An efficient code to solve the Kepler equation. Elliptic case

V. Raposo-Pulido*† and J. Peláez*‡

Space Dynamics Group–Technical University of Madrid, Madrid 28040, Spain

Accepted 2017 January 16. Received 2016 December 15; in original form 2016 November 17

ABSTRACT
A new approach for solving Kepler equation for elliptical orbits is developed in this paper. This new approach takes advantage of the very good behaviour of the modified Newton–Raphson method when the initial seed is close to the looked for solution. To determine a good initial seed the eccentric anomaly domain \([0, \pi]\) is discretized in several intervals and for each one of these intervals a fifth degree interpolating polynomial is introduced. The six coefficients of the polynomial are obtained by requiring six conditions at both ends of the corresponding interval. Thus the real function and the polynomial have equal values at both ends of the interval. Similarly relations are imposed for the two first derivatives. In the singular corner of the Kepler equation, \(M \ll 1\) and \(1 - e \ll 0\) an asymptotic expansion is developed. In most of the cases, the seed generated leads to reach machine error accuracy with the modified Newton–Raphson method with no iterations or just one iteration. This approach improves the computational time compared with other methods currently in use.

Key words: methods: numerical – space vehicles – celestial mechanics.

1 INTRODUCTION
The basic goal of the Kepler equation (KE) is to provide the value of the eccentric anomaly \(E\) in terms of the mean anomaly \(M\) and the eccentricity \(e\) of the orbit:

\[
M = E - e \sin E.
\]

Over the centuries the resolution of the KE has been studied for a wide variety of scientists such that in virtually every decade from 1650 to the present there have appeared papers devoted to the Kepler problem and its solution (Colwell 1993). Although there exists a unique solution of the KE, there are many methods to describe or approximate it. Basically it depends on the solver’s motivations and the mathematical tools available in the epoch of study.

Before 1860 the work was motivated directly by celestial mechanics, but it showed results less sensitive to the needs of astronomers. After 1860 the development and refinement of what we now call the special functions of applied mathematics became the involvement of mathematicians in solving KE more outstanding. In the case of the astronomers, they were generally more interested in being able to solve KE quickly and repeatedly, but with only modest accuracy (Rambaut 1906). From 1860 to sometime after 1890, the work on KE was mostly derivative and computational (Colwell 1993). We can highlight the work developed by Cassini (1719) providing a non-analytic solution, the infinite series solution developed by Levi-Civita (1904) or the iterative methods based on the Newton’s Principia (1686).

From mid-20th century, the classic and more recent methods focused on the development of optimal procedures where the accuracy of the algorithm and the computational effort become the main aspects to consider. Most of the algorithms are iterative and they depend on how the starting estimate for \(E\) is determined. Nevertheless, nowadays it is possible to find analytical algorithms as the method described in Pál (2009) to solve Kepler’s problem. In what follows we briefly describe some of the approaches currently used to solve equation (2).

(i) The Serafin approach described in Serafin (1986). The bounds for \(E\) are chosen through several inequalities to successively divide the root interval and improve the lower and upper bounds. To do that, two linear methods based on elementary properties of convex functions are used.

(ii) The Gooding and Odell approach described in Gooding & Odell (1988). The starter value for \(E\) is provided by solving a cubic equation. After that, a corrected value is given through the Halley’s method which is used as input to approximate the evaluation of equation (2) and the derivative by their Taylor series. The final value is computed with the Newton–Raphson method. It is an iterative method which requires several transcendental function evaluations.

(iii) The Nijenhuis approach described in Nijenhuis (1991). The starter for \(E\) is computed depending on four different regions of \((M, e)\)-space. In three of these regions the starter value is defined with a Halley step based on a version of the KE in which the sine function is replaced by a simple approximation. In the remaining
region, which contains the singular point, the starter value is given by the solution of Mikkola’s cubic equation and refined through a Newton step applied to quintic equation. An mth order generalized Newton method is applied thereafter to obtain the final solution, where m depends on the accuracy required. It is an iterative method which requires two trigonometric evaluations.

(iv) The Markley approach described in Markley (1995). The start point for $E$ is the root of a cubic equation, whose expression is derived through a Padé approximation for sin $E$, which is replaced in equation (2). After that, a fifth-order refinement of the solution is applied only once. It is a no iterative method which requires four transcendental function evaluations (a square root, a cube root and two trigonometric functions).

(v) The Fukushima approach described in Fukushima (1996). The starter value for $E$ is a trivial upper bound of the solution: $\pi$. The approximate solution is determined by transforming the Newton–Raphson method from the continuous $x$-space to a discretized $j$-space, where $x$ represents $E$ and $j$ the solution index. The corrected value for $E$ is another approximation of the Newton–Raphson method by approximating the evaluation of equation (2) and the derivative by their truncated Taylor series around the approximate solution. It is an iterative method which does not require transcendental functions.

(vi) The Mortari and Elipe approach described in Mortari & Elipe (2014). Two ranges of mean anomaly are defined to select where the solution should be determined. The lower bound for $E$ is derived through two implicit functions, which are non-rational Bézier functions, linear or quadratic, depending on the derivatives of the initial bound values. The upper bound for $E$ is estimated by applying the Newton–Raphson method with the lower bound as the starting value. After that, if the machine error accuracy is not reached, the lower and upper bounds define a new range of searching. It is an iterative method which requires several transcendental function evaluations (one quadratic iteration requires 19 additions, 20 products and one square root, while one linear iteration requires seven additions and five products).

In the last century, with the advent of calculators and computers, there is no impediment to achieving quick solutions of great accuracy. In particular, the symbolic manipulators like MATLAB, MAPLE or MATHEMATICA are very powerful calculators which are easy to use and have a very intuitive syntax. Besides, they can be directed to provide the corresponding code in some of the standard programming languages such as C, C++ or FORTRAN, which are specially adapted for numerical calculus. In this paper we develop a procedure to solve the KE for elliptic orbits taking advantage of the full potential of the symbolic manipulators. We focus the study on the determination of an appropriate seed to initialize the numerical method for solving the KE, considering the optimization already tested of the well-known Newton–Raphson method.

2 ELLIPTIC KEPLER EQUATION

The KE for the elliptical case,

$$x = y - e \sin y,$$  \hspace{1cm} (1)

determines a non-linear function $y = y(x, e)$, where $y = E$ is the unknown eccentric anomaly, $M = x$ the mean anomaly which is known and $e < 1$ the eccentricity. For given values of $e \in [0, 1]$ and $M \in [0, 2\pi]$ the function $y = y(x, e)$ is univocally defined by equation (1) and this property can be deduced from the Banach fixed-point theorem.

The function $g(y)$ is contractive if there is a non-negative real number $k \in [0, 1]$ such that $\forall y_1, y_2 \in [0, 2\pi]$:

$$|g(y_1) - g(y_2)| \leq k|y_1 - y_2|.$$  \hspace{1cm}

The smallest value for $k$ is the Lipschitz constant of $g$.

If we consider the equation $g(y) = x + e \sin y$, the function $g(y)$ is a contractive mapping for every value of $x$, since its derivative $g'(y) = e \cos y$ satisfies the condition $|g'(y)| < 1$ when $e \in [0, 1]$. In effect, applying the mean value theorem to function $g(y)$, there exist an intermediate value $y^* \in [y_1, y_2]$ which verifies

$$g(y_2) - g(y_1) = g'(y^*)(y_2 - y_1).$$  \hspace{1cm}

Taking into account that $g'(y) = e \cos y$, we have

$$|g(y_2) - g(y_1)| = |e \cos y^*| |y_2 - y_1| \leq e|y_2 - y_1|,$$  \hspace{1cm}
i.e. $k$, the Lipschitz constant of $g$, verifies $k < e$, proving the property.

Finally, applying the Banach fixed-point theorem for contractive functions, it can be concluded that there is a fixed point which verifies $y = g(y)$ and this fixed point is the looked for solution of the KE. Starting from a given value $y_0$, for example $y_0 = x$, the sequence $y_{n+1} = g(y_n)$ turns out to be convergent and its limit is the fixed point.

However, this sequence is slowly convergent – especially when $e$ is close to unity – and this way to solve KE is not effective, from a numerical point of view. Thus, the interest of this procedure is much more theoretical than practical.

The solution of the KE can be seen as a root finder problem of the function

$$f(y) = y - e \sin y - x = 0,$$  \hspace{1cm} (2)

for given values of $e$ and $x$. Because of the transcendental character of the equation its solution is not trivial. Nevertheless, by introducing the function

$$F(y) = y - \frac{f(y)}{f'(y)} = \frac{e(\sin y - y \cos y) + x}{1 - e \cos y},$$

where $f'(y)$ is the derivative of $f(y)$, the solution of the KE reduces to the search of the fixed point of $F(y)$; note that $y = F(y) \in f(y) = 0$. Again, we can form a convergent sequence $y_{n+1} = F(y_n)$ whose limit is the root of the function $f(y)$; the convergence of this new sequence is faster than the previous one (see Stumpf 1999) but still is insufficient from a practical point of view.

There are several analytical solutions given by expansions in terms of Bessel functions of first kind like

$$E = M + 2 \sum_{k=1}^{\infty} J_k(k e) \frac{\sin(k M)}{k},$$

where $J_k(k e) = \int_0^\pi \cos(k u - k e \sin u) du$.

More analytical solutions can be found in references Batin (1999), Siewert & Burniston (1972) and Colwell (1993). However, these expansions are not competitive among the numerical methods of solution, as J. M. A. Danby comments in his book (Danby 1992).

2.1 Domain of the equation

Initially, $x$ and $y$ range in the interval $[0, 2\pi]$; however, the transformation

$$y = 2\pi - \eta, \quad x = 2\pi - \xi$$  \hspace{1cm} (3)

convert equation (1) into

\[ \xi = \eta - e \sin \eta. \]

As a consequence, if \( x, y \in [\pi, 2\pi] \Rightarrow \xi, \eta \in [0, \pi] \) and the mapping (3) reduces the problem to the interval \([0, \pi]\).

3 NEWTON–RAPHSON ALGORITHMS

The root searched is solution of equation (2) and we wish to establish a successive approximation method starting from a seed \( y_0 \). In the classical Newton–Raphson (CNR) algorithm a sequence is formed whose elements become arbitrarily close to each other as the sequence progresses:

\[ y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)}. \]

This Cauchy sequence converges to the solution and it is very well known that the convergence rate of this algorithm is quadratic; thus the number of correct figures is doubled, approximately, in each iteration of the sequence.

A better convergence rate is obtained with the modified Newton–Raphson (MNR) algorithm which we use in our code to solve the KE. If \( y_n \) in one of the terms of the sequence, the next term will be \( y_{n+1} = y_n + \Delta y_n \), where \( \Delta y_n \) should verify

\[ f(y_{n+1}) = f(y_n + \Delta y_n) = 0, \]

with \( f \) given by equation (2). By approximating \( f(y_{n+1}) \) by its second-order Taylor expansion about \( y_n \),

\[ f(y_{n+1}) \approx f(y_n) + f'(y_n)\Delta y_n + \frac{1}{2} f''(y_n)\Delta y_n^2, \]

the condition \( f(y_{n+1}) = 0 \) provides the value of \( \Delta y_n \):

\[ \Delta y_n = -\frac{f'(y_n)}{f''(y_n)} \sqrt{\frac{f''(y_n)}{f'(y_n)}}. \]

Then, we multiply numerator and denominator by

\[ -f'(y_n) \pm \sqrt{f''(y_n) - 2f(y_n)f''(y_n)} \]

to obtain

\[ \Delta y_n = \frac{-2f(y_n)}{f'(y_n) \pm \sqrt{f''(y_n) - 2f(y_n)f''(y_n)}}. \]

Table 1. The right answer is \( y = 2.78172231 \).

<table>
<thead>
<tr>
<th>Seed ( y_0 = 0.1 )</th>
<th>Seed ( y_0 = 0.3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNR</td>
<td>MNR</td>
</tr>
<tr>
<td>12.25640084</td>
<td>5.82975453</td>
</tr>
<tr>
<td>-29.74030805</td>
<td>1.97868144</td>
</tr>
<tr>
<td>0.74394754</td>
<td>2.76120459</td>
</tr>
<tr>
<td>6.32098624</td>
<td>2.78172169</td>
</tr>
<tr>
<td>-12.55060918</td>
<td>2.78172231</td>
</tr>
<tr>
<td>62.72807874</td>
<td>2.78172231</td>
</tr>
</tbody>
</table>

Table 2. The right answer is \( y = 2.781722308989884 \).

| Seed \( y_0 = 2.8 \) |
|-------------------|-------------------|
| CNR | MNR |
| 2.781748270190563 | 2.78172746897827 |
| 2.78172230904470 | 2.781723208989884 |
| 2.781722308989884 | 2.781722308989884 |
| 2.781722308989884 | 2.781722308989884 |
| 2.781722308989884 | 2.781722308989884 |
| 2.781722308989884 | 2.781722308989884 |

for finding the roots of a polynomial, it works equally as well for KE. The algorithm associated with that method summarizes in the equation

\[ \Delta y_n = \frac{-1(p + 1)f(y_n)}{f'(y_n) \pm \sqrt{|p(p f''(y_n) - (1 + p)f(y_n) f''(y_n))|}}, \]

with \( p = m - 1 \) and \( m \) is the degree of the polynomial. The convergence of the algorithm is almost independent of \( m \); Conway opts for \( m = 5 \). As we can see, the MNR method is a particular case of the Conway method for \( m = 2 \).

The main difference between the CNR and the Conway or the MNR methods is associated with their reliability. The three methods need an initial seed from which start the generation of the convergent sequence. The CNR method is much more affected by the quality of the initial seed than the Conway or the MNR methods. If the quality of the initial seed is poor the CNR method could not converge to the solution.

Table 1 shows the different sequences generated by the CNR and the MNR algorithms when trying to solve the KEs with values \( M = 2.5 \) and \( e = 0.8 \) and two different seeds: the bad one, \( y_0 = 0.1 \), and another not too bad, \( y_0 = 0.3 \). Only the first six terms of the sequence has been calculated.

For the bad seed the CNR does not converge to the real solution; on the contrary, the MNR always converges to the real answer even though we start from a very poor seed.

However, the behaviour of the algorithms is quite different if we use a very good seed as starting point. Table 2 shows the first six terms of the sequences generated for the same values \( M = 2.5 \), \( e = 0.8 \) when the initial seed is very good: \( y_0 = 2.8 \). Both algorithms converge quickly to the real answer.

The accuracy of the Newton–Raphson algorithms is jeopardized when \( M \ll 1 \) and \( 1 - e \ll 1 \); we call this region of the plane \((e, M)\), the singular corner. In such a region \( y \) is also small, and the expansion of the equation for small values of \( y \) provides

\[ y \approx \frac{x}{1 - e}. \]
Table 3. Important values for the polynomial approach, where \( y_i = \sin((i - 1)\pi/12) \) and \( y'_i = \cos((i - 1)\pi/12) \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( x_i = y_i - \sin y_i )</th>
<th>( y_j = (i - 1)\pi/12 )</th>
<th>( d y/y_i )</th>
<th>( d^2 y/y_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/12( \pi )</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>( \pi/12 ) - ( \pi/2 )</td>
<td>( \pi/12 )</td>
<td>1/12( \pi )</td>
<td>2/12( \pi )</td>
</tr>
<tr>
<td>3</td>
<td>( \pi/6 ) - ( \pi/4 )</td>
<td>( \pi/6 )</td>
<td>1/12( \pi )</td>
<td>4/12( \pi )</td>
</tr>
<tr>
<td>4</td>
<td>( \pi/4 ) - ( \pi/3 )</td>
<td>( \pi/4 )</td>
<td>1/12( \pi )</td>
<td>6/12( \pi )</td>
</tr>
<tr>
<td>5</td>
<td>( \pi/3 ) - ( \pi/2 )</td>
<td>( \pi/3 )</td>
<td>1/12( \pi )</td>
<td>8/12( \pi )</td>
</tr>
<tr>
<td>6</td>
<td>( \pi/12 ) - ( \pi/6 )</td>
<td>( \pi/12 )</td>
<td>1/12( \pi )</td>
<td>10/12( \pi )</td>
</tr>
<tr>
<td>7</td>
<td>( \pi/6 ) - ( \pi/4 )</td>
<td>( \pi/6 )</td>
<td>1/12( \pi )</td>
<td>12/12( \pi )</td>
</tr>
<tr>
<td>8</td>
<td>( 2\pi/12 ) - ( \pi/3 )</td>
<td>( 2\pi/12 )</td>
<td>1/12( \pi )</td>
<td>14/12( \pi )</td>
</tr>
<tr>
<td>9</td>
<td>( 2\pi/6 ) - ( \pi/2 )</td>
<td>( 2\pi/6 )</td>
<td>1/12( \pi )</td>
<td>16/12( \pi )</td>
</tr>
<tr>
<td>10</td>
<td>( 2\pi/3 ) - ( \pi/4 )</td>
<td>( 2\pi/3 )</td>
<td>1/12( \pi )</td>
<td>18/12( \pi )</td>
</tr>
<tr>
<td>11</td>
<td>( 2\pi/4 ) - ( \pi/6 )</td>
<td>( 2\pi/4 )</td>
<td>1/12( \pi )</td>
<td>20/12( \pi )</td>
</tr>
<tr>
<td>12</td>
<td>( 2\pi/12 ) - ( \pi/3 )</td>
<td>( 2\pi/12 )</td>
<td>1/12( \pi )</td>
<td>22/12( \pi )</td>
</tr>
<tr>
<td>13</td>
<td>( \pi )</td>
<td>( \pi )</td>
<td>1/12( \pi )</td>
<td>0</td>
</tr>
</tbody>
</table>

i.e. the solution is bad determined from a numerical point of view since it is the ratio of two infinitesimals.

The central idea, and also the goal of this work, is to generate a very good initial seed in all possible cases, also in the singular corner. Then we will use one of the Newton–Raphson algorithms to find the right solution of the KE. Roughly, we exploit the very good characteristics of the Newton–Raphson algorithms (CNR, MNR or Conway) because they assure the convergence and provide the solution with accuracy and velocity if the quality of the initial seed is high.

In this paper, we try to demonstrate that it is much more cost-effective, from a computational point of view, to determine the initial seed and then use it to feed a Newton–Raphson algorithm than to look for new algorithms which would be able to beat the simplicity and performances of the Newton–Raphson algorithms.

Moreover, the solution of the KE is an issue with more than three centuries (see Colwell 1993); along such a long period of time, several methods have been proposed most of them associated with mathematical tools or techniques of its epoch. At present, one important mathematical tool which characterizes this epoch is the symbolic manipulator (MACSYMA, MAXIMA, DERIVE, MAPLE, MATLAB, etc.). We try to exploit these tools to the full, in order to generate a C or FORTRAN code robust, reliable, exact and fast.

4 THE SEED VALUE

Our approach starts discretizing the eccentric anomaly domain \([0, \pi]\) in 12 intervals of \( \pi/12 \) length, where the corresponding \( M_i \equiv x_i \) is obtained from equation (1) (see Table 3). This way, we defined 12 intervals \([x_i, x_{i+1}]\), \( i = 1, \ldots, 12 \).

In each one of these intervals we introduce a fifth-degree polynomial \( p_i(x, \epsilon) \), \( i = 1, \ldots, 12 \) to interpolate the eccentric anomaly:

\[
p_i(x, \epsilon) = a_0^{(i)}(\epsilon) + a_1^{(i)}(\epsilon)x + a_2^{(i)}(\epsilon)x^2 + a_3^{(i)}(\epsilon)x^3 + a_4^{(i)}(\epsilon)x^4 + a_5^{(i)}(\epsilon)x^5.
\]

Notice that the polynomials used in the method have coefficients that depend on the eccentricity (see Appendix A). In order to simplify the equation, we abuse notation by writing the coefficients only as a function of \( i \), i.e. the polynomial they are related, \( p_i(x) = a_0^{(i)} + a_1^{(i)}x + a_2^{(i)}x^2 + a_3^{(i)}x^3 + a_4^{(i)}x^4 + a_5^{(i)}x^5 \).

The polynomial \( p_i(x) \) is only valid inside the interval \([x_i, x_{i+1}]\), \( i = 1, \ldots, 12 \), where we approximate the function by the polynomial: \( y(x) \approx p_i(x) \).

In order to determine the six coefficients of \( p_i(x) \) the following six boundary conditions should be imposed:

\[
p_i(x_i) = y(x_i), \quad p_i'(x_i) = y'(x_i), \quad p_i''(x_i) = y''(x_i), \quad p_i'(x_{i+1}) = y'(x_{i+1}), \quad p_i''(x_{i+1}) = y''(x_{i+1}),
\]

where the two first derivatives of \( y \) are given by

\[
y' = \frac{dy}{dx} = \frac{1}{1 - \epsilon \cos y}, \quad y'' = \frac{d^2y}{dx^2} = \frac{-\epsilon \sin y}{(1 - \epsilon \cos y)^2},
\]

and they can be easily calculated at the ends of the interval. These conditions assure that the function \( y(x) \) and the polynomial \( p_i(x) \) share the same values and the same first and second derivatives at the ends of the interval \([x_i, x_{i+1}]\).

The polynomials \( p_i(x), i = 1, \ldots, 12 \), can be generated easily with the help of a symbolic manipulator only once; moreover, the symbolic manipulator provides the FORTRAN or C code of the polynomials that will be appropriately stored in one module to be used later. We use the version 18.02 of MAPLE.

Consequently, given \( \epsilon \) and \( M \), we determine the starter value \( y_0 \) following the next procedure:

(i) First, given \( \epsilon \) the ends \( x_i, x_{i+1} \) of the intervals described in Table 3 are calculated.

(ii) The interval \([x_i, x_{i+1}]\) should be selected in such a way that \( M \in [x_i, x_{i+1}] \). Note that the corresponding values \([y_i, y_{i+1}]\) are known and the seed \( y_0 \) satisfies: \( y_0 \in [y_i, y_{i+1}] \).

(iii) Use the corresponding polynomial \( p_i(x) \) to obtain the seed to be used with the Newton–Raphson algorithm:

\[
y_0 = p_i(M).
\]

Most of the times, this initial seed is a very good approximation to the solution of the KE; especially when \( M \) is close to the ends of the interval \([x_i, x_{i+1}]\) since this seed is practically the solution of the KE. However, it is well known the singular behaviour the KE in the singular corner. In this zone the approximation of \( y(x) \) by means of the polynomials \( p_i(x) \) is not very effective and a special analysis should be performed.

5 ANALYSIS IN THE SINGULAR CORNER

To describe the solution close to the singular corner we introduce the value \( \epsilon = 1 - \epsilon \) assuming that \( \epsilon \ll 1 \):

\[
x = y - (1 - \epsilon) \sin y = y - \sin y + \epsilon \sin y.
\]

Our goal is to describe numerically the exact solution \( y_c(x) \) of equation (11) with enough accuracy to be part of the seed used to start the Newton–Raphson convergent process. In order to do that, an asymptotic expansion in power of the small parameter \( \epsilon \) will be obtained.
In the singular corner, when $x$ is small, $y$ is also small. However it is possible to distinguish several regimes inside this region.

(i) The inner region, where $x \approx O(\epsilon^2)$ and $y \approx O(\epsilon)$. In this inner region the most important term on the right-hand side of equation (11) is $\epsilon \sin y$.

(ii) The intermediate region, where $x \approx O(\epsilon^2)$ and $y \approx O(\epsilon^\frac{3}{2})$. In this intermediate region both terms on the right-hand side of equation (11) are of the same order: $y - \sin y \approx \epsilon \sin y$.

(iii) The outer region, where $x \approx O(\epsilon^2)$ and $y \approx O(\epsilon^3)$. In this outer region the most important term on the right-hand side of equation (11) is $y - \sin y$.

In each one of these regions it is possible to obtain an asymptotic expansion of the exact solution $y(x)$ of equation (11) in terms of some power of the small parameter $\epsilon$. During the study we assessed the boundaries of the regions, obtaining the best accuracy when

(i) the inner region is defined, approximately, for $(x, \epsilon)$ satisfying $x < 0.001 \epsilon^2$.

(ii) the intermediate and outer region are defined in the same region for $(x, \epsilon)$ satisfying $x \geq 0.001 \epsilon^2$.

As a consequence, only two regions in the singular corner are considered at the end: inner and intermediate-outer; these approximate solutions permit to generate a very good seed to feed the Newton–Raphson convergent process (see Fig. 1).

### 5.1 Inner region

In the inner region we introduce the following change of variables:

$$x = \epsilon^2 \xi, \quad y = \epsilon \eta, \quad \text{with} \quad \xi, \eta \approx O(1). \quad (12)$$

Note that $(\xi, \eta)$ are both positive amounts. We look for a solution of equation (11) given by

$$\xi = \eta + a_1 \epsilon + a_2 \epsilon^2 + a_3 \epsilon^3 + \cdots,$$

where the coefficients $a_i, i = 1, 2, \ldots$ can be obtained expanding appropriately the right-hand side of equation (11). The solution is

$$\xi = \eta + \frac{1}{6} \eta^3 \epsilon - \frac{1}{2} \eta^5 \epsilon^2 - \frac{1}{120} \eta^7 \epsilon^3 + \frac{1}{120} \eta^5 \epsilon^3 + \cdots. \quad (13)$$

The next step is to invert this solution to obtain $\eta$ as a function of $\xi$; to do that we introduce the expansion

$$\eta = \xi + b_1 \epsilon + b_2 \epsilon^2 + b_3 \epsilon^3 + \cdots,$$

where the coefficients $b_i, i = 1, 2, \ldots$ are functions of $\xi$ that must be calculated. Introducing this expansion in equation (13) and requiring that the different coefficients of each order match it is possible to obtain the values of $b_i, i = 1, 2, \ldots$. The result is

$$\eta = \xi - \frac{1}{6} \xi^3 \epsilon + \frac{1}{12} \xi^3 (\xi^2 + 2) \epsilon^2 - \frac{1}{360} \xi^5 (2 \xi^2 + 57) \epsilon^3 + \cdots.$$

\[ (14) \]

### 5.2 Intermediate-outer region

In the intermediate region we introduce the following change of variables:

$$x = \epsilon^\frac{3}{2} \chi, \quad y = \epsilon^\frac{1}{2} \sigma, \quad \text{with} \quad \chi, \sigma \approx O(1). \quad (15)$$

Note that $(\chi, \sigma)$ are both positive amounts. We look for a solution of equation (11) given by

$$\chi = a_0 + a_1 \epsilon + a_2 \epsilon^2 + a_3 \epsilon^3 + \cdots,$$

where the coefficients $a_i, i = 1, 2, \ldots$ can be obtained expanding appropriately the right-hand side of equation (11). The solution is

$$\chi = \frac{1}{6} \sigma^3 + \sigma - \frac{1}{120} \sigma^3 (\sigma^2 + 20) \epsilon + \frac{1}{5040} \sigma^5 (\sigma^2 + 42) \epsilon^3 + \cdots. \quad (16)$$

Let $\sigma_0 > 0$ be the real and positive solution of the cubic equation

$$f(\sigma) = \frac{1}{6} \sigma^3 + \sigma - \chi = 0 \Leftrightarrow f(\sigma) = \sigma^3 + 6 \sigma - 6 \chi = 0. \quad (17)$$

This root always exist; in effect, the function $f(\sigma)$ verifies

$$f(0) = -6 \chi < 0, \quad f(\chi) = \chi^3 > 0,$$

therefore in the interval $[0, \chi]$ there is a root of $f(\sigma) = 0$ at least. But there is only one root, because the derivative

$$f'(\sigma) = 3 \sigma^2 + 6 > 0$$

is always positive in that interval. That root is given by

$$\sigma_0 = S - \frac{2}{S}, \quad (18)$$

where $S = (\Lambda + 3 \chi)^{\frac{1}{3}}$ and $\Lambda = \sqrt{8 + 9 \chi^2}$. It should be noticed that expression (18) can present a rounding-error defect when $S \approx \sqrt{2} (\sigma_0 \ll 1)$. In order to minimize this error, we develop an alternative procedure to provide $\sigma_0$. We rewrite the cubic equation (17) as

$$\sigma (\sigma^2 + 6) - 6 \chi = 0 \Leftrightarrow \sigma = \frac{6 \chi}{\sigma^2 + 6},$$

such that $\sigma$ is replaced by equation (18) on the right-hand side of the equation, obtaining

$$\sigma_0 = \frac{6 \chi}{2 + S^2 + \frac{4}{S^2}}$$

which provides a smaller rounding-error when $\chi \ll 1$, i.e. when $\Lambda \approx 2 \sqrt{2}$. 

Once the value of $\sigma_0$ is known it is possible to invert the solution (16) to obtain $\sigma$ as a function of $\chi$; to do that we introduce the expansion

$$\sigma = \sigma_0 + b_1 \epsilon + b_2 \epsilon^2 + b_3 \epsilon^3 + \cdots,$$

where the coefficients $b_i$, $i = 1, 2, \ldots$ are functions of $\sigma_0$ that must be calculated. Introducing this expansion in equation (16) and requiring that the different coefficients of each order match it is possible to obtain the values of $b_i$, $i = 1, 2, \ldots$. The result is

$$\sigma = \sigma_0 + \frac{\sigma_0^2 (\sigma_0^2 + 20)}{60(\sigma_0^2 + 2)} \epsilon + \frac{\sigma_0^4 (\sigma_0^2 + 25\sigma_0^2 + 340\sigma_0^2 + 840)}{1400(\sigma_0^2 + 2)^3} \epsilon^2 + \cdots.$$  

(19)

**6 ANALYSIS OF RESULTS**

We performed an exhaustive numerical analysis of the algorithm that we propose in these pages. Two main aspects have been considered: (1) accuracy and (2) speed. There are additional considerations that must be taken into account like e.g. the reliability of the procedure, the ease of codification and so on. We start analysing the speed.

**6.1 Speed of the process**

The KE for the elliptical case has been solved with the method proposed in this paper by using the MNR algorithm. The KE has been solved $=4 \times 10^5$ times, using double and quadruple precision and a tolerance equal to $\epsilon_{\text{tol}} = 1.11 \times 10^{-15}$ and $1.0 \times 10^{-24}$, respectively. It should be noticed that when working with tolerances close to the zero of the machine, any numerical procedure based on consecutive approximations could be affected by artificial numerical chaos. By using an appropriate tolerance it is possible to escape from such a numerical chaos. In each run we count the number of iterations needed to reach a solution with a residual $\rho = |y - \epsilon \sin y - x|$ lower than the tolerance $\epsilon_{\text{tol}}$. This number is taken as a measure of the speed of the procedure.

Fig. 2 shows in the plane $(e, M)$ the number of iterations needed to reach a residual $\rho$ lower than the tolerance $\epsilon_{\text{tol}}$; this number is always lower or equal unity. Notice that in the colour code of the figure, white colour corresponds to one iteration and black colour to zero iterations. The green colour (two iterations) and red colour (three iterations) do not appear in the figure. The averaged value obtained in our calculations is 0.9843 iterations. The reliability of this code is remarkable because after many millions of runs, the algorithm never enters in dead loops or other kind of non-convergent phases.

Two different calculations have been carried out: (1) using standard double precision and (2) using quadruple precision. In both cases the results can be summarized in Fig. 2 because from the point of view of the number of iterations there are no differences in both calculations.

**6.2 Accuracy analysis**

In this section we focus the analysis on the SDG code with the iteration scheme provided by the MNR algorithm. In this code the iteration ends when the residual $\rho = |y - \epsilon \sin y - x|$ is lower that the zero of the machine $\epsilon_{\text{tol}} = 2.22 \times 10^{-16}$.

Let us consider the true solution $y_e(x, e)$ for given values of $e$ and the mean anomaly $M = x$. Let $y_e(x, e)$ the numerical solution provided by the SDG code for these particular values. Obviously, due to truncation and round-off errors it is possible to write

$$y_e = y_e + \varepsilon_{\text{abs}},$$

where $\varepsilon_{\text{abs}}$ is the total absolute error associated with the numerical solution $y_e(x, e)$. If the numerical solution $y_e(x, e)$ is a good approximation of the true solution, the value of $\varepsilon_{\text{abs}}$ is an infinitesimal: $\varepsilon_{\text{abs}} \ll 1$. In such a case, this absolute error is closely related with the residual $\rho$. In effect, for the numerical solution $y_e(x, e)$ the residual is given by

$$\rho = |y_e + \varepsilon_{\text{abs}} - e \sin(y_e + \varepsilon_{\text{abs}}) - x|.$$

Taking into account that the true solution verifies

$$y_e - e \sin y_e - x = 0,$$

the residual takes the form

$$\rho = \frac{\varepsilon_{\text{abs}}}{1 - e \cos y_e} + o(e^2).$$

As a consequence the total absolute error is given by

$$|\varepsilon_{\text{abs}}| = \frac{\rho}{1 - e \cos y_e}.$$  

(20)

Basically, it depends on the residual $\rho$, just like the total relative error

$$|\varepsilon_{\text{rel}}| = \frac{|y_e - y_e|}{|y_e|} = \frac{\rho}{|y_e| (1 - e \cos y_e)}.$$  

(21)

Let us fix the value of the eccentricity $e$; then we scan the whole interval $M \in [0, \pi]$ calculating the residual $\rho$ after zero iteration (the residual provided by the starting seed), after one iteration, two iterations and so on. Let $\rho_{\text{max}}$ the maximum residual that we found in the scanning of the whole interval $M \in [0, \pi]$ for each iteration number. These maximum residuals permit to calculate the maximum values of the absolute error $\varepsilon_{\text{max}}$ for different number of iterations, and these absolute errors can be associated with the value of the eccentricity $e$.

If we plot the values of such maximum absolute errors versus the eccentricity $e$ we have a very good idea of the accuracy involved in our calculations after zero, one, two, … iterations. Notice that these values mark the lowest accuracy of the numerical solution $y_e(x, e)$ provided by the SDG code; in fact, for each value of $e$ there
are hundreds of values of $M$ where the accuracy is much better than the indicated by $\varepsilon_{\text{max}}$.

Fig. 3 shows the decimal logarithm of the maximum absolute error versus the eccentricity $e$. The initial seed – 0 iterations – provides an error which, in the worst cases, ranges from $10^{-14}$ to $10^{-12}$. Since the calculations associated with Fig. 3 have been carried out in double precision, the residual obtained after one iteration practically saturates the capacity of the machine providing the solution with 15 or 16 significant figures. It should be underlined that one additional iteration does not improve the accuracy of the numerical solution; figure shows that the lines corresponding to one and two iterations are, practically, the same.

Fig. 4 is the same as Fig. 3 but with the calculations carried out in quadruple precision. It is clear that with one iteration we obtain more than 20 significant figures except for values of $e$ close to unity, where the number of significant digits is about 17–20. With two iterations the SDG code saturates the precision capacities of the machine, providing 34 significant figures.

6.3 Improving the code

Note that the number of polynomials used for the discretization of the interval $[0, \pi]$ of the mean anomaly $M = x$ does not affect to the speed of the algorithm but it has an important impact on the accuracy. Instead of 12 we modified the code to use more polynomials in order to obtain a better initial seed. However, the number of polynomial cannot be increased arbitrarily because the intervals $[x_i, x_{i+1}]$ becomes too small and the polynomials becomes numerically unstable. Thus, when we use 24 polynomials the two first polynomials $p_1(x)$ and $p_2(x)$ associated with the two first intervals $[x_1, x_2]$ and $[x_2, x_3]$ become numerically unstable for some values of the eccentricity $e$. If these two polynomials are replaced by the first polynomial of the 12 family, the SDG code runs without problems and the accuracy of the algorithm is improved in a significant way.

Fig. 5 is the same as Fig. 2 but now the calculations have been carried out using 23 polynomials. The figure does not change if quadruple precision is used instead of double precision. The speed of the code is not affected for the increase of polynomials. Obviously, by using double precision the code is faster than using quadruple precision.

Fig. 6 is the same as Fig. 3 but now 23 polynomials have been used to provide the initial seed. The initial seed – 0 iterations – provides an error which, in the worst cases, ranges from $10^{-7}$ to $10^{-15}$. Since the calculations associated with Fig. 6 have been carried out in double precision, the residual obtained after one iteration
practically saturates the capacity of the machine providing the solution with 15 or 16 significant figures. It should be underlined that one additional iteration does not improve the accuracy of the numerical solution; figure shows that the lines corresponding to one and two iterations are, practically, the same. By comparing Figs 3 and 6 a clear increase in the accuracy of the initial seed is detected, when the 23 polynomials family is used. However, there are no differences in the accuracy of the results after one iteration.

Fig. 7 is the same as Fig. 4 but now 23 polynomials have been used. Since the calculations associated with Fig. 7 have been carried out in quadruple precision, the residual obtained after one iteration practically saturates the capacity of the machine providing the solution with 34 significant figures except in the region $0.75 < e < 1$ where the number of significant digits ranges from 27 to 34. One additional iteration improves the accuracy of the numerical solution for all values of $e$.

7 COMPARISON OF RESULTS

In our calculations, instead of 12 approximating polynomials we use 23. In effect, after check the behaviour of the algorithm by using several number of polynomials we came to the conclusion that (1) the differences are not important when using 12, 16, 18 or 23 polynomials, and (2) by using 23 the algorithm behaviour is slightly smoother from a global point of view. All the calculations have been carried out in a workstation with Intel(R) Core(TM) i7 CPU 860 @ 2.80 GHz, 2.79 GHz microprocessor in a Windows 7 64 bits operative system and with the same Intel c++ compiler.

The Kepler equation for the elliptical case has been solved with the method proposed in this paper by using different Newton–Raphson algorithms. Moreover, the same analysis has been performed with typical solvers of the KE:

(i) the MNR;
(ii) the Conway method;
(iii) the CNR;
(iv) the method described in Gooding & Odell (1988);
(v) the method described in Nijenhuis (1991);
(vi) the method described in Fukushima (1996);
(vii) the method described in Mortari & Elipe (2014).

In each case, the KE has been solved as described in Section 6.1.

Fig. 5 shows, in the plane ($e, M$), the number of iterations associated with the SDG code when using the MNR iteration scheme. Notice that in the 98.737 per cent of cases we reach the solution with only one iteration (white colour)\(^1\) and in the 1.263 per cent no iteration is necessary (see Table 4). These results justify the option of work with quadruple precision which does not slow the calculations due to the very small number of iterations required. The CPU time invested in the $\approx 4 \times 10^6$ of times that we solved the KE was 41.012 s and the average number of iterations was 0.987. In the figure the zero iteration cases are localized in lines that, approximately, mark the extremes of the intervals associated with the approximating polynomials.

Table 4 shows as well the number of iterations associated with the SDG code when using the Conway and CNR methods. Notice that in small number of cases located close to the singular corner – the 0.106 and 0.138 per cent, respectively – two iterations are required. The CPU time invested was 40.779 s with Conway and 40.591 s with CNR, while the average number of iterations was 0.988 and 0.989, respectively. Moreover, Table 4 shows the number of iterations associated with the approaches described in Gooding & Odell (1988), Nijenhuis (1991), Fukushima (1996) and Mortari & Elipe (2014) when using double precision. Notice that all of them require more than three iterations for some particular cases, reaching a maximum value of 18, 6, 9 and 4 iterations, respectively. The CPU time invested was 41.2 s with Gooding code, 42.136 s with Nijenhuis code, 59.872 s with Fukushima code and 44.173 s with Mortari code. The average number of iterations was 2.205, 4.5, 3.495 and 2.284, respectively.

Table 5 is the same as Table 4 but now in quadruple precision. Notice that in this case the SDG code keeps again the smallest number of iterations. The methods described in Gooding & Odell (1988), Nijenhuis (1991) and Fukushima (1996) do not experiment any change, keeping the same percentages as in double precision. The maximum number of iterations reached was the same as in double precision excepting Gooding code with 25 iterations and Mortari code with 17 iterations. The CPU time invested was

\(^1\) The background white colour in the figures corresponds to cases in which only one iteration is needed to reach the solution with the prescribed tolerance.
Figure 8. Number of significant figures, approximately, obtained with the code of Gooding & Odell (1988) (upper), Nijenhuis (1991) (medium) and Fukushima (1996) (lower). On the left, double precision; on the right, quadruple precision.

83.190 s (MNR), 87.840 s (Conway) and 81.840 s (CNR) with SDG code, 64.335 s with Gooding code, 63.071 s with Nijenhuis code, 260.926 s with Fukushima code and 129.824 s with Mortari code. When quadruple precision is applied, the SDG code does not become the fastest algorithm, but provides the most precise solution with more than 30 significant digits in practically the whole plane \((e, M)\). Indeed, the Figs 8 and 9 show the number of significant digits obtained with the different codes when using the MNR method in
An efficient code to solve the Kepler equation

Figure 9. Number of significant figures, approximately, obtained with the Mortari & Elipe (2014) (upper) and SDG code (lower). On the left, double precision; on the right, quadruple precision.

double and quadruple precision. As it is shown, only the SDG code experiments and improvement of 20 significant figures for almost the complete plane when changing from double to quadruple precision. However, the results obtained with the remaining codes highlight the fact that they were developed to work in double precision but not in quadruple precision, especially in the case of Gooding, Nijenhuis and Fukushima codes. As a consequence, the results justify again the option to work with quadruple precision when using the SDG code.

8 APPLICATIONS OF THE SDG CODE

In the introduction of this paper (Wisdom & Holman 1991) we can read: long-term integrations are playing an increasingly important role in investigations in dynamical astronomy. The reason is twofold. First, numerical exploration is an essential tool in the study of complex dynamical systems which can exhibit chaotic behaviour, and there has been a growing realization of the importance of chaotic behaviour in dynamical astronomy.

In this sense, the code developed in this paper is very useful in many situations, especially in the numerical integration of perturbed problems which appears in dynamical astronomy. In particular, when the integration requires to solve the KE at every step, that means, a large number of times. Not only in Keplerian perturbed problems, but in more involved situations like the n-body problem.

We must underline that this method works regardless of whether or not the eccentricity maintains a constant value. In some cases, the eccentricity is constant, like, for example in the traditional Encke’s method (Encke & Schäfl 1854) where the reference orbit is a Keplerian one. The Encke’s method is used in some long-term integration, like the LONGSTOP project (Milani et al. 1986).

In other cases, the value of the eccentricity changes with time. This situation appears, for example, when the integration is performed with the Deprit’s method, based on a Hansen ideal frame (Deprit 1975, 1976). This integration scheme applies for elliptical orbits and requires to solve the KE in each integration step.

The KE must be solved as well in symplectic integration codes as for example in the Wisdom–Holman method (Wisdom &
where the orbital elements experiment small changes at every step.

To avoid the propagation of any kind of error involved in these numerical procedures it makes necessary to solve the KE with the highest possible accuracy. As we have shown, our algorithm, which can be used in double or quadruple precision, satisfies this condition assessing the convergence independent of the numerical method selected.

Finally, we want to highlight that in strong perturbed problems is quite easy to enter in the singular corner. Thanks to the asymptotic expansion, our code is very reliable and accurate and it avoids some chaotic behaviour of numerical nature that appears in that region with other solvers.

9 CONCLUSIONS

Several conclusions can be drawn from the results obtained of our numerical research.

(i) When the starting seed of the Newton–Raphson algorithm is very good, convergence is always assured as confirmed the thoroughly analysis carried out in our computers.

(ii) First of all we should underline the stability and reliability of our scheme combined with the Newton–Raphson algorithm in its different versions. Notice that the CNR provides the solution in the 98.599 per cent (double precision) of cases with only one iteration. This is remarkable result that we do not found in other algorithms used in the Astrodynamics community. Improving the algorithm by using the Conway or the MNR permits to obtain the right solution with only one iteration in the 98.631 and 98.737 per cent of cases, respectively.

(iii) The low number of iterations permits to use quadruple precision if you like, because the speed of the calculations is not jeopardized. Therefore, we obtain the benefit of a greater accuracy with a minimal cost.

(iv) The global algorithm solves successfully the solution of the KE in the singular corner, $M \ll 1$ and $e \approx 1$. The asymptotic expansions used to generate the initial seed assure the reliability and convergence of the Newton–Raphson iterative scheme. This is another remarkable results since the convergence in this special region not always is assured by some of the algorithms usually considered in the literature.

(v) When quadruple precision is used in numerical simulations, the absolute error of the numerical solution provided by the SDG code is clearly under the $10^{-25}$ after one iteration. In any case, after two iterations the error is always lower than $10^{-30}$, including the region $e > 0.75$ (the singular corner).

ACKNOWLEDGEMENTS

This work was carried out in the framework of the research project entitled Dynamic Analysis, Advanced Orbital Propagation and Simulation of Complex Space Systems (ESP2013-41634-P) supported by the State Secretariat for Research, Development and Innovation of the Spanish Ministry of Economy and Competitiveness. Authors thank to Professor T. Fukushima for his support codifying the algorithm of his paper (Fukushima 1996).

REFERENCES

Battin R., 1999, An Introduction to the Mathematics and Methods of Astrodynamics, revised edn. AIAA, Reston, VA
Colwell P., 1993, Solving Kepler’s Equation over Three Centuries. Willmann-Bell, Richmond, VA
Serafin R., 1986, Celest. Mech., 38, 111
Siewert C., Burniston E., 1972, Celest. Mech., 6, 294

APPENDIX A: THE FIFTH DEGREE POLYNOMIAL $p_1(x, e)$

The polynomial $p_1(x, e)$ is given by the expression

$$p_1(x, e) = a_0^{(1)}(e) + a_1^{(1)}(e)x + a_2^{(1)}(e)x^2 + a_3^{(1)}(e)x^3 + a_4^{(1)}(e)x^4 + a_5^{(1)}(e)x^5,$$

where the coefficients are defined as a function of the eccentricity by

$$a_0^{(1)}(e) = 4320 \frac{e}{(0.5393817e^3 - 0.8743466e^2 + 0.2650564e + 0.0453931)},$$

$$a_1^{(1)}(e) = \frac{720 \frac{e}{(0.4657683e^3 - 0.402960e^2 - 0.125744e - 0.002438)}},$$

$$a_2^{(1)}(e) = \frac{12.0 \left(2.1335977e^3 - 7.174617e^2 + 8.438043e - 3.4460552\right)}{d_4^{(1)}(e)},$$

$$a_3^{(1)}(e) = 0.$$
An efficient code to solve the Kepler equation

\[ a_1^{(1)}(e) = \frac{-1.0}{-1.0 + e}, \]

\[ a_0^{(1)}(e) = 0, \]

with

\[ d_1^{(1)}(e) = (3739.002236 e^3 - 2549.554211 e^4 + 926.7532457 e^5
\]
\[ -140.2961154 e^6 - 3082.919872 e^2 + 1355.064876 e - 248.0502135)
\]
\[ (-1.0 + e) \left(9.0 e^2 - 18.84955592 e + 9.869604404\right), \]

\[ d_2^{(1)}(e) = (3.141592654 - 3.0 e) \left(231.3573259 e - 323.4979098 e^2
\right.
\[ + 200.9177486 e^3 - 46.76537182 e^4 - 62.01255338)
\[ \left(-1.0 + e\right) \left(6.928203232 e - 3.0 e^2 - 4.0\right), \]

\[ d_3^{(1)}(e) = \left(-27.0 e^3 + 84.82300166 e^2 - 88.82643964 e + 31.00627669\right)
\]
\[ \times \left(5.196152424 e^3 - 18.0 e^2 + 20.78460970 e - 8.0\right) (-1.0 + e). \]

This paper has been typeset from a TeX/LaTeX file prepared by the author.