Algorithms for Noisy Problems in Gas Transmission Pipeline Optimization[*]

R. G. Carter[†] J. M. Gablonsky, [‡] A. Patrick[†] C. T. Kelley[†]
O. J. Eslinger[§]

April 14, 2000

Abstract

In this paper we describe some algorithms for noisy optimization in the context of problems from the gas transmission industry. The algorithms are implicit filtering, DIRECT, and a new hybrid of these methods, which uses DIRECT to find an initial iterate for implicit filtering. We report on numerical results that illustrate the performance of the methods.

1 Introduction

In this paper we describe some algorithms for noisy optimization and apply these algorithms to problems from the gas transmission industry. As examples we consider simulations actually encountered in the gas transmission pipeline industry. Our optimization tests are performed on piecewise linear interpolations of these data rather than the raw simulations, both to protect proprietary data and to make it easy to distribute the problems to the wider community.

The algorithms we considered are implicit filtering [32, 21, 27], DIRECT [26], and a new hybrid of implicit filtering and DIRECT, which attempts to capture the best features of the two.

1.1 The Application

The United States uses over 20 trillion standard cubic feet of natural gas per year, representing roughly a third of worldwide consumption. The gas is produced from wells throughout the country, with high production concentrations in such regions as Louisiana, the Texas Gulf Coast, and

[*]This research was supported by National Science Foundation grants #DMS-9700569 and #DMS-9714811. Computing activity was partially supported by an allocation from the North Carolina Supercomputing Center.

[†]Stoner Associates, Inc. 5177 Richmond Avenue, Suite 900 Houston TX 77056-6736

[‡]North Carolina State University, Center for Research in Scientific Computation and Department of Mathematics, Box 8205, Raleigh, N. C. 27695-8205

[§]TICAM Room 417, 105 West Dean Keeton Austin, TX 78712, USA

the Texas Panhandle. The gas is then transported through pipelines to major population centers throughout the country for use in diverse applications in industry, residential and commercial heating, and to fuel electric power stations.

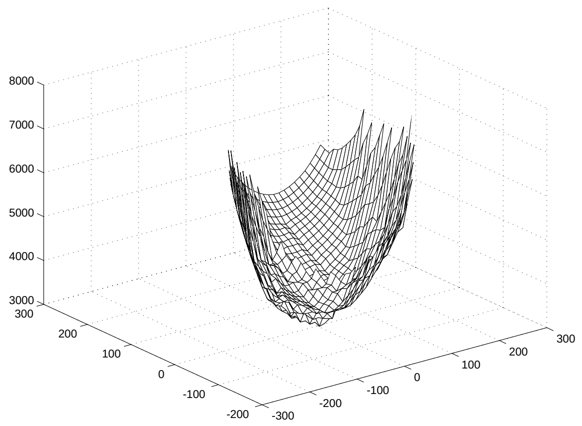
Pipelines can be as short as a few hundred miles or as long as two thousand. They can be simple point-to-point "gunbarrel" systems, or can have extensive branching and even looped topologies offering alternate possible paths for gas flow. Energy must be added to the gas at various points in any pipeline to overcome frictional pressure losses, so series-parallel banks of compressors are typically positioned at 30 to 50 mile intervals along the pipeline. These "compressor stations" also allow operational control of the pipeline. Efficiency of operation can be dramatically effected by the correct choice of pressure and flow distribution.

Some compressors are electrically powered, but most still use the traditional technique of diverting a small part of the transported gas as fuel for their drivers. The standard "rule of thumb" is that 5% of transported gas is diverted for fuel on a long pipeline, or 10^{12} cubic feet per year in the US. With wholesale gas prices in the range of 1\$ to 2\$ per 10^3 cubic feet, clearly even a small improvement in efficiency can have huge economic benefit.

We consider minimization of the cost of fuel and/or electric power for the compressor stations in a gas pipeline network. This cost can be reduced by changing both flow patterns through the system and pressure settings throughout the system. In this paper the problems have two flows as design variables. Each evaluation of f requires an internal optimization of the system-wide pressure settings. The flow variables can be either unknown inlet or outlet flows, or Kirchoff law representations of flow splits between possible alternative paths.

Figures 1, 2, and 3 are optimization landscapes of the three examples we consider in the numerical results in § 5. In the figures of the optimization landscapes, the vertical axis is the amount of gas (in thousand standard cubic feet/day) that is used for fuel and the two horizontal axes are the flow variables (in million standard cubic feet per day).

Figure 1: Problem A.



The pressure optimization that is internal to the function evaluation involves solving a large combinatoric problem using nonsequential dynamic programming [3]. The dynamic programming formulation also attempts to find the best settings for other discrete variables such as whether entire compressor stations should be shut down. Depending on flow conditions a solution may not exist to this discrete problem. When this happens, the objective function will fail to return a value. This

Figure 2: Problem B.

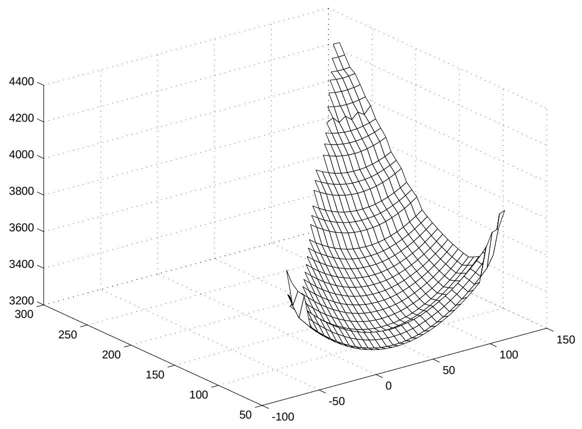
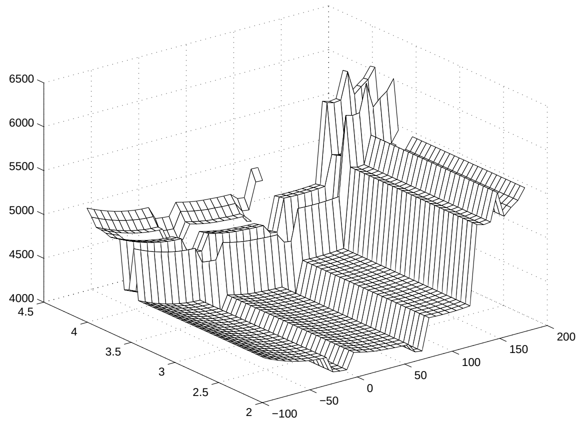


Figure 3: Problem C.



is an example of a “hidden constraint”, one for which feasibility can only be tested by attempting to evaluate the objective function. We can see this in the optimization landscapes in Figures 1, 2, and 3 as areas over which no surface is plotted. Such constraints are also called “virtual constraints” [11, 10] or “yes/no” constraints [4]. In some cases [8, 6, 15] it is even hard to find a feasible point to begin an iteration. Problem C, where less than 3% of the points that satisfy the bound constraints are feasible, is such a problem. Figure 3 shows most of the feasible set.

The discrete nature of variables embedded in this problems also give rise to discontinuities in the landscapes [5]. Many severe discontinuities come from properties of the compressor stations themselves. For example, shutting down and then restarting a compressor can cost as much as running the compressor for several hours. This sort of discontinuity in cost can be seen in Figure 3. The roughness in optimization landscapes like the ones in Figures 1 and 2 may come from entire stations being bypassed, from individual engines in stations turning on and off and producing small discontinuities, ¿from the discontinuous controls for certain types of engines, and from the replacement of continuous pressure variables with discrete ones.

1.2 The Algorithms

The algorithms are the implicit filtering method [21, 27], the DIRECT [26] method, and a hybrid of the two. Implicit filtering is designed for problems that are low-amplitude high frequency perturbations of smooth problems. The method is unlikely to be trapped by a local minimum caused by such perturbations and can be accelerated by quasi-Newton methods to give fast convergence in regions where the objective function is smooth. DIRECT's strength is global search and we use it to generate a small set of initial iterates and bound constraints for implicit filtering, which then explores the smaller regions.

Implicit filtering and DIRECT are sampling algorithms. This means that only evaluations of the function to be minimized are used in the optimization. Sampling methods converge slowly, and when gradient information is available, conventional methods work far better. Therefore, sampling methods are usually applied to difficult problems with complex optimization landscapes [32, 18, 22, 23, 5]. Figures 1, 2, and 3 are typical examples of such problems.

2 Problem Formulation

Mathematically, the problems we consider in this paper are bound constrained optimization problems of the form:

$$\min_{x \in \Omega} f(x) \tag{1}$$

where

$$x^* \in \Omega = \{x \in R^N \mid L_i \leq (x)_i \leq U_i\}. \tag{2}$$

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

$$-\infty < L_i < U_i < +\infty. \tag{3}$$

Here we denote the ith component of the vector x by $(x)_i$ to distinguish the component index from the iteration index.

As we mentioned above, in addition to the bound constraints, the objective function f may fail to return a value for some $x \in \Omega$. The authors and others have observed failure of the function evaluation in previous work, [32, 12, 8, 11, 5], and it is common in this application.

3 Algorithmic Description

Both implicit filtering and DIRECT have been described in detail elsewhere in the literature. We provide brief descriptions in this section for completeness, but refer the reader to [21, 27, 26] for details. Our implementations of the two methods have been described in [7, 19].

3.1 Implicit Filtering

Implicit filtering is a projected quasi-Newton iteration which uses difference gradients, reducing the difference increment as the optimization progresses. The method was designed with objective functions that are low-amplitude perturbations of smooth functions in mind. The convergence theory for implicit filtering [21, 27, 9] is based on that paradigm. Implicit filtering, like other sampling methods, is typically applied to more difficult problems than those covered by the theory. True filtering methods, such as the approach from [28], have exponential complexity in the dimension, and analytical filtering methods, such as the method in [29], are only effective for special classes of problems.

Implicit filtering differs from classical sampling methods such as the Nelder-Mead [30] or Hooke-Jeeves [25] algorithms in that it is readily implemented in parallel [12, 8, 7] by simply performing the function evaluations needed for the difference gradient in parallel. The potential for quasi-Newton acceleration [21, 9, 27] is a feature that other parallelizable sampling methods, such as the PDS method [14, 33, 34] or DIRECT [26, 19], can not exploit.

We let $\mathcal{P}$ denote the **projection** onto Ω, that is the map that takes x into the nearest point (in the l^2 norm) in Ω to x.

$$\mathcal{P}(x)_i = \begin{cases} L_i & \text{if } (x)_i \le L_i \\ (x)_i & \text{if } L_i < (x)_i < U_i \\ U_i & \text{if } (x)_i \ge U_i \end{cases} \tag{4}$$

For bound-constrained problems, implicit filtering computes a difference gradient $\nabla_{h_n} f(u_n)$ at u_n using a difference stencil with difference increment (or *scale*) h_n and takes a projected quasi-Newton step

$$u_{n+1} = \mathcal{P}(u_n - \lambda H_n^{-1} \nabla_{h_n} f(u_n)).$$

Here H_n is a symmetric positive definite quasi-Newton model reduced Hessian; the BFGS [2, 24, 17, 31] or SR1 [1, 16] updates are both used in our codes [27, 7]. λ is a step length that is computed using a standard backtracking line search [13, 27]. The optimization is terminated for a given h_n if either $\|\nabla_h f(x)\| \le \tau h$, where τ is a parameter in the method, if too many reductions in the step length are taken without sufficient decrease in f, or if the center of the difference stencil has the lowest value. In the computations reported here this latter condition (stencil failure) was always the cause of termination at a given scale until the limit on function evaluations was exceeded.

The difference increment h_n is decreased as the iteration progresses and because of this, the expectation is that local minima that correspond only to high-frequency, low-amplitude noise are avoided by the coarse difference and low-amplitude non-differentiable noise is thereby "filtered out".

In the computations we used our implementation of implicit filtering, the FORTRAN code IFFCO [20, 7]. In this code implicit filtering is given a budget of ν_I function evaluations and a lower limit on the difference increment. Once the budget of function evaluations has been exceeded, the optimization terminates after the next complete iteration. The function evaluation budget for IFFCO, like the one in our implementation of DIRECT, is not a hard limit and the iteration is never stopped in midcourse.

Sometimes implicit filtering can find a better point if it is restarted with the results from the previous run as the initial iterate. That was the case with one of the examples in this paper.

3.2 DIRECT

DIRECT repeatedly subdivides the feasible region, managing the subdivision using the history of the optimization based on estimation of the Lipschitz constant of the objective. Initially the feasible region is trisected, forming three hyperrectangles of equal size. In subsequent rounds of subdivision, not all hyperrectangles are divided and, therefore, the number of function evaluations in a round of subdivision depends on the objective function and cannot be determined before the optimization begins. DIRECT is limited to a budget of ν_D function evaluations and, when that budget is exceeded, the current round of subdivisions is allowed to complete before the optimization stops.

4 The Hybrid Algorithm

4.1 Global Search with DIRECT

Our hybrid method uses DIRECT to find an initial iterate for implicit filtering. If, as happens once in the results we report in $\S$ 5, DIRECT fails to find a feasible point within its budget of function evaluations, we allow it to run until a feasible point has been found, allocate the budget anew, and run DIRECT until this new budget has been exhausted. Then we start IFFCO with the best point found so far as initial iterate. We call this method *DIRECT-IFFCO*.

We use the same strategy when DIRECT is run alone.

4.1.1 Hidden Constraints and DIRECT

An obvious way to address the situation where f cannot be evaluated at x_i is to assign a large value to $f(x_i)$. We see a weakness in this approach if the solution x^* lies on a constraint boundary. At any stage of DIRECT, x^* will be within some small hyperrectangle created by the subdivision process. Let x_k be the center of this hyperrectangle and $B(x_k)$ be the hyperrectangle itself.

If x_k is feasible, the selection process within DIRECT will place high priority on subdividing $B(x_k)$, just as in the unconstrained case. But if x_k is infeasible and $f(x_k)$ is assigned a large value, $B(x_k)$ will not be subdivided until *every* hyperrectangle of comparable size with feasible center in the entire search space has been subdivided. Since the number of such hyperrectangles is typically on the order $\frac{1}{||x_* - x_k||^n}$ we see that the "standard" treatment of hidden constraints destroys any possibility of rapid local convergence to a solution. In fact, DIRECT is forced into the worst-case scenario of exponential search-by-subdivision rather than being able to generate at least some subsequence of iterates with fast convergence.

This behavior is insidious, in that function values in DIRECT will appear to be converging to a lower limit. There is no signal that this lower limit is actually far from $f(x^*)$ until the entire space is subdivided to the point that $B(x_k)$ can finally be subdivided. Once the subdivision has occurred, we have a new closest point x_m. If it happens to be infeasible the process starts again, but reaching the point where $B(x_m)$ can be subdivided will require roughly 3^n as many function evaluations as were required previously.

We propose a simple modification of DIRECT that eliminates this behavior. When a function evaluation called by DIRECT fails to return a value, we mark the point infeasible rather than assigning it an arbitrary large function value. At the end of the current round of subdivisions, that is before DIRECT decides which hyperrectangles are divided next, each infeasible point is assigned a function value derived from nearby feasible points rather than blindly assigning an arbitrary large value. This derived function value is then used in the decision for which hyperrectangles to subdivide; hence infeasible points that are close to promising feasible points are also likely to be subdivided for further evaluation.

To be specific, we temporarily expand the hyperrectangle corresponding to an infeasible point by a factor of two. If that larger hyperrectangle contains one or more feasible points from the iteration history, we find the lowest of these function values f_m. We then assign $f_m + 10^{-6}|f_m|$ to the infeasible point. If there are no feasible points in the larger hyperrectangle, we deem this point to be far from the constraint boundary and assign the maximum of the values found so far plus 1. We enlarge the minimal value of a nearby point slightly to ensure DIRECT will preferentially choose a hyperrectangle with a feasible center over one with an infeasible center, provided the hyperrectangles have the same size. For the same reason we increase the maximum value found so far by 1.

Note that the value assigned to an infeasible point can change as the iteration progresses and new nearby feasible points are found. Care must be taken in implementing this approach so that overhead costs do not become significant. Fortunately, fast heap algorithms exist for finding nearby neighbors for each new point x_k. In our implementation, the function count budget for DIRECT was sufficiently limited that such fast algorithms were not needed.

4.2 Local Search with Implicit Filtering

4.2.1 Hidden Constraints and Implicit Filtering

In this work, as was the case in older work using implicit filtering [32], f is assigned an artificial value when the evaluation fails. In our previous work that value was very large ($10^6 - 10^8$). This approach, while successful in the past, failed in these applications. The reason for this failure was that if one point on the difference stencil failed to return a value, then the difference gradient was not an ascent direction for f. While this failure was a possibility in our previous work, this is the first time we have observed it in practice.

In the hybrid algorithm, we begin the implicit filtering iteration with a feasible point. Hence the center of each difference stencil will be feasible and the issue is only the response of the algorithm to a function failure at a point on the stencil that is not the center.

We respond to failure of the function evaluation at a point on the stencil by assigning the point a function value equal to the maximum of the value on the stencil. If all points, other than the center, on the stencil are infeasible, we reduce the size of the stencil.

5 Applications and Examples

The vertical axis in the optimization landscapes is thousand cubic feet per day (MCF/day). The value of a reduction in the objective function can be roughly estimated using a wholesale price of \$1.20/MCF. Using this estimate we can value a thousand cubic foot/day at roughly \$400/year. Using this estimate, the improvement shown by the combined methods in Table 3 from a value of 4400 for implicit filtering alone to a value of 4041 for the combined method implies a savings of \$160,000 per year in operational costs, assuming that this improvement is representative of typical results on runs throughout the year.

The resolution of the models means that values within 50 of each other such as those observed in most of the results from Tables 1 and 2 should be regarded as equally good.

In each of the reports on the three problems, we provide a table that summarizes the progress of three methods: implicit filtering alone, DIRECT alone, and DIRECT-IFFCO. In the table we give the best objective function value found at each stage of the algorithm.

For the DIRECT-IFFCO method, one can see from these values how much the function was decreased by implicit filtering after the DIRECT iteration terminated. One can also see any differences between the three methods. In each of the computations we give DIRECT a budget of ν_D function evaluations and IFFCO a budget of ν_I. Keep in mind that if DIRECT fails to find a feasible point after its budget is exhausted, it is allowed to run until a feasible point is found and then reallocated its budget.

With this notation $DI(\nu_D, \nu_I)$, for example, means a run of DIRECT-IFFCO in which DIRECT was allocated ν_D function evaluations and the final result of the run of DIRECT was used to initialize a run of IFFCO with a budget of ν_I function evaluations. In one of the problems (C) IFFCO was restarted after convergence, the notation IFFCO-R(ν_I) means that IFFCO was run once with a budget of ν_I function evaluations and then restarted from the terminal point with a second allocation of ν_I function evaluations.

Implicit filtering was run for the scales $\{2^{-k}\}_{k=1}^{kmax}$ and $kmax$ is given in the table. The implicit filtering optimization terminated once the scales had been exhausted with termination at each one or the function budget exceeded.

We experimented with several combinations of algorithmic parameters and function evaluation budgets. We present the typical results from each of the three methods with a budget of 30 function evaluations, which gave acceptable results in problems (A) and (B). In problem (C), DIRECT required 829 function evaluations to find a feasible point and restarted implicit filtering with a budget of 100 did extremely well in comparison with the methods that depended on DIRECT. This was largely a matter of luck, as IFFCO reduced the scale six times and then found a feasible point very near the center of the bounding box.

In the tables we show how many function evaluations were actually done in each phase (DIRECT and implicit filtering), the function value after the completion of each phase, and the limit ($kmax$) on the difference increment for implicit filtering.

Table 1: Problem A, minimum value = 3233

Bounds: $[-374.085, 374.085] \times [-374.085, 575.142]$

Method	function evaluations			function value		kmax
	DIR	IFFCO	Total	DIR	IFFCO	
Iffco (30)		37	37		3257	12
DI(20,10)	23	21	44	3320	3253	12
DIRECT (30)	37		37	3291		
DIRECT (40)	45		45	3282		
DIRECT (50)	51		51	3275		

Table 2: Problem B, minimum value = 3204
Bounds: $[-329.075, 329.0759] \times [-329.076, 665.73]$

Method	function evaluations			function value		kmax
	DIR	IFFCO	Total	DIR	IFFCO	
Iffco (20)		25	25		3214	12
Iffco (30)		35	35		3208	12
DI(20,10)	21	18	39	3477	3228	12
DIRECT (30)	33		33	3347		
DIRECT (40)	41		41	3270		
DIRECT (50)	53		53	3228		

Table 3: Problem C, minimum value = 4041
Bounds: $[-99.766, 274.2009] \times [-58.469, 58.469]$

Method	function evaluations			function value		kmax
	DIR	IFFCO	Total	DIR	IFFCO	
Iffco (100)		88	88		4401	12
Iffco-R (100)		168	168		4041	12
DIRECT (30)	857		857	4401		
DIRECT (40)	867		867	4401		
DI(20,10)	849	25	874	4403	4074	12
DIRECT(1000)	1009		1009	4042		

5.1 Conclusions

In two of the problems (A and B), implicit filtering performs at least as well as the other three methods and significantly better in problem B. DIRECT does relatively poorly in problem B, needing a larger budget of function evaluations to get comparable results. In both problems A and B the final function values obtained by the three methods do not differ in a significant way if a sufficient budget is allocated. However, for problem C implicit filtering finds a point that is significantly far from optimal and DIRECT alone needs far more function evaluations to obtain a results as good as DIRECT-IFFCO. If implicit filtering is restarted from that non-optimal point, however, it finds a point as good as the other three methods at much lower cost. However, restarting implicit filtering is not an effective general method, producing no improvement, for example, on problems A and B.

We believe that the hybrid DIRECT-IFFCO algorithm has the best combination of low cost and robustness.

References

[1] Broyden, C. G.: 1967, ‘Quasi-Newton methods and their application to function minimization’. *Math. Comp.* **21**, 368–381.

[2] Broyden, C. G.: 1969, ‘A new double-rank minimization algorithm’. *AMS Notices* **16**, 670.

[3] Carter, R. G.: 1998, ‘Pipeline optimization: dynamic programming after 30 years’. Proceedings of the Pipeline Simulation Interest Group, Paper number PSIG-9803.

[4] Carter, R. G.: 1999, ‘Nonsequential Dynamic Programming for Optimizing Pipelines’. Presentation at the 1999 SIAM Conference on Optimization.

[5] Carter, R. G., W. W. Schroeder, and T. D. Harbick: 1993, ‘Some causes and effect of discontinuities in modeling and optimizing gas transmission networks’. Proceedings of the Pipeline Simulation Interest Group, Paper number PSIG-9308.

[6] Choi, T. D.: 1999, ‘Bound Constrained Optimization’. Ph.D. thesis, North Carolina State University, Raleigh, North Carolina.

[7] Choi, T. D., O. J. Eslinger, P. Gilmore, A. Patrick, C. T. Kelley, and J. M. Gablonsky: 1999, ‘IFFCO: Implicit Filtering for Constrained Optimization, Version 2’. Technical Report CRSC-TR99-23, North Carolina State University, Center for Research in Scientific Computation.

[8] Choi, T. D., O. J. Eslinger, C. T. Kelley, J. W. David, and M. Etheridge: 1998, ‘Optimization of Automotive Valve Train Components with Implicit Filtering’. Technical Report CRSC-TR98-44, North Carolina State University, Center for Research in Scientific Computation. To appear in Optimization and Engineering.

[9] Choi, T. D. and C. T. Kelley: 1999, ‘Superlinear Convergence and Implicit Filtering’. Technical Report CRSC-TR99-14, North Carolina State University, Center for Research in Scientific Computation. To appear in SIAM J. Opt.

[10] Conn, A. R., K. Scheinberg, and P. Toint: 1999, ‘Derivative Free Optimization Algorithms for Constrained Problems’. Presentation at the 1999 SIAM Conference on Optimization.

[11] Conn, A. R., K. Scheinberg, and P. L. Toint: 1998, ‘A Derivative Free Optimization Algorithm in Practice’. Proceeedings of 7-th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St Louis, MO. Sept 2-4, 1998.

[12] David, J. W., C. Y. Cheng, T. D. Choi, C. T. Kelley, and J. Gablonsky: 1997, ‘Optimal Design of High Speed Mechanical Systems’. Technical Report CRSC-TR97-18, North Carolina State University, Center for Research in Scientific Computation. Mathematical Modeling and Scientific Computing, to appear in Vol 9.

[13] Dennis, J. E. and R. B. Schnabel: 1996, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, No. 16 in Classics in Applied Mathematics. Philadelphia: SIAM.

[14] Dennis, J. E. and V. Torczon: 1991, ‘Direct Search Methods on Parallel Machines’. *SIAM J. Optim.* **1**, 448 – 474.

[15] Etheridge, M.: 1998, ‘Preliminary Performance of Carbon-Carbon Valves in High Speed Pushrod Type Valve Trains’. Master’s thesis, North Carolina State University, Raleigh, North Carolina.

[16] Fiacco, A. V. and G. P. McCormick: 1990, *Nonlinear Programming*, No. 4 in Classics in Applied Mathematics. Philadelphia: SIAM.

[17] Fletcher, R.: 1970, 'A new approach to variable metric methods'. *Comput. J.* **13**, 317–322.

[18] Fortune, S. J., D. M. Gay, B. W. Kernighan, O. Landron, R. A. Valenzuela, and M. H. Wright: 1995, 'WISE design of indoor wireless systems'. *IEEE Computational Science and Engineering* **Spring**, 58–68.

[19] Gablonsky, J.: 1998, 'An Implementation of the DIRECT Algorithm'. Technical Report CRSC-TR98-29, North Carolina State University, Center for Research in Scientific Computation.

[20] Gilmore, P.: 1993, 'IFFCO: Implicit Filtering for Constrained Optimization'. Technical Report CRSC-TR93-7, Center for Research in Scientific Computation, North Carolina State University. available by anonymous ftp from ftp.math.ncsu.edu in FTP/kelley/iffco/ug.ps.

[21] Gilmore, P. and C. T. Kelley: 1995, 'An implicit filtering algorithm for optimization of functions with many local minima'. *SIAM J. Optim.* **5**, 269–285.

[22] Gilmore, P., P. Pernambuco-Wise, and Y. Eyssa: 1994, 'An Optimization Code for Pulse Magnets'. Technical report, National High Magnetic Field Laboratory, Florida State University.

[23] Gilmore, P. A., S. S. Berger, R. F. Burr, and J. A. Burns: November, 1997, 'Automated optimization techniques for phase change piezoelectric ink jet performance enhancement'. In: *1997 International Conference on Digital Printing Technologies*. pp. 716–721.

[24] Goldfarb, D.: 1970, 'A family of variable metric methods derived by variational means'. *Math. Comp.* **24**, 23–26.

[25] Hooke, R. and T. A. Jeeves: 1961, "Direct search' solution of numerical and statistical problems'. *Journal of the Association for Computing Machinery* **8**, 212–229.

[26] Jones, D. R., C. C. Perttunen, and B. E. Stuckman: 1993, 'Lipschitzian Optimization Without the Lipschitz Constant'. *J. Optim. Theory Appl.* **79**, 157–181.

[27] Kelley, C. T.: 1999, *Iterative Methods for Optimization*, No. 18 in Frontiers in Applied Mathematics. Philadelphia: SIAM.

[28] Kostrowicki, J. and L. Piela: 1991, 'Diffusion Equation Method of Global Minimization: Performance for Standard Test Functions'. *J. Optim. Theory Appl.* pp. 269–284.

[29] Moré, J. J. and Z. Wu: 1997, 'Global continuation for distance geometry problems'. *SIAM J. Optim.* **7**, 814–836.

[30] Nelder, J. A. and R. Mead: 1965, 'A simplex method for function minimization'. *Comput. J.* **7**, 308–313.

[31] Shanno, D. F.: 1970, ‘Conditioning of quasi-Newton methods for function minimization’. *Math. Comp.* **24**, 647–657.

[32] Stoneking, D., G. Bilbro, R. Trew, P. Gilmore, and C. T. Kelley: 1992, ‘Yield Optimization Using a GaAs Process Simulator Coupled to a Physical Device Model’. *IEEE Transactions on Microwave Theory and Techniques* **40**, 1353–1363.

[33] Torczon, V.: 1991, ‘On the convergence of the multidimensional direct search’. *SIAM J. Optim.* **1**, 123–145.

[34] Torczon, V.: 1997, ‘On the convergence of pattern search algorithms’. *SIAM J. Optim.* **7**, 1–25.