AN OBJECT ORIENTED IMPLEMENTATION OF SPACE MAPPING, EXPLOITING SURROGATE MODELS

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

Abstract

The design of SMX, a state-of-the-art engineering optimization CAD system is described. Object-oriented technology is employed to design the system. The architecture and the modules of the SMX system are described in the Unified Modeling Language (UML). A two-section impedance transformer design is illustrated as an example of the SMX system.

I. INTRODUCTION

In the area of analog circuit design different kinds of simulators and optimizers are utilized [1]. During the optimization procedure, specifications, frequency bands and all kinds of optimization parameters need to be input to the simulator. After each simulation, the responses need to be retrieved into the optimization loop. This procedure is very tedious and error-prone. The SMX system is a state-of-the-art automated optimization computer aided design system. It integrates the latest optimization algorithm [2], automated data transfer, commercial and/or user supplied simulator drivers and built-in optimizers. It supplies an easy and practical frequency-domain circuit design procedure. SMX also provides a platform for new optimization algorithms and CAD methodologies.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants OGP0007239 and STP0201832. J.E. Rayas-Sánchez is funded by CONACYT (Consejo Nacional de Ciencia y Tecnología, Mexico), as well as by ITESO (Instituto Tecnológico y de Estudios Superiores de Occidente, Mexico). M.H. Bakr is supported by an Ontario Graduate Scholarship.

M.H. Bakr, J.W. Bandler, Q.S. Cheng, M.A. Ismail and J.E. Rayas-Sánchez are with the Simulation Optimization Systems Research Laboratory and the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada L8S 4K1.

J.W. Bandler is also with Bandler Corporation, P.O. Box 8083, Dundas, Ontario, Canada L9H 5E7.

The SMX system is described in UML [3]. Using this language, a complicated system can be easily decomposed into relatively independent small objects without losing readability and intuitiveness. These objects interact with each other. The structure of each object can also be represented in UML.

The SMX system takes full advantage of the multi-thread capability of Microsoft Windows operating system [4]. The user-friendly interface can respond smoothly while the SMX core is running in a different thread in the background. Synchronization and communication between threads are properly arranged. The SMX system is capable of optimizing while showing the intermediate results and interacting with user.

II. SYSTEM ARCHITECTURE

The SMX system consists of different modules. Here the module is an instance of certain object-oriented class. Each module can carry out certain functionality. It includes certain data structures describing the properties of the object. These modules interact with each other. The data is sent back and forth between these modules as shown in Fig. 1.

The SMX user interface and algorithm core SMX_Engine run in two separate threads concurrently. The user defined optimization parameters and control signals are sent into SMX_Engine. SMX_Engine then performs optimization and sends back the progress, the current status, responses etc., to the user interface. SMX_Engine can optimize a model using either classic optimization methods such as minimax or the state-of-the-art Space Mapping technology [2,5].

III. RESOURCES SYNCHRONIZATION

Since the interface and algorithm run in two threads, resource accessing between the threads becomes a problem. Having two or more threads simultaneously access the same data can lead to undesirable and unpredictable results. For example, one thread could be updating the contents of a structure while another thread is reading the contents of the same structure. It is unknown what data the reading thread will receive: the old data, the new data, or possibly a mixture of both.

The MFC *CEvent* class [6] is used to signal the other threads that the tasks are done or the resources are available. For example, when an optimization begins, an event is set to notify the interface that the optimization is running. Once the optimization finishes, the event is reset so that the interface knows the results are available.

IV. ALGORITHM CORE: SMX ENGINE

The core of the SMX system, SMX_Engine is an object-oriented implementation of space mapping involving surrogate models. The SMX_Engine class has the interface to setup optimization parameters and constraints for both the coarse and fine models. After setup, the coarse model is optimized by the member function OptimizeCoarseModel. This function uses m_pCoarseModelOptimizer, a pointer to a minimax optimizer to perform the optimization. Then the member function OptimizeSurrogate is called to optimize the surrogate model using space-mapping technology. To carry out space mapping, three base classes, Optimizer, Simulator and Model, need to be built.

The Optimizer base class is an abstract class. It provides the interface for standard optimization routines. With the override of optimization routines, additional parameter setup and objective function, the Huber, Minimax or other optimization classes can be derived from Optimizer. Some of the important functions in the 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 norm for different optimizer. FDF gets the error values and their derivatives using perturbation. It calls the GetErrors for evaluating the errors. SetConstrantMatrix sets the constraints for the optimizer in a matrix form. The inheritance relation of the Optimizer is shown in Fig. 2.

The Simulator class is a parent class for different simulators. Many commercial simulators are currently available. Interface functions are overridden for each new derived simulator class. Additional parameters may also be added. The Simulator class heritage is shown in Fig. 3. More simulators can be

added in the future. Regardless of the simulator type, the easiest and most common way to interact with it is to exchange data through disk files. The procedure to interface to any simulator is divided into three steps: writing input file for simulator (WriteInputFile), simulating it (Simulate) and obtaining the results (GetResponses). To automate the whole processing, commercial or user supplied simulators need to be driven from the SMX system. The advantage of Microsoft Windows operation system makes this feasible. Any window in the Microsoft Windows operating system is an object. As long as we can obtain a pointer to the window object, we could manipulate the window object by sending messages to it. The window object accepts the messages and responds just like interacting with a human being through mouse clicks and keyboard inputs.

The new optimization method involves SurrogateModel which is derived from a base Model class. The Model class extends the functionality of simulator. It functions as a wrapper of a simulator. The responses could be obtained independent of the simulator. Obviously, Simulator should be one of its members. The Model class can send data to the simulator and retrieve the responses from it. Also it can obtain the Jacobian of the model responses with respect to the physical parameter which is essential for the space mapping optimization. SurrogateModel is derived from the Model class. It is a linear convex combination of two different models. This model is used by the novel surrogate model space mapping algorithm in SMX_Engine. The Model class is shown in Fig. 4.

V. SUPPORT CLASSES

Many support classes are created and included in the SMX system. One of the most important classes is the matrix class. The optimization procedure involves a lot of matrix computations. The matrix class has the capability of Broyden update and Singular Value Decomposition etc. Data package classes are constructed to transfer data between modules. The data packages capsulate the related data. This cleans up the usually tedious and vague arguments of the algorithm function. These packages can be easily inserted into arrays to transfer a large amount of data.

The graphic representation of the optimization results is a necessity for engineering CAD. A plotter class is designed to show the curves. Curves can be added into the plotter object one by one. The plotter will fit them into a window by automatically adjusting the scale between ticks. Constraints can also be plotted.

VI. EXAMPLE

A two-section 10:1 capacitively-loaded impedance transformer [7] is implemented in *OSA90/hope*™ [8] as an example of SMX system. Two models are provided. One is considered as a fine model, the other is taken as coarse model.

The "fine model" is a two-section transmission line with shunt capacitors. The "coarse model" is the same transmission line without the shunt capacitors. The design specifications are

$$|S_{11}| \le 0.50 \quad \text{for} \quad 0.5\ \text{GHz} \le \omega \le 1.5\ \text{GHz}$$

The designable parameters are the electrical lengths of the two transmission lines at ω =1.0 GHz. The characteristic impedances are kept fixed at their optimal values. Both the coarse and fine models make use of the ideal transmission line model of *OSA90/hope*. Eleven frequency points are simulated per sweep.

The setup procedure is a step-by-step wizard style. It guides user to input the parameters and constrains. The coarse model response S_{11} is calculated at the start point. The optimal S_{11} responses of coarse model are obtained. Using the optimal coarse model solution as the initial point for the fine model, the corresponding fine model responses is shown in Fig. 5. The surrogate model is then optimized. Finally, the final fine model responses are shown in Fig. 6.

VII. CONCLUSIONS

The SMX system design is described. State-of-the-art optimization technology is utilized in the design process. Object-oriented methodology 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 easily. The SMX design methodology makes it suitable for engineering optimization and algorithm research.

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 MTT-S Int. Microwave Symp. Dig.* (Boston, MA), 2000, pp. 1785-1788.

[3] UML Training in Object Oriented Analysis and Design page at http://www.cragsystems.co.uk/uml_training_080.htm.

[4] C. Petzold, *Programming Windows*, Microsoft Press 1990.

[5] J.W. Bandler, R.M. Biernacki, S.H. Chen, R.H. Hemmers and K. Madsen, "Electromagnetic optimization exploiting aggressive space mapping," *IEEE Trans. Microwave Theory Tech.*, vol. 43, 1995, pp. 2874-2882.

[6] *MSDN Library Visual Studio 6.0*, Microsoft Corporation.

[7] J.W. Bandler, "Optimization methods for computer-aided design," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-17, 1969, pp. 533-552.

[8] *OSA90/hope*™ 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.

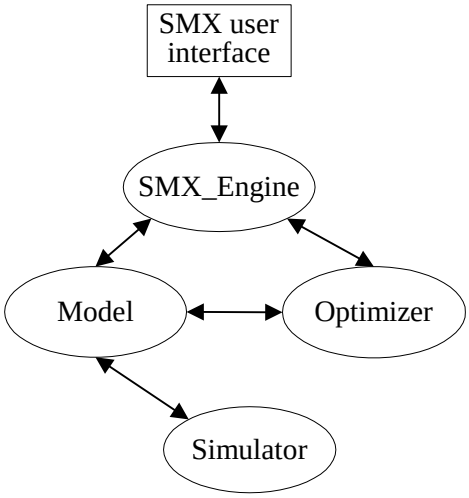

Fig. 1. Illustration of the data flow in the SMX system. The arrows represent the flow of data from module to module.

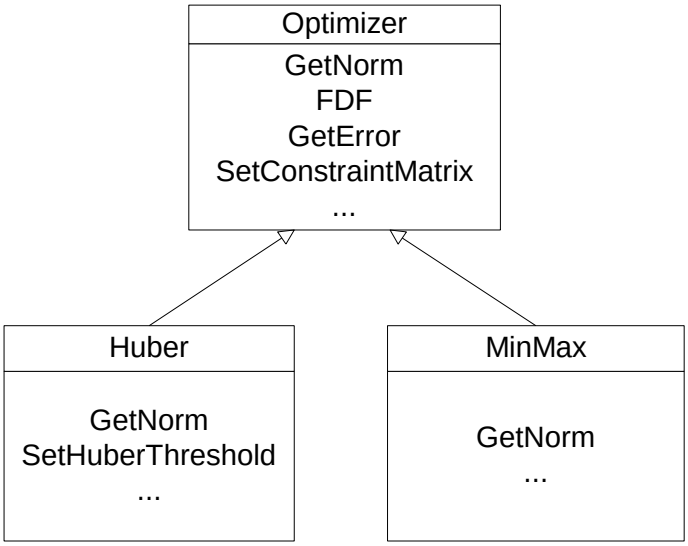

Fig. 2. Illustration of the derivation of basic optimizer class. The basic optimizer class includes basic functionality for gerneralized optimization. These functionality includes obtaining norm and calculating error in the iteration. Huber and minimax are two derived classes. They have their own norm calculation function and other parameter setups.

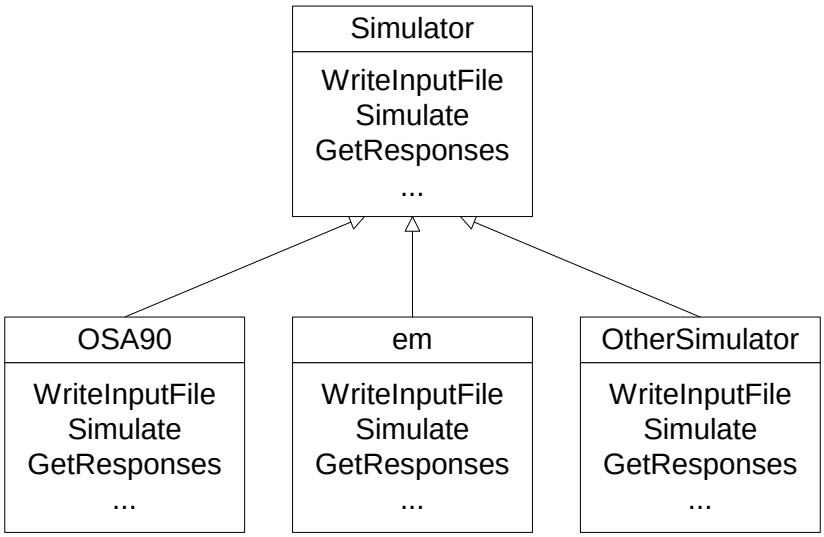

Fig. 3. Illustration of the derivation of simulator classes. The base simulator class is designed for disk file data exchange. When derived to other simulaters, the procedure is the same. But the data exchange method may be different.

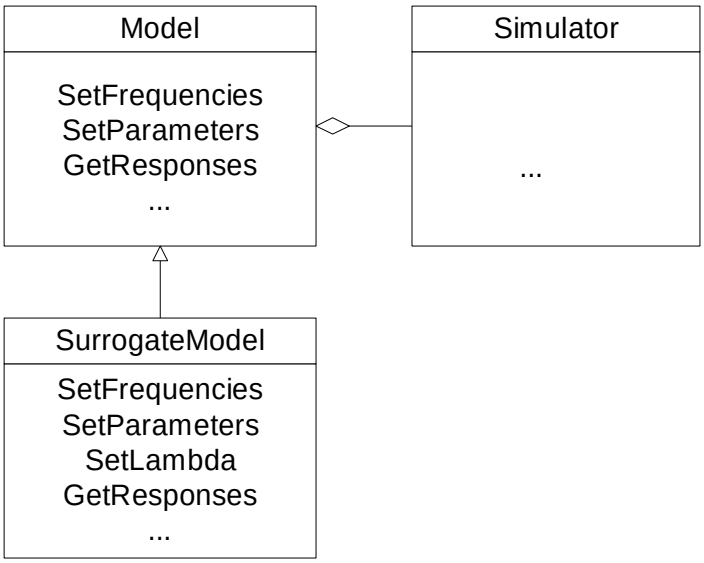

Fig. 4. Illustration of the model class. The model class functions as a wrapper of the Simulator class. It is independent of the simulator and expands its functionality. SurrogateModel is a special model which is special for the surrogate model space mapping technology.

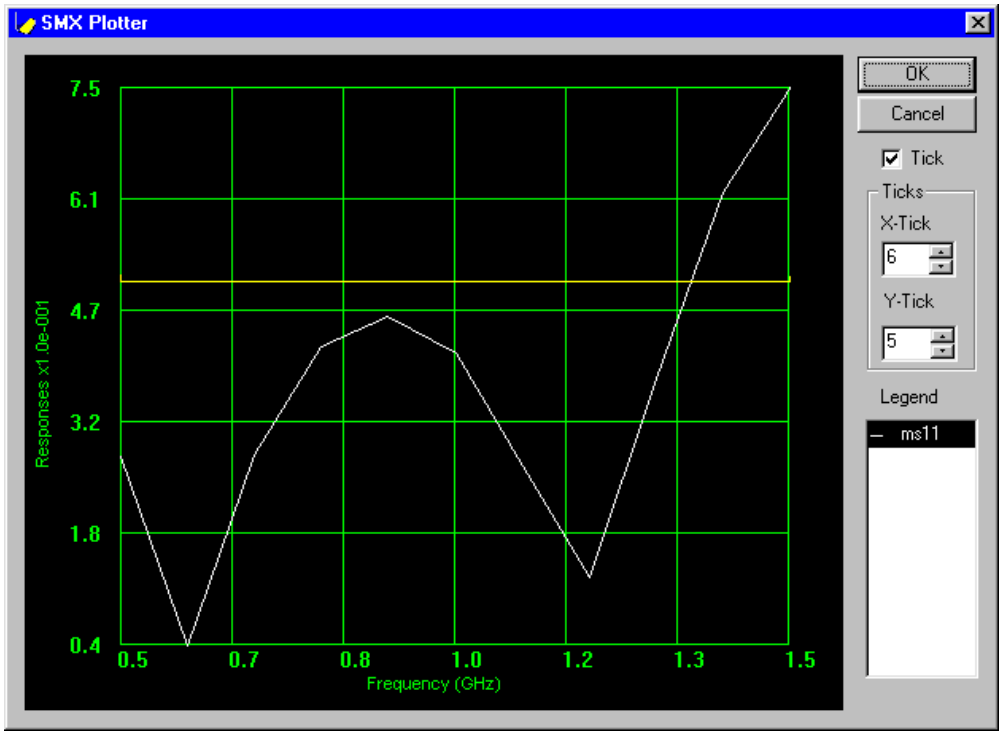

Fig. 5. The initial fine model response of the two-section impedance transformer. The responses are obtained at the optimal point of the coarse model.

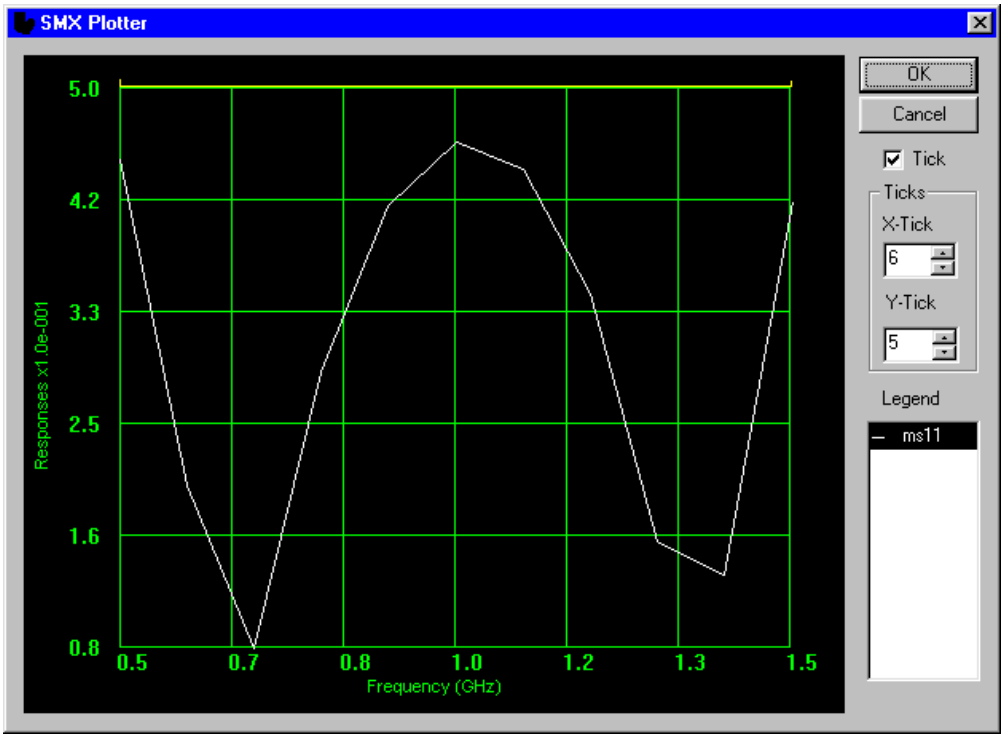

Fig. 6. The optimized fine model response of the two-section impedance transformer coarse model after optimizing the surrogate model using SMX.