Application of MatLab and Python Optimizers to Two Case-Studies Involving Groundwater Flow and Contaminant Transport Modeling

L. Shawn Matott[a,*] Kenny Leung[b], and Junyoung Sim[b]

[a] Assistant Professor, [b] Undergraduate Research Intern
Department of Civil and Environmental Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada, N2L 3G1

[*] Correspondent. Lsmatott@uwaterloo.ca, ph.: 519-888-4567 ext. 38807, fax: 519-888-4349

Abstract

One approach for utilizing geoscience models for management or policy-analysis is via a simulation-based optimization framework – where an underlying model is linked with an optimization search algorithm. In this regard, MATLAB and Python are high-level programming languages that implement numerous optimization routines, including gradient-based, heuristic, and direct-search optimizers. The ever-expanding number of available algorithms makes it challenging for practitioners to identify optimizers that deliver good performance when applied to problems of interest. Thus, the primary contribution of this paper is to present a series of numerical experiments that investigated the performance of various MATLAB and Python optimizers. The experiments considered two simulation-based optimization case-studies involving groundwater flow and contaminant transport. One case-study examined the design of a pump-and-treat system for groundwater remediation, while the other considered least-squares calibration of a model of strontium (Sr) transport. Using these case-studies, the performance of twelve different MATLAB and Python optimizers was compared.

Overall, the Hooke-Jeeves direct search algorithm yielded the best performance in terms of identifying least-cost and best-fit solutions to the design and calibration problems, respectively. The IFFCO (Implicit Filtering for Constrained Optimization)

direct search algorithm and the dynamically dimensioned search (DDS) heuristic algorithm also consistently yielded good performance and were up to 80% more efficient than Hooke-Jeeves when applied to the pump-and-treat problem. These results provide empirical evidence that, relative to gradient- and population-based alternatives, direct search algorithms and heuristic variants, such as DDS, are good choices for application to simulation-based optimization problems involving groundwater management.

Keywords: reactive transport; pump-and-treat; simulation-based optimization; model calibration; MatLab; Python

1. Introduction

A common approach for utilizing geoscience models in a management or policy-analysis context is to incorporate them into a simulation-based optimization framework – where an underlying process-based geoscience model is linked with an optimization search algorithm (Banta *et al.*, 2008; Freeze and Gorelick, 1999). During optimization, a given search algorithm iteratively adjusts various model inputs (i.e. parameters or design variables) in order to minimize an application-specific cost function computed on the basis of model outputs (i.e. response variables). The fundamental output of an optimization algorithm is an 'optimal' configuration of design variables along with the corresponding value of the cost function.

Numerous optimization search algorithms have been developed and may be classified according to a variety of alternative taxonomies (e.g. Dréo *et al.*, 2006; Talbi, 2002). For this paper, it is convenient to informally divide the spectrum of optimizers into four categories: (1) gradient-based algorithms that use derivative information to guide the search; (2) heuristic algorithms that do not utilize derivative information but instead

52 follow empirical guidelines and incorporate elements of structured randomness; (3) direct
53 search algorithms that are derivative-free but utilize deterministic stencil-based
54 procedures; and (4) hybrid methods that combine aspects of the previous three categories.
55 Example applications of simulation-based optimization in a geoscience context
56 include the design of: pump-and-treat systems (McKinney and Lin, 1996), groundwater
57 supply systems (Chu and Chang, 2010), landfill liners (Bartelt-Hunt *et al.*, 2006),
58 agricultural operations (Gitau *et al.*, 2006), waste load allocation strategies (Burn and
59 Yulianti, 2001), and geothermal reservoirs (Akin *et al.*, 2010). In addition, calibration of
60 geoscience models is typically formulated as a simulation-based optimization problem –
61 where uncertain model parameters are adjusted to minimize differences between
62 simulated outputs and corresponding observational data (Massoudieh *et al.*, 2008).
63 For a given application, researchers have generally used optimization routines that
64 are embedded within the code of the underlying geoscience model (e.g. Akin *et al.*, 2010;
65 Lim *et al.*, 2010). However, various efforts by commercial entities, research labs,
66 academics, and independent enthusiasts have led to the development of numerous
67 general-purpose optimization libraries using the MATLAB and Python programming
68 languages (Dahl and Vandenberghe, 2008; Hart, 2009; Kelley, 1999; Venkataraman,
69 2009). These libraries can potentially eliminate the need to implement embedded
70 optimization codes if the associated external optimizers are sufficiently robust and deliver
71 good performance when applied to problems of interest.

72 **1.1. Research Objectives**

73 To assist practitioners with identifying "best-in-class" MATLAB and Python
74 optimizers for a given simulation-based optimization problem, a series of numerical

experiments were performed involving two case-studies relevant to the geosciences community and a suite of twelve MATLAB and Python optimizers. The case studies consider optimization from two distinctly different contexts – a pump-and-treat case-study emphasizes least-cost remedial system design, while a subsurface strontium transport application considers the best-fit calibration of model parameters.

The remainder of the paper is organized as follows: Section 2 describes the selected case-study simulation-based optimization problems; Section 3 describes the optimizers that were selected for consideration in this study along with the associated numerical experiments; Section 4 presents the results of the numerical experiments and compares the performance of the selected optimizers; and Section 5 provides some concluding remarks.

2. Case-Study Simulation-based Optimization Problems

To evaluate the usefulness of MATLAB and Python optimizers in a geosciences context, selected algorithms were applied to two case-study simulation-based optimization problems involving groundwater flow and contaminant transport. The first case-study (i.e. "the calibration problem") considers the calibration of a subsurface reactive transport model where the primary solute of interest is strontium. The second case-study (i.e. "the pump-and-treat problem") involves the design of a pump-and-treat containment system for a site containing contaminated groundwater. Additional case-study details are given below.

2.1. The Calibration Problem

The calibration benchmark problem is motivated by the use of a natural zeolite permeable reactive barrier (PRB) to remediate groundwater contaminated with strontium

90 at the West Valley Demonstration Project site in western New York State (Rabideau *et al.*, 2005). To assess the effectiveness of the PRB, a one-dimensional reactive transport model was developed using the MOUSER (MOderately USEr-friendly Reactive transport model) software (Rabideau, 2003). MOUSER is a multi-purpose finite-element reactive transport model which uses an operator splitting algorithm in which the nodal reaction equations are solved sequentially over each time step. In addition to simulating advective-dispersive transport, MOUSER incorporates competitive cation exchange reaction processes; wherein the Gaines-Thomas convention is used for sorbed phase activity (Gaines and Thomas, 1953). Data from laboratory column experiments served as observations for a calibration exercise in which the values of four species selectivity coefficients (C_k, C_{Mg}, C_{Ca}, and C_{Sr}) were estimated by minimizing the weighted sum of squared residuals (WSSR) between observed data and corresponding MOUSER-generated outputs. In the WSSR formulation, data uncertainty may be accommodated via an appropriate assignment of residual weights (Poeter and Hill, 1999). As shown in Figure 1, sampling of the column effluent yielded multiple observations of Na (the reference ion exchange species), K, Mg, Ca, and Sr.

[Figure 1 goes about here]

2.2. The Pump-and-Treat Problem

The pump-and-treat benchmark problem involves a hypothetical groundwater contamination scenario based on the Lockwood Solvent Groundwater Plume Site (LSGPS) located near Billings, Montana (Tetra Tech Inc., 2003). At the site, industrial practices have resulted in the development of two separate plumes containing chlorinated solvents. As illustrated in Figure 2, the plumes (i.e. Plume A in the south and Plume B in

121 the north) are migrating toward Yellowstone River and six pump-and-treat wells (2 for
122 Plume A and 4 for Plume B) have been proposed to prevent further expansion.
123 Therefore, optimization objectives are to successfully contain both plumes using the
124 minimum amount of pumping.

125 **[Figure 2 goes about here]**

126 Alternative configurations of the proposed pump-and-treat system were modeled
127 using the Bluebird (Craig and Matott, 2005) engine, which uses a high-order
128 implementation (Jankovic and Barnes, 1999) of the analytic element method (AEM)
129 (Hunt, 2006) and assumes two-dimensional steady-state flow in a single-layer aquifer.
130 Unlike finite difference and finite element approaches, AEM groundwater models
131 incorporate hydrologic features as distinct model elements rather than distributed over a
132 spatial grid or mesh. Plume containment goals were evaluated using Bluebird's zone
133 budget feature, which measures the simulated amount of groundwater flux exiting each
134 plume boundary. Following Matott *et al.* (2006), the exit flux was required to be
135 identically zero or else a penalty was assessed and incorporated into the cost function
136 using the additive penalty method (APM) (Hilton and Culver, 2000).

137 **3. Numerical Experiments**

138 A software code known as PIGEON (**P**rogram for **I**nterfacing **G**eoscience models
139 with **E**xternal **O**ptimization routi**N**es) was developed to link the selected Python and
140 MATLAB optimizers with the selected case-study models (i.e. MOUSER and Bluebird,
141 for the calibration pump-and-treat problems, respectively). PIGEON provides users with
142 a human-readable text file interface for configuring the necessary communication
143 between a given combination of geoscience model and external optimizer. The PIGEON

144 software and user-manual is freely available for download from the following web-site:
145 www.civil.uwaterloo.ca/lsmatott/Pigeon/Pigeon.html. Incorporation of PIGEON into
146 other geoscience toolkits, such as HYDROMAD (Hydrological Model Assessment and
147 Development) (Andrews, 2010) and OpenDA (Open-source Data Assimilation) (Verlaan
148 *et al.*, 2010), would increase the choice of optimizers available to users of these packages.
149 The numerical experiments considered a total of twelve optimization routines,
150 corresponding to 6 MATLAB and 6 Python optimizers. The chosen algorithms are
151 briefly described in the following sub-sections. Consisting of four gradient-based
152 optimizers, four direct search optimizers, and four heuristic global optimizers, the
153 selected algorithms represent a manageable cross-section of different approaches.
154 *3.1. Gradient-based Optimizers (GBO)*
155 Two gradient-based optimizers from the MATLAB optimization toolbox were
156 evaluated: *fminunc* and *fminimax*. Both of these routines are highly flexible and will
157 select from a variety of optimization procedures depending on problem characteristics
158 and the settings of various algorithm options. In this regard, the *fminunc* routine was
159 configured to use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Broyden, 1970;
160 Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) algorithm, while the *fminimax* procedure
161 defaulted to using Newton's method (Polyak, 2007) for the selected problems. Both
162 algorithms were instructed to approximate gradients using finite differences.
163 Two gradient-based optimizers from the Python *scipy.optimize* library were also
164 evaluated: namely *fmin_cg* and *fmin_tnc*. The *fmin_cg* routine is an implementation of
165 the Polak-Ribiere Conjugate Gradient (PRCG) algorithm (Grippo and Lucidi, 1997),
166 while the *fmin_tnc* routine implements a Truncated Netwon Code (TNC) (Dixon and

Price, 1988) – an approach that is similar to Newton's method but uses coarser estimates of the optimal search direction to improve efficiency. As with the MATLAB gradient-based optimizers, both *fmin_cg* and *fmin_tnc* were instructed to approximate gradients using finite differences.

3.2. Direct Search Optimizers (DSO)

Two MATLAB routines implementing alternative derivative-free direct search procedures were evaluated: namely, *imfil* (Kelley, 2009) and *fminsearch*. Along with many other optimizers, the *imfil* routine is provided as a companion to Kelley (1999) – the routine is an implementation of Implicit Filtering for Constrained Optimization (IFFCO), a robust stencil-based approach that is designed to tolerate noise (e.g. from numerical errors (Kavetski, 2006)) in the optimization cost function. The *fminsearch* routine is provided with the MATLAB optimization toolbox and is an implementation of the classic Nelder-Mead downhill simplex algorithm (Nelder and Mead, 1965).

Two direct search routines implemented in Python were evaluated, namely *fmin_powell* and *HookeJeeves*. The *fmin_powell* routine implements Powell's classic method of conjugate directions (Powell, 1977) and is provided with the *scipy.optimize* library. The *HookeJeeves* routine is a Python implementation of the Hooke-Jeeves pattern search algorithm (Hooke and Jeeves, 1961) – the Python code was directly translated from a freely available C-language implementation (Amit Saha, personal comm.).

3.3. Heuristic Global Optimizers (HGO)

Two MATLAB heuristic optimization routines were included in the comparison experiments, namely *DDSmincon* and *TurboGA*. The *DDSmincon* routine implements

dynamically dimensioned search (DDS) (Tolson and Shoemaker, 2007), an algorithm that focuses on finding "good-enough" global solutions within a specified maximum computational budget. Designed to mimic the trial-and-error approach commonly employed by practitioners, DDS may be viewed as a kind of stochastic direct search procedure (Tolson, personal communication). The algorithm requires no tuning and the search dimension is dynamically refined as optimization proceeds. The *TurboGA* routine is an implementation of the recently developed 'turbo' Genetic Algorithm (TGA) (Burjorjee, 2010). As a variant of the classic genetic algorithm, TGA emulates the evolutionary processes of gene encoding, natural selection, mutation and crossover while incorporating new procedures that allow for speedier convergence. The DDS and TGA MATLAB codes utilized in this work are freely available from the original developers of these algorithms (i.e. Bryan Tolson for DDS and Keki Burjorjee for TGA).

Two heuristic optimization routines implemented in Python were evaluated, namely *anneal* and *DiffEvolution*. The *anneal* routine is part of the *scipy.optimize* library and is an implementation of Simulated Annealing (SA) (Kirkpatrick *et al.*, 1983) – an algorithm that emulates the process of physical annealing, where a solid is heated to a high energy state and slowly cooled to drive the material to its lowest energy state. The *DiffEvolution* routine is an enthusiasts' (i.e. Amit Saha) freely available implementation of Differential Evolution (DE) (Storn and Price, 1997) – a population-based algorithm that perturbs parameter values using a combination of vector differences and evolutionary operators (i.e. mutation, selection and crossover).

3.4. Numerical Experiments

All 12 of the selected optimizers were applied to each of the benchmark problems. Because heuristic optimizers are stochastic, multiple (i.e. 9) trials of these algorithms were evaluated in order to determine the central tendency of each approach, expressed in terms of the median trial. For each problem, and as summarized in Table 1, the various optimizers were provided with the same initial iterate and, where required, the same lower and upper bounds for design variables.

[Table 1 goes about here]

Most algorithm parameters were allowed to default to the internal settings of the associated Python or MATLAB implementation. Notable exceptions are as follows: (1) DDS, Powell and SA were configured to use budgets of 1000 and 500 evaluations for the pump-and-treat and MOUSER calibration problems, respectively; (2) PRCG was configured to use a maximum of 100 iterations; (3) DE was configured to use population sizes of 20 and 15 and maximum generations of 50 and 33 for the pump-and-treat and MOUSER calibration problems, respectively; (4) initial step-size vectors for the Hooke-Jeeves algorithm were $[5000,5000,5000,5000,5000,5000]^T$ and $[50,500,500,5000]^T$ for the pump-and-treat and MOUSER calibration problems, respectively; (5) for the SA algorithm, a value of 10 was used for the *dwell* parameter and initial temperatures of 5×10^4 and 1×10^6 were used for the pump-and-treat and MOUSER calibration problems, respectively; and (6) the TGA was configured to use a binary gene encoding of 15 bits per parameter, population sizes of 50 and 24, and maximum generations of 19 and 20 for the pump-and-treat and MOUSER calibration problems, respectively. Native MATLAB and Python command-lined interfaces were utilized to assign the aforementioned algorithm settings.

6. Results and Discussion

The performance of the selected algorithms, as applied to the MOUSER calibration problem, is summarized in Table 2. As shown in Table 2, the top 5 optimizers (i.e. Hooke-Jeeves, TNC, IFFCO, DE, and DDS) delivered comparable results (i.e. within 1% of each other) in terms of minimizing the weighted sum of squared residuals (WSSR) cost function. Of these algorithms, the two direct search optimizers (i.e. Hooke-Jeeves and IFFCO) were the most efficient in terms of computational costs (i.e. lowest required number of model evaluations). Although it yielded a low-WSSR solution, the gradient-based truncated Newton code (TNC) was by far the least efficient – requiring between 150% and 350% more evaluations, relative to the other algorithms. Interestingly, the other 3 gradient-based optimizers (PRCG, Newton and BFGS) performed poorly and yielded the highest optimized WSSR values.

[Table 2 goes about here]

Figure 3 plots the convergence history of the various algorithms, as applied to the MOUSER calibration problem and where WSSR values have been converted to equivalent root mean squared error (RMSE) values. The convergence behavior of most of the algorithms is characterized by an initial fast reduction in the cost function over the course of the first 10-20 model evaluations, followed by much more gradual refinements. For example, within the first 10 model evaluations the Hooke-Jeeves algorithm identifies a solution that is only 20% greater than the WSSR value at convergence.

[Figure 3 goes about here]

With respect to the pump-and-treat problem, the results generated by the chosen algorithms are given in Table 3, sorted according to optimized cost values. For this problem, the top 3 algorithms (i.e. Hooke-Jeeves, SA, and DDS) yielded optimized cost

values that are within 7.5% of each other, and the optimized costs of the top 6 algorithms (i.e. Hooke-Jeeves, SA, DDS, DE, IFFCO, and Newton) are within 15% of each other. The Hooke-Jeeves direct search algorithm yielded the least-cost solution, followed by three of the four heuristic optimizers (i.e. simulated annealing, dynamically dimensioned search and differential evolution). Of the top 6 algorithms, the IFFCO, Newton and DDS algorithms are notably more efficient and required between 200 and 700 fewer model evaluations. Unlike the MOUSER calibration problem, where it yielded the second-lowest optimized WSSR value, the truncated Newton code yielded the highest optimized cost when applied to the pump-and-treat problem.

Examination of the optimized pumping rates reported by each algorithm (see Table 3) suggests the presence of multiple nearly equally optimal pumping strategies. However, these strategies are fairly consistent with respect to the required pumping rates for specific wells. For Plume A, the optimal configurations set Q_1 at a low rate and Q_2 at a high rate; and for Plume B, Q_3 is high-rate, Q_4 is medium-rate, and Q_5 and Q_6 are low-rate.

[Table 3 goes about here]

Figure 4 illustrates the convergence behavior of the chosen algorithms, as applied to the pump-and-treat problem. Four of the selected algorithms (i.e. DDS, Hooke-Jeeves, IFFCO and Newton) are characterized by an initial rapid reduction in the cost function, followed by more gradual refinements. Conversely, three of the four heuristic algorithms (i.e. differential evolution, simulated annealing, and Turbo GA) yielded reductions in the cost function that were more evenly distributed over the entire course of the optimization.

The TNC and Nelder-Mead algorithms appear to quickly "stall", possibly becoming entrapped in local minima.

[Figure 4 goes about here]

7. Conclusions

Overall, the results from the two benchmark problems indicate that the Hooke-Jeeves algorithm yielded the best performance in terms of identifying least-cost solutions. For the MOUSER calibration problem the Hooke-Jeeves algorithm was also highly efficient and required only 300 model evaluations. For the pump-and-treat problem, the Hooke-Jeeves algorithm was not as efficient and required 993 model evaluations before converging on a solution. If trading off optimized cost for computational efficiency was acceptable, alternative algorithms such as IFFCO (requiring 79% fewer model evaluations, but yielding a solution that is only 14% more expensive than Hooke-Jeeves) or DDS (requiring 23% fewer evaluations and yielding only a 7% more expensive solution) might be considered as practical alternatives. Both of these algorithms also delivered good performance when applied to the MOUSER calibration problem.

MATLAB and Python have become popular delivery mechanisms for both classical (e.g. Hooke-Jeeves, BFGS, and Simulated Annealing) and up-and-coming (e.g. DDS, IFFCO and the Turbo GA) optimization routines. This paper represents an initial exploration of the suitability of such optimizers for use within the geosciences community. The results lay the groundwork for consideration of additional case-studies extending beyond problems involving subsurface flow and contaminant transport.

From a practical perspective, simulation-based optimization is usually employed within a broader context that seeks to utilize geoscience models in support of policy

304 analysis and decision making. Thus, although this study focused on identifying "best in
305 class" optimization routines, there are a number of additional considerations that need to
306 be addressed to ensure that a given simulation-based optimization effort is appropriate for
307 its intended purpose. For example, sources of uncertainty and variability must be
308 carefully considered using robust methods. Further, calibration efforts must guard against
309 the possibility of over-fitting – where adjusted model parameters yield excellent fits to
310 historical data but nonetheless generate poor predictions.

311 **References**

312 Akin, S., Kok, M. V.,Uraz, I. 2010. Optimization of Well Placement Geothermal Reservoirs
313 using Artificial Intelligence. Computers & Geosciences 36, 776-785.

314 Andrews, F. 2010. HYDROMAD Tutorial. The Australian National University, Canberra,
315 Australia.

316 Banta, E. R., Hill, M. C., Poeter, E., Doherty, J. E.,Babendreier, J. 2008. Building model analysis
317 applications with the Joint Universal Parameter IdenTification and Evaluation of
318 Reliability (JUPITER) API. Computers & Geosciences 34, 310-319.

319 Bartelt-Hunt, S. L., Culver, T. B., Smith, J. A., Matott, L. S.,Rabideau, A. J. 2006. Optimal
320 Design of a Compacted Soil Liner Containing Sorptive Amendments. Journal of
321 Environmental Engineering 132, 769-776.

322 Broyden, C. G. 1970. The Convergence of a Class of Double-rank Minimization Algorithms 1.
323 General Considerations. IMA Journal of Applied Mathematics 6, 76-90.

324 Burjorjee, K. M. 2010. Generative Fixation: A Unified Explanation for the Adaptive Capacity of
325 Simple Recombinative Genetic Algorithms. SIGEVOlution 4, 12-13.

326 Burn, D. H.,Yulianti, J. S. 2001. Waste-Load Allocation Using Genetic Algorithms. Journal of
327 Water Resources Planning and Management 127, 121-129.

328 Chu, H.-J.,Chang, L.-C. 2010. Optimizing Capacity-Expansion Planning of Groundwater Supply
329 System between Cost and Subsidence. Journal of Hydrologic Engineering 15, 632-641.

330 Craig, J. R.,Matott, L. S. 2005. Visual Bluebird User's Manual: Version 2.0. University at
331 Buffalo, Department of Civil, Structural, and Environmental Engineering, Buffalo (NY)
332 (available from www.groundwater.buffalo.edu).

333 Dahl, J.,Vandenberghe, L. 2008. CVXOPT: A Python Package for Convex Optimization.
334 University of California at Los Angeles, Los Angeles, CA (available from
335 http://abel.ee.ucla.edu/cvxopt).

336 Dixon, L. C. W.,Price, R. C. 1988. Numerical Experience with the Truncated Newton Method for
337 Unconstrained Optimization. Journal of Optimization Theory and Applications 56, 245-
338 255.

Dréo, J., Pétrowski, A., Siarry, P.,Taillard, E., eds. 2006. Metaheuristics for Hard Optimization Springer, Heidelberg, Germany.

Fletcher, R. 1970. A New Approach to Variable Metric Algorithms. The Computer Journal 13, 317-322.

Freeze, R. A.,Gorelick, S. M. 1999. Convergence of Stochastic Optimization and Decision Analysis in the Engineering Design of Aquifer Remediation. Ground Water Monitoring & Remediation 37, 934-954.

Gaines, G. L.,Thomas, H. C. 1953. Adsorption studies on clay minerals. II. A formulation of the thermodynamics of exchange adsorption. Journal of Chemical Physics 21, 714-718.

Gitau, M. W., Veith, T. L., Gburek, W. J.,Jarrett, A. R. 2006. Watershed Level Best Management Practice Selection and Placement in the Town Brook Watershed, New York. Journal of the American Water Resources Association 42, 1565-1581.

Goldfarb, D. 1970. A Family of Variable-Metric Methods Derived by Variational Means. Mathematics of Computation 24, 23-26.

Grippo, L.,Lucidi, S. 1997. A Globally Convergent Version of the Polak-Ribiere Conjugate Gradient Method. Mathematical Programming 78, 375-391.

Hart, W. 2009. Python Optimization Modeling Objects (Pyomo). In Operations Research and Cyber-Infrastructure (Chinneck, J. W., Kristjansson, B., and Saltzman, M., eds.), pp. 3-20. Springer, New York, NY.

Hilton, A. B. C.,Culver, T. B. 2000. Constraint Handling for Genetic Algorithms in Optimal Remediation Design. Journal of Water Resources Planning and Management 126, 128-137.

Hooke, R.,Jeeves, T. A. 1961. 'Direct Search' Solution of Numerical and Statistical Problems. Journal of the ACM 8, 212-229.

Hunt, R. J. 2006. Ground Water Modeling Applications Using the Analytic Element Method. Ground Water 44, 5-15.

Jankovic, I.,Barnes, R. 1999. High-order Line Elements in Modeling Two-Dimensional Groundwater Flow. Journal of Hydrology 226, 211-223.

Kavetski, D. 2006. Calibration of Conceptual Hydrological Models Revisited: 1. Overcoming Numerical Artefacts. Journal of Hydrology 320, 173-186.

Kelley, C. T. 1999. Iterative Methods for Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Kelley, C. T. 2009. Users' Guide for imfil Version 0.85. North Carolina State University, Raleigh, NC (available from www4.ncsu.edu/~ctk/matlab_darts.html).

Kirkpatrick, S., Gelatt, C. D., Jr.,Vecchi, M. P. 1983. Optimization by Simulated Annealing. Science 220, 671-680.

Lim, K. J., Park, Y. S., Kim, J., Shin, Y.-C., Kim, N. W., Kim, S. J., Jeon, J.-H.,Engel, B. A. 2010. Development of Genetic Algorithm-Based Optimization Module in WHAT System for Hydrograph Analysis and Model Application. Computers & Geosciences 36, 936-944.

Massoudieh, A., Mathew, A.,Ginn, T. R. 2008. Column and Batch Reactive Transport Experiment Parameter Estimation Using a Genetic Algorithm. Computers & Geosciences 34, 24-34.

Matott, L. S., Rabideau, A. J.,Craig, J. R. 2006. Pump-and-Treat Optimization Using Analytic Element Method Flow Models. Advances in Water Resources 29, 760–775.

McKinney, D. C.,Lin, M.-D. 1996. Pump-and-Treat Ground-Water Remediation System Optimization. Journal of Water Resources Planning and Management 122, 128-136.

Nelder, J. A.,Mead, R. 1965. A Simplex Method for Function Minimization. The Computer Journal 7, 308-313.

Poeter, E. P.,Hill, M. C. 1999. UCODE, a computer code for universal inverse modeling. Computers & Geosciences 25, 457-462.

Polyak, B. T. 2007. Newton's Method and its Use in Optimization. European Journal of Operational Research 181, 1086-1096.

Powell, M. J. D. 1977. Restart Procedures for the Conjugate Gradient Method. Mathematical Programming 12, 241-254.

Rabideau, A. J. 2003. MOUSER Version 1 User's Manual. University at Buffalo, Department of Civil, Structural, and Environmental Engineering, Buffalo, NY (available from www.groundwater.buffalo.edu).

Rabideau, A. J., Van Benschoten, J., Patel, A.,Bandilla, K. 2005. Performance Assessment of a Zeolite Treatment Wall for Removing Sr-90 from Groundwater. Journal of Contaminant Hydrology 79, 1-24.

Shanno, D. F. 1970. Conditioning of Quasi-Newton Methods for Function Minimization. Mathematics of Computation 24, 647-656.

Storn, R.,Price, K. 1997. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341-359.

Talbi, E. G. 2002. A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics 8, 541-564.

Tetra Tech Inc. 2003. Remedial Investigation Report Lockwood Solvent Groundwater Plume Site. Montana Department of Environmental Quality (available from www.epa.gov/region8/superfund/mt/lockwood_solvents/RIreport06-03.pdf), Helena, Montana.

Tolson, B. A.,Shoemaker, C. A. 2007. Dynamically Dimensioned Search Algorithm for Computationally Efficient Watershed Model Calibration. Water Resources Research 43, W01413.

Venkataraman, P. 2009. Applied Optimization with MATLAB Programming. Jon Wiley and Sons, Hoboken, NJ.

Verlaan, M., Velzen, N. v., Hummel, S.,Gerritsen, H. Year. "OpenDA, a generic toolbox for data-assimilation in numerical modelling." Paper presented at the 15th Biennial Conference of the Joint Numerical Sea Modelling Group, Delft, The Netherlands, May 10-12, 2010.

List of Figure and Table Captions

Figure 1: Measured Column Effluent for the MOUSER Calibration Problem [adapted from Rabideau *et al.* (2005)]

Figure 2: Pump-and-Treat Problem Setup

Figure 3: Convergence Behavior of the Selected Algorithms (as applied to the MOUSER calibration problem)

Figure 4: Convergence Behavior of the Selected Algorithms (as applied to the Pump-and-Treat problem)

Table 1: Parameter Bounds and Initial Values for the Benchmark Problems

Table 2: Optimization Results for the MOUSER Calibration Problem

Table 3: Optimization Results for the Pump-and-Treat Problem

Figures

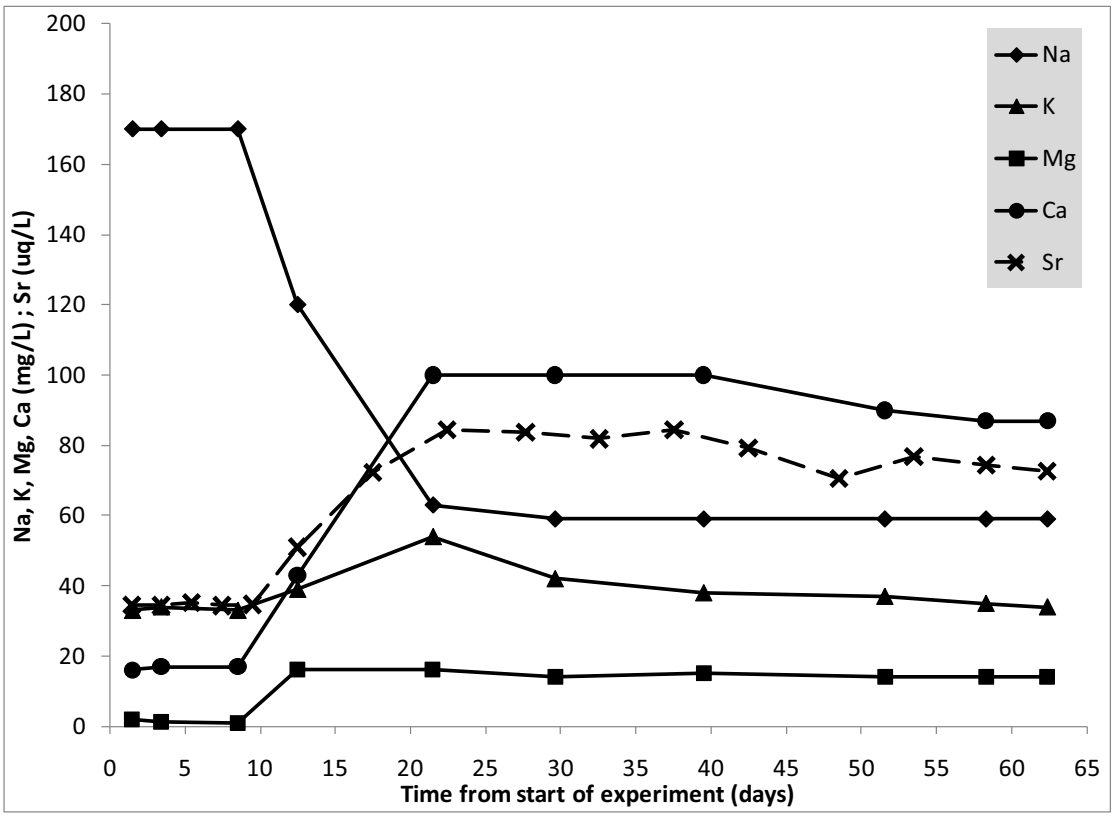

Figure 1: Measured Column Effluent for the MOUSER Calibration Problem [adapted from Rabideau *et al.* (2005)]

Figures (continued)

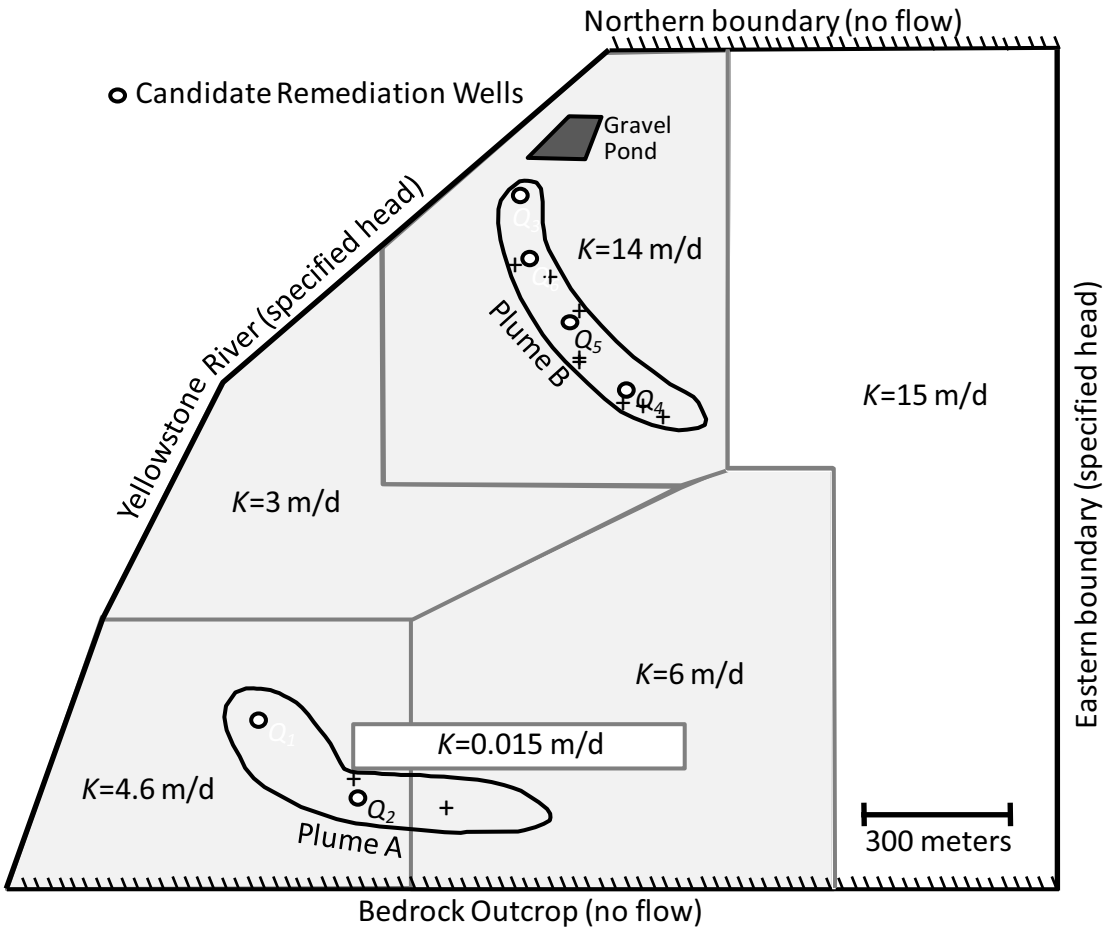

Figure 2: Pump-and-Treat Problem Setup
(*K* values are hydraulic conductivities of various zones of heterogeneity)

Figures (continued)

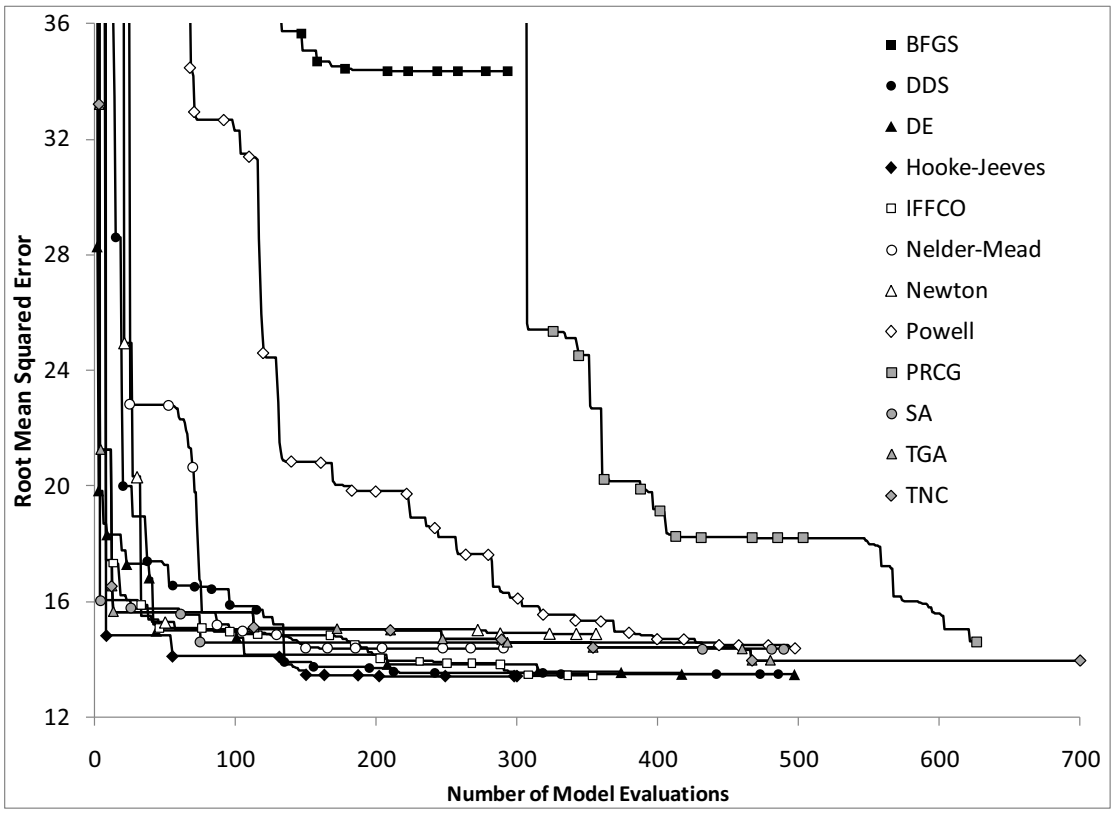

Figure 3: Convergence Behavior of the Selected Algorithms (as applied to the MOUSER calibration problem)

Figures (continued)

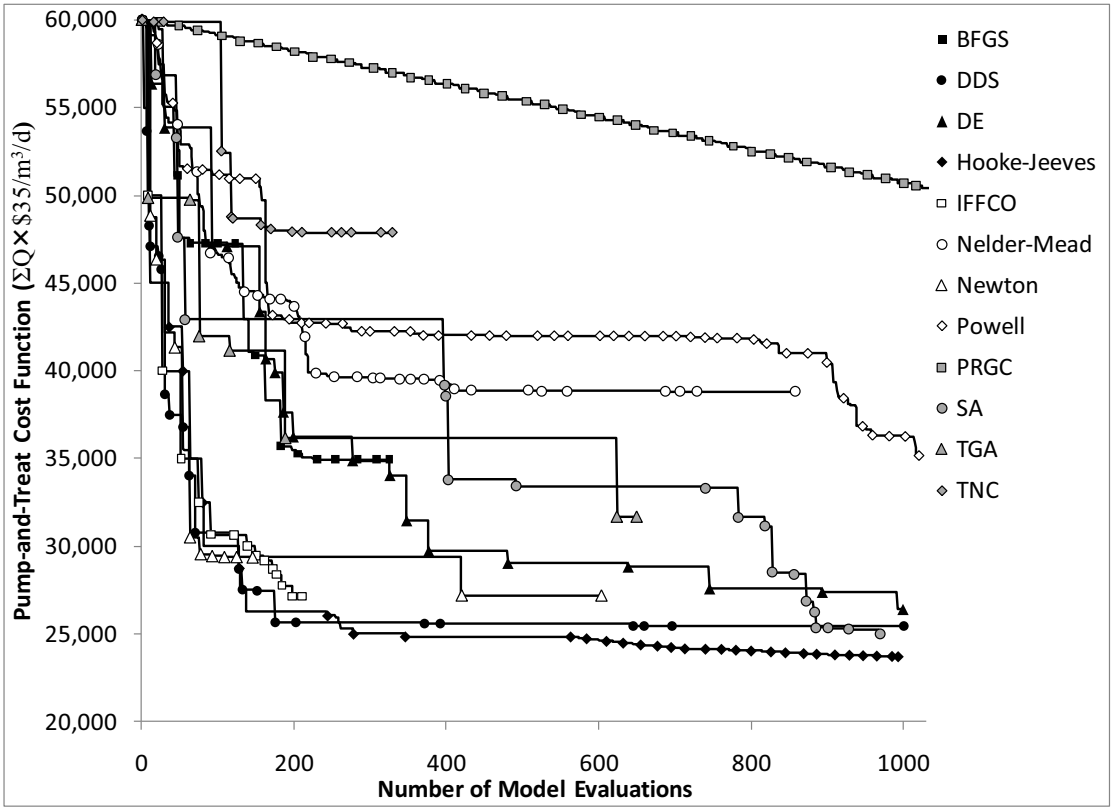

Figure 4: Convergence Behavior of the Selected Algorithms (as applied to the Pump-and-Treat problem)

Tables

Table 1: Parameter Bounds, Initial Values, and Objective Functions for the Benchmark Problems ($_{sim}$ and $_{obs}$ subscripts denote simulated and observed concentrations, respectively; $Q_{out,A}$ and $Q_{out,B}$ are the simulated fluxes exiting the boundaries of Plume A and B, respectively)

(a) MOUSER Calibration Problem				
Parameter	Initial Value	Lower Bound	Upper Bound	units
C_K	80	1	100	n/a
C_{Mg}	950	100	1000	
C_{Ca}	880	100	1000	
C_{Sr}	1290	1000	10000	
$WSSR = \left(\sum_{i=1}^{10} \left[\left(\mathrm{K}_{i,obs} - \mathrm{K}_{i,sim}\right)^2 + \left(\mathrm{Mg}_{i,obs} - \mathrm{Mg}_{i,sim}\right)^2 + \left(\mathrm{Ca}_{i,obs} - \mathrm{Ca}_{i,sim}\right)^2 \right] + \sum_{j=1}^{16} 0.001 \times \left(\mathrm{Sr}_{i,obs} - \mathrm{Sr}_{i,sim}\right)^2 \right)$				

(b) Pump-and-Treat Problem				
Design Variable	Initial Value	Lower Bound	Upper Bound	units
Q_1	283	0.00	566	m^3/day
Q_2	283	0.00	566	
Q_3	283	0.00	566	
Q_4	283	0.00	566	
Q_5	283	0.00	566	
Q_6	283	0.00	566	
$\mathrm{Cost} = \left(\$35 \times \sum_{i=1}^{6} Q_i \right) + \underbrace{\left[\$100000 \times \left(Q_{out,A} + Q_{out,B} \right) \right]}_{\text{penalty function}}$				

Tables (continued)

Table 2: Optimization Results for the MOUSER Calibration Problem

Algorithm [a]	Optimal WSSR	Computational Cost [c]	Optimal Parameters				Engine (routine)	Type [b]
			C_K	C_{Mg}	C_{ca}	C_{sr}		
Hooke-Jeeves	9364	300	28	299	348	4698	Python (HookeJeeves)	DSO
TNC	9366	990	28	302	347	4704	Python (fmin_tnc)	GBO
IFFCO	9366	354	28	294	343	4643	MatLab (imfil)	DSO
DE	9445	497	29	259	355	4655	Python (DiffEvolution)	HGO
DDS	9468	486	30	329	394	5274	MatLab (ddsmincon)	HGO
TGA	10120	480	30	488	475	6173	MatLab (tga)	HGO
SA	10712	481	22	476	348	4717	Python (anneal)	HGO
Nelder-Mead	10758	284	26	898	333	4707	MatLab (fminsearch)	DSO
Powell	10775	498	23	200	220	3098	Python (fmin_powell)	DSO
PRCG	11113	627	25	860	270	3935	Python (fmin_cg)	GBO
Newton	11517	356	37	551	656	8685	MatLab (fminimax)	GBO
BFGS	61304	293	5	939	100	1984	MatLab (fminunc)	GBO

[a] – BFGS: Broyden-Flecther-Goldfarb-Shanno, DDS: Dynamically Dimensioned Seach, DE: Differential Evolution, IFFCO: Implicit Filtering for Constrained Optimization, PRCG: Polak-Robiere Conjugate Gradient, SA: Simulated Annealing, TGA: Turbo Genetic Algorithm, TNC: Truncated Newton Code

[b] – DSO: Direct Search Optimizer, GBO: Gradient-based Optimizer, HGO: Heuristic Global Optimizer

[c] – Number of required model evaluations before convergence on the reported optimal parameter set

Tables (continued)

Table 3: Optimization Results for the Pump-and-Treat Problem

Algorithm	Optimal Cost $\sum_{i=1}^{6} Q_i \times \$35/(\mathrm{m}^3/\mathrm{d})$	Computational Cost [c]	Optimal Pumping Rates (m^3/day)						Engine (routine)	Type
			Plume A		Plume B					
			Q_1	Q_2	Q_3	Q_4	Q_5	Q_6		
Hooke-Jeeves	23714	993	10	170	383	70	22	17	Python (HookeJeeves)	DSO
SA	25018	969	9	196	384	84	17	18	Python (anneal)	HGO
DDS	25441	768	16	169	348	52	23	113	MatLab (ddsmincon)	HGO
DE	26392	992	7	174	369	132	45	21	Python (DiffEvolution)	HGO
IFFCO	27137	210	0	178	317	61	0	213	MatLab (imfil)	DSO
Newton	27193	603	6	155	354	52	70	133	MatLab (fminimax)	GBO
TGA	31705	650	83	176	300	40	100	199	MatLab (tga)	HGO
BFGS	34943	325	10	143	308	79	106	157	MatLab (fminunc)	GBO
Powell	35176	1021	3	158	199	46	177	410	Python (fmin_powell)	DSO
Nelder-Mead	38805	858	258	243	338	27	232	0	MatLab (fminsearch)	DSO
PRCG	47277	1393	223	223	223	223	223	223	Python (fmin_cg)	GBO
TNC	47886	317	272	269	272	272	0	272	Python (fmin_tnc)	GBO

[a] – BFGS: Broyden-Flecther-Goldfarb-Shanno, DDS: Dynamically Dimensioned Seach, DE: Differential Evolution, IFFCO: Implicit Filtering for Constrained Optimization, PRCG: Polak-Robiere Conjugate Gradient, SA: Simulated Annealing, TGA: Turbo Genetic Algorithm, TNC: Truncated Newton Code

[b] – DFO: Direct Search Optimizer, GBO: Gradient-based Optimizer, HGO: Heuristic Global Optimizer

[c] – Number of required model evaluations before convergence on the reported optimal parameter set

Computer Code

Click here to download Computer Code: Computer Code.doc