**POLYMER EXTRUSION FILTER DESIGN WITH A HYBRID PSO-GA OPTIMIZATION APPROACH**

K.R. Fowler1, E.W. Jenkins2, and B. McClune1,

1 Department of Mathematics, Clarkson University, Potsdam NY, 13676-5815

(kfowler@clarkson.edu, bpmcclune@gmail.com)

2Department of Mathematical Sciences, Clemson University, Clemson SC 29634-0975

(lea@clemson.edu)

**ABSTRACT**

We discuss an optimization study of two-layer extrusion filter designs using a three-dimensional computational simulator and derivative-free hybrid optimization methods. The simulator models flow of a non-Newtonian fluid through a multi-layered filter with debris deposition. Previous studies used a derivative-free sampling algorithm to maximize different performance measures of one- and two-layer filters, relative to changes in porosity and pore diameter in each layer. A challenge in those studies, and the motivation for this work, is that the single-search optimization algorithm used would converge to a sub-optimal filter design for certain starting points. In this work, we apply a new hybrid optimization algorithm that combines two heuristic search methods: a genetic algorithm (GA) and a particle swarm optimization (PSO) algorithm. Both are known to exhaustively search the design space and are less likely to stagnate at a local minimum. They do, however, require a significant number of calls to the simulator. This is computationally expensive, as each call may require more than an hour of compute time. We improve the efficiency by incorporating surrogate functions (i.e. a cheaper approximation to the real objective function) into the search phase. We present numerical results for a two-layer extrusion filter design and discuss extensions to more complicated filter designs.

**1-INTRODUCTION**

The Center for Advanced Engineering Fibers and Films, or CAEFF (1), assists industrial partners in the development and manufacture of new polymer products. Toward this end, members have created simulation tools for various stages of polymer production; one such tool is a three-dimensional model of an extrusion filter, which separates debris particles from molten polymer before the material is spun into a fiber. As the mass flow rate of the polymer through this filter must be constant to ensure consistent fiber production, the pressure drop across the filter increases as debris accumulates. The increased pressure will damage the pumping mechanism if it exceeds a certain threshold; thus, the filter must be replaced before this occurs.

The filter may be composed of a sintered metal, compressed with sufficient force to produce a cake material, or layers of wired mesh, with mesh spacing small enough to trap particles only a few microns in diameter. Design parameters that distinguish extrusion filters include the number of individual layers and the characteristics of each layer. Such characteristics include the length of the layer, the porosity, *η*, and the average pore diameter, *dp*. We assume that the density of the debris in the polymer is negligible in comparison to the density of the polymer. We also assume the overall length of the filter is small enough so that effects of gravity can be ignored (2; 3).

One measure of filter performance is its effectiveness in removing debris particles from the molten polymer, as the resultant fiber may be severely compromised by inclusion of particles during spinning. In addition, the cost involved with replacing a filter must be considered--both directly, in terms of filter manufacturing costs, and indirectly, as filter replacement can interrupt the manufacturing process. This cost necessitates that filter performance also be based on its lifetime. Thus, we consider optimal filter design based on competing objectives: minimizing the amount of debris that escapes while maximizing the filter lifetime.

Previously, we attempted to optimize filter performance for a one-layer model using a multi-objective genetic algorithm (GA), which generated a set of Pareto optimal solutions using two competing objective functions (4). The relatively steep computational cost of the GA was magnified by the computational expense of the filter simulator. After 100 calls to the simulator, the final shape of the tradeoff curves was not obvious.

Subsequent studies used barrier methods to collapse the competing objectives into a single fitness function. This allowed for optimization via the implicit filtering algorithm, a single-search, derivative-free quasi-Newton method (5). Both one- and two-layer models were considered (6; 7), and these studies provided insight into the behavior of the objective function in the larger design space. While implicit filtering provided a more direct route to optimal parameters in the search space, the two-layer problem proved to be considerably more complex than its one-layer counterpart. In addition, the optimization algorithm was shown to be highly sensitive to initial inputs and constraints.

The challenges in both the population-based GA and single-search approach motivated the use of hybrid optimization techniques for this application. Hybrid optimization is an emerging branch of mathematics that combines algorithms to overcome weaknesses and exploit strengths of each. As with all optimization techniques, some general means of classification have been developed, formalized most notably in a taxonomy by E.G. Talbi (8). In particular, the particle swarm optimization (PSO) technique has been growing in popularity, both as a stand-alone heuristic and as a component of new hybrid systems. The past few years have produced a number of successful new hybridizations combining a PSO with a GA. Examples include, but are certainly not limited to, ventures by Shi *et al*. (9); Grimaldi *et al*. (10); Settles and Soule (11); and by Kim (12).

Both the GA and PSO can require a large number of function evaluations, a drawback for any computationally expensive problem. Surrogate, or approximation, models that are built and used in the search phase of the optimization can help reduce these costs (13) and do not require any derivative information. Surrogate modeling is attractive for both the PSO and GA since the population of sampled points can be exploited to build a potentially accurate surrogate early in the optimization process.

In this paper, we apply a new PSO-GA hybridization proposed in (14). We attempt to maximize the lifetime of the filter while minimizing the amount of escaped debris. We evaluate algorithm performance by comparing the hybrid method, both with and without an approximation model, to a basic PSO. The numerical results demonstrate the hybrid PSO is competitive with the standard PSO. Moreover, all of these methods are able to identify an optimal filter design without the dependence on initial data, resolving one of the major weaknesses of the results presented in (7) .

**2-FILTER MODEL**

The flow through the filter is governed by mass conservation and Darcy’s Law, modified to account for the non-Newtonian behavior of the polymer. The debris entering the filter is characterized by truncated normal distributions based on particle sizes. As the polymer enters a computational cell, debris particles equal or larger than the average pore diameter of the cell are eligible for deposit. A filter-specific retention function (3), which is distinct for each layer, determines which of these particles are retained in the cell. The remainder are transported to an adjacent cell. The retention functions and the debris distributions were obtained from empirical data (3).

The porosity of the cell is updated at the end of the time step to account for accumulated debris. The updated porosities are then used to adjust the average pore diameter (3). The permeability of the filter, which measures the ability of the porous medium to transport fluid, depends on the porosity and average pore diameter. The permeability parameter *k* is currently modeled using a Blake-Kozeny relationship that depends upon the current average pore diameter and filter porosity (4). The relationship for *k*, derived by Bird, Stewart, and Lightfoot (15), is

 . (1)

As the permeability decreases, the filter loses its ability to transport the fluid, and thus the pressure drop must increase to maintain the constant mass flow rate across the filter. The simulator is designed to stop when the pressure drop reaches 35 MPa (3).

The simulator used in this work is based upon results obtained by CAEFF researchers. Though initial filtration research by Edie and Gooding (2) involved only one-dimensional equations, the work was later extended to three dimensions by Seyfzadeh, Zumbrunnen, and Ross (3); it was from the three-dimensional studies that the simulator was developed. The simulator is written in MATLAB; a typical two-layer execution completes with an average run time of approximately 40 minutes on an AMD Athlon 64 X2 2.41 GHz processor.

**3-FILTER PERFORMANCE MODEL**

As mentioned earlier, the optimization goal is to generate a set of solutions that maximize the lifetime of the filter while simultaneously minimizing the amount of debris that escapes. These are competing objectives, as filter lifetime is maximized if the filter traps nothing. As design parameters, we choose porosity (**) and pore diameter (*dp*), and we combine our competing objectives into a single objective using an additive penalty approach described below.

Suppose the lifetime of the filter consists of time steps. We can measure the total change in pressure by constructing a line through the points and obtained from a pressure drop curve, as seen in Figure 1.

**Figure 1: Representative pressure drop curve**

Our objective is to minimize the slope of this line, given by

, (2)

and has units of MPa/hour. As , is simply the lifetime of the filter and thus . Also note that , , and are easily expressed as functions of . Thus we define our objective function as

 , (3)

where is an additive penalty defined in terms of the total mass of debris that escapes,, over the lifetime of the filter. The penalty function depends on *b* > 0, defined to represent an acceptable limit for escaped debris, and may expressed piece-wise such that

 .

In Eq. (3), *ρ* is a constant with units of (MPa)/(kg h). The additive penalty approach was shown to yield better results than the barrier method proposed in previous work (14). For the numerical results presented we use kg and (MPa)/(kg hour).

**4-HYBRID OPTIMIZATION**

We proceed by describing the basic GA and PSO algorithms, the use of approximation models, and the hybrid algorithm developed in (14) and used for the numerical results.

**4.1 Genetic Algorithms**

Genetic algorithms are part of a larger class of evolutionary algorithms. They are classified as population based, global search heuristic methods (16; 17). Genetic algorithms are based on biological processes such as survival of the fittest, natural selection, inheritance, mutation, and reproduction. Design points are designated as “individuals” or “chromosomes”. The population evolves towards a smaller fitness value using biological processes applied over a specified number of generations. The outline of the algorithm is below.

1. Initialize population randomly or by seeding using an engineering perspective
2. Evaluate fitness (objective function value)
3. Iterate (produce generation)
	1. Select individuals to reproduce
	2. Perform crossover and mutation
	3. Evaluate the fitness of the new individuals
	4. Replace the worst ranked individuals with the new offspring

During the selection phase, better fit individuals are arranged randomly to form a mating pool on which further operations are performed. Crossover exchanges information between two design points to produce a new point that preserves the best features of both ‘parents’. Mutation prevents the algorithm from terminating prematurely to a suboptimal point and is used to explore the design space.

The algorithm terminates when a prescribed number of generations is reached or when the highest ranked individual’s fitness has reached a plateau. Genetic algorithms are often criticized for their computational complexity and dependence on optimization parameter settings, which are not known a priori. However, if the user is willing to exhaust a large number of function evaluations, the GA can provide insight into the design space and locate initial points for fast, local, single search methods.

**4.1 Particle Swarm Optimization**

As with GAs, the PSO is a population-based, global search heuristic whose technique is inspired by methods of optimization naturally existent in the world. Introduced in 1995 by Kennedy and Eberhart (18), the PSO attempts to simulate the social optimization behavior one might witness when schools of fish or swarms of insects seek food. The PSO models this by encoding the individuals as a set of particles, or points, used to search the design space for a global best location. The particles are given an initial position and velocity within the search space and are permitted only the memories of their personal best position and of either a neighborhood or global best position, depending upon whether by-particle communication is restricted to a subset of the other particles, or free between all particles in the swarm. Based upon these two pieces of dynamically updated information, particles move throughout the space, periodically adjusting their velocities (and thus positions), typically either until a global best location is found or until the velocities within the swarm all reach some critically low value, indicating that convergence has occurred. We outline the PSO algorithm below.

1. Initialize swarm

* 1. Random or seeded swarm
	2. Random initial particle velocities
1. Initialize scores
	1. Evaluate fitness of the particles in the initial swarm
	2. Store initial local and global best scores
2. Iterate over the following until a termination condition is met
	1. Update swarm velocities and positions
	2. Evaluate fitness of the particles at their updated positions
	3. Update local and global best scores

While the combination of a PSO and a GA has proven effective in achieving global optima on a wide range of problems, this atypical amalgam of two population-based techniques is not without drawbacks; most notably, the significant computational expense required for a globally optimal solution. Refinement of the hybrid algorithm to improve efficiency is usually possible, but significant limitations remain. If the application requires a computationally expensive fitness function, the cost associated with these algorithms is exacerbated. One attempt to combat issues with expensive cost functions is to introduce *approximation models* to the hybrid systems.

**4.3 Approximation Models**

Use of approximate models in traditional optimization methods has become an increasingly common technique to improve search performance – especially when fitness functions have high computational cost (13). The idea is to use true function values to build a surrogate or cheaper replacement of the objective function to guide the search phase of the optimization. How the information gained from the surrogate model is incorporated is dependent on the optimization algorithm and is an active area of research. We use Kriging interpolation, a statistical method with origins in geostatistics that uses spatial correlation functions. It extends easily to multiple dimensions, making it well-suited for optimization problems with several parameters (19; 20).

In this work, approximation modeling is implemented with the integration of the MATLAB DACE toolbox (Design and Analysis of Computer Experiments) (21). Readers interested in additional details regarding the theory or the MATLAB implementation should see (20), (22), and (21). We detail how the surrogate aids in the hybrid PSO below.

**4.4 PSO-GA Hybrid: Particle Swarm-Dynamic-Crossover**

In both (10) and (9), the hybridization of the GA and PSO generates two partitions of the total population: one partition is subjected to GA operations, the other to PSO operations. In (9), the partitions are stochastically “mixed” as the hybrid algorithm progresses, whereas in (10), the partitions are completely recombined and reformed after GA and PSO iterative operations are executed. However, neither hybrid design considers that the GA has no means of storing or updating the particle velocities or local-best memories generated by the PSO; as a result, each time the partitions are “mixed” or recombined, the PSO must reinitialize, meaning the most recent particle velocities and memories are lost.

Settles and Soule (11) dealt with this problem by incorporating aspects of the GA into the PSO algorithm at a functional level. Their hybrid design uses a modified form of crossover which addresses the more stringent requirements of a PSO by including an additional equation which defines child velocities as a function of the velocities of its parents. This crossover routine allows the hybrid algorithm to rebuild the swarm population after the iterative step in which the lesser-fit half of the swarm is discarded. However, the discarded points may contain information useful for the optimization. In the PSO-based hybrid algorithm used here we address that potential shortcoming by approaching our incorporation of crossover methods from a different perspective – employing it as a means by which to *improve* the lesser fit particles, rather than one to *replace* them outright. In addition, we incorporate ideas from (14) that are used to improve the efficiency of a standard PSO algorithm. Thus, our hybrid design should address the efficiency issues considered by Settles and Soule and preserve more of the information gathered by the weaker particles in the swarm. Preserving this information should discourage the algorithm from converging too quickly toward possibly local optima.

As with the standard PSO, the hybrid algorithm begins by initializing the swarm particles, velocities, and best-location memories. Also as with the standard PSO, the swarm iterates through a series of steps which update the particle positions, velocities, and fitness values until a terminating condition has been met. Unlike the standard PSO, the hybrid “step” potentially performs crossover once every four iterations. If the current iteration is not tagged for crossover, the hybrid executes exactly as the standard implementation does.

In iterations tagged for crossover, the hybrid step first assesses how close to convergence the algorithm is. An approximate measure of closeness to convergence may be computed by comparing the hypervolume of the smallest hypercube that completely contains the fittest twenty percent of the swarm to that which contains the entire design space. More precisely, it is assumed the progress of the algorithm toward convergence will be proportional to the volume ratio between the top twenty percent-containing hypercube and the design space-containing hypercube. In order to make this form of measurement dimensionally consistent from an absolute standpoint, the final quantity examined is actually the *nth* root of the aforementioned ratio, given an *n*-dimensional problem; formally, that quantity is expressed as:

 , (4)

where [*B*1, *B*2, …, *Bn*] = [*U*1-*L*1, *U*2-*L*2, …, *Un*-*Ln*] defines the hypercube dimensions of the design space bounds, and [*s*1, *s*2, …, *sn*] defines equivalently the dimensions of the hypercube which completely contains the fittest twenty percent of the swarm.

That number – scaled down by an arbitrary constant based upon test optimization performance – acts as the value that determines what fraction of the swarm should undergo crossover; it is formally expressed below:

 . (4)

When the optimizer is far from convergence, *α* is small enough that no part of the swarm is engaged in crossover (and the hybrid functions no differently than the standard PSO); the maximum permitted value of *α* is one-half.

Once *α* has been determined, the crossover routine can begin. Selection of the parents occurs in two stages, creating two parental sets from which the crossover function will draw its pairings. The *α* least-fit particles comprise the pool from which the weak-parent vector will be formed, and the top twenty percent fitness-ranked particles comprise the pool with which the strong-parent vector will be formed. The two parental vectors are constructed using tournament selection on the respective parental pools and input to the mating function.

Mating is accomplished by pairing one weak parent with one strong parent for every new particle. This ensures the retention of information from the weakest fraction of the swarm while promising significant improvement upon that information by merging the data with that of a very fit particle. This often moves the child from his weaker parent toward his stronger one, but propels it with a velocity nearly opposite that of his stronger parent – in effect, drawing the weaker particles in toward the fitter points of the swarm, but sending them off with velocities opposite of those fitter points, maintaining a greater variation in the directions the swarm explores.

With that finished, the remaining (1- *α*) particles undergo standard PSO velocity and position updating, completing the hybrid PSO step. The algorithm then proceeds exactly as with the standard PSO implementation. A sketch of the modified particle swarm algorithm is given below.

Algorithm: Particle Swarm-Dynamic-Crossover

1. Initialize swarm:
	1. Random or seeded swarm.
	2. Random initial particle velocities.
2. Initialize scores:
	1. Evaluate fitness of the particles in the initial swarm.
	2. Store initial local and global best scores.
3. Iterate over the following until a termination condition is met:
4. Check to see if current iteration is eligible for hybrid crossover. If no, set *α* = 0, refer to (g).
5. Evaluate optimization progress, compute crossover fraction *α*.
6. For swarm size *N*, partition least-fit *αN* particles from remainder of swarm
7. Perform tournament selection on least-fit *αN* particles to choose *αN* “mothers” for crossover routine.
8. Perform tournament selection on most-fit 0.20\**N* particles to choose *αN* “fathers” for crossover routine.
9. Generate *αN* new children
10. Execute standard particle swarm update on remaining (1- *α*)*N* original particles.
11. Merge *αN* new children with updated particles.
12. Evaluate fitness of the particles at their updated positions.
13. Update local and global best scores.

The hybrid design was also augmented with a framework for surrogate function management in an effort to further improve performance without adding significant computational cost to the algorithm. The structure of the iterative step of the hybrid algorithm remains almost entirely undisturbed by the inclusion of a surrogate model, with the lone modification being the addition of a pair of matrices to store the histories of all swarm positions and corresponding fitness values used to construct the response surface.

Functionality, however, differs for algorithm iterations calling for surrogate aid. An additional "surrogate step" is taken, in which first the swarm history is used to construct an up-to-date surrogate model. That model is then used with a DACE predictor to optimize a copy of the current swarm. The fittest ten percent of the surrogate search results are temporarily saved and subsequently re-evaluated using the true fitness function in preparation for assimilation into the current swarm.

Certain precautions taken to limit how drastically and immediately the surrogate results impact the progress of the search. These aside, the algorithm then proceeds to merge the top surrogate results over the least fit particles of the current swarm population, ensuring that original particles of the swarm are displaced only once they are confirmed to be less fit than the surrogate candidates; any surrogate results not incorporated at this time are discarded. With the current swarm now fully updated by the surrogate step, the hybrid algorithm proceeds exactly as it would in its original form.

**5-NUMERICAL RESULTS**

The polymer fluid that transports debris particles was assumed to have a power law index (*n*) of 0.9; the density of the melt itself was assumed to be 0.00135 kg/cm3. The debris densities for the three distinct debris material types were assumed to be 0.0089 kg/cm3, 0.0040 kg/cm3, and 0.0010 kg/cm3, respectively; their relative concentrations were assumed to be 1.5 ppm, 0.5 ppm, and 1.0 ppm, respectively.

The extrusion filter was assumed to be a sintered metal filter, so a parameter exists in the model that represents the average size of a filter particle. A linear relationship was assumed to exist between the filter pore diameter and the filter particle diameter. The filter is configured to have a circular cross-section one inch in diameter; the two distinct layers comprising the filter are each set to have a thickness of 0.5 centimeters.

Though the intention of a properly designed two-layer filter is to trap the coarse particles in the top layer, and finer particles in the bottom, the bound constraints on *x* were not configured to guide the various optimizers toward this configuration. The constraint set Ω was defined as follows for all numerical experiments presented:

 (6)

The optimization terminated once a threshold fitness value of 1.59e5 Pa/hr was reached. The number of true (i.e., non-surrogate) function evaluations required to reach that level of fitness was returned and appears in Table 1 as *Nopt.*

The data corresponding to the standard PSO is represented as the “non-hybrid function”, or *Fnh*; the particle swarm-dynamic crossover hybrid, PSO surrogate, and particle swarm-dynamic crossover hybrid surrogate methods are abbreviated similarly as *Fh*, *Fs*, and *Fsh*, respectively. Columns two through five correspond to the optimal values of (*1*, *dp1*, *2*, *dp2*) produced by each optimization algorithm; the remaining columns show the resultant filter lifetimes (*t*), masses of total escaped debris (*ξ*), and the average rate of change of the pressure drop across the filter (*m*).

**Table 1: Bounded PSO algorithm results with the additive penalty-based objective function**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *F* | *η*1 | *dp*1(*μm*) | *η*2 | *dp*2(*μm*) | *t*(*hrs*) | *ξ*(*1e-5kg*) | *m*(*1e5 Pa/hr*) | *Nopt* |
| *Fnh**Fh**Fs**Fsh* | 0.58790.60530.59350.6555 | 48.861450.092150.441437.9788 | 0.60990.56100.42040.4904 | 24.020023.321923.094523.3892 | 104.6000113.3000110.700098.0000 | 7.80138.01257.86415.3102 | 1.58351.53741.52801.7807 | 280220166300 |

For illustration purposes, the optimizers were configured to run until their function budgets were consumed. The filter lifetimes returned by each of the four approaches were all quite strong, with the non-surrogate hybrid beating even the surrogate non-hybrid; it is worth nothing, however, that the additional debris permitted through the filter of the hybrid result was sufficiently more than that through the surrogate non-hybrid that it ultimately rated as a slightly less fit design. Figure 2 shows the best objective function value for each optimization iteration (which involves evaluating the objective function at each swarm particle).

**Figure 2: Comparison of optimization histories**

**6-CONCLUSIONS**

We have demonstrated the performance of a PSO-GA hybrid optimization scheme to study the design of an extrusion filter. The hybrid approach identified a design with the longest lifetime, followed closely by the surrogate PSO and the standard PSO. For this work, the hybrid surrogate approach was not competitive, although it did succeed in identifying a design with a larger pore diameter in the top layer, as did all the methods, which was a drawback of the single search approach (7). The surrogate PSO approach returned an optimal filter design whose lifetime was only 3 hours less than that returned from the hybrid approach while using 60 fewer calls to the simulator. Each simulation could take from one to three hours, meaning that these 60 calls represent a significant savings in computational effort. The success of the surrogate PSO approach also supports the use of approximation models for this application. In addition, none of the methods presented here required an initial guess to start the optimization algorithm. This is another improvement over the single search approach from (7), in which the design obtained via the implicit filtering algorithm was highly dependent on the initial iterate.

**REFERENCES**

1. Center for Advanced Engineering Fiber and Films. [Online] 2009. http://www.clemson.edu/caeff.

2. *Prediction of Pressure Drop for the Flow of Polymer Melts Through Sintered Metal Filters.* **Edie, D.D. and Gooding, C.H.** 1, 1985, Industrial and Engineering Chemistry Process Design and Development, Vol. 24, pp. 8-12.

3. *Non-Newtonian Flow and Debris Deposition in an Extrusion Filter Medium.* **Seyfzadeh, B., Zumbrunnen, D.A. and Ross, R.A.** 2001. Plastics - The Lone Star, Society of Plastic Engineers Proceedings. pp. 340-344.

4. *Design Analysis of Polymer Filtration Using a Multi-objective Genetic Algorithm.* **Fowler, K.R., et al.** 4, 2008, Separation Science and Technology, Vol. 43, pp. 710-726.

5. *An Implicit Filtering Algorithm for Optimization of Functions with Many Local Minima.* **Gilmore, P. and Kelley, C.T.** 1995, SIAM Journal of Optimization, Vol. 5, pp. 269-285.

6. *A Simulation-based Optimization Approach to Polymer Extrusion Filter Design.* **Fowler, K.R., et al.** 3, 2009, Filtration, Vol. 9, pp. 224-230.

7. *Understanding the Effects of Polymer Extrusion Filter Layering Configurations Using Derivative-free Optimization.* **Fowler, K.R., Jenkins, E.W. and LaLonde, S.M.** 2009, Optimization and Engineering. doi: 10.1007/s11081-009-9096-0.

8. *A Taxonomy of Hybrid Metaheuristics.* **Talbi, E.G.** 5, 2002, Journal of Heuristics, Vol. 8, pp. 541-564.

9. *An Improved GA and a Novel PSO-GA-based Hybrid Algorithm.* **Shi, X.H., et al.** 2005, Information Processing Letters, Vol. 93, pp. 255-261.

10. *A New Hybrid Technique for the Optimization of Large-Domain Electromagnetic Problems.* **Grimaldi, E., et al.** 2005. Proceedings of the XXVIIIth General Assembly of the International Union of Radio Science.

11. *Breeding Swarms: A GA/PSO Hybrid.* **Settles, M. and Soule, T.** 2005. Proceedings of the Genetic and Evolutionary Computation Conference. pp. 161-168.

12. *Improvement of Genetic Algorithm Using PSO and Euclidean Data Distance.* **Kim, D.H.** 3, 2006, International Journal of Information Technology, Vol. 12, pp. 142-148.

13. *A Rigorous Framework for Optimization of Expensive Functions by Surrogates.* **Frank, P.D., et al.** 17, 1998, Structural Optimization, Vol. 17, pp. 1-13.

14. **McClune, B.** *New Hybrid and Surrogate Techniques for Simulation-based Optimization of a Polymer Extrusion Filter.* Clarkson University. 2009. Masters Thesis.

15. **Bird, R.B., Stewart, W.E. and Lightfoot, E.N.** *Transport Phenomena.* New York : John Wiley & sons, 1960.

16. *ADIFOR: Generating derivative codes from Fortan programs.* **Bischof, C., et al.** 1, 1992, Scientific Programming, Vol. 1, pp. 1-29.

17. *Optimum aerodynamic design using the Navier-Stokes equations.* **Jameson, A., Pierce, N.A. and Martinelli, L.** 1997. AIAA Aerospace Sciences Meeting & Exhibit.

18. *Particle swarm optimization.* **Kennedy, J. and Eberhart, R.C.** Piscataway, NJ : IEEE Service Center, 1995. Proceedings of the IEEE International Conference on Neural Networks, IV. pp. 1942-1948.

19. *Computationally efficient calibration of WATCLASS hydrologic models using surrogate optimization.* **Kamali, M., Ponnambalam, K. and Soulis, E.D.** 2007, Hydrology and Earth Systems Sciences Discussion, Vol. 4, pp. 2307-2321.

20. *Optimal aeroacoustic shape design using the surrogate management framework.* **Marsden, A.L., et al.** Nov. 23-25, 2003. American Physical Society, Division of Fluid Dyanmics, 56th Annual Meeting.

21. **Lophaven, S.N., Nielsen, H.B. and Sondergaard, J.** DACE - A MATLAB Kriging Toolbox, Version 2.0. 2002.

22. *Efficient global optimization of expensive black-box functions.* **Jones, D.R., Schonlau, M. and Welch, W.J.** 4, 1998, Journal of Global Optimization, Vol. 13, pp. 455-492.

23. *Modeling of Debris Deposition in an Extrusion Filter Medium.* **Cox, C.L., Jenkins, E.W. and Mucha, P.J.** 2005. Proceedings of the 21st Annual Meeting of the Polymer Processing Society.