Introducing *KIPET*: A novel open-source software package for kinetic parameter estimation from experimental datasets including spectra

C. Schenk[a,**], M. Short[a,**],
J. S. Rodriguez[b], D. Thierry[a], L. T. Biegler[a,*], S. García-Muñoz[c], W. Chen[d]

[a] *Carnegie Mellon University, Pittsburgh, PA 15213*
[b] *Purdue University, West Lafayette, IN 47907*
[c] *Eli Lilly and Company, Indianapolis, IN 46285*
[d] *Zhejiang University of Technology, Hangzhou 310023, P.R. China*

Abstract

This paper presents *KIPET* (Kinetic Parameter Estimation Toolkit) an open-source toolbox for the determination of kinetic parameters from a variety of experimental datasets including spectra and concentrations. *KIPET* seeks to overcome limitations of standard parameter estimation packages by applying a unified optimization framework based on maximum likelihood principles and large-scale nonlinear programming strategies for solving estimation problems that involve systems of nonlinear differential algebraic equations (DAEs). The software package includes tools for data preprocessing, estimability analysis, and determination of parameter confidence levels for a variety of problem types. In addition *KIPET* introduces informative wavelength selection to improve the lack of fit. All these features have been implemented in Python with the algebraic modeling package *Pyomo*. *KIPET* exploits the flexibility of *Pyomo* to formulate and discretize the dynamic optimization problems that arise in the parameter estimation algorithms. The solution of the optimization problems is obtained with the nonlinear solver *IPOPT* and confidence intervals are obtained through the use of either *sIPOPT* or a newly developed tool, *k_aug*. The capabilities as well as ease of use of *KIPET* are demonstrated with a number of examples.

Keywords: Kinetic Parameter Estimation, Differential Algebraic Equations, Spectroscopic Data, Pharmaceutical Processes, Chemical Processes, Chemometrics

1. Introduction

The determination of accurate reaction kinetics is imperative in many industries to ensure safe, controllable, and scalable processes. In research-based chemical industries, particularly

*Corresponding author
**Co-first authors.
Email addresses: schenk@cmu.edu (C. Schenk), shortm@andrew.cmu.edu (M. Short), rodri324@purdue.edu (J. S. Rodriguez), dmolinat@andrew.cmu.edu (D. Thierry), biegler@cmu.edu (L. T. Biegler), sal.garcia@lilly.com (S. García-Muñoz), wfchen@zjut.edu.cn (W. Chen)

the pharmaceutical industry during the early development phase, little is known about the reaction being studied in advance and it is difficult to determine the chemical species present, reaction mechanisms, and kinetic parameters involved from obtained experimental datasets. Since many of the experiments are costly to run, it is important to develop tools that are able to maximize the information obtained from each experiment. Typical data types that are collected during experiments include spectroscopic data (infrared, near-infrared, ultraviolet-visible, Raman etc.), as well as high-performance liquid chromatography (HPLC), ultra-performance liquid chromatography (UPLC) and calorimetric data. This section will focus on research around the challenges regarding spectroscopic measurements. However, the software presented in the rest of this paper can be extended to estimate parameters from other measurement types as well.

For estimates from spectroscopic data, self-modeling curve resolution techniques can be used to obtain concentration profiles and pure species' absorbance profiles through a bilinear decomposition. Lawton and Sylvestre (1971) introduced Beer-Lambert's law into spectroscopic multivariate curve resolution (MCR) techniques, thereby providing physical meaning to this bilinear decomposition (Lawton and Sylvestre, 1971a,b). Beer-Lambert's law states that the profiles of the pure components are related through:

$$\boldsymbol{D} = \boldsymbol{C}\boldsymbol{S}^{\mathrm{T}} + \boldsymbol{E}, \tag{1}$$

where the data matrix $\boldsymbol{D}$ is of dimension $ntp \times nwp$ with measured time points $t_i, i = 1..., ntp$ and measured wavelengths $\lambda_l, l = 1..., nwp$, $\boldsymbol{C}$ is the molar concentration matrix of dimension ntp x nc with species $c_k, k = 1, ..., nc$, where nc denotes the number of components, and $\boldsymbol{S}$ is the matrix of dimension nc x nwp. $\boldsymbol{E}$ is then a ntp x nwp matrix containing the residual variations in the data. Their approach was to attempt to obtain the estimates of both $\boldsymbol{C}$ and $\boldsymbol{S}$ simultaneously using what is referred to as a soft-modeling approach. By adding non-negativity constraints on $\boldsymbol{C}$ and $\boldsymbol{S}$ the approach guarantees physical meaning and somewhat reduces rotational ambiguity. This class of approaches also contains other bilinear decomposition methods including Principal Component Analysis (PCA), which provides a more abstract decomposition of the data, maximizing the explained covariance while constraining the orthonormality of the components (Massart et al., 1997; Jackson, 1991). A limitation of these approaches is that permutation, intensity, and rotational ambiguity result in solutions that cannot be guaranteed to be unique. Additionally, since no model is postulated, it is difficult to determine whether the solution is a feasible representation of the data.

Another approach to MCR to determine kinetic parameters, is based upon the Gauss-Newton-Levenberg-Marquardt (GNLM) approach and often referred to as a hard-modeling approach (Levenberg, 1944; Marquardt, 1963). In this method, a reaction model is proposed, kinetic parameters initialized, and then a system of equations is integrated. This gives initial concentration profiles of the species, which are used to obtain the pure component absorption profiles from the data matrix by least squares regression. The GNLM is then used to update the kinetic parameter values based on the residuals and derivative information with respect to the kinetic parameters (Bijlsma et al., 2000, 2001; Puxty et al., 2006). This procedure is repeated until convergence. Due to the large size of the data matrices, problem reduction approaches based on factor analysis have been proposed which have proven to be reliable and have thus been commonly applied. An

example of this was proposed by Maeder and Zuberbühler (1990), where $\boldsymbol{D}$ and $\boldsymbol{S}$ are projected into the subspace spanned by the right-singular vectors of $\boldsymbol{D}$.

Instead of using a hard- or soft-modeling approach, a hybrid approach was proposed by De Juan et al. (2000), whereby a kinetic model is incorporated into MCR-ALS (MCR-Alternating Least Squares). Here, the least squares calculation of the concentration matrix is performed using soft constraints and the kinetic model is then fitted to this matrix to obtain kinetic parameters. This results in a new concentration matrix that can be used instead of the one modeled using soft constraints which, in turn, can be used to obtain the pure species absorption profiles under soft constraints. This procedure is repeated until a lack-of-fit below a certain threshold is obtained. This methodology has found wide-spread academic use, however it is not commonly used in industrial applications due to the slow convergence and challenges pertaining to large datasets (Chen et al., 2018).

Neymeyr et al. (2010) proposed an alternative method, called pure component decomposition (PCD), where singular value decomposition (SVD) is first used in order to obtain a low-rank approximation of $\boldsymbol{D}$ and then a rotation matrix is introduced. An objective function which minimizes the error between the approximation and original $\boldsymbol{D}$ matrix is chosen for minimization. Instead of adding constraints for non-negativity penalties are introduced into the objective function with weight factors. Their implementation uses numerous optimization techniques to solve this regularized system for the rotation matrix, which provides the factorization for the $\boldsymbol{D}$ matrix. Furthermore, Sawall et al. (2012) proposed an extension to PCD through combination with a reaction model, whereby the error between the model concentrations and the concentration matrix are used to regularize the objective function (Sawall et al., 2012). This approach, which they called a model-free approach, can obtain kinetic parameters and the rotation matrix simultaneously, however there are many associated weighting factors that need to be determined for the regularization terms, which need to be estimated without proposed guidelines. All of the methods mentioned above suffer from slow convergence and poor performance when instability or ill-conditioning are present in the dynamic system. These methods will fail whenever there are dependent columns in the concentration matrix, which is not an uncommon occurrence, because the $\boldsymbol{C}^T\boldsymbol{C}$ matrix is required to be non-singular. Additionally, these methods do not consider system noise in the reaction model, nor do they provide convergence guarantees. More details of these sequential approaches are presented by Ruckebusch and Blanchet (2013); Golshan et al. (2016). In order to overcome these disadvantages, an entirely new approach was presented by Chen et al. (2016).

The proposed alternative method to these hard- and soft-modeled approaches is to simultaneously obtain the reaction kinetic parameters with the curve resolution. This framework, which will be discussed in more detail in Section 2, makes use of maximum likelihood principles, collocation methods, and large-scale nonlinear programming (NLP). The two steps involved in this framework include a noise deconvolution step, to estimate the measurement noise separately from the model variable noise, followed by a simultaneous parameter estimation that also provides confidence intervals based on the optimal solution sensitivities. This method has been tested extensively, and has been proven to be successful in obtaining robust solutions for a number of different scenarios. Furthermore its capabilities can be extended to address more complex problems (Chen et al., 2018, 2019).

While the number of methodologies shown here are effective under certain assumptions and with certain user skill levels, there are no standard toolkits or methods applied across the chemical and pharmaceutical industries for obtaining kinetic parameters from spectra. A major disadvantage of current approaches is that practitioners are forced to use estimates or only small subsets of data to design further experiments and make predictions about the reaction systems. A typical example of one of the most common methods would be to identify known peaks of certain species, track these peaks to estimate the concentration of those species, and then use these species concentration profiles with least squares estimation to determine the reaction kinetics. While these approaches can be informative in the absence of other methods, they fail to make use of all of the valuable information that can be collected during experiments, and may not provide much information regarding the fundamental reaction mechanisms.

The freely available MCR-ALS GUI2.0 (Jaumot et al., 2015), is available for MATLAB and as a Python package called "pyMCR". This software is an updated implementation of the previously discussed MCR-ALS technique, but therefore also has the same shortcomings mentioned previously.

In the open-source domain, toolkits such as ACADOs (Verschueren et al., 2018) and CasADi (Andersson et al., 2018) provide many useful features for general parameter estimation problems and sensitivity analysis of solutions, but do not provide kinetic parameter estimation from spectra explicitly. The fairly recent HyperSpy can be used to develop hard- and soft-modeling approaches for the estimation of kinetic parameters from spectra, however these require detailed knowledge of the problem and Python programming skills in order to successfully write custom estimation code (de la Peña et al., 2018). HyperSpy is mostly meant for data visualization and manipulation purposes.

On the other hand, commercial software packages such as gPROMS are used by many practitioners for their appealing user interface, ready-made and easily implemented solution strategies, and customer service, however these packages do not provide methods to directly incorporate spectra for simultaneous parameter estimation; rather using implicit measurements as mentioned above. gPROMS allows users to test candidate models sequentially, based on these direct responses and then allows users to validate their model by providing statistical tools (Process Systems Enterprise, 1997-2018).

Similarly, the GREG and GREGPLUS (General regression) tools from Athena Visual Studio can be used for nonlinear parameter estimation and process model analysis from single- or multi-response data (Stewart et al., 1992; Stewart and Caracotsios, 2008). The software is mainly used for model discrimination and includes a number of different objective function formulations for users to choose from as well as tools for estimability based on the rival models and datasets, but cannot deal directly with spectra.

With a lack of software for the treatment of kinetic parameter estimation problems from spectra, combined with the proliferation of multivariate data collection techniques for batch reaction systems and the relatively new and highly adaptable formulation from Chen et al. (2016), there exists an opportunity to create a new, multi-use tool that can be easily extended and manipulated to perform a wide-array of parameter estimation strategies. Many industrial practitioners combine the commercial tools above with a large number of custom codes in different modeling environments to solve their problems. We present a new open-source framework that industrial and research practitioners can utilize to perform a wide-range of tasks related to the MCR problem.

The remainder of this paper is set out as follows: the next section will detail the methodology behind *KIPET*, an open-source Python package for parameter estimation from spectroscopic and other data sources and will give an overview of the software package itself with a specific focus on the various classes and methods. A number of useful functionalities are discussed. Section 3 contains case studies that demonstrate the proposed approach, with sample code and results directly from the software. Finally, we present our conclusions and planned future work on the package.

2. Methodology

The *KIPET* framework is based on the model formulations and concepts developed in Chen et al. (2016). In this section we review the most important features of their work and the algorithms and methods implemented in the toolbox. It should be noted, before beginning this section, that *KIPET* assumes the reaction system is a perfectly mixed batch or fed-batch reaction system which obeys Beer-Lambert's law.

2.1. Simultaneous Kinetic Parameter Estimation

To describe the formulation we begin by rewriting Beer-Lambert's law introduced in eq. (1) in scalar form:

$$d_{i,l} = \sum_{k=1}^{nc} c_k(t_i) s_k(\lambda_l) + \zeta_{i,l}, \qquad i = 1,..,ntp,\ l = 1,..,nwp \tag{2}$$

where $d_{i,l}$ is spectroscopic data obtained at a sampling time t_i and wavelength λ_j, $c_k(t_i)$ is the concentration of species k at sampling time t_i, $s_k(\lambda_l)$ is the absorbance of species k at wavelength λ_j, ntp and nwp are the total number of time points and wavelengths, and $\zeta_{i,l}$ accounts for measurement error. Multivariate curve resolution techniques that follow solely Beer-Lambert's law are known to suffer from permutation, intensity, and rotational ambiguity which cause non-uniqueness in the parameter estimation procedure. To overcome these limitations Chen et al. (2016) proposed solving the parameter estimation problem within an optimization framework that complements eq. (2) with a reaction kinetic model of the form:

$$\begin{aligned} &\frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{f}(\boldsymbol{z}(t), \boldsymbol{y}(t), \boldsymbol{\theta}), \\ &\boldsymbol{g}(\boldsymbol{z}(t), \boldsymbol{y}(t)) = 0, \\ &\boldsymbol{z}(t_0) = \boldsymbol{z_0}, \end{aligned} \tag{3}$$

where $\boldsymbol{z}(t) \in \mathbb{R}_+^{nc}$ is the vector of concentrations predicted by the kinetic model with nc as the total number of components, $\boldsymbol{y}(t) \in \mathbb{R}^{na}$ the vector of algebraic variables (e.g., reaction rates) with na as the total number of algebraic variables, $\boldsymbol{z_0} \in \mathbb{R}_+^{nc}$ is the vector of initial concentrations, and $\boldsymbol{\theta} \in \mathbb{R}^{n\theta}$ the kinetic parameters to be determined in the optimization framework with $n\theta$ as the number of to be estimated kinetic parameters. The kinetic model functions $\boldsymbol{f} : \mathbb{R}^{nc+ny+n\theta} \to \mathbb{R}^{nf}$ with nf as the number of right hand sides of the differential equations and $\boldsymbol{g} : \mathbb{R}^{nc+ny+n\theta} \to \mathbb{R}^{ng}$ with ng as the number of algebraic equations, are assumed to be twice continuously differentiable with respect to the optimization variable $(\boldsymbol{z}, \boldsymbol{y}, \boldsymbol{\theta})^T$. Since the predictions of eq. (3) are subject to model error,

and rarely exactly match experimental values, Chen et al. (2016) proposed to relate the predicted concentration $\boldsymbol{z}(t)$ with the concentration observed experimentally with the following equation:

$$c_k(t_i) = z_k(t_i) + \omega_k(t_i), \quad i = 1,...ntp, \quad k = 1,...,nc \tag{4}$$

where ω_k represents model error. Combining eq. (2), system (3) and eq. (4) leads to a DAE-constrained optimization problem of the following form:

$$\begin{aligned}
\min \quad & \frac{1}{\delta^2}\sum_{i=1}^{ntp}\sum_{l=1}^{nwp}\left(d_{i,l} - \sum_{k=1}^{nc} c_k(t_i)s_k(\lambda_l)\right)^2 + \sum_{i=1}^{ntp}\sum_{k=1}^{nc}\frac{1}{\sigma_k^2}(c_k(t_i) - z_k(t_i))^2 \\
\text{s.t.} \quad & \frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{f}(\boldsymbol{z}(t),\boldsymbol{y}(t),\boldsymbol{\theta}) \\
& \boldsymbol{g}(\boldsymbol{z}(t),\boldsymbol{y}(t)) = 0 \\
& \boldsymbol{z}(t_0) = \boldsymbol{z_0} \\
& \boldsymbol{c}(t_i) \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad i = 1,..,ntp \\
& \boldsymbol{s}(\lambda_l) \geq 0 \qquad (optional) \qquad\qquad\qquad l = 1,..,nwp,
\end{aligned} \tag{P}$$

Throughout the paper this optimization problem is referred to as Problem (P) and it is highlighted as the main formulation solved in *KIPET*. Problem (P) enforces the kinetic model (3) in the constraints while minimizing measurement and model error in the objective function. The first term minimizes deviations from Beer-Lambert's law and the second term minimizes deviations from the kinetic model – i.e., $\zeta_{i,l}$ and ω_k are minimized implicitly. The variables in the optimization problem are the kinetic parameters $\boldsymbol{\theta}$, the species' unnoised and noised concentrations, $\boldsymbol{z}(t)$ and $\boldsymbol{c}(t_i)$ respectively, the species' absorbances $\boldsymbol{s}(\lambda_l)$, and the algebraic variables $\boldsymbol{y}(t)$. Note that the degrees of freedom of Problem (P) are determined by the number of kinetic parameters. The input data for Problem (P) are the spectra $d_{i,l}$ values, and the variances δ^2 and σ_k^2. While spectra are typically available from experiments, the variance values are not. Therefore, to solve Problem (P), Chen et al. (2016) first solve an iterative optimization-based heuristic to estimate the variances. The heuristic procedure solves a sequence of three optimization subproblems. In their work, Chen et al. (2016) present a detailed derivation of each subproblem using maximum likelihood principles and Beer-Lambert's law. Here the steps of this approach are summarized and the inputs and outputs of each optimization subproblem are discussed. The process begins by solving the concentration-based optimization problem:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{nc} \ln\left(\frac{1}{ntp}\sum_{i=1}^{ntp}(c_k(t_i) - z_k(t_i))^2\right) \\
\text{s.t.} \quad & \frac{d\boldsymbol{z}(t)}{dt} = \boldsymbol{f}(\boldsymbol{z}(t),\boldsymbol{y}(t),\boldsymbol{\theta}) \\
& \boldsymbol{g}(\boldsymbol{z}(t),\boldsymbol{y}(t)) = 0 \\
& \boldsymbol{z}(t_0) = \boldsymbol{z_0} \\
& \boldsymbol{z}(t) \geq 0.
\end{aligned} \tag{Pa}$$

We refer to this first subproblem as Problem (Pa). Unlike Problem (P), Problem (Pa) only considers concentration data and not spectra. It minimizes deviations between the kinetic model prediction and a given set of concentration data estimates. While in Problem (P) $\boldsymbol{c}(t_i)$ are optimization variables and in Problem (Pa) these are inputs. In order to initialize the problem, the parameters $\boldsymbol{\theta}$ are guessed and there is assumed to be no noise, such that $\boldsymbol{c}(t_i)$ equals $\boldsymbol{z}(t_i)$ in this case. From (2) we can then compute an initial guess for the absorbance profiles $\boldsymbol{s}(\lambda_l)$. For the solution of Problem (Pa) we now assume $\boldsymbol{c}(t_i)$ to be known and $\boldsymbol{\theta}$ and $\boldsymbol{z}(t)$ to be unknown. The solution of Problem (Pa) then provides an initial estimate for $\boldsymbol{\theta}$ as well as $\boldsymbol{z}(t)$.

$$\begin{aligned} \min \quad & \sum_{i=1}^{ntp}\left(d_{i,l}-\sum_{k=1}^{nc} z_k(t_i)s_k(\lambda_l)\right)^2 \\ \text{s.t.} \quad & \boldsymbol{s}(\lambda_l)\geq 0 \quad \text{(optional)}, \qquad\qquad l=1,...,nwp. \end{aligned} \tag{Pb}$$

The heuristic then proceeds to solve Problem (Pb) to find estimates for $\boldsymbol{s}(\lambda_l)$. Problem (Pb) is a (bound-)constrained least squares optimization problem that takes as input the concentration estimates obtained in Problem (Pa). The variables of Problem (Pb) are $\boldsymbol{s}(\lambda_l)$ and the inputs are $\boldsymbol{z}(t)$ and $d_{i,l}$. The absorbance profiles $\boldsymbol{s}(\lambda_l)$ should be non-negative, unless the spectroscopic data $d_{i,l}$ is preprocessed with a Savitzky-Golay filter taking derivatives of the spectra. Note that Problem (Pb) assumes the error in the kinetic model $\omega_k(t_i)$ to be negligible, minimizing deviations from Beer Lambert's law only. Also, despite the fact that we refer to Problem (Pb) as a single optimization, it actually involves the solution of nwp problems, one for each measured wavelength λ_l. The output of Problem (Pb) provides estimates for $s(\lambda_l)$.

$$\begin{aligned} \min \quad & \sum_{i=1}^{ntp}\sum_{l=1}^{nwp}\left(d_{i,l}-\sum_{k=1}^{nc} c_k(t_i)s_k(\lambda_l)\right)^2 \\ \text{s.t.} \quad & \boldsymbol{c}(t_i)\geq 0 \qquad\qquad i=1,...,ntp. \end{aligned} \tag{Pc}$$

The next step in the optimization heuristic solves Problem (Pc), a bound-constrained least squares problem. Unlike Problem (Pb), Problem (Pc) accounts for model error and solves for $\boldsymbol{c}(t_i)$ using the absorbance estimates obtained from Problem (Pb). With the solution of this third subproblem the heuristic proceeds to solve Problem (Pa) again, but now with the new set of concentration data obtained from solving Problem (Pc). If convergence for $\boldsymbol{\theta}$ and $\boldsymbol{z}(t)$ is achieved the approach is terminated and the variances are estimated by solving the overdetermined system of equations presented in Problem (Pd).

$$\sum_{k=1}^{nc} s_k(\lambda_l)^2\sigma_k^2+\delta^2=\frac{1}{ntp}\sum_{i=1}^{ntp}\left(d_{i,l}-\sum_{k=1}^{nc} z_k(t_i)s_k(\lambda_l)\right)^2, \quad l=1,...,nwp. \tag{Pd}$$

Figure 1 summarizes the overall heuristic approach for estimating the variances. Since the solution of these optimization problems is a complicated process that requires data preprocessing, discretization of the temporal domain, and solution of large-scale nonlinear optimization problems, we have implemented a collection of tools within *KIPET* to facilitate the parameter estimation

procedure. In particular, one of the main challenges for solving Problems (P) and (Pa) comes from the fact that both problems are constrained with a system of differential algebraic equations. In order to solve this DAE system directly and simultaneously with standard optimization solvers the problem has to be reformulated into a general nonlinear programming (NLP) formulation. In the next section we describe the discretization approach followed in *KIPET* to transform Problems (P) and (Pa) into NLPs. We then present other features that make *KIPET* an excellent tool for solving the optimization problems described throughout this section.

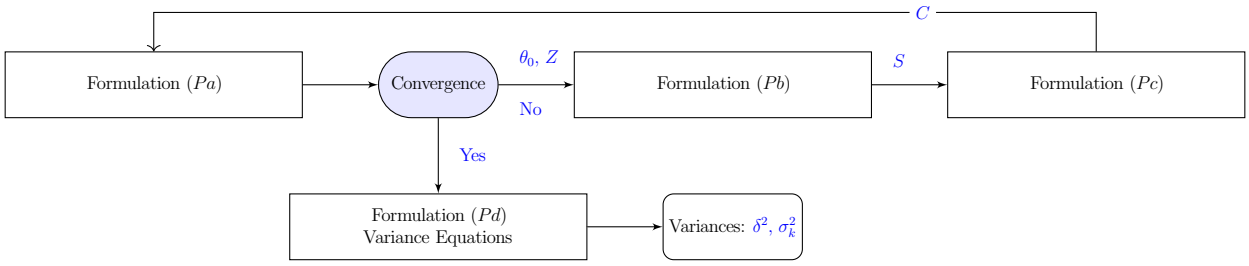

Figure 1: The variance estimation and initialization procedure as in Chen et al. (2016)

2.2. Discretization Strategies

KIPET relies on orthogonal collocation on finite elements to discretize the differential algebraic equations that appear in the constraints of the optimization problems. These techniques are attractive because of their stability properties (Biegler, 2010, Chapter 10). Collocation methods are a subset of implicit Runge-Kutta methods. In every optimization problem that involves DAEs differential and algebraic variables are discretized within *KIPET*. Here $K+1$ interpolation points (polynomial degree $\leq K$) are selected in element m. In the case of Lagrange-Radau collocation these correspond to the shifted roots of the corresponding Lagrange-Radau polynomials. Thus, for the kinetic model introduced in system (3) the differential state $\boldsymbol{z}(t)$ in element m is represented by

$$\begin{aligned} \boldsymbol{z}^K(t) &= \sum_{j=0}^{K} \ell_j(\tau)\boldsymbol{z}_{mj}, \quad t = t_{m-1} + h_m\tau, \quad \tau \in [0,1], \quad t \in [t_{m-1}, t_m] \\ \ell_j(\tau) &= \prod_{q=0, q\neq j}^{K} \frac{\tau - \tau_q}{\tau_j - \tau_q}, \end{aligned} \tag{5}$$

where $\boldsymbol{z}^K(t)$ is now a Lagrange interpolating polynomial of order K with the polynomial basis function $\ell_j(\tau)$ with $\tau_0 = 0$, $\tau_j < \tau_{j+1}$ for $j = 1, \ldots, K-1$ and the length of element m as h_m. Substituting this formulation into System (3) leads to the following collocation equations

$$\begin{aligned} &\sum_{j=0}^{K} \dot{\ell}_j(\tau)\boldsymbol{z}_{mj} - h_m \boldsymbol{f}(\boldsymbol{z}_{mj}, \boldsymbol{y}_{mj}, \boldsymbol{\theta}) = 0, \quad j \in \{1,\ldots,K\}, \quad m \in \{1,\ldots,N\}, \\ &\boldsymbol{g}(\boldsymbol{z}_{mj}, \boldsymbol{y}_{mj}, \boldsymbol{\theta}) = 0, \qquad\qquad\qquad\quad j \in \{1,\ldots,K\}, \quad m \in \{1,\ldots,N\}, \end{aligned} \tag{6}$$

where N denotes the number of finite elements and K the number of collocation points. Here $\dot{\ell}_j(\tau)$ stands for the time derivative of the polynomial basis function $\ell_j(\tau)$, i.e. $\dot{\ell}_j(\tau) = \frac{d\ell_j(\tau)}{d\tau}$.

With these collocation equations the kinetic model in Problem (P) is formulated as the following NLP:

$$
\begin{aligned}
\min \quad & \frac{1}{\delta^2}\sum_{i=1}^{ntp}\sum_{l=1}^{nwp}\left(d_{i,l}-\sum_{k=1}^{nc}c_{ki}s_{kl}\right)^2+\sum_{i=1}^{ntp}\sum_{k=1}^{nc}\frac{1}{\sigma_k^2}(c_{ki}-z_k^K(t_i))^2 \\
\text{s.t.} \quad & \sum_{j=0}^{K}\dot{\ell}_j(\tau)\boldsymbol{z}_{mj}-h_m\boldsymbol{f}(\boldsymbol{z}_{mj},\boldsymbol{y}_{mj},\boldsymbol{\theta})=0, \\
& \boldsymbol{z}^K(t_i)=\sum_{j=0}^{K}\ell_j(\tau)\boldsymbol{z}_{mj}, \quad t_i=t_{m-1}+h_m\tau, \quad \tau\in[0,1], \quad t_i\in[t_{m-1},t_m] \\
& \boldsymbol{g}(\boldsymbol{z}_{mj},\boldsymbol{y}_{mj},\boldsymbol{\theta})=0, \\
& \forall j\in\{1,...,K\},\, m\in\{1,...,N\} \\
& \boldsymbol{z}_{m+1,0}=\sum_{j=0}^{K}\ell_j(1)\boldsymbol{z}_{mj}, \quad m=1,...N-1 \\
& \boldsymbol{z}_f=\sum_{j=0}^{K}\ell_j(1)\boldsymbol{z}_{Nj}, \quad \boldsymbol{z}_{1,0}=\boldsymbol{z}(t_0) \\
& \boldsymbol{c}(t_i)\geq 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i=1,..,ntp \\
& \boldsymbol{s}(\lambda_l)\geq 0 \qquad \text{(optional)} \qquad\qquad\qquad\qquad\qquad\qquad l=1,..,nwp,
\end{aligned}
\tag{Pdisc}
$$

where the index f here refers to the final time, c_{ki} denotes the concentration profile with noise for each component k at each point in time t_i and s_{kl} the absorbance profile for each component k at each wavelength λ_l.
This problem is referred to as Problem (Pdisc) throughout the paper to emphasize that it is a discretized version of Problem (P). It should be noted that Problem (Pdisc) is a large-scale optimization problem with an inherent structure that requires efficient optimization solvers to efficiently find optimal solutions (Zavala et al., 2008; Rodriguez et al., 2018; Chiang et al., 2014; Biegler et al., 2002). In the next section the solution approach for the problem is described.

2.3. The KIPET *Software Package*

KIPET aims to provide a multi-tool all-in-one toolbox to provide the researcher or laboratory chemist with all the tools necessary to read, preprocess, analyze, and determine the details of their data. In order to explain how *KIPET* functions in a more detailed manner in relation to the way we envision the user's workflow when attempting to solve the difficult problem (P), this section describes some of the challenges associated with each step in the solution process. Figure 2 presents the modular structure of the package, with the class name presented in brackets below the functionality.

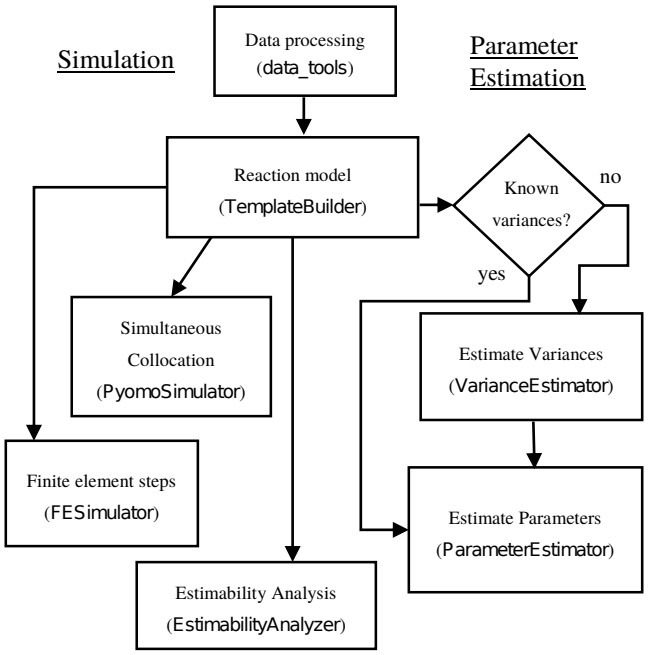

Figure 2: The software framework: *KIPET*'s modules (Figure from Short et al. (2019))

The central component in the solution procedure is the kinetic model. The model can be set up using the `TemplateBuilder` module. This `TemplateBuilder` leverages the modeling capabilities of the Python package *Pyomo* and creates an algebraic representation of the model. Once the model has been built, *KIPET* supports functionality for modeling and optimization – i.e. simulation and parameter estimation. Within the modeling tools, users can find the `PyomoSimulator` and the `FESimulator` modules, as well as the `data_tools` module. The simulators facilitate discretizing and representing the kinetic model in an algebraic form while the data module provides functions for preprocessing and analyzing experimental data. For formulating and solving the optimization problems *KIPET* has the `ParameterEstimator` and the `VarianceEstimator` modules. Due to general challenges faced when dealing with parameter estimation problems, it is often useful to run an estimability analysis to get an idea of which parameters are actually estimable given the kinetic model considered and the data provided. *KIPET* offers functionality for estimability analysis in the `EstimabilityAnalyzer` module but so far this is only implemented for problems with concentration data. Details for all different modules are summarized in the following sections.

2.3.1. Data Processing (`data_tools`*)*

The `data_tools` module consists of a number of tools for pre- and post-processing. The first set of functions are for reading in data from a variety of sources such as spreadsheets and text files, which are then converted into *pandas* dataframes for easy data manipulation within Python. Since most devices from various manufacturers output Comma Separated Value (CSV) spreadsheets, *KIPET* provides tools for detecting the type of data and performing unit conversions. Additionally, if data contains negative values, the software automatically detects these and can perform either a baseline shift (in order to keep the absorbance non-negativity constraints active) or can keep negative values and relax the bound. In a similar way to

reading data, results can be written to spreadsheets and text files using functions in this module.

Determining the Number of Absorbing Species
After data are parsed there are a number of preprocessing tools. The first tool is an SVD tool in order to identify the number of absorbing species. It is not unusual in experimental work that the chemist is unaware of the number of species within a mixture during a reaction as the reaction mechanisms may be unknown. Furthermore, it is difficult to determine the total number of species as linear dependencies between species may exist and certain species may not be detectable through spectra alone. While it may not be necessary to determine how many species absorb to obtain accurate parameters, the number of independently absorbing species can play a significant role in the obtained kinetic parameters if there is significant noise and only a few independent concentration profiles. After performing SVD, a graphic is shown of the ranked singular values that the user can use to select the likeliest number for the number of absorbing species to allow the user to perform a basic principal component analysis (PCA).

Preprocessing Options in KIPET
In many spectroscopic measurement instruments, instrumental noise, scattering effects, background noise, etc. can lead to data including unwanted physical information in addition to information on the chemical system (Gautam R. and Vanga S. and Ariese F., 2015; Rinnan et al., 2009). A number of pretreatment methods have been proposed to remove these, and there is no clear method that outperforms the others and in many instances a combination of techniques is used (Engel et al., 2013). *KIPET* therefore includes a number of these tools that can be used alone or in combination with other methods. Some common preprocessing strategies are included within *KIPET* and are summarized below.

Multiplicative Scatter Correction (MSC)
MSC attempts to correct the raw data for the effects of scaling and baseline offset (Martens and Naes, 1989). In MSC every individual spectrum is fitted to a chosen reference spectrum by:

$$\boldsymbol{d_i} = \boldsymbol{b_i d_{ref}} + \boldsymbol{a_i} + \boldsymbol{\epsilon_i}, \tag{7}$$

where $\boldsymbol{d_i}$ is a spectrum at a specific time i from $\boldsymbol{D}$ and $\boldsymbol{d_{ref}}$ is the reference spectrum. The reference spectrum in *KIPET* can be provided by the user (usually the average of the calibration set) or automatically determined as the average over the wavelengths. $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ are regression coefficients obtained by ordinary least squares of $\boldsymbol{d_i}$ versus $\boldsymbol{d_{ref}}$ over the wavelengths l. $\boldsymbol{\epsilon_i}$ is the un-modeled error in $\boldsymbol{d_i}$ The corrected spectrum, $\boldsymbol{d_{il,MSC}}$ is then calculated:

$$\boldsymbol{d_{il,MSC}} = (\boldsymbol{d_i} - \boldsymbol{a_i})/\boldsymbol{b_i} \tag{8}$$

This corrected spectrum can then be used in conjunction with other preprocessing methods or sent for parameter estimation.

Standard Normal Variate

An approach that is proposed to reduce spectral noise and remove scattering effects is Standard Normal Variate (SNV) (Barnes et al., 1989). In this approach, every element of the original data matrix is subject to:

$$d_{il,SNV} = \left(d_{il} - \overline{d}_i\right) / \sqrt{\frac{\sum_{i=1}^{ntp}\left(d_{il} - \overline{d}_i\right)^2}{p-1}}, \qquad l = 1,..,nwp \tag{9}$$

where $\overline{d}_i$ is the mean of the i-th spectrum and p is the number of values in the i-th spectrum. The square root term is thus just the standard deviation of the i-th spectrum. This approach is similar to MSC, however SNV tends to be more susceptible to noise in the dataset. Since SNV is not a linear transformation, it can also increase the nonlinear behaviour between the concentration matrix and the data matrix.

Savitzky-Golay Derivation

The Savitzky-Golay (SG) filter can be used to remove noise from spectra without significantly affecting the intensity through smoothing (Savitzky and Golay, 1964). In addition to the smoothing functionality, the filter can also be used to obtain derivatives of the spectra, which can be used to remove baseline shifts. The SG filter fits a polynomial across a certain moving window of data points. The user selects the order of the polynomial, order of the derivative and the width of the filter window and then each spectra is fitted to a series of polynomials within each wavelength window.

When derivatives are taken of the spectra, it will no longer be possible to include the non-negativity constraint on the individual absorbance profiles. In *KIPET*, when negative values are found in the data matrix after pre-processing, it is assumed that the user would like to relax the non-negativity constraints on $\boldsymbol{S}$, and it is automatically implemented. This does, however increase the rotational ambiguity and thus the chances of obtaining non-unique solutions.

Other Pretreatment Options

Certain easily implemented tools are also included that can be useful under certain conditions. They can be called from within *KIPET* to easily and quickly identify the best processing strategies for a given dataset. Some of the simple preprocessing strategies include baseline-shift tools and tools that remove certain wavelengths based on whether they are negative or zero as a result of other preprocessing or instrumental problem, or to just indiscriminately remove every x-th value in order to decrease the data size in problems where the size of the data matrix may cause memory issues (where x is any positive integer). A more nuanced approach to this, is the wavelength selection tool, which determines which wavelengths are the most informative for the parameter estimation based on its lack-of-fit. This tool is described in more detail in Section 2.3.7.

2.3.2. Reaction Model (`TemplateBuilder`*)*

The `TemplateBuilder` is the central module for building any model. *KIPET* uses *Pyomo*, an open-source Python-based optimization modeling language (Hart et al., 2017b), to formulate

all optimization problems. That means variables, parameters, constraints, objective functions, etc. are all defined via *Pyomo* in the backend. However *KIPET* has its own syntax loosely based off of the *Pyomo* syntax. For discretization *KIPET* uses the pyomo.dae toolbox (Nicholson et al., 2018). *Pyomo* provides access to many state-of-the-art solvers as well. The solver of choice for *KIPET* is *IPOPT* (Wächter and Biegler, 2006). Moreover, through the `TemplateBuilder` the user can add data to the model, provide initializations, and define feeding times and substrates for fed-batch processes. Via a short command, *KIPET* then builds the model automatically in the background. Once the model is constructed, it can be discretized by orthogonal collocation on finite elements by making use of the pyomo.dae toolbox.
The *KIPET* syntax is designed to be easy to use and aimed towards users of all programming levels. It is advised that the user choose one of the numerous examples included with the installation which is the most similar to the system that the user intends to solve and adjust the example to their own needs line by line.
If some of the species are non-absorbing, *KIPET* provides an option to define them in a different way, such that they will not be included with the absorbance profile estimation and only with the concentration profile estimation. Also, if absorbance profiles for a subset of the species is known in advance, it is possible to provide those and estimate the remaining profiles. It should be noted that in *KIPET* if measurement data are provided the finite elements automatically start at these measurement time points.

2.3.3. Simulation via Simultaneous Collocation (`PyomoSimulator`*)*

As *KIPET*'s main purpose is the solution of large-scale NLP optimization problems, good initializations for variables are crucial. Simulation can be one way of initializing the many unknown variables in the parameter estimation problem. The default method in *KIPET* is to use a simulation of the problem as an initialization. This is especially useful when using nonlinear programming solvers because they require good initial guesses for the optimization variables.
Within *KIPET*, simulation can be performed using orthogonal collocation on finite elements via pyomo.dae for discretization and *IPOPT* for the simultaneous solution of the NLP. More precisely, *IPOPT* is applied to the full NLP with the degrees of freedom fixed and a dummy objective chosen. This is the module of choice for general batch processes. However, if a fed-batch process is to be modeled, the alternative described in Section 2.3.4 may be necessary. The user has to specify the polynomial roots and the number of collocation points and finite elements. For DAE models the Lagrange-Radau collocation scheme is the scheme of choice. Furthermore it is advised to choose a small number of collocation points and to choose the number of finite elements dependent on the preferred discretization fineness level. Once these choices are made, the `PyomoSimulator` solves the fully discretized problem simultaneously using *IPOPT* with the parameters fixed to the values provided. Subsequently the `PyomoSimulator` automatically patches the local solution profiles for C and S into the full model.

2.3.4. Simulation via Finite Element Steps (`FESimulator`*)*

The `FESimulator` module provides an alternative to the `PyomoSimulator`. It is necessary for fed-batch processes with discrete feeds and it can be useful for other complex processes involving inputs, volume change etc., or for other large problems that are difficult to solve simultaneously

due to other reasons. This module is based on orthogonal collocation on finite elements but instead of solving for the full time horizon simultaneously, we move forward step by step, i.e. element by element, such that an undiscretized reference model is created in the background and its information is patched to the original fully discretized model. A march-forward simulation is thus run for the undiscretized reference model using pyomo.dae and *IPOPT*, in a similar fashion as described in Section 2.3.3, and the resulting data is patched to the original fully discretized model. For this approach the choice of Lagrange-Radau roots is important, as the last collocation point falls at the boundary to the next element, such that the solution for the initial point of the next element is available. The `FESimulator` is a subclass of the `PyomoSimulator` and calls `fe_factory`, the step by step finite element discretization tool. This all happens automatically in the background, with the user only having to define the inputs, such as dosing species, times and amounts for the fed substrates, before calling the `FESimulator`. In case of discrete feeding during the process, the finite element by finite element collocation procedure makes sure that the start of an additional element is located exactly at the time where the dosing occurs.

2.3.5. Estimate Variances (`VarianceEstimator`*)*

The variance estimation and initialization procedure illustrated in Figure 1 and described in detail in Section 2.1 can be performed using the `VarianceEstimator` module. In the case where the measurement device variance is known, this can be included and only the system variances can be estimated from the same methodology. The variance estimation is the most computationally expensive part of *KIPET* as a sequence of optimization subproblems has to be solved until convergence for the main subproblem is achieved. This tool not only provides variance estimates but also provides initial guesses for the parameters, absorbance profiles, and concentration profiles. These can be used as an alternative to simulation, the default initialization of the parameter estimation in *KIPET*.

2.3.6. Estimate Parameters (`ParameterEstimator`*)*

The `ParameterEstimator` module then estimates the kinetic parameters. It solves the full simultaneous optimization problem with many degrees of freedom. The known variances or the estimated variances are provided and plugged into the problem. Moreover, the parameters, and absorbance and concentration profiles are initialized. This problem has many variables, in particular all algebraic and differential variables, the concentrations with noise, the species' absorbance profiles, and the kinetic parameters. *KIPET* also provides the option of solving parameter estimation problems from concentration data. In case of dealing with concentration data instead of spectroscopic data, the problem reduces to a smaller problem. This means the first part of the objective in Problem (P) is neglected and this turns it into a least squares parameter estimation problem. Furthermore, we no longer have the absorbance profiles as degrees of freedom in that case.

In addition to providing functionality for estimating kinetic parameters and determining absorbance and concentration profiles, *KIPET* provides tools for assessing the quality of the parameter estimates, i.e. determine their confidence regions. *KIPET* applies the implicit function theorem to the optimality conditions and then computes the covariance matrix as an approximation of the inverse of the reduced Hessian. Inside *KIPET*, the computation of the reduced Hessian is either realized via *sIPOPT* (Pirnay et al. (2012)) or *k_aug* (Thierry and Biegler (2018)). Both of them can be used to determine confidence regions.

2.3.7. Wavelength Selection Based on Lack-Of-Fit

Some wavelengths contain irrelevant or redundant information which may lead to worse overall fits (Brenchley et al., 1997; Cécillon et al., 2008). In order to eliminate such wavelengths from data fitting, *KIPET* introduces wavelength selection for MCR. In order to assess the performance of any of the optimization problems presented, the lack-of-fit (LOF) can be used as a relative metric of the quality of the parameters and model to fit the data (De Juan et al., 2014):

$$LOF = \sqrt{\frac{\sum_{i=1}^{ntp}\sum_{l=1}^{nwp} e_{i,l}^2}{\sum_{i=1}^{ntp}\sum_{l=1}^{nwp} d_{i,l}^2}} \times 100\%, \tag{10}$$

where $e_{i,l}$ is the residual error corresponding to the corrected spectra element $d_{i,l}$. A common metric to determine the correlation between measured responses at certain wavelengths to the predictive properties in relation to a specific component's concentration can be determined by (Jouan-Rimbaud et al., 1995b,a; Skibsted et al., 2004):

$$\mathrm{cor}_{l,k} = \frac{\mathrm{cov}(d_l, c_k)}{s_{d_l} s_{c_k}}, \quad l = 1...,nwp; k = 1...,nc \tag{11}$$

where $cor_{l,k}$ is the correlation between specific measured wavelength λ_l and component k's concentration profile, cov is the covariance function and d_l is the l-th column of $\boldsymbol{D}$ and c_k is the k-th components concentration profile. s_{d_l} and s_{c_k} are the standard deviations for the d_l and c_k vectors respectively. cov is calculated based on the solution to the full parameter estimation problem solved for all wavelengths. This analysis is performed for each component at each wavelength and the maximum $cor_{l,k}$ at each wavelength is selected, i.e.

$$\mathrm{cor}_l = \max_{k=1,...,nc} \left| \frac{\mathrm{cov}(d_l, c_k)}{s_{d_l} s_{c_k}} \right|, \quad l = 1...,nwp. \tag{12}$$

This provides a rigorous analysis of which wavelengths are the most informative for the problem at hand. With this information, a certain cut-off threshold η with $0 < \eta < 1$ for the correlation can be set and the wavelengths with correlations below a certain threshold can be removed, i.e. the subset of selected wavelength variables WS that will be used in the parameter estimation can be selected in the following way:

$$WS = \{l| \quad cor_l \geq \eta, \quad l = 1,...,nwp\}. \tag{13}$$

Since there is no optimal threshold, *KIPET* provides a tool to analyze which threshold should be chosen in order to obtain the best LOF. This is done by running a series of parameter estimation problems with different values for η until a minimum LOF is obtained. This is the first application of such a method used to obtain better parameter estimates and lower residuals directly from spectra. An example of this tool with sample code and output is shown in Section 3.1.

2.3.8. Estimability Analysis for concentration data problems (`EstimabilityAnalyzer`*)*

In many parameter estimation problems, parameters may be inestimable based on the data collected. In order to determine which parameters are estimable an estimability analysis based

on the orthogonalization method from Yao et al. (2003) is implemented, followed by the mean squared error (MSE) approach for selecting the optimal number of parameters to estimate (Wu et al., 2011). This feature has been currently implemented for concentration data problems within *KIPET*. It first obtains ranks of the parameters based on their estimability. The method measures the magnitude of each parameter's influence on the model predictions and also for the correlated effects between the different parameters. In *KIPET* the sensitivity matrices in relation to each parameter is first obtained using *k_aug* at the nominal values (user-provided initial values) of the parameters. These sensitivities are scaled by the user confidence in each initial parameter value as well as the measurements. The orthogonalization scheme is then used to rank the parameters from most to least estimable. Following this, the MSE-based approach from Wu et al. (2011) is used to determine the optimal number of parameters to estimate. In this approach, the parameter estimation problem is solved multiple times with more estimable parameters as variables and less estimable parameters remaining constant. The MSE related to each model is then compared and the model with the best MSE and least number of parameters is selected. An example of the method's implementation within *KIPET* is presented in Section 3.2.
In future versions of *KIPET*, estimability analysis for problems involving spectra will also be included.

2.3.9. Post-processing

As *KIPET* is developed within Python, it is possible to make use of a wide array of standard Python packages for data visualization and post-processing. In order to visualize results, the software currently utilizes *matplotlib*. A number of tools to measure the lack-of-fit of the solutions, confidence intervals, and residuals are provided within *KIPET* to allow for solution verification. In addition to the methods described above, there are a number of useful methods for specific problems that are not demonstrated in the examples below but in the documentation and selection of examples included with the software package.

2.3.10. Distribution and Installation

KIPET uses common Python packages such as *scipy*, *numpy*, and *pandas* for data processing and manipulation. *Pyomo* is used as the algebraic modeling environment, *IPOPT* as the NLP solver, while *sIPOPT* and *k_aug* can be used to obtain NLP sensitivities (Hart et al., 2017a; Wächter and Biegler, 2006; Pirnay et al., 2012; Thierry and Biegler, 2018). The package *matplotlib* is used for data visualization. *KIPET* is made available under the GNU General Public License, GPL-3. Additional documentation is provided on the project GitHub page (Schenk et al., 2016-2019) that includes detailed installation instructions, as well as a detailed list of tutorials and functionality that are not discussed in this paper.

3. Results

In this section we will demonstrate the utility of the software package described above through the use of illustrative examples with sample code. All of the examples from the paper are included in the Github repository.

3.1. Archetypal Kinetic Parameter Estimation Problem

In this 1st example, simulated spectroscopic data is produced for the following batch reaction at isothermal conditions:

$$\mathrm{A} \xrightarrow{\mathrm{k_1}} \mathrm{B} \xrightarrow{\mathrm{k_2}} \mathrm{C},$$

where A has an initial concentration of 10^{-3}. The spectroscopic data matrix $\boldsymbol{D}$ has dimensions 300×100. This system is represented by the following system of differential equations:

$$\begin{aligned}
\frac{dc_A}{dt} &= -k_1 c_A \\
\frac{dc_B}{dt} &= k_1 c_A - k_2 c_B \\
\frac{dc_C}{dt} &= k_2 c_B
\end{aligned}$$

The example contains significant overlaps between the species' individual absorbance. To start with the solution of the problem, the data are first read in and then analyzed using PCA (assuming 5 principal components) to determine the independently absorbing species and then mildly smoothing the dataset using SG:

```
# Read data
D_frame = read_spectral_data_from_txt("mydata.txt")

# Perform data analysis
basic_pca(D_frame, n=5, with_plots=with_plots)

# Run pre-processing filter
D_frame = savitzky_golay(dataFrame=D_frame,
                         window_size=13,
                         orderPoly=3,
                         orderDeriv=0)
```

Listing 1: Basic preprocessing code sample

The resulting singular values, shown in Figure 3, indicate that we are correct in assuming that we have only three absorbing species. After this stage we can set up the model. Note that each component has a corresponding differential equation.

```
# Set up the model
builder = TemplateBuilder()

# Define components
components = {'A':1e-3,'B':0,'C':0}
builder.add_mixture_component(components)
builder.add_parameter('k1', init=4.0, bounds=(0.0,5.0))
builder.add_parameter('k2',bounds=(0.0,1.0))
```

```
# Define explicit system of ODEs
def rule_odes(m, t):
    exprs = dict()
    exprs['A'] = -m.P['k1']*m.Z[t, 'A']
    exprs['B'] = m.P['k1']*m.Z[t, 'A']-m.P['k2']*m.Z[t, 'B']
    exprs['C'] = m.P['k2']*m.Z[t, 'B']
    return exprs
builder.set_odes_rule(rule_odes)

# Specify data
builder.add_spectral_data(D_frame)

# Create model and define time horizon
opt_model = builder.create_pyomo_model(0.0,10.0)
```

Listing 2: Code sample for example 1 `TemplateBuilder`

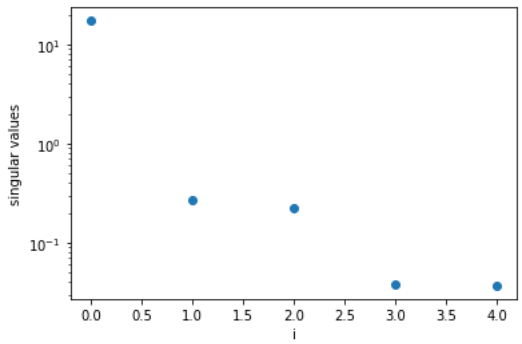

Figure 3: Resulting PCA results showing ranked singular values for 5 components

After the model is defined the variances can be estimated:

```
# Create estimator object
v_estimator = VarianceEstimator(opt_model)

# Discretize time domain
v_estimator.apply_discretization('dae.collocation',
                                 nfe=nfe,
                                 ncp=ncp,
                                 scheme='LAGRANGE-RADAU')

# Solve estimation problem
results_variances = v_estimator.run_opt('ipopt',
                                        tee
                                           =
```

```
                                                    True
                                                    ,


                                                    solver_options=options,
                                                tolerance=1e-5,
                                                max_iter=15)

sigmas = results_variances.sigma_sq
```

Listing 3: Code sample demonstrating `VarianceEstimator`

The variance estimation algorithm converges in a single iteration with results of 'A': 1.0788e-10, 'B': 1.6454e-10, 'C': 9.7184e-11, 'device': 3.2870e-06. The low model variances for the components are to be expected due to the simulated data. Once the variances are obtained the parameter estimation can be performed. Notice that *KIPET* allows for initialization and scaling based on the solutions to the variance estimation:

```
# Create estimator object
p_estimator = ParameterEstimator(opt_model)

# Initialization
p_estimator.initialize_from_trajectory('Z',results_variances.Z)
p_estimator.initialize_from_trajectory('S',results_variances.S)
p_estimator.initialize_from_trajectory('C',results_variances.C)

# Scale trajectories (optional)
p_estimator.scale_variables_from_trajectory('Z',
                                            results_variances.Z)
p_estimator.scale_variables_from_trajectory('S',
                                            results_variances.S)
p_estimator.scale_variables_from_trajectory('C',
                                            results_variances.C)

options = dict()
options['nlp_scaling_method'] = 'user-scaling'

# Run optimization
results_pyomo = p_estimator.run_opt('k_aug',
                                    solver_opts = options,
                                    covariance = True,
                                    variances=sigmas)
```

Listing 4: Code sample demonstrating the use of `ParameterEstimator`

The solution, obtained in 89.63 CPUs, is given as $k_1 = 0.2604 \pm 0.0081$ and $k_2 = 1.3132 \pm 0.0490$. The resulting pure species absorbances and concentration profiles are shown in Figure 4. A lack-of-fit of 0.6183 % is found, meaning that the model provides a very good fit to the data. To demonstrate the wavelength selection tool within *KIPET*, we attempt to find a better fit by

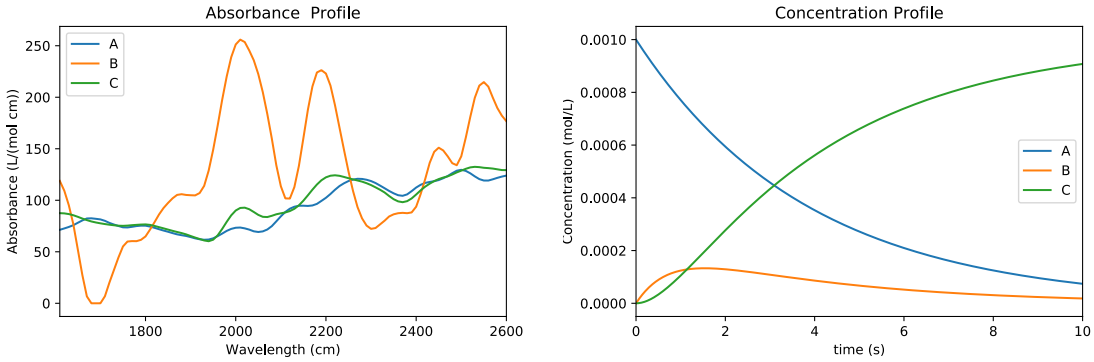

Figure 4: Solution to full parameter estimation problem

removing wavelengths that are less informative using the method described in Section 2.3.7. The following code can be used to first find the $\text{cor}_{l,k}$ plot and then to find wavelengths to remove before running the parameter estimation for the optimal reduced wavelength set to minimize the lack-of-fit.

```
# Create estimator object
p_estimator = ParameterEstimator(opt_model)

# Generate correlations for each wavelength
correlations = p_estimator.wavelength_correlation()

# Run lack-of-fit analysis
p_estimator.run_lof_analysis(builder_before_data,
                             end_time,
                             correlations,
                             lof,
                             nfe,
                             ncp,
                             sigmas)

# Choose subset of components
new_subs = wavelength_subset_selection(correlations=correlations,
                                       n=0.7)
```

Listing 5: Sample code demonstrating wavelength selection

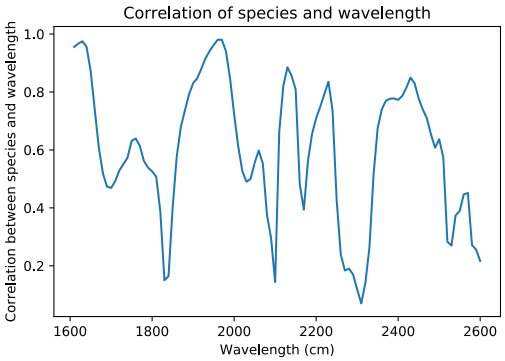

Figure 5: Maximum correlation of each wavelength to concentration profiles

The results from this better-fitting model are shown in Figure 5. Despite fewer wavelengths, the lack-of-fit obtained is reduced insignificantly to 0.5999% and the parameters obtained are $k_1 = 0.2279 \pm 0.0104$ and $k_2 = 1.6368 \pm 0.0962$. From 6 we can see that only the few wavelengths that are most informative are chosen. Since the technique can often ignore large sections of data, it should be used with caution.

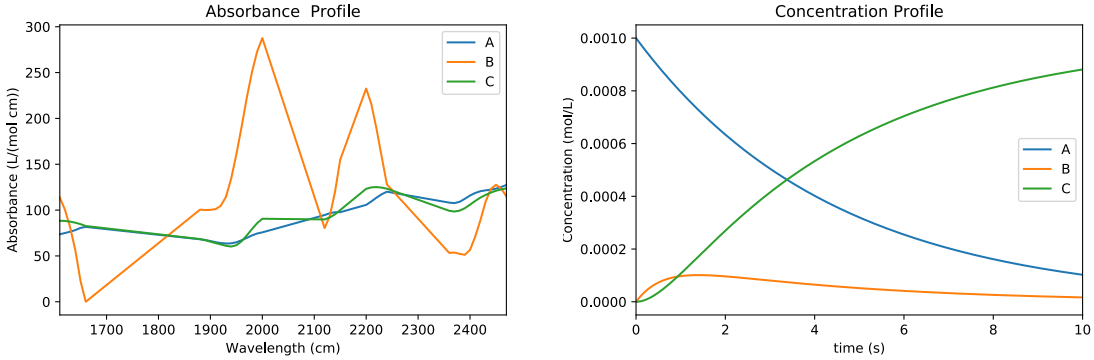

Figure 6: Lack-of-fit minimization solution

3.2. Parameter Estimation from Concentration Data

A useful feature of *KIPET* is its ability to also account for problems where concentrations, or other model responses, are measured directly. In this simulated problem we have the following system of differential equations:

$$\frac{dc_A}{dt} = -k_1 c_A - k_4 c_A - k_5 c_A c_E$$
$$\frac{dc_B}{dt} = k_1 c_A - k_2 c_B - k_3 c_B$$
$$\frac{dc_C}{dt} = k_2 c_B - k_4 c_C$$
$$\frac{dc_D}{dt} = k_4 c_A - k_3 c_D$$
$$\frac{dc_E}{dt} = k_3 c_B - k_5 c_A c_E$$
$$\frac{dc_F}{dt} = k_5 c_A c_E - k_6 c_F c_G^2$$
$$\frac{dc_G}{dt} = -k_6 c_F c_G^2$$
$$\frac{dc_H}{dt} = k_6 c_F c_G^2$$

Significant noise is added to the simulated data and this is used as the data matrix, which takes the form of $\boldsymbol{C}$ in this case, as described in Section 2.3.6. The initial conditions are the following: $c_A(0) = 0.5$, $c_B(0) = 0.0$, $c_C(0) = 0.0$, $c_D(0) = 0.01$, $c_E(0) = 0.0$, $c_F(0) = 0.3$, $c_G(0) = 0.5$ and $c_H(0) = 0.0$.
The data are generated with the following reaction parameters:
$k_1 = 0.3$, $k_2 = 0.1$, $k_3 = 0.1$, $k_4 = 0.4$, $k_5 = 0.02$ and $k_6 = 0.5$.
The noised output is shown on the left of Figure 7. After setting up the model in a similar way to the sample code shown previously, it is possible to add concentration data rather than spectroscopic data with the following function.

```
D_frame = read_concentration_data_from_csv('sim_data.csv')
```

Listing 6: Sample code demonstrating `data_tools` for reading concentration data

This then automatically alters the objective to just use the second term in Problem (P) in Section 2.1 as explained in Section 2.3.6. To demonstrate the estimability analysis implementation described in Section 2.3.8, an estimability analysis will be performed.

```
# Set up the estimability analysis
e_analyzer = EstimabilityAnalyzer(opt_model)
# Problem needs to be discretized first
e_analyzer.apply_discretization
    ('dae.collocation', nfe=50, ncp=3, scheme='LAGRANGE-RADAU')
# define the uncertainty surrounding each of the parameters
param_uncertainties
     = {'k1':0.8, 'k2':1.2, 'k3':0.8, 'k4':0.4, 'k5':1, 'k6':0.3}
# measurement scaling
meas_uncertainty = 0.02
```

```
# using the method of Yao (2003) to rank parameters.
listparams = e_analyzer.rank_params_yao(meas_scaling
    =      meas_uncertainty, param_scaling = param_uncertainties)
# Now we can run the analyzer using the list of ranked parameters
params_to_select
    = e_analyzer.run_analyzer(parameter_rankings = listparams
   , meas_scaling = meas_uncertainty, variances = sigmas)
```

Listing 7: Code sample demonstrating `EstimabilityAnalyzer`

This code sample returns a list of parameters that can be estimated based on the user-provided confidence in the initial parameter values and measurement error and the model mean squared error. In this example, k_5 is the only parameter that is not estimable, due to its high initial uncertainty as well as its relatively low impact on the overall model. We thus fix this parameter and run the parameter estimation problem with all other parameters as variables. The final results from this parameter estimation problem are shown in Figure 7 and the kinetic parameters' results and resulting variances are shown in Table 1. This example, including the data simulation, is available on the Github repository.

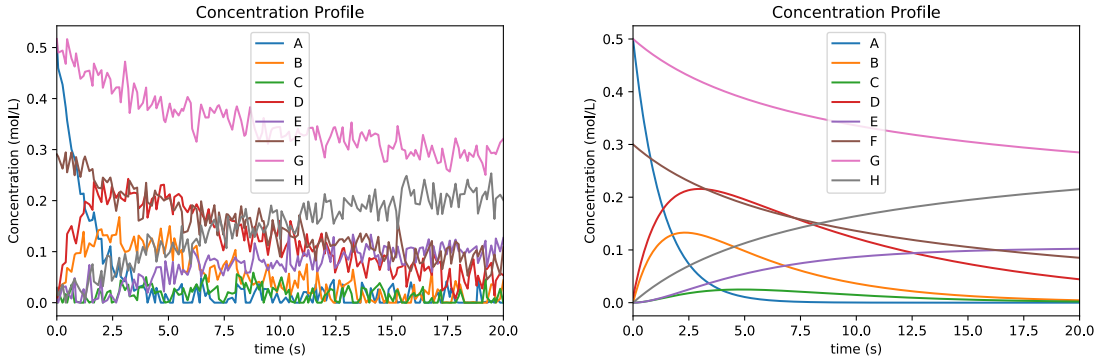

Figure 7: The simulated data on the left and the final fitted concentration profiles for Example 2

Kinetic Parameter	Estimated Value	Variance
k_1	0.3007	4.76e-12
k_2	0.1026	4.70e-13
k_3	0.1005	1.32e-13
k_4	0.3943	2.97e-12
k_5	0.032	N/A
k_6	0.5038	5.99e-13

Table 1: Example 2 results

3.3. Parameter Estimation from Spectra with Inputs, Non-Absorbing Species and Additional States

KIPET contains many other additional useful features. Here dosing during the process and volume changes are considered for a system, where one of the species is assumed to be

non-absorbing. The following chemical reactions are considered:

$$\begin{aligned}
&\mathrm{A} + \mathrm{B} \xrightarrow{\mathrm{k_0}} \mathrm{E} + \mathrm{D}\\
&\mathrm{E} + \mathrm{C} \xrightarrow{\mathrm{k_1}} \mathrm{F}\\
&\mathrm{F} \xrightarrow{\mathrm{k_2}} \mathrm{E} + \mathrm{C}\\
&\mathrm{F} + \mathrm{A} \xrightarrow{\mathrm{k_3}} \mathrm{G+E}\\
&\mathrm{F} + \mathrm{D} \xrightarrow{\mathrm{k_4}} \mathrm{G+B}.
\end{aligned}$$

The process is described by the following system of differential equations:

$$\begin{aligned}
\frac{dV}{dt} &= d_V(t) + \begin{cases} \text{const flowrate}, & t \leq 210\,\text{min} \\ 0, & t > 210\,\text{min} \end{cases}\\
\frac{dc_A}{dt} &= -r_0 - r_3 - \frac{\dot{V}}{V}c_A + d_A(t)\\
\frac{dc_B}{dt} &= -r_0 + r_4 - \frac{\dot{V}}{V}c_B\\
\frac{dc_E}{dt} &= r_0 - r_1 + r_2 + r_3 - \frac{\dot{V}}{V}c_E\\
\frac{dc_D}{dt} &= r_0 - r_4 - \frac{\dot{V}}{V}c_D\\
\frac{dc_C}{dt} &= -r_1 + r_2 - \frac{\dot{V}}{V}c_C + \begin{cases} m_{C_{add}}/V/210\,\text{min}, & t \leq 210\,\text{min} \\ 0, & t > 210\,\text{min} \end{cases}\\
\frac{dc_F}{dt} &= -r_1 + r_2 - r_3 - r_4 - \frac{\dot{V}}{V}c_F\\
\frac{dc_G}{dt} &= r_3 + r_4 - \frac{\dot{V}}{V}c_G
\end{aligned}$$

with the rate laws

$$\begin{aligned}
r_0 &= k_0 c_A c_B\\
r_1 &= k_1 c_E c_C\\
r_2 &= k_2 c_F\\
r_3 &= k_3 c_F c_A\\
r_4 &= k_4 c_F c_D
\end{aligned}$$

and the discrete volume change and the dosing of substrate A given by

$$d_V(t) = \begin{cases} 0.01, & \text{if } t = 303.126\,\text{min}, \\ 0, & \text{else} \end{cases}$$

and

$$d_A(t) = \begin{cases} 0.03, & \text{if } t = 101.035\,\text{min}, \\ 0, & \text{else}. \end{cases}$$

In this case, F is assumed to be non-absorbing. Simulated data is created and estimates for the parameters are obtained. The simulated data are generated with these parameter values:
$k_0 = 0.2545$, $k_1 = 8.93156$, $k_2 = 1.31765$, $k_3 = 0.310870$, $k_4 = 3.87809$
and these initial values:
$c_A(0) = 0.395555$, $c_B(0) = 0.0351202$, $c_C(0) = 0.0$, $c_D(0) = 0.0$, $c_E(0) = 0.0$, $c_F(0) = 0.0$, $c_G(0) = 0.0$ and $V(0) = 0.0629418$.
We specify the non-absorbing species in the following way:

```
non_abs = ['F']
builder.set_non_absorbing_species(model, non_abs)
```

Listing 8: Definition of non-absorbing species

The additional state for the volume is added as follows:

```
extra_states = dict()
extra_states['V'] = 0.0629418
builder.add_complementary_state_variable(extra_states)
```

Listing 9: Definition of additional non-response differential states

Additional arguments have to be defined for the inputs and the `FESimulator` has to be called instead of the `PyomoSimulator`, but first the additional feed times are defined and added to the model before the data are added.

```
#Add time points where feed as discrete jump should take place:
feed_times=[101.035, 303.126]
builder.add_feed_times(feed_times)
model = builder.create_pyomo_model(0, 600)
```

Listing 10: Definition of feed times and inclusion in the model

We can now call the `FESimulator` to run a simulation for the initialization of our parameter estimation problem:

```
# call FESimulator
sim = FESimulator(model)

# define discrete points wanted in the concentration profile
sim.apply_discretization
    ('dae.collocation', nfe=50, ncp=3, scheme='LAGRANGE-RADAU')

# Define inputs:
inputs_sub = {}
inputs_sub['Y'] = ['5']
fixedy = True
yfix={}
yfix['Y']=['5']
```

```
# since these are inputs we need to fix this
for key in sim.model.time.value:
    sim.model.Y[key, '5'].set_value(key)
    sim.model.Y[key, '5'].fix()

# New Inputs for discrete feeds. Component and amount fed.
Z_step = {'A': .03}
X_step = {'V': .01}
jump_states = {'Z': Z_step, 'X': X_step}

# Which component is added at which point in time
jump_points1 = {'A': 101.035}
jump_points2 = {'V': 303.126}
jump_times = {'Z': jump_points1, 'X': jump_points2}
init = sim.call_fe_factory
   (inputs_sub, jump_states, jump_times, feed_times)
options = {}
results_sim = sim.run_sim('ipopt', tee=True)
```

Listing 11: Simulation via `FESimulator` with discrete feeds and changing volume

Now that simulation results are obtained these can be used to initialize the parameter estimation problem. As the parameters are already defined for the simulation, this dictionary can be removed and the parameters and their corresponding bounds can be defined, along with initial values that are only 10% away from the nominal values.

```
model.del_component(params)
builder.add_parameter('k0',init=0.9*0.2545,bounds=(0.0,100.))
builder.add_parameter('k1',init=0.9*8.93156,bounds=(0.0,100.))
builder.add_parameter('k2',init=0.9*1.31765,bounds=(0.0,100.))
builder.add_parameter('k3',init=0.9*0.310870,bounds=(0.,100.))
builder.add_parameter('k4',init=0.9*3.87809,bounds=(0.0,100.))
model = builder.create_pyomo_model(0.,600.)
```

Listing 12: Definition of parameters, their initial values and bounds

`ParameterEstimator` can then be called and initialized from the simulation trajectories as in the first example above. When running parameter estimation with a finite element by finite element discretization with inputs, some additional arguments that we already defined above need to be included.

```
results_pyomo = p_estimator.run_opt('ipopt_sens',
                                    tee=True,
                                    solver_opts=options,
                                    variances=sigmas,
```

```
                                    with_d_vars=True,
                                    covariance=True,
                                    inputs_sub=inputs_sub,
                                    jump=True,
                                    jump_times=jump_times,
                                    jump_states=jump_states,
                                    feed_times=feed_times,
                                    fixedy=True,
                                    report_time = True,
                                    yfix=yfix)
```

Listing 13: Solution of the parameter estimation problem with sensitivities

To determine the quality of the kinetic parameter estimates, *sIPOPT* must be used instead of *IPOPT* to compute sensitivities and calculate the covariance matrix. The solver *sIPOPT* can be called via choosing ipopt_sens as the solver option. The solution takes 125 CPUs and the lack-of-fit is 0.0024%. The resulting parameters and their confidence regions are displayed in Table 2. The resulting absorbance and concentration profiles are illustrated in Figure 8. This

Kinetic Parameter	Estimated Value	Confidence Interval
k_0	0.2544	(0.2539, 0.2549)
k_1	8.9307	(8.6379, 9.2234)
k_2	1.3022	(1.2439, 1.3605)
k_3	0.3095	(0.3066, 0.3125)
k_4	3.8387	(3.7448, 3.9327)

Table 2: Parameter values and confidence intervals

example, including the data simulation, is available on the Github repository.

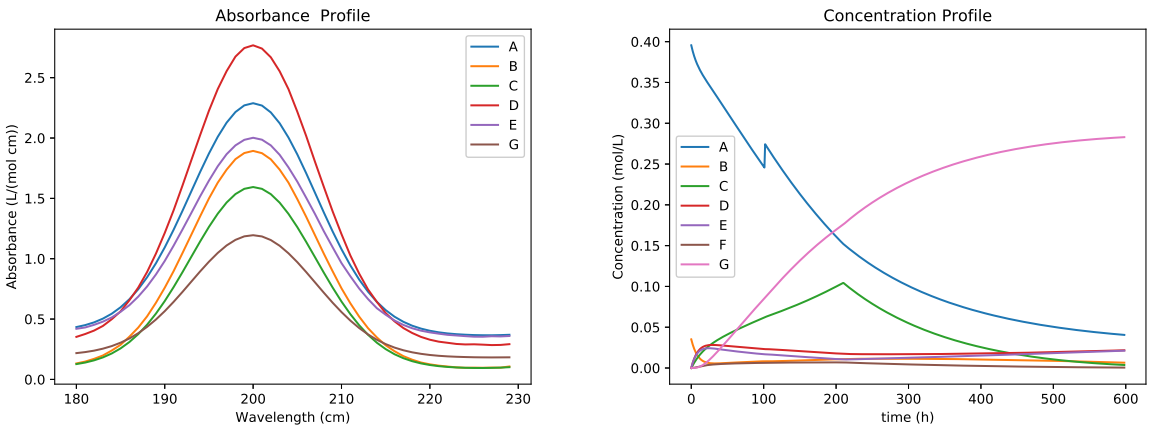

Figure 8: left: Estimated absorbance profile, right: Estimated concentration profile

4. Conclusion

The analysis of data resulting from spectroscopic measurements on batch reactors is an extremely challenging problem, especially when it is desired to directly obtain kinetic parameters and their confidence regions from large noisy datasets. It is crucial to maximize the information obtained from each experiment as each experiment can be costly to run. Since few software packages exist for the determination of kinetic parameters directly from spectra, and with the development of an easily extendable framework from Chen et al. (2016), we propose a novel software package that uses ideas based on maximum likelihood principles, collocation methods, and large-scale nonlinear programming to estimate kinetic parameters directly from spectra.
The package is implemented in the open-source *KIPET* package and freely available for download on Github. The package makes use of *Pyomo* as the algebraic modeling backbone, however *KIPET* uses a simplified syntax to declare the model. *IPOPT* is used to solve the resulting nonlinear programming problem and *sIPOPT* or *k_aug* can be used for finding parameter confidence intervals. For solving such large-scale NLP optimization problems, rigorous and effective initialization schemes are also included. *KIPET* allows for additional functionality in an easy-to-use, well documented format that does not require detailed programming knowledge. Users are able to make use of a set of detailed and adaptable templates for different problem types in order to read and preprocess data, simulate and assess their models, estimate both measurement variances and component model variances, estimate kinetic parameters from various data sources including confidence intervals, and obtain estimability assessments and data visualizations. In addition to the adaptable framework and new open-source tools, *KIPET* also introduces the concept of wavelength selection based on *LOF* for the first time.
KIPET is being actively developed and future releases will look to satisfy requests from the user-base, as well as including additions to the overall framework, such as ways to handle multiple experimental datasets, data coming from different instruments, unwanted spectral contributions, and unknown absorbing species. Additionally, methods for optimal design of experiments and more detailed model discrimination and estimability tools will be included in future releases.

5. Acknowledgements

The authors would like to gratefully acknowledge the funding of Eli Lilly and Company, along with Pfizer Inc. for postdoctoral fellowship (to C.S.).

References

J. A. E. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl. Casadi - A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, in press, 2018.

R. Barnes, M. Dhanoa, and S. Lister. Standard Normal Variate Transformation and De-Trending of Near-Infrared Diffuse Reflectance Spectra. *Applied Spectroscopy*, 43(5):772–777, 1989. doi: https://doi.org/10.1366/0003702894202201.

L. Biegler. *Nonlinear Programming*. Society for Industrial and Applied Mathematics, 2010. doi: 10.1137/1.9780898719383. URL `http://epubs.siam.org/doi/abs/10.1137/1.9780898719383`.

L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57:575–593, 2002.

S. Bijlsma, H. F. M. Boelens, H. C. J. Hoefsloot, and A. K. Smilde. Estimating reaction rate constants: Comparison between traditional curve fitting and curve resolution. *Analytica Chimica Acta*, 419(2):197–207, 2000. ISSN 00032670. doi: 10.1016/S0003-2670(00)00994-6.

S. Bijlsma, H. F. M. Boelens, and A. K. Smilde. Determination of rate constants in second-order kinetics using UV-visible spectroscopy. *Applied Spectroscopy*, 55(1):77–83, 2001. ISSN 00037028. doi: 10.1366/0003702011951281.

J. M. Brenchley, U. Hörchner, and J. H. Kalivas. Wavelength selection characterization for NIR spectra. *Applied Spectroscopy*, 1997. ISSN 00037028. doi: 10.1366/0003702971940837.

L. Cécillon, N. Cassagne, S. Czarnes, R. Gros, and J. J. Brun. Variable selection in near infrared spectra for the biological characterization of soil and earthworm casts. *Soil Biology and Biochemistry*, 2008. ISSN 00380717. doi: 10.1016/j.soilbio.2008.03.016.

W. Chen, L. Biegler, and S. García-Muñoz. An approach for simultaneous estimation of reaction kinetics and curve resolution from process and spectral data. *Journal of Chemometrics*, 30:506–522, 2016. doi: 10.1002/cem.2808. URL `http://dx.doi.org/10.1002/cem.2808`.

W. Chen, L. T. Biegler, and S. García-Muñoz. Kinetic parameter estimation based on spectroscopic data with unknown absorbing species. *AIChE Journal*, 64(10):3595–3613, 2018. ISSN 15475905. doi: 10.1002/aic.16334.

W. Chen, L. T. Biegler, and S. García-Muñoz. A Unified Framework for Kinetic Parameter Estimation Based on Spectroscopic Data with or without Unwanted Contributions. *Industrial and Engineering Chemistry Research*, pages 1–27, 2019.

N. Chiang, C. G. Petra, and V. M. Zavala. Structured nonconvex optimization of large-scale energy systems using PIPS-NLP. In *Proc. of the 18th Power Systems Computation Conference (PSCC), Wroclaw, Poland*, 2014.

A. De Juan, M. Maeder, M. Martínez, and R. Tauler. Combining hard- and soft-modelling to solve kinetic problems. *Chemometrics and Intelligent Laboratory Systems*, 54(2):123–141, 2000. ISSN 01697439. doi: 10.1016/S0169-7439(00)00112-X.

A. De Juan, J. Jaumot, and R. Tauler. Multivariate Curve Resolution (MCR). Solving the mixture analysis problem. *Analytical Methods*, 2014. ISSN 17599679. doi: 10.1039/c4ay00571f.

F. de la Peña, V. Fauske, P. Burdet, E. Prestat, P. Jokubauskas, M. Nord, T. Ostasevicius, K. MacArthur, M. Sarahan, D. Johnstone, J. Taillon, A. Eljarrat, V. Migunov, J. Caron, T. Furniva, and A. Skorikov. HyperSpy, 2018. URL `http://hyperspy.org/`.

J. Engel, J. Gerretzen, E. Szymańska, J. J. Jansen, G. Downey, L. Blanchet, and L. M. Buydens. Breaking with trends in pre-processing? *TrAC - Trends in Analytical Chemistry*, 50:96–106, 2013. ISSN 18793142. doi: 10.1016/j.trac.2013.04.015.

U. Gautam R. and Vanga S. and Ariese F. Review of multidimensional data processing approaches for Raman and infrared spectroscopy. 2(1):1-38. *Epj Techniques & Instrumentation*, 2(1):1–38, 2015.

A. Golshan, H. Abdollahi, S. Beyramysoltan, M. Maeder, K. Neymeyr, R. Rajkó, M. Sawall, and R. Tauler. A review of recent methods for the determination of ranges of feasible solutions resulting from soft modelling analyses of multivariate data. *Analytica Chimica Acta*, 911:1–13, 2016. ISSN 18734324. doi: 10.1016/j.aca.2016.01.011.

W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo Optimization Modeling in Python*. Springer, 2017a. ISBN 978-3-319-58819-3. doi: 10.1007/978-3-319-58821-6.

W. E. Hart et al. *Pyomo–Optimization Modeling in Python*, volume 67. Springer Science & Business Media, 2nd edition, 2017b.

J. E. Jackson. *A User's Guide to Principal Components*. Wiley, 1991. ISBN 9780471622673. doi: 10.1002/0471725331.

J. Jaumot, A. de Juan, and R. Tauler. MCR-ALS GUI 2.0: New features and applications. *Chemometrics and Intelligent Laboratory Systems*, 140:1–12, 2015. ISSN 18733239. doi: 10.1016/j.chemolab.2014.10.003. URL `http://dx.doi.org/10.1016/j.chemolab.2014.10.003`.

D. Jouan-Rimbaud, M. S. Khots, D. L. Massart, I. R. Last, and K. A. Prebble. Calibration line adjustment to facilitate the use of synthetic calibration samples in near-infrared spectrometric analysis of pharmaceutical production samples. *Analytica Chimica Acta*, 1995a. ISSN 00032670. doi: 10.1016/0003-2670(95)00348-4.

D. Jouan-Rimbaud, B. Walczak, D. L. Massart, I. R. Last, and K. A. Prebble. Comparison of multivariate methods based on latent vectors and methods based on wavelength selection for the analysis of near-infrared spectroscopic data. *Analytica Chimica Acta*, 1995b. ISSN 00032670. doi: 10.1016/0003-2670(94)00590-I.

W. H. Lawton and E. A. Sylvestre. Self modeling curve resolution. *Technometrics*, 13(3):617–633, 1971a. ISSN 15372723. doi: 10.1080/00401706.1971.10488823.

W. H. Lawton and E. A. Sylvestre. Elimination of linear parameters in nonlinear regression. *Technometrics*, 1971b. ISSN 15372723. doi: 10.1080/00401706.1971.10488810.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944. ISSN 0033-569X. doi: 10.1090/qam/10666.

M. Maeder and A. D. Zuberbühler. Nonlinear Least-Squares Fitting of Multivariate Absorption Data. *Analytical Chemistry*, 1990. ISSN 15206882. doi: 10.1021/ac00219a013.

D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. ISSN 0368-4245. doi: 10.1137/0111030.

H. Martens and T. Naes. *Multivariate Calibration*. Wiley, Chichester, 1989.

D. L. Massart, B. G. M. Vandeginste, L. M. C. Buydens, S. D. Jong, P. J. Lewi, and J. Smeyers-Verbeke. *Handbook of Chemometrics and Qualimetrics: Part A*. Elsevier, Amsterdam, 1997. ISBN 0-444-89724-0. doi: 10.1021/ci980427d.

K. Neymeyr, M. Sawall, and D. Hess. Pure component spectral recovery and constrained matrix factorizations: Concepts and applications. *Journal of Chemometrics*, 24(2):67–74, 2010. ISSN 08869383. doi: 10.1002/cem.1273.

B. Nicholson, J. D. Siirola, J.-P. Watson, V. M. Zavala, and L. T. Biegler. pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. *Mathematical Programming Computation*, 10(2):187–223, 2018.

H. Pirnay, R. López-Negrete, and L. T. Biegler. Optimal Sensitivity based on IPOPT. *Mathematical Programming Computation*, 4(4):307–331, Dec 2012. ISSN 1867-2957. doi: 10.1007/s12532-012-0043-2. URL `http://dx.doi.org/10.1007/s12532-012-0043-2`.

Process Systems Enterprise. gPROMS, 1997-2018. URL `www.psenterprise.com/gproms`.

G. Puxty, M. Maeder, and K. Hungerbühler. Tutorial on the fitting of kinetics models to multivariate spectroscopic measurements with non-linear least-squares regression. *Chemometrics and Intelligent Laboratory Systems*, 81(2):149–164, 2006. ISSN 01697439. doi: 10.1016/j.chemolab.2005.12.001.

Å. Rinnan, F. van den Berg, and S. B. Engelsen. Review of the most common pre-processing techniques for near-infrared spectra. *TrAC - Trends in Analytical Chemistry*, 28(10):1201–1222, 2009. ISSN 01659936. doi: 10.1016/j.trac.2009.07.007. URL `http://dx.doi.org/10.1016/j.trac.2009.07.007`.

J. S. Rodriguez, B. Nicholson, C. Laird, and V. M. Zavala. Benchmarking ADMM in nonconvex NLPs. *Computers and Chemical Engineering*, 119:315–325, 2018.

C. Ruckebusch and L. Blanchet. Multivariate curve resolution: A review of advanced and tailored applications and challenges. *Analytica Chimica Acta*, 765:28–36, 2013. ISSN 00032670. doi: 10.1016/j.aca.2012.12.028. URL `http://dx.doi.org/10.1016/j.aca.2012.12.028`.

A. Savitzky and M. J. Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964. ISSN 15206882. doi: 10.1021/ac60214a047.

M. Sawall, A. Börner, C. Kubis, D. Selent, R. Ludwig, and K. Neymeyr. Model-free multivariate curve resolution combined with model-based kinetics: Algorithm and applications. *Journal of Chemometrics*, 26(10):538–548, 2012. ISSN 08869383. doi: 10.1002/cem.2463.

C. Schenk, M. Short, J. Rodriguez, D. Thierry, S. García-Muñoz, and L. Biegler. KIPET – Kinetic Parameter Estimation Toolkit. `https://github.com/salvadorgarciamunoz/kipet`, 2016-2019.

M. Short, C. Schenk, D. M. Thierry, J. S. Rodriguez, L. T. Biegler, and S. García-Muñoz. Kipet - an open-source kinetic parameter estimation toolkit. *FOCAPD Conference 2019 (accepted)*, 2019.

E. T. Skibsted, H. F. Boelens, J. A. Westerhuis, D. T. Witte, and A. K. Smilde. New Indicator for Optimal Preprocessing and Wavelength Selection of Near-Infrared Spectra. *Applied Spectroscopy*, 2004. ISSN 00037028. doi: 10.1366/000370204322886591.

W. E. Stewart and M. Caracotsios. *Computer-Aided Modeling of Reactive Systems*. Wiley, 2008. ISBN 978-0-470-27495-8.

W. E. Stewart, M. Caracotsios, and J. P. Sørensen. Parameter estimation from multiresponse data. *AIChE Journal*, 38(5):641–650, 1992. ISSN 15475905. doi: 10.1002/aic.690380502.

D. M. Thierry and L. T. Biegler. Dynamic real-time optimization for CO_2 capture process. *AIChE Journal,*

published online, 2018.

R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, R. Quirynen, and M. Diehl. Towards a modular software package for embedded optimization. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 2006. ISSN 00255610. doi: 10.1007/s10107-004-0559-y.

A. Wächter and L. T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. URL `http://dx.doi.org/10.1007/s10107-004-0559-y`.

S. Wu, K. A. McLean, T. J. Harris, and K. B. McAuley. Selection of optimal parameter set using estimability analysis and MSE-based model-selection criterion. *International Journal of Advanced Mechatronic Systems*, 3(3):188, 2011. ISSN 1756-8412. doi: 10.1504/IJAMECHS.2011.042615. URL `http://www.inderscience.com/link.php?id=42615`.

K. Z. Yao, B. M. Shaw, B. Kou, K. B. McAuley, and D. W. Bacon. Modeling ethylene/butene copolymerization with multi-site catalysts: Parameter estimability and experimental design. *Polymer Reaction Engineering*, 11(3):563–588, 2003. ISSN 10543414. doi: 10.1081/PRE-120024426.

V. M. Zavala, C. D. Laird, and L. T. Biegler. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chemical Engineering Science*, 63(19):4834–4845, 2008.