as many other fields of human activities. At present, computers are playing an increasingly central role in mathematics; they have found the application in almost every branch of mathematics. Many practical problems are solved by numerical methods of various types, for instance, simulating dynamical systems and determining their global properties, or calculating approximate solutions to nonlinear equations where no closed-form solution is available. Symbolic computation, a part of computer algebraic systems, is manipulating, simplifying, factorizing, and expanding complicated expressions that contain variables and non-numerical values. This powerful tool is very useful for solving very difficult mathematical problems producing, in addition, *exact computation*. Graphical representations can visualize complex objects to a good extent and thereby comprehend their properties, see [1].

For a long time (many decades and even centuries) a lot of mathematical problems remained unsolved. Simply, "paper-and-pencil methods", human-memory limitation, impossibility to handle lengthy expressions, primitive computer tools (logarithmic tables, abacus, slide rule) and other objective obstacles, were insufficient to solve them. These problems resisted until the digital computer era emerged. In this paper, we present a short review of some typical mathematical problems solved by computer tools (Section 2) and some new original contributions (Section 3).

2. Computers in mathematical research - a review

First applications of computers in mathematics were restricted to the calculation of complicated numerical expressions and the verification of some particular mathematical identities, relations and other issues. The *brutal force* of computers was used to suggest or test general claims and to pose hypotheses based on a finite number of patterns.

Let us mention some well-known examples concerned with the application of computers. The first calculation of the number π happened in 1949, when the outstanding scientist John von Neumann and his team used a room-sized digital computer with vacuum tubes ENIAC (Electronic Numerical Integrator And Computer) to compute 2 037 digits of π. The time of calculation: 70 hours. Computer-assisted proof of the four-color theorem, given by Appel and Haken in 1977, is a typical example where brute force combinatorial enumeration played an essential role in solving this 125 years old open problem (posed by F. Francis Guthrie in 1852). A similar combinatorial enumeration method (combined with interval arithmetic) was used in Thomas Hales's proof of the Kepler conjecture (posed in 1611), which asserts that the optimal density of packing equal spheres is achieved by the familiar face-centered cubic packing (see, e.g., [2], [3], and pretty interesting, on the markets where oranges are packed).

Today, computers are employed in mathematical research in a number of ways; one of the simplest ways is the implementation of *proof-by-exhaustion*: posting a proof so that a statement is valid for a large but finite number of cases and then check all the cases by a suitable program using a computer. More sophisticated use of computers is to discover and analyze interesting patterns in data, which then

serve to state conjectures. Helping to find conjectures is the first step, a proper advance is a rigorous proof of them.

Extensive development of computer algebra systems (briefly CAS), such as *Mathematica* and *Maple*, provides very fast manipulations with complex mathematical expressions, a work beyond human ability. Can you check that the sequence "0123456789" appears in the decimal expansion of π? Using a computer, Yasumasa Kanada of the University of Tokyo found in 1997 that this sequence begins at position 17 387 594 880. Advance versions of CAS deliver new improvements and very powerful algorithms. Evaluating the infinite product

$$\prod_{n=2}^{\infty} \frac{n^4-1}{n^4+1},$$

Mathematica 6 (issued 2007) gives the result involving the Gamma function. *Mathematica* 10 (2014) delivers the answer directly:

$$\prod_{n=2}^{\infty} \frac{n^4-1}{n^4+1} = \frac{\pi \sinh \pi}{\cosh(\pi\sqrt{2}) - \cos(\pi\sqrt{2})}\,.$$

Both tasks are obviously missions impossible for humans.

Symbolic computation, embedded in computer algebra systems like *Mathematica* or *Maple*, was a great advance in manipulating very complicated expressions of more variables. Suitable algorithms implemented on current powerful computers can solve problems whose answers are algebraic expressions tens or thousands of terms long. David Bailey, a mathematician and computers scientist at Lawrence Berkeley National Laboratory and one of the world leaders in experimental mathematics, said: *"The computer can then simplify this to five or 10 terms. Not only could a human not have done that, they certainly could not have done it without errors."* In & 3.5 we will show how to construct new iterative methods for solving nonlinear equations and determine the order of convergence by using symbolic computation in CAS *Mathematica*. Besides, CAS provides a powerful computer visualization of data, which is a very useful tool in helping us understand the behavior of iterative processes, as shown in & 3.5.

2.1. Short list of mathematical problems solved by computer

Below we give a list of theorems proved (completely or partially) with the help of computer programs. It is assumed that this list is far from being exhaustive.

- Archimedes' cattle problem, **1965** (the most famous ancient Diophantine equation), was solved by H. C. Williams, R. A. German and C. R. Zarnke [4] using computers).
- Euler's wrong hypothesis, **1966**. In 1769 Euler stated that there is no nth degree which can be sum of less than n nth degrees of natural numbers. In 1966 L. L. Lander and T. R. Parker found by computer the counterexample for $n = 5$ in the form of identity $27^5 + 84^5 + 110^5 + 133^5 = 144^5$.

- Four color theorem, **1976**. The four-color theorem states that any map in a plane (divided into contiguous regions) can be colored using no more than four colors so that no two adjacent regions have the same color. The theorem was proved by Kenneth Appel and Wolfgang Haken (published in [5], [6]) by inspecting reduced graph configurations by a computer program. Widely accepted proof of the four-color theorem was given in 2008 by Georges Gonthier with general-purpose theorem-proving software [7].

- Perfect squared square of the lowest order, **1978**. The task is tilling one integral square using only other integral squares of different sizes. In 1978, using a computer program, the Dutch computer scientist A. J. W. Duijvestijn found the perfect squared square of lowest order consisting of 21 smaller squares.

- Mitchell Feigenbaum's universality conjecture in non-linear dynamics, **1982** (proved by O. E. Lanford using rigorous computer arithmetic);

- The non-existence of a finite projective plane of order 10, **1989** (proved by C. W. H. Lam, L. Thiel and S. Swiercz).

- Problems solved by interval arithmetic **1993+**: Kepler's conjecture [8] (partially applied), the existence of eigenvalues of the Sturm-Liouville problem [9], the bound of Feigenbaum constant [10], the double bubble conjecture [11], verification of chaos [12], [13], Lorenz attractor [14], etc.

- BBP (Borwein, Bailey, Plouffe) formula for π, **1996** (published in [15]):

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k}\Big(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\Big).$$

  BBP formula is revolutionary and fascinating since provides the determination, for example, the one-billionth hexadecimal digit (or the four billionth binary digit) of π without needing to compute any of the previous digits. Practical BBP algorithm for computing the requested *individual digit* of π was described in [16, pp. 121–125].

- Robbins conjecture, **1996**: All *Robbins algebra*, supplied with a single *binary operation* denoted by $\vee$ (OR) and a single *unary operation* denoted by $\neg$ (NEGATION) are *Boolean algebras*. This conjecture was proved by W. McCune in 1996.

- Kepler conjecture (from 1611) on the most density package of identical spheres in three-dimensional Euclidean space, **2000**. The measure of the *density* $\delta = V_s/V_c$ is the total volume V_s of all packed spheres divided by the total volume V_c of the container in the form of a cube assuming that the cube edge is infinitely large. In 2000 Tomas Hales completed the solution proving that the so-called *face-centered cubic packing* has the maximum density $\delta_{max} = \pi/\sqrt{18}$, just as Kepler assumed (see the book [3, pp. 137–147] for details). Hales' proof, published in [8], combines methods from the theory of global optimization, linear programming and interval arithmetic.

- Lorenz attractor, **2002**, known as 14th of Smale's problem. It is the solution of Lorenz's system that describes chaotic behavior. Its existence was shown by W. Tucker [14] using validated interval arithmetic and normal forms; he also proved that Lorenz attractor is so-called *strange attractor*. Lorenz attractor appears in fluid dynamics and illustrates the phenomenon now known as the *butterfly effect* which demonstrates sensitive dependence on initial conditions.
- NP-hardness of minimum-weight triangulation. The minimum-weight triangulation problem belongs to computational geometry and computer science that asks for the minimal sum of the length of perimeters which make a triangulation (subdivision by triangles) of a given polygon or the convex hull. In 2008 W. Mulzer and G. Rote proved that this problem is NP-hard.
- Optimal solutions for Rubik's Cube can be obtained in at most 20 face moves starting from arbitrary initial position, **2010** (computer-assisted proof was given by T. Rokicki, H. Koceimba, M. Davidson, J. Dethridge).
- The primality test of very large natural numbers and the factorization of very large numbers, **1949+**. Many cryptographic protocols are based on the difficulty of factoring large composite integers. At present, the largest prime number is $2^{82589933} - 1$ having 24 862 048 decimal digits (found by Laroche et al. in December of 2018).

Although the computer solution of the four-color theorem (1976) and the Kepler's conjecture (2006) attracted considerable attention in mathematics, the proofs were not accepted by all mathematicians who made a serious objection that the presented computer-assisted proofs (better to say, the program codes) were not verifiable for a human by hand. Their reaction with many arguments against Hales' computer-assisted proof was justified; for illustration, Hales' computer program consisted of 40 000 lines. Fortunately, these two stories had a happy ending. As mentioned above, in 2008 G. Gonthier [7] delivered widely accepted proof of the four-color theorem using general-purpose theorem-proving software. Hales started in 2003 with a project named FlysPecK (F, P and K standing for Formal Proof of Kepler) aiming to come up with a formal proof of the Kepler conjecture that can be checked by automated proof verifying software. After 14 years Hales and his team finished this challenging but very difficult project; their formal proof was published in the journal *Forum of Mathematics* in 2017.

2.2. Interval arithmetic and self-validated method

An important use of computers in proving mathematical hypotheses and problems, known as self-validating numerics, is a special kind of computation that preserves strong mathematical rigor. This approach uses interval arithmetic which provides the enclosure, control, and propagation of roundoff and truncation errors of the executed calculation. The fruitful feature of interval arithmetic is the *inclusion principle* (essentially meaning *subset property*) which assures that the results of computations or solutions of the posed mathematical problems are enclosed by the

set-valued output. In this way, it is possible to calculate upper and lower bounds on the sets of solutions. Therefore, self-validating numerical methods deliver true results.

The described very useful property has provided the application of interval arithmetic not only in mathematics but also in many scientific disciplines where the control of a true result is of primary interest. Some of the mathematical problems solved by self-validated methods is listed above. Note that German scientist Siegfried M. Rump (Technische Universität Hamburg) created a special software INTLAB, based on Matlab, intended for the implementation of interval arithmetic for solving a huge number of mathematical problems [17].

Note that Professor Urlich Kulisch, one of the greatest world experts in the field of computer architecture and interval arithmetic, claims that further advance in computer technology and software will lead to the weird situation that the accuracy of results obtained by a computer can only be verified with the help of a computer (again!) and interval arithmetic that would control the intermediate results at every step, see his monograph *Computer Arithmetic and Validity* [18].

2.3. Experimental mathematics

A relatively new approach to mathematics that makes use of advanced and powerful computing technology to investigate mathematical objects and identify properties and patterns is called *experimental mathematics*, the term introduced by J. Borwein, D. Bailey, R, Girgensohn and their contributors, see, e.g., the books [16], [19], [20]. Experimental mathematics, a growing branch of applied mathematics, provides computational methodologies of doing mathematics that include the use of computations for the following activities quoted in [16]:

(1) Gaining insight and intuition.

(2) Discovering new patterns and relationships.

(3) Using graphical displays to suggest underlying mathematical principles.

(4) Testing and especially falsifying conjectures.

(5) Exploring a possible result to see if it is worth a formal proof.

(6) Suggesting approaches for formal proof.

(7) Replacing lengthy hand derivations with computer-based derivations.

(8) Confirming analytically derived results.

In the book *Mathematics by Experiments* [16], J. Borwain and D. Bailey, the world-leading experts in experimental mathematics, gave the list of things computers do better than humans. We cite their list below:

- High precision integer and floating-point arithmetic;

- Symbolic computation for algebraic and calculus manipulations;
- Formal power-series manipulation;
- Changing representations, e.g., continued fraction expansions, partial fraction expansions, Padé approximations;
- Recursion solving (e.g., `Rsolve` in *Mathematica*);
- Integer relation algorithms, e.g., the PSLQ algorithm;
- Creative telescoping (e.g., the Gosper and Wilf-Zeilberger methods) for proving summation identities;
- Iterative approximations to continuous functions;
- Identification of functions based on graph characteristics;
- Graphics and visualization methods.

"Some of the algorithms involved in this list have had the great influence on the development and practice of science and engineering", wrote Dongarra and Sullivan in [21], and added often cited sentence: *"Great algorithms are the poetry of computation."*

2.4. Computer-assisted proofs

Attempts have been also made in the area of Artificial Intelligence research to create new proofs of mathematical theorems using machine reasoning techniques. A computer-assisted proof or automated theorem prover are relatively recent notions which mean that a mathematical proof has been generated (at least partially) by computer. The majority of computer-aided proofs of mathematical theorems up to now were the simple application of *proofs-by-exhaustion* of all items of the problem (*brute force, backtrack algorithms*), for example, in searching for counterexamples of hypotheses in Number theory or solutions of problems having a huge outcomes/configurations. In contrast to the exhaustion method, *interactive proof assistants* most frequently gives human-readable proofs which can be checked for correctness; hence it is considerably preferable. The third type, sometimes named *a proper* computer-aided proofs, is completely based on sets of axioms and logical statements of computer software and gives reliable and correct results. More details devoted to computer-assisted proofs can be found on the link `https://en.wikipedia.org/wiki/Computer-assisted_proof`.

As examples of important achievements in the field of computer-assisted proofs, let us mention theorem-proving packages and algorithms of Wilf-Zeilberger's type. Theorem-proving package methods, such as Microsoft's Z3 Theorem Prover (now available under MIT Open Source), can either verify certain types of statements or find a counterexample demonstrating that a statement is false. The Wilf-Zeilberger method (invented by Doron Zeilberger and Herbert Wilf in 1990) can perform symbolic computations working with variables instead of numbers to produce exact results in a general form free of roundoff errors.

2.5. Computer visualization

The development of high quality computer visualization enables entirely new and remarkable insights into a wide variety of mathematical concepts and objects. Today researchers are able to study the geometric aspects of many mathematical and engineering disciplines. Computer graphics have become powerful tools for discovering new properties on various topics of mathematics and constructing new very efficient algorithms. Undoubtedly, computer visualization delivers modern and novel perspectives of some mathematical topics yielding a new dimension and a deep insight into properties and behavior of many mathematical processes, as well as various processes and phenomena in physics, biology, chemistry, and other scientific disciplines.

As one illustration of high sophistication of computer visualization, we present Tupper's astounding formula

$$\frac{1}{2} < \left\lfloor \mathrm{mod}\Big(\Big\lfloor \frac{y}{17} \Big\rfloor 2^{-17\lfloor x \rfloor - \mathrm{mod}(\lfloor y \rfloor, 17)}, 2\Big)\right\rfloor.$$

published in 2001. Here $\lfloor x \rfloor$ denotes the floor function (the greatest integer part of a number x) and $\mathrm{mod}(a, m)$ is the remainder in dividing the integer a by the integer m (the mod function). The area of graphics is determined by $0 \leqslant x \leqslant 105$ and $k \leqslant y \leqslant k + 16$ where k is the natural number with 543 digits

960 939 379 918 958 884 971 672 962 127 852 754 715 004 339 660 129 306 651 505
519 271 702 802 395 266 424 689 642 842 174 350 718 121 267 153 782 770 623 355
993 237 280 874 144 307 891 325 963 941 337 723 487 857 735 749 823 926 629 715
517 173 716 995 165 232 890 538 221 612 403 238 855 866 184 013 235 585 136 048
828 693 337 902 491 454 229 288 667 081 096 184 496 091 705 183 454 067 827 731
551 705 405 381 627 380 967 602 565 685 016 981 482 083 418 783 163 849 115 590
225 610 003 652 351 370 343 874 461 848 378 737 238 198 224 849 863 465 033 159
410 054 974 700 593 138 339 226 497 249 461 751 545 728 366 702 369 745 461 014
655 997 933 798 537 483 143 786 841 806 593 422 227 898 388 722 980 000 748 404
719

Using Tupper's formula, a simple program in CAS *Mathematica*

```
ArrayPlot[Table[Boole[1/2 < Floor[Mod[Floor[y/17] 2^(-17 Floor[x]-
Mod[Floor[y], 17]), 2]]], {y,n,n+16},{x,105,-2,-1}],
PixelConstrained -> True, Frame -> False, ImageSize -> 400]
```

gives the *self-referential* "plot" presented in the figure below.

In fact, Tupper demonstrated a method of decoding a bitmap stored in the constant k; k is a simple monochrome bitmap image of the formula treated as a binary number and multiplied by 17. Note that Tapper's approach is a general-purpose method to draw any other image.

2.6. Symbolic computation

Symbolic computation, a part of Computer algebra serving as a bridge between Mathematics and Computer science, is handling non-numerical values. Symbolic computation is widely used to experiment in mathematics and to study and design formulas, algorithms and software that are used in numerical programs. Computer algebra systems that perform symbolic calculations contain a lot of routines to carry out many operations, like polynomial factorization, solving nonlinear equations, manipulation with very complicated expressions. They are also capable to expand or simplify mathematical expressions with symbols, or differentiate or integrate them, etc. It should be emphasized that, contrary to numerical computation, symbolic computation produces *exact computation* with expressions containing variables that are manipulated as symbols.

As an illustration of the use of symbolic computation we present everyday practical problem posed by George Polya, a Stanford professor, in *American Mathematical Monthly* article (1956). *In how many ways can you make change for a dollar?* We modify Polya's task and consider Serbian currency assuming that there are 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000 and 5000 coins or banknotes. Hence:

In how many ways can you make change for a banknote of 5000 *Serbian dinars?*

Problems of this type are solved by generating functions. Let P_k be the number of all possible ways of changes. The problem reduces to the generating function (the Serbian currency case)

$$\sum_{k=1}^{\infty} P_k x^k = \frac{1}{(1-x^1)(1-x^2)(1-x^5)(1-x^{10})(1-x^{20})\cdots(1-x^{2000})(1-x^{5000})}.$$

To find P_{5000} it is necessary to develop the expression on the right-hand side into geometric series and sum all coefficients standing next to x^{5000}. Using a *Mathematica* command

```
Series[1/((1-x)*(1-x^2)*(1-x^5)* (1-x^(10))*(1-x^(20))*(1-x^(50))
*(1-x^(100))*(1-x^(200))*(1-x^(500))*(1-x^(1000))
*(1-x^(2000)*(1-x^(5000),{x,0,5000}]
```

computer calculates $P_{5000} = 23\,303\,034\,594\,532$. It is impossible for a human to determine such a huge number. In the case of US currency, one obtains $P_{100} = 292$, which is reachable for a human so that Polya's task had a sense in 1956.

Symbolic computation has successfully substituted lengthy manual calculation with computer-based computation and manipulation. In this paper, we concentrate

in & 3.5 on methods and procedures for the construction, analysis and practical application of algorithms for solving nonlinear equations with the support of symbolic computation. We emphasize that the construction of presented root-solvers is most likely impossible without the use of this specific computer software.

2.7. Computer-assisted proofs: how much can we trust computers

The use of computers in mathematics is undoubtedly widespread and in unstoppable expansion. Many mathematicians have turned to numerical experiments, symbolic computation, computer visualization and other computer methods as their main tools for mathematical investigation. In that way, they have achieved extraordinary results. However, ignoring these advances, a number of researchers often underestimate the role of computers in mathematics. In some cases, their skepticism cannot be fully disregarded since there are some specialized fields in mathematics that do not need the use of computers. Recall that, without using a computer, Andrew Wiles solved the famous Fermat last theorem (stated by Fermat in 1637) in 1995, Grigori Perelman presented a proof of Poincaré's conjecture, one of the most important open problems in topology, through three papers made available in 2002 and 2003 on arXiv of Cornell University. The proof of the Riemann hypothesis on the locations of zeros of the Riemann zeta function (posed in 1859) has not yet been given. Many mathematicians believe that the Riemann hypothesis, one of the most important open problems in mathematics, will be proved by a human using an analytical method, not by computer tools.

It seems that another kind of disputable question is more serious. Today, in search for the exact result or ultimate truth, mathematicians, philosophers and computer scientists (among them, Turing, Voevodsky, Avigard, Teleman, Kim, Mancosu, Hanke), ask: *"How much can we trust computers, whether computer-assisted proofs have the mathematical sense, is it possible to verify so many logical steps, how to evaluate the reliability of the data, how to check that the computer source program is perfectly accurate, whether the researcher can fully believe in the perfect work of hardware, what if there is a bug?, etc."* Errors of this kind could be sometimes avoided by using different programming languages, different compilers, and different computer hardware. For instance, this approach was applied to Gonthier's proof of the four-color theorem, see [7].

Professor Jonathan Hanke, a number theorist and skilled programmer at the Princeton University, is quite careful; he is focused on developing and implementing algorithms to solve concrete problems in programming language Python. To his opinion, software should never be trusted; it should be checked. Besides, in Hanke's opinion, the only way to avoid false results is to use computers in the proofs of theorems step by step very carefully, using special tests applied to separated sections (small or large, as needed) of a global program with unmistakable logic.

The science of program proving was a formally accepted field of computer science. Program proving, model checking, theorem solving – this is the terminology occupying the research space of computer science devoted to making sure programs work correctly. Computer programs analyze, check and inspect key situations and

outcomes by sophistical algorithms, and verify the validity of the theorem using the data collected passing through this process. David Bailey, a mathematician and computers scientist at Lawrence Berkeley National Laboratory (now at University of California, Davis), one of the world leaders in experimental mathematics, said: *"The time when someone can do real, publishable mathematics completely without the aid of a computer is coming to a close. Or if you do, you are going to be restricted into some specialized realms."*

Doron Zeilberger (1950–), a world-renowned Israeli professor at Rutgers University, the winner of prestigious awards such as Ford Award, Steele Prize, Euler Medal and Robbins Prize, does not share the mentioned view of his sceptical colleagues. He said: *"Contemporary mathematics is becoming significantly complicated, making further progress more difficult. In many mathematical disciplines computers are so much incorporated that only at the frontiers of some research areas of mathematics, human proofs still exist."*. Note that Zeilberger writes his own code using a computer algebra system *Maple* and believes computers are overtaking humans in their ability to discover new mathematics. See the link `www.wired.com/2013/03/computers=and=math/` .

Searching for ultimate truth in mathematics, the Field medallist Vladimir Voevodsky (1966–2017) (Institute for Advanced Studies in Princeton) posed the question: *"How do mathematicians know that something they prove is actually true?"* Similarly, as Zeilberger and Bailey, he comprehended that increasing the complexity of mathematics could be resolved only by the computer since a human brain could not keep up a huge amount of data and manipulate with them. To resolve this very hard problem, Voevodsky started, as the leader of a team, a long-term extraordinary project to create fundamentally new computer tools to confirm the accuracy of proofs. For this purpose, Voevodsky and his team have united different research fields, such as homotopy theory, mathematical logic, and the theory of programming languages, to make computer-verified proofs.

We end this section with an interesting story that tells how much Zeilberger believes in computer-assisted proofs and other computer tools for solving mathematical problems. Some thirty years ago several mysterious but excellent research papers (77 in total) appeared in a short period in the renowned mathematical journals (co)-authored by Shalosh B. Ekhad; in addition, notable Rutgers University (New Jersey) was marked as the affiliation. Curious mathematicians have tried to learn anything about the personality of Ekhad for three reasons; this name was fully unknown in the mathematical literature, nobody has ever seen him, and there was no Professor Ekhad employed at Rutgers University. The Israeli mathematician Doron Zeilberger from Rutgers University (the affiliation was correct) resolved the mystery admitting that Shalosh B. Ekhad is not a person but his computer. In Hebrew the words "Shalosh and "Ekhad" mean THREE and ONE respectively, and "three B one" refers to the AT&T 3B1, the first computer that he had been using in his work. Wishing to emphasize the great importance of computers to his research, Zeilberger cited Shalosh B. Ekhad as his co-author of scientific papers.

3. Computers in mathematical research - authors contributions

In this section we present some illustrative mathematical problems of dual nature; they belong to numerical mathematics but also to computer science (roundoff error analysis). An unexpected but interesting behavior of iteration procedure, arising as a consequence of the presence of roundoff errors, are discussed in Section 2.& 3.1 and & 3.2. Strange distribution of zeros of algebraic polynomials with random coefficients is demonstrated by two examples in & 3.3. In & 3.4 we present the dynamic study of root-finding methods by basins of attractions and point to useful benefits of visualization and associate data. A new three-point weighted family of iterative methods for approximating solutions of nonlinear equations is the subject of & 3.5. The derivation of the method and its convergence analysis are performed using symbolic computation. This study deals with very complicated and lengthly expressions (consists of 200 and more outcome lines) so that the construction and analysis of the proposed family is far beyond human capability. Two self-validated iterative methods for the inclusion of a simple zero of a given polynomial are presented and numerically tested in & 3.6.

3.1. Strange recurrent relation and roundoff errors

This example originally constructed in [22], inspired by Kahan's recurrent relations, presents in illustrative way the influence of roundoff error to the accuracy of result of computation. Let us calculate the members of the sequence $\{x_k\}$ in floating-point arithmetic of double or quadruple precision using the recurrent relation

$$\begin{cases} x_0 = 1, \\ x_1 = -5, \\ x_{k+1} = 207 - \dfrac{1412}{x_k} + \dfrac{2400}{x_{k-1}x_k}. \end{cases} \tag{3.1}$$

After a certain number of iterative steps, we observe that x_k approaches 200, see Figure 3.1. However, using methods for solving difference equations we find the general solution of the recurrent relation (3.1) in the form

$$x_k = \frac{200^{k+1}a + 4^{k+1}b + 3^{k+1}}{200^k a + 4^k b + 3^k},$$

where a and b are arbitrary constants. For the given initial values $x_0 = 1$ and $x_1 = -5$ one obtains $a = 0,\ b = -2/3$, so that the above formula reduces to the simple form

$$x_k = \frac{-\frac{2}{3}\cdot 4^{k+1} + 3^{k+1}}{-\frac{2}{3}\cdot 4^k + 3^k} = 4 - \frac{1}{1 - \frac{2}{3}\left(\frac{4}{3}\right)^k}. \tag{3.2}$$

From (3.2) it is clear that $x_k \to 4$ when $k \to \infty$.

Incorrect result ($x_k \to 200$) is the consequence of roundoff error during calculation. Namely, the application of floating-point arithmetic does not calculate the

3.2. Power method for dominant eigenvalue – the benefit of roundoff

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of an $n \times n$ matrix A. λ_1 is called the **dominant eigenvalue** or **spectral radius** of A if

$$|\lambda_1| > |\lambda_i| \quad (i = 2, \ldots, n).$$

The eigenvector corresponding to λ_1 is called **dominant eigenvector** of A. In the literature, the spectral radius is most frequently denoted by $\rho(A)$.

The **power method** is an iterative method which is often applied for approximating spectral radius of a given matrix A. Scaling version of the power method can be presented in the following algorithmic form:

$$(3.3)\quad \begin{cases} 1.\ \textbf{Choose} \text{ starting non-zero vector } \boldsymbol{y}_0 = \{y_{1,0}, \ldots, y_{n,0}\}; \\ 2.\ \textbf{For } k = 1, 2, \ldots \textbf{ calculate} \\ \qquad\qquad \boldsymbol{z}_k = A\,\boldsymbol{y}_{k-1}, \quad \boldsymbol{y}_k = \boldsymbol{z}_k/\alpha_k, \\ \qquad \text{where } \alpha_k \text{ is the coordinate of the vector } \boldsymbol{z}_k \text{ with the largest moduli.} \\ 3.\ \textbf{Finish} \text{ the iterative process when the stopping criterion is fulfilled.} \end{cases}$$

Note that

$$\alpha_k \to \lambda_1 \quad \text{and} \quad \boldsymbol{y}_k \to \frac{\boldsymbol{x}_1}{\| \boldsymbol{x}_1 \|_\infty},$$

where $\boldsymbol{x}_1$ is the dominant eigenvector that correspond to the dominant eigenvalue λ_1. The value α_k is taken to be the approximation of the spectral radius $\lambda_1 = \rho(A)$.

In practical problems, the presence of roundoff error can often cause inaccurate results. Opposite to the previous request, in the application of the power method roundoff errors can play a positive role, as mentioned by Higham [39]. Such a situation is demonstrated by the following example.

Example 3.1. Let us determine the approximative value of the spectral radius of the matrix

$$A = \begin{bmatrix} 0.5 & -0.8 & 0.3 \\ -0.6 & 0.8 & -0.2 \\ 0.24 & 0.67 & -0.91 \end{bmatrix}$$

using the presented power method with scaling. First of all, note that the power method (3.3) applied in single precision fails if we take $\boldsymbol{y}_1 = \{1, 1, 1\}$ since in the next step it produces the zero vector. Hence, there is no indication of the wanted dominant eigenvalue. However, executing the first step in double-precision arithmetic, we get

$$\boldsymbol{y}_1 = A \cdot \{1, 1, 1\} = \{5.55112 \times 10^{-17}, 5.55112 \times 10^{-17}, 0.\}\,.$$

The presence of roundoff errors produces $\boldsymbol{y}_1 \neq \boldsymbol{0}$.. Applying the power method (8), after 18 iterations we obtain $\alpha_{18} = 1.30818$ and take this value as an approximation of the spectral radius. The spectral radius of A with 15 correct decimal digits is $\rho(A) = 1.308114998551363$, which means that the approximation α_{18} has 5 significant decimal digits.

3.3. Distribution of zeros of random polynomials

Algebraic polynomials whose coefficients are random numbers are of great importance since they appear in various problems of physics, engineering and economics such as filtering theory, spectral analysis of random matrices, statistical communication, regression curves in statistics, characteristic equations of random matrices, the study of random difference equations, the analysis of capital and investment in mathematical economics. etc. For these reasons, a number of books and papers have been devoted to the study of random polynomials, see, e.g., [23] and [24]. Working in the area of *Experimental Mathematics*, we have used graphical methods to visualize an important theorem on distribution of zeros and pose a conjecture of the symmetry of complex zeros od random polynomials.

Example 3.2. Denote a sequence of independently identically distributed (real or complex) valued random variables with $\{c_k\}_{k=0}^{\infty}$. Let

$$F_n(z) = c_n z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0$$

be a random polynomial of degree n with the zeros $\zeta_1, \zeta_2, \ldots, \zeta_n$ of F_n. Furthermore, for a, b $(0 \leqslant a \leqslant b < \infty)$ introduce a probability measure on the complex plane $R_n(a,b) = N_n(\{z : a \leqslant |\zeta_i| \leqslant b\})$, where $N_n(\cdot)$ denotes the number of zeros that belong to the ring $\{z : a \leqslant |\zeta_i| \leqslant b\}$ in the complex plane. Then R_n/n defines the *empirical distribution* of zeros of F_n. If for any $\delta \in (0,1)$ define $a = 1-\delta$, $b = 1+\delta$, then δ is called *delta measure* in the empirical distribution.

The following theorem has been proved in [24]:

Theorem 3.1. *If and only if*

$$\mathbf{E}\, log(1 + |c_0|) < \infty,$$

then the sequence of the empirical distributions R_n/n converges to the delta measure at 1 almost surely, that is,

$$\frac{1}{n} R_n(1-\delta, 1+\delta) \xrightarrow{\mathbf{P}} 1, \quad n \to \infty$$

holds for any $\delta \in (0,1)$.

In the above theorem, $\mathbf{E}$ is mathematical expectation while the denotation $\xrightarrow{\mathbf{P}}$ denotes so-called **convergence in probability**. This theorem asserts that, under some weak constraints on the coefficients of a random polynomial, *almost all its zeros* "concentrate uniformly" close to the unit circle with high probability.

Using graphical tools of *Mathematica* we have tested a random polynomial of degree 2000 with random coefficients belonging to the interval $[-2, 2]$. From Figure 3.2 we observe that almost all zeros are located in the ring $\{z \,|\, 1-\delta < |z| < 1+\delta\}$ where $\delta \approx 0.01$, which empirically confirms Theorem 3.1 to a good extent.

using the program BWH in *Mathematica* and presented in Figure 3.3, is of "bagel-with-handle" (BWH) form.

BWH PROGRAM (*Mathematica*)

```
Clear[koef]; koef := Sign[-0.5 + Random[]]; Clear[genP];
genP[n_] := Sum[koef x^k, {k, 0, n}]; Clear[solP];
solP[p_] := Map[x /.  # &, Solve[p == 0, x] // Flatten ] // N;
Clear[graP];
graP[s_] := ListPlot[Map[{Re[#], Im[#]} &, s], Frame -> True,
AspectRatio -> 1, PlotRange -> {{-2, 2}, {-2, 2}},
PlotStyle -> {RGBColor[Random[], Random[], Random[]],
PointSize[0.01]}]; t = Table[genP[RandomInteger[10,18]],{50000}];
s = Map[solP, t]; g = Map[graP, s]; Show[g]
```

From Figure 3.3 we observe that the generated picture BWH is entirely symmetric to any straight line L_α passing through the origin, where $\alpha \in (0, \pi)$ is the angle related to the positive direction of abscissa axes. More precisely, any pair of the boundaries ∂A and ∂B of exterior parts A and B bounded by two lines L_α and L_β are of the same shape. The same is valid for two corresponding interior boundaries. To the authors' hypothesis, the presented symmetry arises following the low of large numbers, an important theorem in Probability theory. This theorem asserts that the average of the results of performing the same experiment a large number of times approaches the expected value, as the case in our experiments. This effect is known in the statistics when dealing with very large randomly chosen numbers with uniform distribution.

The second characteristic of our BWH figure is the existence of an empty space (hole) inside BWH. What is the size of this hole? More generally, what are the bounds of the zeros of the considered polynomials

$$P_{n,m}(z) = a_0^{(m)} z^n + a_1^{(m)} z^{n-1} + \cdots + a_{n-1}^{(m)} z + a_n^{(m)}, \tag{3.4}$$
$$a_k^{(m)} \in \{-1, +1\}, \quad n \in [10, 18], \quad m = 1, 2, \ldots, 50\,000\,?$$

Let $\zeta_{1,k}, \ldots, \zeta_{n,k}$ be the zero of the polynomial $P_{n,m}$. According to Henrici's result [25, p. 457], all zeros of $P_{n,m}$ are contained in the disk centered at the origin and with radius R determined as

$$\rho = 2 \max_{1 \leqslant j \leqslant n} \left| \frac{a_j^{(m)}}{a_0^{(m)}} \right|^{1/j}. \tag{3.5}$$

Note that this result holds for polynomials with arbitrary coefficients. According to (3.4) and (3.5) we find $|\zeta_{j,k}| \leqslant \rho = 2$. Substituting $y = 1/z$ in (3.4) and applying again (3.5), we determine the lower bound $|\zeta_{j,k}| \geqslant \frac{1}{2}$. Therefore,

$$\frac{1}{2} \leqslant |\zeta_{j,k}| \leqslant 2 \quad \text{for any } j \in \{1, \ldots, n\} \ \vee \ k \in \{1, \ldots, 50\,000\}.$$

According to the last inequalities, we conclude that all zeros of all 50 000 polynomials lie in the disk $\{0; 2\}$ and there is "no zero" in the hole containing the disk $\{0; 0.5\}$.

Finally, if we deal with the polynomials of relatively low degree (as in the presented example), we can observe the holes on the real axis at the points -1 and 1, see Figure 3.3. We also see that there is no complex zeros in these holes. This effect was noted in the book [20] but without discussion and explanation.

3.4. Dynamic study of root-finding methods

One of the most challenging tasks in the area of iterative methods for solving nonlinear equations is to detect the best algorithm or at least the group of best algorithms. For a long time, the comparative studies of root-finding algorithms were based on comparisons of (i) the number of iterations needed to provide the required accuracy of produced approximations to the solutions, (ii) the convergence rate, (iii) the number of function evaluations per iteration, and (iv) the computational costs of compared algorithms often measured by the consumed CPU time required to fulfill the given stopping criterion. All of the mentioned criteria suffer from the disadvantage consisting of the request for ideal conditions; namely, they are usable only if the chosen initial approximation to the wanted zero of a given function is sufficiently good to provide the convergence, which is difficult to achieve in practice. Even in those cases when it is possible, the rank of compared methods is not reliable since the convergence behavior of root-finding methods depends in a complicated and unpredictable way on the starting points.

The growing development of computer hardware and computer graphics at the end of the twentieth century has provided the significant advance of a new methodology for the visual study of convergence behavior of root-finding methods. It turned out that a realistic quality study of root-finding methods and their reliable ranking can be successfully accomplished by plotting the basins of attraction for the methods. Basins of attractions are the sets of points in the complex plane which simulate the convergence to the zeros of a given function by applying the iterative process. They are of great benefit since offer essential information and insight into the basic features of a considered iterative method such as its convergence behavior and domain of convergence. Also, we can apply basins of attraction to analyze the computational advantages of one iteration function against another and to rank root-solvers within a class of iteration functions, which is of interest for the user to decide which iteration method is preferable for solving a concrete problem.

Definition 3.1. *Let f be a given sufficiently many times differentiable function in some complex domain $R \subseteq \mathbb{C}$ with simple or multiple zeros $\alpha_1, \alpha_2, \ldots, \alpha_\lambda \in S$, and a (convergent) root-finding iteration defined by*

$$z_{k+1} = g(z_k) \quad (k = 0, 1, 2, \ldots),$$

the basin of attraction for the zero α_i is defined as follows:

$$\mathcal{B}_{f,g}(\alpha_i) = \{\zeta \in R \,|\, \text{the iteration } z_{k+1} = g(z_k) \text{ with } z_0 = \zeta \text{ converges to } \alpha_i\}.$$

The dynamic study for the comparison of root-finding algorithms for simple zeros is based on basins of attraction for a given method and a given example. It was launched by Stewart [26] and Varona [27] and continued in the works of Amat et al. [28]–[30], Scott et al. [31], Chun and Neta [32], [33], Neta et al. [34], Argyros and Magreñan [35], Kalantari [36], I. Petković and Neta [37], I. Petković and Đ. Herceg [38] and others.

In the case of algebraic polynomials, the basin of attraction for a given rectangle R with sides parallel to coordinate axes is plotted in the following way. Let (a_1, b_1) be the lower left vertex and (a_2, b_2) the upper right vertex (a_2, b_2) of this rectangle. Using computer algebra system *Mathematica* by the statement

```
CountRoots[P[z],{z,a1+I*b1,a2+I*b2}]
```

we determine the number of zeros of $P(z)$ inside the rectangle R. Analyzing convergence behavior for all zeros of a polynomial of degree n, the rectangle R must be taken so that the outcome N_P of the above statement is n (= number of polynomial zeros). Otherwise, we continue with the enlargement of the size of the rectangle R until $N_P = n$ is satisfied.

The considered method is tested on the $m_1 \times m_2$ equally spaced points in the rectangle $R = \{a_1, b_1\} \times \{a_2, b_2\}$ (forming an equidistant lattice L_R) centered at the origin. At the beginning we define the limit number of iterations IT; if the iterative process, starting from an initial point $z_0 \in L_R$, does not satisfy the given stopping criterion in $\leqslant IT$ iterations, then this starting point is proclaimed "divergent." For each basin we record the CPU time in seconds for all $m_1 \times m_2$ points, average number of iterations (for all points of the lattice L_R) required to satisfy the stopping criterion $|z_k - \alpha| < \tau$ (τ defines the accuracy of approximations, say, $\tau = 10^{-5}$ or $\tau = 10^{-6}$) and the number of black (divergent) points for each method and each example. We associate exactly one color to each attraction basin of a root following two rules: 1) each basin will have a different color and 2) the shading is darker if the number of iterations is higher. Starting points which do not fulfill the stopping criterion after IT iterations are colored black.

The basin of attraction is a kind of computer visualization that provides visual insight into convergence behavior of a root-finding method but it also delivers some valuable qualitative data such as the CPU execution time, the average number of iterations and function evaluations per point, and the number of "divergent" points. These data are most frequently sufficient for deeper insight into the behavior of an iterative method and its domain of convergence from the point of view of dynamical systems. Obviously, a method is better if the consumed CPU time, the average number of iterations and function evaluations per point, and the number of "divergent" points are smaller. It is desirable that the number of divergent point is 0, which points to global convergence of the method.

Convergence behavior of any method can also be estimated to a certain extent according to the shape of basins of attraction for the tested example. It is preferable

that the basins of attraction for the zeros have as large as possible unvaried contiguous areas, separated by the boundaries that have (approximately) straight-line form. As small as possible blobs and fractals on the boundaries also point to good convergence properties.

To demonstrate the dynamic study of two iterative methods for finding simple zeros, we give three examples. In all examples we have used an equidistant lattice made of 360 000 points, that is, the resolution is 600×600, the permitted number of iterations is $IT = 40$, and the stopping criterion has been given by $|z_k - \alpha| < 10^{-6} = \tau$.

We emphasize that the dynamic study by basins of attraction is most frequently used in comparative study of different methods of the same order of convergence. Since the main goal of this section is only the presentation of a graphical method for the analysis of the quality of particular methods, any comparative study is beyond our consideration.

We have considered the well-known Halley's method of the third order

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \cdot \frac{1}{1 - \dfrac{f''(x_k) f(x_k)}{2 f'(x_k)^2}} \quad (k = 0, 1, 2, \ldots), \tag{3.6}$$

and the three-point-method of order eight

$$\begin{cases} y_k = x_k - \dfrac{f(x_k)}{f'(x_k)}, \\ z_k = y_k - \dfrac{1}{1 - \frac{2f(y_k)}{f(x_k)}} \cdot \dfrac{f(y_k)}{f'(x_k)}, & (k = 0, 1, 2, \ldots), \\ x_{k+1} = z_k - \dfrac{f[z_k, y_k]}{f[z_k, x_k]} \cdot \dfrac{f(z_k)}{2f[z_k, y_k] - f[z_k, x_k]} \end{cases} \tag{3.7}$$

proposed by Sharma and Arora in [40]. It is not difficult to show (see [41]) that this method is a special case of the family of three-point methods constructed in [42]. An extensive investigation presented in [43] and [41] shown that the method (3.7) possesses the best convergence characteristics among three-point methods of the (maximal) order eight in the class of algebraic polynomials.

Example 3.4. We have plotted two basins of attraction applying the methods (3.6) and (3.7) to the polynomial

$$P_1(z) = z^5 - 1$$

and the square $R = \{z = x + iy \mid -3 \leqslant x \leqslant 3, -3 \leqslant y \leqslant 3\}$. The basins are given in Figures 3.4 and Figure 3.5.

FIG. 3.6: The basins of attraction for the method (3.7) applied to $P_2(z)$.

The basin of attraction for all 15 zeros is presented in Figure 3.6. Small circles mark the location of zeros of the polynomial $P_2(z)$. Considering all 360 000 points we have recorded:

- 0 divergent points,
- the average number of iterations = 6.44,
- the CPU time = 298.8 sec.

The fact that the number of divergent points is 0 points to the global convergence of the method (3.7). However, the boundaries of particular basins are not straight lines but strips (corresponding to some other zeros). This undesirable phenomenon is typical for random polynomials of a high degree, which can lead to certain problems when choosing initial approximations.

Example 3.6. The three-point method (3.7) has been applied to the polynomial

$$P_3(z) = \prod_{m=1}^{13} (z - m).$$

of Wilkinson's type. It is well-known that polynomials of this form are ill-conditioned, causing that many root-finding methods work with big efforts in solving this class of polynomials. However, the applied method (3.7) showed very good convergence behavior, which is evident from the basins of attraction presented in Figure 3.7: particular basins have large unvaried contiguous areas with regular boundaries free of fractal parts and with very small blobs. The associated data are given below:

- 0 divergent points (excellent outcome),
- the average iteration = 4.69,
- the CPU time = 422.7 sec.

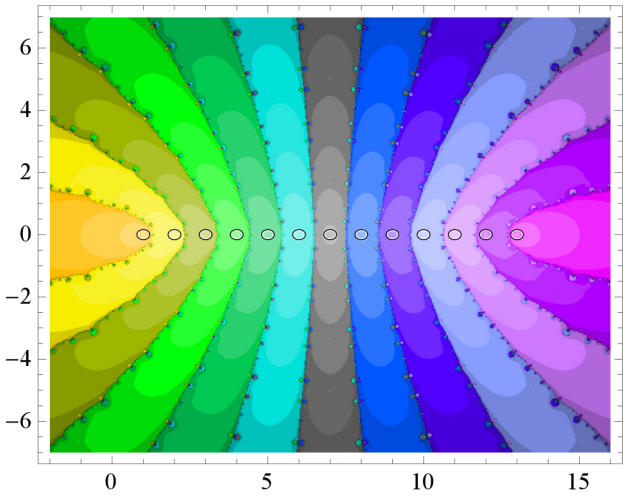

FIG. 3.7: The basins of attraction for the method (3.7) applied to $P_3(z)$.

3.5. On a new three-point weighted method for simple zeros

We start from three-point iterative scheme

$$\begin{cases} N(x_k) = \dfrac{f(x_k)}{f'(x_k)}, \\ y_k = x_k - N(x_k), \\ z_k = y_k - u_k H(u_k) N(x_k), \\ x_{k+1} = z_k - w_k(2w_k + 1)P(u_k)Q(v_k)N(x_k), \end{cases} \tag{3.8}$$

where

$$u_k = \frac{f(y_k)}{f(x_k)}, \quad v_k = \frac{f(z_k)}{f(y_k)}, \quad w_k = u_k v_k.$$

We omit the iteration index k and define the errors

$$\varepsilon = x - \alpha, \quad \varepsilon_y = y - \alpha, \quad \varepsilon_z = z - \alpha, \quad \hat{\varepsilon} = \hat{x} - \alpha,$$

where $\hat{x}$ is a new approximation x_{k+1}. Introduce

$$c_r = \frac{f^{(r)}(\alpha)}{r! f'(\alpha)} \quad (r = 1, 2, \ldots).$$

We will use the following development of the function f about the zero α

$$f(x) = f'(\alpha)\Big(1 + c_1\varepsilon + c_2\varepsilon^2 + c_3\varepsilon^3 + c_4\varepsilon^4 + c_5\varepsilon^5 + c_6\varepsilon^6 + c_7\varepsilon^7 + c_8\varepsilon^8 + O(\varepsilon^9)\Big),$$

and a program in *Mathematica*. As usual, in finding the weight functions H, P and Q, we represent these functions by their Taylor's series at the neighborhood of $u = 0$ (for H and P), and $v = 0$ (for Q):

$$\begin{aligned} H(u) &= H(0) + H'(0)u + \frac{H''(0)}{2}u^2 + \frac{H'''(0)}{6}u^3 + \cdots, \\ P(u) &= P(0) + P'(0)u + \frac{P''(0)}{2}u^2 + \frac{P'''(0)}{6}u^3 + \cdots, \\ Q(v) &= Q(0) + Q'(0)v + \frac{Q''(0)}{2}v^2 + \frac{Q'''(0)}{6}v^3 + \cdots. \end{aligned}$$

The coefficients of Taylor's developments of the weight functions P and Q are determined using an interactive approach by combining the program realized in *Mathematica* (two parts) and the annihilation of coefficients standing at ε of lower degree. For simplicity, we write $H_0 = H(0)$, $H_1 = H'(0)$, $Q_3 = Q'''(0)$, etc, and

`fa` $= f'(\alpha)$, `fx` $= f(x)$, `fy` $= f(y)$, `fz` $= f(z)$, `fx1` $= f'(x)$, `newt` $= f(x)/f'(x)$, `e` $= \varepsilon$, `ey` $= \varepsilon_y$, `ez` $= \varepsilon_z$, `e1` $= \hat{\varepsilon}$.

PART I (*Mathematica*)

```
fxx = 1+c1*e+c2*e^2+c3*e^3+c4*e^4 +c5*e^5+c6*e^6+c7*e^7+ c8*e^8;
fx = fa*e*fxx; fx1 = D[fx, e]; newt = Series[fx/fx1,{e, 0, 8}];
ey = e - newt; fy = fa*ey (1+1*ey+2*ey^2+c3*ey^3+c4*ey^4);
u = fy*Series[1/fx, {e, 0, 8}];
H = H0+H1*u+H2/2*u^2+H3/6*u^3;
ez = Series[ey - u*newt*H // FullSimplify, {e, 0, 8}]
```

This program gives

$$e_z = (c_1 - c_1H_0)e^2 + (-2c_2(-1 + H_0) + c_1^2(-2 + 4H_0 - H_1))e^3 + O(e^4)$$

To annihilate coefficients by e^2 and e^3, it is necessary and sufficient to take

$$H_0 = 1, \quad H_1 = 2, \quad H_2 \text{ and } H_3 \text{ arbitrary},$$

which gives

$$e_z = (-c_1c_2 + c_1^3(5 - H_2/2))e^4 + O(e^5).$$

The part II of the program uses previously found entries and serves for finding additional conditions which provide optimal order eight.

PART II - CONTINUATION (*Mathematica*)

```
fz = fa*ez*(1+c1*ez+c2*ez^2); v = fz*Series[1/fy,{e, 0, 8}];
P = P0+P1*u+P2/2*u^2+P3/6*u^3;
Q = Q0+Q1*v;
e1 = Series[ez-u*v*P*G*(2u*v+1)*newt,{e,0,8}]//FullSimplify
```

The error $\hat{\varepsilon} = \hat{x} - \alpha$ $(= e1)$ is given in the form

$$\varepsilon_1 = \sum_{r=4}^{8} T_r \varepsilon^r + O\big(\varepsilon^9\big)$$

From the conditions $T_4 = 0,\ T_5 = 0,\ T_6 = 0,\ T_7 = 0$, we find the following relations for finding the required coefficients:

$$H(0) = 1, \quad H'(0) = 2,$$
$$P'(0) = 2P(0), \quad P''(0) = P(0)(2 + H''(0)), \quad P'''(0) = P(0)(H''(0) + 6H''(0) - 24),$$
$$Q(0) = Q'(0) = \frac{1}{P(0)}.$$

A natural choice $P(0) = 1$ gives

$$\begin{aligned} &H(0) = 1, \quad H(0) = 2, \\ &P(0) = 1, \quad P'(0) = 2, \quad P''(0) = 2 + H''(0), \quad p'''(0) = H'''(0) + 6H'(0) - 24, \\ &Q(0) = Q'(0) = 1. \end{aligned} \tag{3.9}$$

In this way we have proved the following assertion.

Theorem 3.2. *If the initial approximation x_0 is sufficiently close to the zero α of f and the conditions* (3.9) *are valid, then the order of the three-point family* (3.8) *is eight.*

Kung-Traub hypotheses [44] assert that as high as possible order of convergence of the n-point method that uses $n+1$ function evaluations per iteration is 2^n. Such methods are called *optimal methods*. Therefore, according to this hypothesis and Theorem 3.2, the three-point iterative method (3.8) is optimal.

3.6. Iterative method for the inclusion of a simple complex zero

R. E. Moore, the founder of Interval analysis, introduced in his monograph [45] the interval version of Newton's method, often called Moore-Newton's method. Let f be a differentiable function on a real interval Ω and let $X_0 = [\underline{x}_0, \overline{x}_0] \subset \Omega$ be a real interval containing a simple real zero η of f. An interval extension $F'(X)$

over the interval X is a real interval such that $F'(X) \supseteq \bar{f}(X) = \{x \,|\, x \in X\}$. Moore-Newton's method is defined by

$$X_{k+1} = \Big\{ m(X_k) - \frac{m(X_k)}{F'(X_k)} \Big\} \cap X_k \quad (k = 0, 1, \ldots), \tag{3.10}$$

where $m(X_k) = \frac{1}{2}\big([\underline{x}_k, \overline{x}_k]\big)$ is the midpoint of the interval X_k. It is obvious that this method will be defined if $0 \notin F'(X_k)$ in every iteration.

Moore-Newton's method (3.10) can be applied only for enclosing real zeros, which is a serious disadvantage. Here we present two simple algorithms for finding a simple complex zero ζ of a given algebraic polynomial P that produces a disk $\{c; r\} := \{z \,|\, |z - c| \leqslant r\}$ in the complex plane such that $|c - \zeta| < r$. In this way, these methods provide the upper error bound (given by the radius r) of the approximation c to the desired complex zero ζ. Recall that the inversion of a disk $\{c; r\}$ not containing 0 (that is, $|c| > r$ holds) is defined in [46] by

$$\{c; r\}^{-1} = \Big\{ \frac{\bar{c}}{|c|^2 - r^2}; \frac{r}{|c|^2 - r^2} \Big\}.$$

Finding initial approximation to the sought zero of a function, sufficiently close to this zero to provide guaranteed convergence, is an equally important task as the construction of an efficient iterative method. This topic is beyond the main subject of this paper and it will not be considered here. Instead, we cite the paper [47] and the master thesis [48] where a composed search-subdividing algorithm for the localization of all complex zeros of algebraic polynomials has been presented with the help of CAS *Mathematica*. This algorithm produces arbitrary small inclusion squares, each of which contains one and only one zero, and calculate the multiplicity of these zeros. It can be of benefit for iterative methods implemented in ordinary complex arithmetic and complex interval arithmetic, discussed in what follows.

Algorithm 1. Let $Z_0 = \{a; R\} = \{z_0; \rho\}$ be the disk that contains one and only one zero ζ of a polynomial P of degree n. The following iterative method was proposed in [49]:

$$Z_{k+1} = z_k - \frac{1}{\{c_k; \rho_k\}} = \left\{ z_k - \frac{\bar{c}_k}{|c_k|^2 - \rho_k^2}; \frac{\rho_k}{|c_k|^2 - \rho_k^2} \right\} \quad (k = 0, 1, \ldots), \tag{3.11}$$

$$c_k = \frac{P'(z_k)}{P(z_k)} - \frac{(n-1)(\bar{z_k} - \bar{a})}{R^2 - |z_k - a|^2}, \quad \rho_k = \frac{(n-1)R}{R^2 - |z_k - a|^2}, \quad (k > 0). \tag{3.12}$$

The stopping criterion was given by $|P(c_k)| < \tau$, where τ is, say, 10^{-16} or 10^{-33}.

Considering the formulas (3.11) and (3.12) we observe two drawbacks of Algorithm 1. To avoid the division by a zero-interval in (3.11) (which produces a disk of infinity large radius) and negative radius (formula (3.12)), it is necessary to satisfy two conditions in each iteration

$$\text{(i)} \;\; |c_k| > \rho_k, \quad \text{(ii)} \;\; R > |z_k - a|.$$

Regarding (i) we conclude that c_k must be reasonably large and hence, $|P(z_k)|$ should be rather small. Therefore, z_k should be a very good approximation to the zero ζ. Most frequently this is not the case at the beginning of any iterative process so that the first iterations are very critical. To resolve this inconvenient situation the only way is to choose the center a of the initial inclusion disk $\{a; R\}$ very close to the sought zero ζ, which is rather strong requirement (the first drawback). From this discussion there follows $z_k \approx a$ so that $\rho_k \approx (n-1)/R$. The choice of small R increases ρ_k (see (3.12)) so that the validity of inequality (i) may be endangered. Therefore, contrary to the usual request for as small as possible radius of initial inclusion disk, in the case of Algorithm 1 the radius R should be relatively large. Consequently, in this way the inequality (ii) will be ensured. On the other hand, a large R can lead to an undesired enclosure of other zeros of P (next to the zero ζ). It follows that the choice of R has to be refined, sometimes by trial and error method (the second drawback).

Example 3.7. Using Algorithm 1, determine sufficiently small disk that contains the zero $\zeta = 2i$ of the polynomial

$$P(z) = z^9 + 3zt^8 - 3z^7 - 9z^6 + 3z^5 + 9z^4 + 99z^3 + 297z^2 - 100z - 300,$$

starting from the inclusion disk $Z_0 = \{0.1 + 2.1i; 1.7\}$ and setting $\tau = 10^{-33}$. The locations of all zeros of P and initial disks Z_0 (containing the sought zero $\zeta = 2i$) are displayed in Figure 3.8.

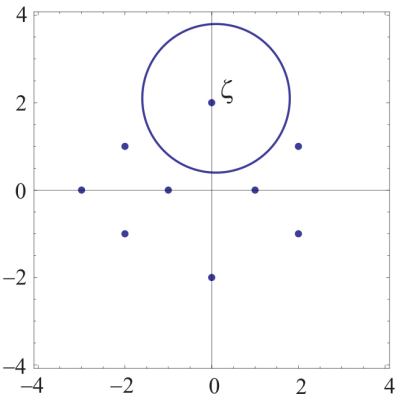

FIG. 3.8: The locations of all zeros of P and initial disks Z_0

We have used CAS *Mathematica* and multi-precision arithmetic (40 significant decimal digits). The following inclusion disks have been obtained:

$$Z_1 = \{0.00473 + 1.97173\,i\,;\, 0.0856\}$$
$$Z_2 = \{-0.00128 + 2.00249\,i\,;\, 0.00456\ldots\}$$
$$Z_3 = \{-1.7 \times 10^{-5} + 2.00000697\,i\,;\, 3.70 \times 10^{-5}\}$$

$$Z_4 = \{-4.33 \times 10^{-10} + 1.99999999931\,i\,;\ 1.6 \times 10^{-9}\}$$
$$Z_5 = \{1.25 \times 10^{-18} + 2.00000000000000000096\,i\,;\ 3.11 \times 10^{-18}\}$$
$$Z_6 = \{5.95 \times 10^{-36} + 2.\underbrace{0000000000\cdots0000000000}_{\text{thirty six 0}}74\,i\,;\ 1.18 \times 10^{-35}\}$$

Algorithm 2. We present a combined method for approximate a simple zero of a given polynomial. This method possesses a low computation cost since it uses Newton's method in ordinary complex arithmetic in all iterations except the last one, where a very simple procedure is applied to provide the upper error bound which is involved in the following theorem due to Laguerre (see, e.g., [25, pp. 466–468]):

Theorem 3.3. *Let z be an arbitrary complex number and let P be a given algebraic polynomial. Then the disk $D = \{z; n|P(z)/P'(z)|\}$ contains at least one zero of P.*

The disk D is usually called *Laguerre's disk*. As in the case of Algorithm 1, Algorithm 2 also requires sufficiently good initial approximation z_0 to the zero.

1° step: *Starting from z_0, apply Newton's iteration*

$$z_{k+1} = z_k - \frac{P(z_k)}{P'(z_k)}$$

for $k = 1, 2, ..., K$, where K is the iteration index of the approximation z_K that fulfils the stopping criterion given in the form $|P(z_K)| < \tau$.

2° step: *We use the last approximation z_K obtained in the first step and, using Laguerre's disk defined in Theorem* 3.3*, calculate the inclusion disk*

$$Z_k = \Big\{z_7; n\Big|\frac{P(z_7)}{P'(z_7)}\Big|\Big\}.$$

The upper error bound is determined by the radius $r = n|P(z_K)/P'(z_K)|$.

Example 3.8. Using Algorithm 2, determine the inclusion disk for the zero $\zeta = 2i$ of the polynomial P given in Example 3.7. In contrast to Algorithm 1, the initial approximation z_0 need not to be very close to ζ and we have chosen $z_0 = 0.2 + 2.3i$. As in Example 3.7, we have used CAS *Mathematica*, multi-precision arithmetic (40 significant decimal digits) and $\tau = 10^{-33}$. First, we have applied Newton's method until the fulfilment of the stopping criterion $|P(z_K)| < 10^{-33}$ and obtained

$$z_1 = 0.11848 + 2.10232\,i$$
$$z_2 = 0.03978 + 2.00577\,i$$
$$z_3 = 0.00151 + 1.99709\,i$$
$$z_4 = -1.97 \times 10^{-5} + 2.0000106\,i$$

$$z_5 = -9.04 \times 10^{-10} + 1.99999999933\,i$$
$$z_6 = 2.26 \times 10^{-18} + 1.9999999999999999983\,i$$
$$z_7 = -4.14 \times 10^{-37} + 1.\underbrace{99999999999999999999999999999999999}_{\text{thirty four 9}}89\,i.$$

Since $|P(z_7)| < 10^{-33}$ we have stopped Newton's method and calculated the radius

$$r = n\Big|\frac{P(z_7)}{P'(z_7)}\Big| = 9.62 \times 10^{-35}.$$

Hence, the inclusion disk containing the zero $\zeta = 2i$ is given by

$$Z_7 = \{-4.14\times10^{-37}+1.\underbrace{99999999999999999999999999999999999}_{\text{thirty four 9}}89\,i\,;\ 9.62\times10^{-35}\}.$$

Considering the results of Examples 3.7 and 3.8, we observe that the upper error bounds are very small and of the same order. Algorithm 1 finished the iterative process through 6 iterations, while Algorithm 2 requested one more. However, the computational cost of Algorithm 2 is considerably less than the cost of Algorithm 1.

Acknowledgement. The authors thank the reviewer for helpful suggestions and comments that improve the presentation of the paper. The authors were supported in part by the Serbian Ministry of Education and Science under Grants 174022 and 174030.

REFERENCES

1. P. Mancosu: *The Philosophy of Mathematical Practice*, Oxford University Press, 2008.
2. T. C Hales: *Cannonballs and honeycombs*. Notices Amer. Math. Soc. **47** (April 2000), 440–449.
3. M. S. Petković: *Famous Puzzles of Great Mathematicians*. American Mathematical Society, 2008.
4. H. C. Williams, R. A. German and C. R. Zarnke: *Solution of the cattle problem of Archimedes*. Math. Comp. **10** (1965), 671–674.
5. K. Appel and W. Haken: *Every planar map is four colorable, Part I, Discharging*. Illinois J. Math. **21** (1977), 429–490.
6. K. Appel and W. Haken, *Every planar map is four colorable, Part I, Reducibility*. Illinois J. Math. **21** (1977), 491–567.
7. G. Gonthier: *Formal proof – The Four-Color Theorem*. Notices Amer. Math. Soc. **55** (Dec. 2008), 1382–1393.
8. T. C. Hales: *A proof of the Kepler conjecture, (unabridged)*. Discrete Comput. Geom. **36** (2006), 1–265.

9. B. M. Brown, D. K. R. McCormack and A. Zettl: *On the existence of an eigenvalue below the essential spectrum*. Proc. R. Soc. Lond. **455** (1999), 2229–2234.
10. J.-P. Eckmann and P. Wittwer: *Computer Methods and Borel Summability Applied Feigenbaum's Equation*. Lecture Notes in Physics, Vol. 227, Springer Verlag, Berlin-Heidelberg-New York-Tokyo, 1985.
11. J. Hass, M. Hutchings and R. Schlafy: *The Double Bubble Conjecture*. Elec. Res. Announcement of the Am. Math. Soc. **1** (1995), 98–102.
12. K. Mischaikow and M. Mrozek: *Chaos in the Lorenz equations: A computer assisted proof. Part II: Details*. Math. Comput. **67** (1998), 1023–1046.
13. A. Neumaier and Th. Rage: *Rigorous chaos verification in discrete dynamical systems*. Physica D, **67** (1993), 327–346.
14. W. Tucker: *A rigorous ODE solver and Smale's 14th problem*. Found. Comput. Math. **2** (2002), 53–117.
15. D. Bailey, P. Borwein and S. Plouffe: *On the rapid computation of various polylogarithmic constants*. Math. Comp. **66** (1997), 903–913.
16. J. Borwein and D. Bailey: *Mathematics by Experiment*. A K Peters, Wellesley, Massachusetts, 2008.
17. S.M. Rump: *INTLAB - INTerval LABoratory*. In: Proceeding of the Conference on Developement in Reliable Computing (Scendes, T., ed.), Kluwer Academic Publishers, 1999, pp. 77–104. `http://www.ti3.tu-harburg.de/rump/intlab/index.html`.
18. U. Kulisch: *Computer Arithmetic and Validity*. Walter de Gruyter, Berlin-New York, 2008.
19. J. Borwein, D. Bailey and R. Girgensohn: *Experimentation in Mathematics*. A K Peters, Natick, Massachusetts, 2004.
20. D. Bailey, J. M. Borwein, N. J. Calcin, R. Girgensohn, D. R. Luke and V. H. Moll: *Experimental Mathematics in Action*. A K Peters, Wellesley, Massachusetts, 2007.
21. J. Dongarra and F. Sullivan: *The top 10 algorithms*. Computing in Science and Engineering, Jan./Feb (2000), 22–23.
22. I. Petković: *Analysis of Processing and Computing Iterations by Applying Modern Computer Arithmetics*. Ph. D. Thesis, Faculty of Electronic Engineering, University of Niš, Niš, 2012.
23. A.T. Bharucha-Reid and M. Sambandham, *Random Polynomials*. Academic Press, 1986.
24. I. Ibragimov and D. Zaporozhets: *On distribution of zeros of random polynomials in complex plane*. arXiv:1102.3517v1 [math.PR] 17 Feb 2011.]
25. P. Henrici: *Applied and Computational Complex Analysis, Vol. I. John Wiley and Sons*. New York, 1974.
26. B. D. Stewart: *Attractor Basins of Various Root-finding Methods*. Master Thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA, June 2001.
27. J. L. Varona: *Graphic and numerical comparison between iterative methods*. Math. Intelligencer **24** (2002), 37–46.
28. S. Amat, S. Busquier and S. Plaza, *Review of some iterative root-finding methods from a dynamical point of view*. Scientia **10** (2004), 3–35.

29. S. Amat, S. Busquier and S. Plaza: *Dynamics of a family of third-order iterative methods that do not require using second derivatives.* Appl. Math. Comput. **154** (2004), 735–746.

30. S. Amat, S. Busquier and S. Plaza, *Dynamics of the King and Jarratt iterations.* Aequ. Math. **69** (2005), 212–223.

31. M. Scott, B. Neta and C. Chun: *Basin attractors for various methods.* Appl. Math. Comput. **218** (2011), 2584–2599.

32. C. Chun and B. Neta: *Basins of attraction for Zhou-Chen-Song fourth order family of methods for multiple roots.* Math. Comput. Simul. **109** (2015), 74–91.

33. C. Chun and B. Neta: *Basins of attraction for several third order methods to find multiple roots of nonlinear equations.* Appl. Math. Comput. **268** (2015), 129–137.

34. B. Neta, M. Scott and C. Chun: *Basin of attractions for several methods to find simple roots of nonlinear equations.* Appl. Math. Comput. **218** (2012), 10548–10556.

35. I. K. Argyros and Á. A. Magreñán: *n the convergence of an optimal fourth-order family of methods and its dynamics.* Appl. Math. Comput. **252** (2015), 336–346.

36. B. Kalantari: *Polynomial Root-Finding and Polynomiography.* World Scientific, New Jersey, 2009.

37. I. Petković and B. Neta: *On an application of symbolic computation and computer graphics to root-finders: The case of multiple roots of unknown multiplicity.* J. Comput. Appl. Math. **308** (2016), 215–230.

38. I. Petković and Đ. Herceg: *Symbolic computation and computer graphics as tools for developing and studying new root-finding methods.* Appl. Math. Comput. **295** (2017), 95–113.

39. N. Higham: *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelpha, 2002.

40. J.R. Sharma and H. Arora: *A new family of optimal eighth order methods with dynamics for nonlinear equations.* Appl. Math. Comput. **273** (2016), 924–933.

41. I. Petković and Đ. Herceg: *Computers in mathematical research: the study of three-point root-finding methods.* Numer. Algor. doi.org/10.1007/s11075-019-00796-6 (online version).

42. J. Džunić, M.S. Petković and L.D. Petković: *A family of optimal three-point methods for solving nonlinear equations using two parametric functions.* Appl. Math. Comput. **217** (2011), 7612–7619.

43. C. Chun and B. Neta: *Comparative study of eighth-order methods for finding simple roots of nonlinear equations.* Numer. Algor. **74** (2017), 1169–1201.

44. H.T. Kung, J.F. Traub: *Optimal order of one-point and multipoint iteration.* Journal of the ACM **21** (1974), 643–651.

45. R.E. Moore: *Interval Analysis.* Prentice Hall, Englewood Cliff, New Jersey, 1966.

46. I. Gargantini, P. Henrici: *Circular arithmetic and the determination of polynomial zeros.* Numer. Math. **18** (1972), 305–320.

47. Đ. Herceg: *An algorithm for localization of polynomial zeros.* In: Proc. VIII International Conf. on Logic and Computer Science (R. Tošić, Z. Budimac, eds.), Institute of Mathematics, University of Novi Sad, Novi Sad, 1997, pp. 67–75.

48. Đ. Herceg: *Computer Implementation and Analysis of Iterative Methods for Solving Nonlinear Equations* (in Serbian). Master Thesis, Faculty of Science, University of Novi Sad, Novi Sad, 1997.

49. M.S. Petković: *Some interval iterations for finding a zero of a polynomial with error bounds*. Comput. Math. Appls. **14** (1987), 479–495.