Chapter 1

AQUARELS : A PROBLEM-SOLVING ENVIRONMENT FOR VALIDATING SCIENTIFIC SOFTWARE

Olivier Beaumont
INRIA, Rennes (until 1999) - now Ecole Normale Supérieure of Lyon
Olivier.Beaumont@ens-lyon.fr

Jocelyne Erhel
INRIA, Rennes
Jocelyne.Erhel@inria.fr, http://www.irisa.fr/aladin/perso/erhel.html

Bernard Philippe
INRIA, Rennes
Bernard.Philippe@inria.fr, http://www.irisa.fr/aladin/perso/philippe.html

Abstract Numerical validation has become a major concern in many scientific applications. This paper presents the software environment Aquarels which integrates through a window-based interface various tools to tackle numerical validation. Aquarels includes tools to control floating-point arithmetic, perturbation tools, interval arithmetic, multi-precision arithmetic. Several real-life applications illustrate the efficiency of this software.

Keywords: Floating-point arithmetic, numerical stability, condition number.

Introduction

Modern current computer systems allow users to conduct numerical simulations at unprecedented scales. This is also due to the development of software systems that exploit the performance potential of these systems more effectively than ever before. Lesser though substantial efforts were put in the development of tools for detecting, analyzing and improving the numerical quality of large codes. Indeed, even though it was always known that computer arithmetic is not exact, many times the output from codes was not questioned as long as that output was produced fast enough. At other times, codes that were considered reliable were found to fail on new computers. Major steps to deal with these issues were taken from the 60's and beyond. These included formalism and techniques to study the effect of rounding errors on the quality of computed results. The first important step was to distinguish the issues of problem stability and algorithm stability; this allowed researchers to separate their investigations in the effects of each towards the total error in the final results [Wilkinson, 1963]. For a given problem, it is possible to bound the accuracy of the result, that is the forward error, if we know the precision of the data, the condition number of the problem and the backward error of the algorithm. A valid computation must take into account these three aspects. Another step was the establishment of a reliable standard for floating-point arithmetic [American National Standard, 1985].

Advances have also been made further up in the hierarchy. Problems and algorithms in some areas, linear algebra for instance, have been thoroughly studied during the last thirty years [Higham, 1995]. The library Lapack [Anderson et al., 1992] integrates most recent results and is rightly regarded as a reference for any further code development in this area. All algorithms included in Lapack could be regarded as backward stable. The library also contains code for estimating the condition number corresponding to most of the problems it handles. On the other hand, Lapack's excellent numerical features are not sufficient to guarantee or even provide information about the numerical quality of codes using Lapack as a component. Even numerical computations dealing primarily with linear algebra are expected to contain not only library calls to Lapack but also user-designed routines or calls to other libraries that are not as well analyzed. For these reasons, it would be desirable to have a tool that could provide information about the numerical quality of entire code suites.

Ideally, we would like to have a fully automated extraction of numerical quality indices for the entire code. This is very difficult as problems are too multifaceted and user requirements too varied. We note, however, that some tools already exist to help the user investigate some of the numerical characteristics of codes. Many of these tools have been described in the literature but it was thought that users would benefit if that technology were made easily available to them by wrapping them as a toolbox running behind an appropriate user interface. This was the motivation for building Aquarels [Erhel et al., 1991, Erhel et al., 1995, Beaumont et al., 2000]. With its current features, we consider the system to be a problem solving environment for the study of numerical quality and reliability of large codes. Aquarels was developed by and is available from Simulog thanks to grants from CNES, DGA and CEA and to scientific expertise from the authors. For up to date information we point to the relevant Web site[1]. Subsequently we assume that the reader is familiar with the basic concepts of accuracy and stability of numerical algorithms as explained in standard introductions to the subject; we refer to [Higham, 1995] for a detailed explanation of the state-of-the-art.

In the first section, we describe the environment Aquarels and its various tools. Then we give in the next three sections three examples of use in real-life applications.

1. OVERVIEW OF AQUARELS

Aquarels provides tools for addressing several types of problems. One is the analysis of numerical problems and possible identification and repair of bugs that could crash a code after many valid runs or the first time a code is ported on a new system. The second is for code validation. The third is to provide some special features for evaluating numerical quality including non-standard precision and interval arithmetic.

Aquarels provides multi-precision arithmetic for computing at very high precision, as well as truncated precision to study numerical stability.

The system also provides interval arithmetic. Interval arithmetic has become quite popular and is used in many areas [2]. Aquarels implements a library for interval operations including elementary functions evaluations based on directed rounding modes. This library also includes an accurate, up to machine precision, dot-product [Kulisch, 1983]. The straightforward use of interval arithmetic often leads to large intervals length because of two main effects [Bajard et al., 1997] : the subdistributivity, due to the decorrelation of the variables, and the wrapping effect, due to the enclosure of any compact of $\mathbb{R}^n$ in a box. Recent research on enclosure techniques has been attempting to overcome the aforementioned problems and thus allow interval arithmetic methods to return small intervals; see for example [Herzberger, 1994]. The Aquarels library Enclos implements such techniques for two important computational problems, namely polynomial evaluation and linear system solution [Neumaier, 1990, Rump, 1992, Rump, 1994].

Another class of techniques that have been applied with some success in analyses of numerical stability are perturbation techniques ; these are based on random perturbations of the operations and have been found to lead to useful estimates of numerical stability. The Cestac method, first defined in [Porte and Vignes, 1974], is built around this idea and leads to probabilistic accuracy estimates. Aquarels implements a basic version of Cestac, whereas a more advanced version is implemented in the software Cadna [Chesnaux, 1995].

In many cases, it is also possible to detect the effect of problem and algorithm stability using a statistical approach based on data perturbations. The key idea that was developed in the Scalp method [François, 1989] is to generate random perturbations of some variables defined as data, run the program several times with perturbations of varying size and to analyze their effect on the set of variables defined as results. A similar method, where residuals (when they exist) are also analyzed is developed in the method Precise [Chatelin and Frayssé, 1995]. Aquarels implements an improved version of Scalp [Erhel, 1993].

A fundamental difficulty in numerical computing is that we use finite precision arithmetic. Not so long ago, error analyses were complicated by the lack of consistency in the rounding, exception and other matters related to the implementation of floating-point arithmetic by different computer manufacturers. This lack of consistency was causing unexpected results when codes were ported to different or new systems. Fortunately, most manufacturers adhere by now to the IEEE standard for floating-point arithmetic [American National Standard, 1985]; the standard provides a solid uniform foundation for subsequent investigations. A fundamental property is required for the operations [Godunov et al., 1993] :

$$\forall \star \in \{+, \times, /\} \;\; |fl(x \star y) - x \star y| \leq \max(\epsilon |x \star y|, \epsilon_u) \tag{1.1}$$

where $fl(z)$ denotes the rounding, ϵ is the relative spacing, x_{min} is the smallest positive normalised number, $\epsilon_u = \epsilon x_{\min}$. Aquarels provides a user-friendly environment over the well-known program Paranoia [Karpinski, 1985] that reports on the consistency of the underlying computer system's floating-point arithmetic with the IEEE standard. It is also useful to apply the program to test the effect and possible violations or differences from the standard for some compiler options. The Pentium bug ([Muller, 1995]), which affected all chips of that particular model, has shown we are still not immune from serious problems caused by inadequate implementations. Other times, problems appear on fewer or on a single system. For example, we discovered in 1995 an instance of a floating-point chip of a workstation for which $\left(\sqrt{17}\right)^2$ was not equal to 17, and by far! Using Paranoia from Aquarels we were able to locate the bug in the rounding mode of the multiplication [Bajard et al., 1997].

Aquarels also includes the program Machar ([Cody, 1988]), which computes information concerning the arithmetic format (e.g. minimum and maximum values, machine epsilon, rounding mode, etc.) and thus is a useful supplement to Paranoia. Aquarels maintains a database with the results of Machar and Paranoia on each back-end computer. A database of mathematical and physical constants is also maintained by Aquarels.

Another issue is the numerical quality of the approximations obtained when invoking implementations of elementary functions [Muller, 1997, Lefevre et al., 1998]. Aquarels provides two tools for testing the implementation of such functions: the program Elefun [Cody and Waite, 1980], which tests the output of elementary functions by checking to what degree they satisfy certain mathematical identities and the program Maudit, which attempts a deeper control

[Tabarly, 1989] than Elefun. Maudit is based on an accurate computation of a reference basis obtained from the multi-precision library of Aquarels.

Finally, Aquarels hosts the symbolic algebra system Maple and the system Foresys[3], devoted to code analysis and restructuring.

The window-based interface allows the user to define the specialized environment necessary for each application. The aforementioned arithmetics are implemented via libraries. Typically, these are cumbersome to use so Aquarels defines an extension of Fortran-77 that creates new types of variables, in a way similar to Fortran-SC [IBM, 1990], and implements a precompiler.

2. COMPUTING DETERMINANTS

Many applications such as finite element codes and computational geometry require at some point the computation of the value of a determinant whose entries are polynomials in the data. For example, the area of a triangle is often computed using the coordinates of the nodes (x_1, y_1), (x_2, y_2), (x_3, y_3). One industrial code that we analyzed computed the area by means of the function

$$\begin{aligned} A &= \pm\tfrac{1}{2} \times \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix} \\ &= 1/2 \times |x_1y_2 + x_2y_3 + x_3y_1 - x_1y_3 - x_2y_1 - x_3y_2|. \end{aligned} \tag{1.2}$$

This formulation turns out to be sensitive to translations and is numerically unstable [Bajard et al., 1997]. For example, when coordinates have large magnitude compared with the final area, the effect of cancellation can lead to catastrophic loss of accuracy. We used the interval arithmetic facilities of Aquarels to investigate the effect of coordinate translation. In one experiment, we selected a triangle of area approximately 1 and translated it away from the origin. We then used formula (1.2) implemented in interval arithmetic to obtain the area and plotted the size of the interval enclosing the computed area as a function of the translation distance. Results are depicted with the solid line in Figure 1.1 and demonstrate the expected increase in the interval length as the distance from the origin increases. We then applied the same experiment but on the alternative formula

$$\begin{aligned} A &= \pm\tfrac{1}{2} \times \begin{vmatrix} x_1 & x_2 - x_1 & x_3 - x_1 \\ y_1 & y_2 - y_1 & y_3 - y_1 \\ 1 & 0 & 0 \end{vmatrix} \\ &= \tfrac{1}{2} \times |(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)|. \end{aligned} \tag{1.3}$$

which is proven to be numerically stable [Bajard et al., 1997]. Results are depicted with the dashed line in Figure 1.1. As expected, formula (1.3) is invariant with translation.

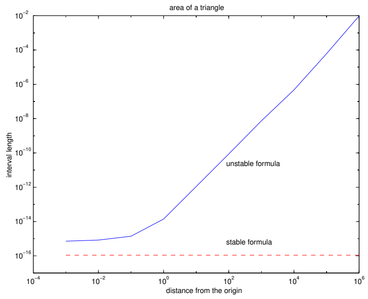

Figure 1.1 Computation of a triangle area : stability analysis

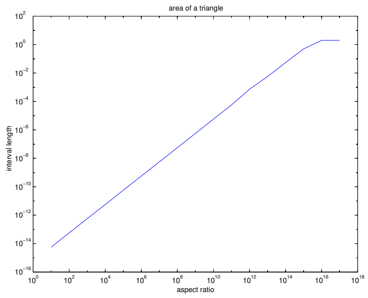

Figure 1.2 Computation of a triangle area : impact of aspect ratio

Consider once more the computation of a determinant corresponding to the area of a triangle. The condition number of this problem is known to depend directly on the aspect ratio of the triangle, defined for example by $\kappa = a/h$ where a is the length of the largest edge and h is the length of the associated height [Georges, 1990]. In our experiments we computed the areas of triangles with varying aspect ratios; these were generated by moving a vertex on a vertical line. We used the stable formula (1.3) and interval arithmetic in double precision. Figure 1.2 shows that, as expected, the accuracy of the computation is roughly equal to $\kappa\epsilon$.

3. COMPUTING THE MATRIX EXPONENTIAL

We next give an example with two algorithms to compute the same mathematical quantity. Some applications such as convection-diffusion models are based on linear partial differential equations. In simple cases there is an analytic expression for the solution. The problem reduces then to compute a matrix exponential. For an industrial application, we analyzed the computation of $\exp(tA)$ where A is a bidiagonal matrix of small size n, where the main diagonal is $-\lambda_1, \cdots, -\lambda_n$ and the subdiagonal is $\lambda_1, \cdots, \lambda_{n-1}$. We studied

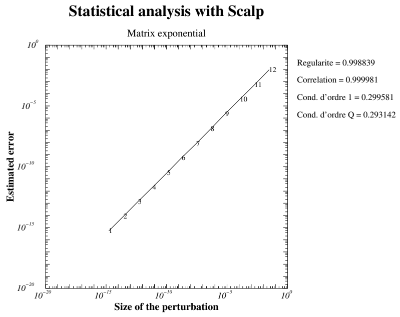

Figure 1.3 Matrix exponential - Example 1

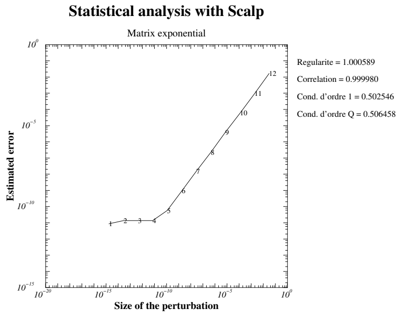

Figure 1.4 Matrix exponential - Example 2

two algorithms. The first one is a method based on the diagonalization of A. Let $A = VDV^{-1}$ where $D = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$ and V is the basis of eigenvectors then $\exp(tA) = V e^{Dt} V^{-1}$. The bases V and V^{-1} are computed by finite recurrences so that the method is direct and gives the exact result in exact arithmetic. This method is sensitive to rounding errors; when computing $\exp(tA)$, these errors are amplified by $\mathrm{cond}(V)$, the condition number of V [Loan, 1977, Moler and Loan, 1978]. The second method for the exponential is based on Padé approximants computed with squaring and scaling. In exact arithmetic this method only yields an approximation, however the algorithm is numerically stable.

The two algorithms were analyzed using Scalp from Aquarels. Results are depicted in Figures 1.3-1.5 and are tabulated in Table 1.1. Labels $\exp_D^{tA}$ and $\exp_P^{tA}$ in the table denote the results obtained with diagonalization and Padé approximation respectively. The results show that the diagonalization method becomes unstable as $\mathrm{cond}(V)$ increases whereas the Padé method remains accurate.

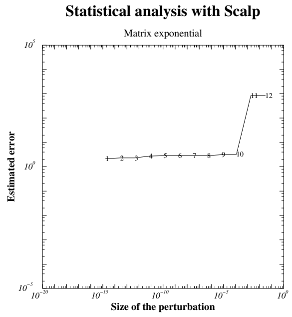

Figure 1.5 Matrix exponential - Example 3

Table 1.1 Computation of $\exp(tA)$ - Comparison between Diag and Pade

example	1	2	3
$\frac{\lVert \exp_D^{tA} - \exp_P^{tA} \rVert}{\lVert \exp_P^{tA} \rVert}$	10^{-14}	10^{-10}	1
nb of ops in Diag	1188	6090	6090
nb of ops in Pade	2987	26243	44243
$cond(V)$	5	10^6	10^{19}

4. SIGN OF AN ARITHMETIC EXPRESSION

Algorithms in computational geometry frequently require the evaluation of geometric predicates [Devillers, 1996, Force, 1996]; e.g. does a point lie left or right to the line defined by two other points ? Usually, one can express the result of the predicate by computing the sign of an arithmetic expression. We do not consider degeneracies (see [Alliez et al., 1997, Edelsbrunner and Mücke, 1990, Emiris and Canny, 1995, Force, 1996]), but only the evaluation of geometric predicates[Avnaim et al., 1997, Force, 1996, Burnikel, 1996]. One might argue that an erroneous evaluation of such a predicate corresponds to the exact evaluation of a slightly perturbed problem. Unfortunately, compounding several wrong evaluations could easily lead to a problem that cannot be considered to be a slight perturbation of the original one. In general, the solution of these problems might be inconsistent or the underlying algorithms might not terminate. Therefore, the reliable evaluation of geometric predicates becomes critical. The exact evaluation of a given predicate can be performed by using multi-precision arithmetic [Fortune and van Wyk, 1993, Burnikel et al., 1995], but at a high computational cost. Our aim is to design filters, in order to find an enclosure that does not contain zero. When zero belongs to the enclosure, the filter cannot conclude and another more expensive filter or exact computation must be used. One can design several filters, combining different precisions and interval arithmetic. This task is enhanced by the tools available in Aquarels.

In what follows, we consider the in-circle predicate.
Let $A = (x, y), A_i = (x_i, y_i), i = 1, 2, 3$, and

$$D = \det \begin{pmatrix} x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ \sqrt{x_1^2 + y_1^2} & \sqrt{x_2^2 + y_2^2} & \sqrt{x_3^2 + y_3^2} & \sqrt{x^2 + y^2} \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Depending ont the sign of D, the point A will lie inside or outside the circle defined by $A_1,\ A_2$ and A_3.

We consider four different filters, $i)$ SPAI : single precision arithmetic and intervals; $ii)$ SPAEB : single precision arithmetic and evaluation of an error bound; $iii)$ DPAI : double precision arithmetic and intervals; $iv)$ DPAEB : double precision arithmetic and evaluation of an error bound. A proven error bound is evaluated by assigning to each operation the maximal error due to finite precision. Table 1.2 shows the results we obtained in an "easy" case, where all four points have random coordinates in $[-0.5, 0.5]$ and in a "difficult" case, where $A_1,\ A_2$ and A_3 have random coordinates in $[-0.5, 0.5]$ and $A = A_1 + \begin{pmatrix} 10^{-6} \\ 10^{-6} \end{pmatrix}$. Coordinates are single precision numbers. The table displays

- the precision, defined by the ratio

$$\frac{\text{width of the predicate estimation}}{\text{mean value of the predicate evaluation}};$$

- the number of failures (cases where zero belongs to the predicate estimation).

All the results were obtained using 10^5 trials.
It is worth noting how interval arithmetic becomes useful in the so-called difficult situation. The benefits of interval arithmetic becomes even more obvious when considering high degree predicates, where the degree is defined as the degree of the polynomial expressing the result with respect to the geometric data [Burnikel, 1996].

Notes

1. `http://www.simulog.fr/aquarels/WWW/aquarels/HomePage.html.`
2. see for example the Web site http://cs.utep.edu/interval-comp/main.html
3. http://www.simulog.fr/FR/html/prods/sgi.html

Table 1.2 Filters for in-circle predicate

	easy case		*difficult case*	
	precision	*failures*	*precision*	*failures*
SPAI	1.17E-06	0	0.50	8896
SPAEB	8.96E-06	0	2.13	52125
DPAI	3.89D-16	0	1.20D-09	0
DPAEB	4.48D-11	0	6.50D-06	5

References

[Alliez et al., 1997] Alliez, P., Devillers, O., and J.Snoeyink (1997). Removing degeneracies by perturbing the problem or the world. Technical Report 3316, INRIA.

[American National Standard, 1985] American National Standard, A. (1985). IEEE standard for binary floating-point arithmetic. ANSI/IEEE Std. 754, IEEE.

[Anderson et al., 1992] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. (1992). *LAPACK Users' Guide*. SIAM.

[Avnaim et al., 1997] Avnaim, F., Boissonnat, J.-D., Devillers, O., Preparata, F., and Yvinec, M. (1997). Evaluating signs of determinants using single-precision arithmetic. *Algorithmica*, 17:111–132.

[Bajard et al., 1997] Bajard, J.-C., Beaumont, O., Chesneaux, J.-M., Daumas, M., Erhel, J., Michelucci, D., Muller, J.-M., Philippe, B., Revol, N., Roch, J.-L., and Vignes, J. (1997). *Qualité des calculs sur ordinateur. Vers des arithmétiques plus fiables?* Masson, Paris.

[Beaumont et al., 2000] Beaumont, O., Erhel, J., and Philippe, B. (2000). S-tudy of the numerical quality of some real-life applications with the system Aquarels. Research report, INRIA.

[Burnikel, 1996] Burnikel, C. (1996). *Exact Computation of Voronoi Diagrams and Line Segment Intersections*. PhD thesis, Technischen Fakultät der Universität des Saarlandes.

[Burnikel et al., 1995] Burnikel, C., Könemann, J., Mehlhorn, K., Näher, S., Schirra, S., and Uhrig, C. (1995). Exact geometric computation in leda. In *Proceedings of the 11th ACM Symposium on Computational Geometry*, pages C18–C19.

[Chatelin and Frayssé, 1995] Chatelin, F. and Frayssé, V. (1995). *Lectures on Finite Precision Computations*. SIAM.

[Chesnaux, 1995] Chesnaux, J.-M. (1995). *L'Arithmétique Stochastique et le Logiciel CADNA*. Habilitation à diriger des recherche, Université Paris VI.

[Cody, 1988] Cody, W. J. (1988). Machar : a subroutine to dynamically determine machine parameters. *ACM Trans. on Math. Software*.

[Cody and Waite, 1980] Cody, W. J. and Waite, W. (1980). *Software Manual for the Elementary Functions*. Englewoods Cliffs, Prentics Hall.

[Devillers, 1996] Devillers, O. (1996). Computational geometry and discrete computations. In Verlag, S., editor, *Proceedings of the 6th Discrete Geometry for Computer Imagery conference*, Lectures Notes in Computer Science.

[Edelsbrunner and Mücke, 1990] Edelsbrunner, H. and Mücke, E. (1990). Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104.

[Emiris and Canny, 1995] Emiris, I. and Canny, J. (1995). A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664.

[Erhel, 1993] Erhel, J. (1993). Experiments with data perturbations to study condition numbers and numerical stability. *Computing*, 51(1):29–44.

[Erhel et al., 1991] Erhel, J., Jégou, Y., and Philippe, B. (1991). Aquarels: a problem-solving environment for numerical quality. In *13th IMACS World Congress on Computation and Applied Mathematics*.

[Erhel et al., 1995] Erhel, J., Philippe, B., Mallejac, N., and Paoletti, J.-C. (1995). Aquarels: a problem-solving environment for numerical quality - review of the project and first use on realistic problems. In *ICIAM*.

[Force, 1996] Force, T. C. G. I. T. (1996). Applications challenges to computational geometry. Technical Report TR-521-96, Princeton University.

[Fortune and van Wyk, 1993] Fortune, S. and van Wyk, C. (1993). Efficient exact arithmetic for computational geometry. In *Proceedings of the 9th Symposium on Computational Geometry*, pages 163–171.

[François, 1989] François, P. (1989). *Contribution à l'étude de méthodes de contrôle automatique de l'erreur d'arrondi: la méthodologie SCALP*. Thèse de doctorat, INPG.

[Georges, 1990] Georges, P.-L. (1990). *Génération automatique de maillages - applications aux méthodes d'éléments finis*. Masson.

[Godunov et al., 1993] Godunov, S., Antonov, A., O.P.Kiriljuk, and Kostin, V. (1993). *Guaranteed Accuracy in Numerical Linear Algebra*. Kluwer Academic Publishers, Dordrecht/Boston/London.

[Herzberger, 1994] Herzberger, J., editor (1994). *Topics in validated computations*. Studies in Computational mathematics. North-Holland, Elsevier, Amsterdam.

[Higham, 1995] Higham, N. (1995). *Accuracy and Stability of Numerical Algorithms*. SIAM.

[IBM, 1990] IBM (1990). Acrith-xsc: Ibm high accuracy arithmetic - extended scientific computation. Technical Report PI 881, IBM Deutschland GmbH, Boblinger.

[Karpinski, 1985] Karpinski (1985). Paranoia: a floating-point benchmark. *Byte*.

[Kulisch, 1983] Kulisch, U. (1983). A new arithmetic for scientific computation. In Kulisch, R. and Miranker, W. L., editors, *A new approach to scientific computation*, pages 1–26. Academic Press.

[Lefevre et al., 1998] Lefevre, V., Muller, J., and Tisserand, A. (1998). Toward correctly rounded transcendentals. *IEEE Transactions on Computers*, 47(11).

[Loan, 1977] Loan, C. V. (1977). The sensitivity of the matrix exponential. *SIAM Journal on Numerical Analysis*, 14(6):971–981.

[Moler and Loan, 1978] Moler, C. and Loan, C. (1978). Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836.

[Muller, 1995] Muller, J.-M. (1995). Algorithmes de division pour microprocesseurs : illustration à l'aide du "bug" du pentium. *Technique et Science Informatiques*, 14(8).

[Muller, 1997] Muller, J.-M. (1997). *Elementary Functions, Algorithms and Implementation*. Birkhauser Boston,.

[Neumaier, 1990] Neumaier, A. (1990). *Interval methods for systems of equations*. Cambridge University Press.

[Porte and Vignes, 1974] Porte, M. L. and Vignes, J. (1974). Etude statistique des erreurs dans l'arithmétique des ordinateurs ; application au contrôle des résultats d'algorithmes numériques. *Numerische Mathematik*, 23(1):63–72.

[Rump, 1992] Rump, S. (1992). On the solution of interval linear systems. *Computing*, 47:337–353.

[Rump, 1994] Rump, S. (1994). Verification methods for dense and sparse systems of equations. In Herzberger, J., editor, *Topics in validated computations*, Studies in Computational mathematics, pages 63–135. North-Holland, Elsevier, Amsterdam.

[Tabarly, 1989] Tabarly, J. H. (1989). Maudit : Qualification des fonctions intrinsŁques. Manuels technique et d'utilisation, CAP SESA.

[Wilkinson, 1963] Wilkinson, J. (1963). *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, NJ.