SAND2018-14150C

PARMEST: PARAMETER ESTIMATION VIA PYOMO

David L. Woodruff[a][1], Andrea Staid[b], Bethany Nicholson[b], and Katherine Klise[b]
[a]Graduate School of Management, University of California, Davis, CA 95616
[b]Sandia National Laboratories, Albuquerque, NM 87185

Abstract
The ability to estimate a range of plausible parameter values, based on experimental data, is a critical aspect in process model validation and design optimization. In this paper, a Python software package is described that allows for model-based parameter estimation along with characterization of the uncertainty associated with the estimates. The software, called parmest, is available within the Pyomo open-source software project as a third-party contribution. The software includes options to obtain confidence regions that are based on single or multi-variate distributions, compute likelihood ratios, use bootstrap resampling in estimation, and make use of parallel processing capabilities.

Keywords
Parameter estimation, Confidence regions, Bootstrap, Likelihood ratio.

Introduction

Parameter estimation, the process of estimating model parameter values derived from a model and experimental data, is often used to calibrate models and to gain insight into complex physical processes that cannot be measured directly. In many cases, there is inherent uncertainty in our best estimate of parameter values, and this uncertainty should be used to inform design decisions. With quantified uncertainty estimates, we can create multiple scenarios of plausible parameter values to be used in design optimization. In this paper, we describe a Python software package that has been developed for model-based parameter estimation along with characterization of the uncertainty associated with the estimates. The software, called parmest, is based on the Pyomo modeling language [5, 6] and developed as part of the IDAES project `https://idaes.org/`. The software is available as a third party contribution within Pyomo (pyomo/contrib module in `https://github.com/Pyomo/pyomo`). The software package includes online documentation and examples within `https://pyomo.readthedocs.io`.

In parameter estimation, we seek to estimate values for a vector, θ, to use in the functional form

$$y = g(x; \theta)$$

where x is a vector, typically in high dimension, θ is a vector in much lower dimension and the response vectors are given as $y_i, i = 1, \ldots, p$ with p also much smaller than the dimension of x. This is done by collecting S data points called *samples*, which are $\tilde{x}, \tilde{y}$ pairs and then finding θ values that minimize some function of the deviation between the values of $\tilde{y}$ that are measured and the values of $g(\tilde{x}; \theta)$ for each corresponding $\tilde{x}$, which is a subvector of the vector x. Note that for most experiments, only small parts of x will change from one experiment to the next.

We start by assuming that the data points are indexed by $s = 1, \ldots, S$ and the parameters are fit

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

using

$$\min_{\theta} Q(\theta; \tilde{x}, \tilde{y}) \equiv \sum_{s=1}^{S} q_s(\theta; \tilde{x}_s, \tilde{y}_s) \tag{1}$$

where

$$q_s(\theta; \tilde{x}_s, \tilde{y}_s) = \sum_{i=1}^{p} w_i \left[\tilde{y}_{si} - g_i(\tilde{x}_s; \theta)\right]^2,$$

i.e., the contribution of sample s to Q, where $w \in \Re^p$ is a vector of weights for the responses. For multi-dimensional y, this is the squared weighted L_2 norm and for univarite y the weighted squared deviation.

Note that in general, the function $g(\cdot)$ often has a large set of parameters that are not included in θ. It is typically computationally expensive to compute $g(\cdot)$, and the computation may fail to provide a solution for some arguments.

The software package parmest solves Problem 1 to get a point estimate, θ^*. The software includes options to obtain confidence regions that are based on single or multi-variate distributions, compute likelihood ratios, use bootstrap resampling in estimation, and make use of parallel processing capabilities. The sequel describes these capabilities and includes a simple example.

Confidence Regions

Confidence regions provide an n-dimensional region that contains the true value of a parameter estimate with stated probability. Most confidence regions are defined by a probability $(1 - \alpha)$ that the true value of θ^* is in the region where α is an input parameter.

Rectangular

Rectangular confidence regions are formed using a combination of simple single-variate estimate where each dimension of θ is treated separately using the student-t distribution. The $1 - \alpha$ level confidence boundary for each dimension, j, is given by $\theta_j^* \pm t_{\alpha/2;\nu} s_i$, where $\nu = S - 2$ is the degrees of freedom and s_j is the sample standard deviation of θ for dimension j.

Ellipse

Elliptical confidence regions rely on strong assumptions about multi-variate normality of the distribution of θ in the region around θ^*. To generate an ellipse confidence region, it is very common to use the Hessian of the likelihood function at θ^* as the inverse of the covariance matrix Σ_θ^{-1} to demarcate ellipsoidal regions based on equi-distant values of the (squared) Mahalanobis distance, defined as

$$(\theta - \theta^*)^T \Sigma_\theta^{-1} (\theta - \theta^*)$$

where θ is considered to be a column vector. These distances obey a χ^2 distribution with S degrees of freedom, which endows the regions with a probabilistic interpretation. An implementation also based on Pyomo is described by Eslick et al. [4].

Likelihood Ratio

The likelihood ratio method was proposed for parameter estimation in Chemical Engineering applications by Rooney and Biegler [8]. To compute the likelihood ratio, the test statistic for univariate y is:

$$(S - 2)\left(\frac{\text{SSE}(\theta)}{\text{SSE}(\theta^*)} - 1\right)$$

where SSE refers to the sum of squared errors, i.e., $Q(\theta^*; \tilde{x}, \tilde{y})$. This is compared with the $\chi^2_{1-\alpha,S}$ statistic where S is the dimension of θ and $1 - \alpha$ is the level. The α confidence region is defined by θ vectors such that

$$(S - 2)\left(\frac{\text{SSE}(\theta)}{\text{SSE}(\theta^*)} - 1\right) \leq \chi^2_{1-\alpha,p}$$

Within parmest, the user specifies a multidimensional search space for the parameters of interest For large search spaces, parallel processing is essential. Currently, the same statistic is used for multi-variate y, but in future releases we hope to enhance that based on models presented by Seber and Wild [9].

Bootstrap

Bootstrap [3] is a widely used technique for resampling from a sample distribution in order to gain insight into the underlying parent distribution. In this context, bootstrap allows us to work with a much

Number of processors	Time (MM:SS)
1	10:52
2	5:35
4	3:24
8	1:42
16	0:56
32	0:36

Table 1: Parallel wall-clock timing results using mpi4py for bootstrap confidence region estimation for a semi-batch process.

larger dataset by resampling with replacement to generate a large number of additional datasets to be used for estimation. Within parmest, the user specifies the number of bootstrap samples, N, and the software creates that number of samples randomly, subject to the constraint that the number of unique experiments in the newly-drawn sample exceeds the dimension of θ. These samples can be used to form confidence intervals based on a multivariate normal or a kernel density estimation procedure.

Parallel computation

Parallelism is implemented within parmest using the mpi4py package. To demonstrate the value and limits of our implementation of this form of parallelism we tested bootstrap resampling on a simulated data set with $S = 14$ with $N = 128$ using an example from the literature [1] which estimates parameters of a semi-batch process. We used the Pyomo DAE package [7] to discretize the differential equations. Results shown in Table 1 were done on a computer with 96 fairly slow processors (2.1GHx Intel Xeon) and 1TB of main memory. As the table shows, the parallel efficiency deteriorates with more than eight processors, but the wall-clock time reduction is nonetheless substantial.

Example

To illustrate the use of the parmest package, here we use a reactor design model, originally presented in [2]. This model represents a continuously stirred tank reactor. Initially, the problem was presented as a design problem, with the goal of choosing the size of the reactor to maximize the production of one component. We instead assume a fixed size reactor and unknown rate constants for three different internal reactions (k1, k2, and k3). We can measure the output concentrations of three different components, which are then coupled with the model reaction equations to estimate the three rate constants.

The entire simulated dataset was used to estimate values for the internal reactions, resulting in k1 of 0.846, k2 of 1.66, and k3 of 0.000168. Bootstrap resampling was then used to gather information on uncertainty that surrounds those estimates. The dataset was resampled 200 times to generate bootstrap parameter estimates, shown in Figure 1.

Confidence regions for the same bootstrap resampling are shown in Figure 2 and demonstrate the differences between rectangular, multivariate normal, and kernel density estimated regions. A slice of each distribution is shown in the 2D figures, where the missing dimension is set equal to θ^*. For bootstrap resampling, parmest returns the multivariate normal and kernel density estimation distributions which can be used to generate scenarios of plausible parameter estimates.

Conclusions

This paper describes a new open source Python package that has been developed for model-based parameter estimation along with characterization of the uncertainty associated with the estimates. The software is distributed as part of Pyomo. Future development will focus on software modularity to support a wider range of process models, optimization objectives, and data formats.

Disclaimer

rity Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] Abel, O. and Marquardt, W. (2004). Scenario-integrated modeling and optimization of dynamic systems. *AIChE Journal*, 46(4):803–823.

[2] Bequette, B. W. (2003). *Process control: modeling, design, and simulation*. Prentice Hall Professional.

[3] Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA.

[4] Eslick, J. C., Akula, P. T., Bhattacharyya, D., and Miller, D. C. (2018). Simultaneous parameter estimation in reactive-solvent-based processes. In Eden, M. R., Ierapetritou, M. G., and Towler, G. P., editors, *13th International Symposium on Process Systems Engineering (PSE 2018)*, volume 44 of *Computer Aided Chemical Engineering*, pages 901 – 906. Elsevier.

[5] Hart, W., Watson, J., and Woodruff, D. (2011). Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3).

[6] Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., and Siirola, J. D. (2017). *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition.

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

[8] Rooney, W. C. and Biegler, L. T. (2001). Design for model parameter uncertainty using nonlinear confidence regions. *AIChE Journal*, 47(8):1794–1804.

[9] Seber, G. A. F. and Wild, C. J. (2003). *Nonlinear Regression*. Wiley.

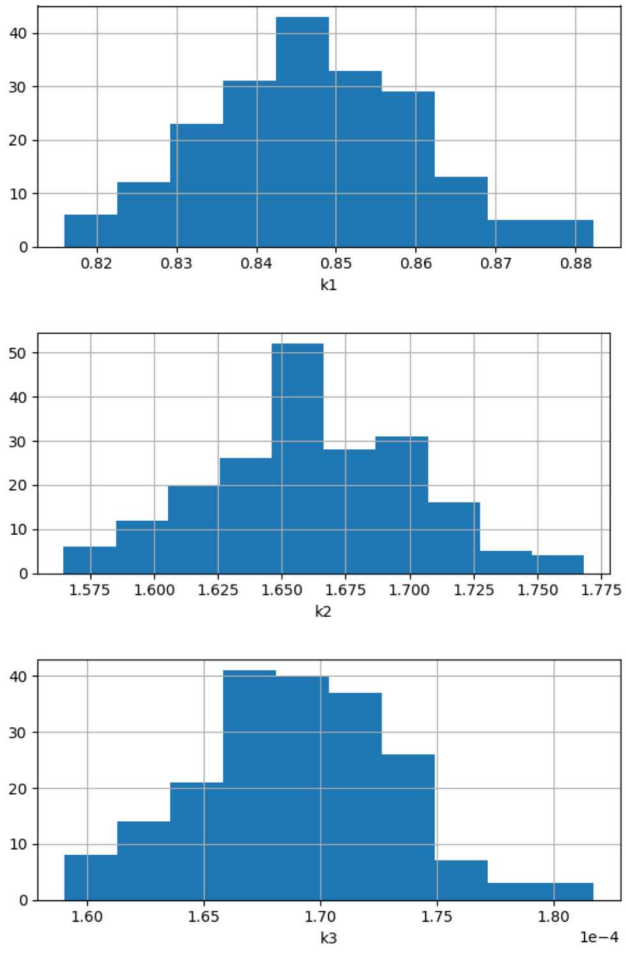

Figure 1: Distribution of parameter estimates using parmest with bootstrap resampling.

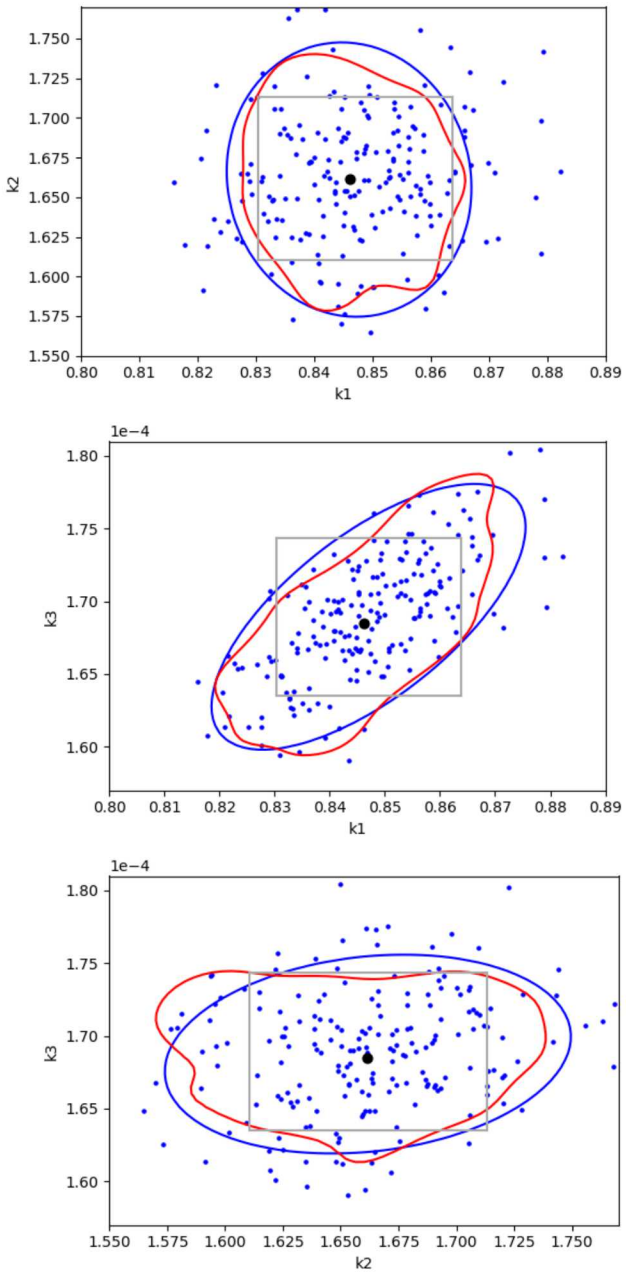

Figure 2: Parameter estimates and confidence regions using bootstrap resampling. Figures include confidence regions using $1 - \alpha$=0.8 (blue = mutli-variate normal, red = kernel density estimation, grey = rectangular), black dot shows the location of θ^*, and the blue dots are the parameter estimates.