Serdica J. Computing **4** (2010), 73–84

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

COMPUTER-ASSISTED PROOFS AND SYMBOLIC COMPUTATIONS*

Walter Krämer

Abstract. We discuss some main points of computer-assisted proofs based on reliable numerical computations. Such so-called self-validating numerical methods in combination with exact symbolic manipulations result in very powerful mathematical software tools. These tools allow proving mathematical statements (existence of a fixed point, of a solution of an ODE, of a zero of a continuous function, of a global minimum within a given range, etc.) using a digital computer. To validate the assertions of the underlying theorems fast finite precision arithmetic is used. The results are absolutely rigorous.

To demonstrate the power of reliable symbolic-numeric computations we investigate in some details the verification of very long periodic orbits of chaotic dynamical systems. The verification is done directly in Maple, e.g. using the Maple Power Tool `intpakX` or, more efficiently, using the C++ class library C-XSC.

ACM Computing Classification System (1998): G.1.0, G.1.5, G.4.

Key words: Computer-assisted proofs, symbolic computations, self-validating numerical methods, dynamical system, verified periodic orbit, intpakX, C-XSC.

*This work is partially supported by DFG: KR1612/7-1.

1. Introduction. Several computer assisted proofs concerning the Kepler conjecture, the Double Bouble conjecture, the Dirac-Schwinger conjecture, as well as some Chaos conjectures are discussed e.g. in [2]. More recent papers are [11, 17, 18]. The lists of references therein are also very beneficial and a good starting point for further investigations. This paper does not consider so called *formal proofs* and computer programs known as *proof assistants* [5].

Let us start our discussion about computer-assisted proofs with a theorem that has already been proved about 400 years ago [12]:

Theorem 1 (Ludolf van Ceulen 1610)**.**

$$\pi - 3.14159\ \ 26535\ \ 89793\ \ 23846\ \ 26433\ \ 83279\ \ 50288 < 10^{-35}$$

We call this statement a "theorem", because at the time the invention was made, it was extremely hard to find the given approximation and to prove its accuracy. Based on the algorithm of Archimedes using the perimeters of inscribed and circumscribed regular n-gons to the unit circle with more and more sides to find lower and upper bounds for the circumference 2π of the unit circle, it took Ludolf van Ceulen about 14 years of tedious and probably error prone manual numerical computations to find this result. The algorithm of Archimedes (about 250 BC) was well known since antiquity but at that time nobody checked the correctness of van Ceulen's "implementation" of it (the reader should compute some bounds to π using regular n-gons to get a better feeling of what has to be done: derive formulas going from the side length of a regular n-gon to the side length of a regular $2n$-gon, compute manually bounds for $\sqrt{2}$ and $\sqrt{3}$, think about a stopping criterion, think about error propagation, and so on [7, 12]). At that time the computer was a human being probably much more error prone in doing numerical tasks using 40-digit numbers than a digital computer of today. Nevertheless, the theorem is correct and was always regarded as correct. Replacing the numerical calculations done by van Ceulen by numerical computations on a digital computer, using directed rounded multiple precision operations or multiple precision interval operations, lead to what we nowadays call a computer-assisted proof. Or the other way around: call van Ceulen a "computing device", then the proof from around 1600 is a perfect computer-assisted proof.

2. Symbolic manipulations/self-validating methods. Symbolic manipulations use built-in mathematical knowledge to find results like [4]

$$\sin\left(\frac{\pi}{5}\right) = \frac{\sqrt{2}\sqrt{5-\sqrt{5}}}{4}.$$

Such an equality result cannot be verified by a self-validating numerical method based on some kind of machine numbers with only finitely many digits. Even using multiple precision numbers does not allow a reliable decision.

The nature of the next example is very similar but more challenging:

$$16\cos(x)^3\cosh\left(\frac{x}{2}\right)\sinh(x) - 6\cos(x)\sinh\left(\frac{x}{2}\right) - 6\cos(x)\sinh\left(\frac{3x}{2}\right)$$

$$= \cos(3x)\left(\exp\left(\frac{3x}{2}\right) + \exp\left(\frac{x}{2}\right)\right)(1 - \exp(-2x)).$$

In this case it is also quite difficult to establish equality using a computer algebra system like Maple. But it is still doable with some (manual) help. Again, this result cannot be verified numerically, due to the fact that an arbitrarily small change in the input data (here the formulas left and right of the equality sign) leads to a totally different result (equality changes to inequality). Note establishing the fact of inequality can be done by using reliable numerical methods, e.g. using multiple precision interval computations: if the intersection of the left hand side interval enclosure with the interval enclosure of the right hand side is empty, equality cannot hold. The fact that the left hand side and the right hand side differ does not change for sufficiently small changes in the input data (in contrast to checking for equality as described above).

As a further example let us briefly discuss integration. Risch's algorithm (1969, see [15, 16]) can be used to decide algorithmically whether some function f consisting of basic arithmetic operations, roots, powers, and elementary functions can be integrated in finite terms, and if so, to compute a close formula for F for the integral. E.g., it allows proving that no finite combination of arithmetic operations, powers, or elementary functions equals $\int \exp(x^2)dx$, i.e., there is no finite combination $F(x)$ of elementary operations whose derivative $F'(x)$ is equal to $f(x) = \exp(x^2)$. Now, the question is how reliable the current implementations of this algorithm are. Can we really rely on say Maple's implementation? This question arises because transforming the Risch decision procedure into an algorithm, which can be executed by a computer, is a very complex task that requires the use of heuristics and many refinements (no software – as of March 2008 – is known to implement the full Risch algorithm, see `http://en.wikipedia.org/wiki/Risch_algorithm`). E.g., no software is known to find an antiderivative for

$$f(x) = \frac{x^2 + 2x + 1 + (3x+1)\sqrt{x + \ln x}}{x\sqrt{x + \ln x}(x + \sqrt{x + \ln x})},$$

while the antiderivative for this expression has the short form

$$F(x) = 2(\sqrt{x + \ln x} + \ln(x + \sqrt{x + \ln x})) + C,$$

C being an arbitrary constant value (we checked this statement for Maple 13, 2009). Note: showing that $F(x)$ is really the antiderivative of $f(x)$ is quite simple. Just compute $F'(x)$ and show that the result is indeed equal to $f(x)$. Doing this is quite a simple task for Maple. Thus we have to ask: if the result of say Maple's implementation of Risch's algorithm applied to the integrand $f(x)$ does not find the antiderivative, does this really mean that the antiderivative cannot be composed using only a finite number of elementary operations? More generally: We have to think about the question: "What is a proof and what is a computer-assisted proof?"

3. Some general remarks on "What is a proof?". Let us list some thoughts concerning mathematical proofs:

- The traditional mathematical proof can be followed line by line with pencil and paper (this is still the classical way mathematicians work).
- Human beings might draw false conclusions.
- There are examples of theorems that were accepted for years but proved later to be false.
- A proof becomes approved by human beings reading, understanding, and accepting it.
- If a proof becomes lengthy and difficult, naturally the number of mathematicians who really follow the ideas decrease.
- Acceptance of a proof is based on mutual trust: if trustworthy colleagues accept a proof, we also tend to accept it.

There are different levels of trust:

- The line of human (expert) reasoning is perfectly accepted.
- The use of a pocket calculator is widely accepted.
- The use of computer algebra systems is more or less accepted.

- The use of floating-point arithmetic seems questionable to many mathematicians.

However:

- Humans may draw (and have drawn) false conclusions.
- There are examples where pocket calculators give wrong results.
- There are examples where computer algebra systems give wrong results.
- There is a possibility of using floating point operations to get guaranteed results: interval computations (trusting in hard- and software; see also [1])

Let us now consider the last point in more detail.

3.1. Remarks on interval computations.

- To do interval computations, floating point operations are replaced by (machine) interval operations and numerical and symbolic constants by intervals that contain these quantities.
- If an interval algorithm works, we get a guaranteed set valued numerical result.
- If an algorithm does not work (e.g., due to ill conditioning, insufficient exponent range, inappropriate method, overestimation effects, ...), the user gets a message, i.e., no false conclusions are drawn and no fictitious solutions are computed (in contrast to pure floating point algorithms).
- Meanwhile, a wide range of applications (expression evaluations, interval systems of linear equations, differential equations, integral equations, optimization problems, first results with PDEs, ...) is covered by interval methods.
- Interval operations are based on (directed rounded) floating point operations, i.e., they are very fast compared to symbolic computation methods.

Why does interval arithmetic not have the best reputation? Used naively, interval results tend to give (useless) bounds $-\infty$ to $+\infty$ on the true result due to overestimations and due to the so-called dependency problem. Therefore a warning: Do not construct interval methods naively. Typically, interval methods first compute an ordinary floating point approximation $\tilde{x}$ to the solution x of the

problem. Using this approximation and feeding back (in some clever way) the original data of the problem, a range computation resulting in the interval E for the error of this approximation with respect to the true solution is done. If the method works on a digital computer, we finally get

$$x \in \tilde{x} + E$$

and this result is guaranteed in the sense of a mathematical proof.

3.2. The interval Newton method. The interval Newton method allows a computer-assisted proof for the existence or non-existence of a zero of a nonlinear system of equations described by a sufficiently smooth function.

For the continuously differentiable function $f : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ in an m-dimensional interval $[x] \subset \mathbb{R}^m$ we define for a point $x \in [x]$ the interval Newton operator

$$N([x]) := x - f'([x])^{-1} f(x).$$

Here, $f'([x])^{-1}$ denotes the interval matrix containing all matrices of the form $f'(x)^{-1}$ for $x \in [x]$.

Theorem 2. 1.) *If $N([x]) \subseteq [x]$, then there exists exactly one point $x^* \in [x]$ s. t. $f(x^*) = 0$.*
2.) *If $N([x]) \cap [x] = \emptyset$, then there is no zero of f in $[x]$.*
3.) *Remark: If $N([x])$ only intersects $[x]$ but is not contained in $[x]$ no conclusion about the existence of a zero of f in $[x]$ can be drawn without further investigations.*

Proof concerning 1.) and 2.): a traditional mathematical proof performed by pencil and paper based on Brouwer's fixed point theorem (note: interval vectors are nonempty, compact, convex sets by definition). Theorem 2 means that we have to check

1.) $N([x]) = x - f'([x])^{-1} f(x) \subset [x]$
or
2.) $N([x]) \cap [x] = \emptyset$.

On a digital computer we verify the inclusion property using machine interval operations. Overestimation in computing the left hand side does not invalidate the statement of the theorem. If the machine evaluation $N_\diamond[x]$ of $N[x]$ fulfills $N_\diamond([x]) \subset [x]$, we trivially have $N([x]) \subset [x]$, proving the existence of a zero $x^* \in N_\diamond([x])$ of f.

The same reasoning works in case of 2.)

4. Computer-assisted proof – a case study. In this section we want to apply the interval Newton method to prove period n-cycles for large n of a (chaotic) dynamical system. We will see that it is necessary to adapt the more or less theoretical considerations of Paragraph 3 to the concrete situation. The Jacobian involved in the Newton operator is of dimension n. For n about 1 000 000 or even larger (we are interested in cycles with very long period) it is not possible to work with full matrices. We will see that the Jacobian is very sparse and that working with only a few arrays of length n instead of an n by n matrix will lead to a successful computer-assisted proof of very long period n-cycles.

4.1. Proving long period n-cycles. The dynamical system is taken to be the very simple logistic map

$$x_{n+1} = rx_n(1 - x_n), n = 0, 1, \ldots$$

where $x_n \in [0, 1]$ represents the population at year n, and hence x_0 represents the initial population (at year 0). $r \in [1, 4]$ represents a combined rate for reproduction and starvation. The numerical behaviour of this iteration is very complex (chaotic) for parameter values r close to 4. E.g., to find just the first two leading digits of x_{1000} for the parameter value $r = 375/100$ and $x_0 = 1/2$ one has to perform more than 150-digit long multiple-precision computations [9]. In each iteration step only 3 arithmetic operations are performed. Nevertheless, finding x_{1000} using rational arithmetic is not possible. To represent x_{20} exactly as a rational number we already need 1893916 decimal digits. And going from x_k to x_{k+1} typically doubles the number of digits needed to represent the new iterate [9].

We are interested in proving that the logistic equation with given values for r and x_1 has a period n-cycle. Let us denote the iteration formula describing the logistic map by f, i.e.,

$$f : \mathbb{R} \longrightarrow \mathbb{R} \quad \text{with} \quad f(x) = rx(x - 1)$$

and let g be defined by

$$g(x) := x - \underbrace{f(f(\ldots f}_{n \text{ times}}(x)\ldots)).$$

Applying the interval Newton method to g we get: $N([x]) \subset [x] \longrightarrow$ period-n cycle of $f(x)$ is guaranteed. This direct approach works only for small values of n. For larger n the diameters of $g([x])$ and $g'([x])$ are large (e.g., due to wrapping).

But we can use the following alternative approach: Define the continuously differentiable function $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n, z := (x_1, x_2, \ldots x_n)^T$:

$$\begin{aligned} F_1(z) &:= x_2 - f(x_1) \\ F_2(z) &:= x_3 - f(x_2) \\ &\ldots \\ F_{n-1}(z) &:= x_n - f(x_{n-1}) \\ F_n(z) &:= x_1 - f(x_n) \end{aligned}$$

z is a zero of F iff x_1 is a fixed point of $g(x) := x - \underbrace{f(f(\ldots f}_{n \text{ times}}(x)\ldots))$.

Let us find the interval Newton operator for F. It holds: $\partial F_i(z)/\partial x_i = -f'(x_i)$, $i = 1, \ldots, n$ and $\partial F_i(z)/\partial x_{i+1} = 1$, $i < n$.
For the Jacobian F' of F at $z = (x_1, \ldots, x_n)^T$ we get:

$$F'(z) = \begin{pmatrix} -f'(x_1) & 1 & 0 & 0 & \ldots & 0 \\ 0 & -f'(x_2) & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & \ddots & & \\ & & & & & \\ & & & & & \\ 0 & 0 & \ldots & 0 & -f'(x_{n-1}) & 1 \\ 1 & 0 & 0 & \ldots & 0 & -f'(x_n) \end{pmatrix}.$$

To prove the existence of a unique periodic orbit in the interval vector $[z]$ we have to verify $N([z]) = z - F'(([z])^{-1}F(z) \subset [z]$.

Difficulty: computing $F'([z])^{-1}F(z)$ if n is large. The special structure of the Jacobian F' allows proceeding as follows: Let $[z] \in I\mathbb{R}^n$ and $z \in [z]$. Compute intervals

$$\begin{aligned} [a_k] &:= 1/f'([x_k]),\ k = 1, \ldots, n \\ [g_k] &:= f(x_k) - x_{k \bmod n + 1},\ k = 1, \ldots, n \\ [h_1] &:= (1 - [a_1][a_2]\cdots[a_n])^{-1} \sum_{i=1}^{n} [a_1]\cdots[a_i][g_i] \\ [h_k] &:= [a_k]([h_{k \bmod n + 1}] + [g_k]),\ k = n, \ldots, 2, \end{aligned}$$

then $z - [h]$ is an enclosure of $N([z])$ and, if $z - [h] \subset [z]$, it follows that there exists a unique period-n orbit of f in $[z]$.

4.2. Some numerical results. Galias [3] used the method described above in 2002 to prove e.g. for parameter $r = 3.9$ and starting at $x = 0.7$ a periodic orbit of length $n = 8\ 076\ 157$. We reverified this result using Maple together with the Maple Power Tool `intpakX` [8]. Using the more efficient C++ class library for verified scientific computations C-XSC [6] we additionally verified an even longer period n-cycle for e.g., the prime number $n = 10\ 032\ 157$, $r =$roundToNearest(39/10), and x close to 0.62. The C-XSC source code is available from [19]. The program produces the following output:

```
n: 10032157
r: [  3.8999999999999999,  3.9000000000000004]
seed:  0.62

Calculation of initial approximation using forward iteration...

x[  1]:   0.6200000000000000
x[n+1]:   0.5447592057705667

Floating point Newton steps ...

   x[  1]:   0.6200000000000000
   x[n+1]:   0.6167104600538781

   x[  1]:   0.6200000000000000
   x[n+1]:   0.6199924494913677

   x[  1]:   0.6200000000000000
   x[n+1]:   0.6199999999598529

   x[  1]:   0.6200000000000000
   x[n+1]:   0.6199999999999999

Starting interval Newton method using the last improved
approximation to verify periodic orbit ...

Result VERIFIED (inclusion property fulfilled):
   x[  1]: [  0.6199999999999996,  0.6200000000000005]
   x[n+1]: [  0.6199999999999987,  0.6200000000000013]
```

First an initial approximation to the orbit is computed via an ordinary floating

point forward iteration (n steps). It can be seen that the initial approximation is far away from being periodic (`x[1]` is far away from `x[n+1]`). After a few Newton steps based on conventional floating point operations to improve the initial approximation, an interval Newton step is performed. It allows verifying successfully the period 10 032 157-cycle.

5. Conclusion. Self-validating methods use machine interval operations in a sophisticated way to produce guaranteed results (proofs). Interval operations are based on directed rounded floating point operations. The realization of such operations in hardware or software is based on quite simple well known algorithms [10]. The level of trust in such operations (which can be performed in a purely mechanical way) seems to be orders of magnitude higher than the level of trust in the infallibility of human reasoning. The reliance in the work of engineers when realizing simple and well known algorithms should be not less than the trust in correct reasoning of mathematicians to prove mathematical facts. In this sense, computer-assisted proofs based on self-validating numerical methods are as trustworthy as ordinary proofs done by mathematicians. This is also true for proofs based on (error free) symbolic manipulations as they are available e.g. in computer algebra packages. Whereas symbolic manipulations often are very slow and lead to very large and cumbersome expressions, self-validating methods are (very) fast. But nevertheless, symbolic computations may help to improve self-validating methods significantly, e.g. by providing a formula simplification mechanism or providing structural information about the problem under consideration.

Computer-assisted proofs will profit by merging symbolic computations and self-validating methods. Up to now, only the first steps have been made in this direction (see e.g., [13, 14, 8] as well as [20] and references given there). Let's go ahead.

As a final remark we want to mention that in 2008 a new GAMM activity group "Computer-assisted proofs and symbolic computations" was founded. See `http://www.math.uni-wuppertal.de/org/WRST/gamm/`

REFERENCES

[1] Bohlender G., J. W. von Gudenberg, W. L. Miranker. Floating-point systems for theorem proving. In: Computer Aided Proofs in Analysis

(Eds K. R. Meyer, D. S. Schmidt), IMA Volumes in Mathematics and Its Application, Springer Verlag, 1991, 22–32.

[2] Frommer A. Proving conjectures by use of interval arithmetic. In: Perspective of Enclosure Methods (Eds U. Kulisch et al.), Springer, Wien, 2001, 1–13.

[3] Galias Z. Interval Newton method and backward shooting for finding long periodic orbits in 1D chaotic maps. In: Proc. of the Int. Workshop Nonlinear Dynamics of Electronics Systems, NDES'00, Catania, 2000, 100–104.

[4] Gräbe H.-G. Einführung in das symbolische Rechnen. Skript, University of Leipzig, 2009.

[5] Harrison J. Formal Proof – Theory and Practice. *Notices Amer. Math. Soc.*, **55** (2008), No 11, 1395–1406.

[6] Hofschuster W., W. Krämer. C-XSC and closely related software packages. In: Numerical Validation in Current Hardware Architectures, (Eds Cuyt et al.), LNCS, **5492** (2009), Springer-Verlag Berlin Heidelberg, 68–102.

[7] Krämer W. Multiple-precision computations with result verification. In: Scientific Computing with Result Verification (Eds E. Adams, U. Kulisch), *Mathematics in Science and Engineering*, **189** (1993), Academic Press, Inc., 325–356.

[8] Krämer W. Introduction to the Maple Power Tool `intpakX`. *Serdica Journal of Computing*, **1** (2007), No 4, 469–504.

[9] Krämer W. Accurate computation of chaotic dynamical systems. In: Proceedings to Mathematics and Computers in Biology and Chemistry MCBC 07 (Ed. A. Aggarwal), Vancouver, Canada, 2007, 74–79.

[10] Kulisch U. Computer Arithmetic and Validity: Theory, Implementation, and Applications. de Gruyter-Verlag, Berlin, 2008.

[11] Neumaier A. Computer-assisted proofs. In: Proc. 12th GAMM-IMACS Symp. Sci. Comp. (Eds W. Luther, W. Otten), SCAN 2006, IEEE Computer Society, 2007.

[12] Oomes R. M. Th. E., J. J. T. M. Tersteeg, J. Top. The epitaph of Ludolph van Ceulen. *Nieuw Arch. Wiskd.* (5), **1** (2000), No 2, 57–62.

[13] Popova E. Computer-Assisted Proofs in Solving Linear Parametric Problems. In: Conference Post-Proceedings of the 12th GAMM-IMACS International Symposium on Scientific Computing, Arithmetic and Validated Numerics (SCAN 2006), Duisburg-Essen, IEEE Computer Society Press. http://doi.ieeecomputersociety.org/10.1109/SCAN.2006.12, 2007.

[14] Popova E. Mathematica connectivity to interval libraries filib++ and C-XSC: In: Numerical Validation in Current Hardware Architectures (Eds A. Cuyt et al.), LNCS **5492**, 2009, Springer, Berlin, Heidelberg, 117–132.

[15] Risch R. The problem of integration in finite terms. *Transactions American Mathematical Society*, **139** (1969), 167–189.

[16] Risch R. The Solution of the problem of integration in finite terms. *Bulletin American Mathematical Society*, **76** (1970), 605–608.

[17] Rump S. M. Computer-assisted proofs and self-validating methods. In: Handbook on Accuracy and Reliability in Scientific Computation (Ed. B. Einarsson), SIAM 2005, 195–240.

[18] Plum M. Existence and multiplicity proofs for semilinear elliptic boundary value problems by computer assistance. *DMV Jahresbericht*, **110** (2008), 19–54.

[19] http://www.math.uni-wuppertal.de/~xsc/cxsc/examples/

[20] http://drops.dagstuhl.de/opus/volltexte/2006/454/

University of Wuppertal
42119 Wuppertal
Germany
e-mail: kraemer@math.uni-wuppertal.de

Received November 6, 2009
Final Accepted February 4, 2010