ARBITRARY PRECISION LIBRARY FOR APPROXIMATE SYMBOLIC ANALYSIS OF ANALOG CIRCUITS

Ing. Martin HORÁK, Doctoral Degree Programme (2)
Dept. of Radio Electronics, FEEC, BUT
E-mail: horakma@feec.vutbr.cz

Supervised by: Dr. Zdeněk Kolka

ABSTRACT

This contribution deals with a mathematical library, which implements operations with complex matrices and vectors for approximate symbolic analysis of circuits. The each element of the matrix or the vector is implemented with user-defined precision. The core of the library is based on the arbitrary precision library GMP.

1 INTRODUCTION

The basic network functions such as input impedance, the voltage transfer function etc. are important while analyzing analog circuit behavior. There are two types of circuit analysis programs – program like the circuit simulator SPICE that gives user numeric values (in graph representation) and program that can produce symbolic expression as a quotient of two algebraic cofactors. While graph representation cannot determine influence of particular circuit part to circuit behavior, description using symbolic expression can determine it due to its generality. However, the symbolic analysis of a relatively small circuit with 10 nodes gives a huge expression that is not interpretable. It leads to some simplification techniques, which lower generality, but significantly improve interpretability of such expression. All approximation schemes require numerical values at desired frequency range of validity. There are three basic approaches to approximate symbolic analysis. A simplification before generation (SBG) is based on simplifying a circuit before generation of a symbolic expression - simply remove or contract admittances in the circuit which have insignificant influence to the network behavior. The next method, known as simplification during generation (SDG), generates directly simplified expression of a circuit. The last method is SAG, simplification after generation – firstly, the exact expression is generated and then the least significant terms are removed. This technique is usable only for small circuit because of huge amount generated terms. Mostly, techniques SAG and SDG are combined with SBG. Currently, the techniques SBG and SAG are implemented in Symbolic Network Analysis Program SNAP [1]. These techniques require computing nominal values at user defined design points (frequencies). Experiments show that for large circuits with 100 nodes, the 80-bit precision that is currently implemented in SNAP is not enough. This leads to using arithmetic with more (user defined) precision that enables solving sets of linear equations, computing determinant, inverse matrix, etc. more precisely.

2 LIBRARY

The library was developed in the Microsoft Developer studio .NET environment in C++ language. Arbitrary precision floating-point arithmetic was already developed by GNU project Multiple Precision Arithmetic Library (GMP) [2], so the part of this library was used as the core. The library implements basic operations with numbers of arbitrary precision as adding, multiplying, conversion functions from standard types as double, integer, etc. Each arbitrary precision number contains two parts – the mantissa and the exponent. The mantissa has a user selectable precision, limited only by available computer memory. The exponent of each number is a fixed precision; on 32 bit systems is approximately in the range $2^{-68719476768}$ to $2^{68719476736}$.

The main implementation of the new mathematical library contains three independent object classes

1) The class *CGmp* encapsulates some arbitrary precision function of GMP library.
2) The class *CComplex* contains two members – the real and the imaginary part of the number and provides operations with complex numbers.
3) The classes *CMatrix* and *CVector* implement operations as the matrix inversion, the matrix determinant and the Gaussian elimination.

Classes *CComplex*, *CMatrix* and *CVector* are implemented as templates so any type of operands can be used, such as float, double or *CGmp*. However, before use of class *CGmp*, the precision should be selected.

The class *CComplex* was designed to be faster as possible – operation multiply minimizes number of required multiplications [3], because this operation is slow. The eq. 1 requires 4 multiplications, one addition and one subtraction, while eq. 2 requires only three multiplications (ac,bd,(a+b)(c+d)), plus two additions and three subtractions

$$(a+ib)(c+id) = (ac-bd) + i(bc+ad)\,, \tag{1}$$

$$(a+ib)(c+id) = (ac-bd) + i[(a+b)(c+d) - ac - bd]\,. \tag{2}$$

Operation modulus (3) and dividing (4) of two complex operands was implemented to prevent undesirable overflows, underflow, or loss of precision [3]

$$|a+ib| = \begin{cases} |a|\sqrt{1+(b/a)^2} & |a| \geq |b| \\ |b|\sqrt{1+(a/b)^2} & |a| < |b| \end{cases}, \tag{3}$$

$$\frac{a+ib}{c+id} = \begin{cases} \dfrac{[a+b(d/c)] + i[b-a(d/c)]}{c+d(d/c)} & |c| \geq |d| \\ \dfrac{[a(c/d)+b] + i[b(c/d)-a]}{c(c/d)+d} & |c| < |d| \end{cases}. \tag{4}$$

Operations that the *CMatrix* class implements (as the matrix inversion or solving sets of linear equations) use full pivoting [3].

3 RESULTS

The random complex matrix was generated for the testing of the library. Fig. 1 shows required time for matrix inversion for precision 1024, 512, 256 and 128 bits using AMD Athlon XP 1200+ processor. Numerical values are in the tab. 1.

Time [ms]		**Matrix order N**						
		10	**30**	**50**	**75**	**100**	**125**	**150**
Precision in bits	**128**	14	311	1462	4877	11587	26108	44724
	256	18	421	1939	6339	15232	29743	50883
	512	24	571	2574	8525	20059	39207	67187
	1024	40	981	4467	14831	34820	67477	116438

Tab. 1: *Time for computing inverse matrix order N using precision 128, 256, 512 and 1024 bits*

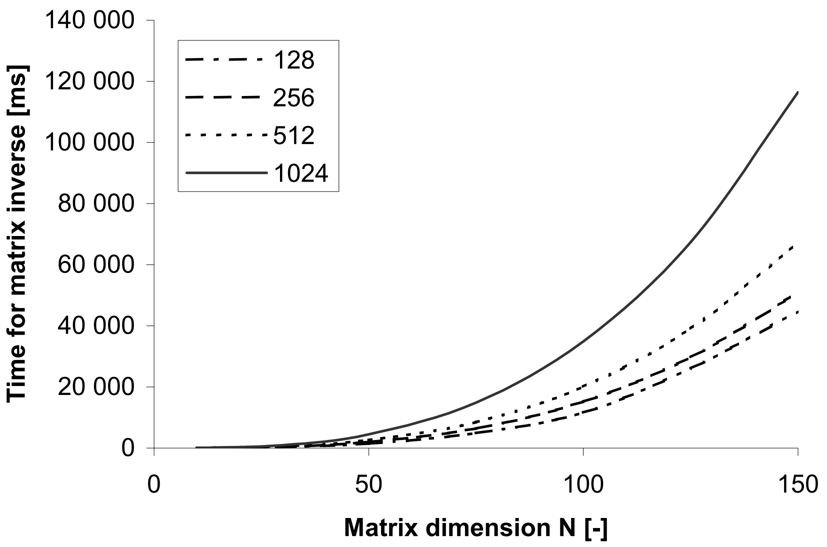

Fig. 1: *Time for computing inverse matrix order N with number precision 128, 256, 512 and 1024 bits*

It can be seen that time required for the matrix inversion significantly increases with matrix dimensions. Large circuits can contain ordinarily 100 nodes, so time for evaluation inverse matrix using precision 1024 bits takes approximately 35 seconds. It is obvious that the implementation of the arithmetic with arbitrary precision requires significantly more CPU power than commonly available 80-bit arithmetic in the Athlon processor.

The fig. 2 shows required time for computing the inverse matrix order N=100 as the function of required precision.

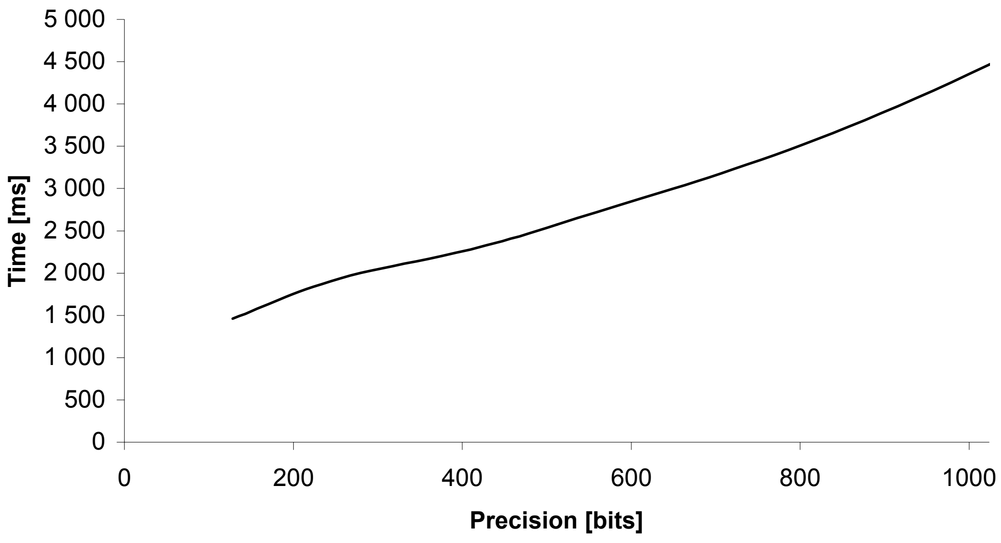

Fig. 2: *Required time for computing inverse matrix order N=100*

4 CONCLUSION

The arbitrary precision library for large complex matrices was described. The library can be used to solve problems, for that basic 80-bit precision is not enough, such as find zeros or poles of the network function. However, computing with matrices with arbitrary precision requires much CPU power, although it can be further reduced by use critical routines written in assembler. For solving larger set of linear equations than one hundred, some relaxation method is probably better choice instead of Gaussian elimination. The main use of this library is supposed by the program SNAP for symbolic analyzing of electrical circuits that was developed in the department of Radioelectronics in last years.

ACKNOWLEDGEMENTS

The contribution has been supported by Grant Agency the Ministry of education in contract number 1622/2004.

REFERENCES

[1] Kolka, Z.: New Version of Snap Program. In: Proc. Of the International Conference Telecommunications and signal processing TSP 2000. Brno 2000, p. 100-103.

[2] GNU Multiple precision arithmetic library, Free Software Foundation, http://swox.com/gmp/ (2004).

[3] Press W., Teukolsky A., Vetterling T., Flannery B.: Numerical recipes in C, 2nd edition, Australia, Cambridge University Press 2002, ISBN 052143108.