DESIGN OPTIMIZATION WITH EXTERNAL SIMULATORS

by J.W. Bandler, R.M. Biernacki and S.H. Chen

Optimization Systems Associates Inc.
P.O. Box 8083, Dundas, Ontario
Canada L9H 5E7

ABSTRACT

We review the concepts involved in design optimization. We formulate design as an abstract optimization problem, regardless of the nature of the object being designed. We show how the error functions for design goals are typically defined and discuss ways of combining them into a single objective function. We also present a novel and powerful approach to CAD software architecture suitable for distributed calculations and interactions between independent programs. This approach is particularly useful for independent software development and for maintenance of large software systems such as those dedicated to field calculations.

INTRODUCTION

Design optimization is a powerful computational tool enabling designers to adjust designable parameters in order to meet design specifications (see, for example, [1-4]). The nature of the designed object is irrelevant to the optimizer. However, the computer simulation of the object must be available. The computer simulator should provide the means for processing a number of input parameters, some of which are (directly or indirectly) designable, into the corresponding set of responses. Although not absolutely necessary [5], it is extremely desirable that the simulator is capable of calculating partial derivatives (gradient) of the responses w.r.t. the designable parameters. The simulator should be efficient enough for repeated calculations. It may be called tens, hundreds, or even thousands of times during optimization.

Efficiency of the algorithms as well as organization of software are of utmost importance. Software modularity must be facilitated and modules of different origin need to be accommodated. For example, advanced, state-of-the-art optimization routines must interact with field simulators, developed separately, possibly in a different language and without optimization as the objective. We describe some of our new developments in the area of open architecture software systems which facilitate these requirements and are particularly suitable for design optimization. A new technique called IPPC (inter-program pipe communication) allows for high speed numerical interaction between independent programs.

SIMULATION, SPECIFICATIONS AND ERROR FUNCTIONS

The response functions that can be of interest to the designer may involve a combination of frequency domain responses, time domain responses, space domain responses, frequency spectra of periodic functions, and their functions such as power, etc. "Of interest" means that design specifications are imposed on the responses. A specification is typically imposed on a range

of domain values. This leads to an infinite number of specifications and it becomes necessary to discretize the domain and consider only a finite subset of representative frequency, time or space points. After discretization, the jth specification can be denoted by S_{uj} or $S_{\ell j}$ if it is an upper or a lower specification, respectively.

In order to formulate an objective function for design optimization the object is simulated at the same frequency or time points at which the upper and/or lower specifications are selected by discretization. The corresponding responses are denoted by $R_j(\phi)$ and the error vector $e(\phi)$ is defined as

$$e(\phi) = [e_1(\phi) \quad e_2(\phi) \quad . \quad . \quad . \quad e_M(\phi)]^T \tag{1}$$

where the individual errors $e_j(\phi)$ are

$$e_j(\phi) = R_j(\phi) - S_{uj} \tag{2}$$

or

$$e_j(\phi) = S_{\ell j} - R_j(\phi). \tag{3}$$

ϕ is the vector of designable parameters and M is the total number of errors. The negative error values indicate that the corresponding specifications are satisfied. For positive error values the corresponding specifications are violated. The acceptability region in the parameter space is defined as

$$A = \{\phi \mid e_j(\phi) < 0 \quad j = 1, 2, \ldots, M\}. \tag{4}$$

Clearly, all specifications are satisfied if the designable parameters fall into A, and at least one specification is violated if that point falls outside the acceptability region A.

OBJECTIVE FUNCTIONS AND ALGORITHMS

For the purpose of optimization all the errors $e_j(\phi)$ have to be combined into a single objective function. The three important types of the objective function are minimax, ℓ_1 and ℓ_2 (least squares). The generalized ℓ_p function $v(\phi)$ from $e(\phi)$ [6,7] takes the form

$$v(\phi) = \begin{cases} \left[\sum_{j \in J(\phi)} (e_j(\phi))^p\right]^{1/p}, & \text{if } \phi \notin A, \quad (5a) \\ -\left[\sum_{j=1}^{M} (-e_j(\phi))^{-p}\right]^{-1/p}, & \text{if } \phi \in A, \quad (5b) \end{cases}$$

where

$$J(\phi) = \{j \mid e_j(\phi) \geq 0\}. \tag{6}$$

Some variations of this function include: (a) one-sided ℓ_p function where (5b) is set to zero, and (b) the ℓ_p norm where only (5a) is used and the summation is over the absolute values of e_j's for all $j = 1, 2, \ldots, M$. The minimax function corresponds to $p \to \infty$ and can simply be expressed as

$$v(\phi) = \max_j (e_j(\phi)). \tag{7}$$

The minimax is the objective function of choice for performance driven design optimization and leads to equi-ripple solutions. The ℓ_2 norm or one-sided ℓ_2 function are also commonly used for performance driven design optimization. The ℓ_1 function is uniquely useful in modeling and in yield optimization. Finally, it is worth mentioning that non-negative multiplicative weighting factors can be applied in (6) and (7) to individual errors. Specialized, robust algorithms exist for minimization of each of the aforementioned objective functions, e.g., [8-12].

OPEN ARCHITECTURE OPTIMIZATION SOFTWARE SYSTEMS

We will now present a recent, advanced technique for open software architecture called IPPC (inter-program pipe communication) facilitating high speed numerical interaction between independent programs. It allows highly repetitive data communication between totally independent programs. It also allows an unlimited number of non-predetermined and new software modules to be added to existing software systems with no modification, no re-compilation and no re-linking of the existing systems. Therefore, a software user can add new modules to an existing IPPC-based system, allowing the existing system's optimizers, statistical drivers, etc., to interact iteratively with his own modules. The user's modules are separate executables programs. Thus, independent development testing and execution of new code are facilitated. The confidentiality of the user's program is also totally secured.

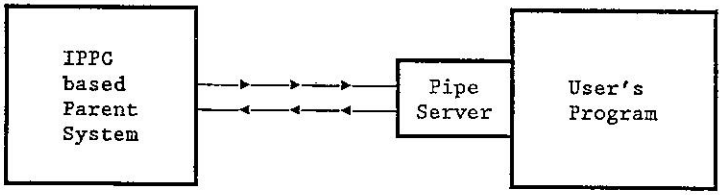

Fig. 1. Schematic diagram of IPPC between two independent programs

The communication, in its basic form, allows the user to combine two application programs: the parent and the child. It requires only minor modification to the child program and no modification to the IPPC-based parent program. A small IPPC server is the vehicle for communication between the two programs. As shown in Fig. 1, the user attaches the IPPC server to his or her program to generate a pipe-ready version. During simulation or optimization involving the child, the parent executes the child as a separate process. In forking the child process, two inter-process pipes are created. The two-way communication is established by using each pipe to transfer data one-way. Communication between one parent and several children and grandchildren is possible. Experiments have been conducted on our new CAD system OSA90™ [13]. The overhead CPU cost in practical situations is found to be negligible - it typically adds only about 1% to the conventional approach of subroutine calls.

[1] J.E. Dennis, Jr. and J.J. Moré, "Quasi-Newton methods, motivation and theory", *SIAM Review*, vol. 19, 1977, pp. 46-89.

[2] J.W. Bandler and M.R.M. Rizk, "Optimization of electrical circuits", *Math. Programming Study*, vol. 11, 1979, pp.1-64.

[3] R.K. Brayton, G.D. Hachtel and A.L. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design", *Proc. IEEE*, vol. 69, 1981, pp. 1334-1362.

[4] J.W. Bandler and S.H. Chen, "Circuit optimization: the state of the art," *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 424-443.

[5] J.W. Bandler, S.H. Chen, S. Daijavad and K. Madsen, "Efficient gradient approximations for nonlinear optimization of circuits and systems", *Proc. IEEE Int. Symp. Circuits and Systems* (San Jose, CA), 1986, pp. 964-967.

[6] J.W. Bandler and C. Charalambous, "Practical least pth optimization of networks," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-20, 1972, pp. 834-840.

[7] C. Charalambous, "Nonlinear least pth optimization and nonlinear programming", *Math. Programming*, vol. 12, 1977, pp. 195-225.

[8] J. Hald and K. Madsen, "Combined LP and quasi-Newton methods for minimax optimization", *Math. Programming*, vol. 20, 1981, pp. 49-62.

[9] J. Hald and K. Madsen, "Combined LP and quasi-Newton methods for nonlinear ℓ_1 optimization", *SIAM J. Numerical Analysis*, vol. 22, 1985, pp. 68-80.

[10] J.W. Bandler, W. Kellermann and K. Madsen, "A superlinearly convergent minimax algorithm for microwave circuit design", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1519-1530.

[11] J.W. Bandler, W. Kellermann and K. Madsen, "A nonlinear ℓ_1 optimization algorithm for design, modelling and diagnosis of networks", *IEEE Trans. Circuits and Systems*, vol. CAS-34, 1987, pp.174-181.

[12] J.W. Bandler, S.H. Chen and K. Madsen, "An algorithm for one-sided ℓ_1 optimization with application to circuit design centering," *Proc. IEEE Int. Symp. Circuits Syst.* (Espoo, Finland), 1988, pp. 1795-1798.

[13] *OSA90*™, Optimization Systems Associates Inc., Dundas, Ontario, Canada, 1990.