SOLUTION OF A GROUNDWATER CONTROL PROBLEM WITH IMPLICIT FILTERING *

A. BATTERMANN †, J. M. GABLONSKY‡, A. PATRICK‡, C. T. KELLEY‡, K. R. KAVANAGH‡, T. COFFEY‡, AND C. T. MILLER§

Abstract. In this paper we describe the application of a parallel implementation of the implicit filtering algorithm to a control problem from hydrology. We seek to control the temperature at a group of drinking water wells by placing barrier wells between the drinking water wells and a well that injects heated water from an industrial site.

Key words. Implicit filtering, Groundwater flow and transport, Optimal control, Parallel algorithms

1. Introduction. The objective of this paper is to show how the implicit filtering algorithm [11, 15] for noisy optimization problems can be applied to optimization problems in hydrology. We focus on a groundwater temperature control problem. This problem has some of the important difficulties, such as nonconvexity and nonsmoothness, that one would expect in more difficult cases, but can use flow and transport models and formulations of the optimization problem that are sufficiently simple to allow for a complete description in a single paper. More difficult problems, with coupled flow and transport, temperature dependent densities and viscosities, three dimensional geometries, and more complex flow and transport equations, will be considered in future work.

In this paper, we solve the subsurface flow control problem with a parallel implementation [3] of the implicit filtering algorithm [10, 11, 15]. Implicit filtering is a sampling method for optimization of noisy functions. The problem has simple bound constraints and four optimization variables. The objective function is nonconvex, nonsmooth, and has several local minima. The optimization landscape in Figure 1.1 is a plot of the objective function with two of the variables set to zero.

We begin in § 2 by briefly discussing the groundwater flow and transport models used in this work and by formulating the control problem.

In § 3 we review the implicit filtering algorithm and its implementation in parallel. Then in § 4 we report on the results of the optimization and the parallel performance.

2. Groundwater Temperature Control. The problem we consider in this paper was given to us by TGU (Technologieberatung Grundwasser und Umwelt) GmbH, a consulting engineering company for groundwater and water resources. We wish to control the temperature in a set of drinking water wells. The site shown in Figure 2.1 is in the recharge region for these wells. There is an industrial zone on the right of the shaded region which injects heated water in a single well,

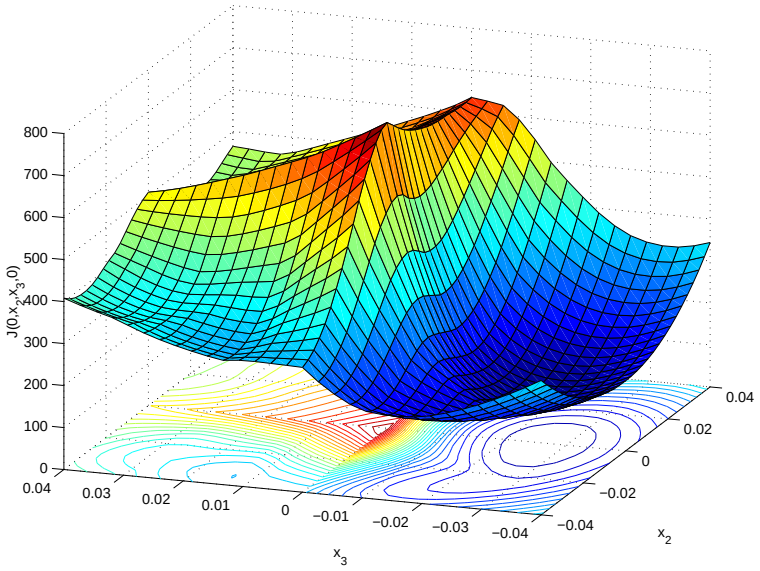
* Version of December 15, 2000.

† Universität Trier, Fachbereich IV, Abteilung Mathematik, 54286 Trier, Germany (batt@uni-trier.de). This author was supported by the foundation Stiftung Rheinland–Pfalz für Innovation.

‡ North Carolina State University, Department of Mathematics and Center for Research in Scientific Computation, Box 8205, Raleigh, N. C. 27695-8205 (tscoffe2@eos.ncsu.edu, jmgablon@unity.ncsu.edu, krkavana@unity.ncsu.edu, Tim_Kelley@ncsu.edu, hapatric@unity.ncsu.edu). This research was partially supported by National Science Foundation grants #DMS-0070641 and #DMS-9714811, Army Research Office grant #DAAD19-99-1-0186, a US Department of Education GAANN fellowship. Computing activity was partially supported by an allocation from the North Carolina Supercomputing Center.

§ Department of Environmental Sciences and Engineering, 104 Rosenau Hall, University of North Carolina, Chapel Hill, NC 27599-7400 (casey_miller@unc.edu).

FIG. 1.1. *Optimization Landscape*



the infiltration well. German law (the Wasserhaushaltsgesetz) requires that anthropogenic changes of groundwater properties be minimized. In this regulation is the requirement that drinking water be provided at the lowest temperature that is possible under undisturbed conditions. We seek to reduce the temperature at the drinking water wells by minimizing a quadratic function involving pumping rates at a set of barrier wells, which is an approximate measure of cost, and a linear combination of pumping rate and temperature at a set of drinking water wells.

Figure 2.2 shows the relative locations of the wells. The injection well is the square at the far right, the barrier wells the vertical row in the middle, and the drinking water wells are the array on the left.

Numerical experiments show that a steady-state solution is obtained after eight to ten years of real time. Because of this we may use the four steady state pumping rates as control variables. For the work reported here we neglect the vertical dimension and the dependence of viscosity and density on temperature. These assumptions enable us to decouple the equations for flow and temperature and to use a two-dimensional simulator for each. Given the controls, we can solve for the flow and use the results from the flow code to compute the temperature distribution.

To determine the flow we compute the piezometric head h from

$$S\frac{\partial h}{\partial t} = \nabla \cdot (BK\nabla h) + q \tag{2.1}$$

and appropriate initial/boundary conditions. In (2.1), S is the storage coefficient, $B(x, y)$ is the thickness of the aquifer, $K(x, y)$ is the hydraulic conductivity, and q is a source term. From the head we compute the mean macroscopic pore velocity vector via

$$\mathbf{v} = -\frac{K\nabla h}{\epsilon}, \tag{2.2}$$

where ϵ is the effective porosity of the porous medium.

FIG. 2.1. *Map of the Site*

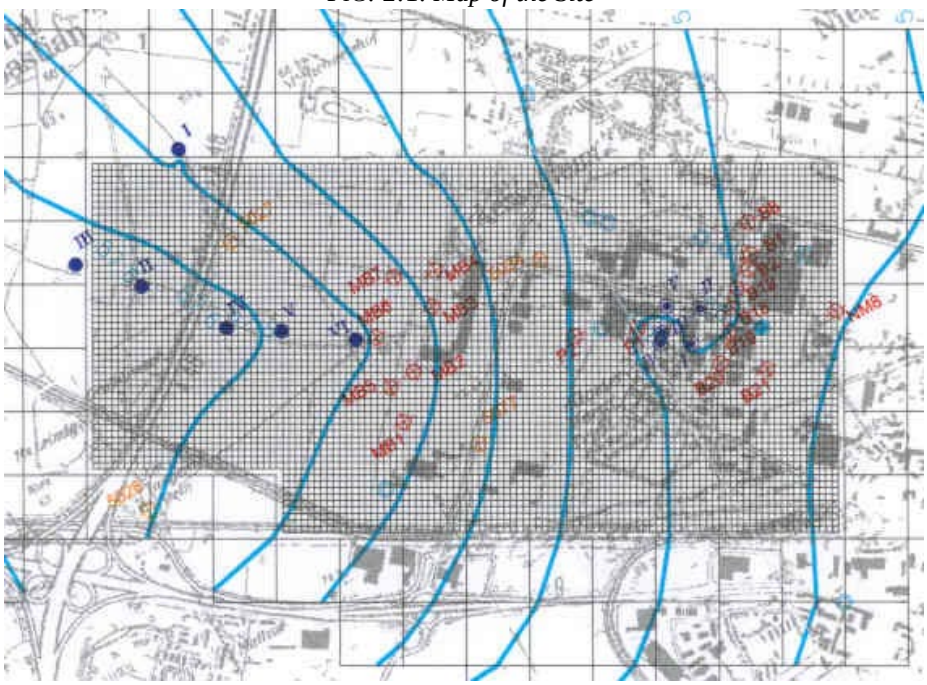
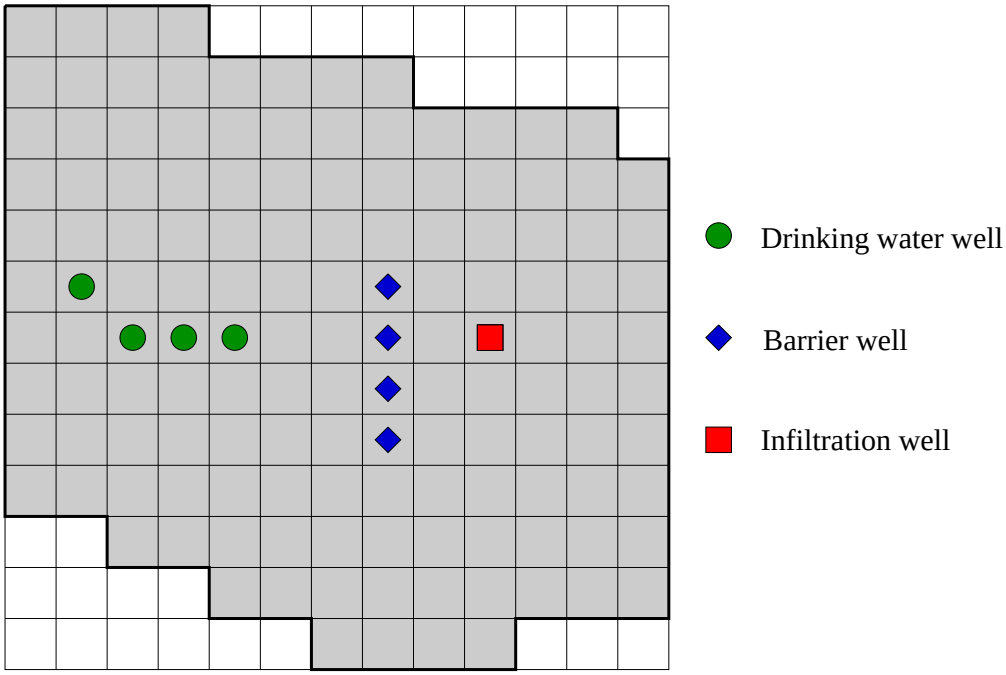


FIG. 2.2. *Well Locations*



After solving the flow equation, we model temperature in a way that a solute transport code can be used to solve the relevant equations [7].

The water temperature T satisfies

$$R\frac{\partial T}{\partial t} = \nabla \cdot (\mathbf{D} \cdot \nabla T) - \mathbf{v} \cdot \nabla T, \tag{2.3}$$

where the thermal retardation factor is

$$R = 1 + \frac{\epsilon_s \rho_s c_s}{\epsilon_a \rho_a c_a}. \tag{2.4}$$

In (2.3), $\epsilon_a = 0.3$ is the volume fraction of the aqueous phase, $\epsilon_s = 0.7$ is the volume fraction of the solid phase, $\rho_s c_s = 1.8 MJ/m^3 K$ is the heat capacity of the soil, and $\rho_a c_a = 4.185 MJ/m^3 K$ is the heat capacity of the fluid. For saturated flow, $\epsilon = \epsilon_a$.

The thermal dispersion tensor is

$$D_{ij} = \beta_t |\mathbf{v}| \delta_{ij} + (\beta_l - \beta_t)\frac{v_i v_j}{|\mathbf{v}|}, \tag{2.5}$$

where δ_{ij} is the Kronecker δ, and $\beta_l = 10m$ and $\beta_t = 1m$ are longitudinal and transversal dispersivity values that are characteristic of the porous medium. $\mathbf{D}$ is a nonsmooth function of $\mathbf{v}$, and hence of u. This accounts for the nonsmoothness that is clearly visible in Figure 1.1.

We formulate the optimization problem as

$$\min_{u \in \Omega} J(u) = u^T u + w^T T. \tag{2.6}$$

Here $u \in R^4$ is the vector of steady-state pumping rates at the control wells, $T \in R^4$ is the temperature at the drinking water wells, and

$$w = (.0325, .0119, .0440, .0461)^T \in R^4$$

is a vector of the relative pumping rates (in m^3/sec) at these wells. The truncation error in the flow and transport codes contribute low-amplitude noise to J.

The bound constraints were imposed to account for limits in the pumping rates. These constraints were not active at the solution, and the optimization was essentially an unconstrained problem.

3. Implicit Filtering. Implicit filtering [11, 15] is a projected quasi-Newton iteration which uses difference gradients, reducing the difference increment as the optimization progresses. The method was designed for problems with objective functions that are small perturbations of smooth functions. Our paradigm is

$$f = f_s + \phi \tag{3.1}$$

where f_s is smooth, and $|\phi(x)|$ is small. In practice ϕ is usually nonsmooth and sometimes discontinuous.

Implicit filtering is a sampling method. This means that the optimization is directed only by information on function values, with no gradient information. Implicit filtering differs from classical sampling methods such as the Nelder-Mead [17] or Hooke-Jeeves [12] algorithms in that it is readily implemented in parallel [3, 4, 6] by simply performing the function evaluations needed for the difference gradient in parallel. The potential for quasi-Newton acceleration [5, 11, 15]

is a feature that other parallelizable sampling methods, such as the PDS method [8, 19, 20] or DIRECT [9, 13, 14], cannot exploit. The results reported in this paper were obtained with IFFCO, a FORTRAN implementation of implicit filtering [3].

Suppose we seek to solve

$$\min_{x\in\Omega} f(x) \tag{3.2}$$

where

$$\Omega = \{x \in R^N \mid L_i \le (x)_i \le U_i\}. \tag{3.3}$$

Here, $\{L_i\}_{i=1}^N$ and $\{U_i\}_{i=1}^N$ are sequences of real numbers such that

$$-\infty < L_i < U_i < +\infty. \tag{3.4}$$

Here we denote the ith component of the vector x by $(x)_i$ to distinguish the component index from the iteration index. We denote by $\mathcal{P}$ the l^2 projection onto Ω. For $x \in R^n$

$$\mathcal{P}(x)_i = \begin{cases} L_i & \text{if } (x)_i \le L_i \\ (x)_i & \text{if } L_i < (x)_i < U_i \\ U_i & \text{if } (x)_i \ge U_i \end{cases} \tag{3.5}$$

Implicit filtering as implemented in IFFCO begins by scaling Ω to the unit cube ($L_i = 0$ and $U_i = 1$ for all i). For $0 < h \le 0.5$, let $\nabla_h f$ denote the finite difference approximation of ∇f with step size h that uses central differences if all points of the central difference stencil are in Ω and one-sided differences in those directions in which one point in the stencil is not in Ω. The restriction $h \le .5$ implies that at least two points will be in the stencil in any coordinate direction (the center and at least one of $x \pm he$, where e is the unit vector in that direction). The stencil is used both to approximate the gradient and to provide one of the termination criteria. Let $S(x, h)$ be the difference stencil about x in Ω with step size h. We call the condition

$$f(x) \le \min_{z\in S(x,h)} f(z) \tag{3.6}$$

stencil failure. In the unconstrained case [2, 15] stencil failure implies that $\nabla f_s = O(h)$. A similar result also holds in the bound constrained case, where stencil failure implies that

$$x - \mathcal{P}(x - \nabla f_s(x)) = O(h).$$

We terminate the quasi-Newton iteration for a given value of h after a stencil failure for this reason.

IFFCO offers a choice of SR1 and BFGS quasi-Newton updates. For bound constrained problems we recommend the SR1 update. We will formally describe the algorithm. We begin with Algorithm **fdquasi**, which is a finite difference projected quasi-Newton iteration for (3.2).

Implicit filtering is a sequence of calls to **fdquasi** with the difference increments or *scales* reduced after each return from **fdquasi**.

There are several convergence theorems for implicit filtering [5, 11, 15]. We state a typical result from [15] for completeness.

THEOREM 3.1. *Let f satisfy* (3.1) *and let ∇f_s be Lipschitz continuous. Let $h_k \to 0$, $\{x_k\}$ be the implicit filtering sequence, and $S^k = S(x, h_k)$. Assume that fewer than $amax$ backtracks are taken for all but finitely many k. Then if*

$$\lim_{k\to\infty} (h_k + h_k^{-1} \max_{z\in S^k} |\phi(z)|) = 0 \tag{3.7}$$

Algorithm 1 fdquasi$(x, f, pmax, \tau, h, amax)$

$p = 1$
while $p \leq pmax$ and $\|x - \mathcal{P}(x - \nabla_h f(x))\| \geq \tau h$ **do**
  compute f and $\nabla_h f$
  if (3.6) holds **then**
    terminate and report **stencil failure**
  end if
  update the model Hessian H if appropriate; solve $Hd = -\nabla_h f(x)$
  use a backtracking line search, with at most $amax$ backtracks, to find a step length λ
  if $amax$ backtracks have been taken **then**
    terminate and report **line search failure**
  end if
  $x \leftarrow \mathcal{P}(x + \lambda d)$
  $p \leftarrow p + 1$
end while
if $p > pmax$ report **iteration count failure**

Algorithm 2 imfilter$(x, f, pmax, \tau, \{h_k\}, amax)$

for $k = 0, \ldots$ **do**
  `fdquasi`$(x, f, pmax, \tau, h_k, amax)$
end for

then any limit point of the sequence $\{x_k\}$ *is a critical point of* f_s.

The implicit filtering method has many parameters, the sequence of scales, the termination parameter τ, and the limits $amax$ and $pmax$ on the inner and outer iterations. We will discuss our settings of those parameters in § 4.

The most significant opportunity for parallelism is in the computation of $\nabla_h f$, where all the function evaluations for $x \in S(x, h)$ are independent. One can also perform the line search function evaluations in parallel. In § 4.2 we show how the parallelism can be effectively exploited by IFFCO.

4. Computational Results. The computations reported in this section were done on the IBM SP/2 supercomputer located at the North Carolina Supercomputer Center running IBM AIX 4.3. This IBM SP/2 consists of 180 nodes, where each node consists of four 375 MHz Power3-II processors. Each node has 2 GB of memory. We used the IBM xlf 7.1 FORTRAN compiler.

The parameters in the implicit filtering algorithm were $\tau = 1$, $amax = 5$, $pmax = 10$, $U_i = .04$ for $i = 1, \ldots, 4$, and $L_i = -.04$ for $i = 1, \ldots, 4$. We used the SR1 quasi-Newton method and imposed a limit of 50 function evaluations on the optimization. The scales were $h_i = 2^{-i}$ for $1 \leq i \leq 9$. We terminated the optimization after we expended the budget of 50 function evaluations.

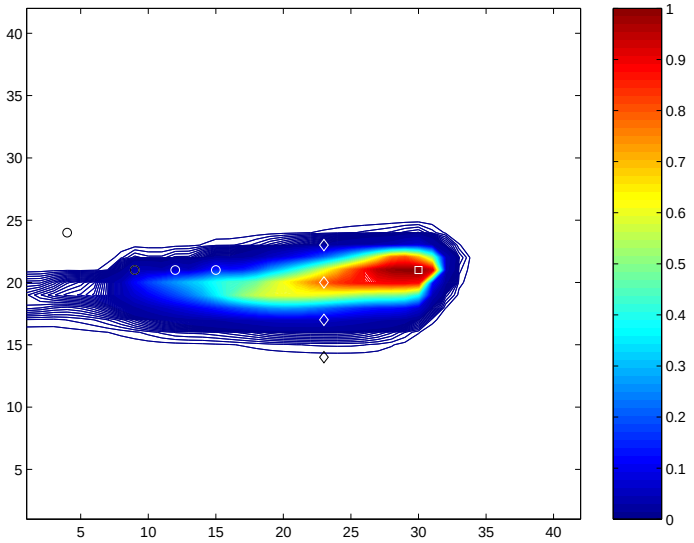
The parallelism was in the simultaneous evaluations of the objective function to form the difference gradients. We discretized the flow equations on a 42×42 mesh and used MODFLOW [16] to compute the piezometric head. From the head we extracted the velocity vector and used MT3D [18], a transport simulator, to compute the temperature distribution on the mesh. See [1]

for a more complete account of the model, the boundary conditions, and the underlying physical assumptions. We computed the steady-state solutions using accurate temporal integration out to ten years. For the flow simulation 120 time steps of 30 days are taken. The transport integration was explicit, and we took 150 transport steps for each flow step. MODFLOW and MT3D communicate via disk I/O.

4.1. Effectiveness of the Control. In Figures 4.1 and 4.2 we plot contours of temperature. We normalize the $10°C$ temperature of the groundwater to zero and the $15°C$ temperature of the water from the injection well to one. The injection well is located at the box on the right side of the plume, the control wells at the vertical row of diamonds in the center of the plume, and the drinking water wells at the circles to the left of the control wells. The temperature of the injected water is $5°C$ warmer that the ambient groundwater temperature of $10°C$. This leads to an increase of $1°C$ at the drinking water wells for the uncontrolled flow, to high satisfy the regulations.

The figures clearly show that the optimized pumping rates reduce the temperature at the drinking wells and that the size of the high temperature plume has been reduded. The maximum temperature at the drinking water wells is $10.1°C$ for the controlled flow, which is within regulatory limits.

FIG. 4.1. *Temperature Distribution: Uncontrolled Flow*

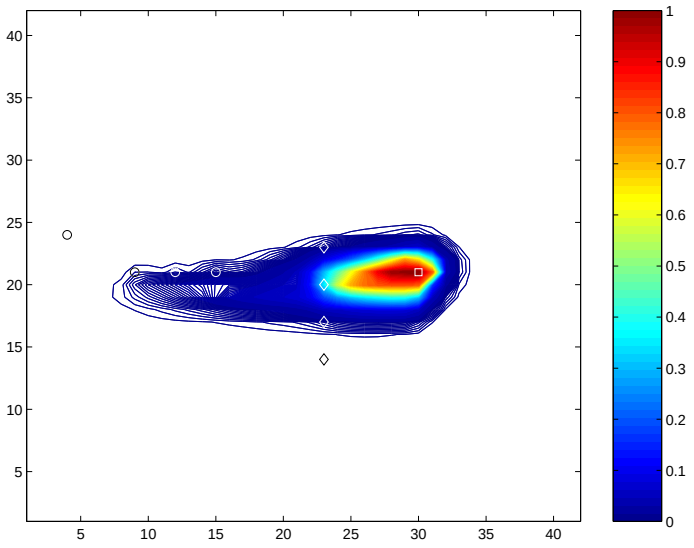


4.2. Parallel Performance. As we described earlier in § 3, there are two opportunities for parallelism in IFFCO, the evaluation of the gradient and the line search. We exploit these possibilities in our implementation by using the PVM parallel programming library.

The processors on each node share 2 gigabytes of memory (which did not affect our computations) and a local, temporary directory. We used this temporary directory for the data files and temporary files we needed in our simulation. Since four processors shared the same local directory, we added a unique task identification number (TID) to each each temporary file to prevent the different processors from writing to the same file.

The PVM programming library leads to the use of the master-slave parallel programming paradigm. In the computation a master processor did all the work in IFFCO except for the function

FIG. 4.2. *Temperature Distribution: Controlled Flow*



Number of Processors	Run-time (in sec.)	Speed up
1	5582.89	1.0000
2+1	2986.31	1.8695
4+1	1618.64	3.4491
8+1	1050.15	5.3163

TABLE 4.1
Paralell efficiency

evaluations. The time needed to do this was much smaller than the time needed to evaluate a function. Therefore, we used the master to run IFFCO and used both the master and the slaves to do the function evaluations needed during the evaluation of the gradient and the line search. We only needed to send short messages between the master and the slaves, so the communication times were very small compared to the computations. This means we only needed to use the basic send and receive mechanisms provided by PVM.

The PVM implementation available on the IBM SP/2 needed a dedicated processor to run the PVM server. In Table 4.1 we show the times needed to solve the problem with different numbers of processors. We record the number of processors as, for example, $2 + 1$ to emphasize that one processor was needed as the PVM server (a characteristic of the IBM SP/2 PVM). The last column of the table shows the speedup factor

$$S_i = \frac{T_1}{T_i},$$

where T_1 is the time needed with one processor and T_i is the time needed with $i + 1$ processors. Perfect speedup for our configuration would be $S_i = i$.

Note that it does not make sense for this problem to use more than nine processors. At most eight processors are required for evaluating the gradient, and one is required as the PVM server. Table 4.1 shows good parallel performance.

REFERENCES

[1] A. BATTERMANN, *Mathematical Optimization Methods for the Remediation of Ground Water Contaminations*, PhD thesis, Universität Trier, Trier, Germany, 2001.

[2] D. M. BORTZ AND C. T. KELLEY, *The simplex gradient and noisy optimization problems*, in Computational Methods in Optimal Design and Control, J. T. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds., vol. 24 of Progress in Systems and Control Theory, Birkhäuser, Boston, 1998, pp. 77–90.

[3] T. D. CHOI, O. J. ESLINGER, P. GILMORE, A. PATRICK, C. T. KELLEY, AND J. M. GABLONSKY, *IFFCO: Implicit Filtering for Constrained Optimization, Version 2*, Tech. Rep. CRSC-TR99-23, North Carolina State University, Center for Research in Scientific Computation, July 1999.

[4] T. D. CHOI, O. J. ESLINGER, C. T. KELLEY, J. W. DAVID, AND M. ETHERIDGE, *Optimization of automotive valve train components with implicit filtering*, Optimization and Engineering, 1 (2000), pp. 9–28.

[5] T. D. CHOI AND C. T. KELLEY, *Superlinear convergence and implicit filtering*, SIAM J. Optim., 10 (2000), pp. 1149–1162.

[6] J. W. DAVID, C. Y. CHENG, T. D. CHOI, C. T. KELLEY, AND J. GABLONSKY, *Optimal design of high speed mechanical systems*, Tech. Rep. CRSC-TR97-18, North Carolina State University, Center for Research in Scientific Computation, July 1997. Mathematical Modeling and Scientific Computing, to appear in Vol 9.

[7] G. DE MARSILY, *Groundwater Hydrology for Engineers*, Academic Press, Orlando, 1986.

[8] J. E. DENNIS AND V. TORCZON, *Direct search methods on parallel machines*, SIAM J. Optim., 1 (1991), pp. 448 – 474.

[9] J. GABLONSKY, *An implementation of the DIRECT algorithm*, Tech. Rep. CRSC-TR98-29, North Carolina State University, Center for Research in Scientific Computation, August 1998.

[10] P. GILMORE, *An Algorithm for Optimizing Functions with Multiple Minima*, PhD thesis, North Carolina State University, Raleigh, North Carolina, 1993.

[11] P. GILMORE AND C. T. KELLEY, *An implicit filtering algorithm for optimization of functions with many local minima*, SIAM J. Optim., 5 (1995), pp. 269–285.

[12] R. HOOKE AND T. A. JEEVES, *'Direct search' solution of numerical and statistical problems*, Journal of the Association for Computing Machinery, 8 (1961), pp. 212–229.

[13] D. R. JONES, *The DIRECT global optimization algorithm*. to appear in the Encylopedia of Optimization, 1999.

[14] D. R. JONES, C. C. PERTTUNEN, AND B. E. STUCKMAN, *Lipschitzian optimization without the Lipschitz constant*, J. Optim. Theory Appl., 79 (1993), pp. 157–181.

[15] C. T. KELLEY, *Iterative Methods for Optimization*, no. 18 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1999.

[16] M. G. MCDONALD AND A. W. HARBAUGH, *A modular three-dimensional finite-difference groundwater flow model*, U.S. Geological Survey Techniques of Water Resources Investigations, Book 6, Ch. A1, Reston, VA, 1988.

[17] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, Comput. J., 7 (1965), pp. 308–313.

[18] S. S. PAPADOPULOS, *MT3D: A modular three-dimensional transport model, Version 1.5, Documentation and User's Guide*, S. S. Papadopulos & Associates, Inc., Bethesda, Maryland, 1992.

[19] V. TORCZON, *On the convergence of the multidimensional direct search*, SIAM J. Optim., 1 (1991), pp. 123–145.

[20] ——, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.