complex to calculate than an ETI and we use the Arviz library [70] for this.

An immediate problem that arises with MCMC is how many steps of the MCMC algorithm are needed. This question does not have a simple answer. The steps are correlated and measuring this correlation is difficult and problem dependent. CADET-Match uses the integrated autocorrelation time (IAT), as recommended [66] for affine invariant MCMC samplers, to measure the correlation between samples. The IAT works as a direct measure of how many steps are taken on average before a new independent sample is generated. The IAT is parameter specific and can be used as a stopping condition. A typical approach is to run until all parameters have enough samples and thus stopping is determined by the slowest converging parameter. There is one small problem with this approach, estimating the IAT itself takes many samples and the same is true for other metrics. With emcee, around 50 times the IAT is required for reliably estimating the IAT [71].

While IAT measures the correlation between samples it does not directly measure how accurate the posterior distribution is. A direct way of measuring the accuracy of the posterior distribution is the Monte Carlo Standard Error (MCSE). The MCSE is the uncertainty in the posterior caused by sampling error due to using a chain of dependent samples. An example should make this clearer. While it is simple to find the mean of the posterior distribution the MCSE measures the error of that mean value, essentially how sure we are that is the correct mean given it was calculated from a chain of dependent samples. MCSE can also be applied to a single point, such as 5%, 95%, etc., of the posterior and return a point estimate for the error. If the posterior is broken up into 1% intervals, for example, and the MCSE calculated at each point a maximum bound on the error in the posterior can be calculated. MCSE is calculated using the Arviz library. Arviz also implements arbitrary point estimates for MCSE beyond the standard application of these metrics to the mean value based on a recent paper [72].

3.1.2. Parameter Estimation

A calibrated model is a pre-requisite for the error model. Calibrating a model typically requires parameter estimation. Parameter estimation in CADET-Match has been covered in section 2 but a few important details will be mentioned here that are relevant to uncertainty quantification. CADET-Match uses a variety of methods for parameter estimation with gradient based and gradient free search methods. The goal for CADET-Match can be single or multiple-objective. For uncertainty quantification only the multiple-objective approach is used here. A multi-objective approach captures more information from the chromatogram and can independently account for the shape, position, and height of the peak and all of these are impacted in different ways by experimental errors. With only a single objective multiple errors can produce the same value while having different probabilities of occurring such as changing the shape of a peak vs shifting it in time. A goal is made up of one or more metrics. A metric is a single scalar value. Some of the metrics include the shape of the chromatogram, the peak height, the time of peak max, the shape of the derivative, etc. By using multiple metrics, a goal can be created to solve a particular problem such as fitting only the front of a peak or fitting fractionation data. CADET-Match supports multiple experiments with multiple components and metrics can be applied to any part of the chromatogram. The resulting goal is a vector of metrics. This vector of metrics measures how similar a proposed parameter set's simulated chromatogram data is to the experimental data being matched. A search algorithm is used to propose parameters and find ones that result in data similar to the matched data. There are two important features of this parameter estimation process for uncertainty quantification. The first feature is that one or more experiments can be reduced to a vector comparing aspects of the chromatogram(s) to the data which reduces the dimensionality of the goal compared to the full chromatogram(s) while keeping more information than a single-objective. The second major feature is that a multi-objective goal results in a Pareto front instead of a single value where there are conflicts between metrics. The metrics used are not inherently in conflict and only come into conflict when the data is inconsistent, or the model is incapable of explaining the data. This Pareto front not only helps identify potential issues in experimental data, but it can also be used to create an initial set of starting points for MCMC. The calibrated model for MCMC is chosen from the Pareto front. If there are no conflicts between the metrics the Pareto front will have one parameter set and that parameter set is used. If there is more than one entry on the Pareto front the parameter set with the lowest geometric mean is used. The selected parameter set is also used to create the error model with the details explained in section 3.2.1. The important part for now is the error model is created using many simulations and as has just been covered the multi-objective goal allows us to create a vector comparing the simulation to the data. The result is a matrix of metrics which then needs to be converted to a probability density function to be used as an error model. Kernel Density Estimation (KDE) is used to convert from a matrix to a probability density function.

3.1.3. Kernel Density Estimation

KDE can be thought of as a continuous version of a histogram and unlike a histogram the probability of any parameter set can also be evaluated. KDE is used to create a probability density function of one or more continuous random variables. Once the probability density function is created the probability of a vector of metrics can be calculated and this function is the error model P(B|A). KernelDensity is used from the Python sklearn [49] library. KDE is a machine learning technique and works best when all dimensions have approximately the same feature scale. RobustScaler is used from the sklearn library for rescaling. It scales each dimension by subtracting the median value for each feature and dividing by the inter-quartile range, 25th quantile to 75th quantile, and involves no hyper parameters. KDE is based on the summing up of kernel functions, such as a Gaussian kernel. KDE has only a single scalar hyper parameter, bandwidth, which adjusts the width of these kernels. If the bandwidth is too large it causes over-smoothing and if the bandwidth is too small, it causes under-smoothing. As more samples are added to the system the bandwidth narrows. This causes an issue where with few samples a kernel density estimator is smooth and as samples are added the KDE can become rougher until it starts becoming smoother again. As the number of samples increases the computational and memory requirements of the KDE also substantially increase. There is no analytical way to determine the best bandwidth. CADET-Match follows best practices and uses grid search with n-fold leave-one-out cross-validation to determine the best bandwidth. The data set is split into n equal sized pieces and n-1 of the pieces are used for training with the remaining piece used for testing. This is repeated for all combinations of pieces. CADET-Match uses a 20-fold cross-validation. The typical bandwidth range is from 1e-2 to 1e-1 after scaling. A grid search is performed in 60 steps in log-space from 1e-3 to 1. The number of steps and range where chosen based on testing to ensure robustness of the process. The lowest mean value is determined and all cross-correlation points from one step lower to one step higher in the grid search are selected. A second-degree polynomial is fit to these points and the minimum selected as the optimal bandwidth. This gives a deterministic approach to bandwidth selection and provides run-to-run reproducibility.

3.2. Error Modeling and Uncertainty Quantification

We present here an extensible error model and demonstrate its use on synthetic and industrial data. Error simulations are created from the calibrated model by using the model as a template and adding in pump delays, pump flow rate, and loading concentration changes. These error simulations chromatograms are then converted to a matrix of metrics as per section 3.1.2. The matrix of metrics is then converted to a probability density function as per section 3.1.3. This probability density function is the error model, P(B|A). The error model can then be used to assess the probability of any given chromatogram being explained from the included error sources. This method of adding errors directly to the calibrated model template is very flexible and allows expanding to further sources of error as needed. For uncertainty quantification, new simulations are then created from the base model with varied parameters of interest such as axial dispersion, column porosity etc. and the resulting chromatograms compared against the error model. MCMC is used to determine the uncertainty on each parameter of interest given the errors included in the error model. As per section 3.1 the MCMC algorithm proposes parameters, and they are accepted or rejected based on the error model. The chains formed by accepting and rejecting parameters forms the posterior distribution.

This entire error modeling and uncertainty quantification process is implemented in CADET-Match and freely available on GitHub. It is implemented as a unified multi-step process with a calibrated model as a prerequisite and by default it will also calibrate a starting model from data.

1. Calibrate base model
2. Add error sources to calibrated model
3. Create error model
4. Set MCMC starting point(s) to high probability region(s)
5. Adjust MCMC bounds around high probability region
6. Burn-In and Run MCMC until convergence
7. Post-processing

3.2.1. Advanced Error Modeling

The first step is to construct the error model, P(B|A). Four types of errors are currently considered in this error model, and all have user supplied parameters to control the distribution of errors:

First, pump delays are caused by the delay between when the pump is instructed to start and when it starts. Based on experimentalists hearing the pump turn on and seeing the impact in the data we know there is a delay and that it is significant. A pump delay is implemented by adding an additional period to the base model before a pump starts where no flow occurs. The length of the period for a pump delay is modeled with a random uniform distribution. More elaborate distributions can be applied as more knowledge becomes available.

Second, pump flow rates are extremely stable but do tend to differ a little from the target flow rates based on experimental data. Pump flow rate variations are implemented by multiplying the volumetric flow rate of the base model by a factor. The factor is drawn from a normal distribution. Most pump manufacturers list their pump accuracy, and this can be directly used in the model.

Third, loading concentration variability is implemented by multiplying the concentration of all inlet units by a factor. The factor is drawn from a normal distribution.

Fourth, UV signal noise is the easiest error to deal with since it is random, independent, and normally distributed. It can be broken down into a detection limit noise and a proportional factor noise. This noise is also typically much higher frequency than the chromatogram and can be almost but not entirely automatically filtered out by CADET-Match. In most experiments this source of error can be neglected entirely. For experiments that have very low concentrations or very short pulses including this source of error can be important. UV signal noise error is applied directly to the resulting chromatogram(s) by multiplying the concentration at each time point in the base chromatogram by a normally distributed factor and adding a normally distributed noise.

As previously discussed, the calibrated model is used as a template and error simulations are created from this model to create the error model. The number of simulations used to create the error model is variable. More simulations don't necessarily improve the error model as discussed in section 3.1.3 due to smoothness of the KDE depending on the number of samples used to calibrate it. While smoothness is not something that is explicitly used as a target for the KDE a non-smooth KDE results in slower MCMC convergence. Based on experimentation 1000 simulations was found to be sufficient while it often took 50,000 simulations or more before the probability density function would become smooth again. From testing this made very little difference in the posterior but substantially added to CPU and RAM usage. To uniformly sample from the previously mentioned error sources a Sobol sequence is used. A Sobol sequence is a quasi-random low discrepancy sequence that is designed to uniformly span a unit-hypercube and is ideal for this usage case. It is defined by the number of samples and the number of dimensions. True random sampling creates clusters and gaps of points, and this causes the error model to slightly change from run to run and causes the posterior distribution to change from run to run which is neither robust nor consistent. The pump delay, pump flow rate and loading variability errors sources all have cumulative probability distributions that range from 0 to 1, the same as the Sobol sequence. A Sobol sequence is created with the number of samples equal to the number of simulations in the error model and the number of dimensions equal to the number of error sources. The cumulative probability distributions are used to convert the Sobol sequence to the desired errors sources. The resulting sequence of error sources is used to create a sequence of error simulations from the calibrated model that is then run, and UV noise is applied resulting in a series of chromatograms.

The chromatograms are then reduced to vectors that describe their important characteristics, as per section 3.1.2. The same goal function with the same metrics is used as for parameter estimation with CADET-Match except that two additional metrics are needed for MCMC. This reuse of the goal function from parameter estimation simplifies the process of uncertainty quantification and allows uncertainty quantification to be applied to any problem that parameter estimation works for. The absolute height and position of the highest peak are added to the other metrics already in the goal function. The parameter estimation metrics for height and position quantify relative differences that are minimized. Flow rate and pump delay errors have an asymmetric impact on the peak, and this must be captured by the error model. This impact can be seen in Figure 31 by varying the flow rate. An increase of 5% results in the peak max coming sooner and being higher while a decrease of 5% results in a lower peak max that comes later. While the absolute peak max and time change the relative difference from increasing or decreasing the flow rate by 5% is the same.

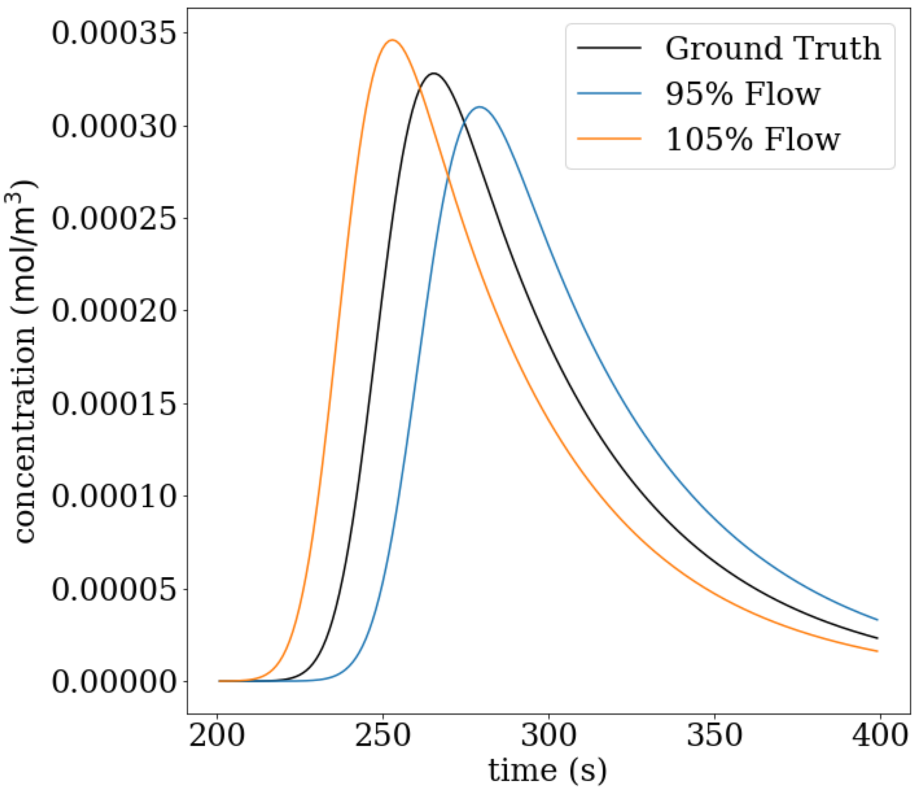

Figure 31: Example pulse along with 5% higher and lower flow rates illustrating asymmetric impact of flowrate changes.

The absolute time of the peak max cannot be used since it also quantizes the problem, the time of peak max is always a time on the simulated time grid. Quantizing the problem results in a non-smooth distribution of time-offsets which degrades the KDE and as a result degrades MCMC. The KDE degrades by requiring a small bandwidth to capture the non-smooth distribution of time-offsets which then creates many local minima in the KDE. These local minima degrade the performance of MCMC by making it less likely a small step will be accepted which degrades the entire process. A spline can be fit to the data to provide a smooth interpolation. Multiple spline libraries where tested and they were found to place a knot at the peak max which results in the time-offset still being quantized. After testing, using the time-offset for the first point to reach 90% of peak max resulted in a smooth time-offset, independent of the simulated grid time. The 90% number is mostly arbitrary. An alternative is to run the simulation on a finer time grid and while this would theoretically work it is not practically feasible. Most experimental systems sample at about 1Hz and to get the same accuracy as the spline method the sample rate had to exceed 1Mhz and would increase memory requirements by a factor of a million as a result.

The next part is to process the error matrix into a probability map, i.e., a function that given a new reduced vector can calculate the probability of that vector being explained by the sources of error in the error matrix. KDE is used to turn the error matrix into a probability map, as per section 3.1.3. The probability map is P(B|A) and can then be used for MCMC.

3.2.2. Bayesian Uncertainty Quantification

Now that the error model has been created, the next steps involve running MCMC on the error model. As mentioned previously, there is a dimensionality problem in creating the initial population. This problem can be solved by ensure all of the walkers start in high probability regions. The Pareto front from the parameter estimation is used as a seed for the initial population of walkers. If there are more entries on the Pareto front than walkers, then starting points are chosen equal to the number of walkers by randomly sampling from the Pareto front without replacement. If there are fewer entries on the Pareto front than the number of walkers, then all entries of the Pareto front are used and entries are randomly augmented with a small amount of noise to fill up the remaining slots. The noise is important because if there are two walkers with numerically identical locations and they are used together in a move the distance between them is zero which causes the code to fail. A small amount of noise with a mean of 1.0 and a standard deviation of 0.02 is multiplied by a randomly chosen entry from the Pareto front and used. The actual amount of noise is not important so long as it is large enough that it does not cause numerical issues and it should be kept above 1e-8 for that reason. The MCMC sampler is then run for 100 steps. All entries with less than 10% of the maximum seen probability are removed. A KMeans [73] clustering algorithm is used with the number of clusters equal to the number of walkers to find new starting points dispersed in the higher probability region. This procedure of running for 100 steps, removing and clustering is repeated until the relative change in the highest probability found is less than 1% to ensure all the walkers start off dispersed in high probably space and based on advice from the developer of emcee [74].

After the walkers have been moved to the high probability region the next step is to adjust the boundaries of the MCMC process around the high probability region. The core problem is that the step size for a walker needs to be appropriate to all dimensions. The ensemble of walkers working together can mitigate this to some extent but not completely and not in all situations, such as when non-linear variable transforms are used. Parameter estimation is normally performed before MCMC and the same boundaries for parameter estimation are used for MCMC. Parameter estimation performs searches with search bounds spanning multiple orders of magnitude and log-transforms used to allow more efficient searching. Parameters, once found, normally take up a tiny slice of the searched region. A solution to this problem is to adjust the boundaries of the MCMC process so that it contains the high probability region and for the high probability region to have approximately the same step size in every dimension. MCMC sampling is performed until the 5% and 95% percentile values in the last 200 steps have a standard deviation of less than 1e-3. The 5% and 95% values were chosen because they are far enough from the center of the distribution to give a good indication of the final width of the distribution. The boundaries are adjusted for each variable by setting the upper bound of the distribution to three times the difference from the 95% bound to the center and the lower bound to three times the different from the 5% bound to the center. This centers the high probability region inside the new boundaries. The high probability region takes up approximately a third of the region between the boundaries in each dimension. This fixes the step size problem and provides the space needed to define the posterior more precisely.

After the MCMC boundaries have been adjusted the process is repeated for locating the walkers in high probability space. The final state of the walkers from the boundary adjustment step is used as the starting state of the walkers for this step. With the change of the search boundaries, areas that may have been difficult to explore in the original area become more accessible and improved starting positions can be found.

The final MCMC step is a combination of burn-in and running the sampler until convergence. CADET-Match by default uses 52 times the IAT as a stopping criterion and discards the first 2 IAT as burn-in. This ensures the samples used are independent of the starting point and sufficient as a stopping criterion while also being fast enough to calculate. With 128 walkers 50 times IAT results in a minimum of 6400 independent samples per parameter while the actual chain length and total number of samples will vary based on how correlated the samples are. Using the MCSE and finding the max error of the posterior could also be used but it takes much longer to calculate to the point that progress either needs to be checked infrequently or the determination of progress dominates the calculation time compared to the time it takes to run the simulations.

The final step of the entire process is postprocessing all the data. The chain is processed with KDE to create a probability density function suitable for use as a prior in another MCMC run following the same procedures that were used to turn the error matrix into a probability density function. The maximum a posteriori (MAP), the point corresponding to the highest posterior probability, is trivial to find once MCMC has completed. With the log probability of each accepted entry in every chain recorded it is a simple search to find the entry with the highest log probability. It is important to note that the parameters are not independent of each other, and the MAP is not the highest probability point for each parameter but is the parameter set that has the highest overall probability.

3.3. Case Studies

All synthetic and experimental case studies share several fixed parameters shown in Table 27 and Table 28. All software versions can be found in Table 29. The synthetic and experimental case studies also share some of the same auxiliary models used to describe the impact of tubing and mixing devices. A dispersive plug flow reactor (DPFR) is used to represent tubing. A continuously stirred tank reactor (CSTR) is used to represent mixing valves and other areas in the tubing that generate mixing like behavior. The General Rate Model (GRM) is used to represent the column. Synthetic and experimental case studies use the same problem discretization and other solver settings detailed in Table 27. The DPFR, CSTR, and GRM are connected in a single simulation using CADET. The goal is to find the distribution that corresponds to each parameter such that, to the extent possible, unaccounted errors in one stage are not carried over to further stages. For example, a stage that does not accurately account for the error in the tubing dispersion will have any residual error carried over into the axial dispersion of the column. This makes it important to design the base model and chosen error sources to minimize unaccounted errors.

Table 27: Common parameters to all simulations.

Parameter	*Value*	*Unit*
Flow rate	8.3e-8	m^3/s
Tube length	0.146	m
Column length	0.25	m
Column cross sectional area	2.01e-4	m^2
Particle radius	4.5e-5	m
Column discretization	150	
Particle discretization	15	
Absolute tolerance	1.0e-8	
Relative tolerance	0.0	

Table 28: Synthetic specific parameters.

Parameter	*Value*	*Unit*
Column dispersion	3.0e-7	m^2/s
Column porosity	0.35	
Film diffusion	2.0e-6	m^2/s
Particle diffusion	2.5e-11	m^2/s
Particle porosity	0.33	
Linear kA	4.0e-4	
Linear kD	4.0e-3	
Tube dispersion	2.7e-6	
Tube cross sectional area	1.9E-5	m^2
CSTR volume	4.0e-6	m^3

Table 29: Software versions used for uncertainty quantification.

Software	*Version*	*Source*
CADET	4.2.0	https://cadet.github.io
CADET-Match	0.8.11	https://github.com/modsim/CADET-Match
CADET-Python	0.11	https://github.com/modsim/CADET-Python
Python	3.7.9	https://www.python.org
Joblib	1.0.0	https://joblib.readthedocs.io
addict	2.2.1	https://github.com/mewwts/addict
corner	2.2.1	https://corner.readthedocs.io
emcee	3.0.2	https://emcee.readthedocs.io
SALib	1.3.8	https://salib.readthedocs.io
pymoo	0.4.2.2	https://pymoo.org/algorithms/index.html
psutil	5.7.2	https://psutil.readthedocs.io
NumPy	1.19.2	https://numpy.org
OpenPyXL	3.0.6	https://openpyxl.readthedocs.io
SciPy	1.5.1	https://www.scipy.org
Matplotlib	3.3.2	https://matplotlib.org
pandas	1.2.1	https://pandas.pydata.org
H5py	2.10.0	https://www.h5py.org
Seaborn	0.10.1	https://seaborn.pydata.org
scikit-learn	0.23.1	https://scikit-learn.org
Jstyleson	0.0.2	https://github.com/linjackson78/jstyleson

Two error models are used for the synthetic and experimental case studies and shown in Table 30 called narrow and wide, respectively. A normal distribution is denoted with an N (mean, standard deviation) and a uniform distribution with U(lower bound, upper bound). The narrow error source is chosen to have a small amount of error while the wide error source is chosen to have error close to what we have typically observed in industrial data. An HDI of 90% is used for reporting upper and lower bounds and the maximum MCSE is reported for this interval. Corner plots and HDI plots for every study can be found in the supplement. A corner plot is a matrix of plots where the diagonal plots contain the distribution for each parameter and below the diagonal contains the joint distribution for each combination of two parameters. These plots provide a quick way to see correlations between parameters and see all the distributions in a single plot.

Table 30: Error source description for Narrow and Wide error sources.

Name	*Narrow*	*Wide*
Pump Delays	-	$U(0,2)$
Flow Rate	$N(1.0,2.5e-5)$	$N(1.0,4.225e-5)$
Load Concentration	$N(1.0,2.5e-5)$	$N(1.0,8.1e-5)$
UV Noise	$N(1.0,1.0e-6)$	$N(1.0,1.0e-6)$

3.3.1. Synthetic Case Studies

The synthetic model is designed to be representative of a real system but with known parameters to verify the proper functioning of the uncertainty quantification process. The synthetic model uses a DPFR connected to a CSTR connected to a column described by the GRM. All parameters can be found in the supplement.

A stagewise estimation is performed with the posterior from one stage used as a prior in the next stage. Four stages are used in the synthetic model with a single component. The four stages are bypass pulse, non-pore penetrating pulse, pore-penetrating but non-binding pulse, and binding pulse. For the bypass pulse, the column is replaced with a zero-volume connector and removed from the simulation. This stage is designed to account for extra column effects. In the next stage, a non-pore penetrating pulse is used to find the column porosity and dispersion. Then, a non-binding but pore-penetrating pulse is used to find the film diffusion, particle diffusion, and particle porosity. Finally, a binding pulse is used with a linear isotherm to find the adsorption and desorption rates. Instead of a single optimal value for each parameter carried over from one stage to the next the parameter distribution is carried over. This means with each stage there are more parameters to estimate.

3.3.1.1. Dextran Pulse with Detached Column

The first stage is a bypass experiment designed to identify the posterior distributions for the tubing dispersion, cross-sectional area, and mixing volume. From the narrow error source shown in Table 31 the MAP is close to the ground truth with narrow bounds on all parameters. The “% MAP” column shows width of the HDI as a percentage of the MAP which makes it easy to see how wide the HDI is for parameters that vary over orders of magnitude. The max(%MCSE) for all parameters indicates a small average error of about 0.06% over the HDI interval. Figure 32 is a corner plot and the diagonals show the posterior distribution for each parameter independently and the intersecting plots show the joint probability distributions. Any of the intersecting plots that is approximately circular indicates the two parameters are independent, ellipsoidal shapes indicate linear correlations, and any other shapes, such as a banana shape, indicate non-linear correlations. From the corner plot in Figure 32 all the parameters have an approximately normal probability distribution with tube dispersion and CSTR volume showing a linear correlation and the other parameters are uncorrelated with each other. From the wide error source shown in Table 32 the impact of the error source can be clearly seen with wider bounds, as seen in % MAP, while the MAP remains close to the ground truth. The corner plot in Figure 33 shows an approximately normal distribution for all parameters and a strong linear relationship between the cross-section area and dispersion with a weak linear relationship between dispersion and CSTR volume. The differences in the relationship between the variables is mostly due to an approximately 1s delay in the error model which illustrates why it is so important to capture these delays in the model instead of having them as errors. From Table 33 both the narrow and wide error sources are very similar with runtimes that only differ by a few seconds and almost the same acceptance rates and IAT.

Table 31: Bypass Narrow.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Tube Dispersion	2.70e-06	2.51e-06	2.71e-06	2.88e-06	14.0	0.13
Tube Cross Section Area	1.90e-05	1.89e-05	1.90e-05	1.91e-05	1.3	0.01
CSTR Volume	4.00e-06	3.94e-06	4.00e-06	4.08e-06	3.5	0.03

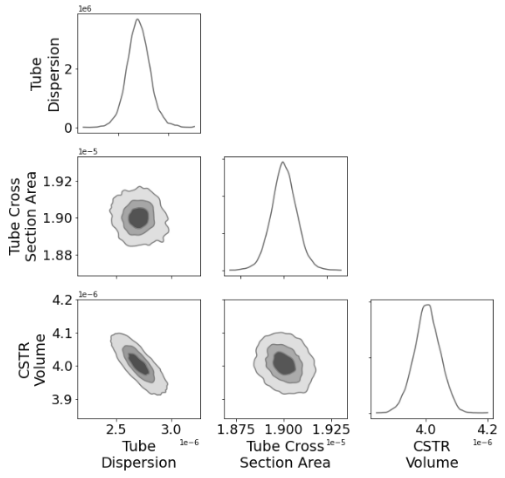

Figure 32: Synthetic Dextran bypass experiment narrow error source corner plot.

Table 32: Bypass Wide.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Tube Dispersion	2.70e-06	2.30e-06	2.70e-06	3.17e-06	32.1	0.34
Tube Cross Section Area	1.90e-05	1.82e-05	1.90e-05	1.97e-05	8.0	0.06
CSTR Volume	4.00e-06	3.91e-06	4.01e-06	4.11e-06	4.9	0.04

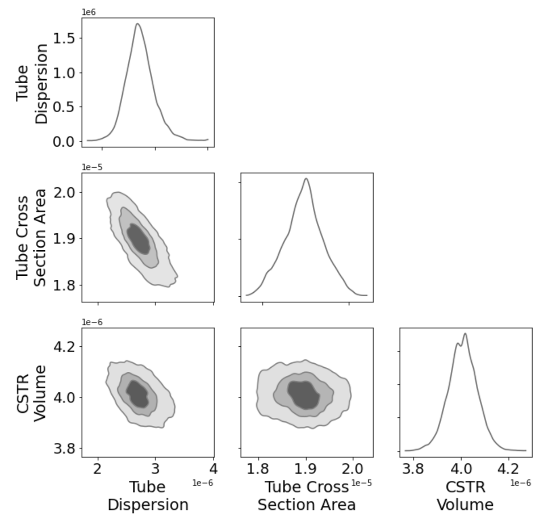

Figure 33: Synthetic Dextran bypass experiment wide error source corner plot.

Table 33: Bypass Experiment Narrow vs Wide metrics.

Name	*Narrow*	*Wide*
Chain Simulations	73,728	67,584
Chain Length	576	528
Simulation Time	0:26:15	0:25:36
max(IAT)	11	10
Acceptance Ratio	0.339	0.354

3.3.1.2. Non-Pore Penetrating Pulse

The second stage is a non-pore penetrating pulse designed to identify the posterior distributions of the column dispersion and column porosity in addition to the variables carried over in the prior. From the narrow error source shown in Table 34 the MAP is close to the ground truth for all parameters with a narrow HDI interval. The corner plot, shown in Figure 34, shows the same correlations as in the previous step along with new weak linear relationships between column dispersion and column porosity with all other variables and a more complex interaction between column dispersion and column porosity. In

Table 35 the MAP is also close to the ground truth for all parameters, but the HDI interval is wider, and the corresponding max(%MCSE) are larger. Figure 35 also shows the same kind of progression in parameter correlations from the previous stage with column dispersion and column porosity having a weak linear relationship to most of the other variables and a linear relationship with each other. From Table 36 both the narrow and wide error sources are similar with the narrow error source having a slightly higher IAT and taking a little less time to run. The wide error source takes about 30% more time to run, this is mostly due to the slightly longer time it takes to run the wide error source simulations and other tasks running on the server and is not significant. Something that really stands out is the increased time, it is about 6x to 8x slower than the bypass pulse was. The bypass pulse had only 3 parameters to work with while the non-pore penetrating pulse has 5 parameters which is where the problem of dimensionality comes in. The result is a larger IAT, lower acceptance rate and a longer total runtime before convergence.

Table 34: Non-Pore Penetrating Narrow.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Column Dispersion	3.00e-07	2.92e-07	3.00e-07	3.10e-07	6.0	0.07
Column Porosity	0.350	0.347	0.350	0.352	1.5	0.01
Tube Dispersion	2.70e-06	2.55e-06	2.71e-06	2.86e-06	11.2	0.10
Tube Cross Section Area	1.90e-05	1.89e-05	1.90e-05	1.91e-05	1.2	0.01
CSTR Volume	4.00e-06	3.95e-06	4.00e-06	4.05e-06	2.6	0.02

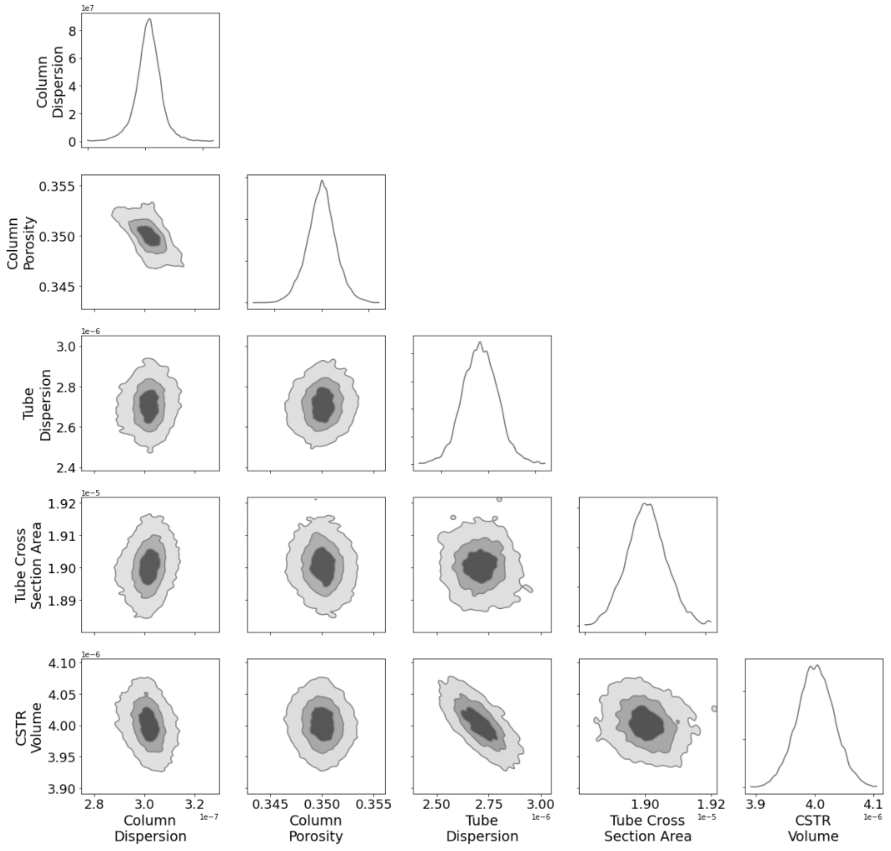

Figure 34: Synthetic Dextran experiment with column narrow error source corner plot.

Table 35: Non-Pore Penetrating Wide.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Column Dispersion	3.00e-07	2.87e-07	3.02e-07	3.16e-07	9.6	0.08
Column Porosity	0.350	0.346	0.350	0.355	2.5	0.02
Tube Dispersion	2.70e-06	2.32e-06	2.73e-06	3.08e-06	27.6	0.26
Tube Cross Section Area	1.90e-05	1.82e-05	1.90e-05	1.96e-05	7.3	0.08
CSTR Volume	4.00e-06	3.93e-06	4.01e-06	4.08e-06	3.8	0.03

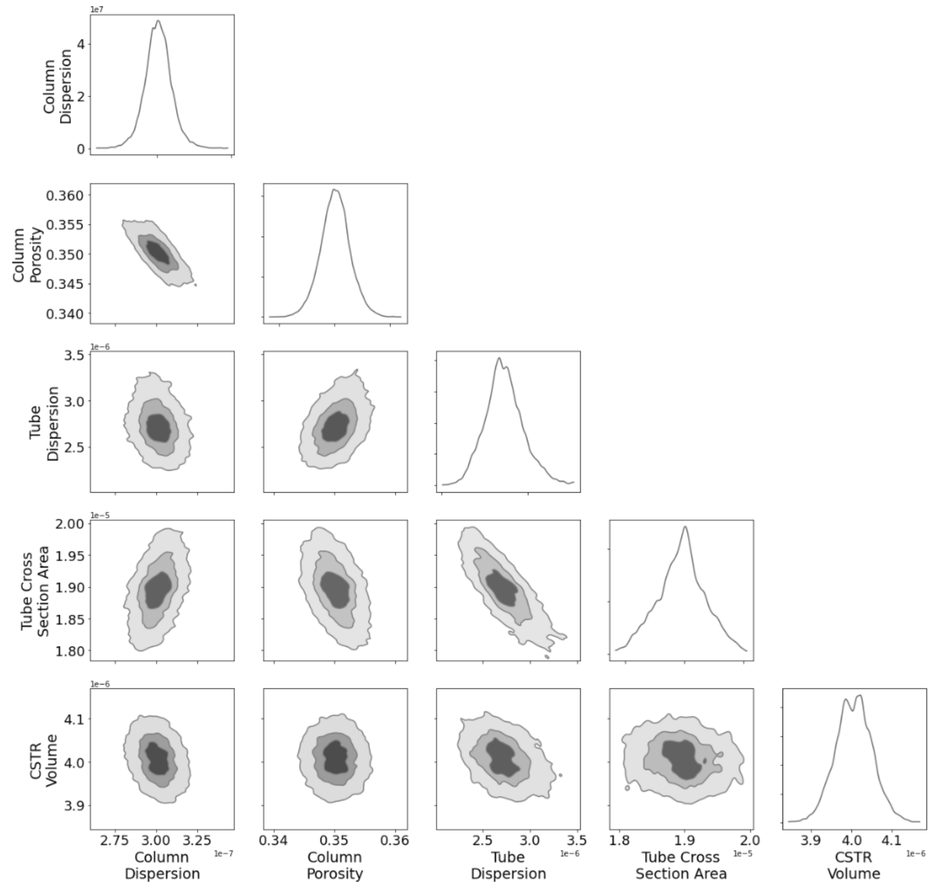

Figure 35: Synthetic Dextran experiment with column wide error source corner plot.

Table 36: Non-Pore Penetrating Experiment Narrow vs Wide metrics.

Name	*Narrow*	*Wide*
Chain Simulations	224,640	196,864
Chain Length	1755	1538
Simulation Time	3:02:39	4:13:10
max(IAT)	34	30
Acceptance Ratio	0.197	0.205

3.3.1.3. Non-Binding Protein Pulse

The third stage is a pore-penetrating but non-binding protein pulse to find the posterior distributions for film diffusion, pore diffusion, and particle porosity in addition to the variables carried over in the prior. This brings the system up to 8 variables and from Table 39 it is clear the IAT is growing, and the acceptance rate is continuing to decrease while the number of simulations required has grown to over a million. Both error sources converged at a similar rate this time with a similar IAT. From Table 37 and Table 38 the pattern of previous stages is repeated where the MAP is close to the ground truth and the HDI interval is narrower for the narrow error source than for the wide error source. The driving force for the large jump in IAT and total runtime is a combination of more variables and the coupling between variables. Figure 36 shows the posterior distribution for film diffusion and particle diffusion along with their joint probability distribution in the center. While film diffusion and particle diffusion are each approximately normal their joint distribution shows a strong correlation between these parameters. Both film diffusion and pore diffusion also have fairly flat peaks indicating small changes near the optimum have a small effect on the simulated chromatogram. Figure 37 and Figure 38 shows strong correlation between parameters. Film diffusion, particle diffusion, and particle porosity have a non-linear correlation which helps explain why these parameters are so difficult to estimate. The non-linear correlation between film diffusion and particle diffusion originally degraded MCMC progress and lead to a large IAT until a variable transform, chapter 2.1, was created to address this issue by using the reciprocal.

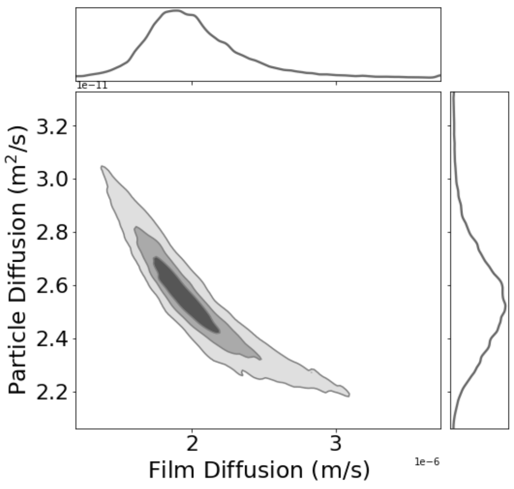

Figure 36: Film diffusion vs pore diffusion posterior distribution plot for the narrow error source.

Table 37: Non-Binding Narrow.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Film Diffusion	2.00e-06	1.48e-06	2.05e-06	2.57e-06	53.7	0.79
Particle Porosity	0.330	0.325	0.331	0.336	3.1	0.04
Particle Diffusion	2.50e-11	2.22e-11	2.54e-11	2.82e-11	23.5	0.18
Column Dispersion	3.00e-07	2.93e-07	3.03e-07	3.09e-07	5.2	0.06
Column Porosity	0.350	0.348	0.349	0.352	1.3	0.01
Tube Dispersion	2.70e-06	2.57e-06	2.68e-06	2.84e-06	10.2	0.09
Tube Cross Section Area	1.90e-05	1.89e-05	1.91e-05	1.91e-05	1.1	0.01
CSTR Volume	4.00e-06	3.95e-06	4.03e-06	4.05e-06	2.4	0.02

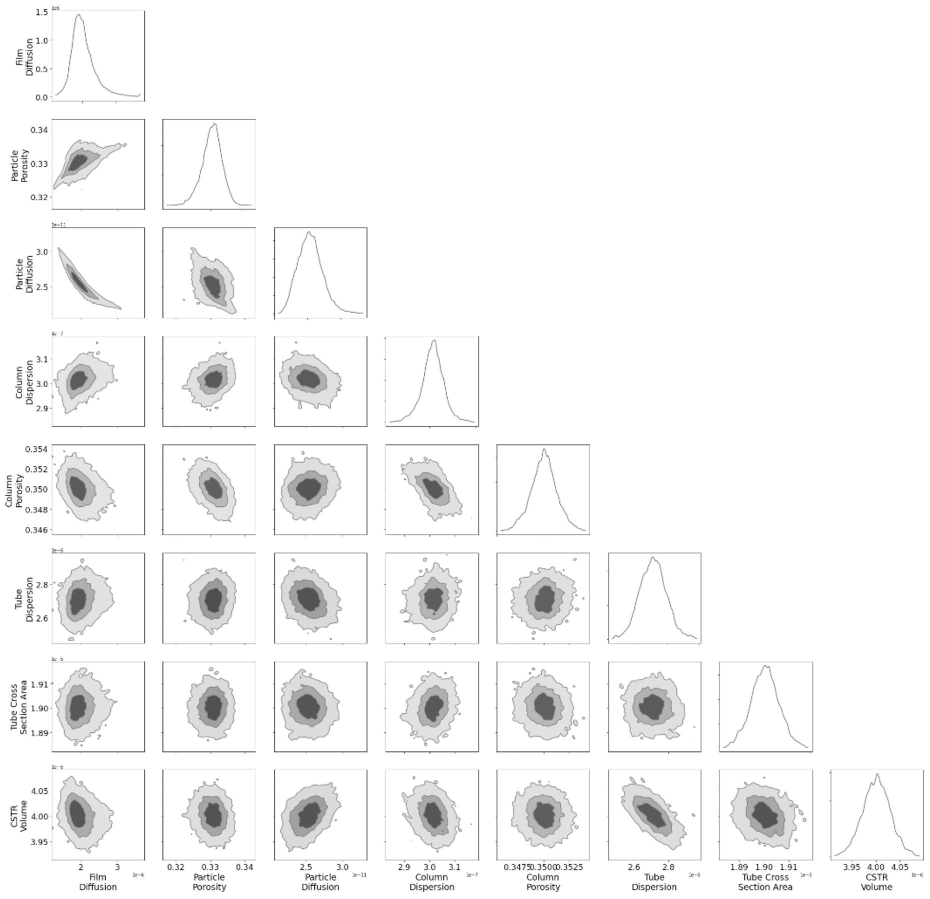

Figure 37: Synthetic Non-Binding protein experiment narrow error source corner plot.

Table 38: Non-Binding Wide.

Name	*GT*	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Film Diffusion	2.00e-06	1.30e-06	1.98e-06	2.67e-06	69.5	0.70
Particle Porosity	0.330	0.322	0.329	0.338	4.7	0.04
Particle Diffusion	2.50e-11	2.20e-11	2.52e-11	2.97e-11	30.5	0.26
Column Dispersion	3.00e-07	2.87e-07	2.93e-07	3.14e-07	9.1	0.10
Column Porosity	0.350	0.346	0.352	0.354	2.3	0.02
Tube Dispersion	2.70e-06	2.38e-06	2.75e-06	3.04e-06	23.7	0.29
Tube Cross Section Area	1.90e-05	1.83e-05	1.88e-05	1.96e-05	6.7	0.09
CSTR Volume	4.00e-06	3.94e-06	4.03e-06	4.07e-06	3.3	0.03

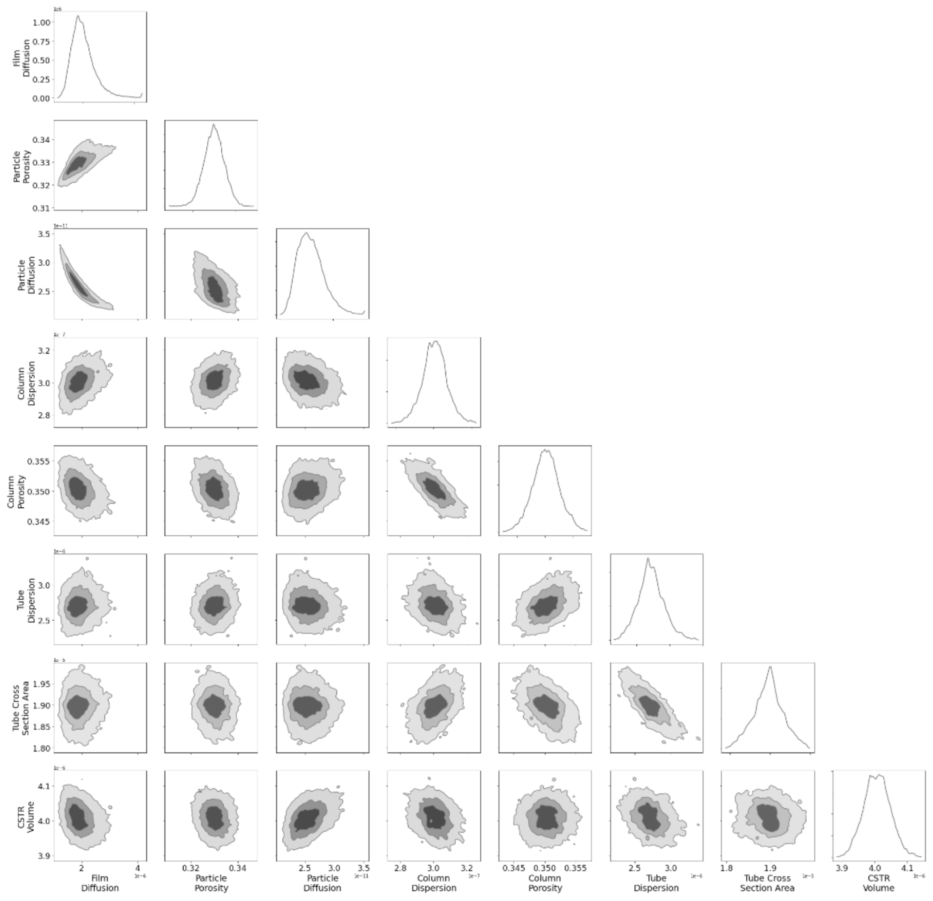

Figure 38: Synthetic Non-Binding protein experiment wide error source corner plot.

Table 39: Non-Binding compare.

Name	*Narrow*	*Wide*
Chain Simulations	1,141,376	981,376
Chain Length	8917	7667
Simulation Time	16:01:42	14:15:51
max(IAT)	178	153
Acceptance Ratio	0.092	0.095

3.3.1.4. Binding Protein Pulse

The final stage uses a pore-penetrating binding protein pulse to find the posterior distributions for the adsorption and desorption rates of a linear binding model in addition to the variables carried over in the prior. Binding is described with a linear binding model for computational reasons. The linear simulation used for this example takes about 1.9 s to calculate while a simple 2 component SMA simulation takes about 75 s and three simulations are typically required for uniqueness which raises the runtime to 225s per parameter evaluated which would raise the runtime to about 2 years and after factoring the scaling due to additional parameters it would take approximately 14 years. This is a topic that will be further addressed in the conclusions. As can be seen in Table 42 the IAT is about 3x to 6x higher compared to the pore-penetrating, but non-binding case. The linear binding simulation takes about 1.9 seconds while the pore-penetrating, but non-binding simulation took about 0.8 seconds. Between longer simulation times and more simulations required the time required for the calculation has grown from 16 hours to about 6 days for the wide error source and 2 days for the narrow error source. Table 40 and Table 41 show the same pattern previously seen where the narrow error source has a narrower HDI interval than the wide error source. The MAP is close to the ground truth for both error sources for all parameters after four stages. Looking into a subset of the posterior distributions from the corner plot in Figure 39, with Figure 78 in the appendix showing the full data set, shows some interesting results. From the joint-distribution plots particle porosity and film diffusion, particle porosity and column porosity, and kA and keq all have weak linear relationships. There is a strong linear relationship between film diffusion and particle diffusion. From the rest of the plots while it is clear that most of the parameters are coupled to various degrees the coupling is not strong. Looking at the same entries in Figure 40, with Figure 79 in the appendix showing the full dataset, the results are quite similar kA and keq have a stronger linear correlation. Film diffusion and particle diffusion also have a strong correlation, but it has become more non-linear. Particle porosity and film diffusion show a similar change with the correlation becoming more non-linear. An important thing to remember is that that the error models are very similar to each other and most of the difference comes down to an approximately 1s delay in pump activation. This helps illustrate the importance of accounting for errors as accurately as possible in early stages of an estimation and the importance of controlling experimental errors because even small errors substantially increase the uncertainty in parameters.

After all these stages, using the posterior of one stage as the prior of the next stage, the results can be seen in Figure 41. The top row shows a visualization of the error model, P(B|A), with the narrow error source shown in A and the wide error source shown in B. The graphs are shaded based on the relative probability of the chromatogram being in that region based on the errors used. The main difference between A and B is that B includes a 0-2s pump delay and a slightly larger error on the flow rate. The impact of these small changes can clearly be seen with the B having not only a much wider probability region, but it is also more asymmetric. Moving down to the bottom row is a visualization of the chromatograms generated by sampling from the posterior distribution with the narrow error source shown in C and the wide error source shown in D and a black line drawn to indicate the ground truth. The posterior is not expected to match the shape or width of the error model. Not all errors generated by pump delays, concentration changes, or flow rate changes can be explained by model parameters which is what causes the expected difference. Looking at C, the 90% HDI confidence tube on the prediction are extremely tight. This indicates that despite how wide the HDI range is for the narrow error source in Table 40 that parameter changes within the HDI range are essentially negligible. Moving on to D while bounds on the prediction are wider than the narrow error source, they are still quite narrow overall. Looking at Table 41 again in context it is interesting to note that while the HDI region for all parameters are much wider they still managed to match the data quite precisely. In both C and D, the ground truth goes through the high probability region as can be seen in the insets.

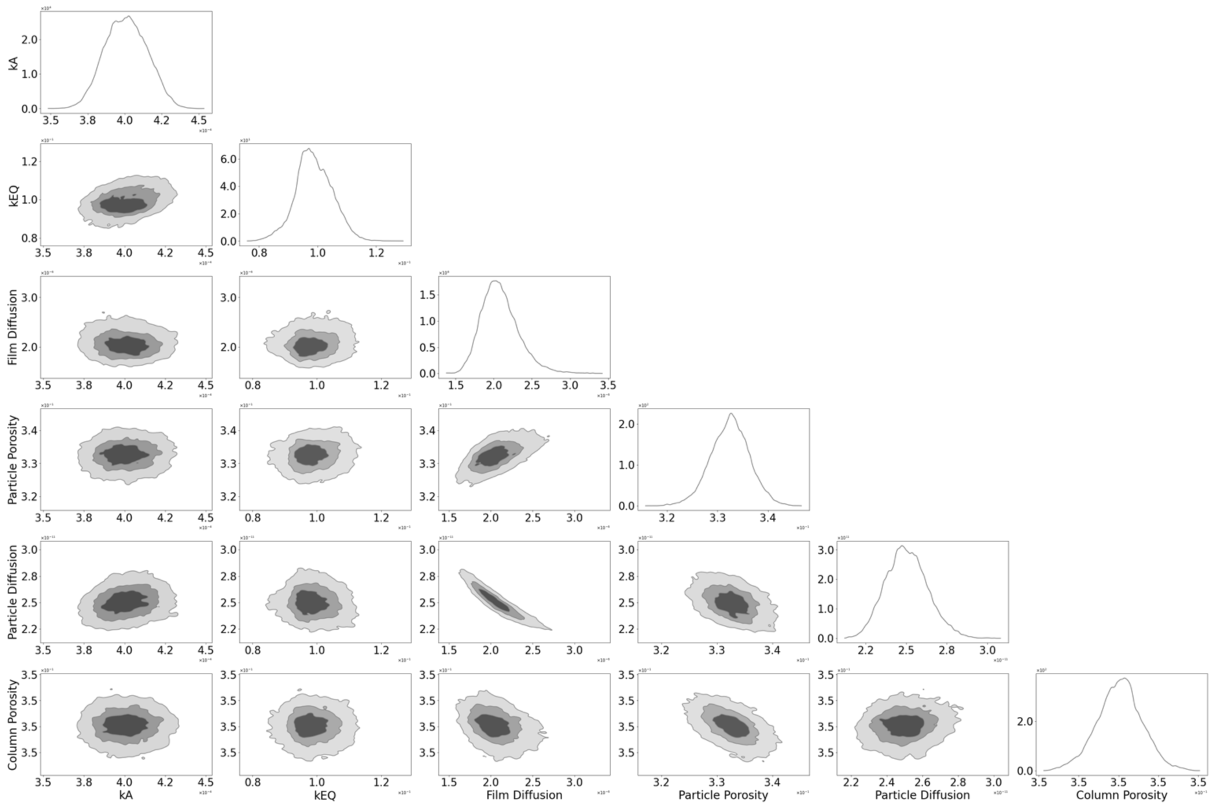

Figure 39: Narrow error source binding pulse posterior plot subset.

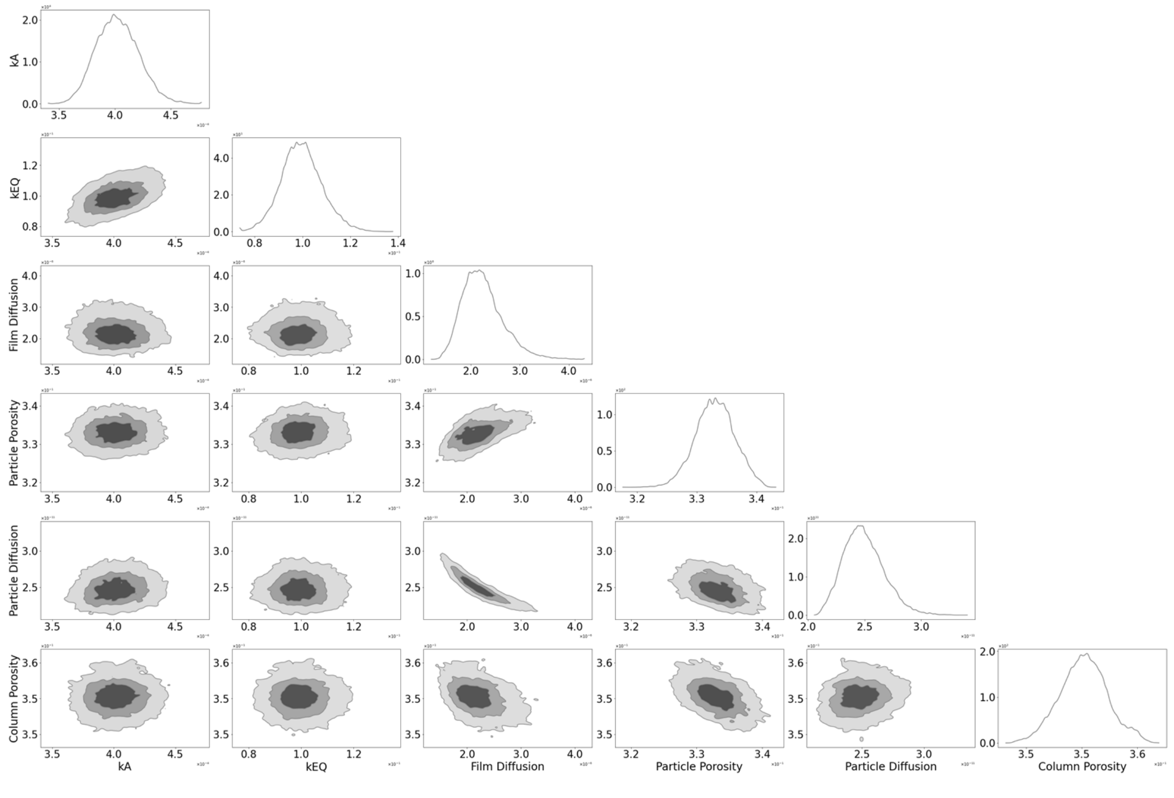

Figure 40: Wide error source binding pulse posterior plot subset.

Table 40: Linear Narrow.

Name	GT	LB HDI	MAP	UB HDI	% MAP	max(%MCSE)
kA	4.00e-04	3.78e-04	3.97e-04	4.24e-04	11.6	0.11
kEQ	1.00e-01	8.94e-02	9.42e-02	1.10e-01	21.9	0.18
Film Diffusion	2.00e-06	1.70e-06	2.03e-06	2.47e-06	38.3	0.33
Particle Porosity	0.330	0.328	0.332	0.334	1.9	0.02
Particle Diffusion	2.50e-11	2.29e-11	2.55e-11	2.71e-11	16.6	0.13
Column Dispersion	3.00e-07	2.94e-07	3.03e-07	3.08e-07	4.7	0.09
Column Porosity	0.350	0.348	0.349	0.352	1.1	0.01
Tube Dispersion	2.70e-06	2.58e-06	2.68e-06	2.83e-06	9.4	0.11
Tube Cross Section Area	1.90e-05	1.89e-05	1.91e-05	1.91e-05	1.0	0.02
CSTR Volume	4.00e-06	3.96e-06	4.03e-06	4.04e-06	2.1	0.03

Table 41: Linear Wide.

Name	GT	LB HDI	MAP	UB HDI	% MAP	max(%MCSE)
kA	4.00e-04	3.71e-04	4.21e-04	4.33e-04	14.6	0.11
kEQ	1.00e-01	8.52e-02	1.02e-01	1.14e-01	28.0	0.22
Film Diffusion	2.00e-06	1.60e-06	1.75e-06	2.92e-06	74.9	0.38
Particle Porosity	0.330	0.328	0.335	0.338	3.2	0.03
Particle Diffusion	2.50e-11	2.20e-11	2.79e-11	2.78e-11	20.7	0.14
Column Dispersion	3.00e-07	2.89e-07	2.85e-07	3.13e-07	8.6	0.08
Column Porosity	0.350	0.346	0.352	0.354	2.0	0.03
Tube Dispersion	2.70e-06	2.41e-06	2.82e-06	3.02e-06	21.6	0.34
Tube Cross Section Area	1.90e-05	1.84e-05	1.91e-05	1.95e-05	6.0	0.09
CSTR Volume	4.00e-06	3.95e-06	4.00e-06	4.07e-06	3.0	0.03

Table 42: Linear binding results.

Name	Narrow	Wide
Chain Simulations	1,550,848	3,824,512
Chain Length	12116	29879
Simulation Time	1 day, 22:20:25	6 days, 9:05:02
max(IAT)	241	597
Acceptance Ratio	0.092	0.066

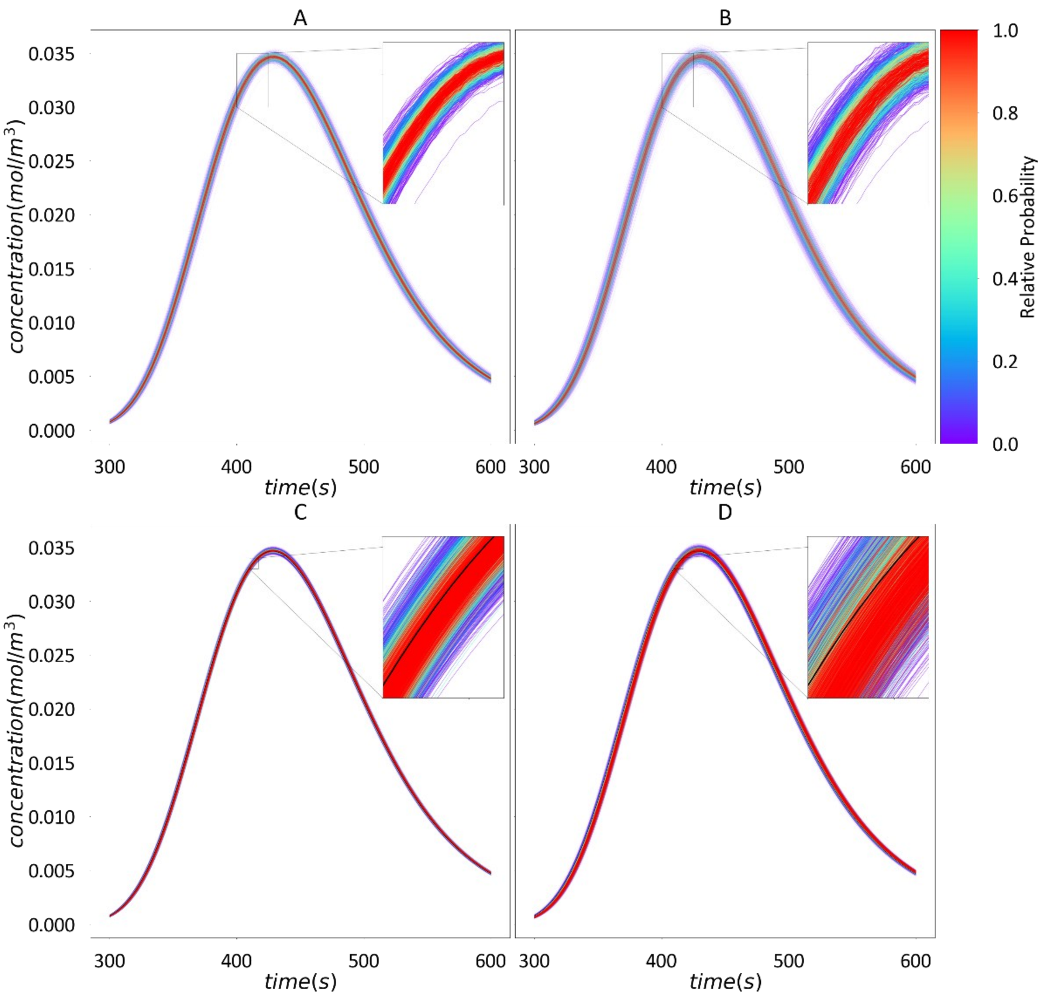

Figure 41: Plots of the error model and posterior chromatograms colored by relative probability A: Narrow Error Model B: Wide Error Model C: Narrow Posterior with ground truth shown in black D: Wide Posterior with ground truth shown in black.

3.3.2. Experimental Case Studies

An experimental dataset for two charge variants of a monoclonal antibody has been measured at Amgen. Available are two dextran pulses with detached column, two dextran pulses with attached column, two protein pulses under non-binding conditions, a gradient elution with fractionation and a gradient elution with extended loading phase but without fractionation. For computational reasons only the dextran pulses and non-binding protein pulses are used. As alluded to earlier a two-component competitive binding experiment is beyond currently available computing resources. As in the synthetic example the posterior from one stage is used as a prior in the next stage. The same two error sources are used from the synthetic case. The experimental setup consists of a DPFR connected to a column implemented with the GRM. The experimental setup presented here does not use a CSTR. More complex models with one or more CSTR and DPFR units were unable to better reproduce the observed behavior but suffered from high parameter correlations and wider HDI ranges (data not shown). Hence, the external holdup volumes are lumped, and the DPFR model parameters are not meant to reflect the real dimensions of the tubing.

3.3.2.1. Dextran Pulses with Detached Column

Like the synthetic case, the first stage is designed to identify the tubing dispersion and cross-sectional area by replacing the column with a zero-volume connector and using a Dextran pulse. The tubing length has been fixed for this experiment due to being highly correlated with the dispersion. So long as the volume is the same the tube can be short and wide or long and thin and the dispersion can be adjusted to compensate.

As can be seen in Table 43 the narrow error source has an extremely narrow HDI and a low maximum error in the interval. The corner plot in Figure 42 shows that dispersion and cross section area are uncorrelated but also shows that experimental data is more complex and the joint probability distributions are no longer as clean as they were in the synthetic cases. In Table 44 the wide error source shows a similar value for the MAP for both parameters to the narrow error source but wider HDI. The corner plot in Figure 43 shows a similar strong linear correlation between the cross-section area and the dispersion as seen in the synthetic example with the wide error source. Most of this strong correlation is caused by an approximately 1s pump delay. This is interesting because it means that not accounting for the delay in the model and instead having it as an error changes the correlation between the parameters and causes a large increase in the uncertainty of both parameters. This uncertainty will propagate through to other parameters. With experimental data the distributions have also gotten more complex with various wiggles in them. This is not because the chains have not converged yet and running them longer has very little impact on these wiggles. They are the result of complex interactions between the various layers of software that make the error model work. The data smoothing is not perfect and this cascades through the various software layers such as the score system and into the KDE for the error model. A number of methods have been looked at, as part of this thesis, to mitigate this issue, but all have come at a large increase in computational cost but little overall change to the probability distribution or the HDI interval and MAP. As with the synthetic examples early stages with few parameters have a much lower IAT and higher acceptance rates as seen in Table 45.

Table 43: Bypass Narrow.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Tube Dispersion	2.71e-06	2.77e-06	2.80e-06	3.3	0.02
Tube Cross Section Area	1.86e-05	1.87e-05	1.89e-05	1.2	0.02

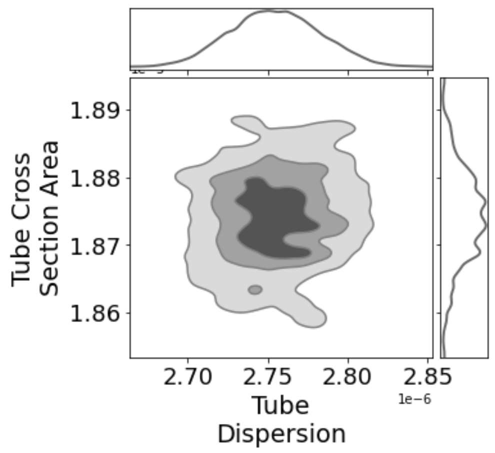

Figure 42: Experimental Dextran bypass experiment narrow error source corner plot.

Table 44: Bypass Wide.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Tube Dispersion	2.47e-06	2.72e-06	3.12e-06	23.8	0.22
Tube Cross Section Area	1.79e-05	1.87e-05	1.94e-05	8.2	0.11

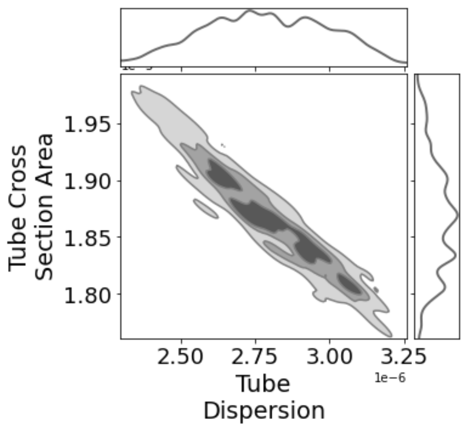

Figure 43: Experimental Dextran bypass experiment wide error source corner plot.

Table 45: Bypass Narrow vs Wide.

Name	*Narrow*	*Wide*
Chain Simulations	52,352	58,240
Chain Length	409	455
Simulation Time	0:31:40	0:27:46
max(IAT)	7	9
Acceptance Ratio	0.345	0.296

3.3.2.2. Dextran Pulses with Attached Column

In the second stage a dextran pulse with attached column is used to estimate the posterior distributions for the column porosity and column dispersion in addition to the variables carried over in the prior. From the narrow error source shown in Table 46 the upper and lower bounds of the HDI are quite narrow and a low max(%MCSE) in the interval. From Figure 44 tube dispersion and column porosity have a weak linear relationship with all parameters and approximately normal distribution. All of this is expected because in the math these parameters are related. Dispersion in the tubing and dispersion in the column are linearly related and it is only by accounting for the uncertainty in one stage that it does not propagate to the next stage. That was mostly successful and only weak linear relationships are seen. In Table 47 with the wide error source the upper and lower bounds of the HDI are much wider and with a higher max(%MCSE) in the interval. Column dispersion and column porosity show a strong linear relationship to all parameters in Figure 45. The strong linear relationship from the previous stage has propagated through to this stage and impacted the uncertainty of all the parameters and is magnified by the pump delay in this stage also. The MAPs are quite close to each other with the wide error source just having a wider interval than the narrow error source. Both models took about the same amount of time to converge with the wide error source taking about 37% longer to converge and the IAT increasing compared to the previous stage.

Table 46: Dextran Narrow.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*

Column Dispersion	5.08e-07	5.17e-07	5.28e-07	3.8	0.03
Column Porosity	0.332	0.337	0.338	1.6	0.01
Tube Dispersion	2.71e-06	2.76e-06	2.80e-06	3.1	0.02
Tube Cross Section Area	1.87e-05	1.87e-05	1.89e-05	1.1	0.01

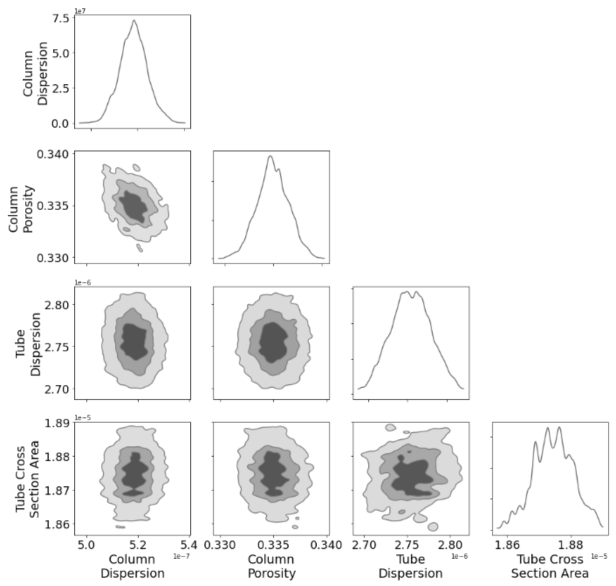

Figure 44: Experimental Dextran experiment with column narrow error source corner plot.

Table 47: Dextran Wide.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Column Dispersion	5.00e-07	5.27e-07	5.38e-07	7.1	0.07
Column Porosity	0.330	0.334	0.339	2.7	0.02
Tube Dispersion	2.46e-06	2.63e-06	3.09e-06	23.8	0.25
Tube Cross Section Area	1.79e-05	1.91e-05	1.94e-05	7.5	0.10

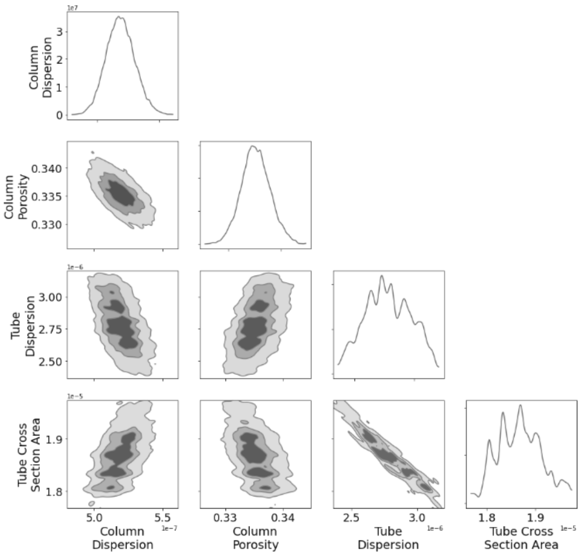

Figure 45: Experimental Dextran experiment with column wide error source corner plot.

Table 48: Dextran Narrow vs Wide Metrics.

Name	*Narrow*	*Wide*

Chain Simulations	181,376	264,448
Chain Length	1417	2066
Simulation Time	6:21:52	8:45:33
max(IAT)	28	41
Acceptance Ratio	0.171	0.129

3.3.2.3. Non-Binding Protein Pulses

In the third stage a non-binding protein pulse is used to estimate the posterior distributions for the particle porosity, film diffusion, and pore diffusion in addition to the variables carried over in the prior. Using the target protein instead of salt as non-binding but pore penetrating tracer is more reliable, as salt diffuses faster and has higher pore accessibility. As in the previous stages the narrow error source, source shown in Table 49, has a narrow upper and lower bound on the HDI except for film diffusion. The Figure 46 corner plot shows strong linear correlations between film diffusion, particle porosity and particle diffusion. The same effect, with a difficulty in estimating these parameters, was noted in the synthetic case in chapter 3.3.1.3 with the additional difficulty of experimental data. The wide error source, shown in Table 50, continues the pattern with a wider upper and lower bound on the HDI. The Figure 47 corner plot also shows a strong mostly linear correlation between film diffusion, particle porosity and particle diffusion and weaker linear relationships to most other variables. The linear relationships have carried over from the previous stage and compounded the difficulty already inherent in this stage. While the MAPs are still close to each other for the narrow and wide error source they are not as close as in previous stages. The narrow error source still has a narrower HDI than the wide error source. From Table 51 the wide and narrow error sources converge at a very similar rate. Approximately 2 million simulations where required for convergence indicating how difficult these problems are to solve.

As in the synthetic Linear binding model example we can look at the error model and chromatograms resulting from the posterior distribution. In Figure 48 the top row shows the error model, P(B|A), with the first pulse in cell A and the second pulse in cell B for the narrow error source. The bottom row is a visualization of the chromatograms generated by sampling from the posterior distribution with the first pulse in cell C and the second pulse in cell D. From cells A and B, the error model is essentially flat and then falls off extremely rapidly at the edges. This indicates the two experiments did create a confined region but that the experiments do not entirely agree with each other. The posterior distributions C and D have approximately the same width as the error model in A and B and the same structure where there is a broad high probability area and a rapid falloff and the edges. What can be seen in C and D is that the experimental data on the very front and back of the peak are completely outside the posterior region or even the region of the error model. This indicates there is an error related to the model. Something is happening in the physical system that the model and all the estimated parameters are not capable of capturing. This is valuable information that cannot be obtained from parameter estimation and indicates where further work is needed. Looking at Figure 49 for the wide error source the analysis is essentially the same. The error model in cells A and B are wider than their counterparts in the narrow error source but exhibit the same flat probability behavior that falls off quickly at the edges. In cells C and D, showing the posterior plots, while the structure still shares the same flat high probability area and a quick falloff it slightly better captures the experimental data within the high probability region. The wide error does not explain the front and back of the peak. While the wide error leads to a wider region which naturally covers more of this area the structure does not follow the experimental data on both data sets. This is further evidence that something is missing from the model and that this effect is not caused by errors related to pump delays.

Table 49: Non-Binding narrow.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Film Diffusion	1.76e-06	2.19e-06	2.52e-06	34.9	0.37
Particle Porosity	0.319	0.327	0.333	4.2	0.04
Particle Diffusion	2.15e-11	2.21e-11	2.33e-11	8.3	0.10
Column Dispersion	5.09e-07	5.17e-07	5.27e-07	3.6	0.03
Column Porosity	0.332	0.337	0.337	1.5	0.01
Tube Dispersion	2.70e-06	2.70e-06	2.70e-06	5.73e-04	
Tube Cross Section Area	1.87e-05	1.89e-05	1.89e-05	1.43e-02	

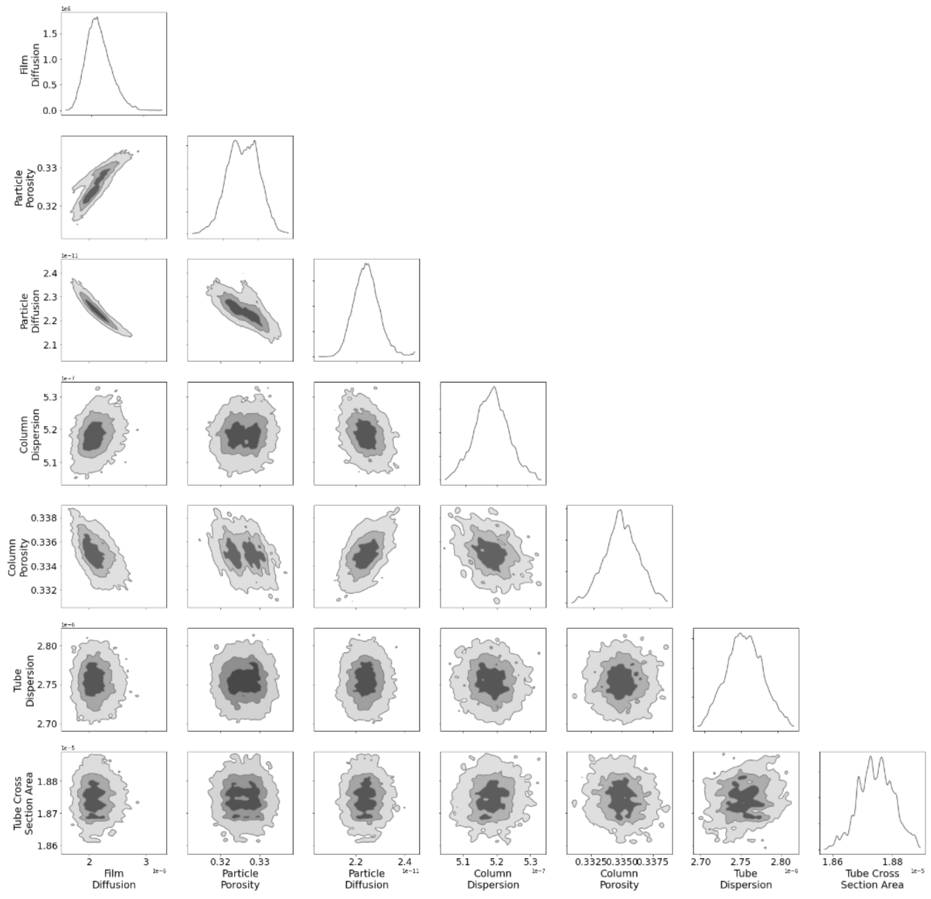

Figure 46: Experimental Non-Binding protein experiment narrow error source corner plot.

Table 50: Non-Binding Wide.

Name	*LB HDI*	*MAP*	*UB HDI*	*% MAP*	*max(%MCSE)*
Film Diffusion	1.44e-06	1.69e-06	2.39e-06	56.1	0.75
Particle Porosity	0.316	0.326	0.335	5.6	0.12
Particle Diffusion	2.19e-11	2.46e-11	2.52e-11	13.1	0.19
Column Dispersion	5.02e-07	5.01e-07	5.36e-07	6.8	0.06
Column Porosity	0.331	0.337	0.340	2.5	0.03
Tube Dispersion	2.52e-06	3.07e-06	3.09e-06	18.5	0.21
Tube Cross Section Area	1.79e-05	1.81e-05	1.92e-05	7.2	0.09

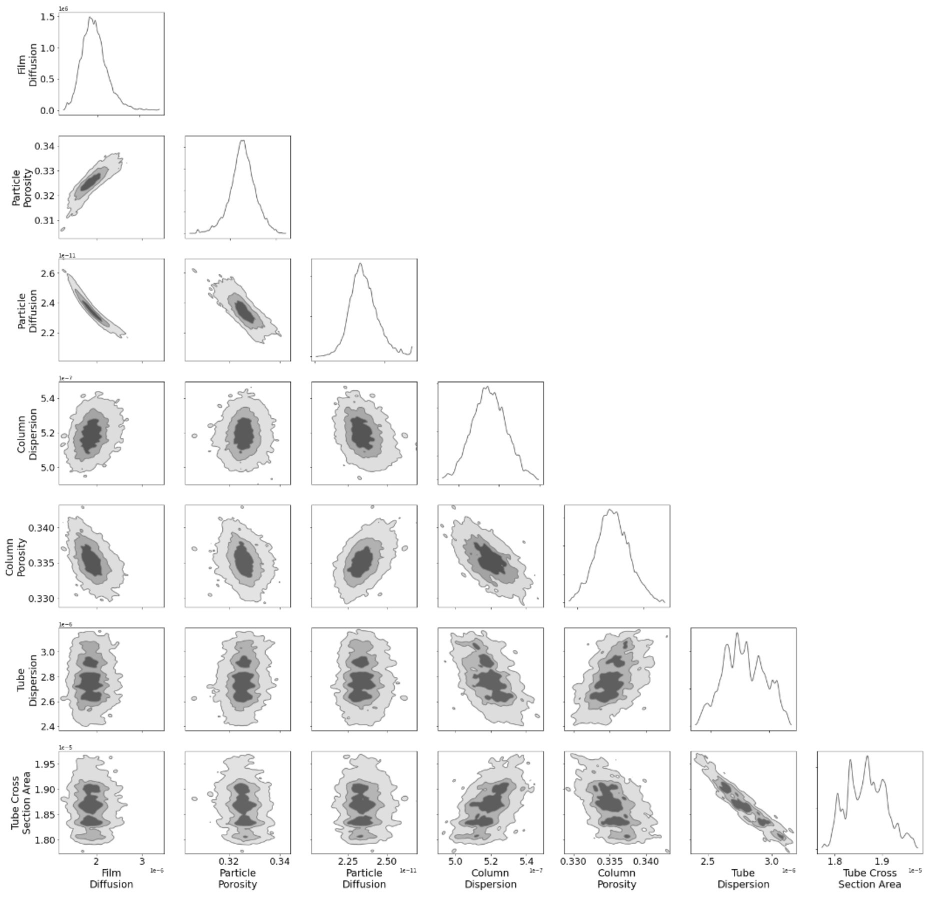

Figure 47: Experimental Non-Binding protein experiment wide error source corner plot.

Table 51: Non-Binding experimental results Narrow vs Wide.

Name	*Narrow*	*Wide*
Chain Simulations	1,689,216	2,046,080
Chain Length	13197	15985
Simulation Time	2 days, 7:43:25	2 days, 3:48:41
max(IAT)	263	319
Acceptance Ratio	0.058	0.048

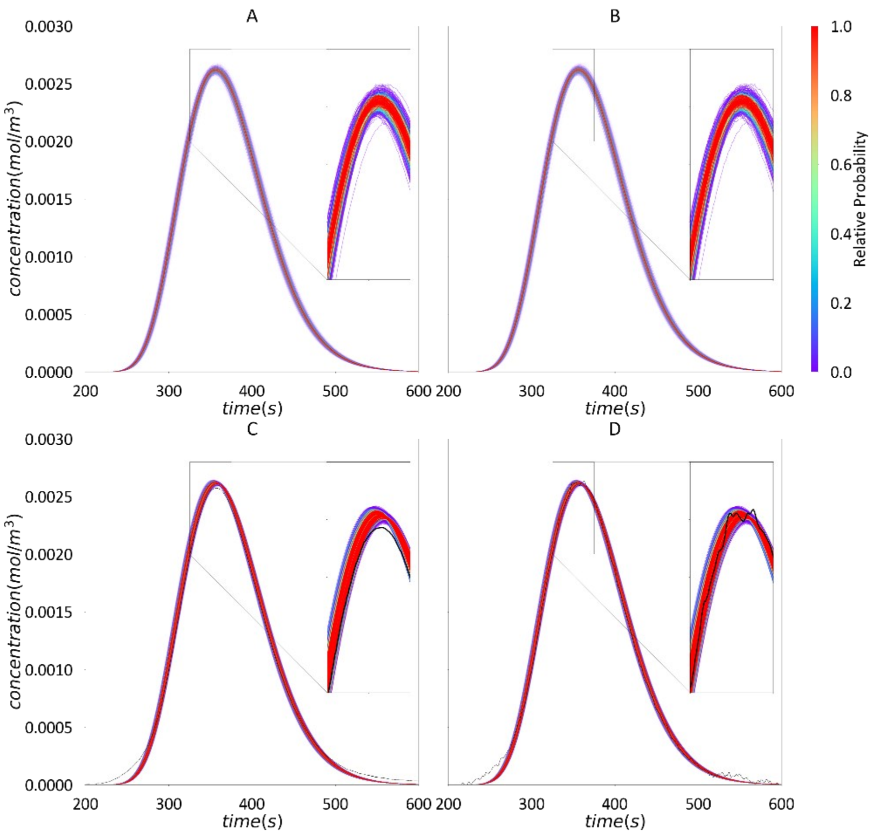

Figure 48: Plots of the narrow error model and posterior chromatograms colored by relative probability A and B: Narrow error model experiment 1 and 2 C and D: Narrow posterior with experimental data shown in black for experiment 1 and 2.

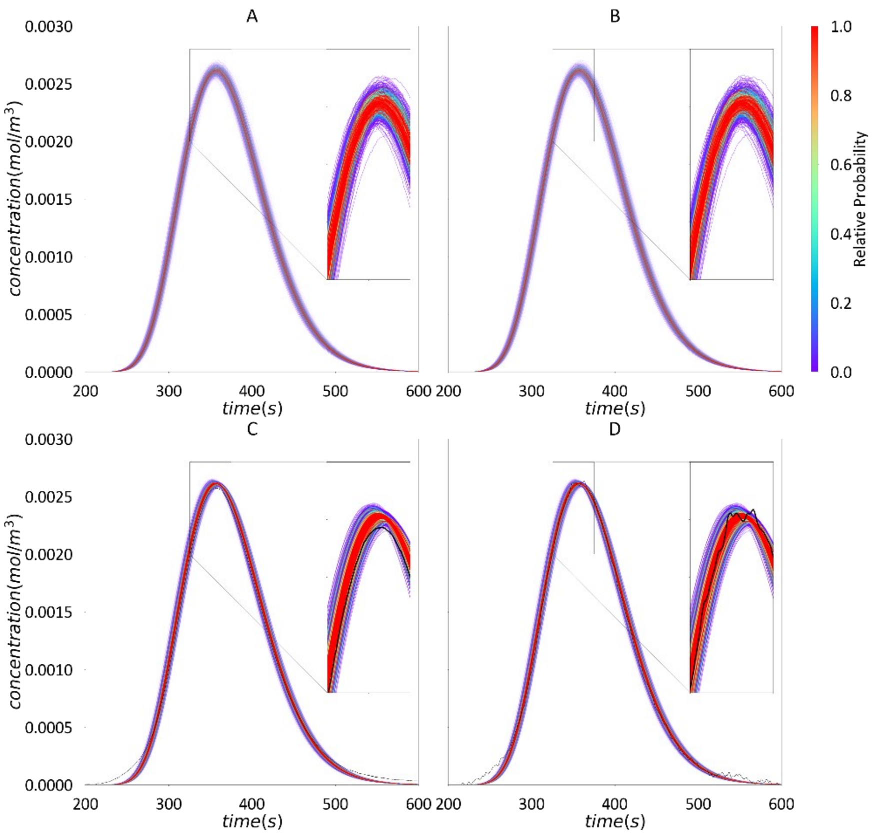

Figure 49: Plots of the wide error model and posterior chromatograms colored by relative probability A and B: Wide error model experiment 1 and 2 C and D: Wide posterior with experimental data shown in black for experiment 1 and 2.

3.4. Summary

A complete process has been demonstrated for modeling errors and obtaining parameter distributions along with the uncertainty on the resulting chromatogram(s). The process covers everything from creating the error model to sampling it to how to assess when sampling has converged. A novel approach of constructing error models for chromatography has been introduced and demonstrated using synthetic and experimental data. The error model presented here covers errors due to pump delays, loading concentration, pump flow rates and UV detector noise and is easy to extend to additional sources of error. This entire process is based on Bayesian uncertainty analysis and MCMC. MCMC provides a way of incrementally refining the probability distribution of the parameters within a single stage until convergence. Bayes' theorem then allows the distribution to be carried over from one stage to the next.

There is a drawback to uncertainty quantification and that is the number of simulations needed and the time it takes. As the dimensionality increases more simulations are needed. To further compound this problem later stages of estimation often involve simulations that become more complex because features like binding are modeled. To uniquely identify the parameters for a linear binding model with a single component only a single experiment is needed, and it takes about 3 seconds to run in the case presented here. For the Steric Mass Action (SMA) binding model a typical simulation, with a single component, grows to about 10 seconds and normally three experiments are needed to uniquely identify the parameters which means 30 seconds are needed for the simulation. Based on extrapolation from the time needed to solve the problems presented here and adjusting for simulation times changing from a linear binding model to a SMA binding model would increase the time to 1 year and for a 2-component SMA model it would increase it to 3 years using the same computing resources used here. Figure 50 shows the correlation between the number of parameters and the number of simulations needed for convergence with every synthetic and experimental example in this chapter as a dot on that plot. Figure 51 takes the number of experiments needed from Figure 50 and then adjusts it for the time simulations take to run. As can be clearly seen in both plots the time and simulations needed scales exponentially with the number of parameters. This naturally leads to the next chapter with surrogate modeling to speed up the simulations.

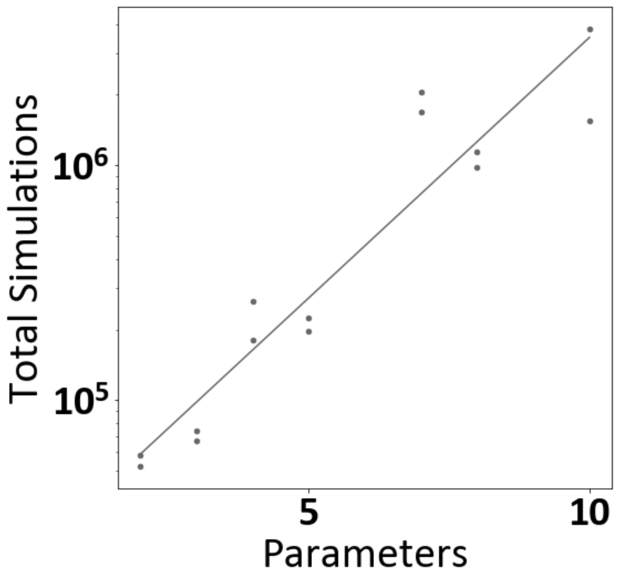

Figure 50: Parameters vs total number of simulations scaling in MCMC.

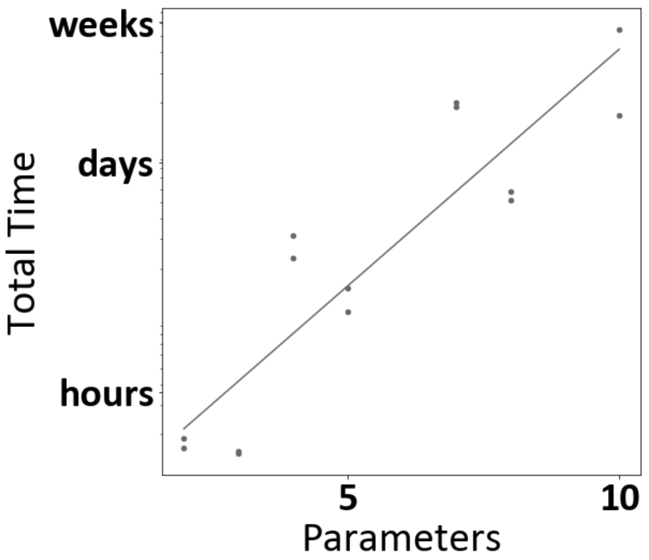

Figure 51: Parameters vs total time in simulation scaling in MCMC.

4. Surrogate Modeling

4.1. Introduction to Surrogate Modeling

A surrogate model is a model designed to approximate another model at lower computational cost. A surrogate model normally operates in a narrower parameter range and with reduced accuracy but at greater speed. The CADET simulator takes a vector of parameters as an input and generates one or more vectors as output which requires a surrogate model with vector input and output.

The simplest type of surrogate model is regression using a known equation such as polynomial, exponential, or logistic regression but, can also include regression using physical equations such as chemical reaction kinetics or equations of motion. This type of regression is widely used, and the equation can normally be determined by inspection of the data. Regression using a known equation is unsuitable as a surrogate for CADET due to no such function being known.

Most surrogate methods have a vector input and a scalar output and as such are unsuitable as a surrogate for CADET. These methods include support vector machines [75] and tree-based methods [76]. While it is possible to build a regression model for each element of the output vector and concatenate them together this does not result in a useful output due to lacking a connection between adjacent entries.

Gaussian Process Regression (GPR) [77] and Artificial Neural Networks (ANN) [37], [42] can approximate a function with a vector output. GPR approximates an unknown function as a linear combination of kernel functions and involves inverting a covariance matrix that scales linearly with the amount of training data used. GPR requires a good choice for a kernel function to function as a surrogate and there is currently no known way to automatically choose an appropriate kernel. GPR also suffers from getting slower to train and predict as more data is added to the system. While the number of rows and columns in the covariance matrix scales linearly with additional training samples the cost of inverting the matrix is proportional to the number of samples cubed. GPR works best with limited experiments where uncertainty information is needed which is not applicable to this case. ANN improve as more data is used for training and training time increases linearly with data while prediction is independent of the amount of data used for training. ANN support incremental training, additional data can be added to improve an already trained model. Finally ANN have excellent library support [78], [79] and help available to design and improve networks for nearly any type of task. All surrogate modeling in this paper is done with ANN.

4.2. Introduction to neural networks

Neural networks are an example of biomimicry. They are modeled after how neurons work in the brain. A neuron can take input from one or more neurons and send output to one or more neurons. ANN can be universal function approximators [37], [42], [43]. Function approximation is also called regression and while neural networks can be used for classification and other tasks that it outside the scope of this thesis. The functions approximated do not have to be smooth or continuous and can be n-dimensional. Chromatography simulations involve solving coupled partial differential equations, ordinary differential equations, and algebraic equations as discussed on page 20. Solving these equations can be slow and create a serious bottleneck for parameter estimation, Table 24 on page 60, and uncertainty quantification, Table 42 on page 86. The chromatogram that results from solving the above equations is smooth and continuous with small changes in equation parameters resulting in small changes in the output chromatogram. A neural network can approximate the output of a chromatography simulator and even on a laptop it can run hundreds of thousands of times faster than a chromatography simulator can. The drawback of neural networks is they need to learn the function approximation, and this requires training on a lot of data.

The math behind neural networks is simple and Figure 52 shows a very simple network that has 3 inputs, 2 hidden layers and a single output. Both hidden layers are fully connected dense layers. The nomenclature for this network is show in Table 3 on pg. 19. Eq. 39–50 contain all the equations for a forward pass through the neural network. Beginning with eq. 39 the inputs, x_n, with n indicating the neuron index at the current layer, are assembled into a vector, x. Next the weight matrix, $W^{(l)}$, eq. 40 and bias matrix, $b^{(l)}$, eq. 41 are assembled with the l indicating the layer index. The terms in these matrixes, $W^{(l)}_{n^*,n}$and $b^{(l)}_n$ with n^* indicating the neuron index in the next layer, are the trainable parameters that a neural network learns. The input to the first hidden layer,$z^{(l)}$, is eq. 42 which is a matrix-vector product plus a vector. This input to the neuron goes through an activation function, $a^{(l)}$, eq. 43 to create the output for the first hidden layer. Activation functions continuous and differentiable but do not have to be smooth. The process then repeats with a new weight matrix eq. 44 and bias matrix 45 to create the input to the second hidden layer eq. 46. The output from the second hidden layer goes through the activation function eq. 47 and with a weight matrix eq. 48 and bias matrix eq. 49 the output is calculated with eq. 50. The entire process of running the network is a series of matrix-vector products, vector-vector additions and applying an activation function to a vector. This system has a total of 17 trainable parameters adding up all the weights and biases with 3 inputs and 1 output.

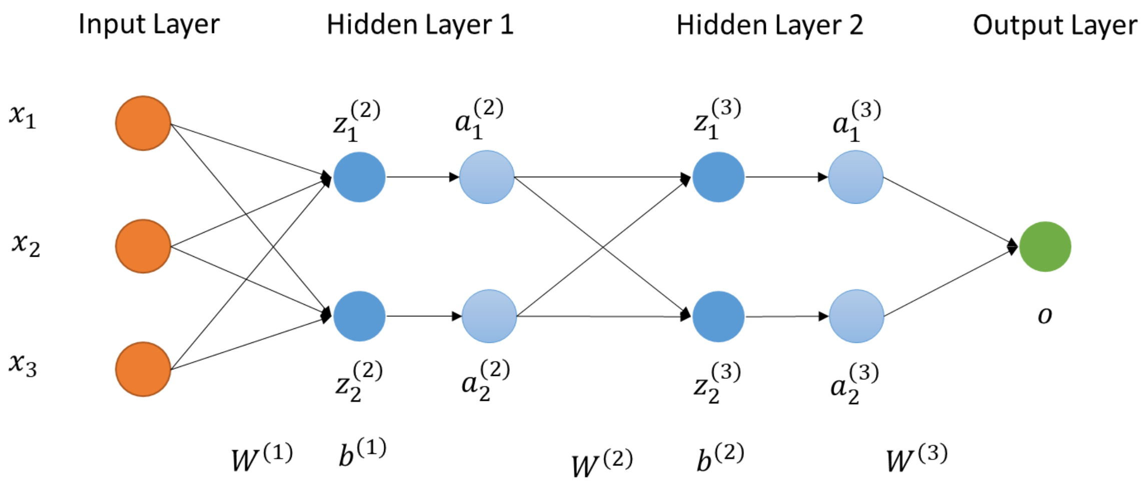

Figure 52: Simple dense network with 2 hidden layers and a single output.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{39}$$

$$W^{(1)} = \begin{bmatrix} W^{(1)}_{1,1} & W^{(1)}_{1,2} & W^{(1)}_{1,3} \\ W^{(1)}_{2,1} & W^{(1)}_{2,2} & W^{(1)}_{2,3} \end{bmatrix} \tag{40}$$

$$b^{(1)} = \begin{bmatrix} b^{(1)}_1 \\ b^{(1)}_2 \end{bmatrix} \tag{41}$$

$$z^{(2)} = W^{(1)}x + b^{(1)} \tag{42}$$

$$a^{(2)} = f\left(z^{(2)}\right) \tag{43}$$

$$W^{(2)} = \begin{bmatrix} W_{1,1}^{(2)} & W_{1,2}^{(2)} \\ W_{2,1}^{(2)} & W_{2,2}^{(2)} \end{bmatrix} \tag{44}$$

$$b^{(2)} = \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} \tag{45}$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)} \tag{46}$$

$$a^{(3)} = f\left(z^{(3)}\right) \tag{47}$$

$$W^{(3)} = \begin{bmatrix} W_{1,1}^{(3)} & W_{1,2}^{(3)} \end{bmatrix} \tag{48}$$

$$b^{(3)} = \begin{bmatrix} b_1^{(3)} \end{bmatrix} \tag{49}$$

$$y_{pred} = W^{(3)}a^{(3)} + b^{(3)} \tag{50}$$

To train a network a loss function must be applied. In neural networks there are many loss functions used depending on the purpose of the network. To train a network with back propagation, which is the standard approach, the only requirement is that the loss function and activation functions are differentiable. The loss function, eq 51, takes only two arguments, the output of the network and the data to match against. An optimizer is then used to reduce the loss function. The simplest optimizer is Stochastic Gradient Descent(SGD) [80]. Other optimizers will be introduced later, and they all expand on this optimization process. SGD uses back propagation by taking the derivative of the loss with respect to each weight and bias to update the weight, eq 52, and bias, eq 53. The learning rate, r, adjusts how quickly the weights and biases are updated. These equations create the core training loop where the network takes an input and predicts an output. The output is compared to what the true output should be. The loss in then calculated, the weights and biases are updated, and the process continues until some termination condition is reached. The termination condition is usually based on either a fixed number of training iterations, a fixed time to train, or lack of improvements to the loss function.

$$L = loss\left(y_{pred}, y_{true}\right) \tag{51}$$

$$w_{n^*,n}^{(l)} \coloneqq w_{n^*,n}^{(l)} - r\frac{\partial L}{\partial w_{n^*,n}^{(l)}} \tag{52}$$

$$b_n^{(l)} \coloneqq b_n^{(l)} - r\frac{\partial L}{b_n^{(l)}} \tag{53}$$

This approach to training does have a problem in that it does not actually work very well for training a network which is why a small extension is needed. The issue is that this presents a single set of data to the network at each evaluation and then updates the weights. This causes the weights to make large swings as the network is trained which leads to poor convergence and predicting points that the network has not been trained on. The solution to this problem is training in batches where instead of a single value presented to the network many are evaluated at once. In the case of this network x would become a matrix with the number of rows equal to the batch size and 3 columns while y would become a vector with length equal to the batch size. This changes eq. 42 to a matrix-matrix product plus a vector where the vector is applied column wise. This results in the weights and biases update with the average weight and bias change of the batch. The average weight and bias change for a batch is the sum of the weight and bias changes divided by the number of entries in the batch. This results in weight and bias updates that better represent the data and results in faster convergence with less computational effort and improved predictive capabilities based on the loss function. This process is also what makes neural networks so well suited for a GPU. At its core a GPU is a matrix processer and neural networks are a bunch of matrix operations. A GPU takes approximately the same amount of time to evaluate a single point as it does to evaluate tens of thousands of points. In most applications the only limit is the amount of memory the GPU has and how quickly data can be transferred back and forth to the GPU. The highest end desktop GPU right now is an RTX 3090 Ti which runs at 40 teraflops or 4e13 operations/second. Video games have pushed GPU development to make better looking games at higher framerates and neural networks are able to use that power also.

The training algorithm is just one part of the training process. A sufficiently large network can memorize the training data without being able to predict the output of data it has not seen before. To solve this problem the entire dataset is split into a training and validation dataset and no data point appears in both data sets. Typically, a dataset is split up with 75% training data and 25% validation data, but this varies with the amount of data available. The network is trained on the training dataset and the weights and biases are updated with back-propagation. The network is then tested with the validation dataset and the loss calculated without a back-propagation step in order to check the accuracy of the network on data it has not been trained on This process is repeated until some stopping condition is reached. Ideally the average of the loss function over the training and validation datasets will be almost the same. If the loss is much better for the training dataset than the validation dataset it means the network is overfitting. Overfitting indicates the network is memorizing the training data instead of the underlying function and some form of mitigation will be needed such as changing the batch size or the learning rate. If the validation dataset is much better than the training dataset an error has occurred with the training data, validation data, or the training process that needs to be examined in detail. This would indicate that the system is doing a much better job predicting data it has never seen than data it has been trained on which should not be possible. It is normal for the validation dataset to be a little better than the training dataset. This is due to updating the weights in batches and so the early batches have a higher loss than later ones which increases the average loss of the training dataset. If training and validation data have a similar loss value and the fit is poor, then the network is underfitting. Underfitting means the network lacks the capability to learn the underlying function and this must be solved by changing the network. If the neural network is accurate enough and there is no sign of under or over fitting it can then be used as a function approximator. It is important to remember that the network should only be used for interpolation and not extrapolation.

SGD, as already explained, is a simple gradient descent optimizer and while technically it is designed to run on one training example at a time it is normally extended to run on a batch. Even with these changes for many problems SGD gives very poor convergence and many extensions have been made to it over the years. One of the important additions to SGD was momentum[81]. Momentum is a linear combination of the last change in the weight and bias update with the current gradient information. It works similarly to momentum in physics where movement in a direction has a resistance to change in its direction and this helps the optimizer move over small local minimums. SGD has a learning rate, r, which is used in eq. 52 and eq. 53. This learning rate is referred to as a hyper-parameter and tuning it is critical for optimizer performance and has a typical range of 1e-1 to 1e-5. Tuning these parameters is very time consuming as different values must be set and then the neural network trained at each value to determine the optimal value. The need for hyper-parameter tuning is one of the primary reasons more advanced algorithms have been developed which need less or no hyper-parameter tuning. RMSProp was created by Geoffrey Hinton as an extension to SGD and has only been published as lecture slides it is a commonly used optimizer that is built into most machine learning libraries and covered in his slides at https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. RMSProp adds an adaptive learning rate so that the learning rate is increased for parameters that have a consistent gradient direction and decreased for those that have an inconsistent gradient direction. This optimizer substantially improved over SGD in practice. Adam is an extension to RMSProp and uses the gradient and second moment of the gradient to adapt the learning rate [82]. Rectified Adam [83] (Radam) is an extension to Adam that prevents the adaptation of learning at the very beginning of optimization. At the very beginning of optimization all the parameters are going through rapid changes and adapting the learning rate during this phase can be detrimental to overall convergence. The Ranger optimizer is built on top of Radam and adds a lookahead feature [84]. Lookahead is a meta-optimizer, and it can be added to other existing optimizers. Lookahead adds slow weights to the system in addition to the normal gradient update and periodically merges in the information from the normal gradient updates and then resets the internal optimizer to the new value of the slow weights. The purpose of this is to allow quick updates to the gradient while training but allow the optimizer to overall move in the direction of the average gradient. This lessens the impact of hyper-parameters on the optimizer and allows the optimizer to move efficiently navigate over local minimums.

4.2.1. Activation functions

Activation functions are a critical part of neural networks and are what add non-linearity to the network. Without activation functions a neural networks output is a linear combination of the inputs no matter how many layers are added. This means that adding additional layers doesn’t increase network capacity which makes such a network effectively useless. Activation functions have a very difficult problem. They must be non-linear to add non-linear learning capabilities to a network. They also must deal with the problems of exploding gradients and vanishing gradients [85]. Exploding gradients is when the gradient for one of the weights or biases tends towards infinity. During back-propagation the derivative of the activation function is needed and if the network is a deep network this derivative will be used many times. This creates a problem because of the chain-rule for derivatives and networks of increasing depth due to the derivatives of the activation function being repeatedly multiplied together. If the derivative is larger than 1 the gradient will increase at each layer and tend towards infinity. Vanishing gradients is the same kind of issue except if the derivative is less than 1 the gradient will decrease at each layer and tend towards zero. That makes it important to have a function that is non-linear and has a bounded derivative around the value of 1. The functions must also be deterministic and simple mathematically for performance reasons and because they will need to run on a GPU and be simple to run in parallel.

The simplest activation function is a linear activation function, eq 54, and while it can’t be used for any of the deep layers there are two places where it is used in a neural network. The activation function for the input layer, by default, is a linear activation function which passes the inputs directly to the first hidden layer. For a regression problem a linear activation function is also used for the final layer which means the output is a weighted linear combination of the inputs to the final layer and can thus have any real value. A linear activation function is a null operation and the same as no activation function since it is just a weighted sum. The hyperbolic tangent[86] is another common activation function, eq 55, that has mostly fallen out of favor due to suffering from the vanishing gradient problem. Rectified Linear Unit (ReLU)[87], eq 56, is currently the most popular activation function. ReLU trains faster and provides better generalization than tanh while not suffering from the vanishing gradient problem. ReLU does have a drawback though and that is when x is less than zero the gradient is zero and this results in a problem known as the dying neuron problem. The dying neuron problem is caused by the chain rule during back-propagation where a zero in any term causes the entire chain to become zero and as a result weights are not updated. One of the solutions to this problem is using a Scaled Exponential Linear Unit (SELU)[88], eq 57, which fixes the problem of vanishing and exploding gradients. The SELU activation function also has the property that if the input has a mean of 0 and a standard deviation of 1 so will the output. This property turns out to be very useful for neural networks because it helps keep the entire network with a mean of 0 and a standard deviation of 1 which improves training speed and generalization due to this region being where activation functions are most sensitive. Swish[89], eq 58, is an activation function found by automated search. Functions were broken down into many basic operations and different combinations were created and used as activation functions in a network to see how it performed. Swish has proven to work well on deep networks [90], [91]. Unlike most of the other activation functions swish is smooth. All the activation functions are shown in Figure 53 along with their derivatives in Figure 54.

$$linear(x) = x \tag{54}$$

$$\tanh(x) = \tanh(x) \tag{55}$$

$$relu(x) = \max(0, x) \tag{56}$$

$$selu(x) = \begin{cases} \quad x \geq 0 & s * x \\ x < 0 & s * a * (\exp(x) - 1) \end{cases}, s = 1.05070098\,, a = 1.67326324 \tag{57}$$

$$swish(x) = x * sigmoid(x) \tag{58}$$

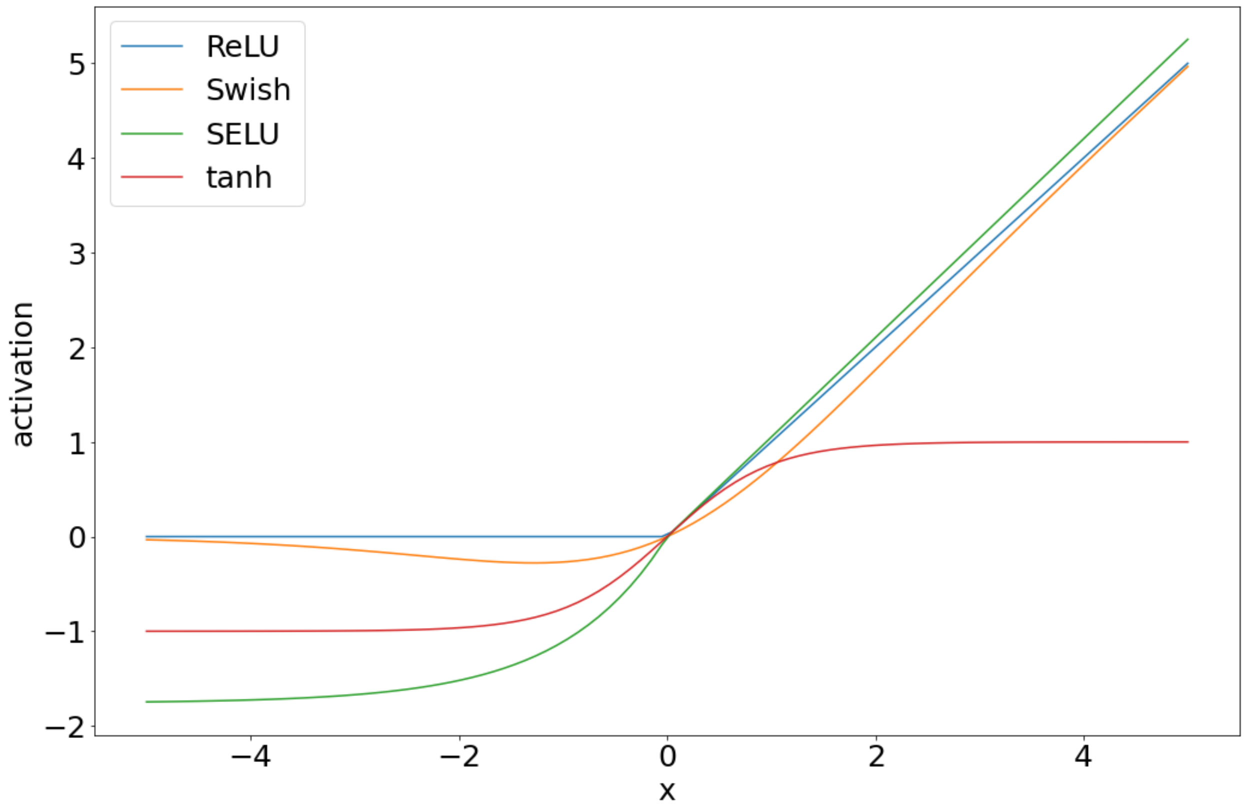

Figure 53: Plot of common activation functions.

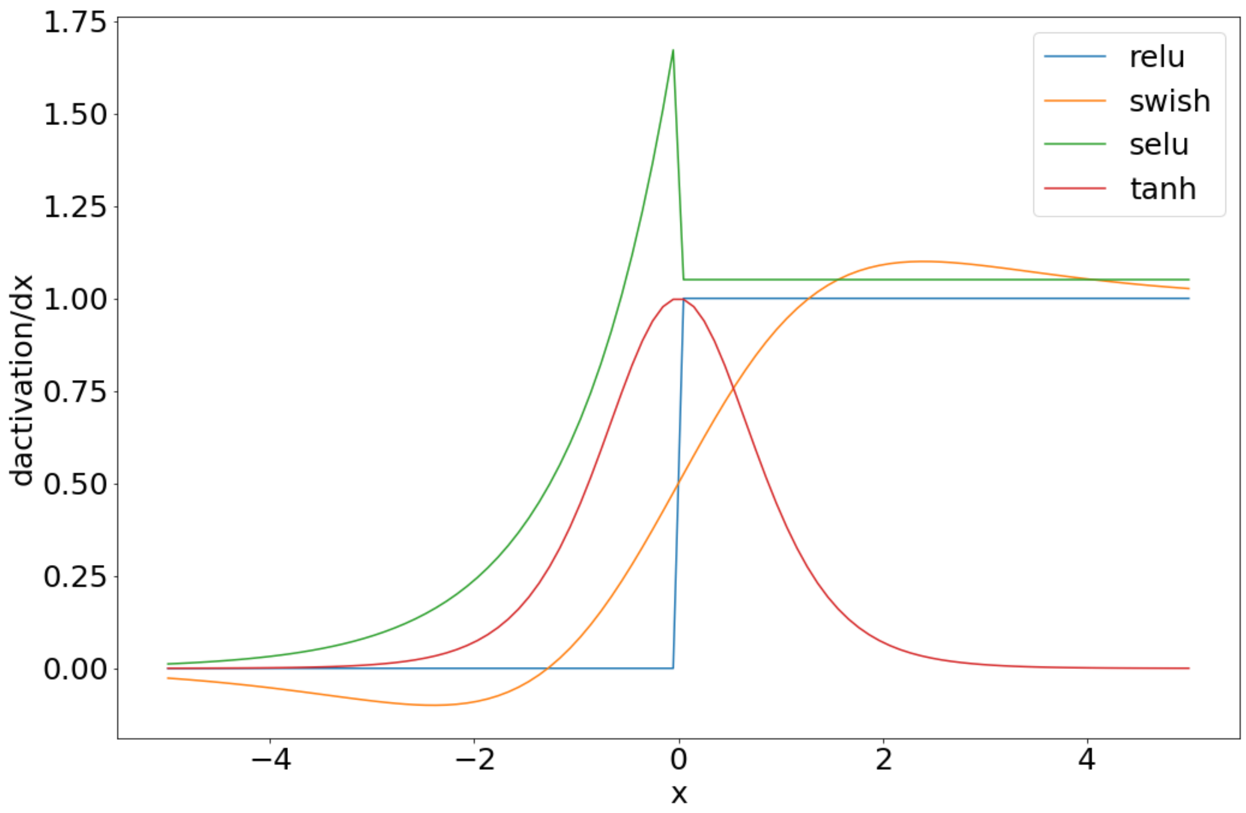

Figure 54: Plot of common activation function derivatives.

4.2.2. Layers

There are many different types of layers in neural networks but only a few are typically used for regression problems. A dense layer is the simplest type of layer used and every neuron connects to every other neuron in the next layer. This makes dense layers very flexible in learning but also requires many weights to learn. The extreme flexibility works very well when approximating a function with a scalar output or a vector output where there is no direct relationship between adjacent entries. When predicting something like a chromatogram, neurons that are near each other should also have a similar output. A dense layer is not an ideal choice for this type of output because it can’t incorporate the additional information known about the problem. Networks with many more parameters than needed can also be very difficult to train. A Convolution layer is a much better choice as it is locally connected which means that neurons in the next layer are connected to nearby neurons from the current layer instead of being fully connected. This local connection reduces the number of weights needed and simplifies the training process. Figure 55 shows the basic structure of a convolution layer. There is a kernel which is a vector of learnable weights. This kernel can have its length and stride specified where the stride is how far the kernel shifts at each step. The layer can apply padding, indicated by orange boxes, which is a virtual extension of the input vector, before the convolution is performed and the default is to use zero for padding. The convolution operation itself is a dot product between a slice of the input vector and the kernel. For this example, the kernel weights have been set to 0.5, 1.0 and 0.5. The kernel is slid over the entire input vector. The example in Figure 55 shows only a single kernel but a convolution layer can have one or more or kernels and each one has its own weights. Convolution layers create a relationship between nearby neurons and requires fewer weights than a dense network does.

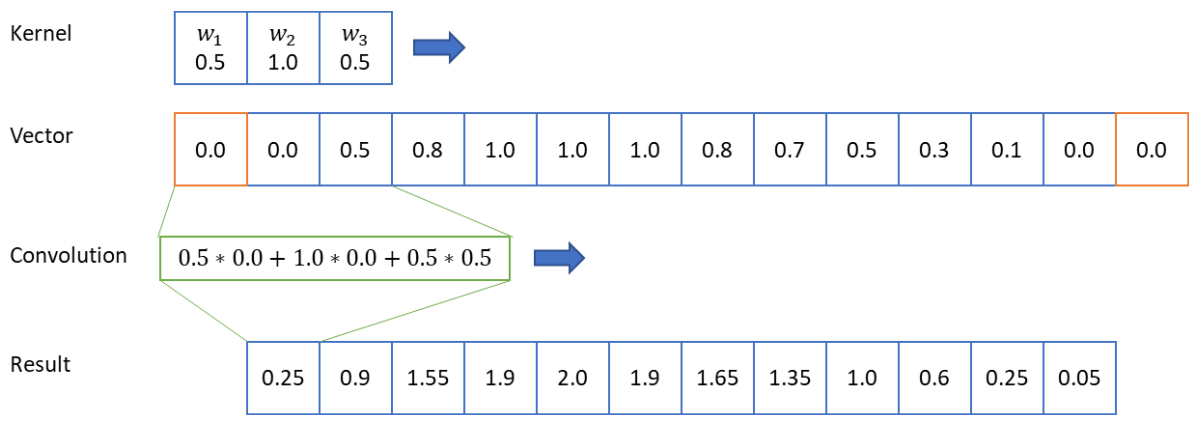

Figure 55: One-dimensional convolution layer with a single kernel of length 3 and a stride of 1 with 0.0 padding.

4.2.3. Loss functions

Loss functions are another critical part of neural network design. The purpose of a loss function is to guide the network towards a better solution. Loss functions must have an analytical derivative for the purpose of back-propagation. There are many standard loss functions for regression, and it is also possible to create custom loss functions. The common regression loss functions are the Mean Square Error(MSE),eq 59, Mean Absolute Error(MAE), eq 60, Mean Absolute Percentage Error(MAPE), eq 61, and the Mean Square Logarithmic Error(MSLE), eq 62. MSE is the most common regression loss function and has a large penalty on outliers. The penalty from MSE is independent of the scale of the ground truth so a prediction error of 1 with a ground truth of 0 has the same impact as a prediction error of 1 when the ground truth is 1000. MSLE scales the absolute impact of the error based on the ground truth. MAE is less sensitive to outliers due to not having the squared error term and is more commonly used for non-synthetic data. MAPE is another way to deal with ground truth scale variant errors however, it also requires that 0 is not in the ground truth which can be solved by adding an offset.

$$MSE = \frac{\sum\left(y_{true} - y_{pred}\right)^2}{y_n} \tag{59}$$

$$MAE = \frac{\sum\left|y_{true} - y_{pred}\right|}{y_n} \tag{60}$$

$$MAPE = \frac{\sum\left|\frac{y_{true} - y_{pred}}{y_{true}}\right|}{y_n} \tag{61}$$

$$MSLE = \frac{\sum\left(\log\left(y_{true} + 1\right) - \log\left(y_{pred} + 1\right)\right)^2}{y_n} \tag{62}$$

4.3. Network Design

For parameter estimation and uncertainty quantification CADET is a function that turns parameters into a chromatogram. CADET takes many parameters but only a few of them are changed when doing parameter estimation and uncertainty quantification and all the constant parameters can be left out of the neural network. As a result, the neural network must take approximately 2-20 parameters as input and produce a chromatogram as an output to function as a surrogate. For some types of applications, the initial concentration profile in the system and/or the inlet concentration profile is also important for tasks like processes optimization or uncertainty quantification under varying feed compositions. While the networks presented here have not been specifically designed to solve this problem, it should just be a matter of extending the input vector and adding more training data. A suitable network must take a vector as an input and product a vector as an output. Many designs were tried that where unsuitable, and I will briefly cover them before presenting the final design. All networks were built in Python using TensorFlow and the code describing the structure of the networks is taken from working code and then slightly simplified. The nomenclature for the code is shown in Table 52. Qualitative graphs have been generated for all of the networks using VisualKeras [92]. Activation layers have been removed from all graphs and the dimensions have been clipped and, in some cases, layers have been removed to aid visualization.

Most of the network designs presented here require the output length to be a power of 2. This is dealt with by resampling the chromatogram using scipy.interpolate.InterpolatedUnivariateSpline[49]. The spline is fit to the data and a new time grid is chosen that is a power of 2. After the network makes a prediction, the same process is used to resample the chromatogram back to the original time grid.

The most obvious design is using stacked dense layers since we already know that such a network can approximate any function theoretically. The Python code for this network is shown in Code 1 and a qualitative graph is shown in Figure 56. This design is simple to build but ran into severe problems during training. The core issue is that a dense layer is an all to all connection between layers and this created many ANN parameters. As layers were added or the number of neurons in a layer were increased the loss did not improve, and the chromatograms produced were too inaccurate, based on visual inspection, to be used as a surrogate for some datasets. While this network did work for some datasets it did not work for others. This rendered it unsuitable for use in an automated system like CADET-Match. With 5 layers and 512 neurons per layer this model has 1.3M trainable parameters.

Table 52 Code nomenclature

Name	*Description*
activation(x)	Apply the activation function to the x layer
Concatenate()([x, y, z])	Concatenate together the layers x, y, and z
Conv1D(filters = filters, kernel_size = kernel_size)(x)	Create a 1D convolution with filters number of kernel and each kernel of length kernel_size
Conv1Dtranspose (filter = filters/2, kernel_size = kernel_size, strides=2)(x)	Create a 1D transposed convolution with filters number of kernels with each kernel of length kernel_size and stride equal to 2
Dense(neurons)(x)	Create a Dense layer with number of neurons and connect it to the x layer
for _ in range(number)	Iterate number times
Input(shape=(length,))	Create an Input layer with length neurons
Model(input_layer, output_layer)	Compile the entire model given the input and output layers
Reshape((1, length))(x)	Change the shape of x from a vector to a 1D matrix
UpSampling1D()(x)	Double the length of x by duplicating each entry [1,2] -> [1,1,2,2]

Code 1 Dense network Python where the input dimension, output dimension, number of layers, neurons per layer and activation function are all variables

```
z_in = x = Input(shape=(input_dimension,))
```

```
for _ in range(layers):
    x = Dense(neurons)(x)
    x = activation(x)

z_out = Dense(output_dimension)(x)
model = Model(z_in, z_out)
```

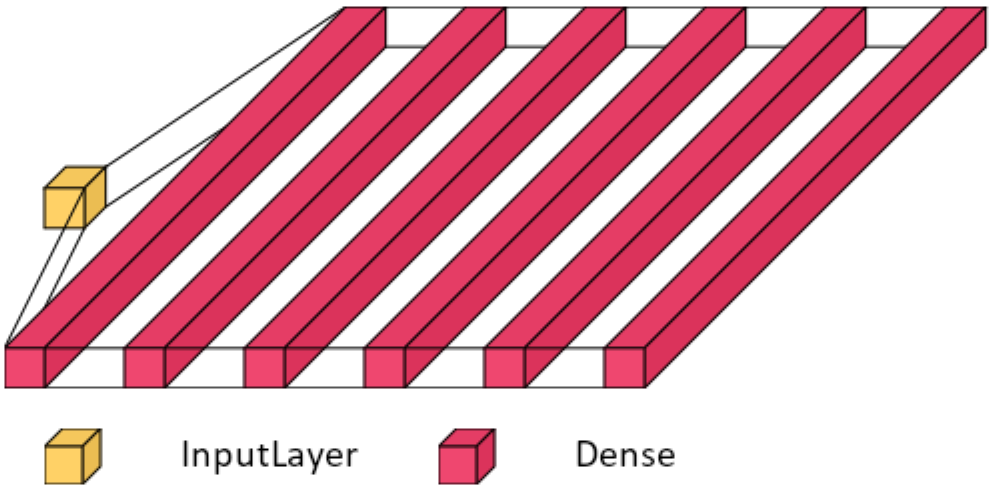

Figure 56: Qualitative graph of Code 1 with fully connected dense layers and activation layers not shown.

The next model I built was an inverted pyramid of convolutional networks based on help from the MLQuestions [93] subreddit and the BE-GAN [94] paper. The Python for this network is shown in Code 2 with a qualitative graph shown in Figure 57. This network is more complex than has been discussed about so far and the layers are not simply connected to the previous layer. This model requires the output vector length to be a power of 2 and resampling is used. This network carries the input data, the previous layer, and the current layer to the next layer. This network worked sometimes but proved to be very difficult to train and was also quite slow. For some unknown reason the network was effectively unstable where sometimes it would train and other times the error would remain high, and no training progress was observed. This seemed to be independent of the data set and would sometimes require multiple attempts to get a trained working network. This model was discontinued due to the unstable training problem. A surrogate that can't reliably train is unsuitable for any kind of automated deployment and can't be part of CADET-Match. With 64 kernels and a kernel size of 16 this model has 680K parameters.

Code 2 Inverted pyramid convolutional neural network based on BE-GAN where the input dimension, output dimension, number of kernels, kernel size and activation function are all variables

```
z_in = x = Input(input_dimension)
z = Reshape((1, input_dimension))(x)
x = Reshape((1, input_dimension))(x)

power = int(log2(output_dimension))

for _ in range(power):
    y = x
    x = Conv1D(filters = number_kernels, kernel_size = kernel_size)(x)
    x = activation(x)
    x = Concatenate()([x, y, z])
    x = Conv1D(filters=number_kernels, kernel_size=1)(x)
    x = UpSampling1D()(x)
```

```
        z = UpSampling1D()(z)
    x = Conv1D(filters = number_kernels, kernel_size = kernel_size)(x)
    x = activation(x)
    x = Conv1D(filters=1, kernel_size=1)(x)
    z_out = Reshape((output_dimension,))(x)

    model = Model(z_in, z_out)
```

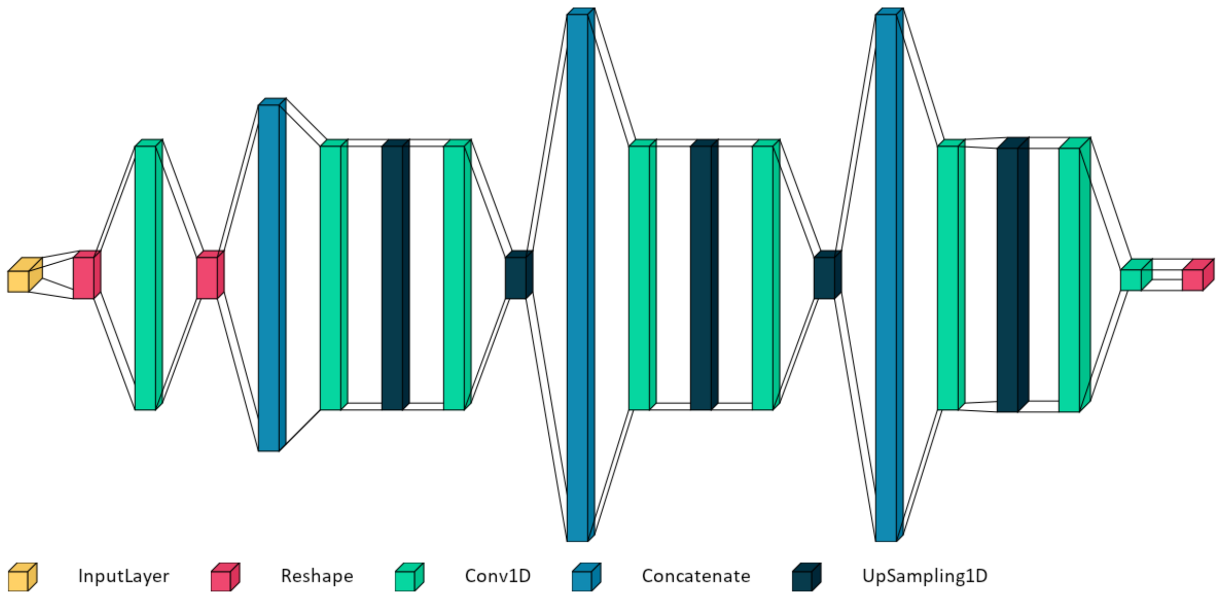

Figure 57: Qualitative graph of code 2 with only two internal layers shown instead of nine and activation layers not shown.

The network presented, shown in Code 3 with a qualitative graph shown in Figure 58, here was initially based on the regression network in https://github.com/PV-Lab/BayesProcess. This network is a simple inverted pyramid with two dense layers follower by convolutional layers where each layer is twice the width of the previous layer. This network also does not suffer from the problem of getting stuck with a high training error on all datasets tested. This model requires the output vector length to be a power of 2 and resampling is used. This network can have a variable number of inputs, but the output layer is a vector of length 512. In total the optimized version of this network has 2.5 million parameters. By neural network standards this is quite small with many common networks having between 100 million and 100 billion parameters. The next step is to tune all of the hyperparameters which are the loss function, optimizer, batch size, learning rate, number of kernels, kernel size, and the activation function.

Code 3 Inverted pyramid convolutional neural network based where the input dimension, number of kernels, kernel size and activation function are all variables

```
    z_in = Input(shape=(input_dimension,))
    z1 = Dense(filters)(z_in)
    z1 = activation(z1)
    z2 = Dense(32 * number_kernels)(z1)
    z2 = activation(z2)
    z3 = Reshape ((32, number_kernels))(z2)
    z4 = Conv1Dtranspose (filter = number_kernels/2, kernel_size = kernel_size,
strides=2)(z3)
    z4 = activation(z4)
    z5 = Conv1Dtranspose(filter = number_kernels/4, kernel_size = kernel_size,
strides=2)(z4)
```

```
    z5 = activation(z5)
    z6 = Conv1Dtranspose(filter = number_kernels/8, kernel_size = kernel_size,
strides=2)(z5)
    z6 = activation(z6)
    z7 = Conv1Dtranspose(filter = 1, kernel_size = kernel_size, strides=2)(z6)
    z7 = activation(z7)
    z_out = Reshape((512,))(z7)

    model = Model(z_in, z_out)
```

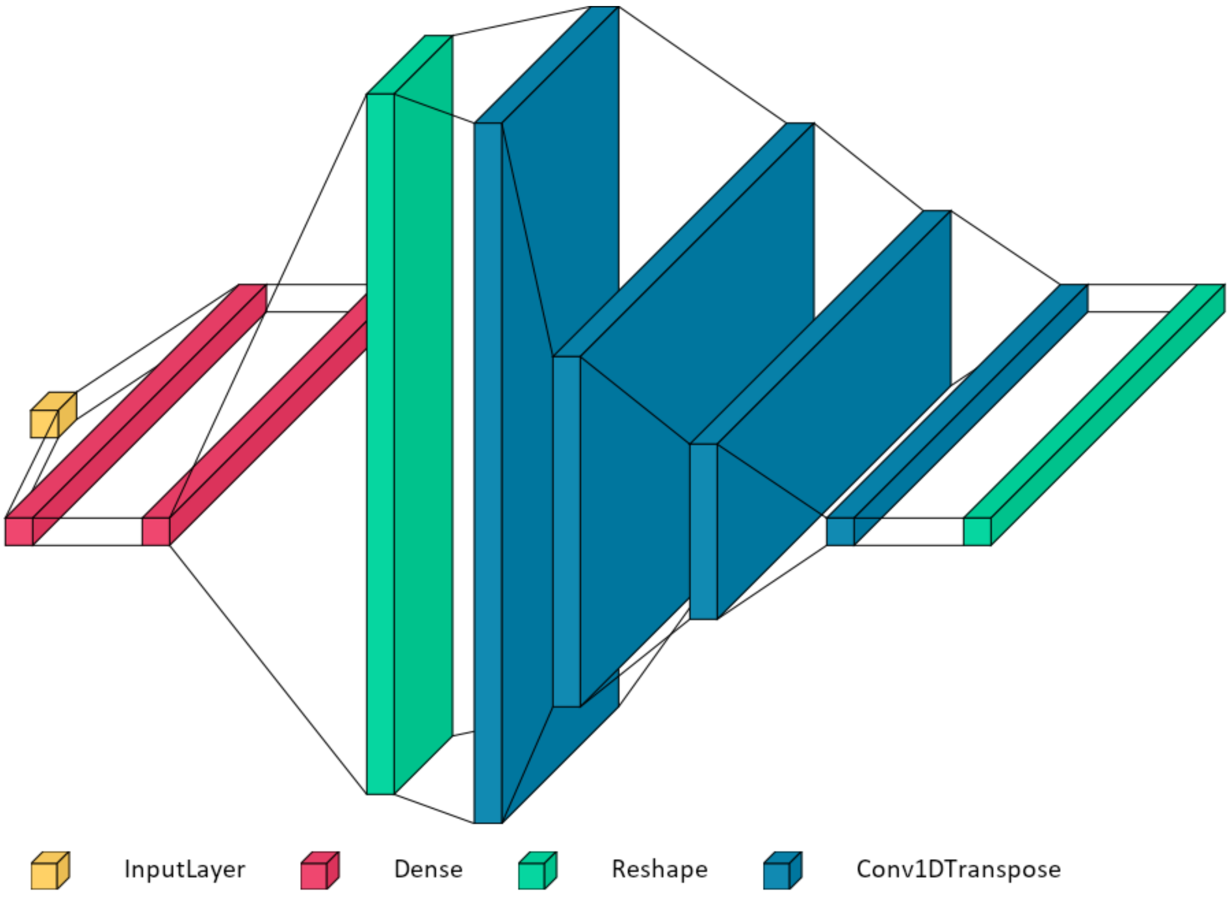

Figure 58: Qualitative graph of code 3 with dimensions clipped and activation layers not shown.

4.3.1. Hyperparameter and Model Tuning

Neural networks are still a young field and unlike older methods we don't have ways to know what the proper settings for a network are ahead of time. As a result there are many tools designed to solve this type of problem like KerasTuner [95] and Ray Tune [96]. However, these tools are largely automatic and don't help in understanding how tuning works and the impact of different hyperparameters. As such a stage wise approach will be used here so that each step can be explained and the impact on the network shown. The optimal parameter for one stage is carried over the next stage. While it is true that hyperparameters could be coupled for the purpose of explanation this will be ignored and is generally also ignored in automatic tuners. The starting configuration is shown in Table 53. The initial data set used is a linear binding example with 10 input parameters from section 3.3.1.4 on page 83. The same bounds are used as the MCMC study uses and 100,000 points are generated using a Sobol sequence. The first 75% of the data is used for training and the last 25% is used for validation. A Sobol sequence spans a unit n-cube, and it has a useful property that a subset of the sequence also spans the unit n-cube without duplication of points and this property ensures the entire space is covered for training and validation. The dataset can also be split randomly into a training and validation dataset but in high dimensions this leads to clusters and gaps in the training and validation datasets. The chromatogram output data is resampled to 512 points.

The network is trained on each hyperparameter value. An epoch is a single run through all the training data followed by testing the network on the validation data. A single network is trained until 100 epochs have occurred without improvement in the loss function on the validation dataset and the weights and biases from the epoch with lowest validation loss is used. Progress is shown in wall time which is the time a clock would record from beginning to end of a complete training run. Some hyperparameters result in more epochs taken but also much faster epochs so comparing by epoch can be misleading. In the end all that matters is how long does it take to get a trained network. The progress graphs are shown using the lowest training and validation loss seen up to that point in time. The selection of the best value for a hyperparameter is based on the lowest validation loss. All training was done on an XMG Apex 15 laptop with a Ryzen 9 3900 12-core CPU, Nvidia RTX 2070 8GB GPU, and 64GB of RAM.

Table 53: Neural Network starting hyperparameter set for Code 3 neural network.

Parameter	*Value*
Loss	Mean Squared Error
Optimizer	ADAM
Batch Size	256
Number of kernels	256
Kernel size	8
Activation function	ReLU
Learning Rate	1e-3

4.3.1.1. Activation function

The first hyperparameter to optimize is choosing which activation function to use. Due to the depth of the network and the usage of the network for regression SELU, ReLU and Swish where considered. The results are show in Figure 59 and the results are clear with Swish being the best choice by having the lowest validation loss. From Table 54 swish is about two orders of magnitude better than SELU and has half the loss of ReLU. The losses in the table are the lowest losses for training and validation for each hyperparameter value at the time training completed and they don’t come from the same epoch. The validation data has a slightly lower loss than the training data which is expected. While training the weights are updated and validation is done on the final weights in an epoch which slightly favors the validation dataset. Swish is chosen as the activation function for future stages.

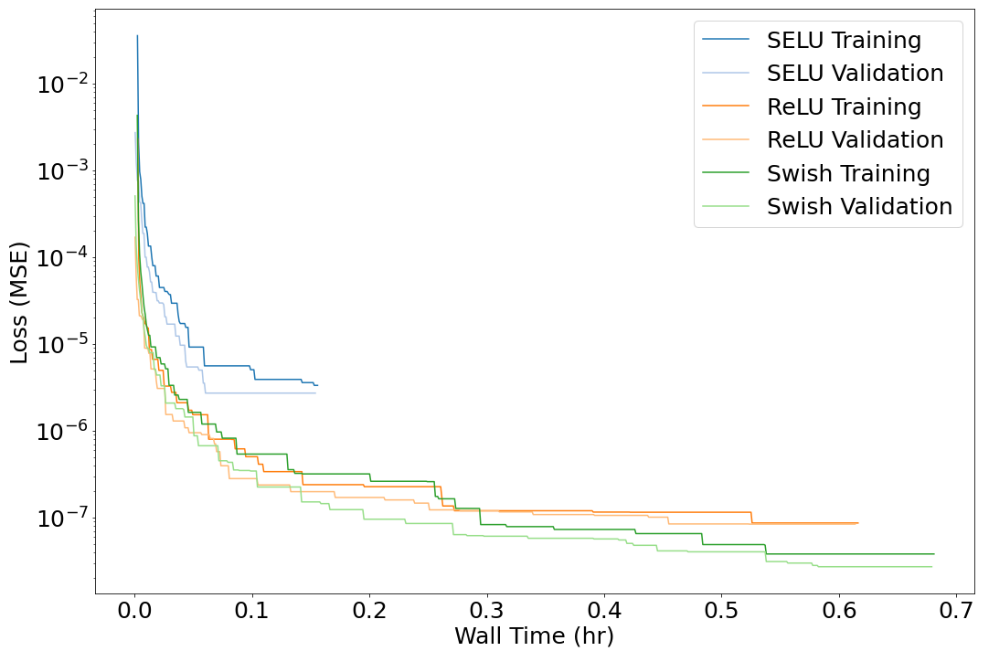

Figure 59: Hyperparameter tuning with the activation functions SELU, ReLU and Swish.

Table 54: Activation function comparison with minimum loss for training and validation along with wall time.

Activation Function	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
SELU	3.35e-06	2.73e-06	0:09:21
ReLU	8.70e-08	8.44e-08	0:36:58
Swish	3.82e-08	2.72e-08	0:40:51

4.3.1.2. Optimizer

The next hyperparameter to optimize is the choice of optimizer. In Figure 60 the results are show for Adam, RAdam and Ranger optimizers. All of them provide similar losses as can be seen in the graph and in detail in Table 55 with Adam doing better overall. Stochastic Gradient Descent (SGD) and other optimizers where also tested for completeness and shown in Figure 61. As a group they performed extremely poorly with SGD, AdaDelta and AdaGrad failing to train entirely. This helps to reinforce the point of why Adam and its variations are the standard. Adam is selected as the optimizer moving forward.

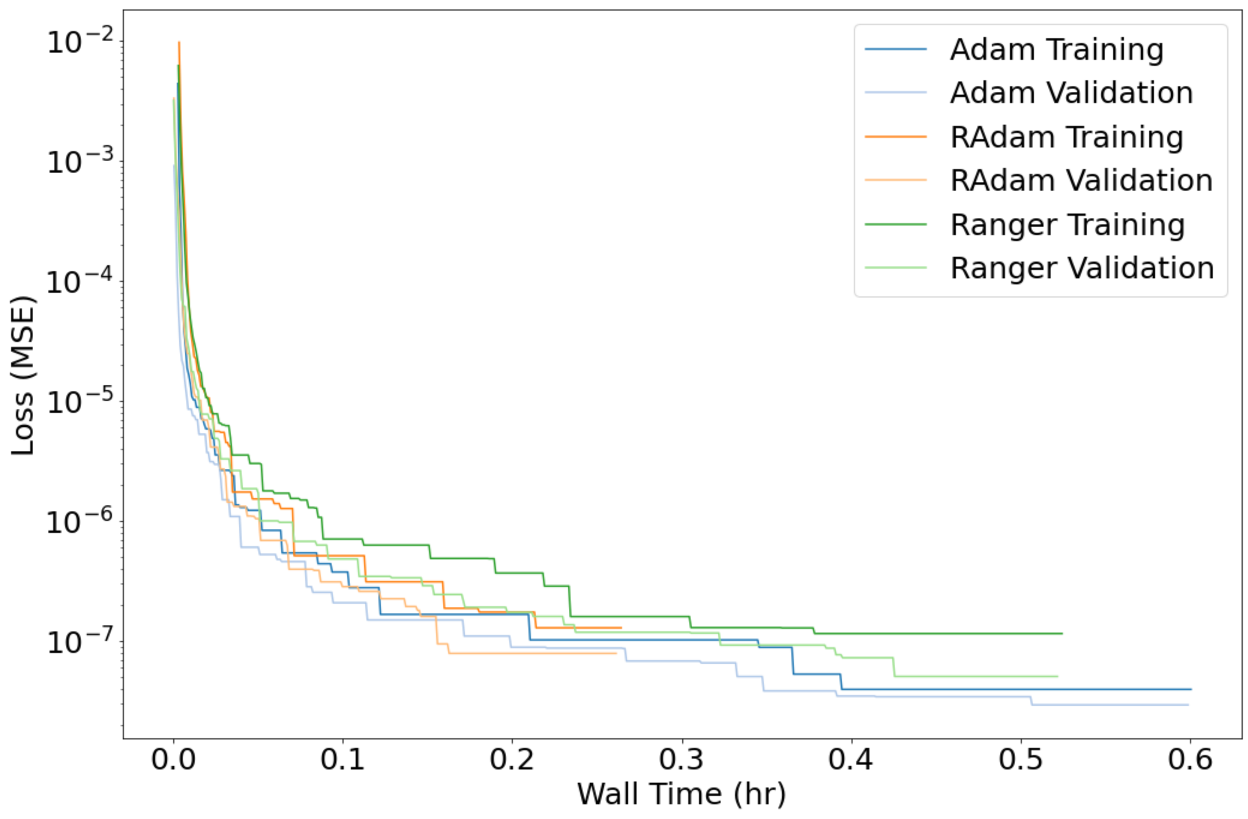

Figure 60: Hyperparameter optimizer comparison for Adam, RAdam, and Ranger optimizers

Table 55: Optimizer comparison with minimum loss for training and validation along with wall time.

Activation Function	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
Adam	3.96e-08	2.95e-08	0:36:02
RAdam	1.29e-07	7.90e-08	0:15:51
Ranger	1.16e-07	5.08e-08	0:31:28

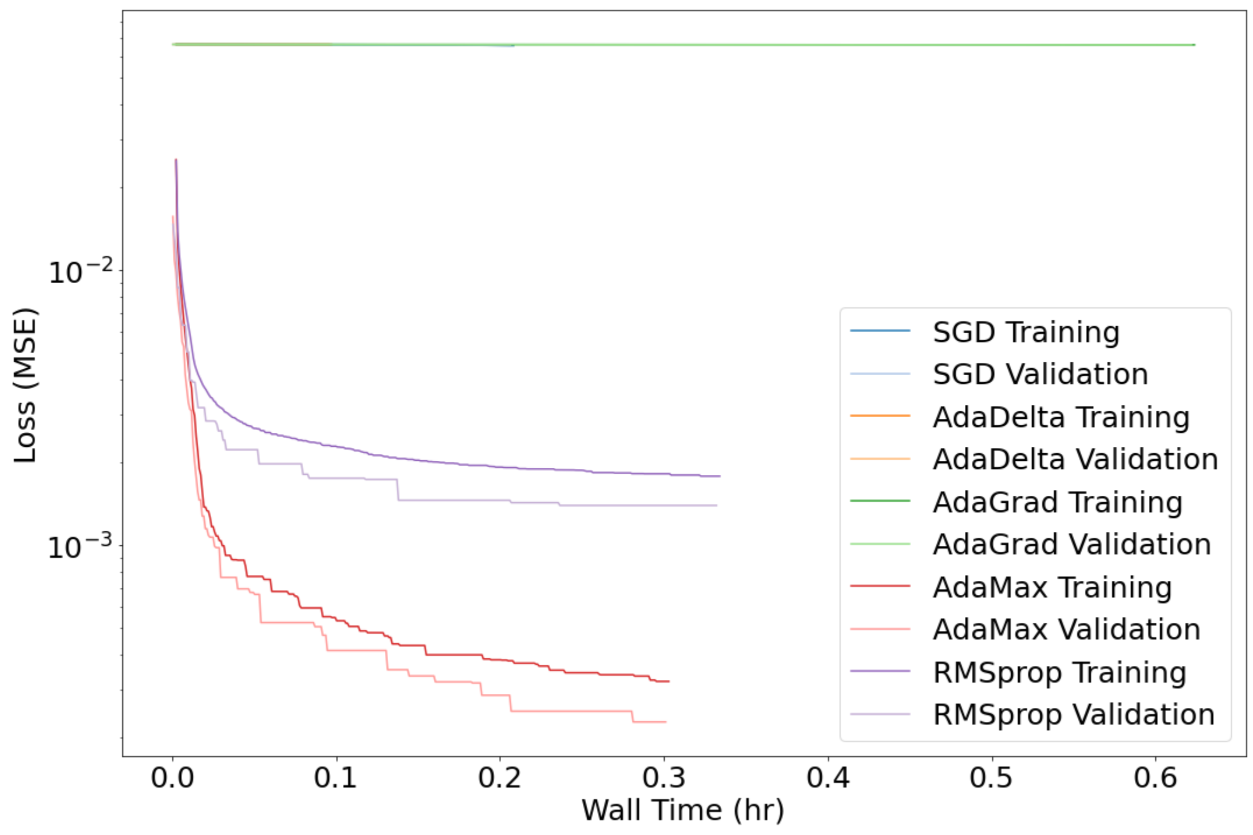

Figure 61: Hyperparameter optimizer comparison for SGD, AdaDelta, AdaGrad, AdaMax, and RMSprop optimizers.

4.3.1.3. Batch size

The third hyperparameter to optimize is the batch size. The batch size is how many items from the training dataset the network is trained on at a time. GPU are typically much faster working with larger batch sizes than small ones due to overhead and larger batch sizes allowing more internal parallelization. The batch size also has a regularization effect on training which means it can help prevent overfitting or getting caught in local minimums. At very small batch sizes the gradient updates won't be representative of the full dataset and slow training while larger batch sizes better represent the full dataset which helps avoid local minimums and improves training speed. There is a point where the batch size becomes too large and there is not enough variation batch to batch and result in stalling. Regularization also helps a network with generalization, which is the ability to predict data it has not been trained on such as the validation data. This makes finding the right batch size of critical importance. Batch sizes from 512 to 8192 are shown in Figure 62 and as can be seen as the batch sizes decreases towards 512 the loss function decreases and the network trains faster. As the batch size decreases there is also a larger separation between training and validation data sets as the regularization effect decreases with the smaller batch size. In Figure 63 batch sizes from 32 to 256 are shown and an interesting effect is seen with a batch size of 32 having worse performance and taking longer to train with a large separation between training and validation data sets. From these two graphs the optimal batch size is somewhere between 128 and 512. Additional batch sizes of 192 and 384 are run and shown in Figure 64. The complete dataset is summarized in Table 56. An optimal batch size of 192 is selected going forward.

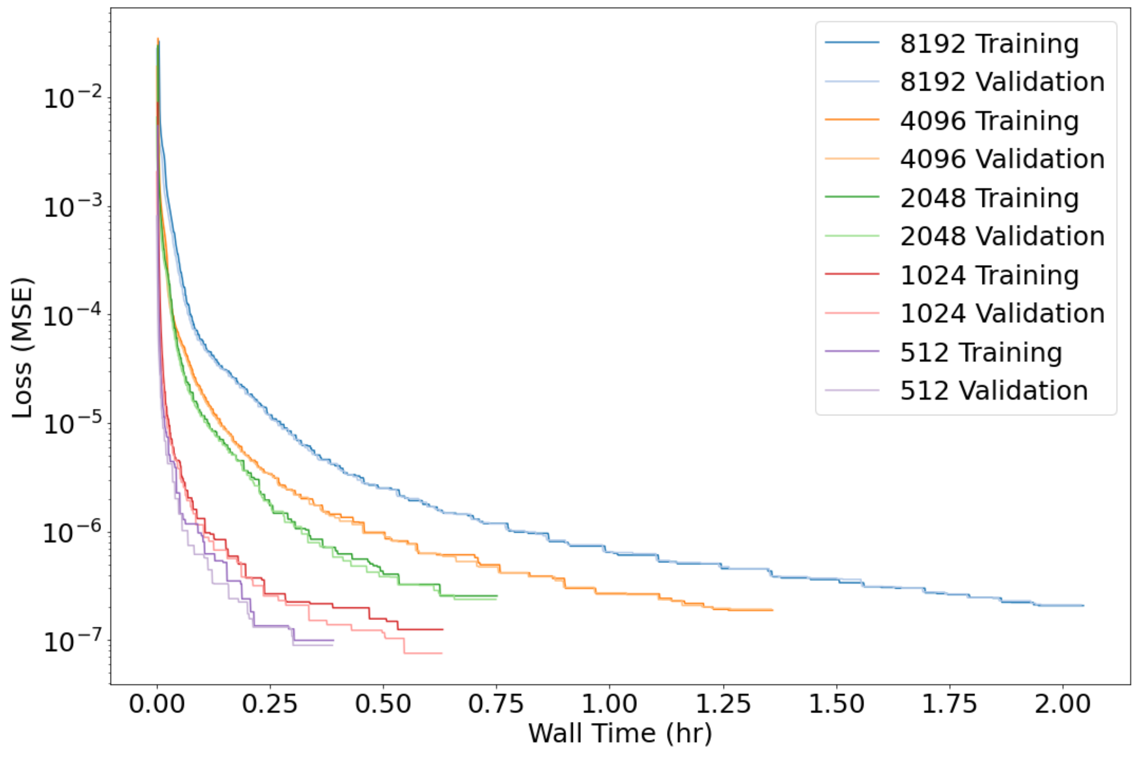

Figure 62: Batch sizes from 512 to 8192.

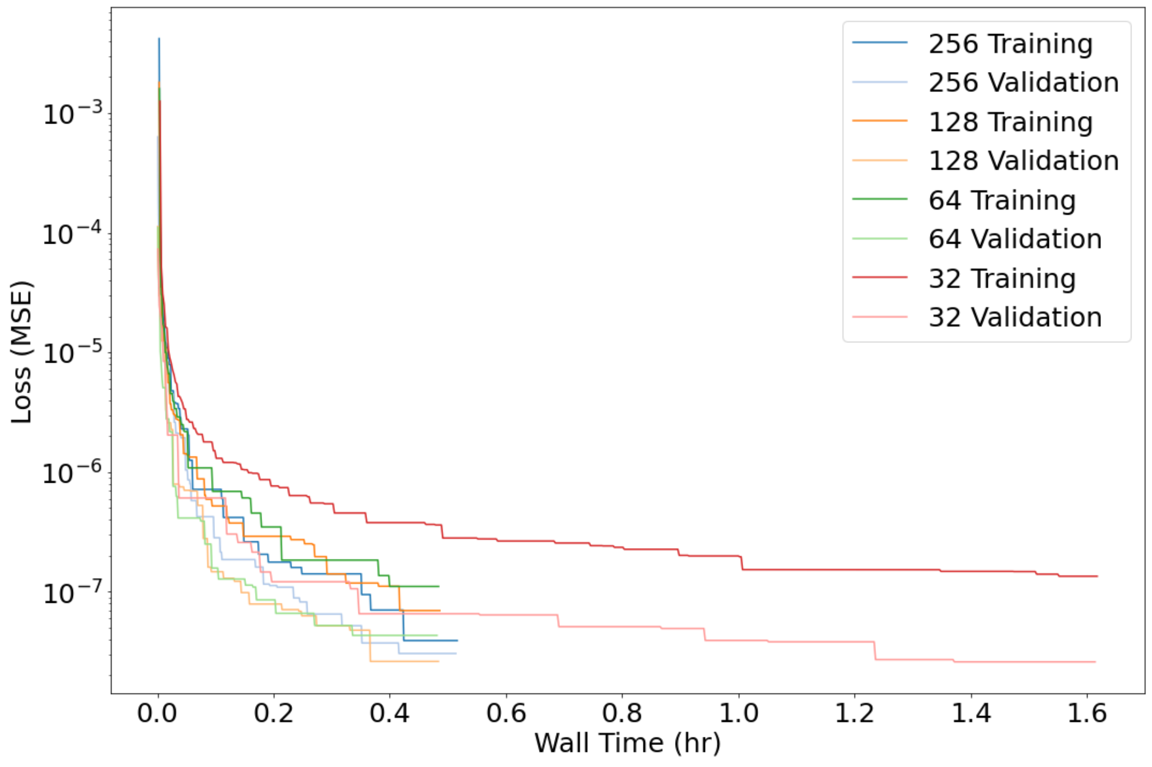

Figure 63: Batch sizes from 32 to 256.

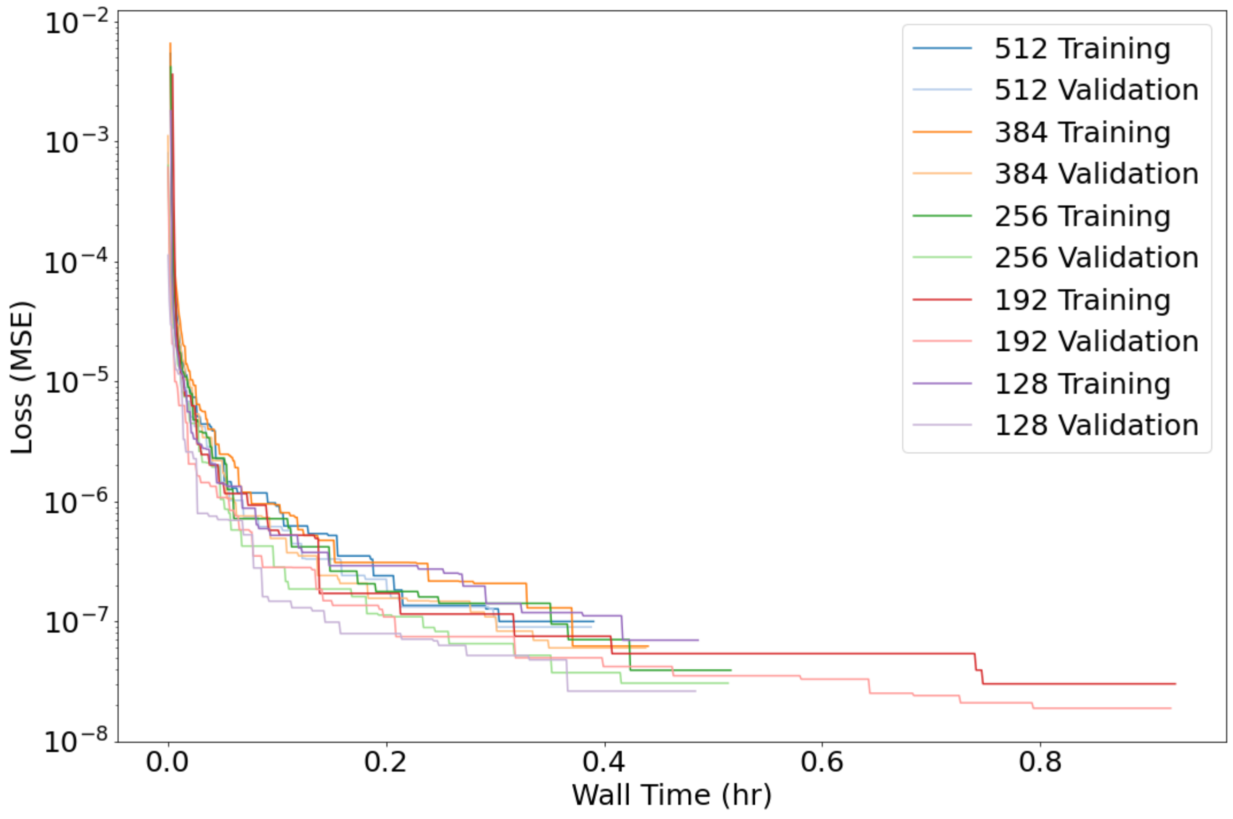

Figure 64: Batch sizes from 128 to 512.

Table 56: Batch sizes from 32 to 8192 with final losses and wall time.

Batch Size	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
32	1.35e-07	2.60e-08	1:37:04
64	1.11e-07	4.32e-08	0:29:02
128	6.97e-08	2.62e-08	0:29:10
192	3.01e-08	1.89e-08	0:55:24
256	3.91e-08	3.05e-08	0:30:58
384	6.21e-08	6.00e-08	0:26:25
512	9.97e-08	8.96e-08	0:23:25
1024	1.25e-07	7.58e-08	0:37:54
2048	2.56e-07	2.38e-07	0:45:05
4096	1.88e-07	1.93e-07	1:21:34
8192	2.08e-07	2.11e-07	2:02:44

4.3.1.4. Learning rate

The next hyperparameter to optimize is the learning rate for the optimizer. The learning rate controls the step size for the optimizer. The learning rate can also have a regularization effect similar to changing the batch size. Small steps tend to make slow and consistent process while large ones tend to make more rapid progress. Small steps are more likely to become trapped in a local area while large ones can move right over good optimums. The default learning rate for Adam in TensorFlow is 1e-3 and learning rates from 1e-3 to 1e-5 where tested and shown in Figure 65 with a summary table show in Table 57. Learning rates of 1e-1 and 1e-2 did not provide any meaningful progress and were omitted from the graph. A loss rate of 1e-3 had rapid convergence but also a larger separation between training and validation losses while 1e-4 achieved a better overall loss and 1e-5 had almost no separation between training and validation data sets. This narrowing of the difference between training and validation datasets losses is caused by a combination of the regularization impact of changing the batch size and an optimal learning rate. A learning rate of 1e-4 would give the optimal results on its own however learning rates can also be combined. I used the function ReduceLROnPlateau from TensorFlow to create an adaptive learning rate that starts at 1e-3 and decreases to 1e-5. The learning rate decreases by 50% whenever the log of the loss changes by less than 0.001 for 50 epochs. It is possible that convergence could be made even faster with more work on the adaptive learning rate, but this already works well enough without adding 3 more parameters to optimize. The adaptive method provided the same quick convergence of 1e-3 while also continuing to improve due the lowest learning rate of 1e-5. The adaptive learning rate is almost an order of magnitude better than any other learning rate and is the chosen method.

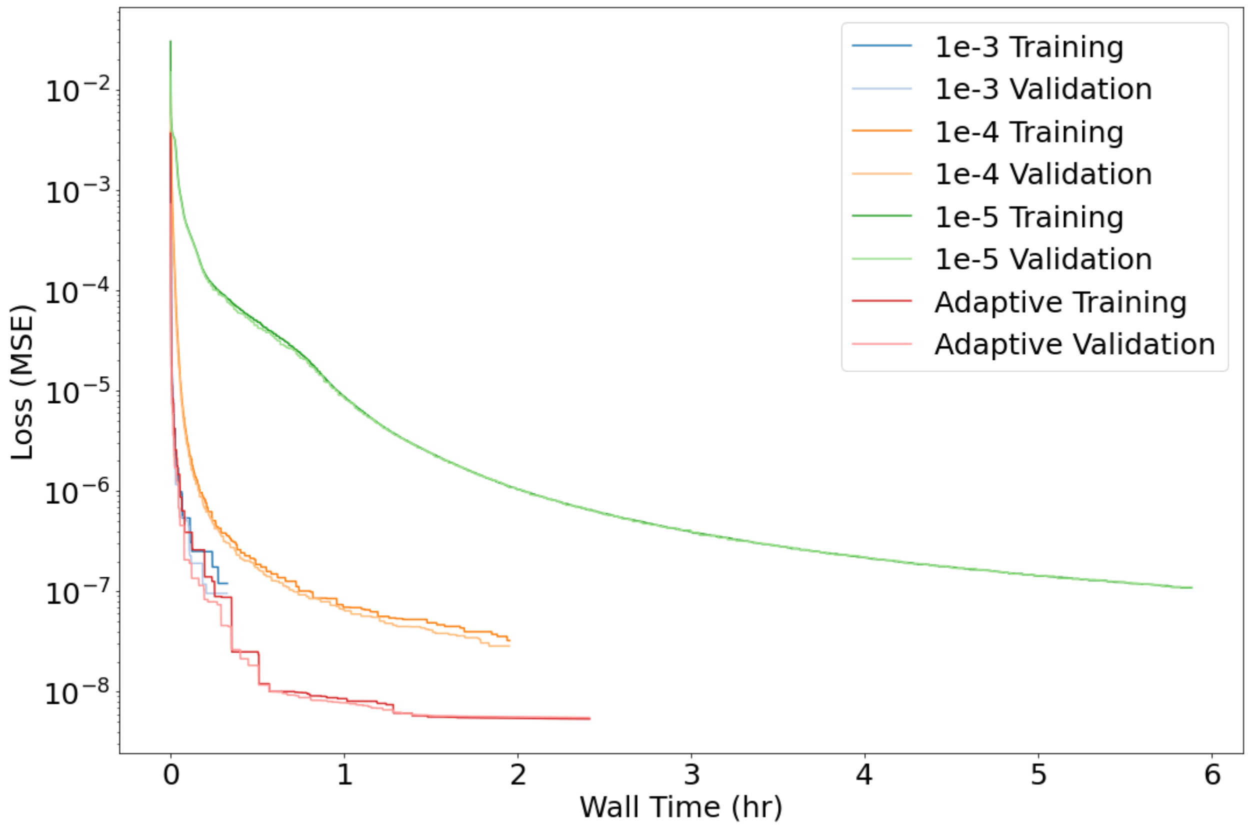

Figure 65: Loss rate graph using the Adam optimizer at learning rates from 1e-3 to 1e-5 along with an adaptive learning rate with 1e-1 and 1e-2 omitted due to poor results.

Table 57: Final loss values and wall time for the Adam optimizer at various learning rates.

Learning Rate	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
1e-1	3.37e-02	3.37e-02	0:06:53
1e-2	6.29e-04	5.70e-04	0:07:18
1e-3	1.20e-07	9.57e-08	0:19:45
1e-4	3.25e-08	2.86e-08	1:57:20
1e-5	1.09e-07	1.10e-07	5:52:59
Adaptive Learning Rate	5.36e-09	5.53e-09	2:24:56

4.3.1.5. Loss function

The next hyperparameter to optimize is the loss function. While it is possible to use a custom loss function that was not investigated and would be an entire topic unto itself. The loss functions MSE, MAPE, MAE, and MSLE were used for training but the loss in MSE was calculated for each of them so they could be compared more easily. In Figure 66 MSE and MAE performed much better than MAPE or MSLE with the final values shown in Table 58. A sample chromatogram is shown in Figure 67 which visually shows the difference in the loss functions. As would be expected from the low loss values MAE and MSE provided a good prediction of the entire chromatogram. MAPE fails to train entirely and predicts a flat line while MSLE only trains on the parts of the chromatogram that are not close to zero concentration. MAE is clearly better than any other option and is more than an order of magnitude better than MSE and about six orders of magnitude better than MAPE and MSLE. MAE is selected as the loss function moving forward.

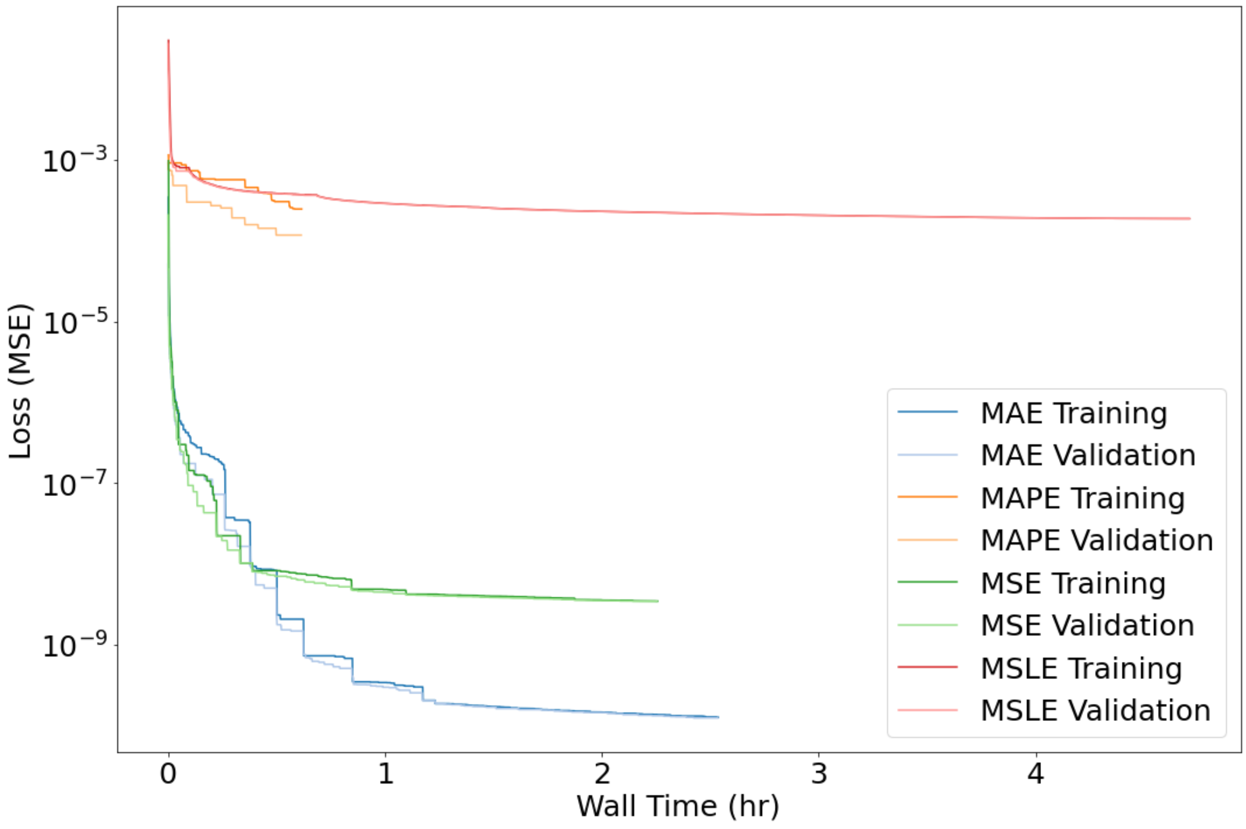

Figure 66: Loss rate graph, shown in MSE, with the loss functions MAE, MAPE, MSE, and MSLE used for training.

Table 58: Final loss values in MSE and wall time for training with the loss functions MAE, MAPE, MSE, and MSLE.

Loss function	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
MAE	1.26e-10	1.24e-10	2:32:13
MAPE	2.46e-04	1.17e-04	0:36:59
MSE	3.45e-09	3.44e-09	2:15:25
MSLE	1.87e-04	1.87e-04	4:42:31

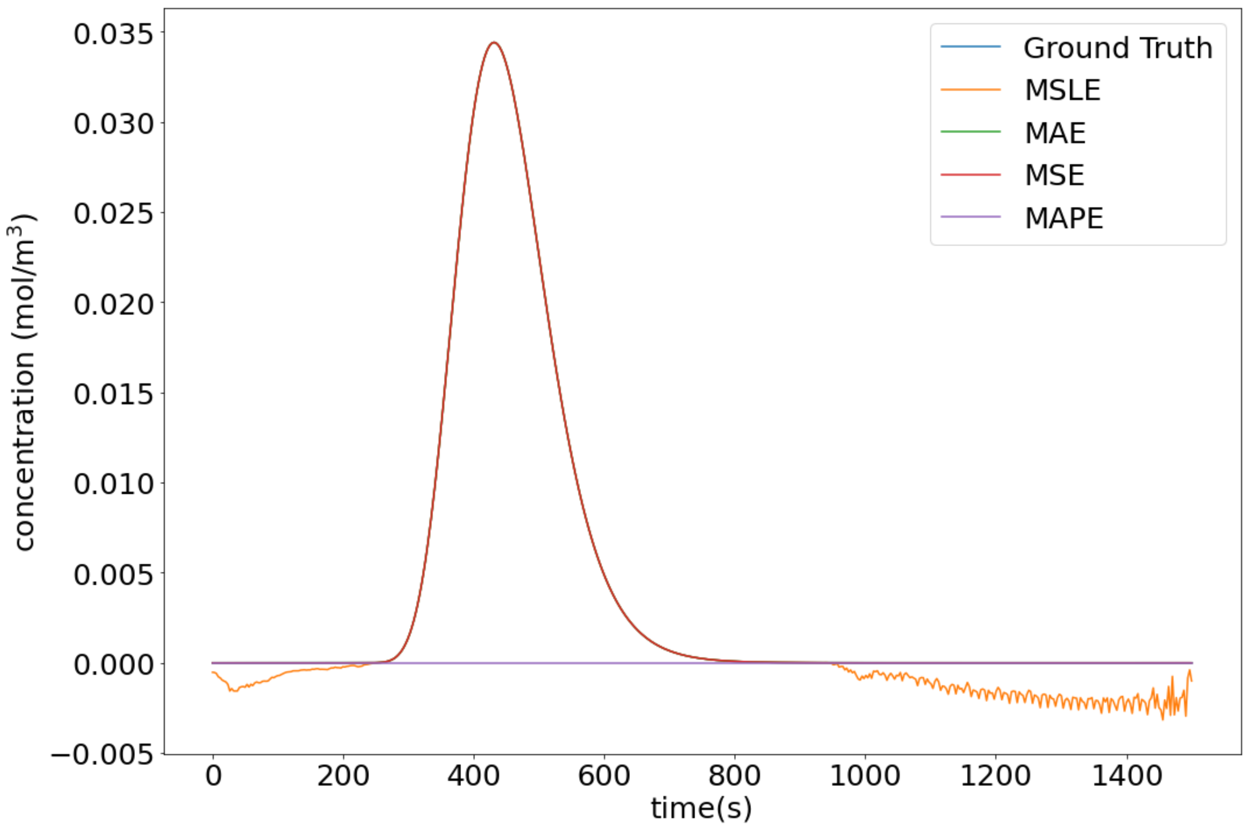

Figure 67: Example chromatogram prediction when trained with MSLE, MAE, MSE, and MAPE to illustrate the visual difference.

4.3.1.6. Network number of kernels

The next hyperparameter to optimize is the maximum number of kernels in the convolution layer in the network from Code 3. While it is possible to give every layer an number of kernels that introduces too many variables. The results are shown in Figure 68 with the final loss shown in Table 59. While setting the number of kernels to 512 results in a slightly lower loss it also causes the training and validation losses to separate from each other which the previous hyperparameter changes had removed. As such 256 filters is the best option to use here due to having a very similar loss but without the separation. Smaller filter sizes of 128 and 64 don't show any separation but they also have a higher loss. A filter size of 256 is selected moving forward.

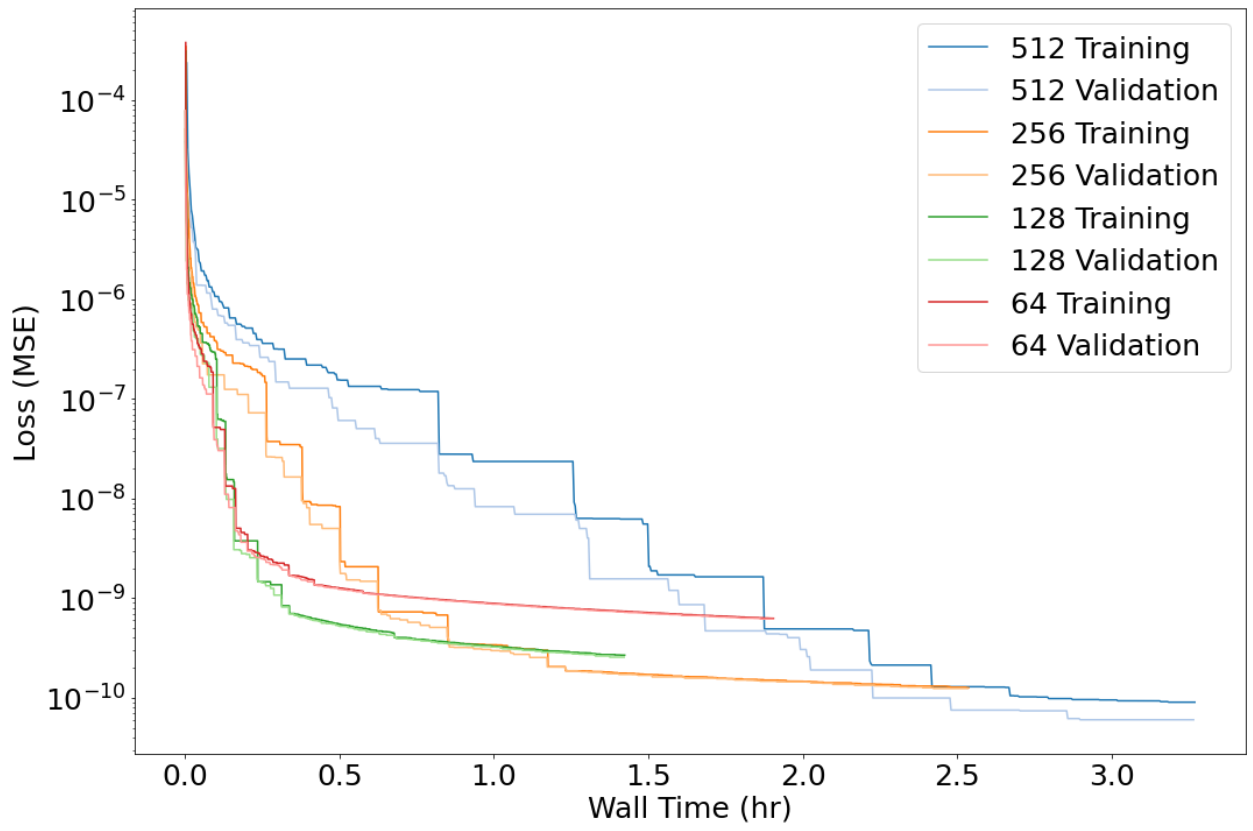

Figure 68: Loss rate graph for maximum filters from 64 to 512.

Table 59: Final loss values in MSE and wall time maximum filter size from 64 to 512.

filters	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
512	9.02e-11	6.01e-11	3:16:11
256	1.26e-10	1.24e-10	2:32:13
128	2.67e-10	2.55e-10	1:25:25
64	6.26e-10	6.17e-10	1:54:19

4.3.1.7. Network kernel size

The final hyperparameter to set is the kernel size used in the convolution layers from Code 3. The kernel size is varied from 4 to 16 and shown in Figure 69 with the final tabulated values in Table 60. In this case kernels 4,8, and 12 don't show any separation between training and validation. A kernel size of 8 and 12 have almost identical training and validation loss. A kernel size of 16 has the same problem as the previous hyperparameter where it re-introduces a separation between training and validation. That is not desirable as the final hyperparameter to optimize. A kernel size of 8 is chosen going forward. A kernel size of 8 or 12 would have been equally acceptable from a loss standpoint so the decision goes to a kernel size of 8 for running faster.

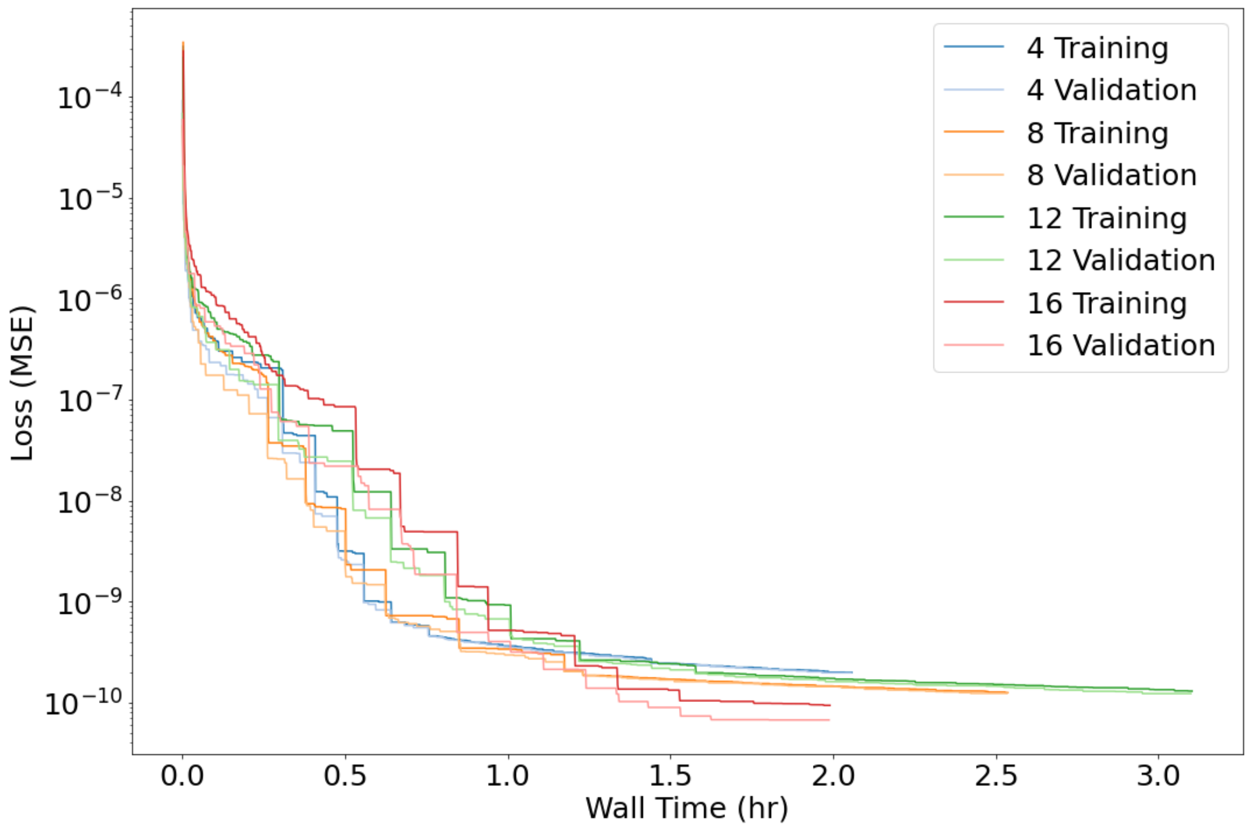

Figure 69: Loss rate graph for convolutional kernels from 4 to 16.

Table 60: Final loss values in MSE and wall time for kernels from 4 to 16.

kernels	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
4	2.00e-10	1.98e-10	2:03:28
8	1.26e-10	1.24e-10	2:32:13
12	1.30e-10	1.23e-10	3:06:14
16	9.42e-11	6.75e-11	1:59:27

4.3.2. Final Network Design with Hyperparameters

With all the hyperparameter optimization completed the set of parameters is shown in Table 61 and the final network shown in Code 4. The changes to the loss function and learning rate had the largest impacts on the final error. The changes in the batch size and learning rate had the largest impact on reducing the variance between training and validation curves. Figure 70 shows the training curve for the original network design along with the optimized version. There is no sign of overfitting in the optimized version and there is very little variance between the training and validation data. A few selected examples are shown in Figure 71 that look at the original network vs the optimized network. These items were selected to show typical differences between the networks. Subplot A shows a combination of parameters where the protein is bound strongly to the column and while the optimized version almost exactly follows the ground truth there is significant deviation for the original version. Subplot B is much closer to a more typical prediction and while the prediction for the optimized version is close to the ground truth the original version has a significant miss on the height, which can be seen in the inset. While the height difference does not appear large it is large enough to be a problem with uncertainty quantification. Subplot C shows a similar story with the peak height wrong, and the back part of the peak also has a lot more noise in it than the optimized version. Subplot D is typical of atypical shapes where the optimized version has a much better fit.

While this shape is inside the search range it is not one that would be encountered during parameter estimation or uncertainty quantification as a correct or near correct fit. What these plots illustrate is that the optimized network performs better over the entire validation space.

To verify that this network is good enough for parameter estimation and uncertainty quantification two additional MCMC experiments were performed with the absolute tolerance for CADET changed from 1e-8 to 1.5e-7 to simulate the same loss the network has based on an equivalent MSE. The resulting parameter distributions with increased absolute tolerance are almost identical to the results shown in chapter 3.3.1.4. This means that using the network is likely to result in the same parameter distributions as using CADET directly and thus a good surrogate model.

Table 61: Final network hyperparameters.

Parameter	*Value*
Loss	Mean Absolute Error
Optimizer	ADAM
Batch Size	192
filters	256
kernel	8
Activation function	Swish
Learning Rate	Adaptive 1e-3 to 1e-5

Code 4 Final inverted pyramid design where the only variable is the input dimension

```
z_in = Input(shape=(input_dimension,))
z1 = Dense(256)(z_in)
z1 = swish (z1)
z2 = Dense(32 * 256)(z1)
z2 = swish(z2)
z3 = Reshape ((32, 256))(z2)
z4 = Conv1DTranspose (filter = 128, kernel_size = 8, strides=2)(z3)
z4 = swish (z4)
z5 = Conv1Dtranspose(filter = 64, kernel_size = 8, strides=2)(z4)
z5 = swish (z5)
z6 = Conv1Dtranspose(filter = 32, kernel_size = 8, strides=2)(z5)
z6 = swish (z6)
z7 = Conv1Dtranspose(filter = 1, kernel_size = 8, strides=2)(z6)
z7 = swish (z7)
z_out = Reshape((512,))(z7)

model = Model(z_in, z_out)
```

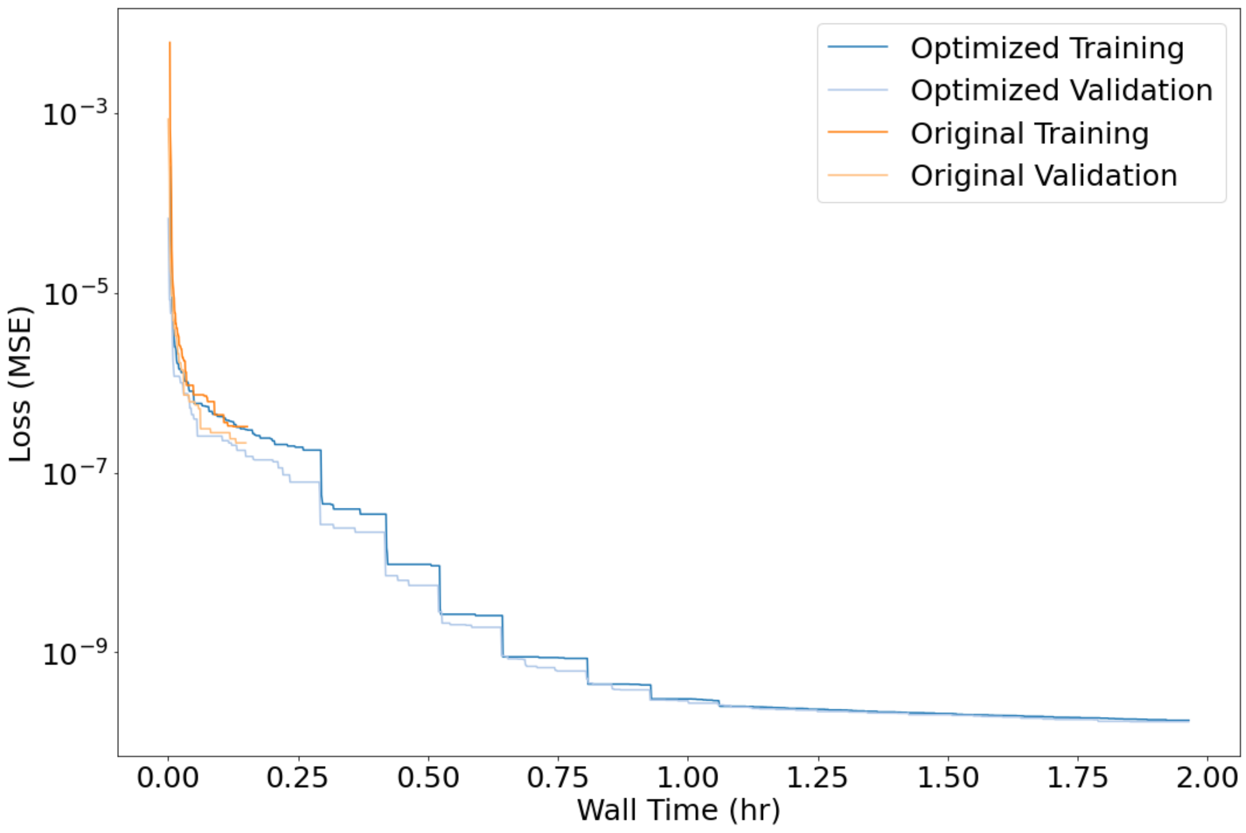

Figure 70: Original and optimized networks training and validation curves.

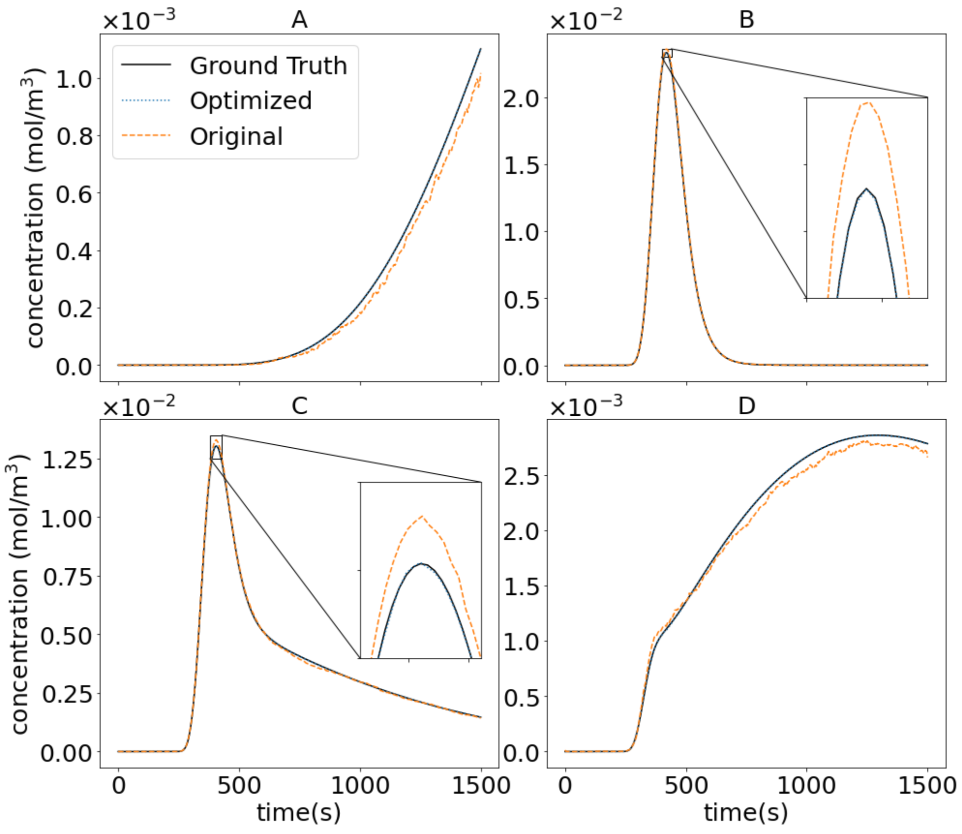

Figure 71: Four selected examples comping original vs tuned neural networks.

4.3.2.1. How large of a data set is needed?

With all the tuning complete and the final network determined the next question to ask is how much data is needed. Training with more data is essentially always better but it also increases training times and has diminishing returns on network loss such that each additional amount of data added linearly increases the training data but decreases the loss less. Since this network is supposed to operate as a surrogate what we want to know is what is the realistic minimum amount of data needed. Figure 72 shows the loss rate at data set sizes range from 10K to 500K total samples with the final values shown in Table 62. With only 10K total samples the network performs quite well but it also shows overfitting with the training loss lower than the validation loss. This indicates the network is likely memorizing the training data set. As the total samples increases to 25K and 50K this effect diminishes and is effectively negligible at 50K total samples. At 100K samples there is no sign of overfitting anymore and there is very little separation between training and validation loss. At 500K the situation is similar to 100K with no sign of overfitting or separation between training and validation. Figure 73 shows the loss as a function of total samples and somewhere around 50K to 100K samples adding more samples is having diminishing returns on the loss. This makes 100K total samples a good choice to use as tradeoff between loss and training time for this kind of problem. Based on the MCMC case studies in section 0 on page 73 it is expected that this surrogate network would provide substantial savings on problem with 7 or more parameters with less than 1/10th of the simulations needing to be run as required now. Running the training on a more powerful GPU would also substantially reduce the training time. For parameter estimation the situation is a little more complicated and generally the network will only speed up systems that take more than a few days to run. Parameter estimation is done over wider parameter ranges, and this will also need more total samples. So long as the loss is low enough a network trained for parameter estimation can also be used for uncertainty quantification and additional simulations are not needed.

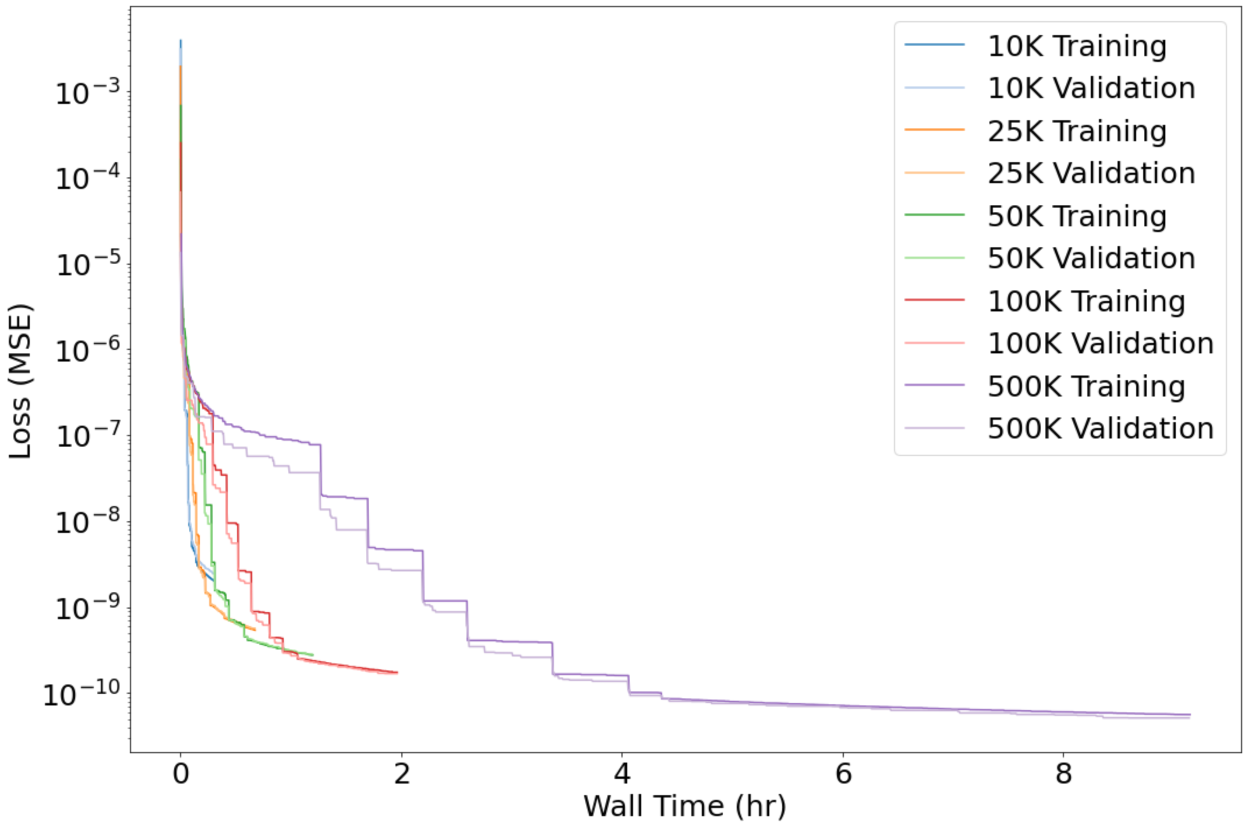

Figure 72: Loss rate graph for total samples from 25K to 500K.

Table 62: Final loss values in MSE and wall time for total samples from 25K to 500K.

Total Samples	*Minimum Training*	*Minimum Validation*	*Wall Time (h:m:s)*
10K	2.02e-09	2.46e-09	0:18:20
25K	5.38e-10	5.66e-10	0:40:38
50K	2.76e-10	2.83e-10	1:11:54
100K	1.74e-10	1.68e-10	1:57:51
500K	5.62e-11	5.13e-11	9:08:54

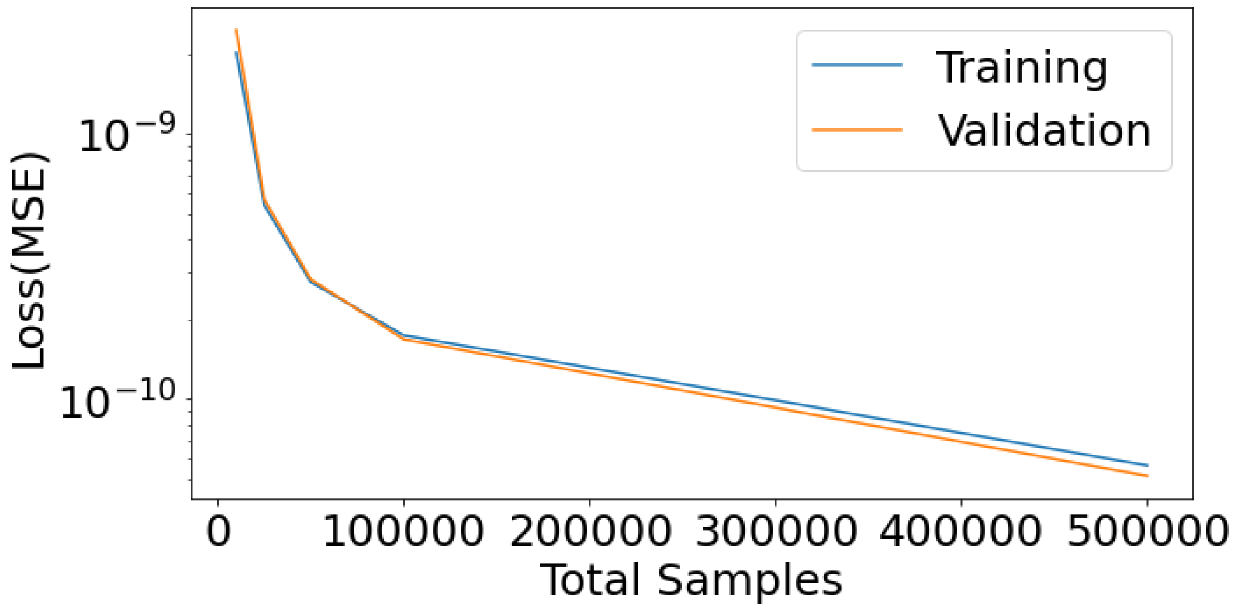

Figure 73: Total samples vs loss shown in a log plot.

4.4. Case Studies

All case studies are taken from the synthetic MCMC case studies in section 3.3.1 on page 76. 100K samples are generated for each case study and the setup is identical to the settings used for hyperparameter tuning with the final network design and hyperparameters used. The sampling bounds for each case study comes from the corresponding study in chapter 3.3.1 after the bounds adjustment step from chapter 3.2.2.

4.4.1. Dextran Pulse with Detached Column

A Dextran pulse with detached column from section 3.3.1.1 on page 76 is the first case study. Table 63 shows the sampling bounds for the 3 parameters. The training and validation loss are shown in Figure 74. From the loss graph there is no sign of overfitting or separation between the training and validation curves. The results are shown in Table 64 with a final validation loss of 1.5e-10 which took about 2 hours to reach. In terms of speed the network could make 83,700 predictions per second while CADET only yielded 43.5 simulations per second. This results in the neural network running 1900 times faster than using the CADET simulator. Even with this speed advantage using a neural network for this problem would be a large loss due to MCMC taking less than 30 minutes to run using the simulator and requiring around 75,000 simulations in total. A neural network for this step would only make sense if the same network could be reused with the same parameter range and experimental setup. However, over time this balance could change as hardware changes and the requirements on the error model changes. This case clearly illustrates the flexibility of the tuned network on being trained on a different dataset with a different number of parameters from the one used to tune the model.

Table 63: Dextran Pulse with Detached Column parameter ranges for sampling.

Name	*LB*	*UB*
Tube Dispersion	1.0e-7	1.0e-5
Tube Cross Section Area	1.0e-8	1.0e-4
CSTR Volume	1.0e-7	1.0e-5

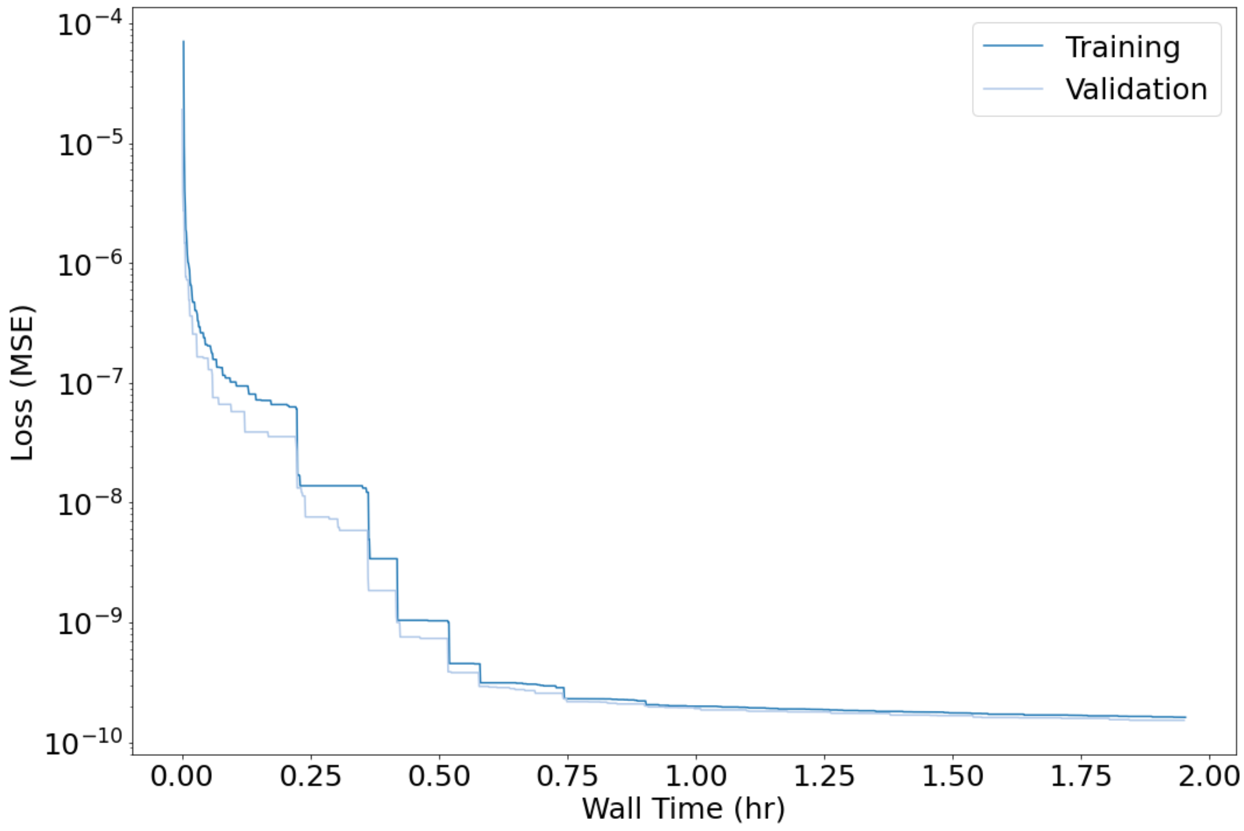

Figure 74: Dextran Pulse with Detached Column loss graph.

Table 64: Dextran Pulse with Detached Column summary statistics.

Training Loss	1.62e-10
Validation Loss	1.53e-10
Training Time	1:57
Predictions/second	83700
Simulations/second	43.5
Times Faster	1900

4.4.2. Non-Pore Penetrating Pulse

The next case is a non-pore penetrating pulse from section 3.3.1.2 on page 78. The sampling bounds are shown in Table 65 and has been expanded to 5 parameters. The training and validation loss graph is show in Figure 75. As before there is no sign of overfitting or separation between training and validation. The summary statistics are shown in Table 66 with a final validation loss of 4.6e-9 after 1.5 hours of training. The network can make 83100 predictions per second, which is a slight decrease, but the CADET simulation is down to 3.3 simulations per second. This results in the network running 25,000 times faster. This network required a little less than half the simulations that the MCMC run required and runs much faster. It is likely that running MCMC on the neural network and running MCMC on the CADET simulator directly would take close to the same amount of time to run when considering training time. However, as hardware changes, this is likely to move in favor of the neural network given current hardware trends. This case ads further evidence to this tuned network being generally applicable for chromatography modeling.

Table 65: Non-Pore Penetrating Pulse parameter ranges for sampling.

Name	*LB*	*UB*
Column Dispersion	1.0e-12	1.0e-5
Column Porosity	0.2	0.7
Tube Dispersion	1.39e-6	4.01e-6
Tube Cross Section Area	1.68e-5	2.15e-5
CSTR Volume	3.70e-6	4.30e-6

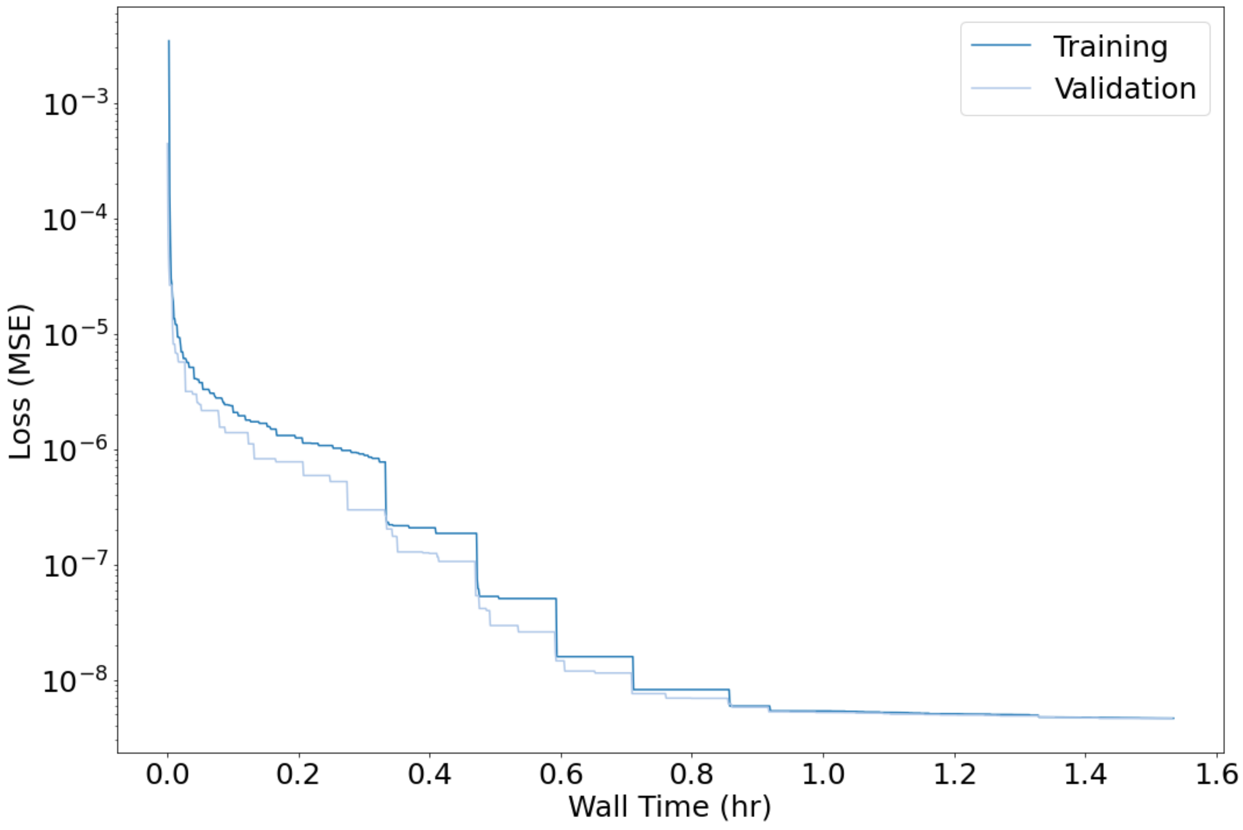

Figure 75: Non-Pore Penetrating Pulse loss graph.

Table 66: Non-Pore Penetrating Pulse summary statistics.

Training Loss	4.65e-09
Validation Loss	4.63e-09
Training Time	1:32
Predictions/second	83100
Simulations/second	3.3
Times Faster	25200

4.4.3. Non-Binding Protein Pulse

The third case is a non-binding protein pulse from section 3.3.1.3 on page 80. The number of parameters has increased to 8 and the sampling bounds are shown in Table 67. Training and loss validation curves are shown in Figure 76 and as before there is no sign of overfitting or a separation between training and validation curves. The summary statistics are shown in Table 68. The final validation loss is 6.7e-10 after 2 hours of training. The network makes 82,700 predictions per second which is a 0.5% slowdown from the previous case while CADET generates 2.1 simulations per second. The network is now 39,400 times faster than using the simulator. The MCMC case in section 3.3.1.3 required about 1 million simulations and 16 hours to run. The neural network is clearly better at this point requiring 10% of the simulations, taking only 2 hours to train, and running much faster. As hardware continues to improve it is likely that the neural network will continue to increase in speed relative to running CADET directly. The tuned network has also proven to be remarkably adaptable with solid performance in 3 different cases so far with different numbers of parameters.

Table 67: Non-Binding Protein Pulse parameter ranges for sampling.

Name	*LB*	*UB*
Film Diffusion	1.0e-9	1.0e-5
Particle Porosity	0.2	0.5
Particle Diffusion	1.0e-14	1.0e-6
Column Dispersion	2.60e-7	3.48e-7
Column Porosity	0.34	0.36
Tube Dispersion	1.39e-6	4.01e-6
Tube Cross Section Area	1.68e-5	2.15e-5
CSTR Volume	3.70e-6	4.30e-6

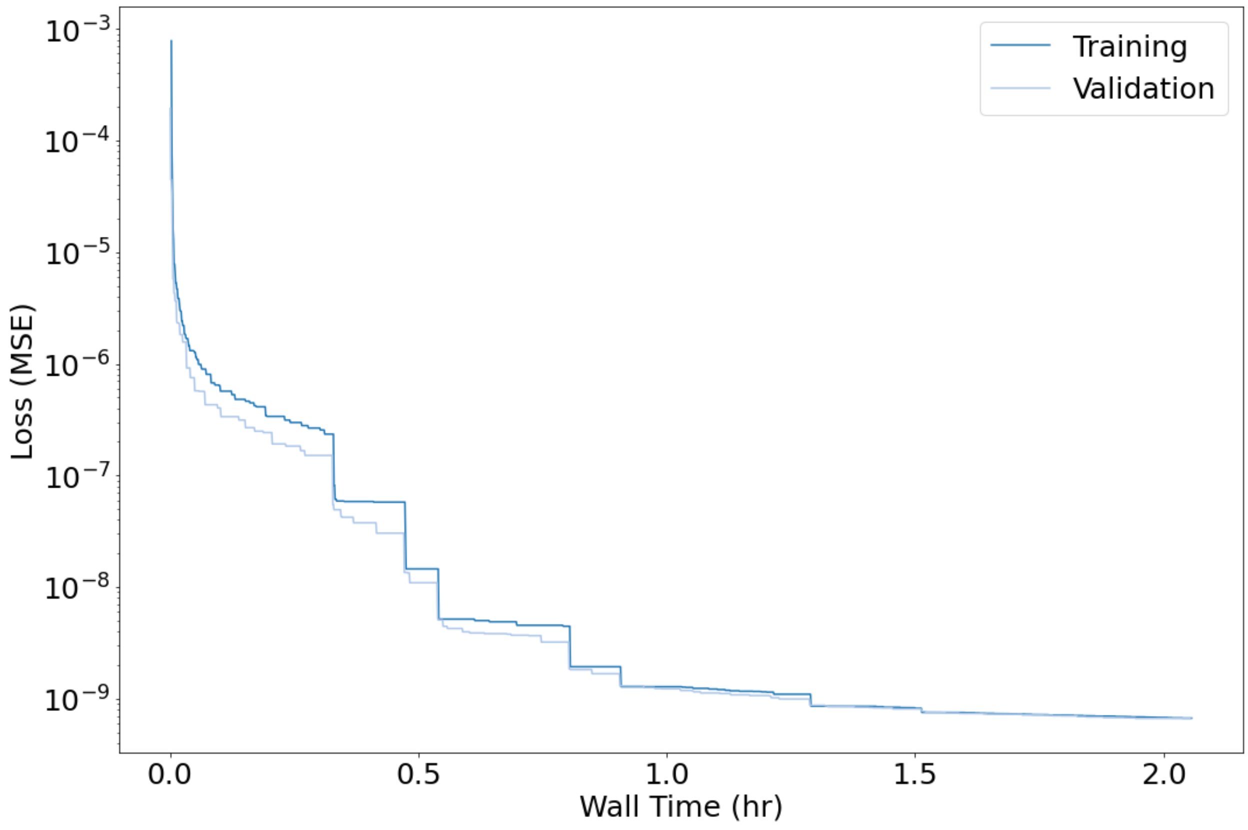

Figure 76: Non-Binding Protein Pulse loss graph.

Table 68: Non-Binding Protein Pulse summary statistics.

Training Loss	6.69e-10
Validation Loss	6.65e-10
Training Time	2:03
Predictions/second	82700
Simulations/second	2.1

Times Faster	39,400

4.4.4. Binding Protein Pulse

The final case study is a binding protein pulse from section 3.3.1.4 on page 83. The number of parameters has increased to 10 and the ranges are shown in Table 69. The training and validation loss curves are shown in Figure 77 and as before there is no sign of overfitting or separation between training and validation. The final statistics are show in Table 70 with a final validation loss of 1.7e-10 after 2 hours. The network making 83,700 predictions per second while CADET is down to 0.8 simulations per second. The network is now 105,00 times faster than running CADET. The MCMC case from section 3.3.1.4 took between 2 and 6 days to run and between 1.5 and 4 million simulations. This means that the network requires between 3% and 7% of the simulations and only a few hours to train. The network is clearly faster and as the cases have grown more complicated network performance has been essentially constant.

Table 69: Binding Protein Pulse parameter ranges for sampling.

Name	*LB*	*UB*
kA	1.0e-8	1.0e8
kEQ	1.0e-4	1.0e-4
Film Diffusion	7.88e-7	4.32e-6
Particle Porosity	0.31	0.35
Particle Diffusion	1.74e-11	3.59e-11
Column Dispersion	2.60e-7	3.48e-7
Column Porosity	0.34	0.36
Tube Dispersion	1.39e-6	4.01e-6
Tube Cross Section Area	1.68e-5	2.15e-5
CSTR Volume	3.70e-6	4.30e-6

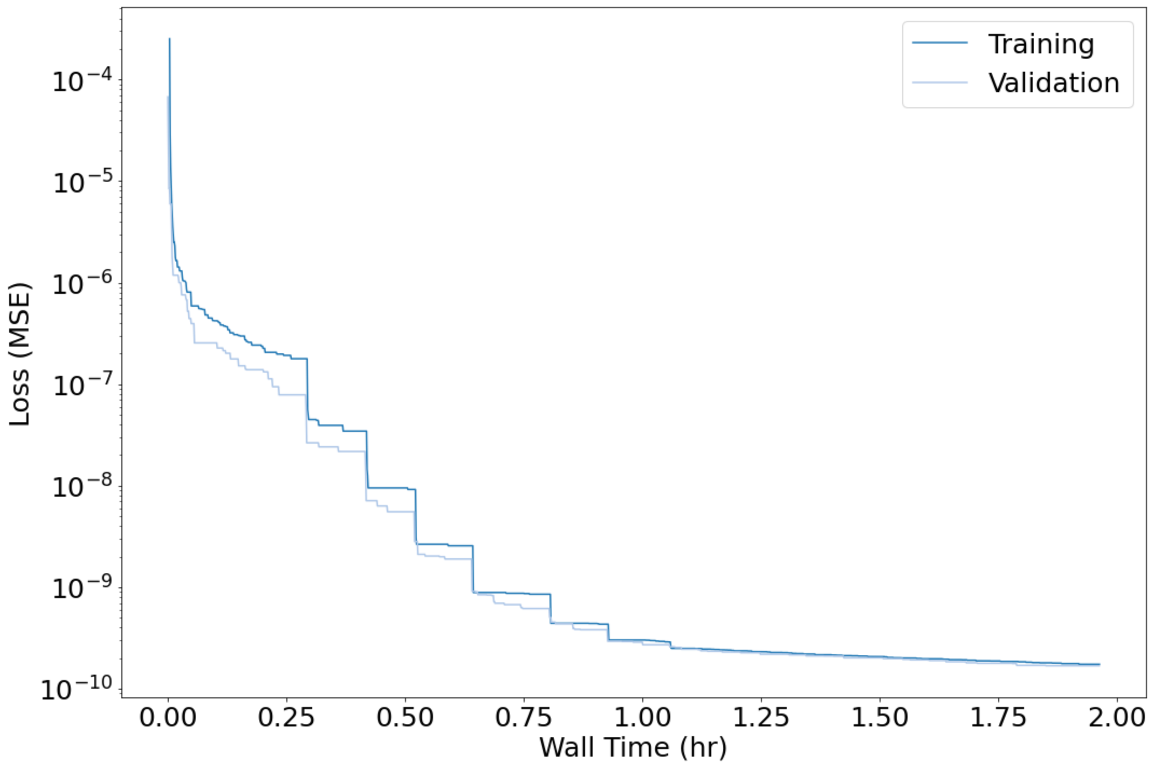

Figure 77: Binding Protein Pulse loss graph.

Table 70: Binding Protein Pulse summary statistics.

Training Loss	1.74e-10
Validation Loss	1.68e-10
Training Time	1:57
Predictions/second	83700
Simulations/second	0.8
Times Faster	105,000

4.5. Summary

A surrogate model has been demonstrated using a neural network capable of approximating the cases presented here. The cases covered a bypass experiment, non-pore penetrating pulse, a pore penetrating but non-binding pulse, and a binding pulse experiment from section 3.3.1. All cases tested here involved training the network to predict the full chromatogram given a few parameters, such as axial dispersion, column porosity, and particle porosity, and did not look inlet profiles. The network used should be able to handle many more parameters or even having the inlet profile as a parameter but, that has not been tested and would likely require a lot more data for training the network. The importance of hyperparameter optimization has also been demonstrated, by improving the loss by 3 orders of magnitude, along with an explanation of the steps and why they are important. Hyperparameter optimization covered the choice of loss function, optimizer, learning rate, batch size, and network parameters. While there are automated tools for network tuning, like KerasTuner [97], they can be difficult to use correctly without understanding the manual tuning process first. There are also cases where automatic tuning only partially works, and manual tuning may be required for some hyperparameters which makes understanding the manual process important. Using a neural network is fast and robust and takes advantage of developments in GPU and machine learning accelerators such as the Mi200 [98] from AMD. The network can make thousands of predictions in parallel very quickly. The resulting tuned network worked across a variety of cases under the same conditions needed for parameter estimation and uncertainty quantification. The resulting network is tens of thousands of times faster than running CADET directly.

5. Conclusions

As stated at the beginning of this thesis to make medicine safer regulations have moved to Quality by Design. Quality by Design focuses on designing quality into a product by understanding the product and process along with the risks involved in manufacturing the product and the mitigation of those risks. The need to understand the product and the manufacturing process led to the development of digital twins. However, developing digital twins is complicated due to the difficulty of creating and calibrating them. The focus of this thesis has been on solving the calibration part of the digital twin's problem using parameter estimation and uncertainty quantification with surrogate modeling as support.

Throughout this thesis synthetic and experimental case studies have been used. The synthetic case studies, where applicable, are designed to be similar to experimental case studies for protein purification for packed-bed liquid chromatography. None of the synthetic cases have been designed to be easier or harder for parameter estimation or uncertainty quantification. Bypass experiments are used to characterize mixing and dispersion behavior external to the column. Binding but non-pore penetrating experiments are designed to characterize the column porosity and axial dispersion. Pore-penetrating but non-binding experiments are designed to characterize the resin properties such as film diffusion, pore diffusion, and particle porosity. Finally, binding experiments are done to evaluate the binding parameters, and this can be as simple as a linear isotherm or something more complex such as multi-component SMA binding. The experimental data presented in this thesis has been provide by Amgen and detailed in section 1.6. These synthetic and experimental case studies are used for parameter estimation, uncertainty quantification, and surrogate modeling.

Parameter estimation is the foundation of the entire thesis. A novel goal system was introduced that is composed of scalar metrics that quantifies different aspects of a chromatogram such as the shape, height, and time of the peak to serve as multi-objective optimization target. These metrics were able to improve convergence on synthetic and experimental data and have been used on other experimental data sets outside of this thesis successfully [45], [99]. These metrics stand in contrast to the de facto standard approach of using a least squares single-objective gradient descent optimizer with SSE. The multi-objective approach presented here has also proven to be revealing for experimental data in exposing deficiencies in the model or data via multiple Pareto optimal values. Visualizing the Pareto front made it clear where tradeoffs were occurring in the parameters when attempting to fit the model to the data. This has been useful in finding errors in the data and deficiencies in the model. For example, peaks might be slightly shifted by pump delays or small changes in the elution buffer that are unknown and hence not covered by the chromatography model. Least squares penalize position offsets much stronger than peak shape variations and would consequently lead to large errors in the estimated parameters of the binding model. The proposed new score system is designed to handle such situations better by allowing tradeoffs between conflicting objectives. It has been found that the mean of involved metrics provides a better measure for the quality of the match between simulation and experiment than the least squares residual. Multiple Pareto optima can also be post-processed by weighting different metrics or experiments to select an optimum appropriate to the problem at hand. The advantage of handling this in post-processing is that it does not require a priori selection of weights.

Another advantage of the new metrics is that multiple experiments and fractionation data can be integrated without needing to weight different objectives. While the objectives can be combined into one for a gradient search algorithm, a multi-objective GA has been observed to be much more robust for complex problems. Pareto optima can indicate inconsistencies between multiple experiments, and the Pareto front carries rich information on potential causes of failure, as demonstrated in the experimental case study. This information is important for understanding the system and designing better experiments. The genetic algorithm as well as the multi-start strategy for gradient search are parallelized with progress monitoring for computational efficiency on compute clusters but also on multi-core processors in personal computers. Due to the modular nature of the metrics, new metrics can be merged into the existing framework without changing the search algorithm. This can be particularly useful for real-world and large-scale industrial data where increasing system complexity might require consideration of additional features. The current metrics have already been shown to properly address non-ideal tracer retention and pump delays.

Once parameter estimation has been completed the next step is uncertainty quantification. In CADET-Match this process is even automated such that parameter estimation is run first and then uncertainty quantification. A complete process has been demonstrated for modeling errors, obtaining parameter distributions, and the uncertainty on the resulting chromatogram(s). The process covers construction of the error model, sampling, and convergence testing. The most commonly used error models assume all errors are random, independent, and normally distributed and these assumptions are often inaccurate for chromatography. Instead, a method was proposed here for construct an error model out of physical error sources such as pump delays, pump flow rates, loading concentration, and UV detector noise with easy extensibility to additional sources of error. The resulting error model has been demonstrated on synthetic and experimental data. The entire process is based on Bayesian uncertainty analysis and implemented with MCMC. The Bayesian uncertainty analysis presented here uses a stagewise approach where the stages are chosen to isolate some parameters from others such as running a bypass experiment to evaluate extra column effects without the impact of the column. Bayesian uncertainty analysis allows parameter uncertainty to be carried over from one stage to the next while MCMC allows incremental refinement of the probability distribution within a single stage until convergence.

Additional stages can be added as needed to further refine parameters. The posterior chromatograms resulting from the posterior probability distribution of parameters also provide information to further understand the studied systems. For reporting results the highest density interval (HDI) is used. The HDI is the smallest interval that contains some defined amount of the probability distribution, such as 90%, such that no point outside the interval has a higher probability that a point inside. The HDI is then visualized as a confidence tube which is a plotting of chromatograms sampled from the HDI and colored according to their probability. In the synthetic binding example, the posterior chromatograms show narrow confidence tubes which indicates that despite some of the diffusion parameters having a wide HDI their effect is minimal over that interval. In the experimental non-binding pulse example, shown in section 3.3.2.3 on pg. 91, the posterior chromatograms don’t capture the very front or tail of the peak. The front of the peak comes sooner and with a shallower rise and the tail of the peak comes later and is also shallower than the model and completely outside confidence tube indicating that no combination of parameters estimated captures the observed behavior. This provides critical information that the model is missing some mechanism that is present in the system and not covered by any of the modeled error sources. The posterior probably distribution of parameters can also be used to determine where improvements are needed in the modeling and experimental process. For example, experiments can be added or changed, and the posteriors compared for all parameters. These error models can be used to find out what types of measurements are needed and how precise they need to be to obtain the desired results.

This thesis also demonstrates several ways of using MCMC that should be considered best practices. While many types of MCMC exist, the one explained in this thesis is an affine invariant ensemble sampler, they all have methods to measure convergence and appropriate burn-in and these methods should be used. MCMC creates a dependent chain between samples, but the quality of the posterior distribution is based on the number of independent samples. The integrated autocorrelation time (IAT) measures how many dependent steps must be taken before a new independent sample is generated. Over the course of this thesis an IAT as low as 7 and as high as 600 have been encountered. To generate enough samples, the chain length must be varied based on the IAT and a static chain length is not appropriate. With an ensemble sampler multiple non-independent chains can be run in parallel, and each chain is called a walker. The walkers are moved cooperatively to improve convergence. The choice of where to start the walkers is also important due to the problem of dimensionality. As the numbers of parameters increase it becomes critically important to start off the walkers in high probability regions and a method for doing that has been demonstrated in this paper. Beyond just a few parameters and without good starting points most of the walkers will end up far away from the main distribution and remain effectively stuck in low probability space and this will distort the posterior distribution.

While uncertainty quantification was demonstrated to work it also demonstrated a limitation due to the amount of computing power required. The number of simulations increases exponentially with the number of parameters and given how long more complex simulations take to run uncertainty quantification quickly increases in time from days for single component linear binding to years for multi-component SMA binding. This is clearly a problem and one that is not going to be solved by just using more powerful servers. While there are new CPUs coming out with hundreds of cores that is not enough to bring the calculation time down to a reasonable time of a few days. A fundamentally different approach is needed to solve this scaling problem and the approach used here is a surrogate model using a neural network to approximate CADET. While CADET must solve a complex PDE on a CPU a neural network can approximate the final solution on a GPU which makes it tens of thousands of times faster and make thousands of predictions in parallel even on a laptop GPU. While new CPUs are coming out with hundreds of cores that won't close the gap as GPUs are also continuing to get faster and dedicated machine learning hardware is even faster than GPUs.

An ANN as a surrogate model for CADET has been demonstrated. The final neural network presented is fast and accurate on all test cases and due to the universal approximation theorem and chromatograms being continuous it should work on any output CADET can generate given sufficient data. The hyperparameter optimization process has been explained and while automated hyperparameter tuners like KerasTuner [97] exist they require understanding of the manual process to be used effectively. Hyperparameter tuning has been shown to be critically important with a 3 order of magnitude reduction in the loss function by tuning the activation function, optimization, learning rate, batch size, and network parameters.

The tuned network had a normalized MSE loss between 5e-9 and 1e-10 for all test problems. Additional MCMC tests where run on the synthetic binding case from chapter 3.3.1.4 with the absolute tolerance changed from 1e-8 to 1.5e-7 to replicate the higher MSE of the surrogate model it had less than a 3% impact on the resulting parameter distributions which indicates the surrogate models should be good enough for parameter estimation and uncertainty quantification. While it would have been ideal run MCMC on the surrogate model that is currently infeasible, and this issue is further covered in the outlook. Based on how quickly the surrogate model runs and that in testing the time is essentially independent of the number of parameters MCMC will complete on the surrogate model in a few hours for even complex cases like 2-component SMA binding although collecting enough data to train the network will still take several days.

The software package CADET-Match is the primary result of this thesis. It is a freely available at https://github.com/modsim/CADET-Match and released under the GPLv3 free software license with full documentation available at https://cadet.github.io/CADET-Match. CADET-Match implements parameter estimation, uncertainty quantification, and data smoothing shown in this thesis along with many other features that have not been shown in this thesis. CADET-Match supports parameter transformations to solve specific industrial problems such as translations related to tubing diameter, volume, and area along with a transformation designed to copy a value to another parameter without increasing the dimensionality of the problem. CADET-Match also contains many more scores than have been presented here such as scores designed for different types of breakthrough curves or a score designed to not allow a peak in a specific region based on an industrial problem. CADET-Match also contains other types of algorithms that are not designed for parameter estimation or uncertainty quantification such as one designed for visualizing a parameter range. When first developing a new model there is often little information on what parameter ranges need to be searched for parameter estimation. An initial step can is to evaluate many samples taken from a very wide range for all parameters. The results of the sampling can then be evaluated to determine where optimization should occur. This has been critical in developing many industrial models and quickly identifying problems with the model. CADET-Match also logs the version of every library it uses, the configuration, and simulation files so that simulation results are self-contained and can be rerun later with exactly the settings used to rerun them. The software also generates hundreds of graphs designed to measure progress and quickly see if there are errors with the scores chosen and to determine if the problem has a well-defined optimum. It has a built-in feature to generate a test suite and example of every feature it has and run them to verify everything is functioning correctly and as an example. The software also uses a simple plugin system making it easy to add in new parameter transforms, scores, and search algorithms by just dropping a file in a folder without having to touch any of the rest of the code. CADET-Match is written using Python and makes extensive use of standard Python libraries. The software is designed to be mostly automatic and robust, and provided a model, data to fit to, and a configuration file on what operations are needed it can perform parameter estimation and continue with uncertainty quantification.

CADET-Match has been designed as a monolithic tool with minimal human intervention. Hence, pragmatic choices were made for some meta-parameters such as approximation order in the smoothing procedure, concentration thresholds for time interval selection, or solver tolerances in the chromatography simulation. Countless numerical tests were performed, and much care has been taken to ensure robustness of the presented algorithm on real-life industrial data. Even though the meta-parameters can be changed, due to the open-source nature of the code, this is not recommended unless the implications are fully understood. The open code provides full transparency of the applied procedures and allows to adapt and integrate the software in operational workflows. CADET-Match can be used with all model variants that are available in the parent project, CADET, and covers a wide range of transport and binding models.

This thesis has achieved all objectives and raised several new issues to explore. CADET-Match is robust and automatic and capable of parameter estimation and uncertainty quantification on a range of synthetic and industrial datasets. It can incorporate experimental error sources and aid in the understanding of the errors along with exploring how important different error sources are to control. It also provides an extensible framework to address future problems.

5.1. Outlook

Over the course of this project while many advancements have been made there have also been areas found where more work is needed in terms of either research or software development. It has been shown that a smooth signal is important to how the metrics shown work and that cleaning up an experimental signal can be difficult. More work needs to be done on either improving the smoothing or modifying the metrics to be less dependent on a smooth signal. There has been developments in machine learning for removing noise from signals using stacked autoencoders that is worth looking into [100].

Work also needs to be done to integrate the surrogate model into CADET-Match. The CADET-Match software currently uses multiprocessing for parallelization however neural networks work best with a single process and run on a GPU or dedicated AI hardware. This requires changes in how the master process calculates and hands out work to the parallel processes and adds additional overhead compared to how the current code works. These changes are not trivial to make and went beyond what could be done in this thesis. There are also changes being made in the Python programming language that may make these changes easier to make in the future related to changes in parallelization. This is work I will continue to be involved with into the future.

The surrogate model is another area where further research is needed. Additional types of networks should be tested on a wider range of datasets. It is also worthwhile to look at automatic hyperparameter tuning tools like KerasTuner [97] to improve the networks. The networks trained here where also all trained in very narrow ranges and an investigation should done on how wide the ranges can be and now much data is then needed. The surrogate model also needs to be tested with different boundary and initial conditions such as changing the inlet profile and may need changes as a result. Just developing the surrogate model is likely to be a large part of a PhD and something that is likely to be very worthwhile.

While the surrogate models shown here are fast and will continue to get faster as hardware develops, they still require data from the CADET simulator for training which is CPU limited. Right now, CADET uses finite volume for discretization. Another possibility would be to use discontinuous Galerkin (DG) methods. DG methods are a kind of hybrid between finite element and finite volume approaches and have the potential to be 10 to 100 times faster than CADET's current method. With a speed gain this large many problems that would need a surrogate would be feasible to solve in higher precision using CADET directly. DG would also make creating the training data for surrogate modeling significantly faster. There is currently ongoing work into this subject and the early results are very promising but extensive testing and integration is still needed.

Looking forward the technology in this thesis will continue to be developed. While CADET-Match is already used in industrial applications the technology developed for it is also being used academically and in industrially. The smoothing filter has been applied to radioactive tracers in plant science. The neural network and uncertainty quantification technologies here have been used in photovoltaics. Other aspects of CADET-Match area also in the process of being integrated into other types of industrial modeling beyond Chromatography.

6. Acknowledgements

I would like to thank Amgen for funding my PhD and providing data and assistance throughout my entire thesis. I would also like to thank Amgen for helping me to come to Germany in the first place to get a master's degree. I would also like to think Prof. Dr. Wolfgang Wiechert and Dr. Eric von Lieres for making it possible to do my PhD at IGB-1 at Forschungszentrum Jülich.

I would like to especially thank my supervisor, Dr. Eric von Lieres, who has helped me throughout this project. I especially appreciate all the help with editing my papers and helping work through the complexities of the project.

I would also like to thank Prof Dr. Andreas Jupke and Prof. Dr. Wolfgang Wiechert for refereeing my thesis.

I especially want to thank Basita Das for being with me during my PhD. Her love and support have helped immensely over the years and especially during the pandemic. She has spent many hours listening to me talking about my work.

7. Bibliography

[1] "Biotechnology Market Growth Analysis Report, 2021-2028." https://www.grandviewresearch.com/industry-analysis/biotechnology-market (accessed Apr. 01, 2022).

[2] S. Zobel-Roos *et al.*, "Accelerating biologics manufacturing by modeling or: is approval under the QbD and PAT approaches demanded by authorities acceptable without a digital-twin?," *Processes*, vol. 7, no. 2, p. 94, 2019.

[3] O. J. Wouters, M. McKee, and J. Luyten, "Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018," *JAMA*, vol. 323, no. 9, pp. 844–853, Mar. 2020, doi: 10.1001/jama.2020.1166.

[4] J. Schmölder and M. Kaspereit, "A Modular Framework for the Modelling and Optimization of Advanced Chromatographic Processes," *Processes*, vol. 8, no. 1, p. 65, 2020, doi: 10.3390/pr8010065.

[5] D. Karlsson, N. Jakobsson, A. Axelsson, and B. Nilsson, "Model-based optimization of a preparative ion-exchange step for antibody purification," *J. Chromatogr. A*, vol. 1055, no. 1–2, pp. 29–39, 2004, doi: 10.1016/j.chroma.2004.08.151.

[6] F. Ojala, M. Max-Hansen, D. Kifle, N. Borg, and B. Nilsson, "Modelling and optimisation of preparative chromatographic purification of europium," *J. Chromatogr. A*, vol. 1220, pp. 21–25, 2012, doi: 10.1016/j.chroma.2011.11.028.

[7] O. Khanal, V. Kumar, F. Schlegel, and A. M. Lenhoff, "Estimating and leveraging protein diffusion on ion-exchange resin surfaces," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 117, no. 13, pp. 7004–7010, Mar. 2020, doi: 10.1073/pnas.1921499117.

[8] S. Yamamoto, K. Nakanishi, R. Matsuno, and T. Kamijubo, "Ion exchange chromatography of proteins?predictions of elution curves and operating conditions. II. Experimental verification," *Biotechnol. Bioeng.*, vol. 25, no. 5, pp. 1373–1391, 1983, doi: 10.1002/bit.260250516.

[9] S. Yamamoto, K. Nakanishi, R. Matsuno, and T. Kamikubo, "Ion exchange chromatography of proteins?prediction of elution curves and operating conditions. I. Theoretical considerations," *Biotechnol. Bioeng.*, vol. 25, no. 6, pp. 1465–1483, 1983, doi: 10.1002/bit.260250605.

[10] C. A. Brooks and S. M. Cramer, "Steric mass-action ion exchange: Displacement profiles and induced salt gradients," *AIChE J.*, vol. 38, no. 12, pp. 1969–1978, 1992, doi: 10.1002/aic.690381212.

[11] S. D. Gadam, G. Jayaraman, and S. M. Cramer, "Characterization of non-linear adsorption properties of dextran-based polyelectrolyte displacers in ion-exchange systems," *J. Chromatogr. A*, vol. 630, no. 1–2, pp. 37–52, 1993, doi: 10.1016/0021-9673(93)80440-j.

[12] T. Gu, "Modeling of Affinity Chromatography," *Mathematical Modeling and Scale-up of Liquid Chromatography*. Springer Berlin Heidelberg, pp. 81–94, 1995. doi: 10.1007/978-3-642-79541-1_8.

[13] U. Altenhöner, M. Meurer, J. Strube, and H. Schmidt-Traub, "Parameter estimation for the simulation of liquid chromatography," *J. Chromatogr. A*, vol. 769, no. 1, pp. 59–69, 1997, doi: 10.1016/s0021-9673(97)00173-8.

[14] F. James, M. Sepúlveda, F. Charton, I. Quiñones, and G. Guiochon, "Determination of binary competitive equilibrium isotherms from the individual chromatographic band profiles," *Chem. Eng. Sci.*, vol. 54, no. 11, pp. 1677–1696, 1999, doi: 10.1016/s0009-2509(98)00539-9.

[15] F. Gritti, W. Piatkowski, and G. Guiochon, "Study of the mass transfer kinetics in a monolithic column," *J. Chromatogr. A*, vol. 983, no. 1–2, pp. 51–71, 2003, doi: 10.1016/s0021-9673(02)01648-5.

[16] A. Felinger, A. Cavazzini, and G. Guiochon, "Numerical determination of the competitive isotherm of enantiomers," *J. Chromatogr. A*, vol. 986, no. 2, pp. 207–225, 2003, doi: 10.1016/s0021-9673(02)01919-2.

[17] D. Karlsson, N. Jakobsson, K.-J. Brink, A. Axelsson, and B. Nilsson, “Methodologies for model calibration to assist the design of a preparative ion-exchange step for antibody purification,” *J. Chromatogr. A*, vol. 1033, no. 1, pp. 71–82, 2004, doi: 10.1016/j.chroma.2003.12.072.

[18] A. Ladiwala, K. Rege, C. M. Breneman, and S. M. Cramer, “A priori prediction of adsorption isotherm parameters and chromatographic behavior in ion-exchange systems,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 102, no. 33, pp. 11710–11715, Aug. 2005, doi: 10.1073/pnas.0408769102.

[19] G. Carta, A. R. Ubiera, and T. M. Pabst, “Protein Mass Transfer Kinetics in Ion Exchange Media: Measurements and Interpretations,” *Chem. Eng. Technol.*, vol. 28, no. 11, pp. 1252–1264, 2005, doi: 10.1002/ceat.200500122.

[20] P. Forssén, R. Arnell, and T. Fornstedt, “An improved algorithm for solving inverse problems in liquid chromatography,” *Comput. Chem. Eng.*, vol. 30, no. 9, pp. 1381–1391, 2006, doi: 10.1016/j.compchemeng.2006.03.004.

[21] M. Schröder, E. von Lieres, and J. Hubbuch, “Direct Quantification of Intraparticle Protein Diffusion in Chromatographic Media,” *J. Phys. Chem. B*, vol. 110, no. 3, pp. 1429–1436, 2005, doi: 10.1021/jp0542726.

[22] T. Müller-Späth *et al.*, “Model simulation and experimental verification of a cation-exchange IgG capture step in batch and continuous chromatography,” *J. Chromatogr. A*, vol. 1218, no. 31, pp. 5195–5204, 2011, doi: 10.1016/j.chroma.2011.05.103.

[23] A. Osberghaus, S. Hepbildikler, S. Nath, M. Haindl, E. von Lieres, and J. Hubbuch, “Determination of parameters for the steric mass action model—A comparison between two approaches,” *J. Chromatogr. A*, vol. 1233, pp. 54–65, 2012, doi: 10.1016/j.chroma.2012.02.004.

[24] Z. Liu, J. Roininen, I. Pulkkinen, T. Sainio, and V. Alopaeus, “Moment based weighted residual method—New numerical tool for a nonlinear multicomponent chromatographic general rate model,” *Comput. Chem. Eng.*, vol. 53, pp. 153–163, 2013, doi: 10.1016/j.compchemeng.2013.02.008.

[25] V. Kumar, S. Leweke, E. von Lieres, and A. S. Rathore, “Mechanistic modeling of ion-exchange process chromatography of charge variants of monoclonal antibody products,” *J. Chromatogr. A*, vol. 1426, pp. 140–153, 2015, doi: 10.1016/j.chroma.2015.11.062.

[26] T. Gu, G. Iyer, and K.-S. C. Cheng, “Parameter estimation and rate model simulation of partial breakthrough of bovine serum albumin on a column packed with large Q Sepharose anion-exchange particles,” *Sep. Purif. Technol.*, vol. 116, pp. 319–326, 2013, doi: 10.1016/j.seppur.2013.06.004.

[27] M. Rüdt, F. Gillet, S. Heege, J. Hitzler, B. Kalbfuss, and B. Guélat, “Combined Yamamoto approach for simultaneous estimation of adsorption isotherm and kinetic parameters in ion-exchange chromatography,” *J. Chromatogr. A*, vol. 1413, pp. 68–76, 2015, doi: 10.1016/j.chroma.2015.08.025.

[28] T. Hahn, P. Baumann, T. Huuk, V. Heuveline, and J. Hubbuch, “UV absorption-based inverse modeling of protein chromatography,” *Eng. Life Sci.*, vol. 16, no. 2, pp. 99–106, 2015, doi: 10.1002/elsc.201400247.

[29] T. C. Huuk, T. Hahn, K. Doninger, J. Griesbach, S. Hepbildikler, and J. Hubbuch, “Modeling of complex antibody elution behavior under high protein load densities in ion exchange chromatography using an asymmetric activity coefficient,” *Biotechnol. J.*, vol. 12, no. 3, p. 1600336, 2017, doi: 10.1002/biot.201600336.

[30] G. Wang, T. Briskot, T. Hahn, P. Baumann, and J. Hubbuch, “Estimation of adsorption isotherm and mass transfer parameters in protein chromatography using artificial neural networks,” *J. Chromatogr. A*, vol. 1487, pp. 211–217, 2017, doi: 10.1016/j.chroma.2017.01.068.

[31] V. Kumar and A. M. Lenhoff, “Mechanistic Modeling of Preparative Column Chromatography for Biotherapeutics,” *Annu. Rev. Chem. Biomol. Eng.*, vol. 11, no. 1, pp. 235–255, 2020, doi: 10.1146/annurev-chembioeng-102419-125430.

[32] “The perceptron: A probabilistic model for information storage and organization in the brain. - PsycNET.” https://doi.apa.org/doiLanding?doi=10.1037%2Fh0042519 (accessed Jan. 03, 2022).
[33] A. G. Ivakhnenko, *Cybernetic Predicting Devices*. CCM Information Corporation, 1973.
[34] S. Dreyfus, “The computational solution of optimal control problems with time lag,” *IEEE Trans. Autom. Control*, vol. 18, no. 4, pp. 383–385, Aug. 1973, doi: 10.1109/TAC.1973.1100330.
[35] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: 10.1007/BF00344251.
[36] “Applications of Advances in Nonlinear Sensitivity Analysis | BibSonomy.” https://www.bibsonomy.org/bibtex/14dce9cc667f89c7c3318a9d8475e3cb9/idsia (accessed Jan. 03, 2022).
[37] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, 1989, doi: 10.1016/0893-6080(89)90020-8.
[38] C.-A. Bohn, “Kohonen Feature Mapping through Graphics Hardware,” in *In Proceedings of Int. Conf. on Compu. Intelligence and Neurosciences*, 1998, pp. 64–67.
[39] Z. Luo, H. Liu, and X. Wu, *Artificial neural network computation on graphic process unit*, vol. 1. 2005, p. 626 vol. 1. doi: 10.1109/IJCNN.2005.1555903.
[40] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition,” *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010, doi: 10.1162/NECO_a_00052.
[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[42] D.-X. Zhou, “Universality of deep convolutional neural networks,” *Appl. Comput. Harmon. Anal.*, vol. 48, no. 2, pp. 787–794, Mar. 2020, doi: 10.1016/j.acha.2019.06.004.
[43] A. Heinecke, J. Ho, and W.-L. Hwang, “Refinement and Universal Approximation via Sparsely Connected ReLU Convolution Nets,” *IEEE Signal Process. Lett.*, vol. 27, pp. 1175–1179, 2020, doi: 10.1109/LSP.2020.3005051.
[44] M. Mouellef, F. L. Vetter, S. Zobel-Roos, and J. Strube, “Fast and Versatile Chromatography Process Design and Operation Optimization with the Aid of Artificial Intelligence,” *Processes*, vol. 9, no. 12, Art. no. 12, Dec. 2021, doi: 10.3390/pr9122121.
[45] J. Diedrich *et al.*, “Multi-state steric mass action model and case study on complex high loading behavior of mAb on ion exchange tentacle resin,” *J. Chromatogr. A*, vol. 1525, pp. 60–70, 2017, doi: 10.1016/j.chroma.2017.09.039.
[46] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Math. Comput.*, vol. 19, no. 90, p. 297, 1965, doi: 10.1090/s0025-5718-1965-0178586-1.
[47] Abraham. Savitzky and M. J. E. Golay, “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.,” *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, 1964, doi: 10.1021/ac60214a047.
[48] L. L. Schumaker and P. Dierckx, “Curve and Surface Fitting with Splines.,” *Math. Comput.*, vol. 63, no. 207, p. 427, 1994, doi: 10.2307/2153590.
[49] P. Virtanen *et al.*, “Author Correction: SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nat. Methods*, vol. 17, no. 3, p. 352, Mar. 2020, doi: 10.1038/s41592-020-0772-5.
[50] S. Butterworth, “On the Theory of Filter Amplifiers - Experimental Wireless - 1930,” *Exp. Wirel. Wirel. Eng.*, vol. 7, no. October, pp. 536–541, 1930.
[51] W. E. Thomson, “Delay networks having maximally flat frequency characteristics,” *Proc. IEE - Part III Radio Commun. Eng.*, vol. 96, no. 44, pp. 487–490, Nov. 1949, doi: 10.1049/pi-3.1949.0101.
[52] W. Heymann, J. Glaser, F. Schlegel, W. Johnson, P. Rolandi, and E. von Lieres, “Advanced score system and automated search strategies for parameter estimation in mechanistic

chromatography modeling," *J. Chromatogr. A*, vol. 1661, p. 462693, Jan. 2022, doi: 10.1016/j.chroma.2021.462693.

[53] J. Samuelsson, P. Sajonz, and T. Fornstedt, "Impact of an error in the column hold-up time for correct adsorption isotherm determination in chromatography," *J. Chromatogr. A*, vol. 1189, no. 1–2, pp. 19–31, 2008, doi: 10.1016/j.chroma.2007.10.032.

[54] K. Pearson, "VII. Note on regression and inheritance in the case of two parents," *Proc. R. Soc. Lond.*, vol. 58, no. 347–352, pp. 240–242, 1895, doi: 10.1098/rspl.1895.0041.

[55] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Q. Appl. Math.*, vol. 2, no. 3, pp. 258–261, 1944, doi: 10.1090/qam/10667.

[56] F. Hayes-Roth, "Review of 'Adaptation in Natural and Artificial Systems by John H. Holland', The U. of Michigan Press, 1975," *ACM SIGART Bull.*, no. 53, p. 15, 1975, doi: 10.1145/1216504.1216510.

[57] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979, doi: 10.1080/00401706.1979.10489755.

[58] H. Niederreiter, "Low-discrepancy and low-dispersion sequences," *J. Number Theory*, vol. 30, no. 1, pp. 51–70, 1988, doi: 10.1016/0022-314x(88)90025-x.

[59] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[60] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2014, doi: 10.1109/tevc.2013.2281535.

[61] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," *TIK-Rep.*, vol. 103, 2001, doi: 10.3929/ethz-a-004284029.

[62] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.

[63] M. A. Branch, T. F. Coleman, and Y. Li, "A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems," *SIAM J. Sci. Comput.*, vol. 21, no. 1, pp. 1–23, 1999, doi: 10.1137/s1064827595289108.

[64] F.-M. De Rainville, F.-A. Fortin, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP," *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*. ACM Press, 2012. doi: 10.1145/2330784.2330799.

[65] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The MCMC Hammer," *Publ. Astron. Soc. Pac.*, vol. 125, no. 925, pp. 306–312, Mar. 2013, doi: 10.1086/670067.

[66] J. Goodman and J. Weare, "Ensemble samplers with affine invariance," *Commun. Appl. Math. Comput. Sci.*, vol. 5, no. 1, pp. 65–80, Jan. 2010, doi: 10.2140/camcos.2010.5.65.

[67] B. Nelson, E. B. Ford, and M. J. Payne, "RUN DMC: AN EFFICIENT, PARALLEL CODE FOR ANALYZING RADIAL VELOCITY OBSERVATIONS USING N -BODY INTEGRATIONS AND DIFFERENTIAL EVOLUTION MARKOV CHAIN MONTE CARLO," *Astrophys. J. Suppl. Ser.*, vol. 210, no. 1, p. 11, Dec. 2013, doi: 10.1088/0067-0049/210/1/11.

[68] C. J. F. ter Braak and J. A. Vrugt, "Differential Evolution Markov Chain with snooker updater and fewer chains," *Stat. Comput.*, vol. 18, no. 4, pp. 435–446, Oct. 2008, doi: 10.1007/s11222-008-9104-9.

[69] J. K. Kruschke, "JAGS," in *Doing Bayesian Data Analysis*, Elsevier, 2015, pp. 193–219. doi: 10.1016/b978-0-12-405888-0.00008-8.

[70] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, "ArviZ a unified library for exploratory analysis of Bayesian models in Python," *J. Open Source Softw.*, vol. 4, no. 33, p. 1143, Jan. 2019, doi: 10.21105/joss.01143.

[71] D. Foreman-Mackey, “Autocorrelation analysis & convergence — emcee,” *Autocorrelation analysis & convergence*. https://emcee.readthedocs.io/en/stable/tutorials/autocorr/ (accessed Aug. 19, 2021).
[72] A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner, “Rank-Normalization, Folding, and Localization: An Improved Rˆ for Assessing Convergence of MCMC,” *Bayesian Anal.*, vol. 1, no. 1, Jan. 2021, doi: 10.1214/20-ba1221.
[73] J. B. MacQueen, “Some Methods for Classification and Analysis of MultiVariate Observations,” in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, vol. 1, pp. 281–297.
[74] “walkers stuck in low probability space,” *Google Groups*. https://groups.google.com/g/emcee-users/c/fg7sQNw8YcU?pli=1 (accessed Aug. 19, 2021).
[75] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.
[76] J. H. Friedman, “Greedy function approximation: A gradient boosting machine.,” *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, doi: 10.1214/aos/1013203451.
[77] “Gaussian Processes for Machine Learning: Book webpage.” http://gaussianprocess.org/gpml/ (accessed May 17, 2022).
[78] “PyTorch.” https://www.pytorch.org (accessed May 17, 2022).
[79] “TensorFlow.” https://www.tensorflow.org/ (accessed May 17, 2022).
[80] L. Bottou, “Online Algorithms and Stochastic Approximations,” in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998. [Online]. Available: http://leon.bottou.org/papers/bottou-98x
[81] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.
[82] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Jan. 2017, Accessed: Jan. 06, 2022. [Online]. Available: http://arxiv.org/abs/1412.6980
[83] L. Liu *et al.*, “On the Variance of the Adaptive Learning Rate and Beyond,” *ArXiv190803265 Cs Stat*, Oct. 2021, Accessed: Jan. 06, 2022. [Online]. Available: http://arxiv.org/abs/1908.03265
[84] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, “Lookahead Optimizer: k steps forward, 1 step back,” *ArXiv190708610 Cs Stat*, Jul. 2019, Accessed: Jan. 06, 2022. [Online]. Available: http://arxiv.org/abs/1907.08610
[85] R. Grosse, “Lecture 15: Exploding and Vanishing Gradients,” p. 11.
[86] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[87] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Madison, WI, USA, Jun. 2010, pp. 807–814.
[88] “Papers with Code - Self-Normalizing Neural Networks.” https://paperswithcode.com/paper/self-normalizing-neural-networks (accessed Jan. 07, 2022).
[89] P. Ramachandran, B. Zoph, and Q. Le, “Swish: a Self-Gated Activation Function,” Oct. 2017.
[90] S. Eger, P. Youssef, and I. Gurevych, “Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks,” arXiv, arXiv:1901.02671, Jan. 2019. doi: 10.48550/arXiv.1901.02671.
[91] A. Al-Safwan, C. Song, and U. bin Waheed, “Is it time to swish? Comparing activation functions in solving the Helmholtz equation using physics-informed neural networks,” arXiv, arXiv:2110.07721, Oct. 2021. doi: 10.48550/arXiv.2110.07721.
[92] P. Gavrikov, “visualkeras,” *GitHub repository*. GitHub, 2020. [Online]. Available: https://github.com/paulgavrikov/visualkeras

[93] Immudzen, “Neural Network predict FFT,” *r/MLQuestions*, Jan. 28, 2020. www.reddit.com/r/MLQuestions/comments/ev3pgp/neural_network_predict_fft/ (accessed Jan. 18, 2022).
[94] D. Berthelot, T. Schumm, and L. Metz, “BEGAN: Boundary Equilibrium Generative Adversarial Networks,” Mar. 2017, Accessed: Jan. 18, 2022. [Online]. Available: https://arxiv.org/abs/1703.10717v4
[95] K. Team, “Keras documentation: KerasTuner.” https://keras.io/keras_tuner/ (accessed Jan. 25, 2022).
[96] “Tune: Scalable Hyperparameter Tuning — Ray v1.9.2.” https://docs.ray.io/en/latest/tune/index.html (accessed Jan. 25, 2022).
[97] T. O’Malley *et al.*, “KerasTuner.” 2019. [Online]. Available: https://github.com/keras-team/keras-tuner
[98] “New AMD Instinct^{TM} MI200 Series Accelerators Bring Leadership HPC and AI Performance to Power Exascale Systems and More.” https://www.amd.com/en/press-releases/2021-11-08-new-amd-instinct-mi200-series-accelerators-bring-leadership-hpc-and-ai (accessed Jun. 06, 2022).
[99] J. Beck, W. Heymann, E. von Lieres, and R. Hahn, “Compartment Model of Mixing in a Bubble Trap and Its Impact on Chromatographic Separations,” Multidisciplinary Digital Publishing Institute. Accessed: May 26, 2022. [Online]. Available: https://doi.org/10.3390/pr8070780
[100] D. Bhowick, D. K. Gupta, S. Maiti, and U. Shankar, “Stacked autoencoders based machine learning for noise reduction and signal reconstruction in geophysical data,” arXiv, arXiv:1907.03278, Jul. 2019. doi: 10.48550/arXiv.1907.03278.

8. Appendix

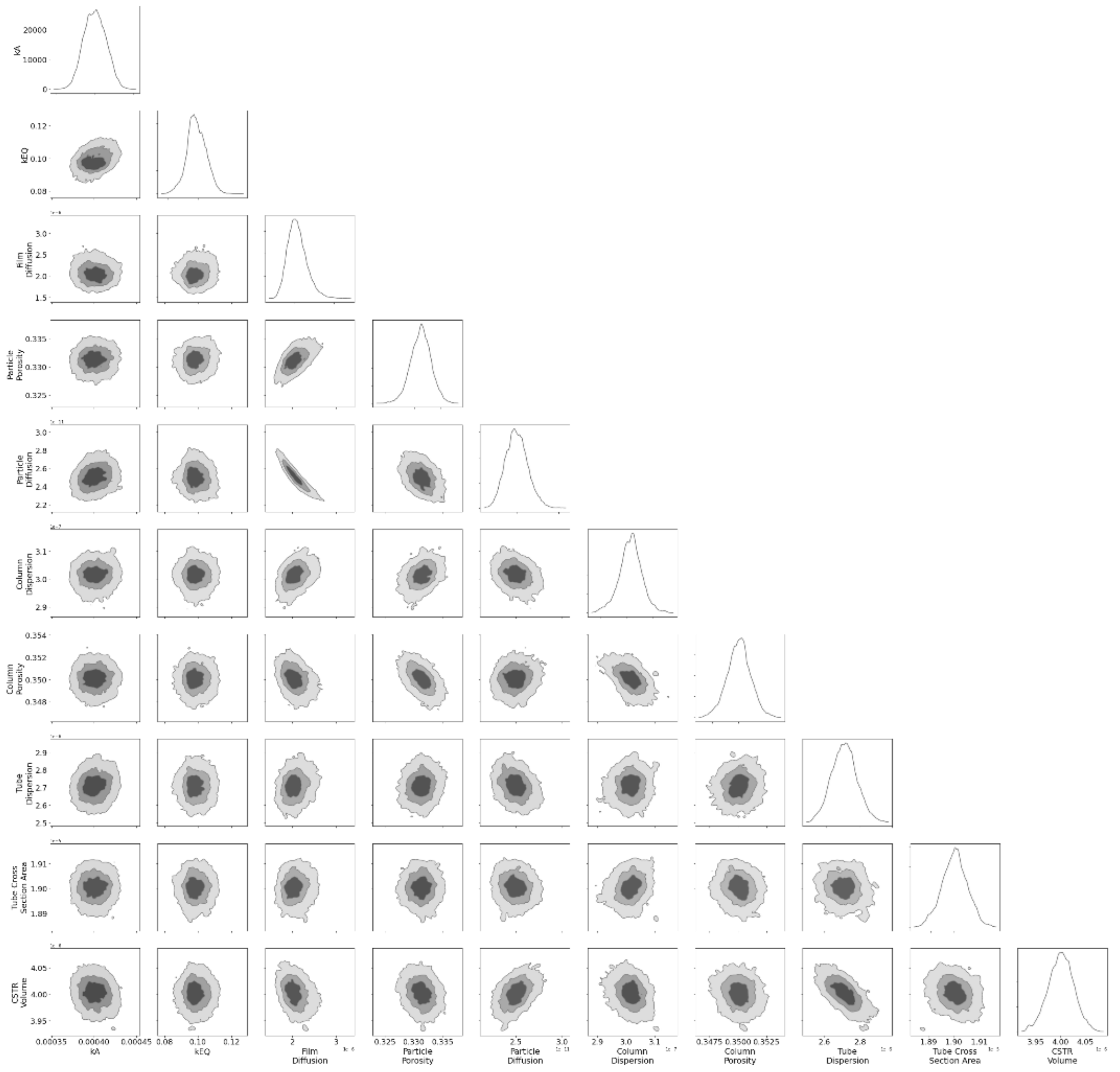

Figure 78: Synthetic linear binding protein experiment narrow error source corner plot.

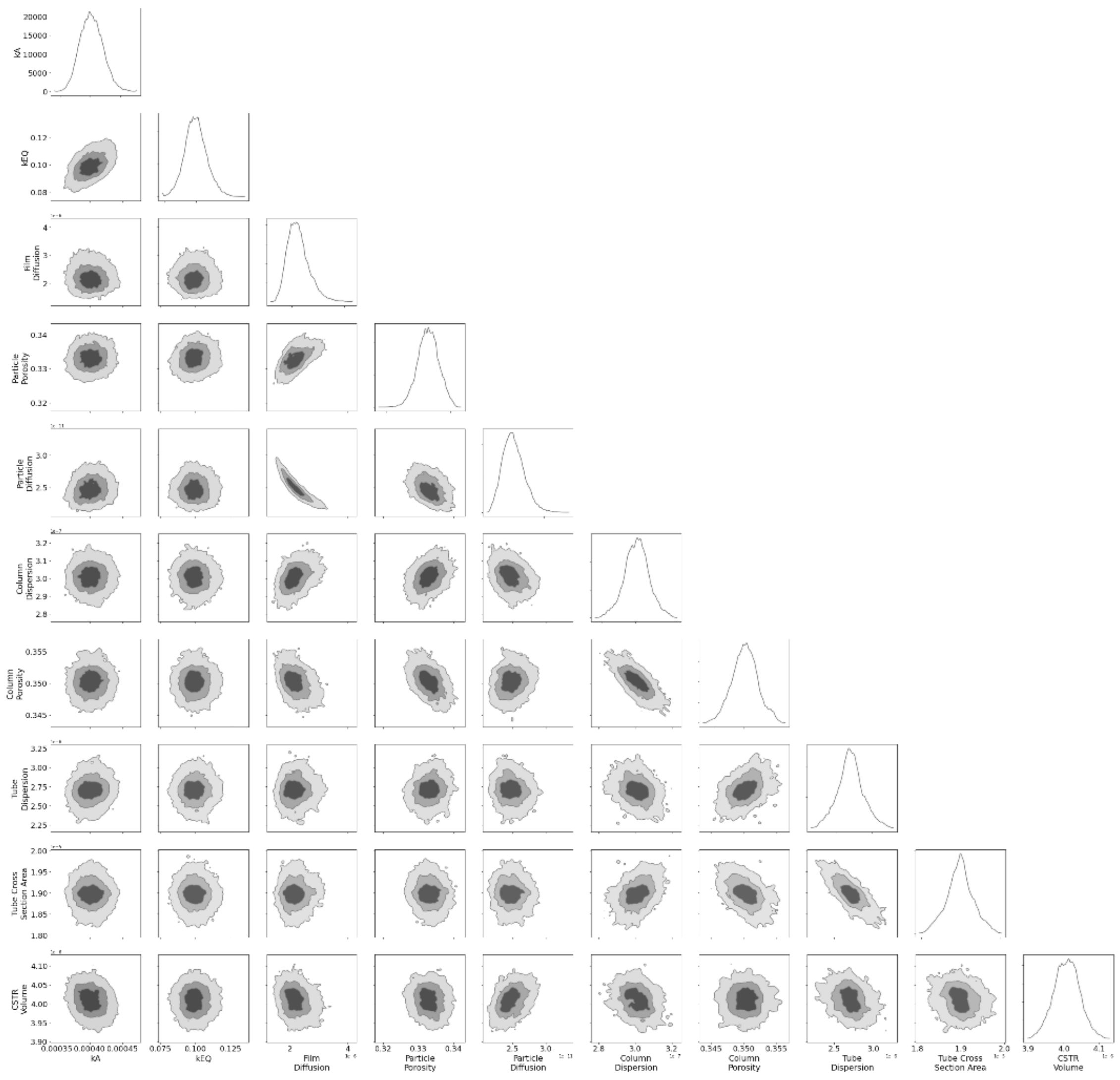

Figure 79: Synthetic linear binding protein experiment wide error source corner plot.