Parameter Identification with Adaptive Sparse Grid-based Optimization for Models of Cellular Processes

Maia M. Donahue[1], Gregery T. Buzzard[2], Ann E. Rundell[1]
[1]Weldon School of Biomedical Engineering, Purdue University,
[2]Department of Mathematics, Purdue University

Key Terms

global optimization, parameter estimation, MAPK, systems biology, genetic algorithm

Abstract

Identifying parameter values in mathematical models of cellular processes is crucial to ascertain if the hypotheses reflected in the model structure is consistent with the available experimental data. Due to the uncertainty in the parameter values, partially attributed to the necessary model abstraction of any cellular process, parameters are pragmatically estimated by varying their values to minimize a cost function that represents the difference between the simulated results and available experimental data. Local searches for these parameter values rarely result in an adequate fit of the model to the experimental data since the optimization gets caught in a local minimum near the initial guess. Typically, larger regions of the parameter space must be searched for acceptable parameter values to support model simulations that replicate the experimental data. Most of these global algorithms use stochastic sampling of the parameter space; however, these methods are not computationally efficient and cannot guarantee convergence. Alternatively, adaptive sparse grid-based optimization samples the parameter space in a more systematic manner and employs selective evaluations of the cost function at support nodes to build an error-controlled interpolated approximation of the cost function from basis functions. The search for the global minimum is performed on the surrogate interpolant rather than relying extensively on simulations of the model; this tends to be more computationally efficient for smooth, continuous models. Additional insight is provided by the cost function

mapping on the parameter space, which can be useful for evaluating and refining the model and parameter identification problem. This chapter describes the methods to create and use an adaptive sparse grid and interpolant to find parameter values, including an example that demonstrates that this method performs more accurately, consistently, and efficiently than the genetic algorithm, a standard global stochastic optimization method.

Introduction

Increasingly, mathematical models are being used to provide insight into cellular processes [1, 2]. The construction of these models is hampered by the sheer number of participating chemical species, the uncertainty and complexity of the interconnected signaling networks, and the complicated regulation of the genetic events within a living cell. Out of necessity, the model structure must explicitly represent only the dominant events and processes for a specific application. Determining the dominant events and processes a priori is usually not trivial; hence, the first step of determining if the model structure is suitable for the specific application typically depends on finding model parameters that produce simulations consistent with available experimental data and observations.

Most model parameter values are not known accurately, due to both experimental issues and omission of non-essential process details in the model. Experimentally, it is difficult to measure the concentrations, rates, diffusion, etc. of elements within a living intact cell [3-5]. Enzyme-substrate association constants and kinase activity rates can sometimes be determined in a test tube, but there is no guarantee that these rates are the same when inside a crowded cellular environment [6]. Furthermore, the inevitable abstraction of the process being modeled causes the majority of model parameters to incorporate the net effect of a multitude of events. As a

result, parameter values typically can be determined only through optimization that minimizes the difference between the simulated model output and the experimental data. This process is straightforward for linear models through linear programming. However, most optimization tools are challenged by these models since they can be highly nonlinear.

In rare cases where very good estimates of parameter values are available, a local search can be adequate to find parameter values that minimize the differences between model simulations and experimental data, which is quantified as the value of a cost function. A local search starts from an initial point and finds the direction that allows the largest decrease in model/data mismatch. The search ends at the nearest minimum in that direction, shown in Figure 1. As Figure 1 demonstrates, the result of the local search is dependent on the initial point. Two of the three example points are caught in the nearest local minimum, a consequence termed the local minimum trap. If possible, a local search will modify the parameter values from the initial set to improve fitting; however it cannot move out of the local minimum trap to find a global minimum. In contrast, global searches consider the entire parameter space when locating the global minimum.

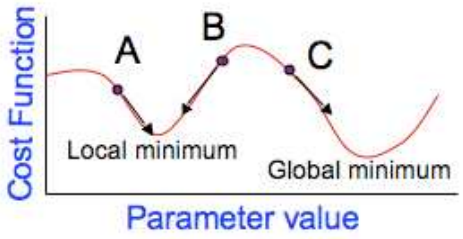

Figure 1: This plot illustrates the concept of a local minimum trap for a one-dimensional parameter space. A local search of this space is initialized at three different points: A, B, and C. Searches starting at points A and B will find a local minimum of the cost function, while a search starting at point C results in the global minimum.

Many global optimization methods can be used to solve the parameter identification problem but these can also suffer from the local minimum trap as well as from poor convergence rates [7] and are computationally costly. For a subset of systems, the problem can be solved using deterministic optimization methods that transform the problem into a convex function or the difference between convex functions [8]. In these systems, achieving the global minimum can be guaranteed [7, 8]. As the applicability of these deterministic strategies is limited for complex, nonlinear models, many researchers resort to global optimization techniques that sample the parameter space in a stochastic manner. Existing popular stochastic methods include simulated annealing, genetic algorithms (GA) and multiple shooting strategies. The GA, for example, uses evolution-based strategies to modify a population of parameter sets, with a higher probability of keeping sets with high fitness (low cost function values) than those with low fitness; the retained parameter sets become parent sets that are randomly combined to create children sets for evaluation in the next iteration [9]. Due to the probabilistic nature of these stochastic methods, the parameter set values and corresponding cost function value can vary considerably from run to run. In addition, these stochastic global optimization methods are computationally expensive and do not necessarily converge to a solution; hence, it can take a long time to discover that no solution exists.

Alternatively, the entire parameter space can be searched using a grid algorithm. Grid algorithms divide the parameter space in a patterned manner and evaluate the cost function at each grid point, see Figure 2. Local or global searches can be initiated from one or more of the best grid points to find acceptable parameter estimates. As the dimension of the uncertain parameter space increases, the number of model evaluations required to cover the entire space increases exponentially for optimization with an evenly spaced (full) or pseudo-randomly spaced

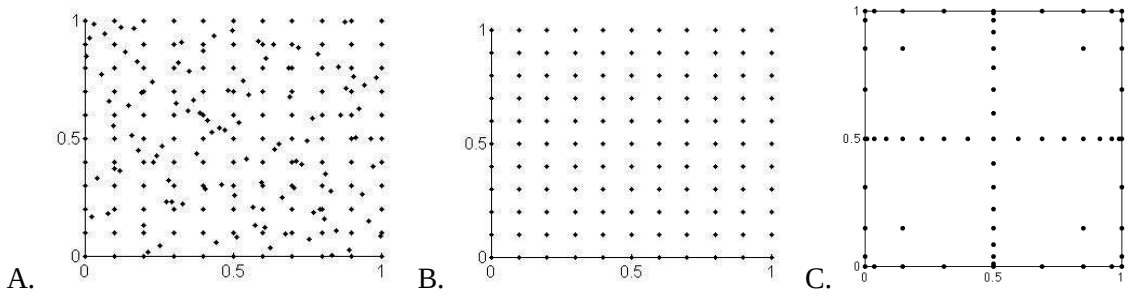

Figure 2: Examples of grids sampling a two dimensional parameter space. A. latin hypercube sampling B. full uniform grid C. Chebyshev sparse grid, generated with the Sparse Grid toolbox [10].

(latin hypercube sampling, LHS) grid [11]. Randomly spaced samples are not recommended due to inefficient clustering at some areas and inadequate sampling at others. However, adaptive sparse grid schemes avoid this exponential increase in points by selective positioning of support nodes.

Adaptive sparse grid interpolation

Recently sparse grid interpolation approaches have been developed that support deterministic global optimization for the minimization of functions with bounded mixed derivatives [12]. These methods are currently being refined for efficiently solving large dimension problems, more than 10 uncertain parameters [13]. Sparse grid interpolation techniques were originally developed to reduce the computational cost for multivariate integrals [11, 14, 15]. A thorough review of sparse grid-based interpolation and integration is given in [16].

Adaptive sparse grid-based optimization utilizes the error-controlled interpolant as a surrogate of the cost function to search for the minimum. The process for optimizing with sparse grids requires model evaluations at selected grid points (support nodes) strategically positioned within the uncertain parameter space. An interpolated function is created by combining basis functions at the support nodes to approximate the cost function evaluated over the entire uncertain parameter space. The search for the best parameter values is performed on the

interpolated function. Typically, a search along a polynomial-based interpolation function is significantly faster than a search involving repeated numerical integrations of a model, see Figure 3. Using sparse grid interpolation, the number of actual model evaluations is limited to just the number of support nodes. However, since the sparse grid technique relies on an approximation, the best results are usually obtained if a local search using the actual model is performed starting from the optimal values identified by the interpolation function.

This adaptive sparse grid-based optimization approach and its computational efficiency rely heavily upon the construction of the interpolating function and selection of the support nodes. In brief, the construction of the interpolating function is based upon the tensor products of a univariate interpolation of the function, *f*, at the support nodes, $x^{i_k} \in \chi^{i_k}$ for $k \in [1, d]$, with a basis function, a_{x^i} :

$$\left(U^{i_1} \otimes \cdots \otimes U^{i_d}\right)(f) = \sum_{x^{i_1} \in \chi^{i_1}} \cdots \sum_{x^{i_d} \in \chi^{i_d}} \left(a_{x^{i_1}} \otimes \cdots \otimes a_{x^{i_d}}\right) \cdot f\left(x^{i_1}, \ldots, x^{i_d}\right).$$

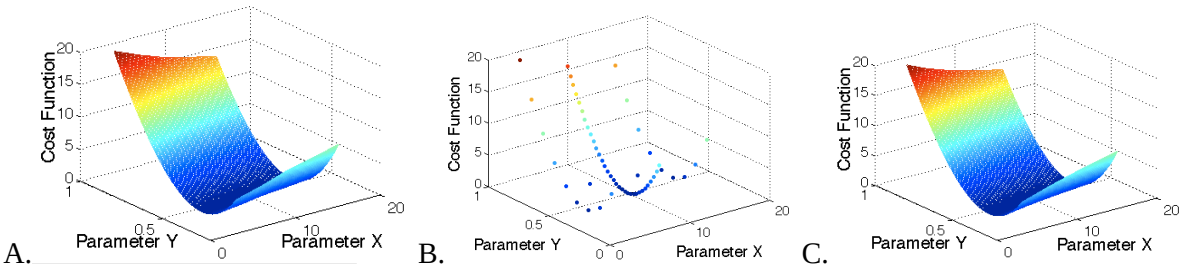

Figure 3: Comparison of meshes created by actual cost function evaluations and from an interpolated cost function for a two-parameter search of a MAPK model [17]. A. This mapping of the cost function was created from a 100x100 evenly spaced grid of parameter sets, for a total of 10000 model evaluations. A local search on the best mapping point returned the actual parameter values, with an additional 18 model evaluations. B. The 53 adaptive sparse support nodes used to create the interpolated function, generated by the Sparse Grid Toolbox [10]. C. An evenly spaced 100x100 grid of parameter sets was created and evaluated by the interpolated function, creating an identical mapping to A that only required the 53 model evaluations used to create the support nodes. A local search from the best support node took an additional 66 model evaluations to return the actual parameter values.

In the sparse grid approach introduced by Smolyak, the interpolating function is obtained by summing a selected set of such tensor products. The computational efficiency of this method is a result both of the fact that this selected sum requires a relatively small set of support nodes and the fact that the sets of support nodes are nested as the interpolation depth increases. This nesting property greatly reduces the number of required function evaluations by reusing the support nodes upon increased sampling refinement of the grid for a higher interpolation depth. The interpolation depth is the degree of the polynomial, *k*, which the univariate interpolation function can exactly match.

It has been shown that the error of the interpolating function strongly depends upon the degree of the bounded mixed derivative (smoothness) and is a weak function of the dimension of the problem, $O\left(N^{-k}\left(\log N\right)^{(k+1)(d-1)}\right)$, where N is the number of function evaluations performed on the sparse grid at the support nodes [15]. Hence these methods are considered nearly optimal (up to a logarithmic factor) [15] and are significantly better than those of quasi-Monte Carlo algorithms, $O\left(N^{-1}\left(\log N\right)^{d}\right)$ [18]. A uniform sparse grid cannot avoid a logarithmic dependence of the error on dimension; however, adaptive sparse grids sample most along the dimension of greatest importance as ascertained by the ability of samples in that direction to decrease the estimated interpolation error (Figure 4) [18]. This "problem-adjusted refinement" [19] most effectively reduces the computational costs for the optimization on models whose roughness is confined to a subset of the dimensions of the uncertain space and it does no worse than the uniform sparse grid methods. This adaptive sparse grid-based optimization method is deterministic so the numerical values of the identified parameter values and the quality of the results will not differ from one run to the next. Furthermore, it is anticipated that the quality of

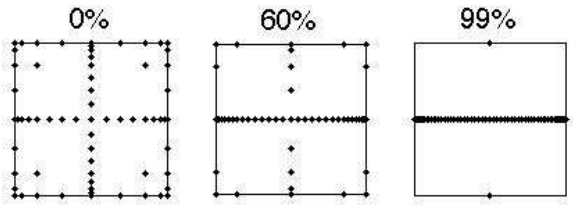

Figure 4: Examples of two-dimensional adaptive Chebyshev sparse grids, with increasing degree of adaptivity from left to right, generated with the Sparse Grid toolbox [10]. This figure demonstrates that the parameter along the x-axis is more important to decreasing interpolation error than the parameter along the y-axis.

the results will improve with an increased sample size since the error of the interpolant approximation of the cost function mapping on the parameter space decreases with large N.

Experimental Design

The method described in this chapter will determine, in a relatively efficient manner, the optimal parameter values to fit a model to all available experimental data. A number of factors must be considered to formulate the problem as a parameter identification experiment utilizing sparse grid based interpolation. These factors involve ascertaining the dimension of the uncertain parameter space, the size of the uncertain parameter space, the form of the cost function, and the selection of the basis functions for the interpolating function. The *dimension of the uncertain parameter space* must be determined; that is, the number of parameter values to be found needs to be established. It is desirable to keep this dimension as small as possible; initially assign the parameter values for which reasonable estimates exist. For instance, total numbers of molecules or concentrations of certain elements may be experimentally established or obtained from other published models. In cases where there are too many parameters for which there is no good estimate of their values, the dimension of the problem can be reduced by conducting a local or global sensitivity analysis [20] about some initial starting guess to ascertain which parameters should be targeted for fitting the data (see Troubleshooting) [21, 22]. The initial starting guess

values can be roughly estimated by back-of-the-envelope calculations or obtained from published models of similar reactions or processes. Parameters with the lowest sensitivity ranks can be neglected and fixed at these initial guesses. For each remaining parameter, which will be labeled as 'uncertain,' an estimated initial search range must be provided. The product of these search ranges defines *the span of the uncertain parameter space*. Our experience has found that using a search range encompassing an order of magnitude below and above an initial guess will, in most cases, be large enough. The search for the potential parameter values should typically be conducted on the log of the uncertain parameter space to more equally spread out the support nodes over the ranges, which vary in orders of magnitude.

For parameter identification, the optimization problem typically minimizes a *cost function* that penalizes differences between the model simulations and the experimental data. The most commonly used cost function, when quantitative experimental data is available, is the weighted least square error:

$$F(p) = \log(\sum_{j=1}^{q} \sum_{i=1}^{n_j} w_{ij}[y_j(p,t_i) - \hat{y}_{j,i}]^2), \quad (1)$$

where q is the number of states with experimental data, n_j is the number of experimental time points for state j, $\hat{y}_{j,i}$ is the data for state j at time i, $y_j(p,t_i)$ is the simulated model output for state j at time i for parameter set p, and w_{ij} is the weight for that point. For this construction of the cost function, it is important that the simulated data, $y_j(p,t_i)$, be converted into the same units as the experimental data, $\hat{y}_{j,i}$, i.e. numbers of molecules, concentration, percent of total, etc. The weights are used to normalize and/or incorporate confidence information in the data points; the confidence in the experimental data is typically taken into account by making the weights the reciprocal of the standard deviation of the experimental data at each time point and

state, while the maximum value of the data or simulations for each state is typically used when the values of the states differ significantly in magnitude. When quantitative data is not available, a qualitative cost function can be constructed that, on a smooth scale, penalizes or rewards attributes of the simulations. It is important that the cost function be continuous in order for the interpolating function to approximate it accurately without large numbers of support nodes.. Abrupt jumps due to, for instance, if-else statements will severely increase the interpolation error, as the cost function is interpolated with continuous basis functions. It is also recommended to search over log space; taking the log of the cost function will increase its smoothness.

A wide variety of different *basis functions* can be used to support the construction of the interpolation function on the sparse grid including piecewise linear, Chebyshev polynomials, polynomial chaos [23], and multi-wavelet formulations [24]. Though the choice of basis function changes the placement of the support nodes, the construction of the interpolant is the same. Barthelmann et al compared the two most popular basis functions: piecewise linear and Chebyshev polynomial interpolation [15]. They concluded from theory and computation that if the function to be approximated is three (or more) times differentiable, then polynomial basis functions are better in the sense that the interpolation converges to the correct answer more quickly as the number of support nodes increases. If the function to be interpolated is discontinuous, then convergence is slow for both. In general, the authors recommend Chebyshev polynomial interpolation [15]. Therefore, this chapter is written from the assumption that Chebyshev interpolation will be used.

Materials

The materials needed to apply this adaptive sparse grid-based optimization method for model parameter identification from available experimental data are described in Table 1. The specific implementation discussed in this chapter requires MATLAB® and the Sparse Grid toolbox (http://www.ians.uni-stuttgart.de/spinterp) [10]. However, this method can be implemented with alternative coding packages, such as C++. Therefore, the required materials are described generically, with specifics provided in parentheses.

For the examples in this chapter, a published four state, ordinary differential equation (ODE) model [25] of the mitogen activated protein kinase (MAPK) cascade was used. For these illustrations, mock experimental data was generated by model simulation. The mock data consisted of seven time points of the simulations for two of the four states.

Table 1: Materials needed for optimization with adaptive sparse grid interpolation

	Implementation (with specifics provided in parenthesis)
Hardware	Computer capable of running preferred model simulation software (MATLAB®)

Software	Simulation software (MATLAB®)
	Adaptive sparse grid algorithm (The Sparse Grid toolbox [10], installed, initialized, and modified slightly to store the best grid point for future use: the function `spcgsearch` was modified by inserting the code
	`pb = x;`
	`cf_pb = fprev;`
	`save pb pb cf_pb`
	at line 106, which is immediately after the lines:
	`% Determine optimization start point`
	`[x, fval] = spgetstartpoint(z, xbox, options);`
	`fprev = fval; )`
	A local search algorithm such as the conjugate gradient method (`fmincon`)

Model code	Model and cost function files written in preferred software format (m-files). These files should output the cost function value for the model evaluated at a given set of uncertain parameter values.
Data	Experimental data to fit uncertain model parameters. A local and global sensitivity analysis can help ascertain if the data available is sufficient to identify uncertain model parameters [21, 22]. The data must be accessible by the preferred software (.mat file).

Methods

The general method for parameter identification with sparse grid-based optimization is outlined below with example code provided in Figure 5 using MATLAB® and the Sparse Grid Toolbox [10].

General Procedure:

1. The objective of step 1 is to specify the range over which the search will be conducted for the uncertain parameter value in each dimension. Create a matrix containing the lower bound and upper bound for each uncertain parameter (typically, an order of magnitude lower and an order of magnitude higher than the initial point, respectively). *Specific:* The matrix should have size $d \times 2$ where d is the dimension, the number of uncertain parameters.
2. The objective of step 2 is to select the basis function type and establish the desired grid size and type. As the support node locations are a function of the basis functions used to create the interpolating function, the basis function type must be indicated. To constrain the computational time, we recommend setting the maximum number of grid points to 50-500 times the number of uncertain parameters, depending on how long the model simulations take. One could instead specify a minimum interpolation error to achieve, but this method could take a significant number of model evaluations, which is unknown a priori. We highly recommend enabling dimension adaptivity since the computational effort required is no worse than that for a uniform grid, but for some models it can be significantly more efficient. The degree of adaptivity can be modified to ensure a moderate coverage of the uncertain parameter space if desired. *Specific:* Use `spset` to set these options.

```
% STEP 1: Create search range for parameter space

% In this example, the initial point, k0, is a random point between
% 0.1 and 10 times the actual parameter values, kp
load kp
k0 = rand(0.1,10,'unif')*kp;

% the dimension of the space is the number of parameters
dim=length(k0);

% the span of the search is 0.1 to 10 times the initial point
span=[.1, 10];

% the search is performed over the log of the parameter space
rnge = log(k0*span);


% STEP 2: set the options for the grid type and size and other options
options = spset('KeepFunctionValues','on', ...  % Keep function values at grid points for later use
                'KeepGrid','on', ...            % Keep grid points for later use
                'GridType', 'Chebyshev', ...    % The Chebyshev grid type is used
                'DimensionAdaptive', 'on', ...  % The grid is set to 100% dimension adaptive
                'DimAdaptDegree', 1, ...
                'MaxPoints', 1000);             % The maximum number of points is set

% STEP 3: Create the sparse grid using 'spvals' The cost function is evaluated at each grid point
% the structure, z, stores the grid point locations and values and is used
% to evaluate the interpolated function
z=spvals(@cost_function, dim, rnge, options);

% Sort grid points by cost function value ('points_sorted') and determine the number
% of unique points per parameter ('uniquepts')
for k=1:dim
    points(:,k)=z.grid{1}(:,k);
    uniquepts(k)=size(unique(points(:,k)),1);
end

for k=1:length(points(:,1))
    c(k)=spsurfun(points(k,1:dim),z);
end

[c_sorted i_sorted]=sort(c);

points_sorted=zeros(length(c),dim);

for j=1:length(c)
    points_sorted(j,:)=exp(points(i_sorted(j),1:dim));
end

% STEP 4: Search the interpolated function for the optimal parameter set,
% psgi, which has an interpolated cost function of cf_psgi
[pi, cf_pi] = spcompsearch(z);


% STEP 5: From the returned optimal point, perform a local search on the actual
% cost function, returning the second optimal point with an actual cost
% function of cf_psgi
[pil cf_pil]=fmincon(@cost_function,pi,[],[],[],[],zeros(1,dim));

% STEP 6: Load the previously-stored best grid point, psgb, in the data file
% psgb.mat, which has a cost function value of cf_psgb
load pb

% If the point x differs from the point xopt perform a local search on the actual cost
% function from x, returning the third optimal point with an actual cost function
% of cf_psgb
pick_pbl = 0;
if sum(pi-pb)~=0
    [pbl cf_pbl]=fmincon(@cost_function,pb,[],[],[],[],zeros(1,dim));
    if cf_pbl<cf_pil
        pick_pbl=1;
    end
end

% STEP 7: Of psgi and psgb, pick the parameter set with the lowest cost
% function value as the result
if pick_pbl==0
    display('Optimal parameter set found is')
    pil
    display('With a cost function of')
    exp(cf_pil)
else
    display('Optimal parameter set found is')
    pbl
    display('With a cost function of')
    exp(cf_pbl)
end
```

Figure 5: Example code for implementing optimization with adaptive sparse grid interpolation using the Sparse Grid toolbox [10].

3. The objective of this step is to evaluate the cost function value at each support node and to use these values to create the interpolating function. This requires an iterative solution to add and locate the support nodes in the sparse grid to continuously improve the accuracy of the interpolating function to the cost function value until the maximum number of grid points has been reached or minimum relative or absolute error tolerance has been achieved. In addition, sort the grid points by cost function value (low to high) and determine the number of unique points per parameter. The sorted grid points and number of unique points can be used for further analysis. *Specific:* Use `spvals` to construct the grid and the interpolating function from the basis functions, and use the `sort` function to sort the grid points.
4. The objective of step 4 is to use the interpolated function from the previous step to estimate the 'optimal' parameter set. A search is performed on the interpolated function, which serves as a surrogate for the cost function, to find the optimal parameter values that minimize the interpolated estimate of the cost function. *Specific:* Use the appropriate search function for the basis functions selected: `spcgsearch` for Chebyshev polynomials. This algorithm will find the grid point with lowest cost function and run a local search on the interpolated function about that point. The result of this search we will denote as p_i, which has an interpolated cost function of cf_{pi}.
5. The objective of this step is to refine the sparse grid interpolated 'optimal' parameter set, p_i, by searching for the nearest minimum of the actual cost function about the 'optimal' point found in step 4. A local search using the original cost function and model is performed about p_i. This will result in a candidate parameter set we will denote as p_{il},

with a cost function of cf_{pil}. *Specific:* run `fminunc` from p_i, calling the cost function file.

6. The objective of step 6 is to identify an alternative candidate for the 'optimal' parameter set by starting from the support node with the lowest cost function value, which we denote as p_b, which has a cost function of cf_{pb}. Load the data file containing the best grid point, and, if the point differs from the returned optimal point, run a local search using the original cost function about p_b. This will result in a candidate parameter set we will denote as p_{bl} with a cost function of cf_{pbl}. *Specific:* load the data file `pb.mat` and run `fminunc` from the point, p_b, calling the cost function file.

7. The parameter set from these two local searches with the lowest cost function value is considered the optimal parameter set: $p^* = \begin{cases} p_{il} & cf_{pil} < cf_{pbl} \\ p_{bl} & cf_{pbl} < cf_{pil} \end{cases}$.

8. Examine the resulting simulation for consistency and feasibility (i.e. both quantitative and qualitative fit with experimental data). The objective of this final step is to confirm that minimizing the cost function resulted in an acceptable fit to the experimental data. If the fit is acceptable, the optimization process is complete.

9. The objective of step 9 is to search in other areas of the cost function space with low values, besides the one containing the best support node, if step 8 is not successful. Determine the distance between the sorted grid points with the lowest cost function values (for instance, the lowest 1%), where distance can be defined as the sum of the absolute percent change in each parameter over all parameters. Run local searches on the cost function from the points with the farthest distance from the best support node. If one of these searches results in an acceptable fit, the optimization process is complete.

10. If step 9 is unsuccessful, consult the Troubleshooting section and consider increasing the number of maximum grid points by two to ten times and return to step 2.

Data Acquisition, Anticipated Results and Interpretation

The anticipated result of the method described above is a parameter set that acceptably fits the available experimental data. Whether or not the returned parameter set is adequate, the function `spvals` of the Sparse Grid toolbox [10] returns a structure containing a significant amount of information that may be helpful for understanding the returned 'optimal' set of parameter values as well as information about the cost function values mapped onto the uncertain parameter space. This information includes the number and locations of the grid points, the cost function values at those points, the minimum and maximum cost function values, the degree of adaptivity, estimated errors, and the computational time. From this information, it takes only a few extra steps to extract other useful information, namely the sorted grid points and unique points.

Sorted grid points

Grid points sorted from lowest to highest corresponding cost function value are returned in step 3 of the Methods (see Figure 5). Reviewing these points as a sorted list can provide some insight. For instance, this method can reveal disparate, equally valid areas of the uncertain parameter space, as shown in Figure 6. In this example, three parameters of a MAPK model [17] were fitted to an incomplete data set, consisting only of the MAPK data (red in Figure 6B). The resulting three-dimensional grid of the cost function values on the parameter space indicated two

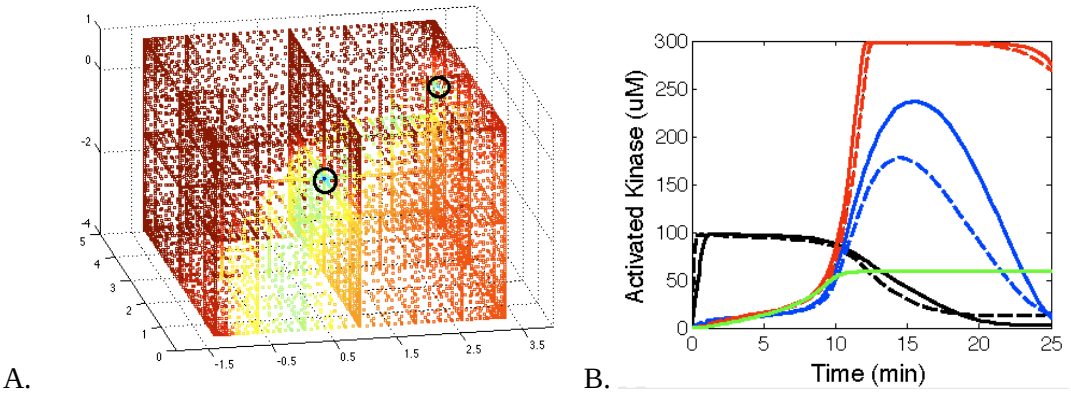

Figure 6: An example of a three-dimensional search of a MAPK model [17] that revealed two parameter sets that fit the mock data equally well, but predicted different dynamics for another model element. A. Three-dimensional adaptive grid, generated with the Sparse Grid toolbox [10] and color-coded by cost function (red: high, blue: low). The two, circled areas have similar cost functions when only the mock MAPK data is fitted. B. Simulation results with parameter sets from each 'optimal' area (solid: center area, dotted: right area). While three of the four state simulations (red: MAPK, green: Raf, black: Rkip) are similar, different MAPKK (blue) dynamics are predicted, suggesting that MAPKK data would be required to distinguish between the two parameter sets.

disparate regions with equally low cost functions (circled in Figure 6A). Simulations with a set of parameter values from each space (Figure 6B) were nearly identical for three of the four states; however, the simulation of MAPKK (blue) showed a distinct difference in its peak. This information suggests that in order to determine which, if either, of the parameter sets are valid, experimental data for the MAPKK, particularly at the 15-minute time point, is required. The sorted grid points can be used to determine the size of the parameter space that results in acceptable dynamics, termed the acceptable space, which can reveal properties of the model, such as the amount of confidence that can be placed in the chosen parameter values [26, 27]. In addition to the above example, as noted in Methods, step 9, in the event that the returned parameters were not adequate, the sorted grid points can provide alternative starting points for additional local or global optimizations that can refine the solution.

Unique points

With adaptive sparse grids, the number of unique points is the number of distinct locations of grid points along a parameter direction. This value can be obtained by applying the MATLAB® `unique` function to a parameter's grid points (see Figure 5, step 3). For three or fewer dimensions, the number of unique points can be seen by plotting the grid, as shown in Figure 7. Unique points correlate with each parameter's importance to increasing the accuracy of the interpolant. This information is valuable because it demonstrates which parameters required the highest resolution for the interpolation. A use for unique points in aiding the optimization process is described in Troubleshooting under 'Large problems' and demonstrated in the Application Notes.

Unstable points

In the process of creating the sparse grid, the algorithm may return, or end with, integration errors when the model is integrated with particular parameter sets. In some cases, this

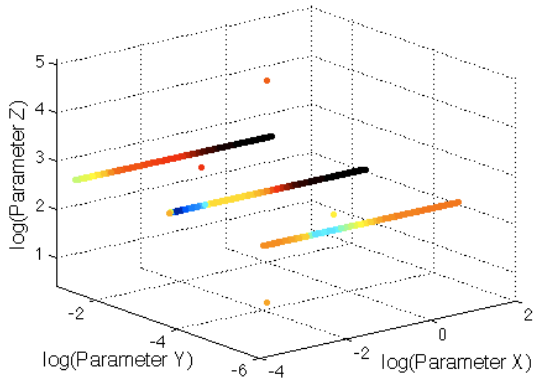

Figure 7: This adaptive sparse grid (generated with the Sparse Grid toolbox [10]) of a three-parameter search of a MAPK model [17] demonstrates the concept of unique points. The parameter on the z axis has three unique points, the y parameter has five, and the x parameter has 513. The points are color-coded by cost function (black and red: high, blue: low).

error may be due to improper range setting (see Troubleshooting) for certain parameters. For instance, a certain search range could allow a parameter to be a value that results in a division by zero. These unstable points should be carefully evaluated as they often reveal weaknesses in the model structure that may need revision to ensure the model is stable over the allowable parameter ranges.

Interpretation and Conclusions

As stated above, if the optimization process is successful, one can conclude that the parameter values found are adequate for fitting the model to experimental data. However, one cannot conclude that these values are physically correct or even unique. If the process is unsuccessful, one should examine the model structure to determine whether or not it is capable of recreating experimental data. One method for examining model structure is a parameter sensitivity analysis. Conducting a sensitivity analysis not only quantifies the sensitivity of the output with regards to the model parameter values but also provides information for directing parameter fitting [21, 22]. The output of a sensitivity analysis helps to identify dominant processes or elements and recognize events/elements that can be considered negligible, including parameters whose values have little impact on fitting the experimental data [21, 22].

Troubleshooting

For the cases where the methods described above do not result in a parameter set that allows the model simulations to acceptably fit the experimental data, Table 2 lists some suggested approaches to deal with common issues. In addition to these general recommendations, there are two special cases that are also considered: small (less than or equal

Table 2: Common issues with adaptive sparse grid-based optimization and suggested solutions

Issue	Suggestion(s)
Method takes too long	Decrease the number of maximum grid points or the number of evaluations allowed by the local search algorithm.
Method returns integration errors	Examine parameter ranges: typically, certain parameters cannot be zero. Have parameter values automatically recorded when integration errors occur and then examine them to determine areas of instability. As appropriate, alter parameter ranges to avoid these areas or modify the model structure to eliminate the problem. Do not artificially set the cost function to an arbitrarily high value when these points occur, as this will interfere with the adaptive algorithm and with the interpolation.
Parameter sets with low cost function values do not result in a fit to the data	Redesign the cost function to more accurately reflect the data. Consider changing the weights of the LSE cost function or adding qualitative goals to the cost function.
Method returns lower or upper bounds for some parameters	Expand the ranges of these parameters beyond the boundaries, if possible.
Method does not produce an acceptable fit	Increase maximum number of allowable support nodes.
	Decrease the problem dimension. Run a global sensitivity analysis, such as extended FAST [28]. Fix the least sensitive parameters at the best guess and search for the remainder.
	Consider an alternative model structure: the current structure may be incapable of producing the desired dynamics.

to three uncertain parameters) and large (10 or more uncertain parameters) problems, since certain trouble-shooting techniques are more helpful, or only applicable, to problems with specific dimensions of the uncertain parameter space.

Trouble-shooting special cases: small and large problems

Small problem: three or fewer parameters

1. The objective of this step is to make use of the ability to visually inspect the cost function in the parameter space when the space has three or fewer dimensions. The cost function itself can be visualized using a mesh for two-dimensional problems and a plot

for one-dimensional problems, and the grid can be visualized for one to three dimensional spaces using a scatter plot. Visually examine the sparse grid (in the one, two, or three dimension case) and the cost function plot (in the one or two dimension case):

a. Sparse grid: Evaluate the interpolated function at each of the grid points and then plot the results. *Specific:* use `scatter` or `scatter3`, with the color of each point corresponding to the cost function value at the point.
b. Create a one or two –dimensional mapping of the cost function. *Specific*: use `plot` or `mesh` as appropriate.

2. The objective of this step is to analyze the plots from step 1 to determine whether or not an appropriate search space was used. For instance, in the example of a two-parameter search of a MAPK model [17] shown in Figure 8 A-C, it can be seen that the optimal point is beyond the lower bounds for both parameters. Both the mesh and the grid suggest that the ranges should be shifted down. In the next search, the ranges were corrected and the results are shown in Figure 8 D-F. The improvement in fit can be seen in the simulation results (Figure 8D). Therefore, for this step, analyze the plots and update the parameter ranges as needed. If nothing can be concluded from the plots, refer to the Troubleshooting table.

Large problem: 10 or more parameters

This Troubleshooting solution takes advantage of the information contained in the number of unique points per parameter, which is calculated in the general procedure (see step 3 of Methods

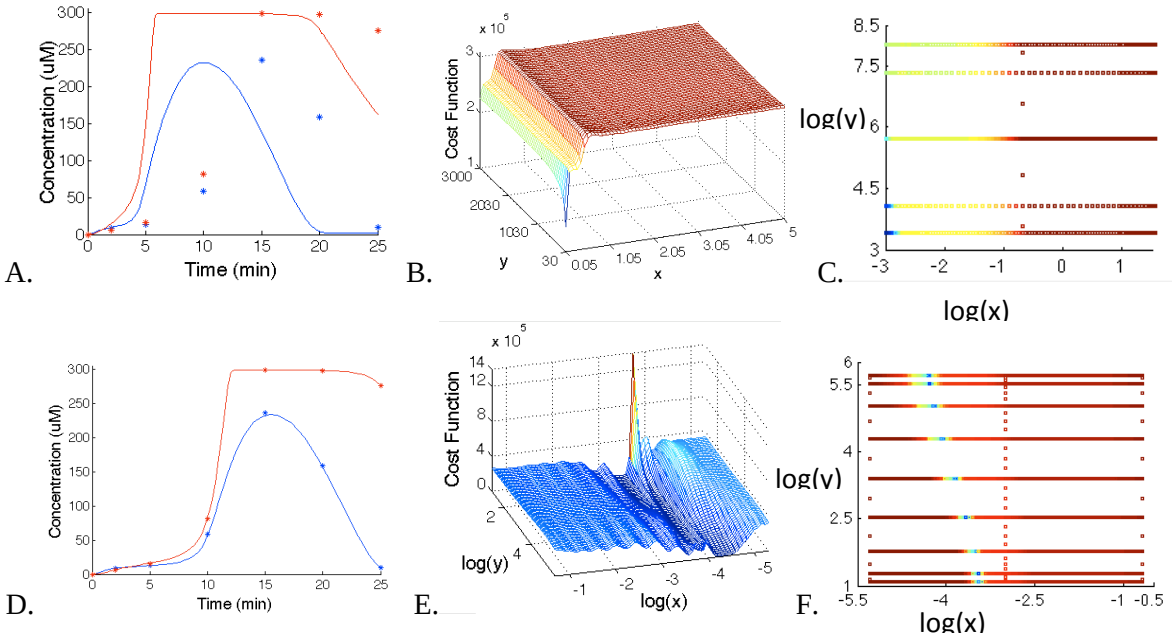

Figure 8: An example of a small (two-dimensional) parameter search of a MAPK model [17]. In this example, the initial search range did not include the optimal parameter values to match the mock data (blue and red stars). A. The fit for the returned parameters (solid lines) compared to the mock data (stars) over the initial search range. B. The mesh of the corresponding cost function. C. The adaptive sparse grid, generated with the Sparse Grid toolbox [10] and color-coded by cost function (red: high, blue: low). The search range is shifted lower, based on the mesh and grid from the original search (B and C). D. The fit for the returned parameters (solid lines) compared to the mock data (dots) with the shifted search range. E. The mesh of the corresponding cost function. F. The corresponding adaptive sparse grid, generated with the Sparse Grid toolbox [10] and color-coded by cost function (red: high, blue: low).

and Figure 5). Typically, these extra steps using the number of unique points are not necessary for smaller-sized problems. Optimization issues with smaller problems are more commonly due to an issue described in Table 2. For a definition and description of unique points see Data Acquisition, Anticipated Results and Interpretation. In brief, the number of unique points for a parameter is the number of unique locations of grid points for that parameter and correlates with the importance of the parameter to decrease the interpolation error. Parameters with the lowest number of unique points are the least important (or can easily be fit with low-dimensional basis functions, such as cubic polynomials).

1. This step assumes that, at a minimum, steps 1-9 of the Methods have been completed and not resulted in an acceptable fit of the model simulations to the data. Therefore, for

the parameters with lowest number of unique points, set their values to the corresponding values of the best grid point, p_b, returned by step 6. With the Sparse Grid toolbox [10], the lowest possible number of unique points is three when using Chebyshev polynomial basis functions, as the lowest interpolation depth allowed is three.

2. Start a new, lower dimension adaptive grid search for the remainder of the parameters, centering the ranges on the corresponding values of p_b and following the steps in Methods. If the dimension of the new problem is three or fewer parameters, then examine the resulting grid for range appropriateness as in 'Small problem' (if necessary, repeat the search with adjusted ranges). Save the returned best grid point, which we will denote p_b^{new}

3. Create a new initial point by replacing the appropriate values of p_b with the appropriate values of p_b^{new}, this new initial point we will denote $p_b^{initial}$. Perform a local search on the cost function starting from $p_b^{initial}$, resulting in the parameter set p_{bl} with a cost function value of cf_{bl}.

4. If the fit is acceptable, the optimization is completed. If not, try increasing the maximum number of grid points by two to ten times and repeating the procedure.

Discussion and Commentary

High-dimensional nonlinear models are becoming common in biomedical applications because of their usefulness in understanding biological processes, predicting behaviors, and developing therapies. However, identifying appropriate model parameters is challenging. As

parameters are typically unknown, they are most often fitted to limited experimental data. Parameter optimization is a well-researched field, and many algorithms exist, including local or global and stochastic or deterministic. Local algorithms are of little use for nonlinear models since their results are highly dependent on the starting location. Global algorithms are computationally expensive and typically have no guarantee of finding, or even converging to, the global minimum [29]. Exceptions exist for smooth, twice-differentiable functions that are convex/concave or can be converted into convex/concave problems can be solved with global deterministic approaches such as branch and bound [7, 8]. However, it is highly unlikely for a cost function based on numerical integration of large, highly nonlinear models of cellular processes to be of that form. Until recently, the alternatives have been global, stochastic algorithms such as the genetic algorithm, which have no guarantee of convergence, or LHS/full grid initialization of local searches, which grow exponentially with dimension.

The alternative presented in this chapter employs an adaptive sparse grid-based search. The adaptive sparse grid is designed map the cost function onto the uncertain parameter space using interpolation with basis functions (typically polynomials) at support nodes. The error between the interpolating function and the cost function decreases with increased numbers of support nodes. This method has two benefits: grid points are placed in the most important locations (importance being defined by [30] as requiring higher level polynomials to reduce the error of the interpolating function) and a the interpolant serves as a surrogate cost function. The former is like an informed LHS/full grid: entire dimensions can be mainly neglected if they are easy to fit with low-dimension polynomials, slowing the increase in model evaluations needed with dimension. The second allows searches without additional model evaluations, which, depending on the interpolation accuracy, can find an optimal point very close to the global

minimum. Like the stochastic and local methods, adaptive sparse grid optimization does not guarantee finding the global minimum. However, unlike, those methods, it does return valuable information about the uncertain parameter space, as described in Data Acquisition, Interpretation, and Conclusions. For instance, the unique points give an indication of parameter importance and can be used to improve the adaptive sparse grid search. In addition, as shown in Applications Notes, adaptive sparse grid-based optimization can result in larger, more consistent decreases in cost function values with increased numbers of model evaluations than the GA, even when followed by a local search. One current limitation of the Sparse Grid toolbox is that the software does not allow extending previously created grids, which would be useful in cases where the initial attempt did not result in an acceptable fit. Future work will create the necessary code to allow an increase in interpolation depth by extending the previously created grid without having to completely start over with a larger maximum number of support nodes.

Complicating factors in optimization searches include parameter correlations (where changes in one parameter can compensate for changes in another) and low parameter sensitivities (where changes in a parameter have little effect on the model output or cost function). As a result, some parameters may not be identifiable from a set of experimental data [22]. The recognition of these parameters can play a key role in parameter identification as they should be neglected, and fixed at some best guess value, until further information can be obtained. Neglecting parameters decreases the dimension of the search, thereby increasing the likelihood of finding the global minimum in the fewest number of model evaluations. In the authors' experience (data not shown), parameters with very low sensitivity coefficients (as determined by extended FAST [28]) are more easily fit with low dimension polynomials; therefore, these parameters will have fewer unique points. However, this correlation will not always be the case

and is certainly not guaranteed for all problems. Future work will explore altering the adaptive scheme for selecting sparse grid points to incorporating information on the parameter sensitivities; this is expected to facilitate the problem of parameter identification.

Applications Notes

The MAPK model published by Wolkenhauer et al [17] was used as an example to demonstrate the described methods. Mock data was generated for two (MAPK and MAPKK) of the four elements (the remaining are Raf and Rkip) by simulating the model with the published parameter values and taking seven time points for each element from zero to 25 minutes. The posed parameter identification problem attempted to identify all 18 model parameters from this mock data set. The results and computational efficiency of the adaptive sparse grid-based optimization method is compared to that of the GA. The sparse grid method, due to symmetry, automatically evaluates the center point of the parameter space. Therefore, in order to avoid biasing the sparse grid towards the actual parameter values, a new center point was created by selecting a random initialization point within an order of magnitude above and below the actual values. The uncertain parameter search range for both the sparse grid and GA was assigned with a lower limit set to an order of magnitude smaller than this initial point and the upper limit set to an order of magnitude larger.

Comparison of adaptive sparse grid and GA based optimization

The resulting cost function value (the least squared error or LSE) was calculated for the adaptive sparse grid-based optimization method and the GA for increasing numbers of model

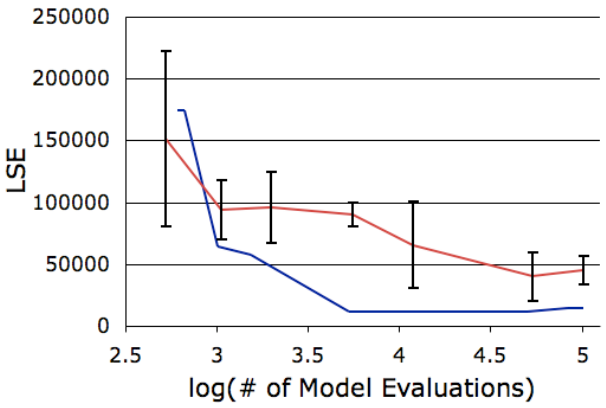

Figure 9: For a MAPK model [17], a comparison of the performance, indicated by the least squared error (LSE) between the model simulations and the mock data set, of the adaptive sparse grid-based optimization (blue) and the GA (red). The adaptive sparse grid method consistently performed better than the GA for larger numbers of model simulations. The GA results are the average of at least five runs, with the error bars representing the standard deviation of the results. The adaptive sparse grid method followed steps 1-7 of Methods while the GA was run with an increasing number of allowed generations and/or population sizes, followed by a local search on the result.

evaluations. The results are shown in Figure 9. For the adaptive sparse grid method, steps 1-7 of the Methods were followed to achieve the resulting cost function value (LSE), with the total number of model evaluations being the sum of the number of grid points and the number of evaluations performed by the local search(es). For the GA method, the GA was run at least five times (because of its stochastic nature, each outcome is different), each followed by a local search. The number of model evaluations in this case is the sum of the evaluations used by the GA and the local searches. The results of the local searches were averaged and the error bars in Figure 9 represent the standard deviations of the results.

To illustrate the differences between the GA and the adaptive sparse grid performance, an example where the maximum number of points for each method was limited to approximately 5000 is given. In addition, the use of the number of unique points per parameter, as described under ‘Large problem’ in the Troubleshooting section, is demonstrated.

Adaptive sparse grid-based optimization

An 18-dimensional, 2017-point grid was created, resulting in an optimal point with a cost function of 1.98E4. Eleven of the 18 parameters were found to have three unique points each, the remaining seven had five to nine each. The parameters that had three unique points were set at the returned values. A second, seven-dimensional grid with 2031 points was created to search over the remaining parameters. The returned optimal point had a cost function of 1.48E4. The values for the 11 parameters returned by the first grid and the seven parameters returned by the second grid were combined into an initial point for a local search on the actual cost function. This local search, using 532 model evaluations, returned an optimal point with a cost function of 1.20E4. The resulting simulations with this parameter set are shown in Figure 10A, which shows that the simulations are slightly shifted from the mock data but otherwise are quite similar and consistent with the observed trends in the mock data. A total number of 4580 model evaluations were used. In order to improve the fit, the next step would be to increase the number of model evaluations by five to ten times.

Genetic algorithm

The GA was run five times and a local search was run from each returned point. With an average of 5517 model evaluations (5000 due to the GA and an average of 517 due to the local search), this method returned an average cost function of 8.57E4, with a standard deviation of 1.33E4. The results are shown in Figure 10B. Figure 9 suggest that the fitting could be improved by increasing the number of model evaluations, but again, the GA would have to be run multiple times in order to have a greater chance of seeing an improved fit.

This 18-dimensional example demonstrates that the adaptive sparse grid-based optimization improves the fit of the model simulations to the experimental data set with increasing numbers of model evaluations while the GA fitness did improve on average but required multiple runs to assure this progress. The example also demonstrates the utility of the unique points identified by the adaptive sparse grid approach; the parameters least sensitive to reducing the error of the interpolant were temporarily fixed while the most important parameters were identified in a subsequent search. This sparse grid process provided information to refine the parameter identification problem to lead to an acceptable solution while the GA failed to identify a reasonable solution even with multiple attempts. This inability of the GA to find acceptable parameter values may be inappropriately interpreted as the modeled hypotheses are inconsistent with the experimental data. However, with the same number of total model evaluations, the adaptive sparse grid-based optimization was able to find parameter values that supported the modeled hypotheses.

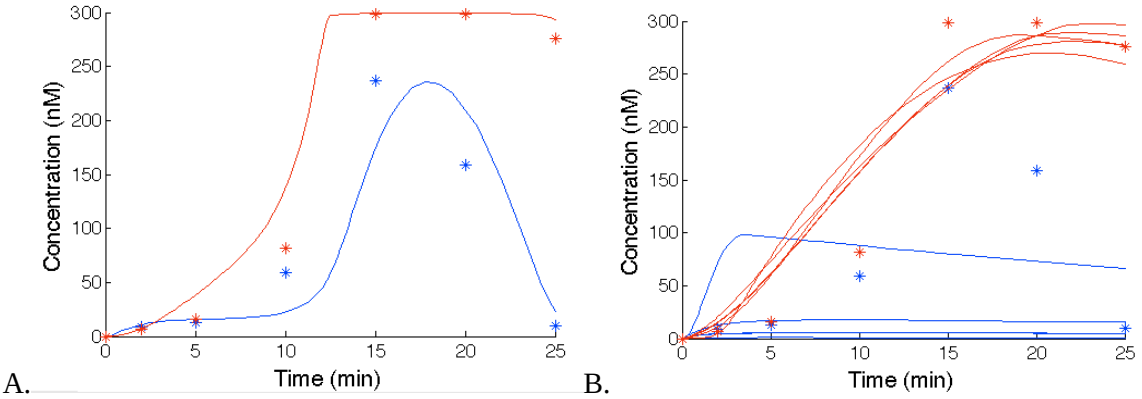

Figure 10: Fitting the 18 parameters of a MAPK model [17] to a mock data set (stars) using approximately 5000 model evaluations. Red: MAPK, Blue: MAPKK. A. Results of adaptive sparse grid-based optimization B. Results of five independent GA runs followed by local searches. The inability of the GA to find acceptable parameter values prematurely suggests the modeled hypotheses may be inconsistent with the experimental data.

Summary Points

The adaptive sparse grid-based optimization approach described herein has a number of advantages over other stochastic global optimization techniques.

- There can be a large variance in the resulting parameter values employing multiple runs of a stochastic optimization approach whereas the adaptive sparse grid-based optimization approach will always return the same parameter values when posed with the same number of maximum model evaluations.
- With an increased number of model evaluations with the adaptive sparse grid-based optimization, the error of the interpolant approximation of the cost function mapping on the parameter space decreases, so eventually the parameter values returned will minimize the cost function value, while the probabilistic sampling of the parameter space by stochastic optimization methods does not provide any assurances of an improvement of the solution with more supporting model evaluations.
- The interpolant mapping of the cost function on the uncertain parameter space and the unique points generated during the adaptive sparse grid search may provide insight to refine and improve the identification process.
- While the GA method can lead to incorrectly discarding a model hypothesis due to its inability to find well-fitting model parameters, adaptive sparse grid-based optimization allows a more thorough examination of the parameter space for a better evaluation of the appropriateness of the model structure.

Acknowledgments

This work supported in part by a National Science Foundation Graduate Research Fellowship.

References

[1] H. Dharuri, L. Endler, N. Le Novere, R. Machne, B. Shapiro, C. Li, C. Laibe, and N. Rodriguez, "http://www.ebi.ac.uk/biomodels/," 2006-2008. Database of Systems Biology Markup Language models. Accessed 6/17/08

[2] R. J. Orton, "http://www.brc.dcs.gla.ac.uk/projects/bps/links.html." Compilation of useful modeling links including model databases. Accessed 6/17/08.

[3] B. B. Aldridge, J. M. Burke, D. A. Lauffenburger, and P. K. Sorger, "Physicochemical modelling of cell signalling pathways," *Nature Cell Biology*, vol. 8, pp. 1195-1203, Nov 2006.

[4] O. Wolkenhauer and M. Mesarovic, "Feedback dynamics and cell function: Why systems biology is called Systems Biology," *Mol Biosyst*, vol. 1, pp. 14-6, May 2005.

[5] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, pp. 523-9, Apr 22 2005.

[6] R. S. Balaban, "Modeling mitochondrial function," *Am j Physiol Cell Physiol*, vol. 291, pp. 1107-1113, 2006.

[7] W. R. Esposito and P. W. Zandstra, "Global Optimization for the Parameter Estimation of Differential Algebraic Systems.," *Ind. Eng. Chem. Res.*, vol. 39, pp. 1291 -1310, 2000.

[8] P. Pardalos, H. E. Romeijn, and H. Tuy, "Recent developments and trends in global optimization," *Journal of Computational and Applied Mathematics*, vol. 124, pp. 209-228, 2000.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*: Addison-Wesley, 1989.

[10] A. Klimke and B. Wohlmuth, "Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB," *ACM Transactions on Mathematical Software*, vol. 31, 2005.

[11] H.-J. Bungartz and S. Dirnstorfer, "Higher Order Quadrature on Sparse Grids," *ICCS 2004*, pp. 394-401, 2004.

[12] I. Ferenczi, "Global Optimization using Sparse Grids," Technische Unversitat Munchen, 2005, p. 140.

[13] A. Klimke, "Sparse grid surrogate functions for nonlinear systems with parameter uncertainty," in *Proceedings of the 1st International Conference on Uncertainty in Structural Dynamics*, 2007, pp. 159-168.

[14] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numerical Algorithms*, vol. 18, pp. 209-232, 1998.

[15] V. Barthelmann, E. Novak, and K. Ritter, "High dimensional polynomial interpolation on sparse grids " *Advances in Computational Mathematics*, vol. 12, pp. 213-288, 2000.

[16] H.-J. Bungartz and M. Griebel, "Sparse Grids," *Acta Numerica*, vol. 13, pp. 147-296, 2004.

[17] O. Wolkenhauer, S. N. Sreenath, P. Wellstead, M. Ullah, and K. H. Cho, "A systems- and signal-oriented approach to intracellular dynamics," *Biochem Soc Trans*, vol. 33, pp. 507-15, Jun 2005.

[18] T. Gerstner and M. Griebel, "Dimension-Adaptive Tensor-Product Quadrature," *Computing*, vol. 71, pp. 65-87, 2003.

[19] A. Klimke, "Uncertainty Modeling using Fuzzy Arithmetic and Sparse Grids," Universität Stuttgart, 2006.

[20] Y. Zheng and A. Rundell, "Comparative Study of Parameter Sensitivity Analyses of the TCR-activated Erk-MAPK Signaling Pathway," *IEE Systems Biology*, vol. In Press, 2006.

[21] M. M. Donahue, W. Zhang, M. Harrison, J. Hu, and A. E. Rundell, "Employing Optimization and Sensitivity Analyses Tools to Generate and Analyze Mathematical Models of T cell Signaling Events," in *Data Mining, Systems Analysis and Optimization in Biomedicine*, Gainesville, FL, 2007, pp. 43-63.

[22] H. Yue, M. Brown, J. Knowles, H. Wang, D. S. Broomhead, and D. B. Kell, "Insights into the behaviour of systems biology models from dynamic sensitivity and identifiability analysis: a case study of an NF-kappaB signalling pathway," *Mol Biosyst*, vol. 2, pp. 640-9, Dec 2006.

[23] D. B. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *Siam Journal on Scientific Computing*, vol. 27, pp. 1118-1139, 2005.

[24] O. P. Le Maitre, H. Najm, P. Pebay, R. Ghanem, and O. M. Knio, "Multi-Resolution-Analysis for Uncertainty Quantification in Chemical Systems," *SIAM J. Sci. Comput.*, vol. 19, pp. 864-889, 2007.
[25] O. Wolkenhauer, M. Ullah, P. Wellstead, and K. H. Cho, "The dynamic systems approach to control and regulation of intracellular networks," *FEBS Lett*, vol. 579, pp. 1846-53, Mar 21 2005.
[26] K. S. Brown and J. P. Sethna, "Statistical mechanical approaches to models with many poorly known parameters " *Physical Review E*, vol. 68, 2003.
[27] R. H. Gutenkunst, F. P. Casey, J. J. Waterfall, C. R. Myers, and J. P. Sethna, "Extracting falsifiable predictions from sloppy models," *Ann. N.Y. Acad. Sci.*, vol. 1115, pp. 203-211, 2007.
[28] A. Saltelli, S. Tarantola, and K. P.-S. Chan, "A quantitative model-independent method for global sensitivity analysis of model output," *Technometrics*, vol. 41, 1999.
[29] C. G. Moles, P. Mendes, and J. R. Banga, "Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods," *Genome Res*, vol. 2003, pp. 2467-2474, 2003.
[30] H.-J. Bungartz and S. Dirnstorfer, "Multivariate Quadrature on Adaptive Sparse Grids," *Computing*, vol. 71, pp. 89-114, 2003.

Related sources and Supplementary Information

The Sparse Grid toolbox for MATLAB® used in this chapter is available at:
http://www.ians.uni-stuttgart.de/spinterp/