SMX—A Novel Object-Oriented Optimization System

M.H. Bakr, J.W. Bandler, Q.S. Cheng, M.A. Ismail and J.E. Rayas-Sánchez

McMaster University, Hamilton, ON, Canada L8S 4K1, www.sos.mcmaster.ca

Abstract — **The novel, object-oriented SMX optimization system implements the powerful Surrogate Model-based Space Mapping (SMSM) algorithm, which is automated for the first time. SMX is capable of driving commercial EM, circuit and user-supplied simulators. The SMX architecture permits easy integration of new simulators and optimization tools. The power of SMX is illustrated through an optimization example exploiting Momentum.**

I. INTRODUCTION

Space Mapping (SM) aims at efficient engineering optimization [1]. It establishes a mapping between the parameter spaces of a "fine" model, e.g., an electromagnetic (EM) simulator, and a coarse model. Aggressive Space Mapping efficiently directs the optimization steps.

In the recent SMSM approach [2], a surrogate [3] of the fine model is iteratively used to solve the original design problem. This surrogate model is a convex combination of a mapped coarse model and a linearized fine model. It can exploit a frequency-sensitive mapping.

The state of the art SMX engine implements the SMSM algorithm. Object-Oriented Design (OOD) abstracts the basic behavior of the models and optimizers modules. A universal parameter setting and results retrieval method is utilized for all simulators and optimizers. The SMX architecture integrates these modules.

Another advantage of OOD is reusability and extendability. SMX can support a number of EM and circuit simulators. Here, the basic functionality of simulators and optimizers is abstracted in the two basic classes Simulator and Optimizer. Many commercial simulators and optimizers can be derived from these classes.

The SMX system is described in the Unified Modeling Language (UML) [4]. Using this language, a complicated system can be decomposed into relatively independent

small objects without losing readability and intuitiveness. The structure of each object can be represented in UML.

SMX takes full advantage of the multi-thread capability of the Microsoft Windows operating system [5]. The user-friendly interface responds smoothly while the SMX core is running in a different thread in the background. Synchronization and communication between threads are properly arranged. SMX is capable of optimizing while showing intermediate results and interacting with the user.

II. THE SMSM ALGORITHM

We denote the fine model responses at a point $x_f \in \Re^{n \times 1}$ and frequency ω by $\boldsymbol{R}_f(x_f, \omega) \in \Re^{N_r \times 1}$. These responses may include the real and imaginary parts of S_{11}, etc. The vector $\boldsymbol{R}_f(x_f) \in \Re^{m \times 1}$ denotes the responses at all the N_ω simulation frequencies where $m = N_r N_\omega$. The vector $x_c \in \Re^{n \times 1}$ and $\boldsymbol{R}_c(x_c) \in \Re^{m \times 1}$ denote a coarse model point and the corresponding coarse model responses, respectively. The coarse model responses at a frequency ω_c are similarly denoted by $\boldsymbol{R}_c(x_c, \omega_c) \in \Re^{N_r \times 1}$.

The fine model is optimized indirectly by using a surrogate [3] in the form of a convex combination of a mapped coarse model (MCM) and a linearized fine model (LFM). We denote the surrogate model in the *i*th iteration by $\boldsymbol{R}_s^{(i)}(x_f^{(i)}) \in \Re^{m \times 1}$. The *i*th iteration solves

$$h^{(i)} = \arg \min_{h^{(i)}} \ U(\boldsymbol{R}_s^{(i)}(x_f^{(i)} + h^{(i)})), \ \left\| h^{(i)} \right\| \leq \delta^{(i)} \qquad (1)$$

where $U(\boldsymbol{R}_s^{(i)}(x_f^{(i)} + h^{(i)}))$ is the value of the objective function evaluated using the surrogate model at $x_f^{(i)} + h^{(i)}$. $\delta^{(i)}$ is the trust region radius. The surrogate model is given by

$$\boldsymbol{R}_s^{(i)}(x_f) = \lambda^{(i)} \boldsymbol{R}_m^{(i)}(x_f) + (1 - \lambda^{(i)})(\boldsymbol{R}_f(x_f^{(i)}) + J_f^{(i)} \Delta x_f),$$

$$\Delta x_f = x_f - x_f^{(i)}, \ \lambda^{(i)} \in [0, 1], \qquad (2)$$

$J_f^{(i)} \in \Re^{m \times n}$ is an approximation to the Jacobian of fine model responses at $x_f^{(i)}$. The LFM in (2) ensures that the algorithm will work if the coarse model is poor or wrong.

The MCM $\boldsymbol{R}_m^{(i)}(x_f)$ utilizes

$$\boldsymbol{R}_m^{(i)}(x_f, \omega_j) \approx \boldsymbol{R}_f(x_f, \omega_j) \qquad (3)$$

$$R_m^{(i)}(\boldsymbol{x}_f, \omega_j) = \boldsymbol{R}_c(\boldsymbol{P}^{(i)}(\boldsymbol{x}_f, \omega_j), P_\omega^{(i)}(\boldsymbol{x}_f, \omega_j)) \qquad (4)$$

a linear frequency mapping, where $j=1, 2, \ldots, N_\omega$,

$$\begin{bmatrix} \boldsymbol{P}^{(i)}(\boldsymbol{x}_f, \omega_j) \\ P_\omega^{(i)}(\boldsymbol{x}_f, \omega_j) \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}^{(i)} & \boldsymbol{s}^{(i)} \\ \boldsymbol{t}^{(i)T} & \sigma^{(i)} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_f \\ \omega_j \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}^{(i)} \\ \gamma^{(i)} \end{bmatrix} \qquad (5)$$

The parameters $\boldsymbol{B}^{(i)} \in \mathfrak{R}^{n \times n}$, $\boldsymbol{s}^{(i)} \in \mathfrak{R}^{n \times 1}$, $\boldsymbol{t}^{(i)} \in \mathfrak{R}^{n \times 1}$, $\boldsymbol{c}^{(i)} \in \mathfrak{R}^{n \times 1}$, $\sigma^{(i)} \in \mathfrak{R}^{1 \times 1}$ and $\gamma^{(i)} \in \mathfrak{R}^{1 \times 1}$ are the mapping parameters. ω_j is the jth simulation frequency, $j=1, 2, \ldots, N_\omega$. Thus, a fine model point $\boldsymbol{x}_f$ and frequency ω_j correspond to a coarse model point $\boldsymbol{P}^{(i)}(\boldsymbol{x}_f, \omega_j)$ and coarse model frequency $P_\omega^{(i)}(\boldsymbol{x}_f, \omega_j)$.

The MCM should approximate the fine model over a region of fine model parameters and frequency. The mapping parameters are obtained through the optimization procedure

$$[\boldsymbol{B}^{(i)}, \boldsymbol{s}^{(i)}, \boldsymbol{t}^{(i)}, \sigma^{(i)}, \boldsymbol{c}^{(i)}, \gamma^{(i)}] =$$
$$\arg \min_{\boldsymbol{B}, \boldsymbol{s}, \boldsymbol{t}, \sigma, \boldsymbol{c}, \gamma} \left\| [\boldsymbol{e}_1^T \quad \boldsymbol{e}_2^T \quad \ldots \quad \boldsymbol{e}_{N_p}^T]^T \right\| \qquad (6)$$

$$\boldsymbol{e}_k = \boldsymbol{R}_m(\boldsymbol{x}_f^{(k)}) - \boldsymbol{R}_f(\boldsymbol{x}_f^{(k)}) \quad \forall \, \boldsymbol{x}_f^{(k)} \in V^{(i)} \qquad (7)$$

where $V^{(i)}$ is a set of fine model points of cardinality $|V^{(i)}| = N_p$. $V^{(i)}$ is mainly constructed from previously simulated fine model points as discussed in detail in [2].

III. SMX ARCHITECTURE

SMX automates the algorithm and drives EM/circuit simulators. Object-oriented design is employed to decompose both system and algorithm into independent modules (objects). The module or object is an instance of a certain class. Each module carries out certain functionality and includes data structures describing the properties of the object. Using the encapsulation concept, the SMX system is decomposed into 6 modules, as shown in Fig. 1.

The user interface and SMX engine run in two separate threads concurrently. The user chooses the simulators and setup problem specifications through a user interface. The interface initiates the starting point $\boldsymbol{x}_c^{(0)}$, the constraints and the control signals for the coarse and fine models. The SMX engine performs optimization and returns the progress, the current status, responses $\boldsymbol{R}_f$ and $\boldsymbol{R}_c$ etc., to the user interface. The user interface feeds back the optimization status such as objective function, designable parameters and critical mapping parameters as $\boldsymbol{B}^{(i)}$, $\boldsymbol{s}^{(i)}$, $\boldsymbol{t}^{(i)}$, $\sigma^{(i)}$, $\boldsymbol{c}^{(i)}$ and $\gamma^{(i)}$ in graphical and numerical format to the user. The engine can optimize a model using either classical optimization methods such as gradient-based minimax or the state-of-the-art SMSM algorithm [2].
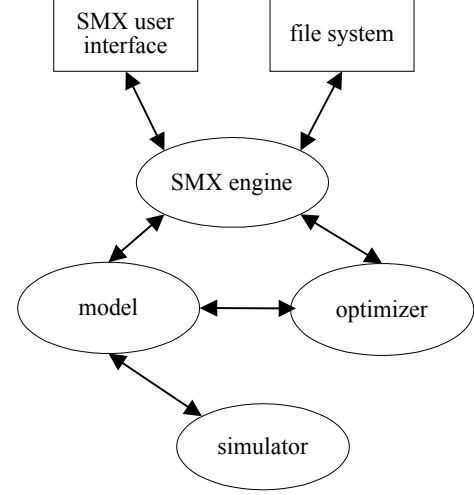

Fig. 1.   The modules of SMX.

IV. ALGORITHM CORE: SMX ENGINE

The SMX engine is abstracted as the SMX_Engine class. After setup, the coarse model is optimized by the member function OptimizeCoarseModel from the starting point $\boldsymbol{x}_c^{(0)}$. This function uses m_pCoarseModelOptimizer, a pointer to a minimax optimizer object to obtain $\boldsymbol{x}_c^*$. Then the member function OptimizeSurrogate is called to optimize the surrogate model $\boldsymbol{R}_s^{(i)}(\boldsymbol{x}_f)$ starting from the optimized coarse model. The Huber optimizer is used for parameter extraction (6) in OptimizeSurrogate. To carry out space mapping, three base classes, Optimizer, Simulator and Model, are abstracted and built.

The Optimizer base class is an abstract class. It provides the interface for standard optimization routines. With override of optimization routines, additional parameter setup and objective function, the Huber, Minimax or other optimizer classes can be derived from Optimizer. Some of the important functions in Optimizer are GetNorm, GetErrors, FDF and SetConstraintMatrix. Different optimizers use different norms as their objective functions. The purely virtual function GetNorm is overridden to obtain the appropriate norm. FDF gets the error values and their derivatives by perturbation. It calls GetErrors to evaluate the error $\boldsymbol{e}_k$ used for (7), as well as for minimax design optimization. SetConstraintMatrix sets constraints in matrix format. The inheritance relation of Optimizer is shown in Fig. 2.
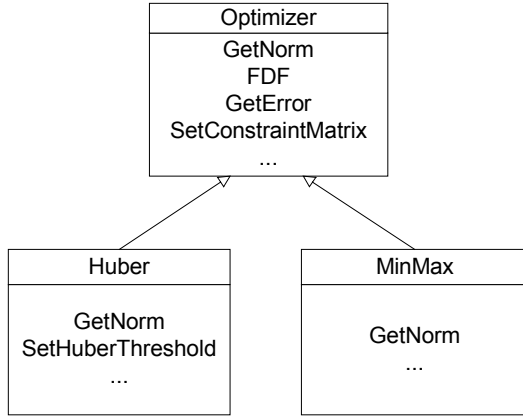
Fig. 2.   Illustration of the derivation of basic optimizer class.

Parameter	$x_f^{(1)}$	$x_f^{(4)}$
L_1	187.50	185.55
L_2	198.84	191.71
L_3	187.91	185.82
S_1	20.04	21.03
S_2	98.08	99.44
S_3	100.90	114.21
all values are in mils		

Similar to Optimizer, the Simulator class is a base class for different simulators. Commercial simulators and user defined simulators are derived classes. Interface functions are overridden for each new derived simulator class. Additional parameters may also be added. OSA90/hope^{TM} [6] and Agilent Momentum^{TM} [7] are commercial simulators currently derived from the Simulator class.

The SMX_Engine utilizes SurrogateModel which is derived from a base Model class. The Model class functions as a wrapper of a simulator. The responses are obtained independent of the simulator. Obviously, Simulator is one of the Model members. The Model class sends data to the simulator and retrieves responses from it. Since Optimizer needs normalized parameters, scaling factors are added.

V. HTS FILTER EXAMPLE

We consider two cases of the HTS filter problem [8]. In Case 1, the "coarse" and "fine" models are both empirical models of OSA90/hope, the former uses ideal open stubs while the latter uses empirical models. Slightly changed design specifications (w.r.t. [8]) are

$$|S_{21}| \le 0.05 \text{ for } \omega \le 3.961 \text{ GHz and } \omega \ge 4.099 \text{ GHz}$$

$$|S_{21}| \ge 0.95 \text{ for } 4.008 \text{ GHz} \le \omega \le 4.058 \text{ GHz}$$

The designable parameters are the lengths L_1, L_2 and L_3 of the coupled lines and their separation S_1, S_2 and S_3.

Here, we use decoupled frequency and space mapping, that is $s^{(i)} = t^{(i)} = 0$ in (5). The rest of the mapping parameters are obtained using (6)-(7). The SMX system obtained the optimal solution in 4 iterations (5 fine model sweeps). The "fine" model response in the first iteration is shown in Fig. 3 The "fine" model response at the final iteration is shown in Fig. 4. Table I shows the initial and final parameters obtained by SMX optimization.
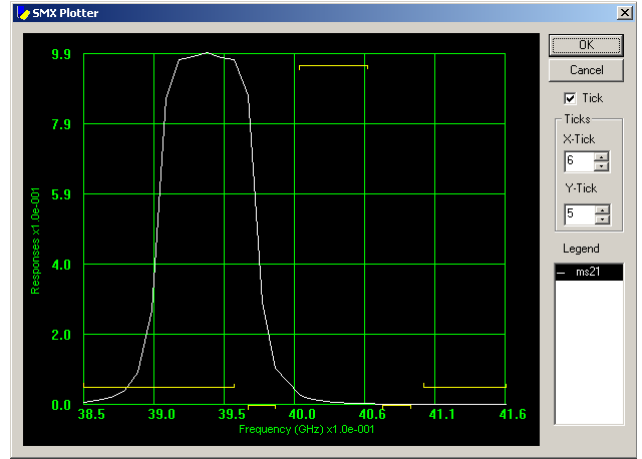

Fig. 3.   Case 1: The initial response for the "fine" model (OSA90).
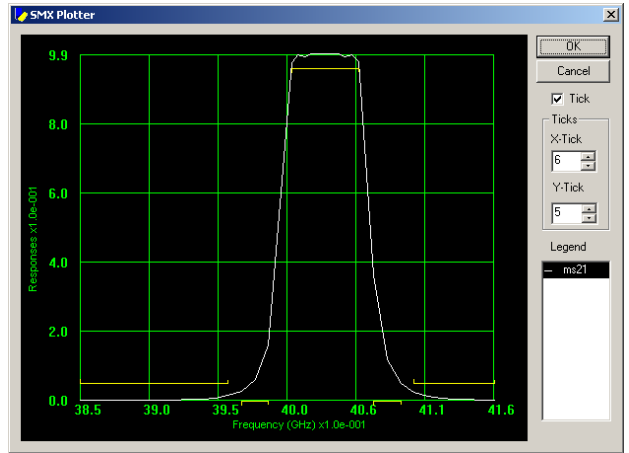

Fig. 4.   Case 1: The optimal "fine" model (OSA90) response.

In Case 2 we use Momentum as the fine model; the coarse model is as in Case 1. SMX obtains the solution in 4 iterations (5 fine model sweeps). Fig. 5 shows the fine model responses at the 4th SMX iteration. The minimax optimizer in Momentum is used to refine this solution. It takes approximately 32 hours (SMX plus refinement by Momentum optimization) on an IBM Aptiva, AMD-K7 650MHz CPU, 384MB RAM. We use a fine parameter interpolation resolution (0.1mil). See Fig. 6.

TABLE II
THE INITIAL AND FINAL DESIGNS OF THE FINE MODEL
(AGILENT MOMENTUM) FOR THE HTS FILTER

Para- meter	$x_f^{(1)}$	$x_f^{(4)}$	x_f^{Mom}
L_1	187.50	193.28	194.87
L_2	198.84	184.95	184.95
L_3	187.91	193.03	194.39
S_1	20.04	18.84	18.97
S_2	98.08	82.72	79.60
S_3	100.90	93.36	94.09
all values are in mils			

VI. CONCLUSIONS

The SMX system design is formally presented for the first time. State-of-the-art optimization technology is utilized in the design process. Object-oriented programming is used to construct the system. This makes the system easy to understand and highly extendable. New optimization methods and new simulators can be plugged in. The SMX methodology makes it suitable for engineering optimization and algorithm research.

ACKNOWLEDGEMENT

REFERENCES

[1] J.W. Bandler, R.M. Biernacki, S.H. Chen, P.A. Grobelny and R.H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Trans. Microwave Theory Tech.*, vol. 42, 1994, pp. 2536-2544.
[2] M.H. Bakr, J.W. Bandler, K. Madsen, J.E. Rayas-Sánchez and J. Søndergaard, "Space mapping optimization of microwave circuits exploiting surrogate models," *IEEE Trans. Microwave Theory Tech.*, vol. 48, 2000, pp. 2297-2306.
[3] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D. B. Serafini, V. Torczon and M.W. Trosset, "A rigorous framework for optimization of expensive functions by surrogates," *Structural Optimization*, vol. 17, 1999, pp. 1-13.
[4] UML Training in Object Oriented Analysis and Design page at http://www.cragsystems.co.uk/uml_training_080.htm.
[5] C. Petzold, *Programming Windows*, Microsoft Press 1990.
[6] *OSA90/hope*^{TM} Version 4.0, formerly Optimization Systems Associates Inc., P.O. Box 8083, Dundas, Ontario, Canada L9H 5E7, now Agilent Technologies, 1400 Fountaingrove Parkway, Santa Rosa, CA 95403-1799.
[7] *Advanced Design System* Release 1.3, Agilent EEsof EDA, November 1999.
[8] J.W. Bandler, R.M. Biernacki, S.H. Chen, W.J. Gestinger, P.A. Grobelny, C. Moskowitz and S.H. Talisa, "Electromagnetic design of high-temperature super-conducting filters," *Int. J. Microwave and Millimeter-Wave Computer-Aided Engineering*, vol. 5, 1995, pp. 331-343.
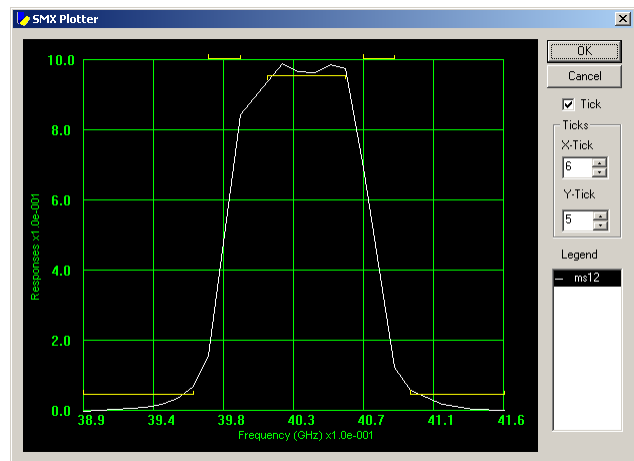

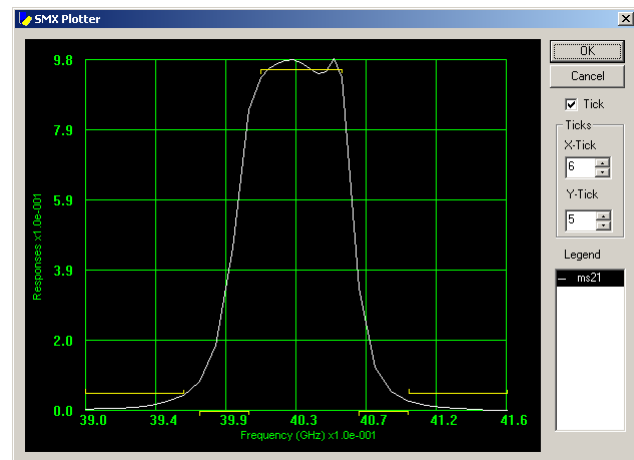Fig. 5. Case 2: The SMX optimized fine model (Agilent Momentum) response.

Fig. 6. Case 2: The final Momentum optimized fine model response with a fine interpolation step of 0.1mil.