Using Machine Learning to Improve the Hard Modeling of NMR Time Series

Jan Hellwig[a,b], Tobias Strauß[a], Erik von Harbou[c], Klaus Neymeyr[a,b]

[a] *Universität Rostock, Institut für Mathematik, 18057 Rostock, Germany*
[b] *Leibniz-Institut für Katalyse e.V., 18059 Rostock, Germany*
[c] *RPTU Kaiserslautern-Landau, Fachbereich Maschinenbau und Verfahrenstechnik, 67663 Kaiserslautern, Germany*

Abstract

Modeling time series of NMR spectra is a useful method to accurately extract information such as temporal concentration profiles from complex processes, e.g. reactions. Modeling these time series by using nonlinear optimization often suffers from high runtimes. On the other hand, using deep learning solves the modeling problem quickly, especially for single spectra with separated peaks. However, the accuracy decreases significantly when peaks overlap or cross. We propose a hybrid approach combining the strengths of both methods while mitigating their drawbacks. This hybrid methods employs neural networks to predict initial parameters for the optimization algorithm, which only needs to fine-tune the parameters afterwards. We present results for both constructed and experimental data sets and achieve improvements in both runtime and accuracy.

Keywords: NMR time series, Model-based decomposition, Nonlinear optimization, Convolutional neural networks

1. Introduction

NMR spectroscopy has become a popular tool for analyzing complex processes such as reactions and distillations [8, 18, 19]. To follow the process, repeated acquisitions of NMR spectra are performed. To extract quantitative information about the composition of the investigated mixture as a function of time, each individual spectrum must be analysed. To facilitate these analyses, the peaks of the spectra have to be detected and modeled. Recently, it has become common practice to use neural networks for peak detection in single NMR spectra [7, 15, 25]. However, the application of neural networks to time series of NMR spectra goes beyond a simple consecutive application to a few single spectra. The problem of peak shifts and possible crossings, for example due to change in the pH-value, requires correct peak tracing over time. Furthermore, situations where multiple peaks overlap lead to multiple viable solutions for current neural networks, even though the mathematical solution is unique [12]. Using the additional information provided by the sequence of spectra helps to resolve this type of ambiguity. Our proposed solution for modeling time series of NMR spectra assumes that the peak parameters are smooth functions in time, which allows their interpolation by cubic spline functions, thus limiting the number of parameters at given points in time that need to be optimized [20]. The smoothness of the splines guarantees robust peak tracking and deconvolution of overlapping peaks, provided the peak parameters are correctly optimized. However, this is not always the case. A large number of peaks and thus of peak parameters not only leads to a loss of accurate tracing, but also to long program runtimes. To address the shortcomings of nonlinear optimization algorithms, we use convolutional neural networks to provide more accurate initial estimates, from which the optimizer only needs to fine-tune the predicted parameters to achieve numerical convergence.

1.1. Problem Definition

A time series of NMR spectral data can be represented by a data matrix $D \in \mathbb{R}^{|T|\times|X|}$, where X is the set of frequency channels x and T is the index set of spectra t. For the sake of simplicity, we also refer to T as the set of points in time at which the spectra are measured. To ease the explanation of the methods, we use a constructed data set, which is displayed in Fig. 1. The goal of modeling spectral data is to construct a simple model fitting the data well. This is best described by minimizing the sum of squares, or the squared Frobenius norm

$$\|D - M\|_F^2 = \sum_{t\in T}\sum_{x\in X}(D(t,x) - M(t,x))^2 \to \min!, \tag{1}$$

where $M(t, x)$ denotes the value of the model at time t and frequency x. The so-called hard modeling [1] aims to model each peak with a given type of parametrized model function f_p. We define the model matrix row-wise, with each row representing a single spectrum. If n_t peaks are present in a single spectrum $t \in T$, we define

$$M(t, x) = \sum_{i=1}^{n_t} f_{p_i(t)}(x) \qquad \forall x \in X. \tag{2}$$

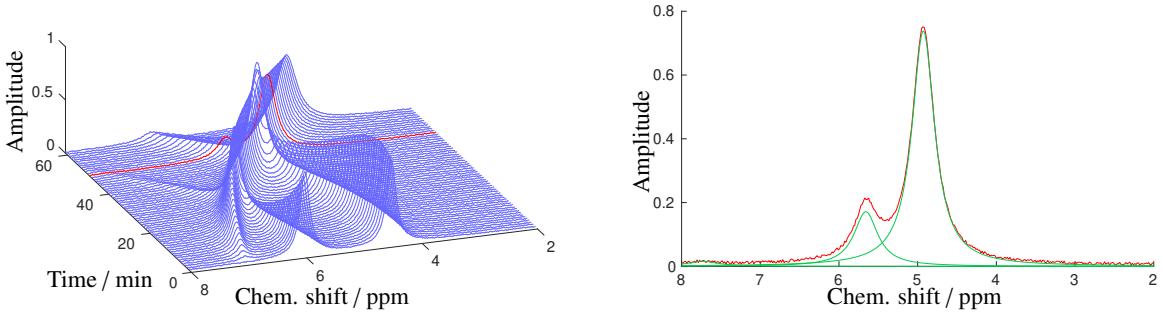

Figure 1: Constructed data set containing three peaks and two crossings on the left hand side. The spectrum highlighted in red for $t = 49$ is modeled by three green peaks on the right hand side, resulting in a lack of fit of $\|D(49, :) - M(49, :)\|_F = 0.7317$.

We require the spectra of the data to be *connected* in a sense that the data should describe a continuous and coherent evolution process of some kind, e.g. a gradual change in pH value, pressure, a chemical reaction occurring, etc. The modeling problem is thus equivalent to finding values for n_t, $p_i(t)$ for all $t \in T$ and $i = 1, \ldots, n_t$ such that (1) is minimal.

1.2. Mathematical Notation and Definitions.

In this work, we use the following notation:

T	Set of points in time $t \in T$
X	Set of frequency channels $x \in X$
D	Data matrix containing row-wise the series of connected spectra $D(t, :)$
M	Matrix containing row-wise the series of hard modeled spectra $M(t, :)$
n	Maximum number of peaks present in one spectrum
m	Multiplicity of a multiplet
$p_i(t)$	Parameter tuple of the model function $f_i, i = 1, \ldots, n$ at point $t \in T$. Usually, $p_i(t) = (c_i(t), h_i(t), w_i(t), \lambda_i(t))$ where $p_i(t)$ is a continuous, differentiable function in time. The parameters (c, h, w, λ) are defined in Sec. 1.3

Using the Frobenius norm to measure the model quality can be suboptimal, since small changes in model parameters can cause large changes in the Frobenius norm. In addition, comparing the accuracy across multiple different data sets is particularly difficult. For example, two different data sets may have models of similar quality, but the relative error with respect to the Frobenius norm may differ by a factor of 6. This can be seen by comparing the relative errors of the first columns of Table 1. Therefore, we introduce another metric to determine the model quality.

The Wasserstein metric considers the function curves of two spectra as profiles of sand piles and denotes the effort required to transform one such pile into the other. It originates from probability theory and calculates the distance between two probability measures. Since computing the cost of an optimal transport plan is non-trivial at first sight, we use an alternative definition, suited for one-dimensional, discrete measures. The usual definition and the proof of equivalence can be found in Appendix B. Our definition further allows to compute the Wasserstein metric in linear time.

Definition 1.1 (Wasserstein metric)**.** *Let μ and ν be probability measures over a discrete set $X = \{x_1 < \ldots < x_k\}$. Then,*

$$w(\mu, \nu) = \sum_{i=1}^{k-1} (x_{i+1} - x_i) \left| \sum_{j=1}^{i} \mu_j - \nu_j \right| \tag{3}$$

defines the 1D-Wasserstein metric for discrete probability measures, where $\mu_j = \mu(\{x_j\})$ and $\nu_j = \nu(\{x_j\})$ respectively.

This metric is well-known in the context of machine learning [26]. In NMR spectroscopy, it is used by [4]. The Wasserstein metric is particularly robust to offset peaks. Furthermore, when normalized, it provides a good comparison for model accuracy between different data sets. All comparisons can be found in Sec. 4.

1.3. Peak Model Functions

Hard models of NMR spectra typically use Lorentz, pseudo-Voigt, or Voigt functions to model individual peaks. [14, 17, 23, 25] We denote the mean or *center* of all functions by $c \in \mathbb{R}$. The *height* at $x = c$ is denoted by $h \geq 0$. The parameter $w > 0$ describes the so-called *half width* (at half height). This is the offset from the center where $f(c \pm w) = \frac{h}{2}$ holds. Pseudo-Voigt functions have an additional parameter $\lambda \in [0, 1]$, denoting the coefficient of the convex combination of the Lorentz and Gauss functions.

First, we introduce Lorentz and Gauss functions. Lorentz functions correspond to an ideal NMR signal when measured perfectly without noise or other physical interferences

$$L_{c,h,w}(x) = h \cdot \frac{w^2}{w^2 + (x - c)^2}. \tag{4}$$

Gauss functions are usually combined with Lorentzians to achieve a wider range of possible line shapes. Note that the Gaussians appear slightly unusual to accommodate for the half width

$$G_{c,h,w}(x) = h \cdot \exp\left(- \ln(2) \cdot \frac{(x - c)^2}{w^2} \right). \tag{5}$$

If ideal conditions cannot be assumed, pseudo-Voigt functions can be used. These functions are defined by a convex combination of a Gaussian and a Lorentzian

$$PV_{c,h,w,\lambda}(x) = h \cdot \left((1 - \lambda) \cdot L_{c,1,w}(x) + \lambda \cdot G_{c,1,w}(x) \right). \tag{6}$$

Another type of model function is the Voigt function. If the data is contaminated with Gaussian noise, the physical spectra should appear as a sum of Voigt functions. Let v be a half width parameter, similar to w. Then, Voigt functions are defined as

$$V_{c,h,v,w}(x) = h \cdot \int_{-\infty}^{\infty} G_{0,1,v}(y) \cdot L_{c,1,w}(x - y) \mathrm{d}y. \tag{7}$$

The improved accuracy of the line shapes defined by Voigt functions is outweighed by the computational expenses necessary to evaluate Voigt functions. Since programming languages such as Python provide lookup tables, Voigt functions can be of use.[1] Still, the lack of an explicit formula makes them difficult to use when training neural networks, since in some cases the necessary gradients cannot be calculated. For this reason, we neglect Voigt functions and focus on hard modeling using only pseudo-Voigt functions.

1.4. Uniqueness of Hard Models

Let

$$s(x) = \sum_{i=1}^{n} f_{p_i}(x)$$

[1] See `https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.voigt_profile.html`

denote a single spectrum consisting of $n \geq 1$ peaks, defined for all $x \in \mathbb{R}$. Then, [12] shows that there is only one possible choice for the number of peaks n and for the parameter tuples p_i, if the pairs (c_i, w_i) of each peak are distinct. This holds for Lorentz, Gauss, pseudo-Voigt and Voigt functions.[2] Therefore, the mathematical problem of hard modeling single spectra in the continuous setting of $X = \mathbb{R}$ is uniquely solvable. Even the discrete case $|X| = N$ has a unique model, provided that the number of basis functions is bounded and N is sufficiently large.

In these cases, the model of a series of spectra also is unique. However, if noise is present and we only require our model to fit the data up to an error tolerance of $\varepsilon > 0$, multiple solutions with vastly different parameters may become feasible. Especially when peaks overlap, ambiguous solutions can be found.

Furthermore, the proofs in [12] are not constructive in the sense that they do not allow to derive an algorithm for finding these unique solutions. Therefore, we use other methods of modeling, such as nonlinear optimization or predictions using neural networks.

1.5. Existing Approach

A time series of coherent spectra provides more information than a single spectrum. This plus of information comes at the cost of having to select parameters for multiple points in time. Since the peaks appear as a result of chemical and physical processes such as chemical reactions, it is reasonable to assume that their parameters behave smoothly over time. To reduce computation time, it is possible to optimize the peak parameters on only a few spectra and interpolate the parameter functions in between. The smoothness of the parameters allows for interpolation using cubic spline functions. On the contrary, interpolation with cubic spline functions forces the modeled parameter functions to be continuous and differentiable.

The existing approach, which we improved upon, is described in [20]. Initially, one spectrum is selected manually and peaks are detected using a numerical approximation of the first and second derivatives. Subsequently, all parameters of this spectrum are optimized for an accurate model. We use the trust-region reflective algorithm as is implemented in MATLAB for all optimizations.[3]

To reduce the dimension of the underlying optimization problem, parameters are optimized only on a few selected spectra and each parameter function is interpolated by cubic splines. Optimization only occurs in a moving window along the time dimension, see also Fig. 2. During optimization, the lack of fit on the intermediate spectra is still taken into account. This leads to a more stable optimization. The differentiable splines force peak crossings to be modeled smoothly and, in most cases, correctly and allow reasonable solutions to be found when peaks overlap. Still, the optimization algorithm suffers from incorrect tracing and long runtimes of up to 24 hours for a single data set, as the complexity suffers drastically from an increase in the number of peaks and thus, parameters.

Furthermore, the optimizer may converge to a local rather than to the global minimum. If, as in Fig. 3, there is a doublet present and the initial guess of the optimizer is far off, then all optimizers using some sort of gradient descent would likely tend toward one of the local minima. Suffice it to say that for more complex situations, the number of local minima increases further. The severity of this problem can be mitigated by using the information provided by the temporal connections between spectra, but despite that, it frequently occurs in practice.

In conclusion, we identify two main areas for improvement. First, the runtime can be reduced by decreasing the number of steps needed for numerical convergence and by reducing the optimization dimension. Second, convergence to local minima can be prevented by improving the accuracy of the starting points of each optimization problem.

2. Conventional Improvements to the Spline Algorithm

2.1. Multiplets

In order to reduce the optimization dimension and thus the runtime, we include multiplets in our model. Multiplets are common in NMR spectroscopy and occur due to interactions between nuclei in close proximity [11]. This effect is called J-coupling. Multiplets consist of several peaks arranged in a regular pattern. All peaks of a multiplet are equidistant from each other, have the same half width and have a regular height pattern. We denote the distance by

[2] In the case of Voigt functions, it is only required that the parameter triples (c_i, v_i, w_i) are distinct.

[3] See `https://de.mathworks.com/help/optim/ug/least-squares-model-fitting-algorithms.html#broz0i4`

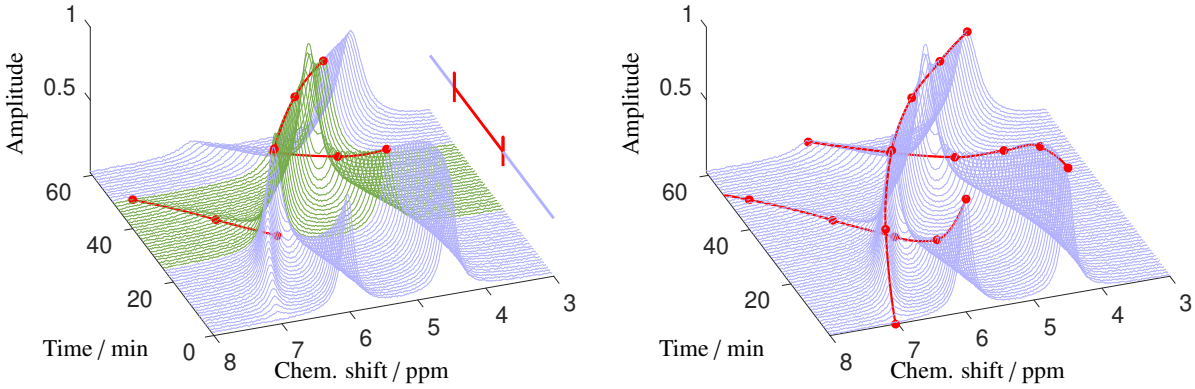

Figure 2: Optimization procedure of the spline-based algorithm from [20] applied to the constructed data set. Optimization takes place in a sliding window through time (green), parameters on intermediate spectra are interpolated between parameters on selected spectra (red dots). On the right hand side the optimized model consists of $3 \cdot 6 \cdot 4 = 72$ parameters compared to the 30,500 data points.

$d > 0$. Additionally, we assume that the convex combination coefficient λ is the same for different peaks in a multiplet. In our model, a multiplet of $m \in \mathbb{N}$ pseudo-Voigt peaks is defined as follows

$$\sum_{i=1}^{m} \binom{m-1}{i-1} \cdot PV_{c+d\cdot(i-1),h,w,\lambda}(x) = PV_{c,h,w,\lambda}(x) + (m-1) \cdot PV_{c+d,h,w,\lambda}(x) + \ldots + PV_{c+(m-1)\cdot d,h,w,\lambda}(x). \tag{8}$$

Each multiplet is defined by six parameters. It inherits the four parameters (c, h, w, λ) from single peaks, plus has a distance parameter $d > 0$ as well as a parameter denoting the multiplicity $m \in \mathbb{N}$, which does not need to be optimized. If a singlet is present, only 4 parameters are optimized since there is no use for d. Therefore, the number of optimizable parameters for an m-tuplet decreases from $4 \cdot m$ to $\min\{4 \cdot m, 5\}$. Usually, this reduces the number of parameters by $40 - 60\%$, depending on the data set. In data set 3, see Sec. 4, there are two septuplets and nine singlets present. Implementing multiplets reduces the number of parameters per spectrum from $23 \cdot 4 = 98$ to $2 \cdot 5 + 9 \cdot 4 = 46$.

One drawback of using multiplets is the loss of model flexibility, as modeled peaks are forced to appear in the multiplet pattern defined in (8). Multiplets that do not fit this pattern cause unavoidable residues. The cases in which this presents a problem for the optimization are limited in practice and can be handled by preventing the detection of multiplets using a designated hyperparameter.

On the contrary, the use of multiplets also serves to stabilize the algorithm, since in general the error function has fewer local minima. To illustrate that, consider a single doublet as in Fig. 3. If the goal is to optimize only the center parameter of the doublet, three local minima can be found, one of which is also the global minimum. If, instead, two center parameters of two singlets are to be optimized, then six local minima can be found, two of which are perfect fits. We can also observe several valleys with vanishing gradients. In the presence of noise, numerical convergence may occur in these regions as well.

Multiplet detection is performed on the initial spectrum after peaks have been identified. A heuristic method first finds multiplets with three or more peaks, whose distances are similar within a given tolerance and whose peak heights follow an ascending, then descending pattern. Note that we do not require the heights to be similar to binomial coefficients, since possible overlaps and inaccurate heights as well as skewed multiplets would limit the number of successfully detected multiplets. After assigning the peaks to multiplets of multiplicity $m \geq 3$, doublets are detected, where peaks of similar height are in close proximity. All remaining peaks are then classified as singlets.

2.2. Splitting up the Optimization

To further reduce the optimization dimension, we can reduce the number of parameters being optimized simultaneously by splitting the large optimization problem into several smaller ones. Note that if the correct center parameters

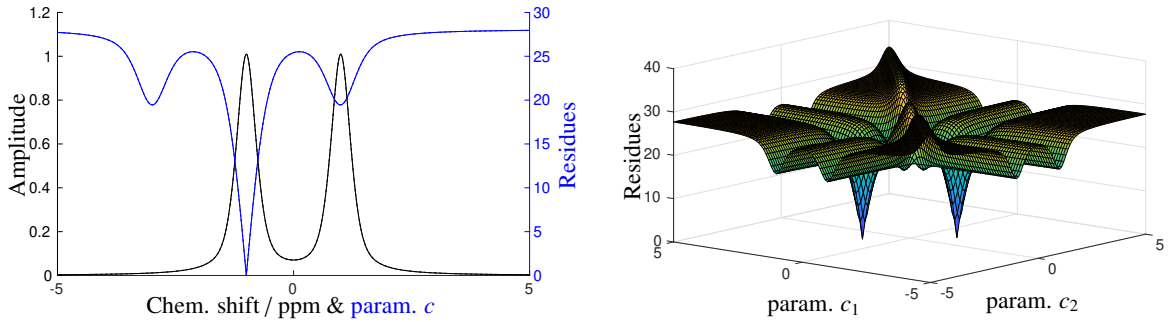

Figure 3: Optimization of one center parameter (c) for a single doublet on the left hand side. Residues (blue) to the given data (black) show three local minima. Residue function of the optimization of the centers (c_1, c_2) of two singlets on the right hand side. There are six local minima, four of which are visible.

of all peaks or multiplets are known, the optimization of all other parameters is not only more stable, but also faster. Conversely, if some centers are far from their correct positions during optimization, the optimization of all other parameters will most likely not succeed, as can be seen in Fig. 4.

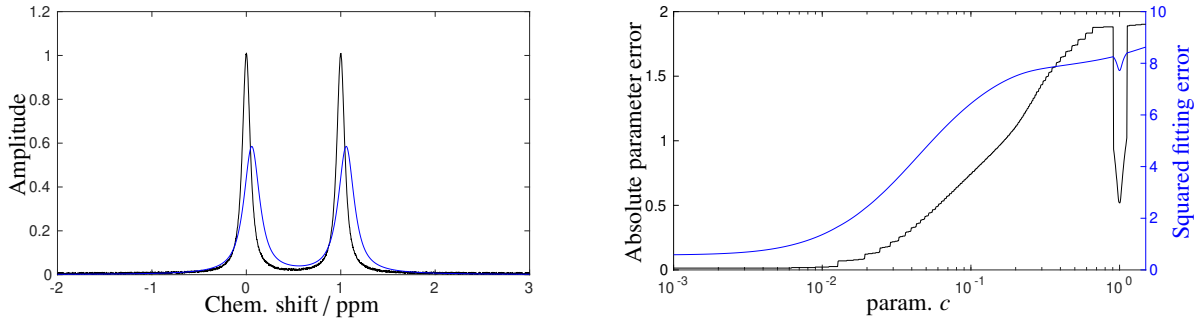

Figure 4: For a slightly displaced center ($c = 0.05$), the parameters w_c and h_c of the doublet best fitting the data differ significantly. We compare absolute errors of the parameters $\|(w_c, h_c) - (w, h)\|_1$ (black) and quadratic fitting errors (blue) for values of c on a logarithmic scale. Around $c = 1$, we experience the same local minimum also appearing in Fig. 3.

This phenomenon also often occurs for experimental data sets. If the peaks are traced incorrectly the optimizer might converge to a local minimum, leading to large fitting errors. This problem can be mitigated by selecting spectra more densely or by choosing stricter hyperparameters for the optimization algorithm, both of which further increase the runtime.

2.2.1. Optimizing Centers First

The first approach to optimization splitting is to first optimize only the center parameters of all multiplets and then optimize all other parameters. In theory, correctly optimized centers stabilize the remaining optimization, see Fig. 3. It is obvious that minimizing the lack of fit is unsuitable here, since parameters other than centers are not known yet. Instead, we heuristically predict peak positions and optimize the centers in such a way, that the distance to the nearest heuristically detected peak is minimized. Heuristic peak detection is preformed using first derivatives after smoothing the data with a Savitzky-Golay Filter [24] to reduce the influences of noise. This process is displayed in Fig. 5.

The heuristic peak detection has the advantage of robustly detecting the existence of peaks in a given area. Conversely, if no peaks are detected, then it is safe to say there are none. However, using it as described above does not result in a stable algorithm. When peaks overlap or cross, the heuristic method tends to neglect at least one of the peaks.

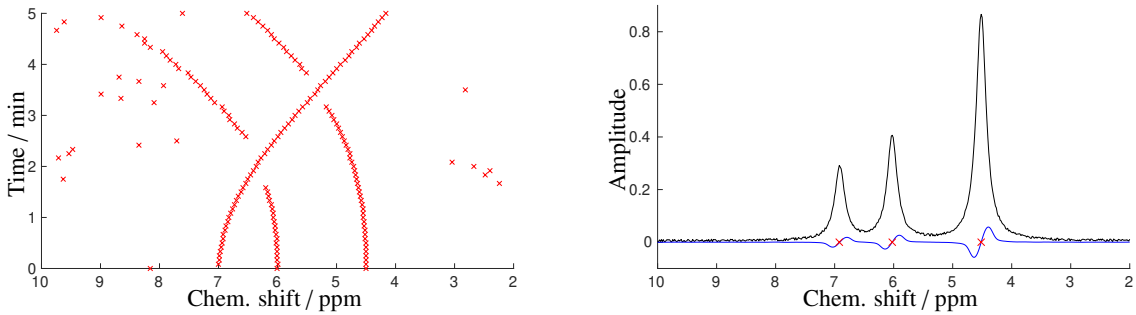

Figure 5: Heuristic peak detection using SG-smoothed first derivatives (blue) of the spectrum (black). The red crosses denote the midpoints of greatest variations in the first derivative. The left hand side shows the predictions (red) for all spectra of the constructed data set. Note the inaccurate predictions for overlapping / vanishing peaks. The initial optimization minimizes the distance of each center to the closest red cross.

In practice, optimizing the centers first does not yield sufficiently accurate results, small deviations create large problems for the subsequent optimization of all other parameters, see Fig. 4. Therefore, this optimization method is impractical, but the heuristic peak detection is still of use.

2.2.2. Splitting the Frequency Axis

A more practical approach is to split the frequency axis, creating smaller subsets $D = D_1 + \ldots + D_s$. Recall that the heuristic peak detection as described in Sec. 2.2.1 can accurately detect areas where no peaks are located. Splits can thus be defined at points $x \in X$ where for all $t \in T$ the minimum distance to any of the predicted peaks is greater than some tolerance δ_s. We take advantage of the fact that different subsets can be modeled independently of each other. When splits are found, the corresponding optimizations are preformed sequentially, but with a reduced dimension of the parameter space. Runtime improvements are documented in Sec. 4.

Splitting the dataset parallel to the frequency axis is never useful. If there are no peaks present in a given spectrum, then there are several disconnected data sets that might as well be defined completely separately.

3. Neural Networks

The most impactful improvement to the algorithm is the implementation of neural networks (NNs). Deep learning has recently become the method of choice for NMR data analysis. For an overview on the topic of deep learning, we refer to [10]. When properly trained, NNs are able to identify the number of peaks and make accurate predictions of peak parameters to achieve good model fits. Depending on the architecture, the spectrum is given as a whole or as a sequence of moving windows. There are a number of different goals and strategies when it comes to using NNs in the analysis of NMR data. An overview is given in [3]. Feed-forward networks (FFNs) are used for compound detection and quantitative analysis of NMR data [7, 13]. Convolutional neural networks (CNNs) are used in [15] to detect 2D NMR peaks and in [16] for fitting pseudo-Voigt lines under 2D X-ray diffraction peaks. Detection of regions containing peaks and hard modeling of single 1D NMR spectra using both CNNs and recursive neural networks (RNNs) can be found in [6, 25].

We choose a different approach in that we aim to predict parameters for single spectra, or sections thereof, and use them as a starting point for the nonlinear optimization algorithm, in the hope that the predicted model is close enough to the global minimum to only require a few optimization steps.

3.1. Architecture

Since peak detection is a subproblem of feature recognition in the field of deep learning, we choose to use convolutional neural networks, a common choice for this type of problem. The dimensions of the input and output vectors are fixed, so each of our CNNs can only detect a fixed number of peaks for a given spectral vector. Therefore, we trained multiple CNNs for small numbers of peaks ($n = 1, \ldots, 5$). The output of each CNN consists of $4 \cdot n$ values,

representing the peak parameters $(c_i, h_i, w_i, \lambda_i)$. For numerical stability and consistency, each output parameter is in the interval $[0, 1]$. To obtain the corresponding model, the parameters are transformed afterwards.

The schematic architecture of our NNs is displayed in Fig. 6. Each CNN consists of the following nine layers.

1 A convolutional layer creating two data channels
2 A max-pooling layer
3-4 A second convolutional layer followed by a second max-pooling layer
5-8 Four dense layers of decreasing size
9 Another dense layer of size $4 \cdot n$, computing the output parameters

All layers except for the max-pooling and output layers are activated using the standard sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{9}$$

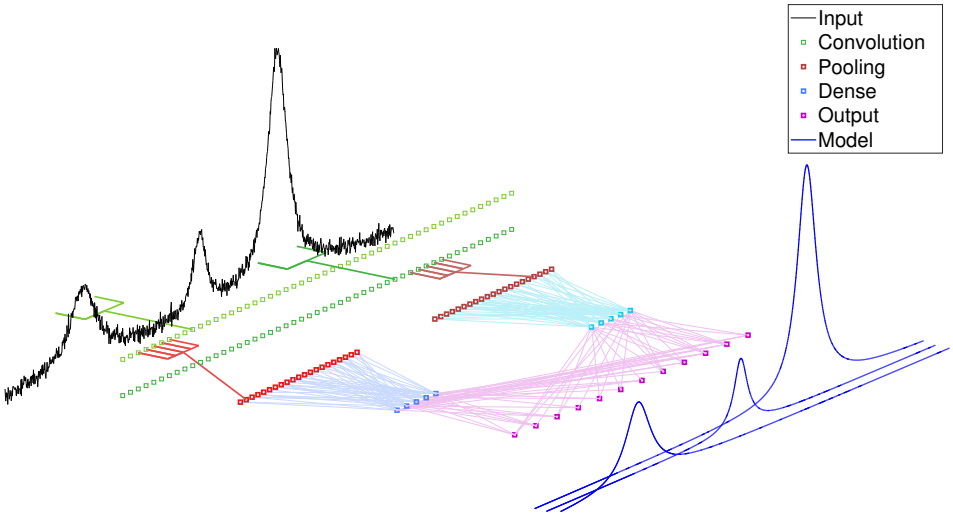

Figure 6: Schematic representation of the CNN for $n = 3$ and the reconstruction process using model functions.

We choose a common architecture for our CNNs. Similar architectures for CNNs can be found in [6, 16, 25]. First, the input vector is convolved and max-pooled twice. The results are then given to a number of dense layers, feeding into the output layer. All network parameters such as number of nodes, convolution window width, etc. were chosen by trial and error.

A drawback of using this type of architecture is that, due to pooling, features such as peak position can only be located approximately. The same type of inaccuracy can occur if the windows we choose from the whole spectrum need interpolation or binning to achieve a length of 1000, dictated by the fixed input dimension.

However, the task for which these networks are trained does not require exact parameters and perfect fitting, but rather a prediction that is sufficiently close to the correct solution. Given such a prediction, an optimizer initialized with it likely achieves quick and stable numerical convergence to the global minimum. If the spectral resolution is sufficiently high, this problem is further diminished. Another disadvantage lies in the fixed number of output nodes. This forces the inelegant solution of using different networks for different numbers of peaks. We trained networks for $n = 1, \ldots, 5$ peaks. This has the obvious drawback of not being able to predict parameters for six or more peaks. However, using this training scheme we can easily train networks for $n \geq 6$ peaks.

3.2. Training

Similar to [7, 15, 25], we only used algorithmically generated training data consisting of spectra of n pseudo-Voigt peaks with added Gaussian noise. The parameters were chosen randomly, uniformly distributed on $[0, 1]$. If the sample size is large enough, then overlapping situations as well as large height and width variations appear in the training spectra. These tend to be the situations where CNNs struggle the most. The frequency of these situations can be artificially inflated to increase the robustness of the CNNs. Predicting the parameters of pseudo-Voigt functions allows for different types of loss functions during training. First, we can measure the distance between the predicted parameters and the parameters defining the training spectra. Secondly, we can measure the distance between the generated and the reconstruced spectra. Through trial and error, the best loss function turned out to be a weighted sum of the mean squared error (MSE) in parameter space and the MSE in function space. This approach leads to initial estimates that are close to the global optimum in parameter space, and increase the chances of a stable convergence. Let $y \in [0, 1]^{4n}$ be the vector of parameters defining the input spectrum $s(y)$ and $z \in [0, 1]^{4n}$ the vector of predicted parameters defining the reconstructed spectrum $s(z)$. Then, during training we minimize

$$l(y, z) = \frac{\|y - z\|_2}{4n} + \alpha \cdot \frac{\|s(y) - s(z)\|_2}{1000}, \tag{10}$$

where $\alpha = 0.1$.

For the number of training samples, we used data sets in the range of 10^6 to 10^7 spectra. The CNNs were trained in Python using the PyTorch library [21] on the GPU servers of the Institute of Mathematics, University of Rostock.

3.3. Introducing CNNs to the Algorithm

To recapitulate, parameters are optimized belonging to several spectra in a sliding window through time, see Fig. 2. When the window slides to the next spectrum, we want to predict the parameters of this next spectrum using the CNNs. Therefore, we first need to split the frequency axis of this spectrum into disjoint subspectra. Second, we need to estimate the number of peaks present, or which prediction to use for a given subspectrum. Lastly, we need to sort the parameters into the given data structures for our peaks, since the order of the peaks may change throughout the data set.

When splitting the spectrum, intervals larger than 0.25 ppm in the frequency domain are found where no peaks are detected using the heuristic peak detection mentioned in Sec. 2.2.1 and Fig. 5. We initially split the spectrum in the middle of these intervals. Now, extreme[4] binning or interpolation can lead to prediction errors. As spectra usually have a resolution of more than 1000 points, we then check for subspectra with more than 5000 points. When such subspectra exist, we define an additional split in the largest area without heuristically detected peaks. This procedure is iterated until no subspectrum has more than 5000 points.

To prevent subspectra with less than 200 points, we check the following condition. If a window has too few points and the assumed number of peaks in this window and one of its neighboring windows is less than $n = 5$, we combine both windows. To obtain a guess on the number of peaks per subspectrum, we linearly extrapolate the peaks from previous spectra.

Although this procedure does not guarantee that no subspectra with more than 5000 points exists, nor that none with less than 200 points exist, the subspectra generated by this procedure work well for all experimental data sets.

After the spectrum has been split, all subspectra are normalized to have a size of 1000 points. This is either done by interpolating or by binning to increase or decrease the size.

Second, the parameters of the peaks in each subspectrum are predicted, and we need to decide which prediction from which network to use. Since computing prediction takes < 1 ms each, we choose to input the spectrum into all available networks. We then compute the model from these parameter predictions and calculate the lack of fit to the original data in the range of the subspectrum. The prediction with the smallest lack of fit is then chosen. Note that we need to include the case where the subspectrum contains no peaks. Therefore, if all lacks of fit are greater than the 2-norm of the data, we assume that there is no peak. Additionally, we can use the assumed number of peaks to give a further edge to the corresponding network.

[4] We consider a factor of five or more to be extreme for both cases.

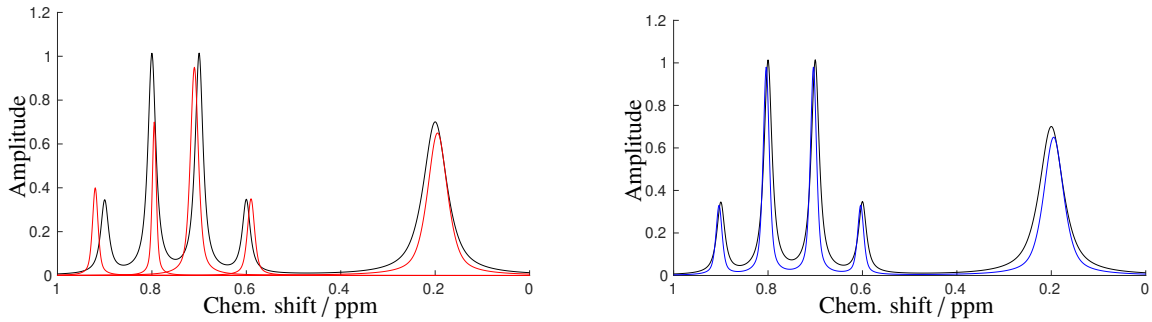

Figure 7: Parameter prediction in a single constructed subspectrum for a singlet and a quartuplet. The underlying spectrum is shown in black, whereas the peaks reconstructed from the predicted parameters are shown in red (left). After averaging and assigning the parameters to the multiplets, the final initial prediction is shown in blue (right).

Finally, the parameters need to be assigned to the correct peak. Since peak crossings may occur, the order of the peaks can change over time. Therefore, we again use the extrapolated peak positions and compare them to the predicted peak positions. In each step, we calculate the smallest distance between the remaining predicted and extrapolated centers and define the initial parameters of the corresponding peak accordingly. When using multiplets, this procedure is slightly more complex, because we have to calculate the positions of all subpeaks and deal with the fact that multiple peaks share the same multiplet parameters. In these cases we calculate the average of the corresponding predictions as our initial guess for the given multiplet.

If, for some reason no parameters from the CNNs are assigned to a peak or multiplet, then the extrapolated parameters are selected as a backup.

Consider the case of Fig. 7. In this subspectrum, a singlet and a quartuplet can be found. All networks try to predict the parameters and it turns out that the prediction of the 5-peak network has the smallest lack of fit with respect to the Frobenius norm. We then assign the parameters to the nearest peaks respectively and average all parameters belonging to the quartuplet. The result is shown on the right hand side of the figure.

3.4. Supplementary Material

The Code that implements the algorithm described above can be found at

`https://github.com/CITlabRostock/nmr_timeseries_model`.

It is divided into two parts. First, the code written in Python that was used to train the CNNs. Second, the code written in Matlab for creating models of NMR time series. Version requirements and further information can be found in the *ReadMe*.

4. Results

4.1. Data Sets

First, we introduce the experimental data sets.

Data set 1. Esterification of methanol (CH_3OH) and formic acid (HCOOH) to methyl formate ($HCOOCH_3$) and water. The data set is displayed at the top of Fig. 8. For further details, we refer to [2]. Therefore, only a brief description is given here. For the esterification of methanol ($\geq$ 0.998 mass-%, Sigma Aldrich) and methyl formate ($\geq$ 0.99 mass-%, Sigma Aldrich), 2 mass-% of sulfuric acid ($\geq$ 0.96 mass-%, Roth) was used as catalyst. For the acquisition of the spectra, a 400 MHz NMR spectrometer (magnet: Ascend 400, console: Avance 3 HD 400, Bruker) was used with single scans, a flip angle of 10° and an acquisition time of 1 s. The temperature during the reaction was 333 K, or about 60° C. A total of 199 spectra with 36,264 data points were recorded. The chemically relevant part of the data set contains slightly less spectra and fewer data points and we have $D_1 \in \mathbb{R}^{195\times 24577}$.

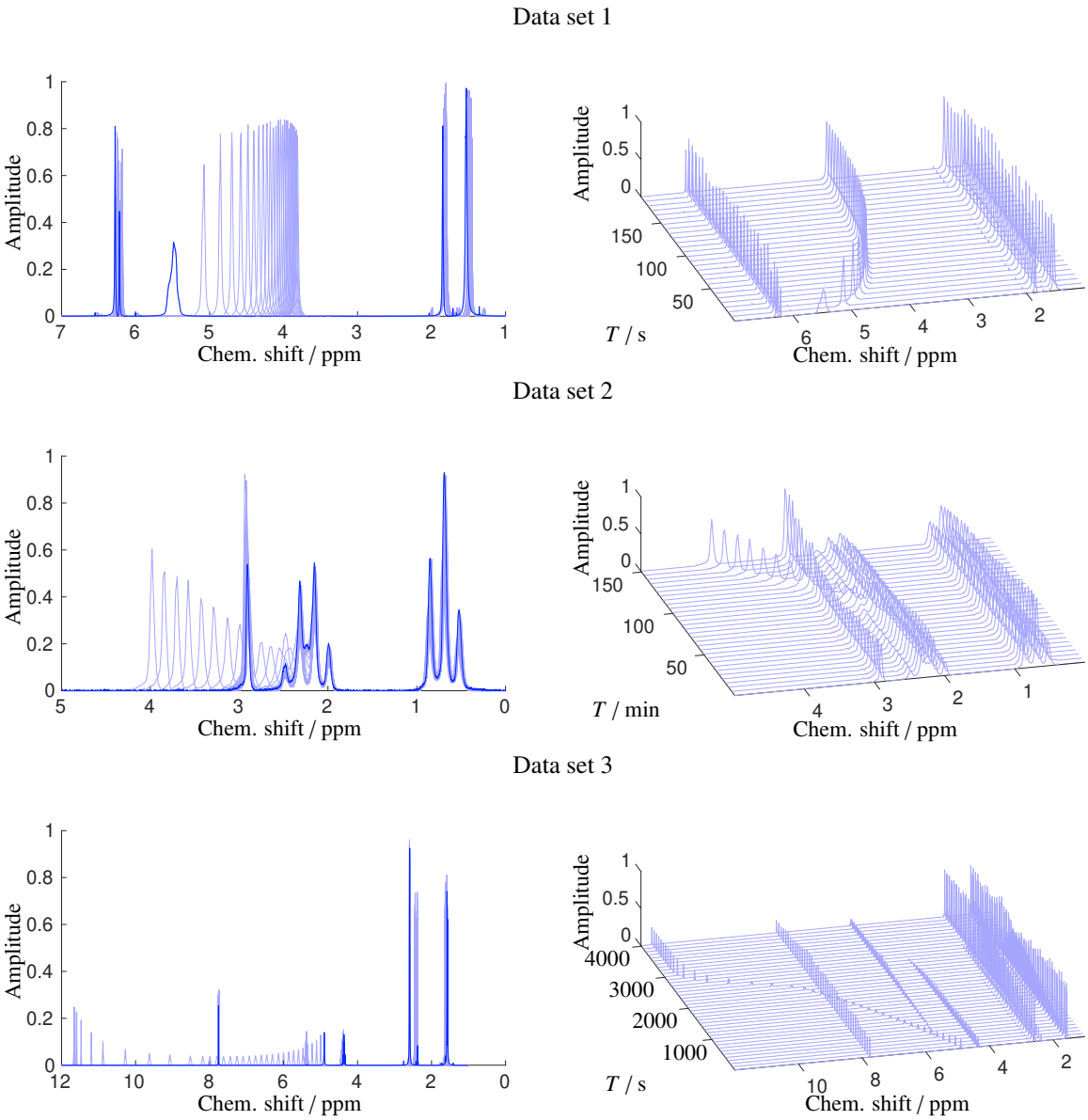

Figure 8: All experimental data sets as 2D plots on the left and as 3D plots on the right. Note the moving peaks in all data sets and the peak crossings in data sets two and three.

Data set 2. Batch distillation of methanol (CH_3OH) and diethylamine ($C_4H_{11}N$). The data set is shown in the middle row of Fig. 8. For further details we refer to [9]. Therefore, only a brief description is given here. For the isobaric batch distillation of methanol (≥99.8 mass-%, Carl Roth) and diethylamine (≥99 mass-%, Acros Organics) a glass batch distillation equipment was used. The composition of the liquid phase was analyzed online by a medium field NMR spectrometer (Spinsolve Carbon, Magritek) with a field strength of 1T corresponding to a proton Lamor frequency of 42.5 Mhz, which was connected to the batch distillation still by a sample loop. NMR spectra were recorded at 1 minute intervals with a single scan, an acquisition time of 3.2 s, and a pulse angle of 90°. In total, 151 spectra with ∼16,000 data points each were recorded. The relevant part of the measurement covers about 6001 data points, thus $D_2 \in \mathbb{R}^{151\times 6001}$.

Data set 3. Reaction of acetic anhydride and 2-propanol in the presence of H_2SO_4 to isopropylacetate and acetic acid. A detailed view on the data set is provided at the bottom of Fig. 8. An equimolar mixture of acetic anhydride (≥99 mass-%, Sigma Aldrich) and 2-propanol (≥99.9 mass-%, Merck KGaA) was prepared gravimetrically using a precision balance (Mettler Toledo, AG204) with a specified absolute uncertainty of ±0.0001 g. As an inert component, benzene (≥99.8 mass-%, AppliChem) was added to the reacting mixture. The reaction was accelerated by sulphuric acid (95–97 mass-%, J.T. Baker). The reactants were thoroughly mixed, then loaded into a 5 mm NMR tube, and placed in a high-field NMR spectrometer (magnet: Ascend 400, console: Avance 3 HD 400, Bruker) equipped with a 9.4 T vertical superconducting magnet corresponding to a proton Larmor frequency of 400.25 MHz. The temperature of the reacting mixture was approximately 25 °C. NMR spectra were recorded with a single scan, an acquisition time of 6 s and a pulse angle of 30°. A total of 680 spectra was recorded with ∼65,000 data points each. The relevant part of the measurement covers 48,055 data points, thus $D_3 \in \mathbb{R}^{680\times 48055}$.

4.2. Comparing the Initial Values

First, we visually compare the starting points for the optimization algorithm. In the previous version of the algorithm, they were given by linear extrapolation of the peak parameters from already optimized parameters of previous spectra. In the latest version, they are given by the predictions of the CNNs. We compare the starting points for two spectra from the constructed data set in Fig. 9. Similar comparisons for each data set can be found in Appendix A, see Figs. A.11, A.12, A.13.

In Fig. 9 we can see that extrapolating the parameters leads to initial estimates that differ greatly from the optimum in parameter space. A change in the rate of shift of the middle peak leads to significantly offset centers in the lower left plot, which are particularly difficult to deal with, as mentioned in Sec. 2.2.1. Furthermore, a decrease in peak width or height may be extrapolated to a negative value. Since these are chemically meaningless, the corresponding parameters must be reset to some positive value close to 0. However, this increases the potential error range. This can also be observed for the tiny peak in the bottom left plot of Fig. 9. On the other hand, the initial estimates of the neural networks are visually, and with respect to the Frobenius norm, close to the data and the optimizer shows stable convergence in a few steps. For $t = 14$, the optimizer stops prematurely after 250 steps in both cases. This is due to reaching the limit for the number of optimization steps. If we disable this stopping criterion, then the optimizer converges after exactly 500 steps for the initial prediction on the right hand side, and after 19,053 (!) steps for the initial prediction on the left hand side. Conversely, for the poor initial prediction on the bottom left, the optimizer converges after 80 steps, and after 35 steps on the bottom right.

The contrast in the number of optimization steps between top and bottom of Fig. 9 can be explained by the relative stopping criterion of the optimizer used. As it turns out, the optimizer terminates, when the reduction in lack of fit is smaller then some pre-defined tolerance multiplied with the current lack of fit, thereby creating a more precise stopping criterion when initialized with smaller lacks of fit. This fact even dampens the measurable effects of our version using neural networks. Still, the improvements are substantial, see Sec. 4.3.

4.3. Comparing Accuracy and Runtime

To compare the two versions of the algorithm, called non-linear optimization (NLO) and the hybrid method using neural networks, we use the following measurements. First, we compare the relative errors with respect to the Frobenius norm. This error, similar to (1), is defined as follows

$$\text{Rel. err.} = \frac{\|D - M\|_F}{\|D\|_F}. \tag{11}$$

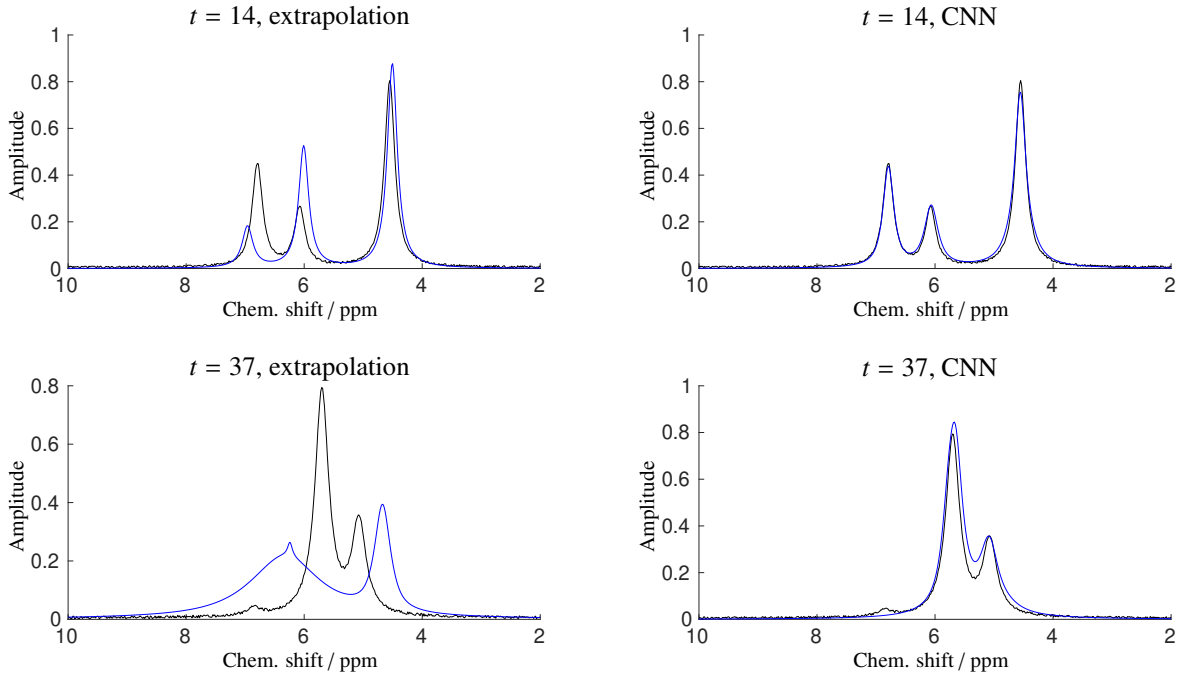

Figure 9: Comparing initial estimates from the constructed data set. The data spectrum is displayed in black, while the model constructed from the initial parameters is shown in blue.

Second, due to the many drawbacks of this Frobenius metric, we also compare the Wasserstein distance. For a better comparability between data sets, this distance is averaged over all spectra and divided by the length of the frequency axis. After normalization, we can think of this metric as the percentage of the length of the frequency axis that each pixel must be moved, on average, to transform the modeled spectrum into the data spectrum. Third, we measure the runtime of the entire program from start to finish. Fourth, we also sum up the time required for all optimization routines. Fifth, we count the number of steps of all optimization routines. Note that splitting the data set columnwise reduces the runtime, but increases the number of optimization steps. This is due to the reduced dimension of each optimization routine. Lastly, we show whether or not the optimization successfully traced all peaks through time.

The results shown in Table 1 have some key takeaways. First, the Frobenius norm alone is not useful for indicating whether or not the optimization has traced all peaks successfully, or how accurate the model is. The values of the relative error vary drastically between different data sets. Even when comparing the two versions on the same data set, there can be very similar values for a large difference in model accuracy, as can be seen in data sets 2 and 3.

	Constr. data set		Data set 1		Data set 2		Data set 3		Data set 3, split opt.	
Criterion \ Version	NLO	hybrid	NLO	hybrid	NLO	hybrid	NLO	hybrid	NLO	hybrid
Rel. err. (%, $\|\cdot\|_F$)	5.04	5.06	32.44	33.73	18.99	16.81	17.18	16.63	17.13	16.25
Wasserstein (%)	0.97	0.96	0.42	0.45	0.75	0.38	2.26	0.44	0.51	0.50
Runtime (s)	80.2	48.0	254.3	162.6	642.2	294.0	1248.5	1398.5	349.0	255.3
Opt. time (s)	78.9	46.7	233.2	140.4	638.7	289.9	1232.0	1374.1	324.4	224.7
Opt. steps (#)	782	516	125	98	314	185	446	554	2996	1699
Peak tracing (y/n)	yes	yes	yes	yes	no	yes	no	yes	yes	yes

Table 1: Comparison of the two modeling approaches. Pure nonlinear optimization (NLO) is in most cases outperformed by the CNN-based hybrid modeling approach.

Second, using the Wasserstein metric is more helpful in determining the quality of the models. Furthermore, it serves to compare model quality between different data sets. Still, data sets containing large and broad peaks, e.g. the constructed data set, are hard to compare with experimental data sets with their thinner and sparser peaks. We find that a properly accurate model has a normalized Wasserstein distance of $\leq 0.5\%$. This means that in order to transform the model into the data spectrum, we need to move each pixel under the model curve approximately $\leq 0.5\%$ of the length of the x-axis. If a given x-axis ranges from 0 to 10 ppm, the modeled peaks are on average about 0.05 ppm off.

Third, note that using the hybrid method generally improves stability and can reduce the runtime by up to 50%. Furthermore, splitting the data set beforehand further reduces runtime by a large factor, depending on the number of subsets. This can be seen in the last two columns. Note that after the peak tracing failed for data set 3, the optimization continued in less time, as some of the parameters did not affect the error function and were subsequently neglected. Therefore, the algorithm takes fewer steps to converge to a suboptimal local minimum.

4.4. Integrating the Modeled Peaks

We calculate the peak integrals of each individual peak using basic calculus.

$$\int_{-\infty}^{\infty} PV_{c,h,w,\lambda}(x)\mathrm{d}x = h \cdot \left((1-\lambda)\cdot w \cdot \pi + \lambda \cdot w \cdot \sqrt{\frac{\pi}{\ln(2)}}\right) \tag{12}$$

The advantage of using this formula is that we avoid numerical integration, which can lead to inaccurate results in the case of overlapping peaks. Since we know that individual multiplets belong to only one chemical substance, we can combine their integrals by adding the integrals of their composite peaks. To obtain the concentration profiles from the peak integrals requires further analysis with information about the chemical compounds. However, this may be as simple as knowing the number of nuclei corresponding to a given signal.

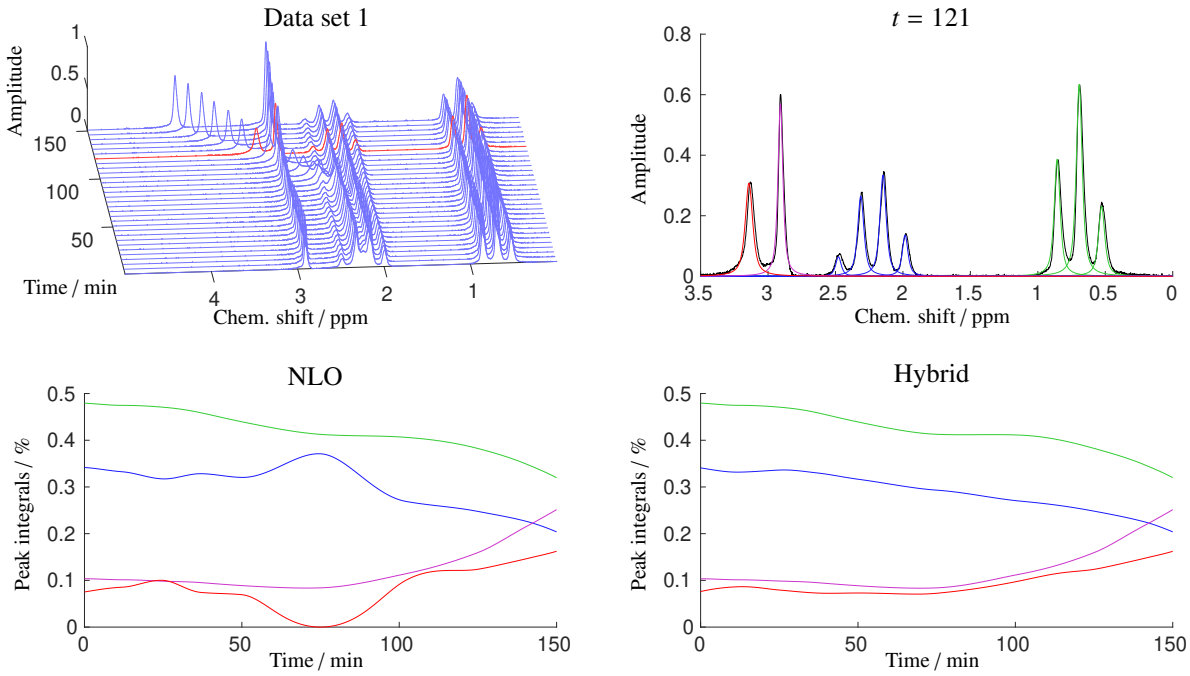

Figure 10: Data set 1 with a single spectrum showing the four multiplets (top right) and their corresponding integrals modeled in two different ways. The lower left plot shows the model obtained by NLO alone and the lower right plot shows the model obtained using the hybrid method.

In Fig. 10 we present four different integral curves; each corresponding to the normalized integrals of one of the four multiplets of data set 1. Note that during the peak overlaps ($t \in [50, 100]$), the peak integrals can be modeled differently without significantly increasing the error function. Nevertheless, the integral curves obtained by NLO do

not correspond to meaningful reaction kinetics. The behavior of the integral curves obtained by the hybrid method is smoother and each curve resembles an exponential growth or decay.

5. Conclusion

Using the suggested method to predict initial parameters and running a nonlinear optimization algorithm solves most of the problems that both methods suffer from on their own. An algorithm based on neural networks alone benefits from very fast runtimes, and models of entire data sets can reasonably be created in less than a minute. However, the ambiguities resulting from peak overlaps and the uncertainties that peak crossings infer cannot generally be resolved without further additions to the program. To accurately model moving and crossing peaks, we need to consider the movement of the peaks over time.

The approach of using only a nonlinear optimizer to minimize the lack of fit can generally handle such situations. However, due to the nature of the optimization function, the stability of convergence and the runtime are the two main problems of this approach. Both can be mitigated by minimizing the dimension of the optimization problem, which we do by using multiplets instead of peaks and by splitting the data set into smaller subsets. Crucially, a fast convergence speed and convergence into the correct optimum can be forced by initializing the optimizer close to the optimal point.

To provide a quick and accurate estimate for any given single spectrum, we use neural networks. This approach exploits the advantages of both methods - a quick and reasonably accurate estimate of the peak parameters in each spectrum - and the information provided by the time series and, finally, a sufficiently accurate model after optimizing. With this hybrid method we achieve satisfying results in terms of model quality and runtime. Compared to the approach presented in [20], where an accurate optimization of data set 3 took up to 24 hours, we achieve a more accurate model in about five minutes on the same computer hardware.

6. Acknowledgement

The authors are grateful to Prof. Holger Kösters, Institute of Mathematics, University of Rostock, for his useful advice on the Wasserstein metric.

Appendix A. Comparing starting points of the optimizer

We visually compare the initial predictions for all experimental data sets and discuss the advantages and disadvantages of peak detection using neural networks. The initial predictions for the constructed data set are displayed in Fig. 9

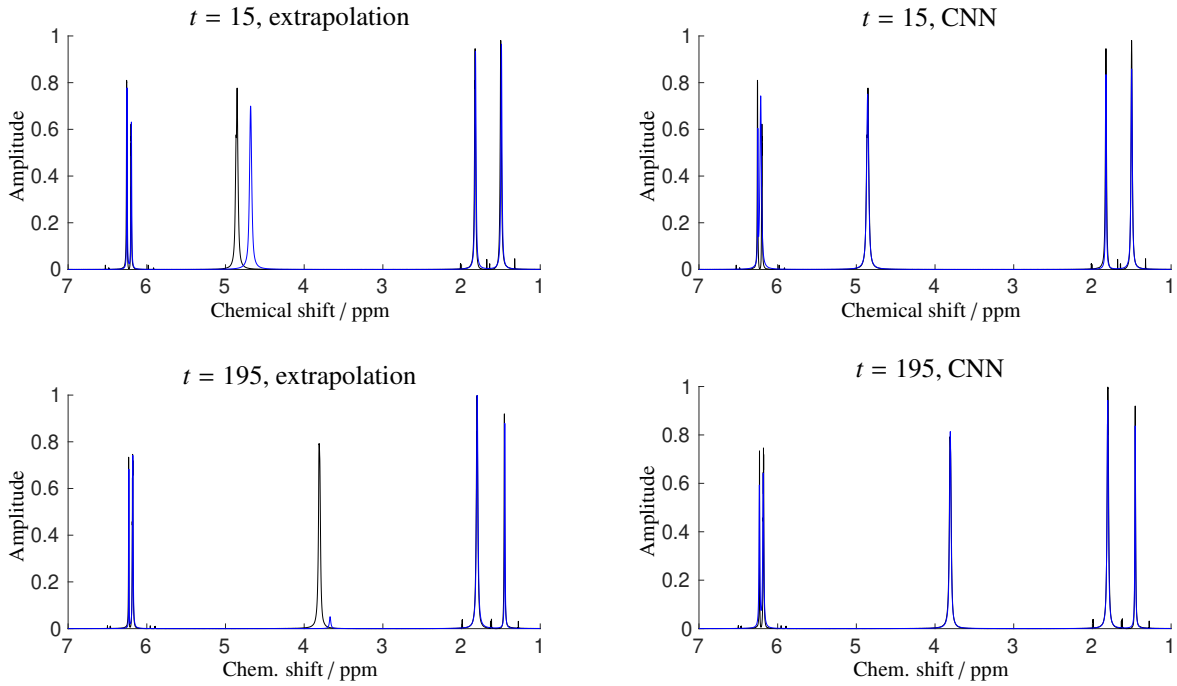

Figure A.11: Comparing initial estimates from two spectra of data set 1.

Data set 1. It can be seen that the linear extrapolation method struggles with the middle, moving peak, while the parameters of the stationary peaks on either side are properly initialized. Especially in the bottom left of Fig. A.11, we can see that linear extrapolation can lead to large differences in parameters, e.g., in height. The neural network approach predicts the moving peak much more accurately, while losing some accuracy in predicting the stationary peaks. Since both versions of the algorithm correctly trace all peaks, there is no significant difference in accuracy.

Data set 2. Due to the peaks in the latter spectra being very well separated, we optimize from back to front. On the left of Fig. A.12 we can see that the extrapolation approach struggles to correctly trace the moving peak. The initial prediction in the bottom left is comparatively close, but as the optimization progresses (with decreasing t), the center of the moving peak is lost and the extrapolation moves it even further to the right (top left picture), where no meaningful optimization can occur. On the other hand, the neural networks predict the peaks of the middle cluster slightly incorrectly (top right picture). However, the predictions are sufficiently close for the optimizer to correctly trace each peak through the entire data set. Again, the stationary peaks are not a problem for either method. Since the moving peak was incorrectly traced on the left, we see a significant difference in the Wasserstein distance from the linear extrapolation version (0.75 %) to the hybrid version (0.38 %). The relative error with respect to the Frobenius norm differs only slightly (18.99 % vs. 16.81 %). In this case, the improvement in accuracy also coincides with a large improvement in runtime. Running the improved algorithm takes only half the time.

Data set 3. Again, in Fig. A.13 it can be seen that the version using linear extrapolation on the left struggles to obtain correct initial parameters for the moving peak at about 5.8 ppm in the top left and at around 9.6 ppm in the bottom left. While the moving peak is correctly traced for $t \in [1, 100]$, the algorithm loses track of it in the second

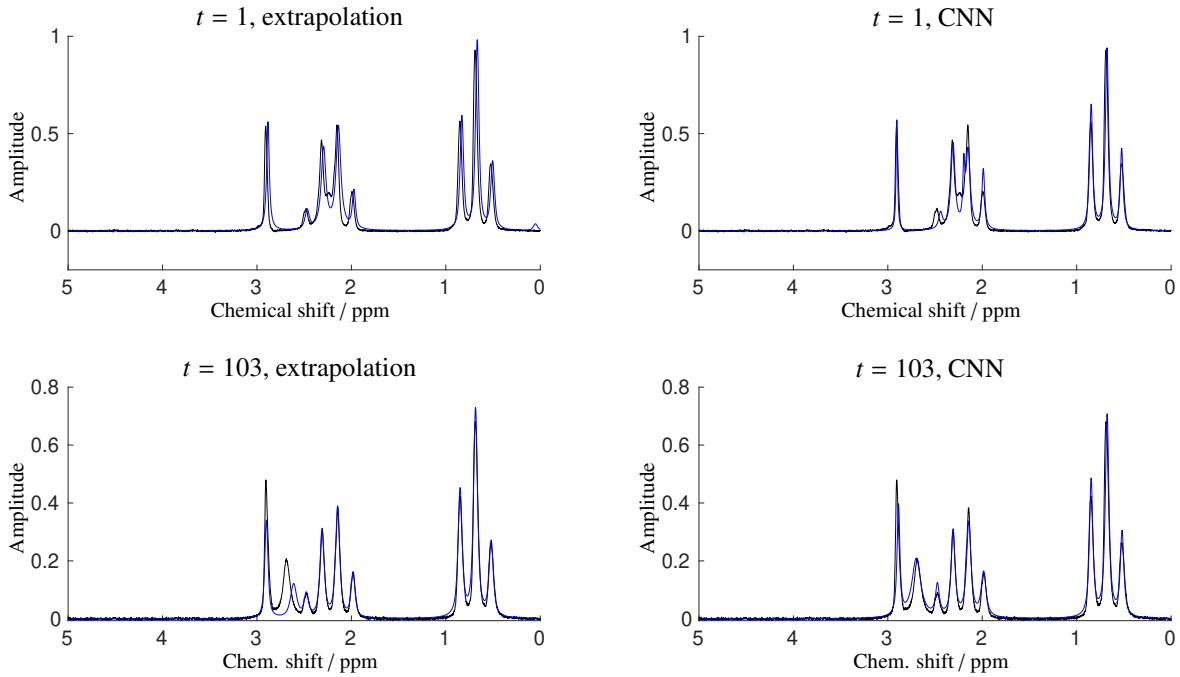

Figure A.12: Comparing initial estimates from two spectra of data set 2.

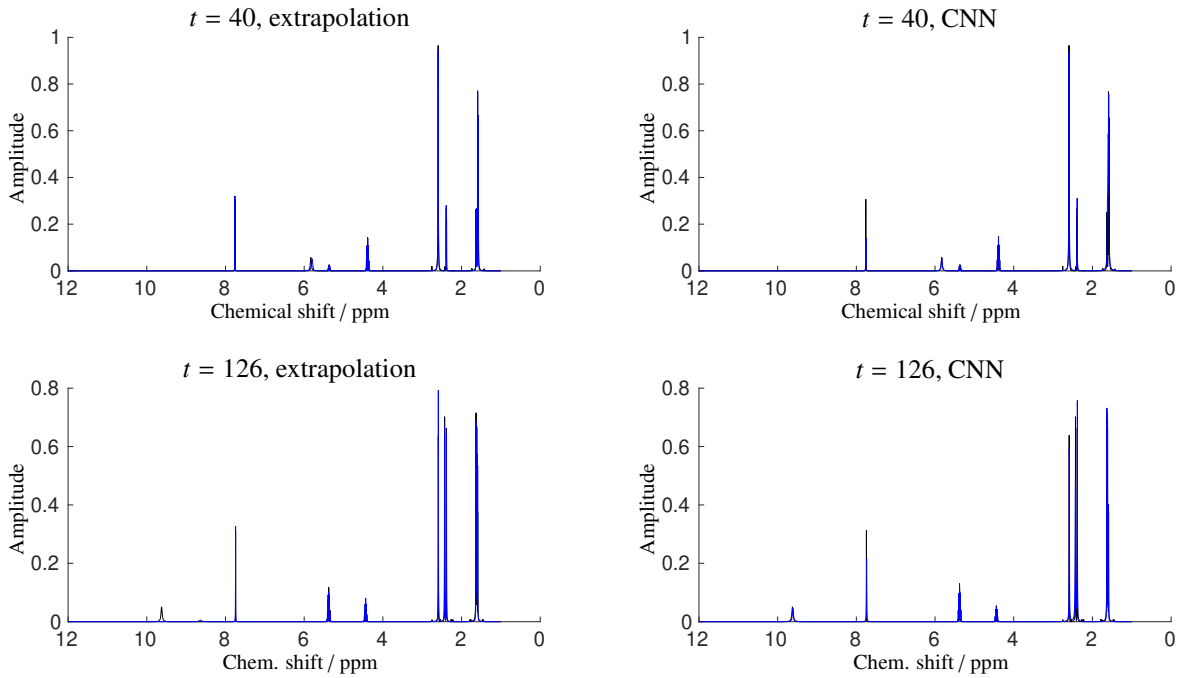

Figure A.13: Comparing initial estimates from two spectra of data set 3.

half of the data set, where $t \in [101, 204]$. This fact can be measured by the Wasserstein metric, which equals 2.26 % compared to the correct peak tracing of the hybrid version and its metric of 0.44 %. Even though the Wasserstein metric differs by a factor of 5, the relative error only differs by a factor of $17.18/16.81 = 1.03$. Losing one peak position did have a positive effect on the runtime (1232.0 vs. 1374.1 s). Since changing the parameters of a peak whose position is far from correct does not significantly change the lack of fit, the height and / or half width ultimately vanish and optimization essentially continues with four fewer parameters, leading to quicker numerical convergence.

Appendix B. Additional Information on the Wasserstein Metric

The Wasserstein metric is usually defined as follows:

Definition Appendix B.1. *Let μ, ν be probability measures on $\mathbb{R}$. Then,*

$$W_1(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \int_{\mathbb{R}\times\mathbb{R}} |x-y| \mathrm{d}\gamma(x,y), \tag{B.1}$$

defines the one-dimensional p-Wasserstein metric for $p = 1$, where $\Gamma(\mu,\nu)$ is the set of all measures on $\mathbb{R}\times\mathbb{R}$, fulfilling

$$\gamma(B\times\mathbb{R}) = \mu(B) \qquad \textit{and} \qquad \gamma(\mathbb{R}\times B) = \nu(B)$$

for all measureable sets $B \subseteq \mathbb{R}$.

For a more general definition, see [5, 22]. In [22][Prop. 1], it is proved that the Wasserstein metric has the following form

$$W_1(\mu,\nu) = \int_0^1 \left|F^{-1}(z) - G^{-1}(z)\right| \mathrm{d}z \tag{B.2}$$

using the quantile functions of the cumulative distribution functions F of μ and G of ν. Following Prob. 2 in Ch. 11 of [5], we obtain the following lemma.

Lemma Appendix B.2. *Define μ, ν, F, G similar to Eqs.* (B.1) *and* (B.2)*. Then,*

$$W_1(\mu,\nu) = \int_{\mathbb{R}} |F(x) - G(x)| \,\mathrm{d}x \tag{B.3}$$

holds.

Proof. The integral of Eq. B.2 is equal to the area in between both quantile functions. Let λ_2 denote the two-dimensional Lebesgue measure and define

$$\begin{aligned}
M_q &:= \left\{(x,z) \in \mathbb{R}\times[0,1] \mid \min\left\{F^{-1}(z), G^{-1}(z)\right\} \le x \le \max\left\{F^{-1}(z), G^{-1}(z)\right\}\right\} \\
N_q &:= \{(x,z) \mid x = \max\{F^{-1}(z), G^{-1}(z)\}\} \\
M_p &:= \{(x,z) \in \mathbb{R}\times[0,1] \mid \min\{F(x), G(x)\} \le z \le \max\{F(x), G(x)\}\} \\
N_p &:= \{(x,z) \mid z = \min\{F(x), G(x)\}\}.
\end{aligned}$$

Note that N_q and N_p are both λ_2 null sets.

We can assume without loss of generality that for any point $(x,z) \in M_q \setminus N$, we have $F^{-1}(z) \le x < G^{-1}(z)$. Since $F^{-1}(z) \le x$, by definition of the quantile function and monotonicity of F we obtain $F(x) \ge z$. Additionally, since $G^{-1}(z) > x$, we obtain that $G(x) < z$. Therefore, every point $(x,z) \in M_q \setminus N_q$ must be an element of M_p.

Conversely, if $(x,z) \in M_p \setminus N_p$, we again assume without loss of generality that $F(x) < z \le G(x)$. The first inequality implies $F^{-1}(z) > x$ and the second inequality implies $G^{-1} \le x$. Therefore, $(x,z) \in M_q$. Finally, we can easily see that the integral in Eq. (B.2) can be written as $\lambda_2(M_q)$ and that the integral in Eq. (B.3) can be written as $\lambda_2(M_p)$. Finally, we have

$$\begin{aligned}
W_1(\mu,\nu) = \lambda_2(M_q) = \lambda_2(M_q \setminus N_q) \le \lambda_2(M_p) \quad \text{and} \\
\lambda_2(M_p) = \lambda_2(M_p \setminus N_p) \le \lambda_2(M_q) = W_1(\mu,\nu),
\end{aligned}$$

completing the proof. □

We then obtain Eq. (3) after discretizing Eq. (B.3). The cumulative density functions can be computed as the first summation index increases, resulting in a linear computation time.

References

[1] F. Alsmeyer, H.-J. Koß, and W. Marquardt. Indirect spectral hard modeling for the analysis of reactive and interacting mixtures. *Appl. Spectrosc.*, 58(8):975–985, 2004.

[2] A. Brächer, R. Behrens, E. von Harbou, and Hasse H. Application of a new micro-reactor 1H NMR probe head for quantitative analysis of fast esterification reactions. *Chem. Eng. J.*, 306:413–421, 2016.

[3] D. Chen, Z. Wang, D. Guo, V. Orekhov, and X. Qu. Review and Prospect: Deep Learning in Nuclear Magnetic Resonance Spectroscopy. *Chem. Eur. J.*, 26(46):10391–10401, 2020.

[4] B. Domżał, E. K. Nawrocka, D. Gołowicz, M. A. Ciach, B. Miasojedow, K. Kazimierczuk, and A. Gambin. Magnetstein: An Open-Source Tool for Quantitative NMR Mixture Analysis Robust to Low Resolution, Distorted Lineshapes, and Peak Shifts. *Anal. Chem.*, 96(1):188–196, 2023.

[5] R. M. Dudley. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2002.

[6] G. Fischetti, N. Schmid, S. Bruderer, G. Caldarelli, A. Scarso, A. Henrici, and D. Wilhelm. Automatic classification of signal regions in 1H Nuclear Magnetic Resonance spectra. *Front. Artif. Intell.*, 5:1116416, 2023.

[7] F. Fricke, M. Brandalero, S. Liehr, S. Kern, K. Meyer, S. Kowarik, R. Hierzegger, S. Westerdick, M. Maiwald, and M. Hübner. Artificial Intelligence for Mass Spectrometry and Nuclear Magnetic Resonance Spectroscopy Using a Novel Data Augmentation Method. *IEEE Trans. Emerg. Top. Comput.*, 10(1):87–98, 2021.

[8] A. Friebel, E. von Harbou, K. Münnemann, and H. Hasse. Reaction monitoring by benchtop NMR spectroscopy using a novel stationary flow reactor setup. *Ind. Eng. Chem. Res.*, 58(39):18125–18133, 2019.

[9] A. Friebel, E. von Harbou, K. Münnemann, and H. Hasse. Online process monitoring of a batch distillation by medium field NMR spectroscopy. *Chem. Eng. Sci.*, 219:115561, 2020.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, 2016. `http://www.deeplearningbook.org`.

[11] E.L. Hahn and D.E. Maxwell. Spin Echo Measurements of Nuclear Spin Coupling in Molecules. *Phys. Rev.*, 88(5):1070, 1952.

[12] J. Hellwig and K. Neymeyr. On the uniqueness of continuous and discrete hard models of NMR-spectra. *Preprint, J. Math. Chem.*, 2024.

[13] S. Kern, S. Liehr, L. Wander, M. Bornemann-Pfeiffer, S. Müller, M. Maiwald, and S. Kowarik. Artificial Neural Networks for Quantitative Online NMR Spectroscopy. *Anal. Bioanal. Chem.*, 412:4447–4459, 2020.

[14] H.-W. Koh. *Feature extraction in NMR data analysis*. PhD thesis, TU Dortmund, Fachbereich Informatik, 2010.

[15] D.-W. Li, A.L. Hansen, C. Yuan, L. Bruschweiler-Li, and R. Brüschweiler. DEEP picker is a deep neural network for accurate deconvolution of complex two-dimensional NMR spectra. *Nat. Commun.*, 12(1):5229, 2021.

[16] Z. Liu, H. Sharma, J.-S. Park, P. Kenesei, A. Miceli, J. Almer, R. Kettimuthu, and I. Foster. BraggNN: fast X-ray Bragg peak analysis using deep learning. *IUCrJ*, 9(1):104–113, 2022.

[17] W.F. Maddams. The scope and limitations of curve fitting. *Appl. Spectrosc.*, 34(3):245–267, 1980.

[18] M. Maiwald, H. H. Fischer, Y.-K. Kim, K. Albert, and H. Hasse. Quantitative high-resolution on-line NMR spectroscopy in reaction and process monitoring. *J. Magn. Reson.*, 166(2):135–146, 2004.

[19] T. Maschmeyer, P. L. Prieto, S. Grunert, and J. E. Hein. Exploration of continuous-flow benchtop NMR acquisition parameters and considerations for reaction monitoring. *Magn. Reson. Chem.*, 58(12):1234–1248, 2020.

[20] D. Meinhardt, H. Schröder, J. Hellwig, E. Steimers, A. Friebel, T. Beweries, M. Sawall, E. von Harbou, and K. Neymeyr. Model-based signal tracking in the quantitative analysis of time series of NMR spectra. *J. Magn. Reson.*, 339:107212, 2022.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. et al. Antiga. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[22] A. Ramdas, N. García Trillos, and M. Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.

[23] D.A. Ramsay. Intensities and shapes of infrared absorption bands of substances in the liquid phase. *J. Am. Chem. Soc.*, 74(1):72–80, 1952.

[24] A. Savitzky and M.J.E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.*, 36(8):1627–1639, 1964.

[25] N. Schmid, S. Bruderer, F. Paruzzo, G. Fischetti, G. Toscano, D. Graf, M. Fey, A. Henrici, V. Ziebart, B. Heitmann, et al. Deconvolution of 1D NMR spectra: a deep learning-based approach. *J. Magn. Reson.*, 347:107357, 2023.

[26] J. Shen, Y. Qu, W. Zhang, and Y. Yu. Wasserstein distance guided representation learning for domain adaptation. *Proc. AAAI Conf. Art. Intell.*, 32(1), 2018.