An Adaptive Parallel Tempering Method for the Dynamic Data-Driven Parameter Estimation of Nonlinear Models

Journal:	*AIChE Journal*
Manuscript ID	Draft
Wiley - Manuscript type:	Research Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Armstrong, Matthew; US Military Academy, Chemistry and Life Science Beris, Antony; University of Delaware, Chemical Engineering Department Wagner, Norman; University of Delaware, Chemical and Biomolecular Engineering Department
Keywords:	Global Optimization, Parameter Estimation, Parallel Tempering, Differential Algebraic Equations, Large Amplitude Oscillatory Shear

An Adaptive Parallel Tempering Method for the Dynamic Data-Driven Parameter Estimation of Nonlinear Models

Matthew J. Armstrong*, Antony N. Beris, Norman J. Wagner**

Center for Molecular and Engineering Thermodynamics, Department of Chemical and Biomolecular Engineering. University of Delaware, Newark, DE 19716

*Present Address: Department of Chemistry and Life Science, United States Military Academy, West Point, NY 10996
**Corresponding Author. Tel. 1- 302- 831-8018
E-mail address: beris@udel.edu (A.N. Beris)

Wednesday, June 15, 2016

An adaptive parallel tempering algorithm is developed in a user-friendly fashion that efficiently and robustly generates near-optimum solutions. Using adaptive, implicit, time-integration methods, the method allows fitting model parameters to dynamic data. The proposed approach is insensitive to the initial guess and requires minimal fine-tuning: most of the algorithm parameters can be determined adaptively based on the analysis of few model simulations, while default values are proposed for the few remaining ones, the exact values of which do not sensitively affect the solution. The method is extensively validated through its application to a number of algebraic and dynamic global optimization problems from Chemical Engineering literature. We then apply it to a multi-parameter, highly nonlinear, model of the rheology of a thixotropic system where we show how the present approach can be used to robustly determine model parameters by fitting to dynamic, large amplitude, oscillatory stress vs. shear rate, data.

Keywords: Global optimization, Parameter estimation, Parallel tempering, Differential algebraic equations, Large amplitude oscillatory shear

Introduction

The evaluation of multiple parameters involved in nonlinear models based on dynamic data is often a non-trivial step. As in any parameter estimation, this is based on the minimization of the sum of the squares of the differences between the model predictions and the experimental data. However, the use of a local method, such as in classical least squares, using a local, gradient method[1], only converges to a

local minimum that is not necessarily the global one. Thus, the answer typically depends crucially on the initial guess. Furthermore, an analysis of the sensitivity of the local minimization solution to the initial guess can be time consuming. This is especially true if dynamic data are used, as then the model predictions require the time-integration of a set of ordinary differential equations (ODEs), which is itself computationally costly. There are several variations of this basic strategy. Some examples are the general Newton's method, Quasi-Newton, trust-region methods, line search methods, and Levenberg-Marquardt methods[1,2]. The use of some of those local methods, as well as other variants that can also accommodate constraints, such as the Nelder-Mead-Simplex and the sequential quadratic programming, in the particular problem of the parameter estimation of nonlinear models based on dynamic data, is illustrated by Yuceer *et al.*[3] Alternatively, stochastic methods have been developed, such as Simulated-Annealing[4,5], in order to better explore the available parameter space in search of a global minimum and/ or alleviate the dependence of the solution on the initial guess.

Two encompassing references that cover the current state of minimization algorithms are *Nonlinear Programming Concepts, Algorithms, and Applications* by Biegler[6] and *Introduction to Stochastic Search Optimization* by Spall[7]. The first reference tends to focus on the direct search, local methods[6], while the latter has a focus on global stochastic methods, such as the simultaneous perturbation stochastic approximation, and simulated annealing algorithms[7]. In addition to these, *Nonlinear Regression* by Seber and Wild[8], offers a historical account of classical approaches using direct methods. Moreover, in *Numerical Recipes in Fortran* by Press *et al.*[2] there are several detailed explanations and algorithms developed both for these local direct methods as well as for the simulated annealing. Furthermore, an outstanding description of the available direct methods is provided by both MATLAB and Mathematica, in their documentation pages[9,10]. *Deterministic Global Optimization: Theory, Methods, and Applications* by Floudas[11] and *Global Optimization Deterministic Approaches* by Horst[12] represent further repositories of deterministic global optimization algorithms and approaches. A concise description of deterministic global optimization tools and their benefits in their application to solve systems engineering and

computational biology problems can be found in work by Floudas[13] with a description of more recent advances to be found in the review by Floudas and Gounaris[14].

In dynamic data-driven parameter estimation the evaluation of the objective function to be optimized is often prohibitively expensive. Under such conditions, the use of direct methods for global optimization may force one may to employ various approximations of the function to be optimized in building surrogate (response) surfaces[15,16], such as those obtained through a stochastic simulator (also called Kriging[15-18]) or through radial basis function approximations[19]. More recent developments involve the use of a derivative-free optimization for expensive constrained problems[18]. These methods were developed in order to handle global optimization in systems with high uncertainty. Nevertheless, for complex systems involving many parameters and considerable uncertainty, such as found in biology, the current tools that combine a local nonlinear programming method with a stochastic global search[20,21], such as deterministic global optimization methods, tend to become unfeasible[21].

Alternatively, there are many global optimization methods based on stochastic approaches, starting with the simulated annealing, as already mentioned above[4,5]. Its standard form utilizes the Metropolis Monte-Carlo algorithm that was originally developed as a numerical tool to help to evaluate thermodynamic behavior by coercing the selection of initially randomly selected microstates, through their weighting by a Boltzmann factor, towards the generation of a representative to the true thermodynamics population of microscopic samples at a given temperature[22]. In the global optimization application the thermodynamic potential is replaced with the function to be minimized[4,5]. Furthermore, the thermodynamic temperature is replaced by an effective one (representing a characteristic value of the objective function to be minimized) and instead of being constant (as in a traditional Monte-Carlo thermodynamic simulation) is now allowed to vary. This variation is selected to emulate the annealing process in solidification, so as to ensure that the global minimum is eventually reached when the final temperature in the simulation approaches zero[4,5]. Various temperature variation schemes have been developed over the years with some of the most successful discussed in standard numerical analysis treatises[2]. Recently various variants of the classical simulated annealing have also been developed, such

as the parallel simulated annealing[23] and the coupled simulated annealing[24-25] that allowed exchanges of state information between simultaneous simulated annealing runs in global optimization. However, as discussed in a recent review of the subject of the use of stochastic methods in global optimization[26], despite its successes the simulated annealing and its variations suffer from slow convergence and/or high sensitivity to the annealing schedule which is highly problem-dependent.

In addition to the simulated annealing and its various variants discussed above, several other stochastic methods have over the years been developed and successfully applied to a number of global optimization problems[7,27-31] such as genetic algorithms, evolutionary algorithms, the particle swarm method, etc. The genetic algorithms[27,30] try to reach the global optimum by following the evolution of a population of states generated randomly by first codifying the model into a genetic framework and then letting the "genes" evolve stochastically by promoting the evolution of the best "fit" models, their fitness being judged from a comparison of their predictions to available data. An alternative approach is the Simultaneous Perturbation Stochastic Approximation (SPSA)[7], which uses a stochastic approach to generate, model parameters, evaluate their performance though comparison of their predictions to the data and then direct their evolution through a local minimum approach by estimating the Jacobian (which is in this case the Hessian) to guide the direction of the optimization. More recent progress in this field includes the differential evolution algorithm, which represents an alternative implementation of the genetic algorithms and has also been used to estimate parameter values[31]. In addition, a further refinement of the genetic algorithms is the harmony search algorithm[32], which uses all the previous generation solutions to generate the new guesses instead of just two parents. This has been further improved and applied to constrained and unconstrained algebraic minimization problems[33] whereby a novel method of generating solution vectors has been proposed. This supposedly enhances the accuracy and rate of convergence[33]. However, a common feature among all of these stochastic global optimization methods, similar to the simulating annealing algorithm, is the significant number of adjustable algorithm parameters that must be tuned for each problem to make the algorithms successful. Consequently the methodology for their selection is unclear.

Nevertheless, there has been a significant extension of the simulated annealing method, the parallel tempering, which appears to overcome at least some of the drawbacks associated with simulated annealing by altogether avoiding the use of, and therefore the need for, a temperature annealing schedule. Instead, the parallel tempering approach (also called "Replica-exchange Monte-Carlo") employs the use of multiple Monte Carlo runs, which are all executed in parallel at constant but different temperatures, with the possibility of infrequent but consistent exchanges between adjacent temperature runs[26,34,35]. However, it is of interest that although the parallel tempering approach has been used, quite successfully, to a number of global optimization problems arising in molecular systems-based physical chemistry problems[36-40], and despite its general accolades and substantial computational improvements[35,38,39,41], its adoption for use in more general optimization problems has been unexpectedly slow, with only few applications reported to date[38,42-45]. In particular, there have been very few applications for general parameter-estimation problems[44], of which, as far as the authors know, none involves dynamic data, which is the subject of the present investigation. The main explanation for this lack of progress is believed to be that the current implementation of the parallel tempering approach has not taken full advantage of recent know-how on the development of critical algorithmic parameters adaptively (such as the temperature values or the frequency of replica exchanges) implemented under a more general global optimization setting, thereby significantly reducing the need of fine tuning of algorithmic parameters. This is exactly what we aim to offer here in the particular context of dynamic data-driven parametric estimations.

Indeed, despite all the activity in implementing efficient global optimization techniques, only a small part of which has been referenced above, there is still a need for robust formulations that can be easily implemented in real applications. For example, this need has been recently acknowledged in work by Nallasivam *et al.*[46] in connection to the optimization of multicomponent distillation configurations. In that work the weaknesses of the sequential minimization algorithm currently used for multicomponent distillation configurations are made apparent[46]. It is shown that as the number of components in the stream is increased the algorithm becomes less reliable. From these recent examples it is clear that there

is a current need for a user-friendly, yet robust global optimization algorithm, with insensitive, but intuitive adjustable algorithm parameters. The new parallel tempering algorithm proposed here circumvents both of these deficiencies by providing a capability, fully exploiting the power of Monte-Carlo methods[47], to obtain a near-global minimum without requiring either a very good initial guess or finely tuned algorithmic parameters, and also has the capability to integrate constraints seamlessly.

The method described in this work is applied to multiple parameter estimation problems in dynamic systems, such as those describing the dynamics in complex chemical reaction systems, biological systems, complex rheological systems and others. The algorithm is validated against several model problems involving systems of chemical reactions[6,11,48] that they have been employed before in testing global parameter estimation methods. The efficacy of the algorithm is also demonstrated through its application to complicated algebraic systems with known multiple minima[18,49]. Finally, the algorithm is applied to the parameter estimation in fitting complex nonlinear rheology models to measurement data. Such experiments include several dynamic sets of experiments, such as large amplitude oscillatory shear data (LAOS)[50-54]. Large amplitude oscillatory shear flow is a regime used in rheological systems to physically probe a nonlinear mechanical response, both temporally and spatially, either by programming a sinusoidal strain oscillation and measuring the stress response or vice versa, using a rheometer. LAOS presents a unique way to measure and quantify material properties using nonlinear flow conditions[50-54]. In the following it is shown that our approach ensures, for reasonably complex realistic problems, the reproducible approximation of the global minimum independent of the initial guess in a systematic way and with very few adjustable algorithmic parameters that also do not sensitively affect the performance of the method. Consequently, we anticipate application to be of value for a broad range of parameter fitting challenges concerning dynamic data.

The structure of the rest of the paper is as follows. In the next section we present the problem formulation. Following this, we offer a brief overview of necessary background information, consisting of highlights from simulated annealing and parallel tempering. The proposed algorithm is then described in detail in the next section, dynamic parameter estimation. The implementation of the parallel tempering

is performed so as to avoid adjustable parameters with high solution sensitivity. Default values are offered for the remaining few numerical parameters needed, while sensitivity analysis in the following section shows the solution to be fairly insensitive to their values. In the subsequent sections a detailed set of parametric evaluation tests is offered. These involve problems from the literature, starting from simple algebraic ones with known multiple minima, some also involving constraints, the more complex chemical reaction network models that have been typically used in the past as benchmarks for dynamic parametric estimation evaluations, leading to a novel application in nonlinear dynamic rheology. Lastly we state our conclusions.

Problem Formulation

The dynamic modeling of all the systems of interest can be described as follows. First let as denote by a vector $\mathbf{q}$, the N internal model parameters

$$\mathbf{q} \equiv \left[q_1, q_2, \ldots q_N\right]^{\mathrm{T}}. \tag{1}$$

In addition to the N internal parameters of the model, we also have, in general, P external, control, parameters. Those are user-specified, and control the experimental conditions. Those form the elements of the P-vector $\mathbf{z}$ and can also be a function of time:

$$\mathbf{z} = \left[z_1, z_2, \ldots z_P\right]^{\mathrm{T}} = \left[z_1(t), z_2(t), \ldots z_P(t)\right]^{\mathrm{T}}, \tag{2}$$

Where both $\mathbf{q}, \mathbf{z}$ contribute, in general, to a system of J ordinary differential and K algebraic equations:

$$\frac{\mathrm{d}\, y_j}{\mathrm{dt}} = f_j\left(t, \mathbf{y}, \mathbf{q}, \mathbf{z}(t)\right); \quad j = 1, 2, \ldots, J, \tag{3}$$

$$0 = g_k\left(t, \mathbf{y}, \mathbf{q}, \mathbf{z}(t)\right); \quad k = 1,2,...,K. \tag{4}$$

Those equations describe the time evolution of the J+K dynamic variables of the system:

$$\mathbf{y} = \left[y_1, y_2, ...y_{J+K}\right]^{\mathrm{T}}, \tag{5}$$

subject to a commensurate, J, number of initial conditions

$$y_j\left(t_0\right) = x_j\left(\mathbf{q}, \mathbf{z}(t_0)\right); \quad j = 1,2,...,J. \tag{6}$$

Furthermore, the N model parameters are considered to be subject to a number of constraints

$$a_n \le q_n \le b_n; \quad n = 1,2,...,N. \tag{7}$$

Note that as the parameters a_n, b_n in Eq. (7) are also allowed to take the limiting $-\infty, \infty$ values, respectively, these equations can represent generically, both the presence or absence of limiting constraints on the parameter values[11].

The general dynamic parameter estimation problem can then be defined as finding the N internal parameters, $\mathbf{q}$, of the above-defined dynamic model based on available dynamic (and possibly also static) experimental data. In general, let us assume that the experimental data involve L sets of dynamic and M sets of static (or steady-state) experimental data corresponding to the following measurable quantities:

$$\tau_l = \tau_l\left(t, \mathbf{y}(t), \mathbf{q}, \mathbf{z} = \mathbf{z}^{(l)}(t)\right); l = 1,2,...,L; \alpha_l \le t \le \beta_l, \tag{8}$$

where $\mathbf{z}^{(l)}(t)$ is a user-prescribed function that defines the time evolution of the control variables corresponding to the *l*-th set of dynamic experimental data, defined over the time interval (α_l, β_l) and

$$\tau_{m,j} = \tau_{m,j}\left(\mathbf{y}, \mathbf{q}, \mathbf{z} = \mathbf{z}^{(m,j)}\right); m = 1,2,...,M;\ j = 1,2,...,N_m\ , \tag{9}$$

where $\mathbf{z}^{(m,j)}$ is a user-prescribed vector that defines the set of control variables corresponding to the *j*-th discrete point of the *m*-th set of static experimental data (i.e. there are *M* sets of data, and each of the data sets has N_m discrete data points). Note that each one of the *L* dynamic sets involves, in general, continuous variables with a continuous dependence on time over a specific interval, whereas each one of the *M* static sets involves a finite discrete number of data points that do not involve time. It is important to make that distinction even when in actual applications one ends up having to deal with discrete dynamic data sets, as the experimental data are practically collected over a finite, discrete, number of times. This is because it is best (and most practical) to explore the continuity of the dynamic data by developing continuous approximations of the experimental measurements. It is through those continuous approximations that the dynamic data enter in the formulation of the objective function, as described below. This is even more critical when the dynamic data involve a time-periodic behavior, which can be most appropriately captured by using time-periodic functional approximations, such as those provided by a Fourier series expansion. In this case the actual values of the limits α_l, β_l are not defined; simply their time span (period)

$$T_l \equiv \beta_l - \alpha_l; \quad \text{for a periodic dynamic behavior of } l\text{-th data set}\ . \tag{10}$$

Note that this time-periodic behavior can either be forced on the system through the imposition of a suitably time-periodic history (forced oscillation) on the control parameter vector, $\mathbf{z}^{(l)}$ or, sometimes, it can be spontaneously observed even for time-independent control parameter values, such as from a naturally occurring limit-cycle behavior. In the first (more common) case, the fundamental period is externally imposed, whereas in the second it is one of the model predictions.

A crucial item to be decided is the construction of a suitable objective function, F_{obj}, the minimization of which leads to the evaluation of the model parameters. A reasonably general form, based on a weighted sum of the square of the differences between the L continuous sets and M discrete sets of experimental data

$$\left[\left\{\left(\hat{\tau}_l(t);\alpha_l \le t \le \beta_l\right);\ l=1,2,...,L\right\};\ \left\{\left(\hat{\tau}_{m,j};j=1,2,...,N_m\right);m=1,2,...,M\right\}\right], \tag{11}$$

and the corresponding model predictions provided by Eqs. (8)-(9), is

$$F_{obj}=\sqrt{\sum_{l=1}^{L}\frac{1}{t_l}\int_{\alpha_l}^{\beta_l} w_l\left(\hat{\tau}_l(t)-\tau_l(t)\right)^2 dt+\sum_{m=1}^{M}\frac{1}{N_m}\sum_{i=1}^{N_m} w'_{m,i}\left(\hat{\tau}_{m,i}-\tau_{m,i}\right)^2}, \tag{12}$$

where t_l are characteristic times of the l-th dynamic set of data defined as

$$t_l=\begin{cases} T_l & \text{l-th time domain is periodic} \\ \lambda_l & \text{characteristic relaxation time} \end{cases}, \tag{13}$$

and where $w_l=w_l(t),\alpha_l\le t\le\beta_l; l=1,2,...,L$ and $w'_{m,i}; m=1,2,...,M; i=1,2,...,N_m$ are positive weight, continuous functions and discrete coefficients, respectively. Note that in association with dynamic

data gathered within a non-time-periodic experiment, it is assumed here that the behavior to be captured is some type of relaxation with

$$\hat{\tau}_l(t) \xrightarrow[t=\beta\to\infty]{} 0 \ , \tag{14}$$

i.e., the data to be fitted represent the departure from the limiting value reached at infinite times. It is only with respect to this formulation that the convergence of the corresponding integral can be guaranteed as the length of the observation time window $\beta_l - \alpha_l \to \infty$, also under the assumption that, at least for long times, the behavior is decreasing in a sufficiently rapid fashion such as, for example, described by the decaying exponential

$$\hat{\tau}_l(t) \xrightarrow[t\to\infty]{} \hat{\tau}_{l,\infty} \exp(-\,{}^{t}\!/_{\lambda_l}) . \tag{15}$$

Whenever applicable, Eq. (15), can also used to define the characteristic relaxation time λ_l with equivalent expressions used under other circumstances. Now, if the limiting value is trivial (zero) this is all that is required. If it is not (i.e., when the limiting value is anticipated to be non-zero and model-dependent) then that limiting value should also appear on its own right, as part of static data, in the definition of the objective function.

An important factor in the quality of the fit is the choice for the weight functions. Only an appropriate choice to the available information will allow for a balanced fit that is consistent and optimum. In particular, any statistical noise information should also be used to advantage. The expression provided by Eq. (12) can accommodate noise and uncertainly that is unequally distributed between the elements of any given set. Under such conditions the statistically best fit is provided by taking the weights to be inversely proportional to the variance of the corresponding data[55]. Thus, as far as the dynamic data are concerned, the weight functions should be determined as

$$w_l(t) = c_l \frac{1}{Var_l(t)}, \tag{16}$$

where $Var_l(t)$ describes an estimate of the variance of the corresponding $\hat{\tau}_l(t)$ data, and c_l a dimensionless scaling factor of the *l*-th dynamic set of the data. Similarly, as far as the static data are concerned, the weights should be determined as

$$w'_{m,i} = c_m \frac{1}{Var_{m,i}}, \tag{17}$$

where $Var_{m,i}$ describes an estimate of the variance of the corresponding $\hat{\tau}_{m,i}$ data, and c_m a dimensionless scaling factor of the *m*-th static set of the data. Note that as the absolute value of the variance in any given set can always be absorbed within the weight coefficient c_l only the relative variance variations within a given set are important to be defined.

In the absence of any specific, data-dependent information, the variance in either case may be estimated. Of importance is to distinguish two limiting cases (illustrated here for the static case, but the dynamic case can be handled in a similar way): a) where the variance is taken to be proportional to the square of the data value

$$Var_{m,i} = \left(\hat{\tau}_{m,i}\right)^2, \tag{18}$$

with the proportionality coefficient absorbed within the dimensionless scaling factor c_m and b) where the variance is taken to be some fixed absolute value

$$Var_{m,i} = \left(\hat{\tau}_{m,0}\right)^2, \tag{19}$$

where $\hat{\tau}_{m,0}$ is some reference value for the *m*-th set of the data and, where, again, any proportionality coefficient is absorbed within the dimensionless scaling factor c_m. Note that in all cases, given the dependence of the variance to the square of the data values, the scaling factors for the weights are dimensionless as well as the objective function.

As far as the scaling factors are concerned, those can be defined based on a) the relative weights for each one of the dynamic and scalar data sets and b) the absolute maximum value desired for the objective function. In the absence of any specific information to the contrary, a safe choice (and one that is followed throughout this work) is to use equal relative weights and an absolute value that provides for a maximum objective function the value of unity. For simplicity and generality the model predictions used to determine the maximum objective function is one are all assumed zero. This has as a side effect to always allow for a safe choice of the maximum Boltzmann energy, $E_{Bhot} = 1$, in the proposed parallel tempering-based methodology discussed below, thereby eliminating this from the parameters that need to be determined to run that method. To better understand how that method works, we first need to provide an overview of some related background information, below.

Background Information

Parallel tempering overview

The simulated annealing algorithm is a stochastic method that attempts to direct a biased "random" walk through the allowed parameter space to that set of parameter values that best minimizes an objective function, such as the one defined in Eq. (12)[4,5]. This biased random walk through parameter space is generated using the same Metropolis algorithm that is used in the more familiar Monte Carlo (MC) stochastic simulations in statistical physics[2,22]. Simply, here the objective function plays the role of

microstate energy and the parameter space that of the physical microstate space. Briefly, the stochastic simulation proceeds as follows. Starting with an initial guess for the parameters, $\mathbf{q}_{old}$, the corresponding (initial) value of the objective function, $F_{obj,old} \equiv F_{obj}\left(\mathbf{q}_{old}\right)$, is generated. Then, a new set of parameter values, $\mathbf{q}_{new}$, is proposed based on a random perturbation of the previous set of parameters. The new value of the objective function, $F_{obj,new} \equiv F_{obj}\left(\mathbf{q}_{new}\right)$, corresponding to the new set of parameter values is then calculated. To complete the MC step, the new state is accepted (and thus replaces the previous "old" state) based on the Metropolis acceptance probability, P_{accept}:

$$P_{accept} = \begin{cases} 1 & F_{obj,new} \le F_{obj,old} \\ \exp\left(-\dfrac{\left(F_{obj,new} - F_{obj,old}\right)}{E_B}\right) & F_{obj,new} > F_{obj,old} \end{cases}, \quad (20)$$

where E_B is a scaling factor (Boltzmann energy; also denoted in physical MC simulations in terms of the system's temperature as $k_B T$ where k_B is the Boltzmann constant). A cartoon of a prototypical cooling schedule is shown below.

Figure 1. Schematic of a typical simulated annealing cooling schedule; here "Time" is proportional to MC steps.

The parallel tempering is a stochastic method that attempts to circumvent the extreme parameter sensitivity to the cooling schedule of the standard simulated annealing procedure by concurrently generating a number of parallel MC sequences. Each one of these (say N_{run} in number) MC sequences is executed in parallel, advancing each one step at a time, at different (but constant) Boltzmann energy levels, $E_{B1} > E_{B2} > \ldots > E_{BN_{run}}$. Simultaneously, additional state exchanges between these sequences are

allowed to take place, but these are infrequent, and only occur between adjacent energy levels sequences[34-36]. Still those exchanges are sufficient to allow the potential "trickling-in" of good, low objective function, states from a high Boltzmann energy level (where there is a much higher probability of randomly generating a low value objective function, given the wider range of the parameter space explored under those conditions) to the lowest Boltzmann energy level (where the chance of "loosing" that good guess due to a random "uphill" move, is very improbable)[7,34-38,47]. Thus, the annealing can take place naturally without the need to implement a cooling schedule. Furthermore, the fact that the state exchanges are infrequent provides an advantage as the parallel tempering algorithm can be well executed in parallel, thus significantly minimizing the computational time. Indeed this has been exploited in the past,[2,47] albeit it was not necessary in the examples shown here. Here is the approach:

1. Start with a selection of the Boltzmann energy levels, from a maximum, $E_{B1} = E_{B,Hot}$, to a minimum $E_{BN_{run}} = E_{B,Cold}$; with a recommended schedule for intermediate Boltzmann energy levels:

$$E_{B1} = E_{B,Hot} > E_{B2} > ... > E_{BNrun} = E_{B,Cold}, \tag{21}$$

$$E_{B(i+1)} \Big/ E_{Bi} = \left(E_{B,Cold} \Big/ E_{B,Hot} \right)^{\left(1/(N_{run}-1)\right)}; \quad i = 1,2,...,N_{run}-1. \tag{22}$$

2. The proper selection of the values of the Boltzmann Energy vector, $E_B = (E_{B1}...E_{BN_{run}})^T$, can be verified by evaluating the probability distribution functions (pdf) corresponding to any parameter p, that are obtained from each one of the parallel MC sequences corresponding to the N_{run} Boltzmann energy levels shown in Figure 2 with data from the catalytic cracking of oil example[6].

Figure 2. Plot of pdfs for the state quantity p=k_1 reaction rate constant based on normal Probability density distribution fits using statistics data ($\mu_{k_1,i}, \sigma_{k_1,i}$) drawn from Catalytic Cracking of Gas example[6] as shown in Table 1. The Boltzmann energy

levels corresponding to the pdf curves shown are indicated on the right of the figure.

Table 1. $\mu_{k_1,i}, \sigma_{k_1,i}$ statistics from a sample run of Catalytic Cracking of Gas example[6]

The normal density pdfs used in Figure 2 are:

$$pdf: f_N(x) = \frac{1}{\sqrt{2\pi}\sigma_N} \exp\left\{-\frac{1}{2}\left(\frac{x-\bar{x}_N}{\sigma_N}\right)^2\right\} . \quad (23)$$

3. The next item to be specified is the number of steps, N_{Ex}, after which a chance will be offered to a given MC run (except to the first) to exchange its state, he exchange takes place according to an acceptance probability P_{accept} that is defined in terms of the objective functions corresponding to the "warm" and "cool" states as[34,35,56]

$$\mathrm{P}_{accept} = \begin{cases} 1 \text{ if } \mathrm{F}_{obj,cool} > \mathrm{F}_{obj,warm} \\ \exp\left\{\left(\frac{1}{E_{B,cool}} - \frac{1}{E_{B,warm}}\right)(F_{obj,cool} - F_{obj,hot})\right\} \end{cases} . \quad (24)$$

4. For the determination of the number of steps follow the work of Bittner *et al.*[35], and the autocorrelation function, R_k, which is a function of the step difference *k*, calculated with meta-data obtained on a state quantity, p, collected during an initial trial run of the algorithm as

$$R_k \equiv \frac{\sum_{i=1}^{Nsteps}(p_i - \bar{p}_0)(p_{i+k} - \bar{p}_k)}{\sqrt{\left(\sum_{i=1}^{Nsteps}(p_i - \bar{p}_0)^2\right)\left(\sum_{i=1}^{Nsteps}(p_{i+k} - \bar{p}_k)^2\right)}}, \quad (25)$$

where the subscripts denote the MC step number and the overbar the average calculated over a set number of steps, $Nsteps$. The k values for which the autocorrelation function R_k = 0.5, $k \equiv k_{0.5}$, define an optimum point of exchanges, $N_{Ex} = k_{0.5}$. We choose R_k = 0.5 because this is a typical value to reveal the characteristic time scale for the decay of correlations---1/e could also be used with no significant changes to the results. This is shown in Figure 5a,b calculated based on data corresponding to catalytic cracking of oil example, discussed in Dynamic Parameter Estimation Examples section, below as calculated for the state quantity p=k_1 reaction rate constant.

Figure 3. Autocorrelation function for a) E_B=1 and b) E_B=10^{-5} (*red arrow indicates $N_{Ex} \equiv k_{0.5}$).

Comparing Figures 3a and 3b shows that each Boltzmann Energy level will have its own unique N_{Ex} that can be obtained via an autocorrelation function calculated with metadata obtained from the corresponding Monte-Carlo sequence. $\mathbf{N}_{Ex}$ is therefore a vector of length that is identical to the number of Boltzmann Energy levels. Furthermore, as also can be realized from a comparison of Figures 3a and 3b, due to a faster decaying correlation the "hotter" tracks will request an information exchange, on average, more than their "colder" track counterparts, i.e. we anticipate:

$$\mathbf{N}_{Ex} = [N_{Ex,Hot}, N_{Ex,2}, ..., N_{Ex,Cold}]^T : \quad N_{Ex,Hot} < N_{Ex,2} < ... < N_{Ex,Cold}. \quad (26)$$

5. An approximation can be developed for $\mathbf{N}_{Ex}$ that only requires the determination of $N_{EX,Hot}$ and $N_{EX,Cold}$ alone (i.e., based on the autocorrelation functions corresponding to the two extreme cases) by using a formula similar to that for E_B, i.e. assuming that N_{Ex} always increases from one Boltzmann energy level to the next by a constant ratio:

$$N_{Ex,(i+1)} \Big/ N_{Ex,i} = \left(N_{Ex,Cold} \Big/ N_{Ex,Hot} \right)^{\left(1/(N_{run}-1)\right)}; \quad i = 1,2,...,N_{run}-1 \quad . \tag{27}$$

In this way, only two MC runs are needed at the hottest and coldest Boltzmann energy states, respectively. The third and last item that needs to be specified, how the selection of the proposed new random parameter states is taking place, is problem-dependent and will be addressed at the dynamic parameter estimation methodology section.

In summary, the parallel tempering method can be visualized with the following schematic diagram, shown in Figure 4:

Figure 4. Parallel tempering graphic depiction. Arrows depict the flow of information between parallel MC runs whereas the color indicates qualitatively the Boltzmann energy level magnitude at which each run is being carried out, red being higher, i.e. "hotter" and blue lower i.e. "colder."

Dynamic Parameter Estimation

The core of the proposed methodology for the parameter estimation, the parallel tempering, has been described in the *Parallel Tempering Overview section*. However, the key advantage of the methodology is the minimization of the need to finely adjust numerical parameters. Instead, all the required information to which the parallel tempering algorithm is sensitive to is proposed to be generated adaptively from a limited number of initial Monte-Carlo runs together with several dynamic simulations based on an initial set of model parameters.

Parallel tempering implementation

First, the number N_{run} of the parallel tempering runs needs to be selected (a default value of 15 is proposed; as it will be shown later, the results vary little when N_{run} varies between 10 and 20). Then, with the upper limit for the Boltzmann energy determined from the objective function normalization as $E_{B,Hot}=1$, only the lower limit, $E_{B,Cold}$, or, equivalently, the ratio $\frac{E_{B,Hot}}{E_{B,Cold}}$, needs to be selected. We recommend a value of $\frac{E_{B,Hot}}{E_{B,Cold}}=1e5$, albeit, this is another variable that can vary considerably without significantly affecting the results, as it will be shown later. The span of the Boltzmann Energies, E_B, are selected following a power law relationship indicated in Eq. (22).

Next, of critical importance is the specification of the process of selecting the proposed new state variables (i.e., the new parameter vector, $\mathbf{q}_{new}$) based on a previous value, $\mathbf{q}_{old}$. As most often one has to deal with parameters that are strictly positive (and even when this is not the case the problem can be easily reduced to such a case by examining separately the cases where a parameter can be positive or negative and in the latter instance replace the parameter by its negative) it is with no loss of generality that we consider each one of the components of the parameter vector, $q_n;\quad n=1,2,...,N$, to be strictly positive

$$q_n > 0; \quad n = 1,2,...,N. \tag{28}$$

Under those conditions, it is also sometimes advantageous to apply the new parameter values generation process to the logarithm of the original parameter (if there is high uncertainty on the order of magnitude of the parameter value)

$$\theta_n \equiv \ln q_n, \tag{29}$$

while, otherwise, we can also work directly with the parameters, assumed to be of order 1.

Then, the parameter selection process proceeds as follows. For each component n of the log parameter vector θ_n the new value is selected based on the selection of a normal random variable ζ (i.e., which obeys the normal distribution with zero mean and unity standard deviation, N(0,1) as

$$\theta_{n\,new} = \theta_{n\,old} + \sigma_n \zeta, \tag{30}$$

where the standard deviation σ_n is taken to be a function of both the component n as well as (empirically) the Boltzmann energy E_{Bj} of the corresponding MC j-th run, as

$$\sigma_n = (\frac{E_{Bj}}{E_{B,Hot}})^{\frac{1}{p}} A_n, \tag{31}$$

where the scaling parameter A_n is obtained from the statistics of an initial run (performed at the higher Boltzmann energy and with unity standard deviation) corresponding to the n-th parameter's standard deviation and where the exponent p is taken as the order of accuracy of the ODE solver, typically here p=5. The rationale behind Eq. (31) is to let the parameter variation decrease in a matter roughly proportional to the time step size used in the numerical integration.

The recursive generation of proposed parameter guesses is then given for each one of the N_{run} parallel MC runs by Eq. (30) with the $F_{obj,new}$ determined based on the new (proposed) parameter values following Eq.(12). Additionally, to improve the effectiveness of the algorithm, based on the fact that you need higher accuracy (and therefore tighter error criteria) as the Boltzmann energy decreases, we have

introduced a direct correlation between the time step size used in the dynamic DAE/ ODE solver of the code (as well as in the numerical integration required to evaluate the integral contributions of the objective function entering Eq. (12)) and the Boltzmann Energy level. It is noted that excellent sources of DAE/ ODE solvers can be found in the literature[57-64]. The numerical integration is typically performed by using a high order algorithm, such as Simpson's rule[2,65] for non-periodic domains and the trapezoidal rule for periodic ones. All that is needed to define the proper integration time step is to correlate the targeted relative integration error, ε, to the Boltzmann Energy level, E_{Bj}, used. As a first approximation, we can assume the two to be directly proportional

$$\varepsilon = Error_{MAX}^{E_{Bj}/E_{B,Hot}}. \tag{32}$$

That leaves the maximum error, $Error_{MAX}$, as the only adjustable parameter. Again, this is a parameter to which the solution does not depend sensitively; a value of $Error_{MAX}$=0.001 is used which seems to be backed up with a sensitivity study that is presented later. For a p-th order integration method this implies a time step size h that can be found iteratively from the time step, h_{test}, used in a test integration that has resulted to a relative integration error ε_{test} as

$$h = h_{test}\left(\frac{\varepsilon}{\varepsilon_{test}}\right)^{\frac{1}{p}}. \tag{33}$$

Similarly, the influence of the time step size used within the numerical integration of the ODE/DAE to the accuracy of the solution can be determined separately from its influence to the error resulting from the numerical integration of the dynamic data involved in the evaluation of the objective function as given by Eq. (12). Then the time step size used can be chosen as the highest one that fulfills all error requirements. In consequence, the order of accuracy of the ODE/DAE system has to be commensurate to the order of accuracy of the numerical integrator (typically a fourth order Simpson method).

The other parameters needed specifically for the parallel tempering is the vector of the MC steps before an exchange is attempted, $\boldsymbol{N}_{EX}$. This can be obtained adaptively by running two test Monte-Carlo

runs, one at the highest and one at the lowest Boltzmann energies. Evaluation of the autocorrelation functions during these two runs enables obtaining estimates for the minimum and maximum values of N_{EX}, with all other values calculated following Eq. (27).

Lastly, an important consideration is also to decide on a sensible way when to stop the algorithm. There are several options there. What we have followed is a simple criterion where the calculations are terminated if, within a certain number of MC steps, the improvement obtained in the objective function, $F_{obj,Best}$, (defined as the smallest value of F_{obj} over all of the parallel runs) is smaller than the smallest Boltzmann energy $E_{B,Cold}$. As a reasonable estimate of the number of MC steps over which the change in the objective function has been evaluated we used a small multiple, N_{min}, of the maximum N_{EX}, $N_{Ex,Cold}$:

$$\Delta F_{obj,Best} \equiv F_{obj,Best}\left(current - N_{min}N_{Ex,Cold}\right) - F_{obj,Best}\left(current\right) \le E_{B,Cold}; \quad \text{termination criterion}. \quad (34)$$

The small multiplicative factor, N_{min}, used in this work is 5. Again, this is a parameter on which the solution depends in a non-sensitive fashion. The value 5 has been chosen as optimal from the results of a sensitivity study. It was observed that (see below) whereas larger values of N_{min} do not improve significantly the minimum of the objective function reached they significantly increase the CPU requirements.

The parallel tempering algorithm does also lend itself to easily applying parameter constraints. This can be, for example, easily accomplished by transforming the constrained parameter (which again is assumed here to be positive) $q_i; 0 \le a_i \le q_i \le b_i$ to an unconstrained (still positive) parameter $\hat{q}_i; 0 \le \hat{q}_i \le \infty$ through the following bilinear transformation

$$\hat{q}_i = \begin{cases} q_i - a_i & b_i = \infty \\ \dfrac{q_i - a_i}{b_i - q_i} & b_i \ne \infty \end{cases}. \quad (35)$$

In conclusion, the parameters that need to be set a-priori to use the proposed algorithm are only 4, as indicated in Table 2 below. Furthermore, there are specific default values for all of those, as also indicated in the same table. Note that the suggested default values are selected conservatively in order to provide the most robust behavior even at the expense of computational efficiency as robustness is the key issue addressed in the present work.

Table 2. Default values of parallel tempering algorithm

Moreover, as also seen in indicative data supplied in the next section below, the algorithm performance depends only weakly on the values of those parameters. In contrast, all the other parameters on which the algorithm performance crucially depends upon are determined (as explained above) adaptively for each problem separately following a small number of simulation and MC runs. A graphical summary of the proposed methodology is described in Figure 5.

Figure 5. Schematic of the proposed algorithm where P_A and P_B are given by Eqs. (20) and (24), respectively.

Algebraic Examples

To further validate the algorithm and demonstrate its utility we offer in the present section a series of test cases from literature. We start with one- and two-dimensional algebraic equations examples that exhibit several local minima in their respective parameter space to demonstrate the robustness of the present approach in the presence of multiple local minima. Then we demonstrate several algebraic test cases with constraints. We then present three classic dynamic systems examples drawn from the chemical engineering literature. We conclude with a very nonlinear complex dynamic rheological example. When possible we compare our results from other methods from literature. For consistency, in all cases we use

for the numerical parameters the default values indicated in Table 2, although, on a case by case basis, we also offer sensitivity results to the numerical parameters, initial guess and experimental noise.

Unconstrained algebraic examples

The algebraic examples chosen are taken from the literature[49,66] so that they involve functions in one- and two-dimensional parameter space that are highly variable with many secondary local minima. Those correspond to the following one- and two-dimensional functions

$$f(x) = \tan\left(x + \frac{1}{4}\right) + \cos\left(10x^2 + \exp(\exp(x))\right), \tag{36}$$

$$f(x,y) = \left(\frac{x^2}{4} + e^{(\sin(50x))} + \sin(70\sin(x))\right) + \left(\frac{y^2}{4} + e^{(\sin(60y))} + \sin(80\sin(y))\right) - \cos(10x)\sin(10y) - \sin(10x)\cos(10y), \tag{37}$$

which are shown in graphical form in Figures 6 and 7, respectively. These examples are used to demonstrate that the parallel tempering algorithm can distinguish between many local minimum locations and out of those successfully select the global minimum in a robust and relatively computationally efficient fashion. It was also chosen to demonstrate that traditional methods in MATLAB will struggle with this algebraic problem because to get the correct answer via a traditional method, a good guess will be required. The *Chebfun*, and *Chebfun2* over-ride software[49,66] was used below to overcome these limitations. Typical results obtained with the two methods are compared in Tables 3 and 4 below.

Figure 6. Local vs. global min. (1-D): Objective function (y) vs. parameter values (x) for the 1D algebraic system described in Algebraic Examples section

Figure 7. Local vs. global min. (2-D): Objective function (F) vs. parameter values (x,y) for the 2D algebraic system described in Algebraic Examples section

Table 3. 1-D algebraic equation parallel tempering results

*Comparison made with Toshiba, 16G SDRAM, Intel® Core™ i7-4700; 2.40GHz

Table 4. 2-D algebraic equation parallel tempering results

*Comparison made with Toshiba, 16G SDRAM, Intel® Core™ i7-4700; 2.40GHz

Several things should be noted here. The first being that traditional MATLAB inbuilt functionality will fail here, unless given an initial guess close enough to the solution. The Chebfun, and Chebfun2 override software add-on can solve the minimization problem much quicker and computationally efficiently, as seen in Tables 3 and 4, but of course is limited to algebraic problems. It should also be noted that for the *simulannealbnd* command with MATLAB (simulated annealing algorithm), it was tested 100 times for each of the two cases (the 1-dimensional and the 2-dimensional case). The average and standard deviations of these runs are reported, as well as the "best" values (i.e., corresponding to the run that provided the closer to the actual solution answer). The simulated annealing algorithm can do a better job than the more traditional local minimization routines in MATLAB but still behaves rather erratically without always leading to the global minimum, as can be evidenced from the large mean errors and standard deviations. For the 2-dimensional case, the best out of 100 simulated annealing runs offered a solution that was not better than six significant figures, while the parallel tempering algorithm can capture at least 12 significant figures each time. The generality and additional complexity involved in the parallel tempering approach makes it substantially less efficient than the more streamlined Chebfun-based software for these simple algebraic problems. However, it is instructive to note that as the complexity of the problem increases (i.e., moving from the 1d to the 2d case) the CPU time ratio between the parallel tempering and the Chebfun-based approach drops from about 100 to about

15. This provides evidence that the computational efficiency of the proposed algorithm may be reasonable for relatively complex minimization problems,

Algebraic examples with constraints

The intent for the algebraic examples with constraints is to demonstrate that our parallel tempering implementation can easily incorporate them in two ways. In the first we explicitly limit the selection of limit parameter values over a specific allowed range using a bilinear mapping shown in Eq. (35). In the second approach, at each MC step we enforce the algebraic constraint inequalities iteratively and implicitly by applying a penalty to the calculation of the F_{obj}. Neither of the two methods is better than the other. For constraints of the form $g_i(\mathbf{x}) \le 0$, the objective function is modified as

$$F_{obj} = F_{obj} + a\sum_{i=1}^{n} \max\left[0, g_i(x)\right] , \tag{38}$$

where *n* is the number of constraints, and α is calculated as follows: For the first iteration,

$$\alpha = \left| \frac{F_{obj}}{\sum_{i=1}^{n} \max\left[0, g_i(x)\right]} \right| , \tag{39}$$

while for any subsequent one, and as long as there exists at least one inequality that is not satisfied, $\alpha = 2\alpha$ so the value of α, grows as the algorithm runs, and the penalty gets greater and greater until all constraints are satisfied and $\sum_{i=1}^{n} \max\left[0, g_i(x)\right]$ =0 (this procedure, as well as the calculation of the parameter α, is of course not needed if from the very beginning all constraints are duly satisfied). In this work both methods were used for each algebraic constraint problem attempted.

The results shown below include the results for optimization of several problems from literature including Constrained Sasena Function, the Constained Branin Function, and the New Branin Function

with parallel tempering, all with the Table 2 parallel tempering algorithm parameters. We compare our results of all three with recently published results for the same constrained algebraic functions by Boukouvala and Ierapetritou[18] . All results and comparisons are presented in Table 5. Note the ability of the parallel tempering algorithm to estimate error bars on the parameters. This is accomplished by running the code 15 times, and collecting the best results over all of the runs, followed by presenting the average, and using the standard deviation as an error bar estimate. With the array of best parameter values one can also calculate the covariance matrix. We note here that to accomplish this task one must randomize the initial guesses of each of the parameters to ensure a different and unbiased starting point (within the allowable parameter range of each parameter of course, and reasonable value which is chosen as same order of magnitude). The goal is to show that our user-friendly, yet robust algorithm in which we use parallel tempering is competitive with the best optimization algorithms in literature.

We start with brief description of the Constrained Sesena Function[18,67]:

$$\begin{gathered} \min f(x) = -(x_1 - 1)^2 - (x_2 - 0.5)^2 \\ \text{s.t.} \\ g_1(x) = (x_1 - 3)^2 + (x_2 + 2)^2 10^{(-x_2^7)} - 12 \le 0 \\ g_2(x) = 10x_1 + x_2 - 7 \le 0 \\ g_3(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.2 \le 0 \\ 0 \le x_i \le 1 \, i = 1,2 \end{gathered} \quad , \tag{40}$$

where, there is a function *f(x)* where $\boldsymbol{x} = [x_1, x_2]$, to be minimized, with the three constraints, g_1, g_2, and g_3. Additionally, the values of $\boldsymbol{x}$ are found in the range [0 1]. To incorporate these constraints we use Eqs. (38)-(39), as well as the bilinear mapping to ensure that our values of $\boldsymbol{x}$ stay in the required range. Additional information about this function can be found in work by Boukouvala and Ierapetritou[18] and Sasena *et al.*[67]. Results and comparison are shown in Table 5.

Our next functions to be optimized using parallel tempering are the constrained Branin and new Branin Functions shown here[18]:

$$\begin{gathered} \min f(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + h(1-e)\cos(x_1) + h \\ \text{s.t.} \\ x_1(1-x_2) - x_2 \le 0 \\ -5 \le x_1 \le 10 \\ 0 \le x_2 \le 15 \\ a = 1, b = 5.1/4\pi^2, c = 5/\pi, d = 6, h = 10, e = 1/8\pi \end{gathered} \quad , \tag{41}$$

and

$$\begin{gathered} \min f(x) = -(x_1 - 10)^2 - (x_2 - 15)^2 \\ \text{s.t.} \\ a(x_2 - bx_1^2 + cx_1 - d)^2 + h(1-ff)\cos(x_1) - 5 + h \le 0 \\ -5 \le x_1 \le 10 \\ 0 \le x_2 \le 15 \\ a = 1, b = 5.1/4\pi^2, c = 5/\pi, d = 6, h = 10, ff = 1/8\pi \end{gathered} \quad . \tag{42}$$

The functions shown above, Eq. (41) and Eq. (42), each have ranges for the parameters that must be obeyed and we again apply this in the code using a bilinear mapping[68].in order to enforce the required constraints on the parameter values. The additional constraints are also applied with Eqs. (38) -(39). Note that our code can obtain both sets of parameters that lead to the same global minimum for this problem.

Lastly we show for the sake of completion one more result that incorporates algebraic constraints. This is the classic benchmark problem minimization of weight of the spring, recently optimized by Kazemi *et al.*[69]. For more details about the system we refer the reader to this recent publication[69]. Suffice to say here that there are 3 parameters, 4 constraints, and each of the parameters is further constrained with a specific allowable range of values. To keep our parameters in the allowable parameter space we

once again use the bilinear mapping shown in Eq. (35), and for the algebraic constraints Eqs. (38) and (39). For this demonstration problem we show our average parameter values, along with our best parameter values, shown in Table 5.

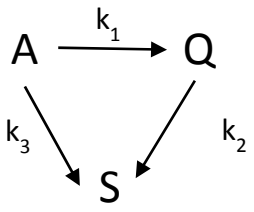
Table 5: Comparison of Parallel Tempering results to the results of Boukouvala and Ierapetritou (2014)[18] and Kazemi et al. (2011)[69] for the indicated functions

Dynamic model parameter estimation examples

The intent for the first three test cases is to demonstrate the entire process of the parallel tempering methodology, from running the Monte-Carlo dynamic model investigation runs to the meta-data analysis of the results. In addition, by choosing classical examples from the Chemical Engineering Literature we validate our approach against previously reported ones. We also use the opportunity of these well-defined and relatively simple cases to show the sensitivity of the results on several numerical parameters, thus validating the default values proposed in Table 2. We will also demonstrate the robustness of the algorithm by demonstrating insensitivity of the results to the initial guess of parameter values in appropriate cases.

Catalytic cracking of gas oil model problem: 1. Application

The first full demonstration problem is a classic example from chemical reaction engineering initially used by Tjoa and Biegler[70], and more recently by Kristensen[62,63]---see also *Nonlinear Programming Concepts, Algorithms and Applications to Chemical Processes* by Biegler[6] and work by Biegler and Damiano[71]. This example has been featured in several other publications in order to demonstrate parameter fitting algorithms[6,70,71]. The model reaction network refers to the (irreversible) catalytic cracking of gas oil from reactant (A) to gasoline (Q) and additional products (S):



(43)

The corresponding set of governing differential equations expressed in terms of the dimensionless concentrations y_A, y_Q and y_S are:

$$\frac{dy_A}{dt} = -(k_1 + k_3)y_A^2, \tag{44}$$

$$\frac{dy_Q}{dt} = k_1 y_A^2 - k_3 y_Q, \tag{45}$$

$$\frac{dy_S}{dt} = k_3 y_A + k_2 y_Q. \tag{46}$$

The data sets used for the parameter estimation, involving the time evolution of the dimensionless concentrations for species A and Q, are generated within the time interval $0 \le t \le 1$ using the dynamic model, Eqs. (44) and (45), subject to the initial conditions

$$\begin{aligned} y_A(0) &= 1 \\ y_Q(0) &= 0 \end{aligned} \tag{47}$$

For the parameter values $k_1 = 12, k_2 = 8$ and $k_3 = 2$ this is the test case mentioned in the publications referenced above. Therefore, this demonstration involves a system of **two** ordinary differential equations, *J=2*, zero algebraic equations, *K=0*, with **two** sets of accompanying data, *L=2*, and a total of **three** parameters to fit, *N=3*. Note that the third of the ODEs, Eq. (46) was not used in the simulation as the corresponding concentration was decoupled from the equations governing the others and it was also not present in the experimental data.

Sample results from the application of the proposed parallel tempering algorithm are shown in Table 6 in comparison with typical results obtained with other methods from the literature. It should also be noted that to run our algorithm it took 3.06 seconds, and 446 iterations. The "experimental" data used and the fit achieved with the method's obtained parameters are shown in Figure 8.

Table 6. Optimized parameter values for catalytic cracking problem (Tjoa & Biegler, 1991)[70]

Figure 8. Solution to the catalytic cracking problem for the parameter values

$$k_1 = 12, k_2 = 8 \text{ and } k_3 = 2$$

Catalytic cracking of gas oil model problem: 2. Sensitivity analysis

In addition, we took advantage of the simplicity of this problem to carry out some sensitivity analyses of the results on different numerical parameters. In Figures 9a,b we show the dependence of both the best value for the objective function, $F_{obj,best}$, and the CPU times on the number of parallel runs, N_{RUN}, used and on the ratio of $E_{B,Hot}/E_{B,Cold}$, respectively, all the other parameters remaining constant to their recommended default values as shown in Table 5. Note that the CPU times reported are for the core of the parallel tempering (i.e. without accounting for the MC initialization---which is typically much less) and for the code running on a single CPU---one can expect that time to decrease roughly in proportion of the CPU units running in parallel (up to the number of the parallel tracks used) if the code is implemented to run in parallel given the good parallelization characteristics as also documented in our previous work[56].

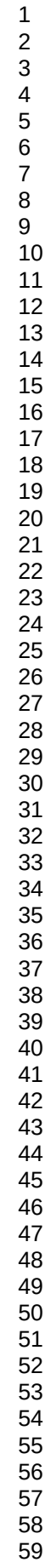


Figure 9. Sensitivity of $F_{obj,best}$ and CPU time on a) the number of $E_{B,i}$ levels, N_{RUN} and b) the $\frac{E_{B,Hot}}{E_{B,Cold}}$ ratio for the catalytic cracking problem.

Similarly, Figures 10a,b show the sensitivity of the same quantities ($F_{obj,best}$ and CPU time) to the other two remaining numerical parameters, $Error_{Max}$ and N_{min}.

Figure 10. Sensitivity of $F_{obj,best}$ and CPU time on a) the maximum Error, $Error_{Max}$, and b) the factor N_{min} for the catalytic cracking problem.

Figures 9 and 10 clearly show that while all four parameters impact the efficiency the dependence on their exact values is rather weak. In particular, the solution accuracy, as judged from the value of $F_{obj,best}$, is influenced very little from changes with respect to either N_{RUN} or N_{min} (suffice that a the numerical values chosen are greater than a critical minimum value) within the investigated regimes of values. Even in case of the other two parameters, to see an impairment of the solution accuracy one has to change the magnitude of $E_{B,Hot}/E_{B,Cold}$ below 1000 or of the $Error_{Max}$ above 0.01, i.e., an order of magnitude or more from their default values. Similarly, we also see that although the CPU times do show higher variations than the objective function there is still a significant region of parameter values within which the CPU time remains reasonably small within a factor of 3 from its minimum, 5, calculated under the constraint that the $F_{obj,best}$ is close to its own minimum value, 0.00312. Considering that this application is a rather simple one, the above observations justify the selection of the recommended default

values as those are well within the optimum operating region (minimum $F_{obj,best}$). The penalty factor of 3 more CPU time than the absolute minimum required for this problem is considered legitimate in order to allow for the operating point to be far away from the boundaries of the optimum region.

Finally, in order to further get insight as to how the algorithm operates, we provide in Figure 11 a graphic of the evolution of F_{obj} within the 15 parallel MC runs as a function of the iteration (MC step) number as obtained from a sample run. From this figure one can appreciate the rapid convergence of the parallel tempering as well as the power of exploring widely the allowed parameter space, as indicated by the peaks of high values for the objective function that continue to appear throughout the run.

Figure 11. The dependence of F_{obj} within the 15 parallel MC runs on the iteration (MC step) number for the catalytic cracking problem. The legend values on the right show the normalized Boltzmann Energy levels used in the parallel tempering.

Reversible chemical reactions network

The second dynamic system is based on a reversible chemical reactions network

$$\mathrm{A} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{B} \underset{k_4}{\overset{k_3}{\rightleftharpoons}} \mathrm{C}, \tag{48}$$

which has been previously modeled[70,71] and it was also used to benchmark parameter fitting algorithms[6,11,70]. To generate the appropriate value of h_{test} and ε_{test} in accordance with Eqs. (33) and (32) several test runs were made of the ODE solver. This was also performed to explore the sensitivity of the both the ODE integration error and the Composite Simpson's Rule integration in the F_{OBJ} formulation. For this test problem it was thus determined that the Simpson's Rule integration was more sensitive and this was used as a basis to pick the h_{crit}, or the largest *h* that the algorithm could tolerate while still

maintaining our $Error_{Max}$ criteria. In addition to this we were able to extract from this numerical experiment the approximate value for h_{test} and ε_{test}. This system represents liquid reactant A, reacting to form liquid B, and finally reacting to form product C in a batch reactor. The corresponding set of differential equations governing the time evolution of the dimensionless concentrations of species A, B, and C is :

$$\frac{dy_A}{dt} = -k_1 y_A + k_2 y_B, \tag{49}$$

$$\frac{dy_B}{dt} = k_1 y_A - (k_2 + k_3) y_B + k_4 y_C, \tag{50}$$

$$\frac{dy_C}{dt} = k_3 y_B + k_4 y_C. \tag{51}$$

The three data sets used for the parameter estimation, one for each one of the three species concentrations, are generated within the time interval $0 \le t \le 1$ using the dynamic model, Eqs. (49)-(51), subject to the initial conditions:

$$\begin{aligned} y_A(0) &= 1 \\ y_B(0) &= 0. \\ y_C(0) &= 0 \end{aligned} \tag{52}$$

The parameter values $k_1 = 4, k_2 = 2, k_3 = 40$ and $k_4 = 20$ correspond to the test case cited in the publications referenced above. Therefore, this demonstration involves a system of **three** ordinary differential equations, *J=3*, zero algebraic equations, *K=0*, with **three** sets of dynamic data, *L=3*, and a total of **four** parameters, *N=4*, to fit. Note that the third of the ODEs, Eq. (51), can be replaced by an algebraic equation taking advantage of the overall mass balance, which for the case considered here can be simply expressed as

$$y_A + y_B + y_C = 1. \tag{53}$$

Alternatively, (as it was done here), the constraint condition represented by Eq. (53) can be used to check the accuracy of the ODE integrators. Corresponding to the $Error_{Max} = 0.001$ value used for the numerical integrations, for the problem at hand, this condition was found to be satisfied to within a maximum of 8.23 10^{-5} error. This value was calculated using the Composite Simpson's Rule, and verified by taking the infinity norm in MATLAB of the difference between the ODE solution for y_C over the time period of integration, and the y_C concentrations calculated using mass balance. Also, consistent to the analysis offered in the description of the objective function in the Problem Formulation section, in this case as the infinite time concentration values do not necessarily go to zero, their steady state values needed to be subtracted from the transient results and used independently as elements of the objective function. These equilibrium values were solved for using linear algebra, with MATLAB a priori so as to use the values in the F_{OBJ}. The $Ax = b$, standard format was utilized, using the known solution vector $k_1, k_2, k_3,$ and k_4, as well as the fact that at equilibrium Eq. (49),(50), and (51) can be set to zero. The integrations within the objective function, F_{OBJ} were calculated using the fourth order composite Simpson's rule so it is of the same accuracy as the ODE integration method used here. For this example we used an explicit, linear-multi-step integration scheme, the 4th order Adams-Bashforth-Moulton method[65], with Runge-Kutta Dormand-Prince method to start (5th order part)[57,64]. A typical solution, shown graphically in Figure 12, was found in 91.7 seconds, and it compares favorably to the solutions from literature[70] as shown in Table 7.

Figure 12. Time evolution for the three species concentration as obtained from the Solution for the reversible chemical reactions problem for the parameter values $k_1 = 4, k_2 = 2, k_3 = 40$ and $k_4 = 20$.

Table 7. Optimized parameter values for the reversible chemical reactions problem

First order irreversible chemical reaction network: 1. Application

The next demonstration problem has also been previously modeled and is used to demonstrate parameter fitting algorithms, such as featured in the work of Biegler[6], Tjoa and Biegler[70], Biegler and Damiano[71], and Cizniar *et al.*[72]. The third dynamic system is based on a two-step, irreversible isothermal reactions network that model the irreversible transformation of a liquid reactant (*A)* initially to a liquid product (*B)* and then finally to another liquid product (*C)*:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C}. \tag{54}$$

This reaction is carried out in a batch reactor, and is modeled by the following equations expressed in terms of the dimensionless concentrations y_A, y_B and y_C [6,70-72]:

$$\frac{dy_A}{dt} = -k_1 y_A, \tag{55}$$

$$\frac{dy_B}{dt} = k_1 y_A - k_2 y_B, \tag{56}$$

$$\frac{dy_C}{dt} = k_2 y_B. \tag{57}$$

The three data sets used for the parameter estimation, involving the time evolution of the dimensionless concentrations for all species, are generated within the time interval $0 \le t \le 1$ using the dynamic model, Eqs. (55)-(57) and subject to the initial conditions:

$$y_A(0) = 1$$
$$y_B(0) = 0 \quad . \tag{58}$$
$$y_C(0) = 0$$

The parameter values $k_1 = 5$ and $k_2 = 1$ correspond to the test case mentioned in the cited referenced above. It should be noted that there are **three** sets of data to fit, *L=3*, **three** differential equations, *J=3*, zero algebraic equations, *K=0*, and **two** parameters to optimize, *N=2*. For this solution a 5th order explicit Runge-Kutta technique was used to start the problem with transition to the Adams – Bashforth – Moulton 4th order linear multistep method[57,64,65]. Several test runs were made of the ODE solver to generate the appropriate value of h_{test} and ε_{test} in accordance with Eq. (33). This was also performed to explore the sensitivity of the both the ODE integration error and the Composite Simpson's Rule integration in the F_{OBJ} formulation. For this test problem it was thus determined that the Simpson's Rule integration was more sensitive and this was used as a basis to pick the h_{crit} , or the largest h that the algorithm could tolerate while still maintaining our $Error_{Max}$ criteria. Note that, as was the case in the previous problems, the third of the ODEs, Eq.(57), can be replaced by an algebraic equation taking advantage of the overall mass balance, which for the case considered here can be simply expressed by Eq. (53), as before. Alternatively, (as it was done here), the constraint condition represented by Eq. (53) can be used to check the accuracy of the ODE integrators. Even with when the $Error_{Max} = 0.001$value was used for the numerical integrations, for the problem at hand, this condition was found to be satisfied to within machine accuracy using the Composite Simpson's Rule, and verified with the infinity norm in MATLAB. The integrations within the objective function, F_{OBJ} were calculated using the fourth order composite Simpson's rule so it is of the same accuracy as the ODE integration method used here. A typical solution was found in 5.8 seconds, and it compares favorably to the solution from literature [70] as shown in Table 8.

Table 8. Irreversible chemical reaction parameter comparison

First order irreversible chemical reaction network: 2. Sensitivity to initial guess

Using this dynamic model we have investigated the resilience of the proposed method to poor initial guesses. The results of this investigation are represented in Figure 13, which shows the convergence behavior for the objective function as obtained from four different initial guesses, including the initial guess from Tjoa and Biegler[70]. In addition, in each case we have used three independent stochastic runs to show the influence of the stochastic nature of the method.

Figure 13. Plot of iterations vs. error of different initial guess values for irreversible chemical reaction

Table 9 presents the final parameter values of k_1 and k_2 obtained after each one of the parallel tempering algorithm runs, with mean and standard deviation values. As seen from the results both Figure 13 and Table 9, the method is quite resilient, always leading to the same solution with small fluctuations in both the accuracy and the CPU time required.

Table 9. Best parameter values after three random trials with different initial guesses of $k_{1,0}$ and $k_{2,0}$

Fitting Complex Dynamic Nonlinear Rheological Example

Fitting LAOS data: 1. Application

A crucial test for our dynamic parameter estimation approach is represented by the final test problem. This involves the dynamic mechanical behavior of a highly nonlinear thixotropic colloidal system. For the dynamic parameter evaluation study a structural, albeit phenomenological, rheological model is considered, the de Souza Mendes model[50]. The model equations are flexible enough for predicting both steady state and large amplitude oscillatory shear conditions. However the fitting of those

parameters that only affect the dynamic behavior, as probed here from the model predictions to Large Amplitude Oscillatory Shear (LAOS). This nonlinear shear oscillation probes critically the gradual transition observed in these highly concentrated colloidal systems, from elastic to viscous, as the elastic strain saturates and the plastic strain rate increases. In the phenomenological model used here to describe such a behavior the material properties, such the elastic modulus, G, and the viscosity, η, are all functions of the structure parameter. As this model allows both the elastic modulus and the viscosity to vary with the structure parameter, one can accurately predict cases of more structure as more elastic and less viscous, and vice versa. This model also involves comparison to the equilibrium value of the structure parameter, based on current stress or shear rate, to calculate the new value. It uses the difference between current conditions and equilibrium conditions as the driving force to calculate the next value of structure parameter, modulus, viscosity, etc.[50-54]

Briefly, the mathematical description of the de Souza Mendes model has as follows. The heart of the model is a viscoelastic differential equation that relates implicitly the shear stress, τ, to the shear rate, $\dot{\gamma}$ and their time derivatives (denoted by double dots),

$$\dot{\gamma} + \theta_2 \ddot{\gamma} = \frac{\theta_2}{\eta_\infty}\left(\frac{\tau}{\theta_1} + \dot{\tau}\right). \tag{59}$$

The key difference from any other (simple) viscoelastic model is that the material parameters appearing in this equation are not constant but vary in time following the development of the structure within the material. The material structure is modeled by a scalar variable, λ, for which a separate relaxation equation is proposed:

$$\frac{d\lambda}{dt} = \frac{1}{t_{eq}}\left[\left(\frac{1}{\lambda} - \frac{1}{\lambda_0}\right)^a - \left(\frac{\lambda}{\lambda_{eq}}\right)^b \left(\frac{1}{\lambda_{eq}} - \frac{1}{\lambda_0}\right)^a\right]. \tag{60}$$

The parameters entering Eqs. (59) and (60) satisfy the following algebraic equations

$$\theta_1 = \left(1 - \frac{\eta_\infty}{\eta_v(\lambda)}\right)\frac{\eta_v(\lambda)}{G_v(\lambda)}, \tag{61}$$

$$\theta_2 = \left(1 - \frac{\eta_\infty}{\eta_v(\lambda)}\right)\frac{\eta_\infty}{G_v(\lambda)}, \tag{62}$$

$$\eta_{eq}(\dot{\gamma}_{eq}) = \left[1 - \exp\left(-\frac{\eta_0 \dot{\gamma}_{eq}}{\tau_y}\right)\right]\left\{\frac{\tau_y - \tau_{yd}}{\dot{\gamma}_{eq}} e^{-\dot{\gamma}_{eq}/\dot{\gamma}_{yd}} + \frac{\tau_{yd}}{\dot{\gamma}_{eq}} + K\dot{\gamma}_{eq}^{\,n-1}\right\} + \eta_\infty, \tag{63}$$

$$G_v(\lambda) = G_0 e^{m\left(\frac{1}{\lambda} - \frac{1}{\lambda_0}\right)}, \tag{64}$$

$$\eta_v(\lambda) = \eta_\infty e^{\lambda}, \tag{65}$$

$$\lambda_{eq} = \ln\left(\frac{\eta_{eq}}{\eta_\infty}\right), \tag{66}$$

$$\lambda_0 = \ln\left(\frac{\eta_0}{\eta_\infty}\right), \tag{67}$$

and

$$\sigma = \eta_{eq}(\dot{\gamma}_{eq})\dot{\gamma}_{eq}. \tag{68}$$

The above set of equations contains a total of ten parameters, of which seven are to be separately fit based on the steady shear and small amplitude oscillatory shear (SAOS) material behavior (not considered in the present study). A list of values for the parameters used here are presented in Table 10[50].

Table 10. Complex rheological model parameters fit to steady state and SAOS data

The remaining parameters that are to be fit against model LAOS data are the power law index, *n*, the power law viscosity prefactor *K* and the relaxation time t_{eq}. To test the present code in evaluating those parameters, three sets of LAOS data have been prepared for a fixed set of model parameters and three sets of stress amplitudes, $\tau_a = 5, 10, 20$ Pa that control a time=periodic material behavior assuming a sinusoidal stress

$$\tau = \tau_a \sin(\omega t) . \tag{69}$$

The dynamic system consists of **two** ordinary differential equations, Eqs. (59) and(60), *J=2*, and **six** algebraic equations, Eqs. (63)-(68), *K*=6. Note that some of those equations are highly nonlinear requiring an iterative approach for their solution. We used **three** sets of dynamic periodic data, *L=3*, generated as mentioned above, for evaluating **three** of the parameters, *N=3*. A *fully implicit* 5th order Runge-Kutta method[64] (Radau 1a), was used due to the stiffness of the differential algebraic system. Typical results are shown in Table 11 and Figure 14. The fully implicit 5th order Runge-Kutta (Radau 1a) DAE/ ODE solver has no way to check a n^{th} order solution vs. an (n+1)th order solution during the ODE integration, at each time step. Therefore, error was estimated by integrating the trapezoid rule between the model prediction and the solution for each of the sets of dynamic data and comparing values using successive different values of *h*. The composite trapezoid rule was used in this problem for the calculation of F_{OBJ} due to the periodic nature of the dynamic data.

Table 11. Complex rheological model parameters fit to transient (LAOS) data

Figure 14. Complex dynamic rheological parameter oscillatory behavior shown in the

form of Lissajous-Bowditch plots for two different stress amplitudes as shown in the figures: a) $\tau_a = 5\,\text{Pa}$ **; b)** $\tau_a = 10\,\text{Pa}$ **,**

Fitting LAOS data: 2. Sensitivity analysis

As far as the algorithm is concerned, we have explored the robustness of this model in great depth with the logic that success with the most challenging and non-linear system would provide clear evidence of ability to successfully be independent of bad guesses. To conduct this test thoroughly, a series of increasing perturbations to the initial guesses were made to demonstrate the correct parameters +/-<1% could still be found by the algorithm. The perturbations of the bad guess were conducted by using increasing powers of ten. The results are shown in Figure 15. One can see that as the magnitude of the perturbation is increased the solution will still converge to the correct answer. Interestingly, and this is an additional evidence towards the robustness of the method, there does not appear to be a systematic correlation of the initial guess to the use of more iterations, and therefore more CPU time.

Figure 15. Sensitivity of the $F_{obj,best}$ and CPU time on the magnitude of perturbation applied to the initial guess

In addition for this parameter fitting demonstration it should be noted to the user that when performing the Monte Carlo runs to find the proper *h* values, one should note that for certain stiff DAE/ ODE systems there may exist a critical value of *h* whereby all values greater than h_{crit} will give poor solutions, none at all, or register as "NaN". For this particular DAE system this was the case and a h_{crit} value of 0.1 was discovered. During the running of the parameter fitting there was a simple loop that overwrites this value when h values are larger than 0.1. In addition it should be mentioned that the initial runs to evaluate adaptively the best operating parameters, like the time step *h* in the numerical integrations and the Monte Carlo runs needed to determine Boltzmann Energy levels, as not too time

consuming but requiring the user interaction, were performed best in the interacting environment of MATLAB. In contrast, the more time-consuming calculations involving the parameter fitting through the parallel tempering method are more efficiently performed using a compiled language like Fortran90. This was due to the fact that Fortran90 is approximately 25 times faster than MATLAB.

Additionally an investigation into how the algorithm can fit parameters successfully to data sets with experimental noise has been conducted with the results shown in Figure 16. The noise is superimposed onto the experimental data, which in this case means that the empirical functions used to recreate the data were modified with a varying degree of imposed noise. The magnitude of the noise is varied in sequence by increasing orders of ten, while the number of iterations was purposefully kept the same between all cases. The results shown in Figure 16 refer to F^*_{OBJ} , evaluated in Eq. (70), that represents the average, normalized deviation from the actual values of the parameters, where the deviation is caused by the noise in the 'data'. This demonstrates that the parallel tempering algorithm works reasonably well even when the data to be fit has noise provided, of course, the noise levels are not excessive.

Figure 16. Sensitivity of F_{obj} to the magnitude of the noise applied to empirical data

$$F^*_{OBJ} = \sqrt{\sum_{i=1}^{N}\left(\frac{\Theta_i - \Theta_i^{act}}{\Theta_i^{act}}\right)} \Big/ N \ . \qquad (70)$$

Finally, in Figure 17 F_{OBJ}, and the value calculated for the quantity used in developing the stopping criterion, $\left|F_{OBJ,Best,(new-N_{Min})} - F_{OBJ,Best,new}\right|$, is plotted versus CPU time. This figure shows that the stoppage criterion $\left|\Delta F_{OBJ}\right| \le N_{min} E_{B,Cold}$ for any small value of N_{min} (5 was used here) works well in capturing the best value for F_{OBJ}. Continuing the calculations beyond the conditions satisfying the stopping criterion only results in increasing the CPU time without improving the results. The results shown justify the choice of the stoppage criterion for our algorithm.

Figure 17. Evolution of F_{OBJ} and stoppage criteria with CPU time during the parameter fit of the complex dynamic rheological data.

Conclusions

In this manuscript we demonstrate an approach to parameter fitting in dynamic systems that is based on a parallel tempering algorithm and that allows for an optimization that goes beyond local optimization and has the potential to avoid local trapping and reach a global optimum provided sufficient number of parallel runs are use. The objective function that is minimized is constructed using an L2 norm average of the differences between the model predictions and dynamic as we all as static data. Furthermore, the approach is fully adaptive with few (only 4) adjustable parameters, for which default values are suggested. The solution is shown to be independent of those parameter values as well as on the initial guess for a number of test cases. All the other necessary parameters are determined adaptively using a few Monte-Carlo runs at the outset (such as, for example, for the N_{Ex} data). The algorithm's capabilities have been demonstrated on two algebraic equations with several known minima, as well as on several dynamic systems from chemical engineering and one from complex rheology.

During the course of testing the algorithm on many cases and using several dynamic systems from literature there were several critical lessons learned. The first is that although a good initial guess may give faster results, it is not necessary for convergence and even the CPU time penalty, starting from a poor guess, is fairly small. What is important is the robustness of the approach, which was always found to provide a good parameter estimate, and the lack of any need for fine-tuning. Second, as a side benefit, one can use the proposed approach to evaluate the relative sensitivity of the objective function to each model parameter as well as for possible correlations between the different parameters, as demonstrated by

Armstrong *et al.*[73]. Regarding the overall computational load, which of course increases substantially with the problem complexity, we note that the underlining algorithm is really only needed to provide an approximate answer for the global minimum by identifying its locality. Once this is achieved, any direct local optimization method can be used to gain as much accuracy as desired. As with all stochastic methods, and as it is applied to engineering problems not necessarily well defined, no a-priori guarantees can ever exist for the performance of the method to find quickly the global optimum. Finally, we emphasize that the parallel tempering process is inherently parallelizable, which may be necessary for much more complex problem than those explored here.

Acknowledgments

This work was performed with support from the Department of Chemistry and Life Sciences, United States Military Academy, United States Army and National Science Foundation Award CBET 1235863. The views expressed herein are those of the authors and do not reflect the position of the United States Military Academy, the Department of the Army, or the Department of Defense. The authors will also like to acknowledge the help of two anonymous reviewers of a previous version of the present manuscript in helping us to considerably improve its presentation.

Notation

A,B,C,Q,S	reactant; product
A_i	relative parameter sensitivity
a_n, b_n	lower, upper bounds of parameter values
a_l	Runge Kutta time steps
b_{lk}	Runge Kutta weights
c_i	Runge Kutta weights on $i+1$ soln.; empirical eqn. coefficients
d_i	ESDIRK34 error weights

E_B Boltzmann energy vector

$E_{B,Cold}$ Boltzmann Energy of the cold (lowest) level

$E_{B,Hot}$ Boltzmann energy of the hot (highest) level

E_{Bi} Boltzmann Energy of *ith* level

F_{obj} objective function

G elastic modulus

h time step

i iteration

J number of first order ODES

K number of algebraic equations

k_i; p_i reaction coefficients

k_B Boltzmann constant

L number of cont. sets of dynamic data

M number of sets of static data

N number of internal model parameters

N_{Ex} number of steps between exchange trials

N_m number of discrete data of m-th static set

N_{run} number of parallel MC sequences

N_{steps} number of steps during the time int.

c_i empirical equation coefficient

p_i parameter value at i^{th} iteration

P number of external control parameters

P_{accept} probability of acceptance

q vector of N internal model parameters

R_k autocorrelation function

w_i weights used in objective function

x vector of J initial conditions

y vector of $J + K$ dynamic variables

y_A, y_B, y_C reactant/ product conc.

y_Q, y_S reactant/ product conc.

z vector of P external control parameters

α_i,β_i lower, upper bounds of i-th time span

γ ESDIRK34 weight

$\dot{\gamma}$ shear rate

$\ddot{\gamma}$ derivative of shear rate

ε error

η viscosity

θ time scale; vector of parameters

λ structure parameter

λ_I char. relaxation time

μ_i average of parameter at *ith* E_B level

σ_N standard deviation

τ_n empirical equation time constants

τ vector of measured quant., shear stress

Literature Cited

1. Marquardt DW. An algorithm for Least-squares estimation of nonlinear parameters. *J SIAM*. 1963; 11: 431-441.
2. Press WS, Teukolsky ST, Vetterling WT, Flannery BP. *Numerical Recipes in Fortran*. Second Edition. Cambridge: Cambridge University Press, 1992.
3. Yuceer M, Atasoy I, Berber R. A software for parameter estimation in dynamic models. *Braz J Chem Eng*. 2008; 25: 813-821.
4. Kirkpatrick S, Gelatt CD Jr, Vecchi MP. Optimization by simulated annealing. *Science*. 1983; 220: 671-680.

5. Corana A, Marchesi M, Martini C, Ridella S. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans Math Soft*. 1987; 13: 262-280.
6. Biegler L. *Nonlinear Programming Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia: SIAM, 2010.
7. Spall JC. *Introduction to Stochastic Search Optimization*. Hoboken: John Wiley & Sons, 2003.
8. Seber GA, Wild CJ. *Nonlinear Regression*. New York: John Wiley & Sons, 1989.
9. MATLAB Help/ Documentation, version 2013. Version: 8.1.0.604 (R2013a), License: Student. MathWorks, Natick, Massachusetts, U.S.A.
10. Wolfram Research, Mathematica (2008). Wolfram Mathematica Tutorial Collection Mathematics and Algorithms. Worlfam Research Inc., 1-365. http://www. reference.wolfram.com
11. Floudas CA. *Deterministic Global Optimization: Theory, Methods and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.
12. Horst R, Tuy H. *Global Optimization Deterministic Approaches*. Berlin: Springer 1996.
13. Floudas C. Research challenges, opportunities and synergism in systems engineering and computational biology. *AIChE J*. 2005; 51: 1872-1884.
14. Floudas CA, Gounaris CE. A review of recent advances in global optimization. *J Glob Optim*. 2009; 45: 3-38.
15. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive blackbox functions. *J Glob Optim*. 1998; 13: 455–492.
16. Jones DR. A taxonomy of global optimization methods based on response surfaces. *J Glob Optim*. 2001; 21: 345–383.
17. Kleijnen JPC. Kriging metamodeling in simulation: A review. *Eur J Operat Res*. 2009; 192: 707-716.
18. Boukouvala F, Ierapetritou MG. Derivative –free optimization for expensive constrained problems using a novel expected improvement objective fucntion. *AIChE J*. 2014; 60: 2462-2474.
19. Jacobson S, Patricksson M, Rudholm J, Wojciechowski A. A method for simulation based optimization using radial basis functions. *Optim Eng*. 2010; 11: 501-532.
20. Balsa-Canto E, Banga JR. AMIGO, a toolbox for advanced model identification in systems biology using global optimization. *Bioinformatics*. 2011; 27: 2311-2313.

21. Villaverde AF, Henriques D, Smallbone K, Bongard S, Schmid J, Cicin-Sain D, Crombach A, Saez-Rodriguez J, Mauch K, Balsa-Canto E, Mendres P, Jaeger J, Banga JR. BioPreDyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC Systems Biology*. 2015; 9:8 (15 pp).
22. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E. Equation of state calculations by fast computing machines. *J Chem Phys*. 1953; 21: 1087-1090.
23. Onbaşoğlu E, Özdamar L. Parallel simulated annealing algorithms in global optimization. *J Glob Optim*. 2001; 19: 27-51.
24. Xavier-de-Souza S, Suykens JAK, Vanderwalle J, Bollé D. Coupled simulated annealing. *IEEE Trans Systems Man Cybernetics---Part B : Cybernetics*. 2010; 40: 320-335.
25. Meybodi MK, Shokrollahi A, Safari H, Lee M, Bahadori A. A computational intelligence scheme for prediction of interfacial tension between pure hydrocarbons and water. *Chem Eng Res Des*. 2015; 95: 79-92.
26. Sahimi M, Hamzehpour H. Efficient computational strategies for solving global optimization problems. *Comp Sci & Eng*. 2010; 12(4): 74-82.
27. Floudas CA, Esposito WR. Optimization for the Parameter Estimation of Differential Algebraic Systems. *Ind Eng Chem Res.* 2000; 39: 1291 - 1310.
28. Clerc M, Kennedy J. The particle swarm---explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans Evol Comp.* 2002; 6(1): 58-73.
29. Calvo F. Non-genetic global optimization methods in molecular science: An overview. *Comp Mater Sci*. 2009; 45: 8-15.
30. Li H, Qin SJ, Tsotsis TT, Sahimi M. Computer simulation of gas generation and transport in landfills: VI—Dynamic updating of the model using the ensemble Kalman filter. *Chem Eng Sci*. 2012; 74: 69-78.
31. Angira R. A Comparative Study of Differential Evolution Algorithms for Estimation of KineticParameters. *Adv Mod Optim*. 2012; 14: 135 – 145.
32. Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. *Simulation*. 2001; 76: 60-68.
33. Mahdavi M, Fesanghary M, Damingir E. An improved harmony search algorithm for solving optimization problems. *Appl Math Comp*. 2007; 188: 1567-1579.

34. Swendsen RH, Wang JS. Replica Monte Carlo simulation of spin-glasses. *Phys Rev Lett*. 1986; 57: 2607 – 2609.

35. Bittner E, Nussbaumer A, Janke W. Make life simple: Unleash the full power of the parallel tempering algorithm. *Phys. Rev. Lett.* 2008; 101: 130603 (4 pp).

36. Sugita Y, Okamoto Y. Replica-exchange molecular dynamics method for protein folding. *Chem Phys Lett.* 1999; 314: 141-151.

37. Schug A, Herges T, Verma A, Wenzel W. Investigation of the parallel tempering method for protein folding. *Phys. Cond. Matter, special issue: Structure and Function of Biomolecules*. 2005; 17: 1641-1650.

38. Earl DJ, Deem MW. Parallel Tempering: Theory, applications and new perspectives. *Phys Chem.* 2005; 7: 3910-3916.

39. Gront D, Kolinski A. Efficient scheme for optimization of parallel tempering Monte Carlo method. *J Phys: Condensed Matter.* 2007; 19: 036225 (9pp).

40. Theodorou DN. Progress and outlook in Monte Carlo simulations. *Ind & Eng Chem Res*. 2010; 49(7): 3047-3058.

41. Guidetti M, Rolando V, Tripiccione R. Efficient assignment of the temperature set for parallel tempering. *J Comp Phys.* 2012; 231: 1524-1532.

42. Habeck M, Nilges M, Rieping W. Replica-exchange Monte Carlo scheme for Bayesian data analysis. *Phys Rev Lett*. 2005; 94(1): 018105 (4 pp).

43. Wang C, Hyman JD, Percus A, Caflisch R. Parallel tempering for the traveling salesman problem. *Int J Mod Phys C*. 2009; 20(4): 539-556.

44. Ochoa S, Wozny G, Repke J-U. A new algorithm for global optimization: Molecular-inspired parallel tempering. *Comp & Chem Eng*. 2010; 34: 2072-2084.

45. Sambridge M. A parallel tempering algorithm for probabilistic sampling and multilodal optimization. *Geophys J Int.* 2014; 196: 357-374.

46. Nallasivam U, Shah VH, Shenvi AA. (2012). Global Optimization of Multicomponent Distillation Configuration: 1. Need for a Reliable Global Optimization Algorithm. *AIChE J*. 2012; 59: 971- 981.

47. Amar, JG. The Monte Carlo Method in Science and Engineering. *Comp Sci & Eng.* 2006; 8(2): 9 - 19.

48. Renotte CA, Wouwer AV. Stochastic Approximation Techniques Applied to Parameter Estimation

in a Biological Model. *Institution of Electrical Engineering, Computer Science and American Institute of Physics*. 2003; 261 - 263.

49. Battles Z, Trefethen L. An Extension of MATLAB to Continuous Functions and Operators. *SIAM J Sci Comp.* 2005; 25: 1743 - 1770.
50. De Souza Mendes P, Thompson R. A unified approach to model elasto-viscoplastic thixotropic yield-stress materials and apparent-yield-stress fluids. *Rheologica Acta.* 2013; 52: 673-694.
51. Rogers S, Lettinga P. A sequence of physical processes determined and quantified on LAOS: An instantaneous local 2D/3D approach. *J Rheol*. 2012; 56: 1129-1151.
52. Radhakrishnan R, Underhill P. (2014) Oscillatory Shear Rheology of Dilute Solutions of Flexible Polymers Interacting with Oppositely Charged Particles. *AIChE J.* 2014; 60: 1365 -1370.
53. Deshpande AP, Krishnan M, Sunil-Kumar PB. Oscillatory shear rheology for probing nonlinear viscoelasticity of complex fluids: Large amplitude oscillatory shear (LAOS) in Rheology of Complex Fluids. New York: Springer-Verlag, 2010.
54. Macosko CW. *Rheology Principles, Measurements, and Applications*. New York, NY: Wiley VCH, 1994.
55. Mandel J. *The statistical analysis of experimental data*. New York, NY: Dover Publications, 1984.
56. Mukherjee J, Beris AN. Qualitative Lattice Simulations of the Dense Amorphous Phase in Semicrystalline Polymers: Size and Energy. arXiv:0805.0382, 2004.
57. Dormand JR, Prince PJ. A Family of Embedded Runge Kutta Formulae. *J Comp Appl Math*. 1980; 6: 19-26.
58. Feagin TA. Tenth-Order Runge Kutta Method with Error Estimate. Proceedings of the IAENG Conf. on Scientific Computing. 2006.
59. Feagin TA. Higher Order Explicit Runge-Kutta Methods Using m-Symmetry. *Neural. Parallel & Scientific Computations*. 2012; 20: 437-458.
60. Fehlberg, E. Classical Fifth, Sixth and Seventh Order Runge Kutta formulas with Step Size Control. NASA. 1968; 1-82.
61. Gear B. The Simultaneous Numerical Solution of Differential Algebraic Equations. SLAC-PUB-723. *IEEE Trans. on Circuit Theory*. 1970; 1-21.

62. Kristensen MR. *Parameter Estimation in Nonlinear Dynamical Systems*. Masters, Technical University of Denmark. [Thesis] 2004.

63. Kristensen MR, Jorgensen JB, Thomsen PG, Michelsen ML, Jorgensen SB. (2005) Sensitivity Analysis in Index-1 Differential Algebraic Equations by ESDIRK Methods. *IFAC*. 2005; 6: 1-9.

64. Butcher JC. *Numerical Methods for Differential Equations*. Great Britain: John Wiley and Son, 2008.

65. Mathews JH, Fink KD. *Numerical Methods Using MATLAB*. Upper Sadle River, NJ: Pearson Prentice Hall, 2004.

66. Townsend A, Trefethen L. (2013). An Extension of Chebfun to Two Dimensions. *SIAM J Sci Comp*. 2013; 35: 95–C518.

67. Sasena MJ, Papalambros P, Goovaerts P. Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. *Eng. Opt.* 2001; 34: 263-278.

68. Bronshtein IN and Semendyayev KA. *Handbook of Mathematics.* Third Edition. New York: Van Nostrand Reinhold, 1985.

69. Kazemi M, Wong GC, Rahnamayan S, Gupta K. Metamodel-based optimization for problems with expensive objective and constraint functions. *J Mech Design.* 2011; 133: 1-7.

70. Tjoa I, Biegler L. Simultaneous Solution and Optimization Strategies for Parameter Estimation of Differential-Algebraic Equation Systems. *Ind Eng Chem Res*. 1991; 30: 376-385.

71. Biegler L, Damiano JJ. Nonlinear Parameter Estimation: a Case Study. *AIChE Journal.* 1986; 32: 2-54.

72. Cizniar M, Podmajersky M, Hirmajer T, Fikar M, Latifi AM. Global optimization for parameter estimation of differential-algebraic systems. *Chemical Papers.* 2009; 63:, 274 – 283.

73. Armstrong MJ, Beris AN, Rogers, SA, Wagner NJ. Dynamic Shear Rheology of a Thixotropic Suspension: Comparison of Improved Structure-Based Models with Large Amplitude Oscillatory Shear Experiments. *J. Rheology.* 2016; 60(3): 433-450.

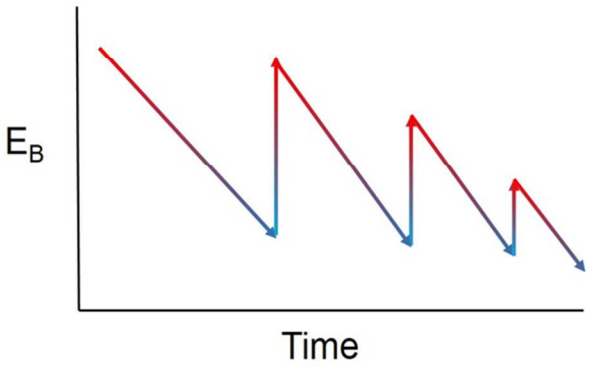


Figure 1. Schematic of a typical simulated annealing cooling schedule; here "Time" is proportional to MC steps.

81x60mm (300 x 300 DPI)

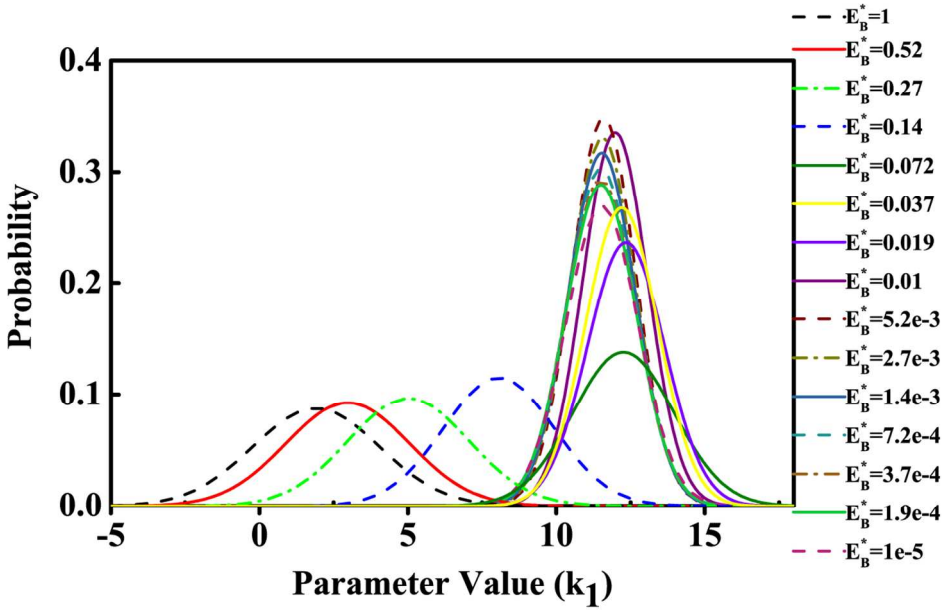


Figure 2. Plot of pdfs for the state quantity p=k1 reaction rate constant based on normal probability density distribution fits using statistics data ( ) drawn from Catalytic Cracking of Gas example 6 as shown in Table 1. The Boltzmann energy levels corresponding to the pdf curves shown are indicated on the right of the figure.

368x228mm (100 x 100 DPI)

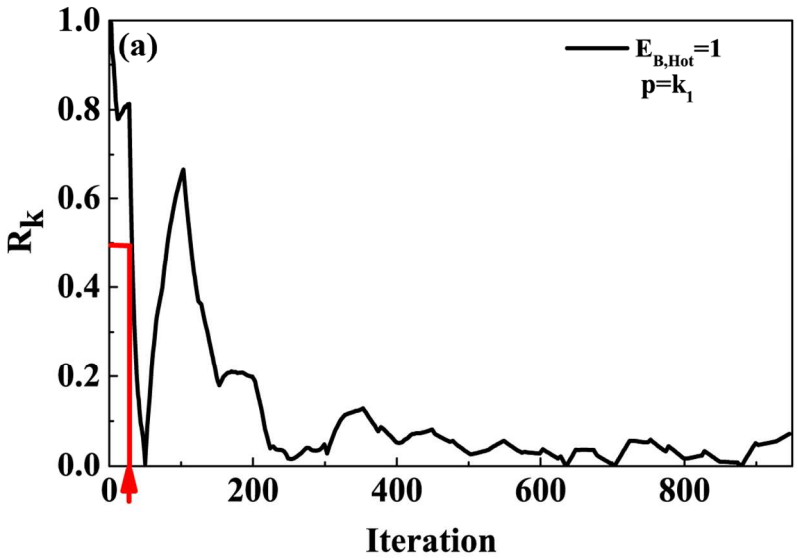


Figure 3. Autocorrelation function for a) EB=1 and b) EB=10-5 (*red arrow indicates).
368x228mm (100 x 100 DPI)

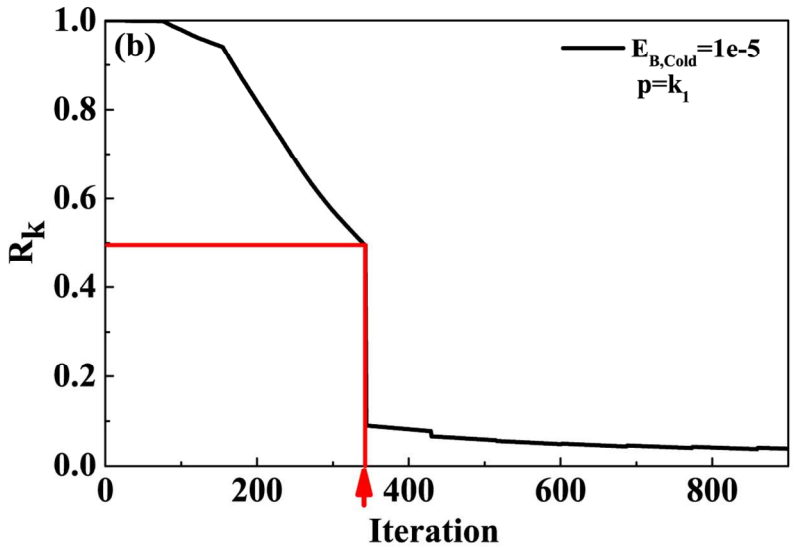


Figure 3. Autocorrelation function for a) EB=1 and b) EB=10-5 (*red arrow indicates).
368x228mm (100 x 100 DPI)

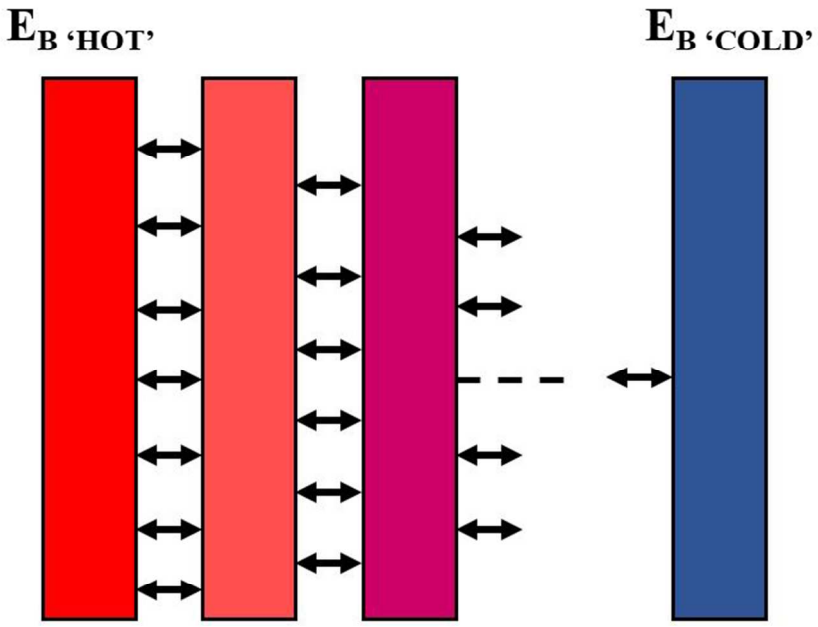


Figure 4. Parallel tempering graphic depiction. Arrows depict the flow of information between parallel MC runs whereas the color indicates qualitatively the Boltzmann energy level magnitude at which each run is being carried out, red being higher, i.e. "hotter" and blue lower i.e. "colder."

222x166mm (96 x 96 DPI)

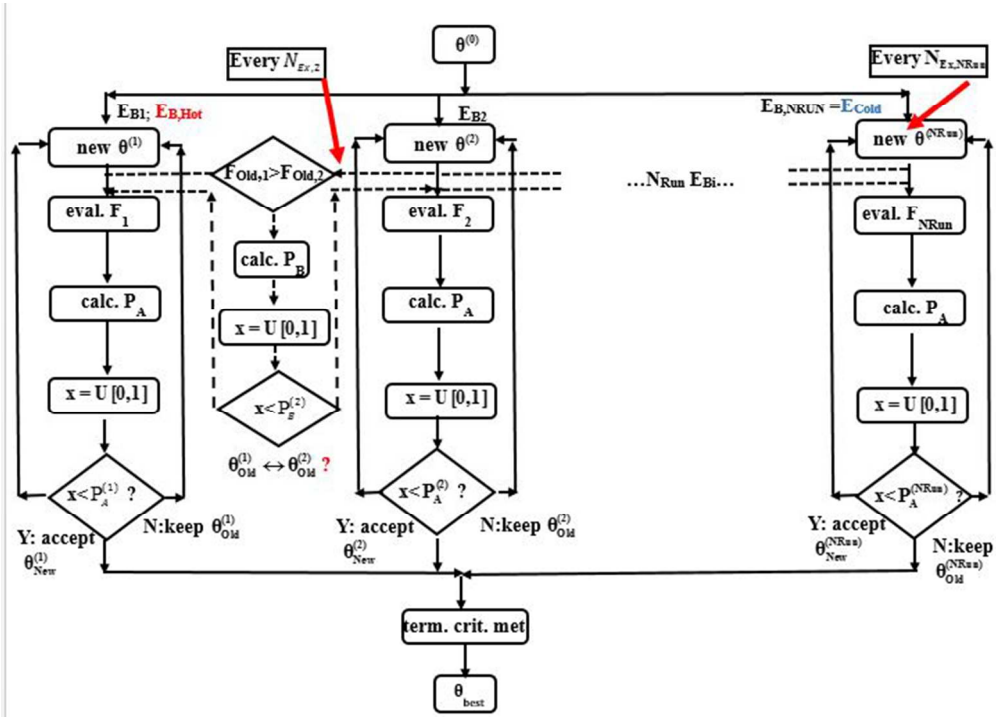


Figure 5. Schematic of the proposed algorithm where P_A and P_B are given by Eqs. (20) and (24), respectively.

191x136mm (96 x 96 DPI)

Figure 5. Schematic of the proposed algorithm where P_A and P_B are given by Eqs. (20) and (24), respectively.
13x4mm (300 x 300 DPI)

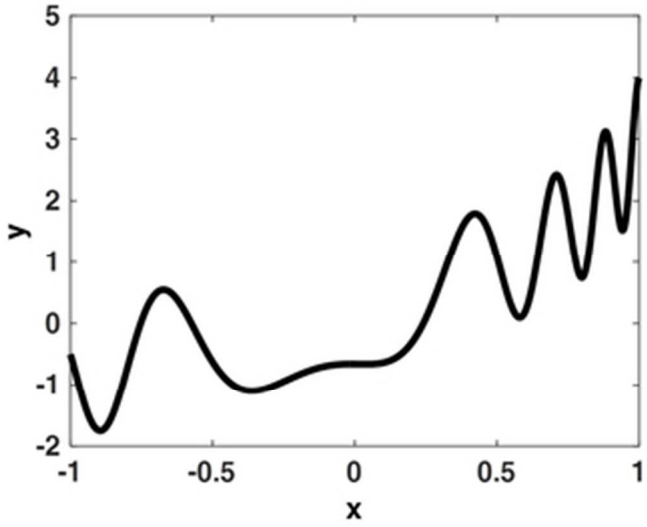


Figure 6. Local vs. global min. (1-D): Objective function (y) vs. parameter values (x) for the 1D algebraic system described in Algebraic Examples section
30x22mm (300 x 300 DPI)

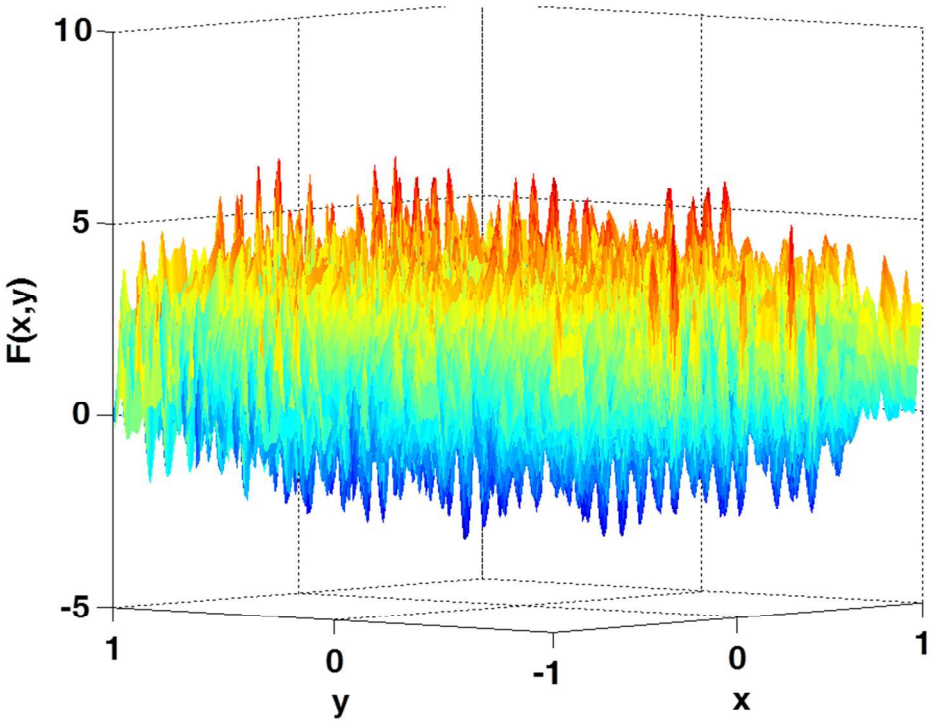


Figure 7. Local vs. global min. (2-D): Objective function (F) vs. parameter values (x,y) for the 2D algebraic system described in Algebraic Examples section
81x60mm (300 x 300 DPI)

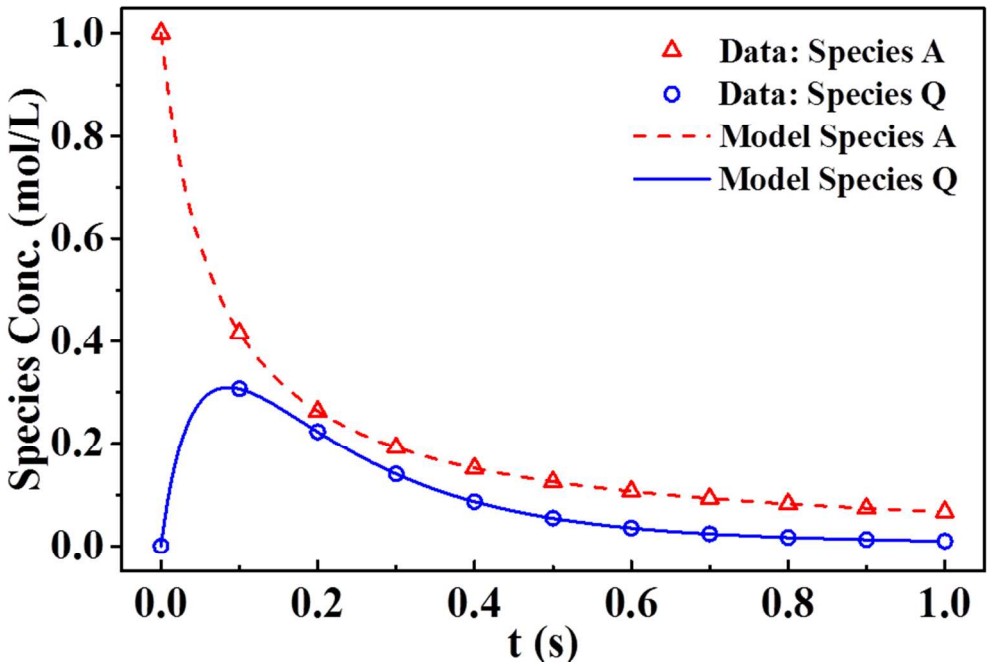


Figure 8. Solution to the catalytic cracking problem for the parameter values k_1=12, k_2=8 and k_3=2

81x60mm (300 x 300 DPI)

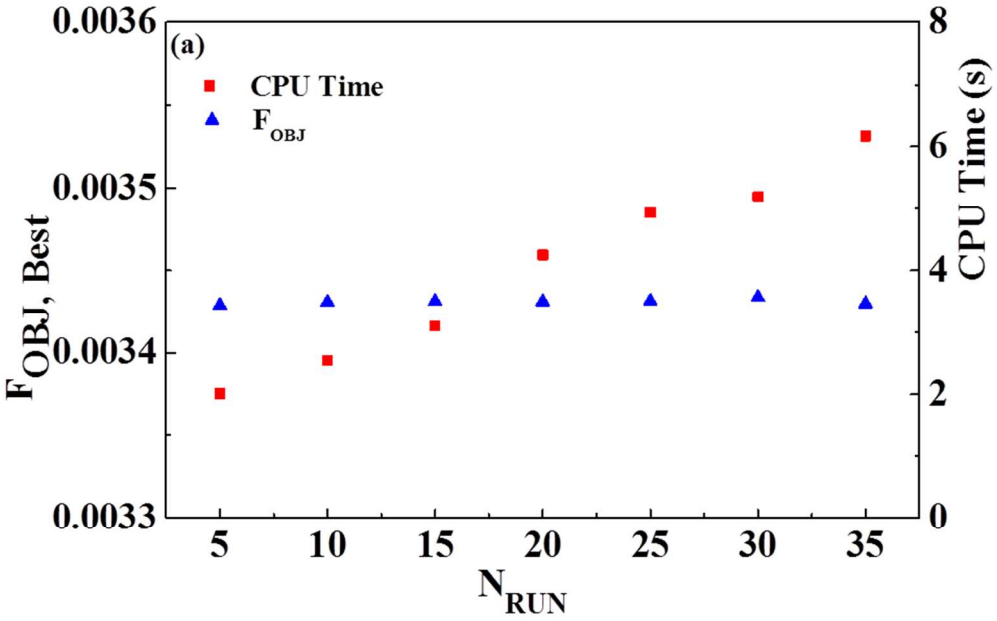


Figure 9. Sensitivity of $F_{obj,best}$ and CPU time on a) the number of $E_{B,i}$ levels, N_{RUN}
81x60mm (300 x 300 DPI)

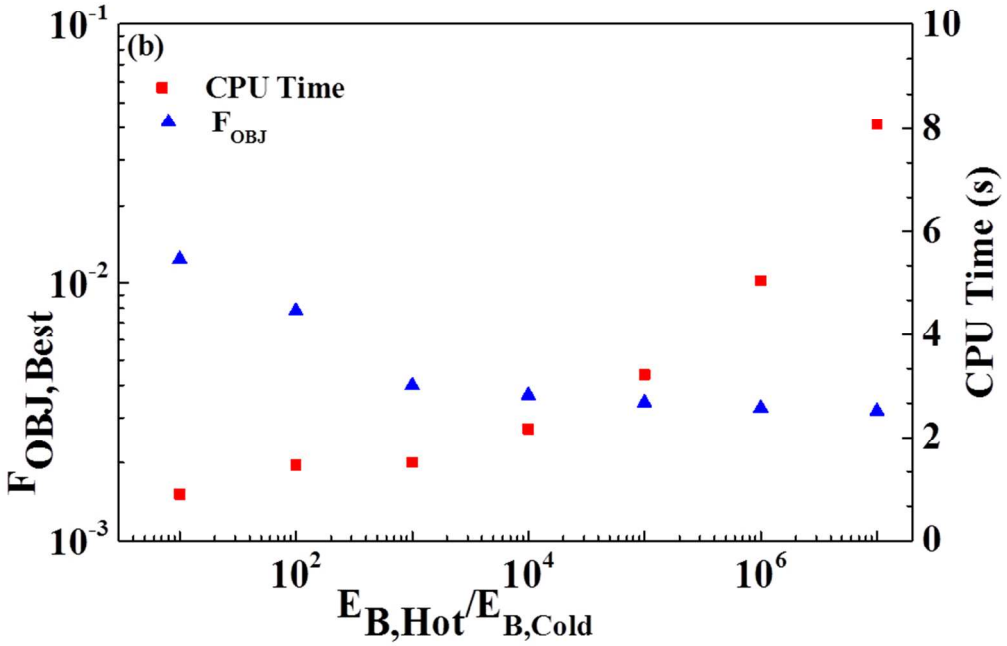


Figure 9. Sensitivity of $F_{obj,best}$ and CPU time on b) the $E_{B,Hot}/E_{B,Cold}$ ratio for the catalytic cracking problem.
81x60mm (300 x 300 DPI)

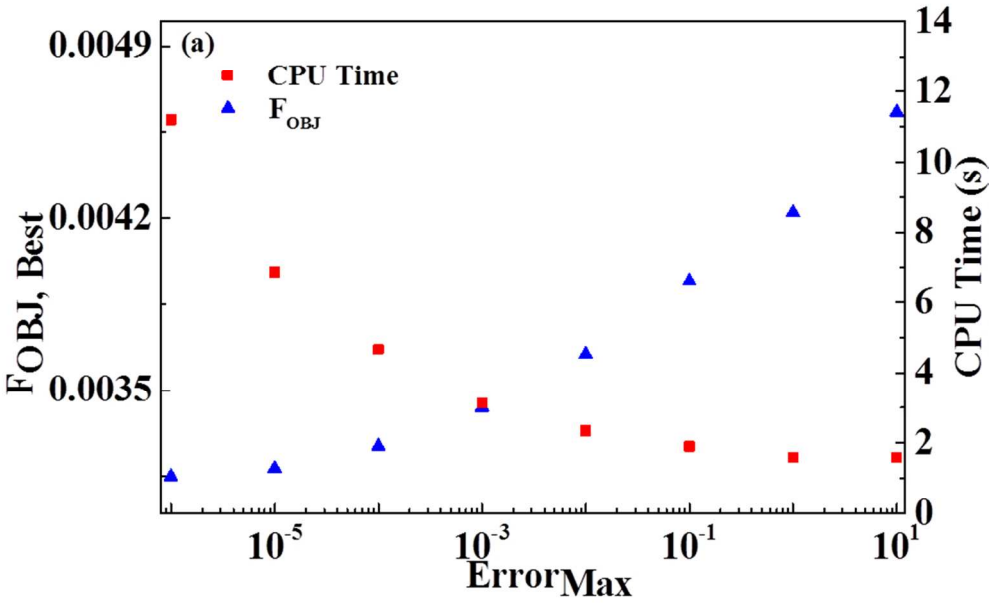


Figure 10. Sensitivity of $F_{obj,best}$ and CPU time on a) the maximum error, $Error_{Max}$
81x60mm (300 x 300 DPI)

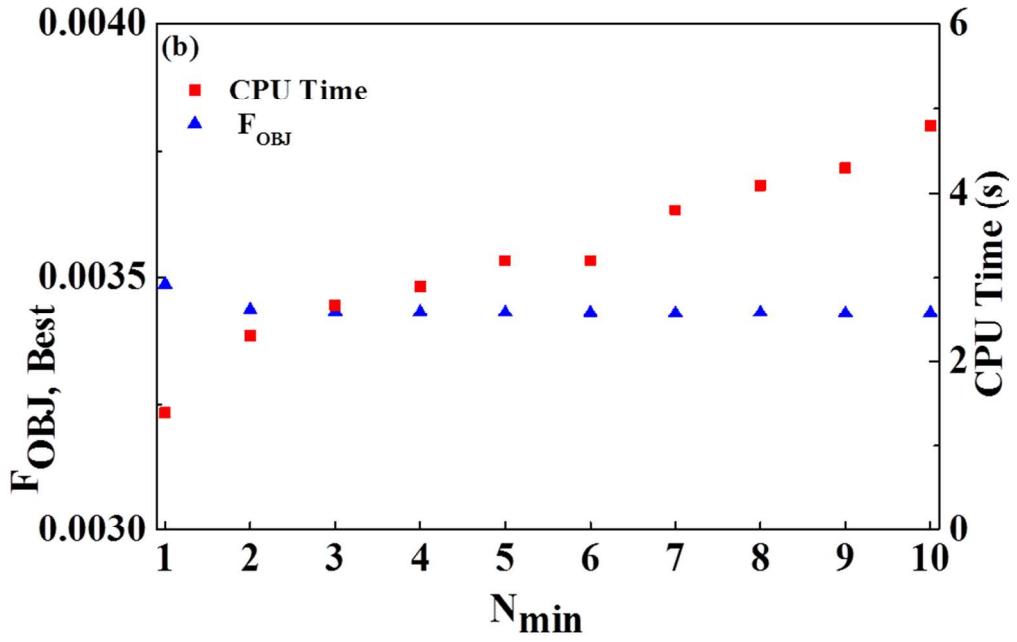


Figure 10. Sensitivity of $F_{obj,best}$ and CPU time on b) the factor N_{min} for the catalytic cracking problem
81x60mm (300 x 300 DPI)

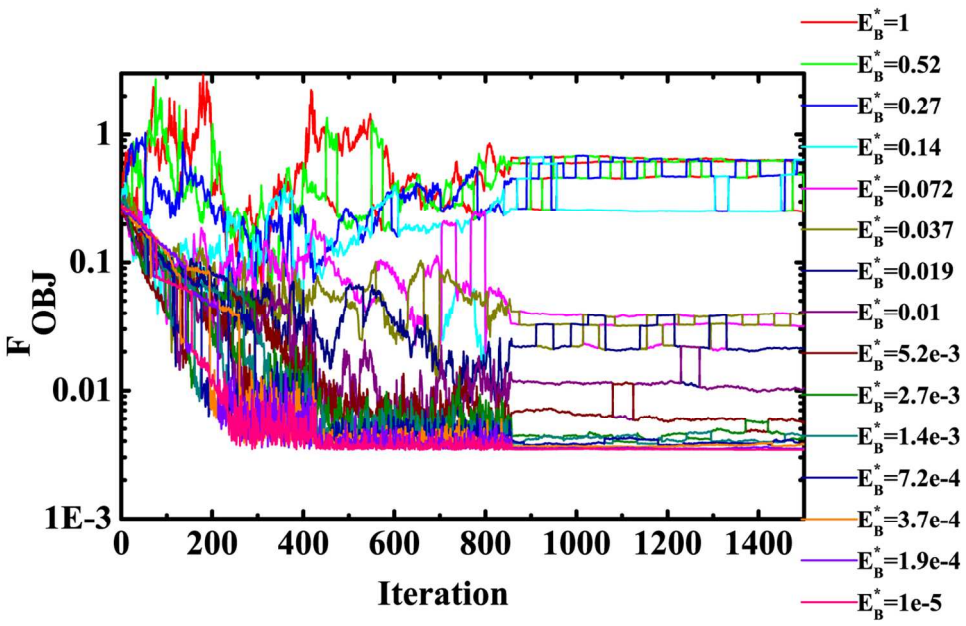


Figure 11. The dependence of F_{obj} within the 15 parallel MC runs on the iteration (MC step) number for the catalytic cracking problem. The legend values on the right show the normalized Boltzmann Energy levels used in the parallel tempering.
368x228mm (100 x 100 DPI)

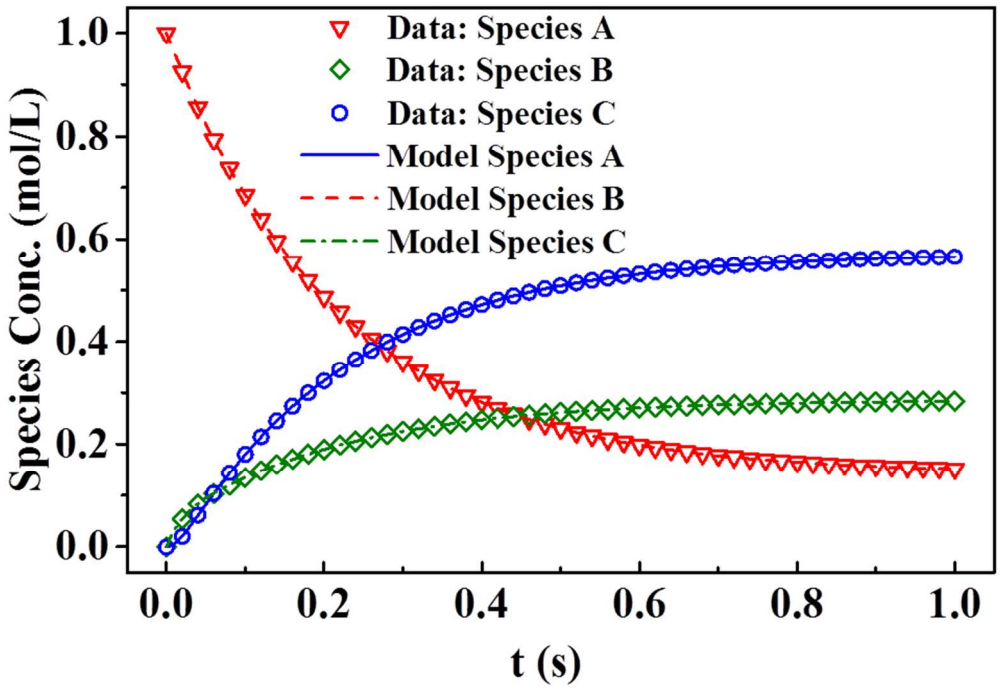


Figure 12. Time evolution for the three species concentration as obtained from the solution for the reversible chemical reactions problem for the parameter values k_1=4, k_2=2, k_3=40 and k_4=20.

81x60mm (300 x 300 DPI)

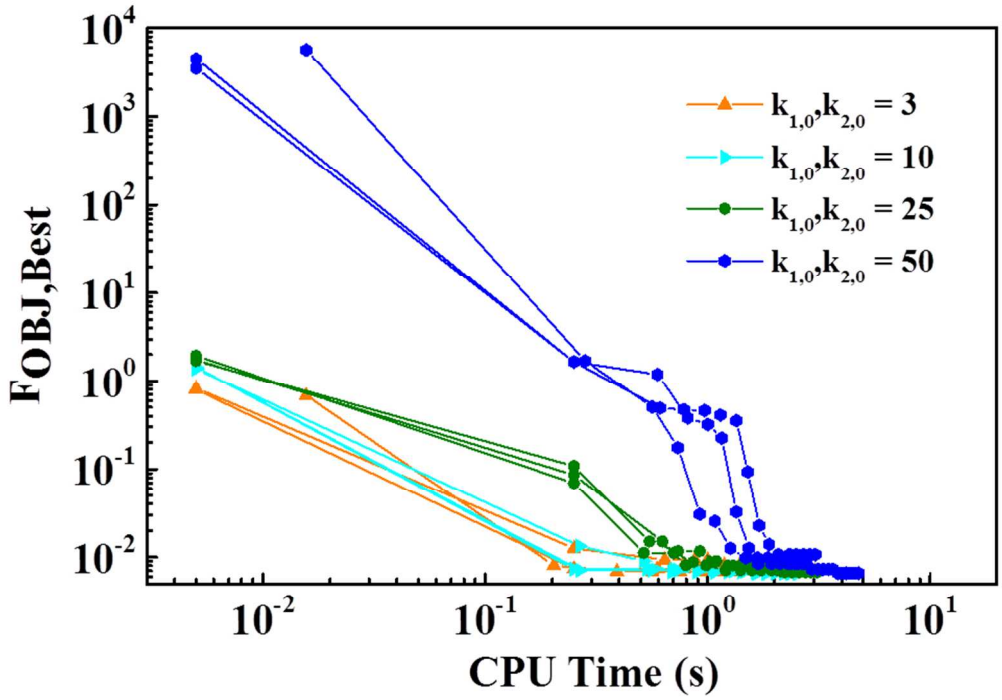


Figure 13. Plot of iterations vs. error of different initial guess values for irreversible chemical reaction

81x60mm (300 x 300 DPI)

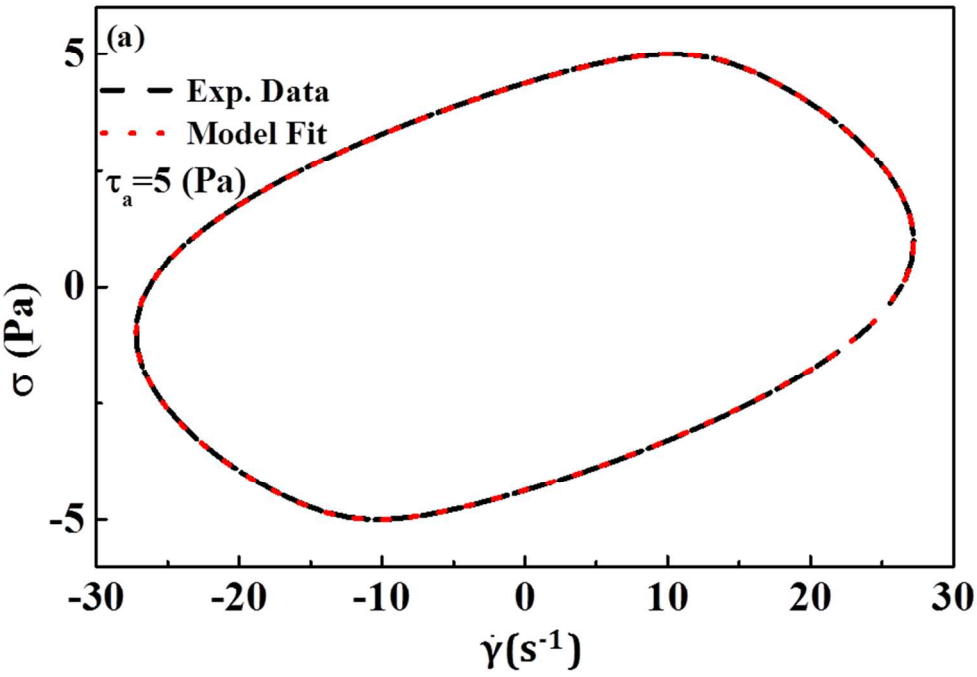


Figure 14. Complex dynamic rheological parameter oscillatory behavior shown in the form of Lissajous-Bowditch plots for two different stress amplitudes as shown in the figures: a) τ_a= 5 Pa

81x60mm (300 x 300 DPI)

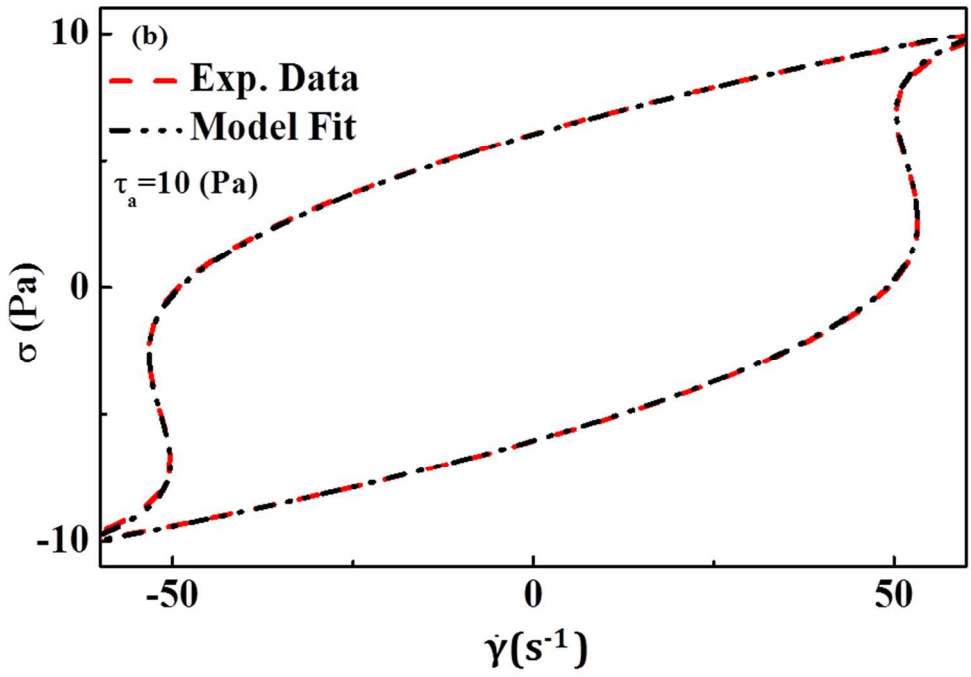


Figure 14. Complex dynamic rheological parameter oscillatory behavior shown in the form of Lissajous-Bowditch plots for two different stress amplitudes as shown in the figures: b) τ_a= 10 Pa
81x60mm (300 x 300 DPI)

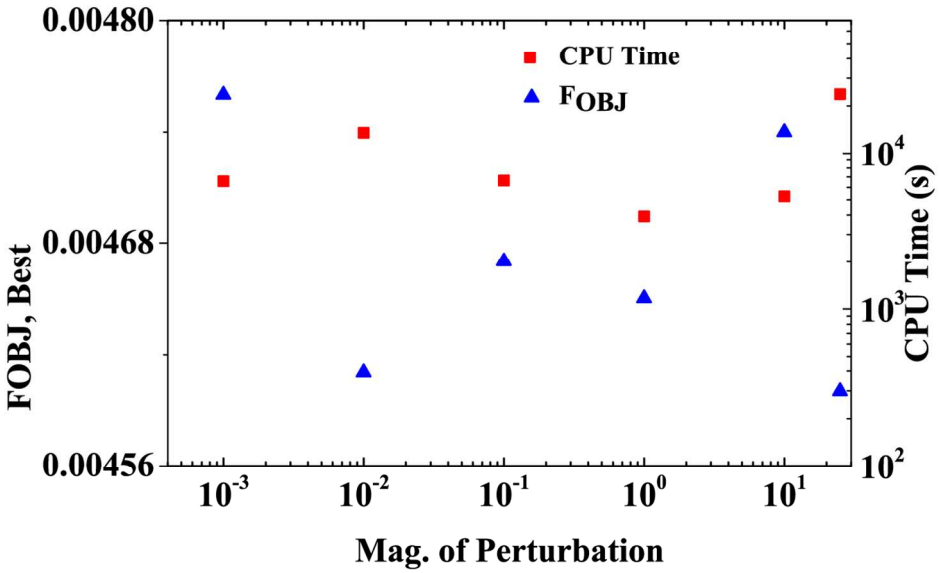


Figure 15. Sensitivity of the $F_{obj,best}$ and CPU time on the magnitude of perturbation applied to the initial guess
368x228mm (100 x 100 DPI)

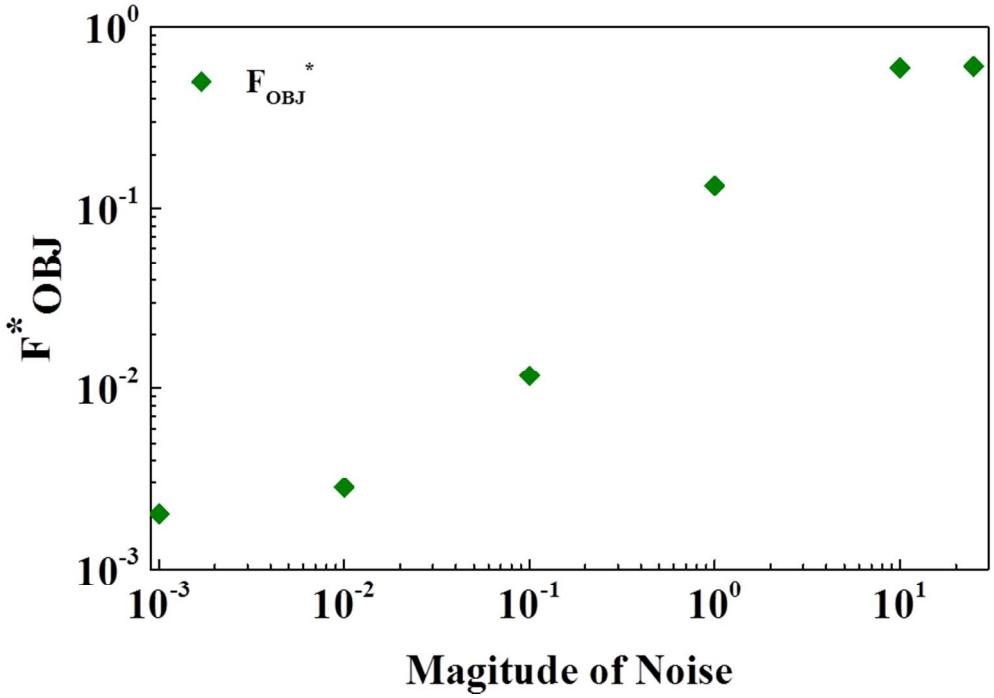


Figure 16. Sensitivity of the F_{obj} to the magnitude of the noise applied to empirical data
81x60mm (300 x 300 DPI)

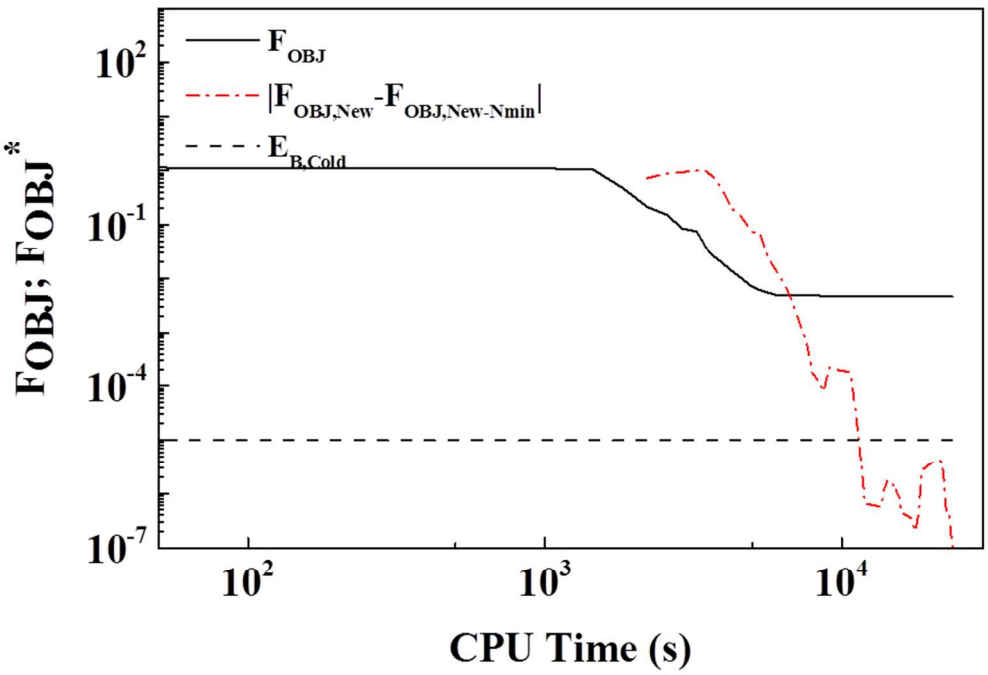


Figure 17. Evolution of F_{obj} and stoppage criteria with CPU time during the parameter fit of the complex dynamic rheological data.

81x60mm (300 x 300 DPI)

Level,i	$E_{B,i}$	$\mu_{k1,i}$	$\sigma_{k1,i}$
1	1.0000	1.91	4.58
2	0.5179	2.98	4.32
3	0.2683	5.02	4.15
4	0.1389	8.07	3.48
5	0.0720	12.26	2.90
6	0.0373	12.35	1.69
7	0.0193	12.20	1.49
8	0.0100	11.98	1.19
9	0.0052	11.59	1.15
10	0.0027	11.56	1.21
11	0.0014	11.53	1.26
12	0.0007	11.51	1.32
13	0.0004	11.50	1.36
14	0.0002	11.50	1.38
15	0.0001	11.54	1.48

72x116mm (300 x 300 DPI)

Description	Parameter	Default Value
Energy Ratio	$E_{B,Hot}/E_{B,Cold}$	1.0+05
Energy Paths	N_{Run}	15
Maximum Error	$Error_{Max}$	1.00E-03
Stoppage Criteria	$\Delta F_{obj,Best}$; N_{min}	$E_{B,Cold}$; 5

215x279mm (200 x 200 DPI)

	Matlab w/ Chebfun*	Matlab w/ simulannealbnd**	Parallel Tempering
Iteration	N/A	Best: 1299; (μ_{iter}=977; δ_{iter}=259)	2150
CPU Time (s)	0.031	Best: 0.313; (μ_{time}=977; δ_{time}=259)	2.79
Error	4.21E-15	Best: 1.85E-13; (μ_{error}=1.3E-03; δ_{error}=1.3E-2)	4.77E-15

*using starting point [0]
**μ, δ calculated from 100 runs

205x74mm (300 x 300 DPI)

	Matlab w/ Chebfun*	Matlab w/ simulannealbnd**	Parallel Tempering
Iteration	N/A	Best: 2914; (μ_{iter}=1948; δ_{iter}=697)	3970
CPU Time (s)	0.531	Best: 0.765; (μ_{time}=0.530; δ_{time}=0.193)	7.1
Error	5.06E-13	Best: 2.90E-6; (μ_{error}=0.359; δ_{error}=0.204)	3.50E-14

*using starting point [0,0]
**μ, δ calculated from 100 runs

213x78mm (300 x 300 DPI)

			Parallel Tempering			AIChE Results[18]	
Function	Number of Parameters	Number of Constraints	F_{OBJ}	Parameter Values	Error Bar	F_{OBJ}	Parameter Values
Sasena	2	3	-0.7465 +/-0.0002	[0.2043, 0.8354]	+/-[1.6 10^{-4},1.3 10^{-4}]	-0.7483	[0.2017, 0.8332]
New Branin	2	1	-268.7833+/-0.0038	[3.274, 0.0487]	+/-[4.6 10^{-3}, 2.1 10^{-3}]	-268.7879	[3.273, 0.0489]
Constrained Branin	2	1	0.39788 +/-2.1 10^{-6}	[9.4249, 2.476]		0.39789	[9.4247, 3.1415]
(Second Solution)				[3.1411, 2.274]			[2.4750, 2.275]
						J. Mech. Design Results[69]	
Weight of Spring	3	4	0.01562+/-3.9 10^{-3}	[0.04679, 0.4058, 10.995]	+/-[1.1 10^{-2}, 2. 10^{-2}, 3.7 10^{-2}]	0.012665	[0.05156, 0.35363, 11.47]
			*Best: 0.0068	[0.02558, 0.40523, 10.9411]			

279x215mm (200 x 200 DPI)

θ_i	Tjoa and Biegler (1991)[70]	Parallel Tempering*
k_1	11.948	12.005
k_2	7.993	7.998
k_3	2.024	2.002

*using starting point: [6, 4, 1]

215x279mm (200 x 200 DPI)

θ_i	Tjoa and Biegler (1991)[70]	Floudas and Esposito (2000)[27]	Parallel Tempering*
k_1	3.997	4.001	3.999
k_2	1.998	2.001	1.999
k_3	40.538	39.8	39.97
k_4	20.264	19.9	19.98

*using starting point: [10, 10, 30, 30]

215x279mm (200 x 200 DPI)

θ_i	Tjoa and Biegler (1991)[70]	Parallel Tempering*
k_1	5.002	5.000
k_2	1.000	1.000

*using starting point: [3, 4]

215x279mm (200 x 200 DPI)

	$k_{1,0};k_{2,0}=3$	
	$k_{1,best}$	$k_{2,best}$
1	5.008	0.999
2	5.005	1.001
3	5.009	1.001
μ_i	5.005	1.001
σ_i	1.70E-03	9.43E-04

	$k_{1,0};k_{2,0}=10$	
	$k_{1,best}$	$k_{2,best}$
1	4.996	1.001
2	5.001	1.003
3	5.002	0.998
μ_i	5	0.997
σ_i	2.63E-03	2.06E-03

	$k_{1,0};k_{2,0}=25$	
	$k_{1,best}$	$k_{2,best}$
1	4.992	0.998
2	4.993	1.001
3	4.993	0.998
μ_i	5.005	1.41E-03
σ_i	4.71E-04	9.43E-04

	$k_{1,0};k_{2,0}=50$	
	$k_{1,best}$	$k_{2,best}$
1	4.999	1.000
2	4.971	1.007
3	5.002	1.005
μ_i	4.991	0.997
σ_i	1.40E-02	2.94E-03

173x110mm (300 x 300 DPI)

θ_i	de Souza and Thompson[50]	Parameter Estimate
τ_{YD} (Pa)	1.00	1.00
τ_Y (Pa)	2.00	2.00
γ_{YD} (Pa)	1.00E-04	1.00E-04
η_0 (Pa s)	1.00E+07	1.00E+07
η_∞ (Pa s)	0.01	0.01
G_0 (Pa)	1.00	1.00
m	1.00	1.00

137x102mm (300 x 300 DPI)

θ_i	de Souza and Thompson[50]	Parallel Tempering
n (Pa)	0.50	0.49999
K (Pa s^n)	1.00	1.00001
t_{EQ} (s)	0.010	0.01000

141x54mm (300 x 300 DPI)