Elektrotehniški vestnik 68(1): xxx-xxx, 2001
Electrotechnical Review, Ljubljana, Slovenija

Integration of Generic Optimisation Algorithms in SPICE

Janez Puhan, Iztok Fajfar, Tadej Tuma and Arpad Bűrmen

University of Ljubljana, Faculty of Electrical Engineering,
Tržaška cesta 25, 1000 Ljubljana, Slovenia
e-mail: janez.puhan@fe.uni-lj.si

Abstract. Though rather awkward and limited in its usage, SPICE has been used by several authors solving their various optimisation problems. In this paper we propose how some generic optimisation algorithms can be integrated into SPICE. The implemented program has proven to be an extremely efficient optimisation tool for an experienced circuit designer. It is free of charge available at our web site (http://fides.fe.uni-lj.si/spice/).

Keywords: SPICE, Nutmeg, optimisation, circuit simulation

Vgradnja generičnih optimizacijskih algoritmov v SPICE

Povzetek. Čeprav so nekateri avtorji že uporabljali SPICE za reševanje različnih optimizacijskih problemov, je njegova uporaba v optimizacijskih postopkih še vedno nerodna in omejena. V članku je predstavljen način integracije nekaterih splošno znanih optimizacijskih algoritmov v SPICE. Program se je izkazal za izredno učinkovito optimizacijsko orodje v rokah izkušenega načrtovalca električnih vezij. Na voljo je brezplačno na naši spletni strani (http://fides.fe.uni-lj.si/spice/).

Ključne besede: SPICE, Nutmeg, optimizacija, simulacija vezij

1 Introduction

Among the numerous digital programs aiding in circuit design, SPICE, that was originally developed at the University of California, Berkeley, is undoubtedly the one that is most widely used by circuit designers. Optimisation is one of the important aspects of circuit design that neither SPICE nor any other circuit design software by itself can support adequately. It is by no means an easy and straightforward task. It requires a great deal of understanding of the circuit to be optimised for which no cookbook recipe is available. There exist many generic optimisation algorithms performing differently, depending upon the problem in question, the way optimisation criteria are constructed in a form of objective function, or the choice of initial parameter values, to name just few of them. But even an experienced circuit designer, well understanding the problem, cannot attack the optimisation problem efficiently by using the flavours of SPICE alone.

Received xxx
Accepted xxx

Although many cases of using SPICE in the optimisation process have been reported, SPICE has mostly been used as a verification tool checking whether the optimised circuit performs as anticipated [e.g. Wei L (1999)]. Efforts have been taken of linking SPICE to some other software to automate the optimisation process (e.g. SPICE and Matlab [Siu K W (1995)], or SPICE and Mathematica [Puhan J (1997)]). These approaches, however, proved to be both slow (problem of communication between two separate applications) and difficult to use (the designer has to cope with different applications and languages).

Our aim was to integrate into SPICE some sort of generic optimisation so that both would yield adequate processing speed and be easy for use by circuit designers. The result of the project was a tool implemented as a new Nutmeg command (Nutmeg is an interactive interpreter language used for controlling certain aspects of the circuit simulation process in SPICE). It incorporates 11 different well-known optimisation algorithms. Apart from selecting an algorithm, the tool allows the user to specify almost arbitrary the cost function for which minimal coding is required.

One of our most serious problems were the numerous memory leaks and some other bugs found in the original Berkeley code. Since all the today available variations of SPICE depend on that code, they also suffer from the same memory leaks and bugs. Although the bugs are mostly harmless with regard to ordinary circuit simulations, they are a serious problem during an optimisation process where several hundreds of circuit analyses must be carried out in a loop.

Consider, for example, the following simple loop written in Nutmeg:

```
while 1
  rusage
end
```

The `rusage` command in infinite, whereas the loop reports the system memory usage which increases constantly although there is no explicit command in the code that consumes memory. We found no version of SPICE to run that loop without eventually occupying all the system memory and crashing the program. Therefore, our first step was the painstaking task of finding and fixing those bugs.

2 Practical Implementation

After fixing the bugs and before coding the optimisation algorithms, we had to decide how the algorithms would be integrated into SPICE. Several approaches have been proposed and tested [Puhan J (1997)]. We finally chose the one which offered both the fastest speed and a simple to use interface.

The optimisation was integrated into the existing SPICE front-end as depicted schematically in Fig 1. From the user point of view, the optimisation is performed through a single Nutmeg command called `optimize`. Originally, parser parses all the Nutmeg commands and calls appropriate functions that are either executed inside the front-end or act as an interface to the simulator. The optimize command is not parsed by the parser but rather passed on to the optimisation part of the program. This part implements the general optimisation loop, which, when necessary, calls internal front-end functions in order to perform circuit analyses and other control commands. The structure of that general optimisation loop is always the same. The algorithm is as follows:

```
do
 change parameter values
 satisfy explicit constraints
 if implicit constraints satisfied
  then
  perform required analyses
  calculate cost function
 endif
while termination criteria not
 satisfied
```

The selection of a certain optimisation algorithm affects only the way in which the parameter values are changed in the first line within the above do...while statement.

The great advantage of this approach is the fact that it is not necessary to run the simulator separately for each analysis thus speeding up the optimisation significantly. One could, of course, code the optimisation loop directly in Nutmeg but such approach would be quite awkward since Nutmeg is not a very flexible programming language.

The optimisation algorithms themselves are hard coded in the C language separately from the SPICE code although their coding demanded detailed knowledge of internal functioning of SPICE (e.g. data structures and internal structure of the Nutmeg interpreting language).

3 The Algorithms

After the implementation technique had been decided upon, our next step was to implement some most widely known generic optimisation algorithms. We implemented 11 of them:

1. Steepest descent*
2. Newton's method*
3. Davidon-Fletcher-Powell's method*
4. Monte Carlo method*
5. Grid search method
6. Search along co-ordinate axes*
7. Powell's method*
8. Hooke-Jeeve's method
9. Constrained simplex method
10. Simple genetic algorithm
11. Brute force method (evaluation of cost function across the whole parameter space)

For finding a minimum along a selected direction, algorithms marked with asterisk use quadratic approach, golden section search, or Fibonacci search.

It is very convenient for the designer to have many different optimisation algorithms available. Especially in hard optimisation problems, different methods can yield quite different minima. Given the additional possibility to easily change certain parameters of algorithms, the designer has a powerful tool that can significantly shorten the design time and possibly yield more optimal solutions.

4 An Example

In order to demonstrate the use of our optimize command, we look at a simple Schmitt trigger depicted in Fig 2. Our objective is to find values of resistors R_2 and R_3 such that switching will take place at voltages of 3V (up) and 2V (down). The high and low level output voltages should differ for at least 10V.

Before starting to optimise the circuit we need to set up necessary constraints and the cost function. Two explicit constraints on the two resistors are derived empirically. The chosen ones are those at which the switching voltage is anywhere within the power supply voltage range:

$$\begin{aligned} 10k\Omega \le R_2 \le 30k\Omega \\ 1k\Omega \le R_3 \le 50k\Omega \end{aligned} \qquad (1)$$

The implicit constraint ensures the required difference in output voltages:

$$v_{5\max}(R_2,R_3) - v_{5\min}(R_2,R_3) > 10V \qquad (2)$$

Finally, the cost function deals with switching voltages:

$$E = \int_0^T \left| \left(v_5 > \frac{v_{5\max} + v_{5\min}}{2} \right) - v_{5ideal} \right| dt \qquad (3)$$

Here, the Boolean operator greater than (>) returns 1 or 0, depending on whether the left hand side of the operator is or is not greater than its right hand side, while v_{5ideal} is a normalised ideal output response.

Now, using the optimize command, the above problem can be formulated in a few lines of the Nutmeg code:

```
* parameters, explicit constraints, and initial
* point:
optimize parameter 0 element r2 parameter
+ resistance low 10k high 30k initial 12k
```

```
optimize parameter 1 element r3 parameter
+ resistance low 1k high 50k initial 8k
* implicit constraint:
optimize implicit 0 v(5)[5500] - v(5)[0] gt 10
* analyses performed in each iteration:
optimize analysis 0 tran 1ms 11s
optimize analysis 1 linearize
* specify cost function (v(5)[5500] = v(5)max):
let ideal = (vector(11001) gt 2999) and
+ (vector(11001) lt 9001)
optimize cost mean(abs((v(5) gt
+ (v(5)[5500] + v(5)[0]) / 2) - ideal))
* use constrained simplex method with default
* parameters:
optimize method complex
* run optimisation algorithm:
optimize
```

Running this code we arrive at the global minimum of the cost function at $R_2 = 18k\Omega$ and $R_3 = 20k\Omega$.

5 Conclusions

We have integrated eleven well-known optimisation methods into SPICE. Their usage is, of course, not trivial and requires a certain level of proficiency and expertise in optimisation. Nevertheless, the implemented optimisation tool proved to be easy to use by experts. They can, with minimal coding, quickly arrive to desired results, even when different optimisation methods need to be combined, e.g. a genetic algorithm applied so as to find suitable initial points for some gradient algorithm.
More examples, executable code running under Windows 95/98/NT and Linux, and further information about the developed software are available free of charge at our web site (http://fides.fe.uni-lj.si/spice/).

6 References

[1] Puhan J and Tuma T (1997) Optimisation of analog circuits with SPICE 3f4. Proc. European Conference on Circuit Theory and Design. Budapest. 1: 177-180

[2] Siu K W and Lee Y S (1995) MATSPICE - putting circuit simulation and design optimisation together. Proc. European Conference on Circuit Theory and Design. Istanbul. 2: 1161-1164

[3] Wei L, Chen Z, Roy K, Johnson M C, Ye Y and De V K (1999) Design and optimisation of dual-threshold circuits for low-voltage low-power applications. IEEE Trans. Very Large Scale Integration (VLSI) Systems. 7: 16-24

Janez Puhan graduated from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia in 1993, where he also obtained his M.Sc. degree in 1998. His current research interests are in computer aided design of analog circuits, optimisation methods and computer circuit analysis.

Iztok Fajfar received the Dipl. Ing., M.Sc. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, in 1991, 1994 and 1997, respectively. Since 1992 he has been with the same faculty he is currently an assistant professor. His research interests include design and optimisation of electronic circuits with focus on cellular neural networks.

Tadej Tuma received his diploma degree from the Faculty of Electrical Engineering, University of Ljubljana in 1988. Soon thereafter he joined the same faculty as a teaching assistant. At the same time he began his postgraduate studies, which were completed with his M.Sc. thesis in 1991 and his Ph.D. dissertation in 1995. His research interest is mainly in the field of computer aided circuit design, especially in analog circuit optimisation methods.

Arpad Bűrmen received the B.Sc. degree in 1999. He is currently a junior researcher at the Faculty of Electrical Engineering of the University of Ljubljana. He teaches laboratory practice for undergraduate courses in computer basics and programming. His research interests include analog and mixed-mode simulation, and modelling and optimisation of circuits and systems.

List of Figures