ISSN 1063-7729, Astronomy Reports, 2020, Vol. 64, No. 12, pp. 1060–1066. © Pleiades Publishing, Ltd., 2020.
Russian Text © The Author(s), 2020, published in Astronomicheskii Zhurnal, 2020, Vol. 97, No. 12, pp. 1022–1029.

Solution of Kepler's Equation with Machine Precision

M. K. Abubekerov[a], * and N. Yu. Gostev[a], **

[a] *Sternberg Astronomical Institute, Moscow State University, Moscow, 119991 Russia*

**e-mail: marat@sai.msu.ru*

***e-mail: ngostev@mail.ru*

Received May 31, 2020; revised July 8, 2020; accepted August 15, 2020

Abstract—An algorithm for the numerical solution of Kepler's equation with machine precision is presented. The convergence of the iterative sequence of Newton's method is proved for the indicated initial approximation. The problem of finding a numerical solution to Kepler's equation as a floating point number is formulated. Aspects related to computations near machine zero are taken into account. We analyzed the accuracy of the possible result. A problem is identified that arises when tending for the highest possible accuracy and a solution is proposed. An estimate is given of the computer time required to solve Kepler's equation by this method.

DOI: 10.1134/S106377292012001X

1. INTRODUCTION

As a rule, in problems related to the motion of stars (as well as other cosmic bodies) in elliptical orbits, including the problem of interpreting the light curves of binary systems (see, for example, [1–4]), it becomes necessary to solve Kepler's equation:

$$x - e\sin x = \mathcal{M} \qquad (1)$$

relative to x at given values of e (eccentricity) and $\mathcal{M}$ (mean anomaly).

This equation is a transcendental equation, which has a unique solution for $0 < e < 1$. Various methods for solving this equation are currently known. For example, the solution is often used by the method of expansion in powers of e [5]. This method is mainly useful in the analytical study of motion in an eccentric orbit, assuming small e not exceeding the Laplace limit 0.6627.... However, the accuracy of the numerical results obtained by the power series method deteriorates with increasing values of e. At the same time, in many problems, it is desirable to obtain a solution to Kepler's equation with the maximum accuracy possible, based on the computer format used for representing real numbers. Although modern computers can operate on numbers whose accuracy is much higher than the accuracy of modern observations (for example, the accuracy of 80-bit extended-precision numbers with a 64-bit mantissa corresponds to 19 decimal signs), we assume that obtaining the result with the highest possible accuracy is still relevant for the following reasons.

First, sometimes, to increase the computation speed, it is advisable to use a machine representation of less bit depth than the maximum possible, since the same operations with numbers of less bit capacity are performed faster. For example, to speed up calculations, it sometimes makes sense to use not 80-bit extended-precision numbers, but 64-bit double-precision numbers (with a 52-bit mantissa corresponding to 15–16 decimal signs). Or maybe even 32-bit single-precision numbers with a 24-bit mantissa corresponding to seven decimal signs. And the lower the accuracy, due to the computer format used for representing the real number, the more important it is to prevent losses associated with the peculiarities of solving this problem.

Secondly, the solution to Kepler's equation is usually an intermediate result used for further calculations, which are sometimes very difficult. And, in the course of these calculations, some loss of accuracy is also possible (see, for example, [6]). However, it is not always possible to reliably estimate and correctly take into account these losses in accuracy. Thus, it is useful to have the maximum possible margin of accuracy in calculating the solution to Kepler's equation.

Third, even if the error associated with the calculation of the theoretical value of a physical quantity is much less than the observation error, in some cases such a computational error can significantly distort the results associated with the statistical analysis of observations. Thus, in [7], using the example of the interpretation of the transit light curve of the system HD 209458, it was shown how an error in calculating the model light curves leads to a statistically significant change in the residual χ^2.

Also, the need for high accuracy can manifest itself regardless of the accuracy of observations when solv-

ing the problem of minimizing the residual χ^2 during the interpretation of the transit light curve. Many nonlinear optimization methods (for example, the Levenberg–Marquardt method) use the derivatives of the expression to be minimized (or the functions included in it), and the convergence of the method sequence can be significantly affected by the accuracy of calculating these derivatives, the expression for which includes the solution of Kepler's equation for various $\mathcal{M}$ and e and at fixed values of the observational data (the measurement accuracy of which does not matter for the minimization process).

To date, there are various algorithms for solving Kepler's equation, based on rapidly converging iterative sequences of real numbers. However, floating point numbers are different from real numbers, and the difference between floating point calculations and corresponding real calculations can become significant if these calculations involve evaluating expressions close to machine zero. And, in some cases, rounding errors in intermediate results cause a noticeable loss of precision in the final result.

We will consider this problem using an example from [8, 9], which describes an algorithm for solving Kepler's equation based on an iterative sequence of real numbers that quickly converges to the required result. It should be noted here that this convergence statement is true for $0 < \mathcal{M} < \pi$. An elementary numerical check can make sure that the sequence diverges for $e = 0.93$ and $-0.41 \le \mathcal{M} \le -0.39$. It is possible that this divergence is eliminated by choosing the correct initial approximations for the corresponding ranges of parameters. However, even for $0 < \mathcal{M} < \pi$, if we implement this algorithm with 64-bit double precision numbers, we can find that for some values of the initial data the corresponding sequence of floating point numbers does not approach the required result closer than a certain value, a noticeably larger error in the last bit. It is possible to indicate such $\mathcal{M}$ and e that the sequence of computer numbers constructed according to the mentioned algorithm does not approach the solution of Kepler's equation closer than by $10^3\epsilon$, where $\epsilon = 2^{-52}$ is the error in the last digit of double precision numbers. Consequently, the solution of Kepler's equation for such values of the parameters according to this algorithm is possible with an error not less than $10^3\epsilon$. If, as a condition for interrupting the iterative loop in the program, we set the achievement of greater accuracy, then the loop becomes infinite.

It should be noted that such a "failure" in the convergence of a computer sequence is not a result of the singularity and/or instability of the problem of solving Kepler's equation formulated on the set of real numbers. It is due precisely to a given computer representation of numbers. When we go to another representation, for example, to 80-bit extended-precision numbers, for these parameters, the values of the iterative sequence approach the solution to Kepler's equation with an accuracy of the order of the corresponding machine ϵ. Also, there will be no such convergence "failures" with a slight deviation from the above values of $\mathcal{M}$ and e, for example, when only their last digits change. Moreover, the presence of such a "failure" may also depend on the choice of the compiler used to program the algorithm. Therefore, such "failures" of convergence are difficult to detect by simple testing of the program; with testing with a small number of source data variants. However, this "failure" can manifest itself when processing large amounts of data.

In this paper, an algorithm for solving Kepler's equation is constructed, which ensures the achievement of the machine accuracy of the result by taking into account the peculiarities of calculations near the machine zero. For this, the tangent method (Newton's method) is used, the error of which is reduced to machine zero on average in about 5 iterations when using extended precision numbers (the exact number depends on the values of e and $\mathcal{M}$). The construction of an algorithm based on Newton's method seems to be the most suitable for our purposes, since it is in this way that it is more convenient to control the influence of the features of calculations near machine zero.

An essential point in using Newton's method is the choice of the initial approximation, at which the iterative sequence certainly converges to the solution of Kepler's equation. Note that an arbitrarily chosen initial approximation may not provide optimal convergence of the corresponding iterative sequence. The iteration sequence may even be divergent for some values of the initial approximation. Therefore, the choice of the initial approximation is important for solving Kepler's equation. At the same time, the problem of numerically solving Kepler's equation is formulated precisely as the problem of finding the corresponding floating point number. Testing the accuracy and speed of the algorithm is also described.

2. CONSTRUCTING AN ITERATION SEQUENCE

For $0 \le e < 1$ and any real $\mathcal{M}$, we write Kepler's equation as

$$f(x) = 0, \tag{2}$$

where

$$f(x) = x - e\sin x - \mathcal{M}. \tag{3}$$

It is easy to see that $f(\mathcal{M} - e) = -e(1 + \sin(\mathcal{M} - e)) \le 0$, $f(\mathcal{M} + e) = e(1 - \sin(\mathcal{M} - e)) \ge 0$, that is, the function f changes sign at the segment $[\mathcal{M} - e, \mathcal{M} + e]$. Since the function $f(x)$ is continuous and increasing for any x, Eq. (2) always has a unique solution.

In the trivial case $\mathcal{M} = \pi k$, where k is an integer, its solution is $x = \mathcal{M}$, i.e., $\mathcal{K}(\pi k, e) = \pi k$.

To solve this equation by the tangent method, it is necessary to have an interval D containing the desired solution, and such that the second derivative $f''(x)$ does not change sign on it, and one of the ends of this interval z satisfies the condition

$$f''(x)f(z) > 0, \quad \forall x \in D. \tag{4}$$

In this case, according to the well-known statement [10], the sequence of the tangent method defined by the iterative expression

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5}$$

converges to the desired solution if we take z as an initial approximation.

Let us take z as the closest number to $\mathcal{M}$ of the form $\pi(2m+1)$, where m is an integer. If $z > \mathcal{M}$, then

$$f''(x) = e\sin x > 0 \quad \forall x \in (2\pi m, z),$$
$$f(z) = z - \mathcal{M} > 0,$$

thus condition (4) is satisfied if we take the interval $(2\pi m, z)$ as D. If $z < \mathcal{M}$, then

$$f''(x) = e\sin x < 0 \quad \forall x \in (2\pi(m+1), z),$$
$$f(x_0) = z - \mathcal{M} < 0,$$

thus condition (4) is satisfied if we take the interval $(2\pi m, z)$ as D. Therefore, if we choose $x_0 = \pi(2m+1)$ as the initial approximation, the sequence of the tangent method converges to the required solution of Eq. (1). Note that this initial condition can be improved by noting that if x is a nontrivial solution to Eq. (1), then

$$\mathcal{M} - e < x < \mathcal{M} + e,$$

that is, the solution to the equation belongs to the interval $[\mathcal{M} - e, \mathcal{M} + e]$. Therefore, if $z > \mathcal{M}$, we can take $\min(z, \mathcal{M} + e)$ as an improved initial approximation x_0, and if $z < \mathcal{M}$, then take $\max(z, \mathcal{M} - e)$. Condition (4) will also be fulfilled when it is replaced z by x_0.

Substituting (3) into (5), we obtain

$$x_{n+1} = x_n - \frac{x_n - e\sin x_n - \mathcal{M}}{1 - e\cos x_n} = \frac{e(\sin x_n - x_n\cos x_n) - \mathcal{M}}{1 - e\cos x_n}. \tag{6}$$

Thus, sequence (6) converges to the required solution of Kepler's equation if we take as the initial approximation

$$x_0 = \begin{cases} \max(\pi(2m+1), \mathcal{M} - e), & \pi(2m+1) < \mathcal{M} \\ \min(\pi(2m+1), \mathcal{M} + e), & \pi(2m+1) > \mathcal{M}, \end{cases}$$

where m is an integer at which the value of $|\pi(2m+1) - \mathcal{M}|$ takes on a minimum value. The iterative sequence $\{x_n\}$ constructed above is monotonic, and each of its terms is closer to the required solution than the previous ones.

It should be noted that the problem of solving Kepler's equation for an arbitrary value of the mean anomaly can be reduced to the problem of solving Kepler's equation within the first cycle. We consider the general case for convenience in the practical implementation of the method, including in order to exclude the application of the operation of obtaining the remainder of division by 2π to the mean anomaly, taking into account that the architecture of modern computers allows calculating the values of trigonometric functions for any values of the argument at the hardware level.

We emphasize that it is the choice of the initial approximation x_0 that plays an essential role. The specified behavior of the iterative sequence cannot guarantee random selection. For example, choosing as an initial approximation $x_0 = \mathcal{M}$ will, in many cases, also give a converging sequence, and some authors suggest such a choice (see, e.g., [11]). However, there may be values of $\mathcal{M}$ and e at which the members of the sequence with $x_0 = \mathcal{M}$ are significantly removed from the sought solution, for example, if $\mathcal{M} = 0.2$ and $e = 0.9747$ (see Fig. 1). The paper [11] also pointed out the existence of values of $\mathcal{M}$ and e for which the iterative sequence diverges.

3. COMPUTATIONAL ACCURACY NEAR MACHINE ZERO

The construction of an iterative sequence, which converges to the solution of Kepler's equation, presented in the previous section, was obtained in relation to abstract real numbers. In real computing, we operate with floating point numbers, which are considered as approximate values of real numbers. And in the case of replacing real numbers in calculations with floating point numbers, the corresponding results are also only approximately true. In this case, the magnitude of error when directly replacing a real number with a floating point number corresponds to the size of mantissa. If a real number is approximated by an extended precision number whose mantissa contains 64 bits (which guarantees 19 decimal significant digits), then the value of the relative approximation error will be 2^{-64} or $\sim 10^{-19}$.

However, the relative error of the floating point result can be much larger. First of all, this applies to the case of addition (subtraction) of two numbers, when the modulus of the result is significantly less than the terms of sum. If two real numbers are so close that the corresponding floating point numbers do not differ, then the result of a computer calculation of their difference will be zero. While the final result of solving the problem for real numbers depends on this differ-

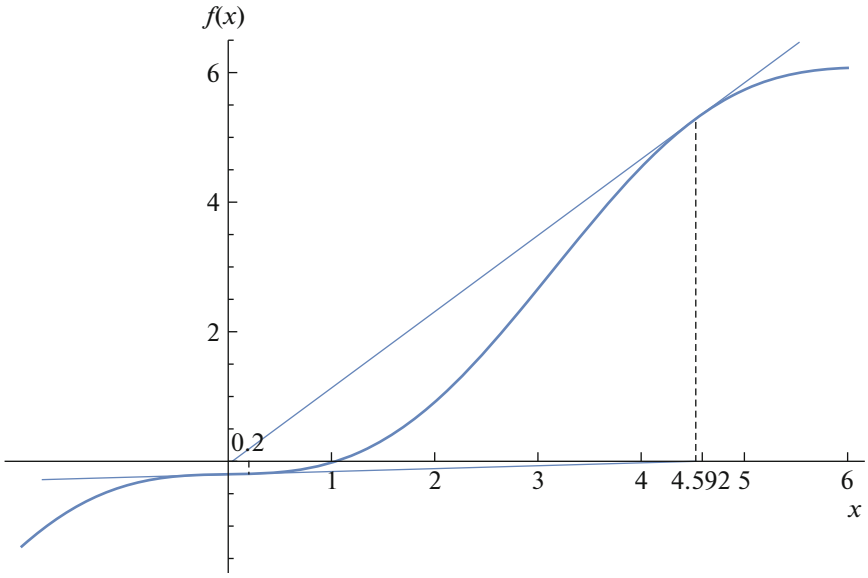

Fig. 1. An example of using the initial approximation $x_0 = \mathcal{M}$, leading to a diverging sequence of solutions to Eq. (3), with the values of the mean anomaly $\mathcal{M} = 0.2$ and eccentricity $e = 0.9747$.

ence as an intermediate result and turning this difference to zero significantly distorts the final result.

If two floating point numbers differ in the last few digits, the relative accuracy of their difference will correspond to exactly this number of digits (say, if the numbers in the last two decimal signs differ, the relative error in calculating their difference will be 1%). Therefore, replacing abstract real numbers with computer floating point numbers can significantly degrade the accuracy of calculating the final result, despite the fact that the formal problem is posed correctly and is resistant to small changes in the input data in the form of abstract real numbers (changes in 2^{-64} in the original data in the form of real numbers entail changes of the same order in the final result). To obtain a result with a given precision ε by a formal implementation of an algorithm that guarantees such precision in relation to abstract real numbers, it is necessary to use floating point numbers whose precision is much higher than ε.

Of course, we can get the result with a predetermined precision ε by formally implementing the algorithm with arbitrary precision numbers. However, the use of this method is undesirable due to a significant increase in the computation time, since the emulation of elementary operations with arbitrary precision numbers requires computer time consumption that is much larger than the execution of elementary operations implemented at the machine level with 80-bit numbers, not to mention 64-bit ones.

In view of the above, in order to obtain the solution to Kepler's equation with the highest possible accuracy, within the framework of using operations with floating point numbers, it seems appropriate to pose the problem not as the problem of finding a numerical value that differs from the true solution of the equation by no more than some given one, but as finding a machine a floating point number, the substitution of which into Eq. (1) gives the minimum modulo value of the left-hand side, that is, the function $f(x) = x - e\sin x - \mathcal{M}$.

Note that one of the results of calculating (1) using floating point numbers may be machine zero. That said, it is also possible that for some machine values of $\mathcal{M}$ and e machine zero will not be reached at any machine value x.

To solve such a problem, the sequence given by expression (5) or (6) is first calculated. As applied to abstract numbers, the difference between x_n and the true solution of Kepler's equation, as well as the value $f(x_n)$, do not change their sign for any n. If $f(x_n)$ it does not change sign for any n in relation to the machine x_n, its absolute value will decrease as n increases, and at some step N (based on the form (5)) will become equal to zero. In this case, we can consider x_n as the required solution.

However, due to the inaccuracy associated with the final representation of real numbers as floating point numbers, the machine value of $f(x_n)$ may change sign at some step. In this case, the further elements of the sequence x_n cannot be said to be the best approximation of Kepler's equation. However, they are close enough to the true Kepler's equation, so it makes sense to fix some two of them, for which the function $f(x_n)$ has different signs, and, starting from the segment formed by them, to solve Eq. (1) by the method of half

division. This method converges much more slowly than the tangent method, and, therefore, it is impractical to use it at large intervals. At the same time, this method is based solely on calculating the value of function $f(x_n)$ and comparing it with zero; therefore, it is not sensitive to errors that arise during operations with the value of $f(x_n)$. Therefore, it makes sense to use it to refine the result obtained using the rapidly converging tangent method.

We emphasize that this approach to finding the numerical value of the solution to Kepler's equation eliminates the error associated with intermediate calculations (with the final representation of the numbers used in it). However, in any case, the final result may contain an error caused by an error in the original data, which, in turn, is associated with the final representation of these initial data (values M and e) and corresponds to the value of the last machine bit (2^{-64} for extended precision numbers with a 64-bit mantissa). Such an error (exceeding the round-off error) appears if the derivative $f'(x) < 1$. We can say that this error is due to the sensitivity of the result to the original data.

For comparison, we implemented the algorithm described in [8, 9] with 64-bit double precision numbers, the relative rounding error of which is $\epsilon = 2^{-52}$. This algorithm is based on the use of an iterative sequence converging to the solution of Kepler's equation

$$E_{n+1} = E_n - \frac{(\mathcal{M} + e\sin E_n - E_n)^2}{E_n - 2(\mathcal{M} + e\sin E_n) + \mathcal{M} + e\sin(\mathcal{M} + e\sin E_n)},$$

where $E_0 = M + 0.85e$. It is proposed to calculate such a sequence, while $|E_{n+1} - E_n| > \varepsilon$, where ε is the required accuracy of calculations. Recall once again that for the indicated initial approximation, this sequence does not converge for some ranges of values of $\mathcal{M}$ outside $0 < \mathcal{M} < \pi$. However, even when there may be difficulties with the construction of the corresponding sequence of computer numbers for the most accurate calculation. Similar difficulties can arise with other algorithms for solving the Kepler's equation; we will consider them using the example of the algorithm [8, 9].

For the implementation with 64-bit double precision numbers, as already mentioned above, for some values of $\mathcal{M}$ and e, the values of this iterative sequence do not approach the solution of Kepler's equation closer than by a value significantly greater than the rounding error to the last bit of the 64-bit machine number ϵ. Recall that such numbers depend on the specific computer implementation of the algorithm. For example, when implemented on one of the C compilers, for $\mathcal{M} = 0.09912109375$ and $e = 0.70849609375$ the values of the relative difference $|(E_{n+1} - E_n)/E_{n+1}|$ do not become less than $\epsilon 10^3$, where ϵ is the rounding error to the last bit of a 64-bit machine number. For implementation on another compiler, such values are, for example, $\mathcal{M} = 0.00653076171875$ and $e = 0.9605560302734375$. At the end of this paper, we provide a link to a program that detects such values $\mathcal{M}$ and e. If, as a condition for interrupting an iterative loop, a decrease in the absolute value of the relative difference ($|(E_{n+1} - E_n)/E_{n+1}|$) between the elements of the iterative sequence is set by less than by less than by $\epsilon 10^3$, such a cycle becomes infinite (the program loops). In order to guarantee to avoid this kind of situation for any values of the initial parameters, in the given algorithm it is necessary to choose a sufficiently large ε. At the same time, a rather nontrivial question arises as to what exactly this ε should be. But already from the above example, it is clear that this value of ε should at least 10^3 times exceed the rounding error to the last bit of the machine representation of the number.

The implementation of our algorithm with double precision numbers allows us to compute the solution of Kepler's equation with the above values $\mathcal{M}$ and e with an accuracy of 15 decimal places, that is, at the level of machine precision corresponding to such a representation of the number.

Further, we carried out numerical testing of the accuracy of the values of the solution to Kepler's equation obtained using the described algorithm. To do this, we implemented an algorithm with extended precision numbers (80-bit numbers with a 64-bit mantissa, which gives 19 significant digits in decimal form) and with numbers of higher (emulated) precision. As mentioned above, for practical calculations, the use of such numbers in many cases is ineffective, since it greatly increases the computation time. However, with their help it is convenient to check calculations made with ordinary machine numbers.

We performed calculations using the described algorithm, both with emulated numbers containing significantly more characters than a machine 80-bit number, and with 80-bit numbers (with which elementary operations are performed at the machine level). The result with emulated numbers was guaranteed to be more accurate than the result with 80-bit numbers. Next, we calculated the difference between the result obtained using 80-bit numbers and using emulated numbers of increased precision with the same input values $\mathcal{M}$ and e. In total, we carried out calculations of such a difference for 10^8 values of the pair of $\mathcal{M}$ and e, as the value $\mathcal{M}$ was taken a pseudorandom number with a uniform distribution on the interval $[0, \pi]$, and as the value of e was taken a pseudorandom number with a uniform distribution on the interval [0, 1]. The same calculations were performed with the values $\mathcal{M}$ and e located in the form of a uni-

form grid of 10^4 values in the corresponding interval. In case $f'(x) < 1$, we multiplied the resulting difference by $f'(x)$ to account for the inevitable error caused by the sensitivity of the result to the rounding error of the input values. In each case, the result turned out to be less than 10^{-19}, which allows us to conclude that the 80-bit value of the solution to Kepler's equation was obtained with an accuracy that is maximum possible for an 80-bit representation of the number, and is practically not distorted as a result of intermediate calculations.

For the same 10^8 pairs of input values M and e (random and located on a uniform grid), we tested the number of iterations required to achieve maximum accuracy. The average number of such iterations, taking into account the iterations by the method of half division, turned out to be approximately 5.51 (it can increase at e close to unity). Note that if we consider only iterations by the tangent method, their average number turns out to be slightly less, approximately 5.28. Thus, we can conclude that the need to refine the result by the half-division method arises, although the total volume of such calculations is relatively small, about 4% of the total number of iterations. However, in mass processing of observations, we almost inevitably encounter such situations.

4. CONCLUSIONS

The solution to Kepler's equation is often an intermediate result used for further, often very complex, calculations, in which a significant loss of accuracy inevitably occurs. An example of such calculations would be the interpretation of the light curve. At the same time, for statistical analysis of observational data, sometimes an accuracy may be required that significantly exceeds the accuracy of observations [7]. Equally, a high accuracy in the calculation of expressions containing the solution of Kepler's equation, regardless of the accuracy of the observational data, is desirable when solving the problem of minimizing the residual χ^2 (see, for example, [1–4]). In complex calculations, it seems useful to use every opportunity to increase the accuracy of calculations, including because it is sometimes very difficult to estimate the required accuracy, especially when it comes to computational problems that may arise in the future. Therefore, when solving Kepler's equation, it is important to strive for the most accurate result.

The authors in this paper propose an algorithm for quickly calculating the solution of Kepler's equation with the best accuracy for a given machine representation of real numbers. The proposed algorithm consists of two stages.

The first step is to use the fast converging tangent method to obtain a result with an accuracy that this method can provide, taking into account the finiteness of the machine representation of numbers. In this case, an effective choice of the initial approximation is essential, such that the successive approximations are a converging monotonic sequence, each term of which is closer to the required value than the previous ones.

The second step is to use the half division method to finalize the obtained value, which in some cases may contain inaccuracy in the last digits. In this case, Newton's method provides a simple criterion by which it is possible to effectively determine the need for a special refinement of the solution (sign change of $f(x_n)$. In this case, the problem of solving Kepler's equation is formulated in relation to the machine representation of numbers as the problem of finding the machine number representing the best possible approximation of the solution of Kepler's equation for given values of the mean anomaly and eccentricity.

A comparison is made of the efficiency of the algorithm proposed by the authors for solving Kepler's equation and another algorithm, in many respects similar to Newton's method. It is noted that for some values $\mathcal{M}$ and e the iteration sequence constructed in the machine representation of real numbers may stop converging, differing from the minimum possible error for a given representation by at least a factor of 10^3. Moreover, such a situation arises in a relatively small percentage of possible values $\mathcal{M}$ and e; it is not associated with any peculiarities of the solution of Kepler's equation as applied to abstract real numbers. Therefore, the possibility of such a situation may well not be detected by simple testing of the formal implementation of the algorithm (which does not take into account the specifics of the machine representation of numbers), but it may manifest itself in the future, when using such an implementation when processing a large dataset. At the same time, taking into account the possibility of such a situation, without going beyond the scope of the algorithm described in [8, 9], one can only significantly overestimate the error for the final result ε.

The algorithm proposed by the authors makes it possible to obtain the solution to Kepler's equation with an accuracy at the level of round-off error for the used computer representation of a real number (machine accuracy) for absolutely any input values of $\mathcal{M}$ and e. The efficiency of using such an algorithm is manifested primarily in the complex processing of large datasets, for example, in the problem of interpreting modern observed transit light curves, which can contain up to several tens of thousands of observation points. Thus, the algorithm proposed by the authors has a certain advantage over the algorithm described in [8, 9], even if for most input values of $\mathcal{M}$ and e for these algorithms there is no significant difference in the convergence rate and the achieved accuracy.

By means of a numerical experiment on a large set of synthetic initial data, the accuracy provided by the algorithm has been verified; it has been shown that it is the maximum achievable for the used representation of a real number. The average number of iterations was also estimated, on which the computation time directly depends on 80-bit floating point numbers (which in many modern computers provide maximum machine precision).

The algorithm developed by the authors is publicly available. Its software implementation in the C language is located on the website of Sternberg Astronomical Institute.[1] Here is the file with the implementation of the algorithm for 64-bit and 80-bit representation. There is a separate implementation for the 128-bit representation used in some C/C++ compilers. Although for most modern tasks such accuracy is likely to be excessive and does not justify the increased time spent on operations with such long numbers, this implementation is useful for testing accuracy. Also, at the given link, there is a program with the implementation of the Danby method [8, 9] for 64-bit numbers, with control over the achievement of a given relative accuracy by counting the number of iterations. It demonstrates the absence of convergence of the sequence E_n mentioned in the paper for some values $\mathcal{M}$ and e, which are found by random search. In this case, as the criterion that the sequence does not converge, the condition is taken that at $n > 500$, that is, after 500 steps, $|(E_{n+1} - E_n)/E_{n+1}| > N\varepsilon$, where N is a given number (it was initially set as 1000).

Examples of using the algorithm developed by the authors for solving the Kepler's equation from [7] are also located on the website of the Sternberg Astronomical Institute,[2] in the OccultationPack3 and DemoPack1 software complexes.

REFERENCES

1. M. K. Abubekerov, N. Yu. Gostev, and A. M. Cherepashchuk, Astron. Rep. **52**, 99 (2008).
2. M. K. Abubekerov, N. Yu. Gostev, and A. M. Cherepashchuk, Astron. Rep. **53**, 722 (2009).
3. M. K. Abubekerov, N. Yu. Gostev, and A. M. Cherepashchuk, Astron. Rep. **54**, 1105 (2010).
4. N. Yu. Gostev, Astron. Rep. **55**, 649 (2011).
5. G. N. Duboshin, *Celestial Mechanics. Fundamental Problems and Methods* (Moscow, Nauka, 1968) [in Russian].
6. M. K. Abubekerov and N. Yu. Gostev, Astron. Rep. **63**, 107 (2019).
7. M. K. Abubekerov and N. Yu. Gostev, Astron. Astrophys. **633**, A96 (2020).
8. J. M. A. Danby, *Fundamentals of Celestial Mechanics,* 2nd ed. (Willmann-Bell, USA, 1995).
9. N. V. Emel'yanov, *Dynamics of Natural Satellites of Planets Based on Observations* (Vek-2, Fryazino, 2019) [in Russian].
10. A. N. Kolmogorov and S. V. Fomin, *Elements of Function Theory and Functional Analysis* (Moscow, Nauka, 1976) [in Russian].
11. A. W. Odell and R. H. Gooding, Celest. Mech. Dyn. Astron. **38**, 307 (1986).

Translated by E. Seifina

[1] http://lnfm1.sai.msu.su/ngostev/Files/Kepler.zip

[2] http://lnfm1.sai.msu.su/ngostev/algorithm.html