Q-40

The Effectiveness of Four Direct Search Optimization Algorithms

Randall W. Rhea

President
Circuit Busters, Inc.
1750 Mountain Glen
Stone Mountain, GA 30087

Principal Engineer
Scientific-Atlanta, Inc.
3845 Pleasantdale Road
Atlanta, GA 30340

ABSTRACT

Effective computer circuit simulation programs require efficient optimization routines. This paper describes four different direct search routines [1] beginning with a simple random direct search and concluding with the presented algorithm. The four routines are tested on three different circuits using the same computer circuit simulation program, so that conclusions may be drawn as to the relative effectiveness of each routine.

INTRODUCTION

An efficient optimization routine (1) causes the greatest reduction of the objective function with the fewest evaluations of the circuit function, (2) is immune to behavior of the circuit which tends to reduce effectiveness, (3) is tolerant of weights of specified parameters in the objective function, and (4) requires a minimum of computer code and memory.

OBJECTIVE FUNCTION

The objective function used in the simulation program is least square.

$$\text{Error} = \sum_{l=1}^{k} \left[\sum_{f=m}^{n} \sum_{i=1}^{5} W_{i,l} (S_{i,f} - T_{i,l})^2 / (n - m) \right] \quad (1)$$

where l = optimization frequency band 1 to k.
f = frequency point m to n .
i = parameter being optimized.
$W_{i,l}$ = weight for parameter i in band l.
$T_{i,l}$ = target for parameter i in band l.
$S_{i,f}$ = value of parameter i at frequency f.

CIRCUIT EVALUATION ROUTINE

Evaluation of the objective function requires the computation of $S_{i,f}$. The parameters used were scattering parameters (S-parameters). The computer program used to determine the S-parameters was SuperStar, version 2.1, by Circuit Busters, Inc. Each optimization routine evaluated was inserted into this program. Using the same computational routine to evaluate each optimization algorithm is essential in determining the relative effectiveness of each routine.

SuperStar is a general purpose circuit simulation and optimization program. It was written in IBM BASICA 3.0 and compiled with Microway 87Basic and 87Basic/Inline. The code produced by this compiler executes faster than code from efficient Fortran compilers [2]. The three circuit examples in this paper use the cascaded ABCD parameter facilities of SuperStar. No matrix manipulations are required in this mode. The times presented in this paper include the time required to graphically present the results in S-parameter form at intermediate optimization iterations. This overhead penalizes slightly the times of the faster, more efficient algorithms relative to the slower routines.

The objective function evaluation routine in SuperStar allows multiple frequency bands and use of inequalities. The desired targets, and their weights, are included in the ASCII file which also describes the circuit to be optimized.

The program was run on an IBM PC with a 4.77 MHz clock and an 8087 floating point coprocessor. The final algorithm presented in this paper is the optimization algorithm used in SuperStar.

DIRECT SEARCH ROUTINES

A study of the three preliminary and the presented algorithm was done first on a low-pass filter. The algorithms were (1) a simple random search, (2) a simple pattern search, (3) added adaptive step size to the pattern search and (4) added quadratic estimation of the element value to the third algorithm.

0149-645X/87/0000-0697$01.00 © 1987 IEEE

LOW-PASS FILTER

The first circuit chosen to evaluate the optimization algorithms was a three element-T 1dB passband ripple Chebyshev low-pass filter. This circuit was chosen because 1) the exact solution is known and 2) only two unique element values are involved, so the error contours versus element values can be plotted in two dimensions.

The filter has a 3dB cutoff frequency of 1GHz. The attenuation is 13.41dB at 1.5GHz. A load and source impedance of 50 ohms is used. The exact solution is 16.104nH for the two series inductors and 3.1643pF for the shunt capacitor.

Error contours of 1E-6, 1E-5, 1E-4 and 1E-3 versus element values are given in Fig.1. These contours result from declaring that the insertion loss be less than 1dB from 0 to 1000MHz and greater than 13.41dB at 1500MHz and higher. A frequency point every 100MHz was used.

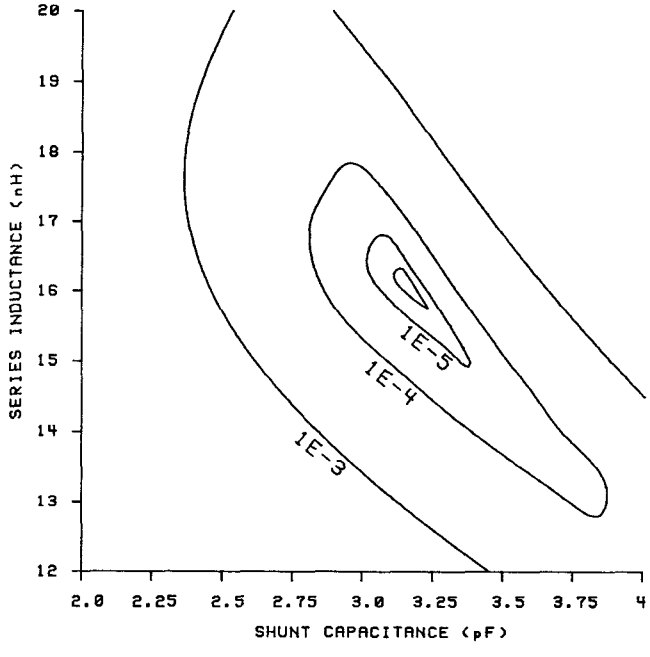


FIG.1. Error contours for a low-pass filter.

RANDOM SEARCH

The first algorithm was a random direct search [3]. Each element, in order, is stepped down and up a specified percentage. The circuit and error are evaluated for each case. The element value, of these three, with the lowest error is the new element value. The process is repeated with the next element. When all elements have been adjusted, the process repeats with the first element.

Two initial starting locations for the inductor and capacitor were evaluated. Both were near the 1E-3 error contour, however, one set was near the valley of the contours while the second was orthogonal to the valley. The evaluation was done for several step size percentages.

Results of this algorithm evaluated on the low-pass filter are given in Fig.2. The time to reach an error of 1E-6 or less is given. Also given (in parenthesis) is the minimum error achieved by the routine. The term "hang" signifies that the routine never achieved a minimum error of 1E-6 or less.

Interesting conclusions may be drawn from the results. First, a small step size was required to avoid hang. A step size larger than 1% hung. This is because certain element values within 1% of the exact values result in an error greater than 1E-6.

Second, the starting point was extremely important. Starting near the valley was unfortunate. A step size of .125% or less was required to avoid hang.

ALGORITHM	INITIAL STEP SIZE	TIME TO ERROR=1E-6 (MINIMUM ERROR) 3 ELEMENT LOW-PASS FILTER	
		14nH, 2.8pF	13nH, 3.8pF
RANDOM	16%	HANG (4.8E-5)	HANG (8.9E-4)
RANDOM	4%	HANG (4.2E-6)	HANG (8.9E-4)
RANDOM	1%	86s	HANG (2.4E-5)
RANDOM	.25%	336s	HANG (2.3E-6)
PATTERN	16%	HANG (4.8E-5)	HANG
PATTERN	4%	HANG (4.2E-6)	HANG
PATTERN	1%	86s	HANG (2.4E-5)
PATTERN	.25%	333s	HANG (2.3E-6)
ADAPTIVE	16%	15s	419s
ADAPTIVE	4%	30s	417s
ADAPTIVE	1%	87s	384s
FULL	16%	15s	479s
FULL	4%	21s	481s
FULL	1%	51s	445s

FIG.2. Results of the algorithms on the filter.

PATTERN SEARCH

In this algorithm, each element is stepped down and up and the errors are recorded. Only the element with the best error reduction is permanently changed. This process repeats by again searching for the element with the best error reduction. This algorithm therefore attempts to align the search in an optimum direction [4].

Results of the pattern search algorithm on the low-pass filter are given in Fig.2. The results were disappointing. The tendency to hang and the need for small step size remained. Furthermore, time to 1E-6 was actually slightly increased. However, results with a second example later will show the pattern search was better than the random search.

An obvious tendency is apparent from the results with the random and pattern search algorithms. Larger step sizes converge more quickly, but increase the likelihood of hang. It is probably safe to assume that different circuits will have different threshold step sizes to avoid hang, which makes determination of an appropriate step size extremely difficult. The only safe approach to avoiding hang is to make the step size small, which results in long convergence time.

ADAPTIVE STEP SIZE

To overcome this difficulty, adaptive step size was added to the pattern search algorithm. In this algorithm, optimization begins with a large initial step size, for example 16%. The pattern search algorithm is begun, with a step down being the element value divided by 1.16 and a step up being the value multiplied by 1.16.

When all elements have been optimized to the point that stepping any element down or up by 1.16 doesn't reduce the error, the step size is reduced. This particular algorithm reduces the step size to the fourth root of 1.16 or 1.0378. The pattern search is repeated with the lower step size. The step size is reduced again when no step of an element results in a reduced error. This process continues until the step size reaches a specified lower limit, and program execution is terminated. The lower limit was set at .02% for these tests and in the program SuperStar.

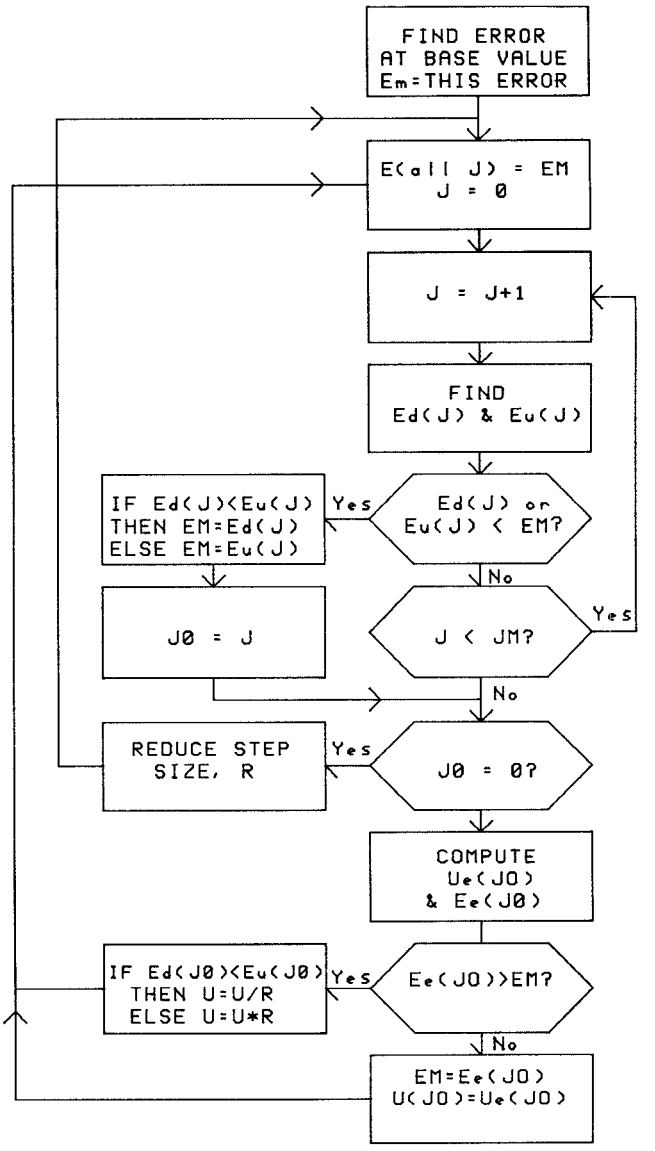
Results of the adaptive step size algorithm on the low-pass filter are given in Fig.2. Results were gratifying. Optimization converged to less than 1E-6, even with an initial step size of 16%.

FULL ALGORITHM

The final algorithm adds quadratic estimation of an element value to the adaptive step size algorithm [5]. After a pattern search identifies the element which produces the greatest error reduction, the value of that element which would produce the lowest error is estimated by fitting a quadratic to the stepped down, initial and stepped up values of the element.

A flow chart of the full algorithm is given in Fig.3. Results of this algorithm on the low-pass filter are given in Fig.2.

Convergence time for the fortunate starting element values were reduced. Times were somewhat greater for the unfortunate starting values. However, it will be shown that quadratic estimation gave significant improvement for the other two circuit examples.



E(J) = error at variable base value.
Ed(J) = error with variable J multiplied by R.
Eu(J) = error with variable J divided by R.
Ee(JO)= error at estimated variable value.
JM = number of variables being optimized.
JO = variable with greatest error reduction.
R = ratio by which a variable is multiplied or divided.
U = variable base value.
Ue(JO)= estimated value of variable.

FIG.3. Flow diagram for the full algorithm.

STARTING VALUES

The starting values of the elements in this example have a profound effect on the optimization time. The reason for this is readily understood by examination of the error contours in Fig.1. The valley of the error contours for this example doesn't align with either variable. In fact, it is at nearly a 45 degree angle to both axis. When starting very near the valley, even a small step in any direction for either variable results in an error greater than the starting values. Therefore, programs with fixed step size algorithms may hang at the first iteration, or soon thereafter if values happen very near the valley. With adaptive step size, the algorithm immediately steps down to a size sufficiently small to avoid hang, and then must transverse the distance to minimum error with very small step size.

The latter type of algorithm is prefered. It may take a long time, but at least it is much more likely to reach the optimum values.

Gupta, et al [5], gives an algorithm based on work by Rosenbrock [6] for rotating coordinates such that the search is aligned along the valley. This probably could be a basis for improving the algorithm presented here.

COMPUTING THE QUADRATIC ESTIMATE

Reference [5] includes an algorithm for quadratic estimation of the minimum error value of a circuit variable. This routine is based on arithmatic delta steps from the base variable value. If geometric steps are used, (the base value is multiplied and divided by a ratio, "R") then the algorithm is simpler. The value of a variable, Xe, which is estimated to result in the lowest error is calculated by assuming the error curve is quadratic in the area of the base value.

$$A = \frac{R^2*Ed - (R+R^2)*E + R*Eu}{(R^3-R^2-R+1)*X^2} \qquad (2)$$

$$B = \frac{E - Eu + A*(R^2X^2 - X^2)}{X*(1 - R)} \qquad (3)$$

$$Xe = -B/2A \qquad (4)$$

where E = error at base value of variable
Ed = error at base value/R
Eu = error at base value*R
X = variable base value
Xe = estimated min error value

PARAMETER WEIGHTS

The parameters of the objective function for a passive circuit, such as the low-pass filter example, range from unity (no loss) to zero (no transmission). In the general optimization problem, there may be a wide varience in the target parameters or in one target parameter for different frequency bands. So that all parameters contribute adequately to the objective function, weights for the parameters (W_{ij}), which may be different for each frequency band, are allowed.

Because the weights are selected by the user, who can only estimate appropriate values, the algorithm should be tolerant of the selected values.

The full algorithm was tested on the low-pass filter with three sets of selected weights. Results are given in Fig.4. In case I, SuperStar default weights of unity were selected. In case II, the inverse of the target S-parameters were selected; 1.122 for the passband and 4.683 for the stopband. This is a natural choice, because it tends to balance the effect on the objective function of the pass and stop bands. In case III, the pass and stop band weights of case II are reversed. This is probably an exceptionally poor choice, because the passband parameter is overly significant.

The results indicate reasonable tolerance to weight selections. In some cases slightly longer optimization times resulted, in other cases improved times resulted. Undesirable results were only noted with the poor choice of weights in combination with the unfortunate starting values.

Weights can have two different effects on the optimization problem. When an exact solution exists (most likely if conditionals are used) weights tend to effect only optimization time. When an exact solution doesn't exist, weights effect the final values. For example, if the passband is given a high weight, passband requirements tend to determine the outcome, and other targets are missed further.

CASES	INITIAL STEP SIZE	TIME TO ERROR=1E-6 or (MINIMUM ERROR) 3 ELEMENT LOW-PASS FILTER	
		14nH, 2.8pF	13nH, 3.8pF
CASE I	16%	15 s	479 s
CASE I	4%	21 s	481 s
CASE I	1%	51 s	445 s
CASE II	16%	15 s	523 s
CASE II	4%	26 s	266 s
CASE II	1%	71 s	297 s
CASE III	16%	30 s	(2.1E-6)
CASE III	4%	26 s	(1.9E-6)
CASE III	1%	46 s	(2.6E-6)

FIG.4. Filter results using different weights.

TRANSISTOR AMPLIFIER

The second circuit example is given in Fig.5. It is a broadband bipolar amplifier with distributed element matching and feedback [7]. The final values depend somewhat on the start step size and the algorithm being tested. The final values given in Fig.5 were program outputs at automatic termination using the full algorithm, with an initial step size of 16%. The starting values and optimization targets are given in the SuperStar circuit file, Fig.6. The targets were 10 dB S21 (forward gain) and best possible match from 500 to 1500 MHz.

Results of the optimization are given in Fig.7 for all four algorithms with initial step sizes of 16%, 4% and 1%.

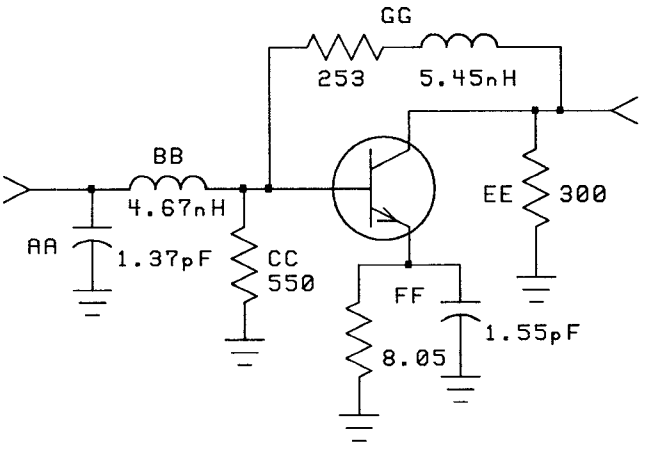


FIG.5. Transistor amplifier schematic. Optimized values are shown.

DELAY NETWORK

In Fig.8 is the schematic and SuperStar circuit file for a seven element Bessel low-pass filter with a cutoff frequency of 1 MHz.

This circuit was the starting point for a delay network with desired characteristics after optimization of (1) delay of 500nS up to 1 MHz, (2) insertion loss less than 1dB up to 1 MHz and (3) equal inductor values.

Before optimization, the delay is approximately 480nS. The inductor values of the original Bessel prototype are significantly different from each other. For the optimization starting point, all inductors were set equal to each other with a value of 5500nH.

Results of optimization are given in Fig.7 for all four algorithms with initial step sizes of 16%, 4% and 1%.

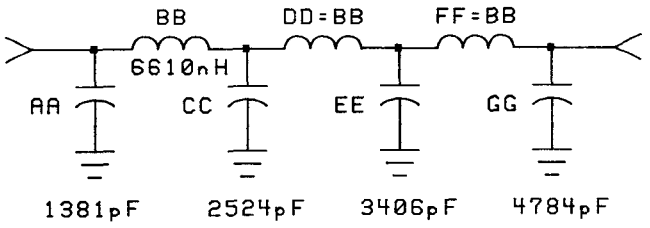
```
CIRCUIT
CAP AA PA ?2
IND BB SE ?2
RES CC PA 550
TWO DD SP 50 'NEC645.S2P
RES EE PA 300
PRC FF PA ?6 ?2
SRL GG SE ?180 ?16
SER DD FF
PAR DD GG
CAX AA EE
OUTPUT
DSP AA SD 50
FREQ
SWP 500 1500 11
OPT
500 1500 S21=10 W21=2 S11<-100 S22<-100

The NEC645.S2P data file is:

200 .50 -75 22.5 145 .02 58 .78 -20
500 .55 -118 12.6 112 .03 52 .53 -28
1000 .55 -158 8.6 97 .04 54 .44 -31
1500 .45 -178 6.0 86 .05 59 .40 -31
```

FIG.6. Amplifier circuit file. Starting values are shown.

For both the amplifier and delay network examples, the starting values were far from the optimium values. In some cases, the starting values were off by more than a factor of two. In spite of this, optimization times for the full algorithm were short, in contrast to significant hang and long time problems with the random algorithm.

ALGORITHM	INITIAL STEP SIZE	TIME TO ERROR=1E-1 AMPLIFIER	TIME TO ERROR=1E-3 DELAY NETWORK
RANDOM	16%	198s	HANG
RANDOM	4%	HANG	477s
RANDOM	1%	HANG	1749s
PATTERN	16%	74s	HANG
PATTERN	4%	265s	606s
PATTERN	1%	1032s	2271s
ADAPTIVE	16%	74s	195s
ADAPTIVE	4%	265s	606s
ADAPTIVE	1%	1032s	2271s
FULL	16%	57s	89s
FULL	4%	67s	274s
FULL	1%	89s	597s

FIG.7. Amplifier and delay network results.




```
CIRCUIT
CAP AA PA ?352.1
IND BB SE ?5500
CAP CC PA ?1671
EQU DD BB
CAP EE PA ?2766
EQU FF BB
CAP GG PA ?7213
CAX AA GG
OUTPUT
GPH AA S21 50 -10 0
GPH AA DLY 50 0 1000
FREQ
SWP 0 1 21
OPT
.05 .95 S21>-1 DLY=500 WDL=1E-5
```

FIG.8. Schematic and circuit file for the delay network. Starting values are in the file.

SUMMARY

The direct random search has severe limitations. In all three cases studied, hang was a prevelent problem, and optimization time was long. The initial step size was critical. The minimum errors achieved after any time were large. Errors could have been reduced by choosing smaller step sizes, but times would have been further increased.

The pattern search was somewhat more effective. Time was slightly increased for the delay network, but significantly reduced for the amplifier. Very little effect was observed on the low-pass filter.

The addition of adaptive step size to the pattern search algorithm was key to the elimination of the hang problem. No case of hang remained in these examples with adaptive step size. The fact that no effect was observed on the amplifier example when adaptive step size was added is probably because of the broadband nature of that example; the routine didn't reduce the step size prior to reaching the error limit of 0.1.

Most significantly, with adaptive step size, the user (or computer program) has been relieved of the responsibility of accurately guessing an appropriate initial step size.

The addition of quadratic estimation of variables to the algorithm was a significant improvement for the amplifier and delay network examples.

The full algorithm presented here, used in the program SuperStar, has been succesfully applied to a wide variety of circuit applications.

REFERENCES

[1] J. W. Bandler, "Optimization Methods for Computer-Aided Design," IEEE Trans. Microwave Theory and Techniques, vol. MTT-17, pp. 533-552, August 1969.

[2] S. S. Fried, "The 8087/80287 Performance Curve, Fall 1985 Byte, Inside the IBM PCs, pp. 67-88.

[3] D. J. Wilde, Optimum Seeking Methods, Englewood Cliffs, N.J.: Prentice-Hall, 1967.

[4] R. Hooke and T. A. Jeeves, "Direct search solution of numerical and statistical problems," J.ACM, vol. 8, pp. 212-229, April 1961.

[5] K. C. Gupta, R. Garg, R. Chadha, Computer-Aided Design of Microwave Circuits, Dedham, Massachusetts: Aertech House, 1981.

[6] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer J., vol. 3, pp. 175-184, October 1960

[7] Touchstone Users Manual, EESof, Westlake Village, CA., 1986.