Guide to Utmost IV Optimizers

Utmost IV currently includes 6 optimizers for parameter extraction. Selecting an optimizer can sometimes be confusing. This article reviews the optimizers and attempts to provide some guidelines in selecting an appropriate one.

Optimization tasks are usually divided into two major categories: local and global. Local optimization assumes that there is a single minimum of a cost function which needs to be located. Locating this minimum is relatively simple – starting from some initial parameter values the optimizer successively finds search directions and tries to locate the point with lowest cost function along the search directions. The choice of good search directions depends on the initial parameter values and the methods the optimizer is using to calculate the directions. In contrast, a global optimization problem assumes that there may be multiple minima to the cost function and attempts to locate the global minimum, which is the minimum with lowest value of the cost function. Global optimization is much more difficult because of the many possibilities that exist. To make it feasible, global optimizers normally use randomization (or stochastic) methods in generating possibilities.

A further distinction between optimization tasks is whether or not they assume that the cost function has first, and sometimes second, derivatives. Using derivatives in locating a minimum may considerably shorten optimization time, particularly for local optimizers.

The derivatives are often calculated numerically. However, in practice, derivatives of model parameters may not exist or are too expensive to calculate. More importantly for optimization tasks that rely on simulators, such as SmartSpice, for calculating the cost function, the numerical noise generated by the simulator makes it difficult or impossible to calculate numerical derivatives. Numerical noise is the result of simulators using some internal algorithmic parameters (e.g. stepsizes, number of iterations etc.) to reach a numerical solution. Changing model parameters even by a little amount may cause the algorithmic parameters to adapt to this change. The modification of algorithmic parameters adds some “noise” which may complicate or prohibit accurate calculations of numerical derivatives needed for optimization.

Now back to the optimizers included with Utmost IV: Table 1 summarizes the main characteristics of each optimizer.

There are two local optimizers; one uses the Levenberg-Marquardt (LM) [1], [2] algorithm and the other the Hooke-Jeeves (HJ) [3] algorithm. LM is a local optimization algorithm that uses first derivatives. It is particularly good for optimizations in which the first derivative matrix, the Jacobian, is nearly singular which is often the case in nonlinear models. Because it uses derivatives, it can be quite fast. For simple models, or when the number parameters to optimize is small, the LM optimizer may work very well. The downside of LM is that it requires derivatives which may be difficult or impossible to calculate in practice because of numerical noise or because they are too expensive. The HJ optimizer, which does not require derivatives, is provided as an alternative to LM. HJ belongs to a class of pattern-search algorithms which, instead of using derivatives, look for patterns in search directions which are more promising in reducing the cost function.

Optimizer	Local/Global	Uses derivatives	Stochastic
Levenberg-Marquardt	Local	Yes	No
Hooke-Jeeves	Local	No	No
Simulated Annealing	Global	No	Yes
Parallel Tempering	Global	No	Yes
Genetic Algorithm	Global	No	Yes
Differential Evolutions	Global	No	Yes

Table 1. Comparison of Utmost IV Optimizers.

The other 4 optimizers in Utmost IV are global optimizers and are all using randomization (stochastic) methods in generating parameter values to examine. Because of their stochasticity, repeating the same optimization with the same optimizer, may result in locating a different minimum. This is an advantage of randomization, as a second attempt may succeed even if the first failed.

The Simulated Annealing (SA) algorithm[4] mimics a physical system in equilibrium at some temperature. The cost function replaces the energy, and the "temperature" is just a control parameter in the optimization. At higher "temperatures", it is more probable to find parameter values which produce high cost. This is advantageous at initial stages of the optimization as it causes a more extensive search. As the optimization proceeds, the system cools down and the search becomes more focused around the global minimum.

SA is a robust algorithm, which may be very powerful in highly non-linear models with many parameters. The downside of SA is that can sometime be slow. The Parallel Tempering (PT) algorithm [5] is similar to SA, but instead of gradually reducing the temperature, it keeps several copies of the system at different temperatures. Once in a while, these copies are exchanged, which is useful for pulling the system out of local minima and accelerating convergence. PT is particularly useful in situations where there are many local deep minima where it can outperform the simpler SA algorithm.

The last two optimizers belong to a class of evolutionary algorithms. Unlike SA & PT, which mimic a physical system, the evolutionary algorithms mimic instead evolution of biological systems. In this class of algorithms, there is a "population" which includes several solutions ("chromosomes") to the optimization problem. At each "generation" this population goes through a process of selection, in which some of the members are being dropped because of bad "fitness" (which is closely related to the cost function) and those who survive go through a process of "mating" and "mutations". In mating, parts of two selected members (the parents) are combined into one or more "children". This ensures that the global optimization is not stuck in some local minimum but continue the search for the global optimum. In mutation, individual components of each member are being modified. For the optimization process, mutations make sure that solution near a minimum keep converging toward that minimum. The evolutionary algorithms are stochastic: selection, mating and mutations all involve random choices, much like the biological systems they model.

In Utmost IV, the two members of the evolutionary algorithm class are a Genetic Algorithm (GA) and Differential Evolution (DE). When the original genetic algorithms were first invented [6], the representation of solutions was a string of bits or some other discrete quantities, because of the biological similarity. These representations are more appropriate for discrete (combinatorial) optimization. More recently[7], researchers extended representations of mating and mutation operators to real valued quantities.

The GA algorithm in Utmost IV incorporates several recent developments.

Unlike genetic algorithms, Differential Evolution [8] was developed for real-valued representations and is therefore suitable for optimizing model parameters. Like other evolutionary algorithms each generation includes several solutions. The main difference between DE and GA is in the way members of each generation are selected and modified. Unlike GA, which mate and mutate, DE first calculates the difference between two (or more) randomly selected members and then stochastically adds the differences to another randomly selected member.

DE and GA can often be very fast global optimizers. However, they may depend on tuning of several algorithmic parameters to achieve their speed. DE depends on less algorithmic parameters than GA.

References

[1] Levenberg, K A method for the solution of Certain Non-linear Problems in Least Squares, Quart. Appl. Math. 2, 164-168, 1944.

[2] Marquard, D An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM J. Appl. Math. 11, 431-441, 1963.

[3] R. Hooke, T. A. Jeeves, Direct Search Solution of Numerical and Statistical Problems, Journal of ACM, 8, pages 212-229, 1961.4]

[4] S. Kirkpatrick, Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing, Science, 220 (671-680), 1983.

[5] C. J. Geyer, in Computing Science and Statistics Proceedings of the 23rd Symposium on the Interface, American Statistical Association, New York, 1991, p. 156.

[6] J. Holland, Adaptation In Natural and Artificial Systems, University of Michigan Press, 1975.

[7] A.H. Wright, Genetic Algorithms for Real Parameter Optimization, in G. Rawlins (Ed.), Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, 1991. Z. Michalewicz, G. Nazhiyath, M. Michalewicz, A Note on the Usefulness of Geometrical Crossover for Numerical Optimization Problems, Proceedings of the 5th Annual Conference on Evolutionary Programming, San Diego, CA, 29 February-3 March, MIT Press, Cambridge, MA, 1996.

[8] Storn, K. Price, Differnetial Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, Kluwer Academic Publishers, 11(341-359), 1997; K. Francken, G.E. Gielen, M. S. J. Steyaert, An Efficient, Fully Parasitic-aware Power Amplifier Design Optimization Tool, IEEE Trans. On Circuits and Systems, 52, 8, 1526-1534, 2005.