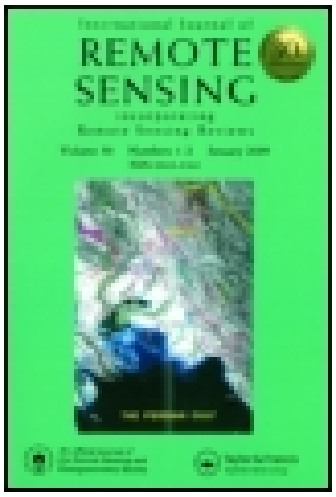
This article was downloaded by: [University North Carolina - Chapel Hill]
On: 06 December 2014, At: 09:02
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Remote Sensing
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tres20

Technical Note Kepler's equations in 'C'

P. S. CRAWFORD [a]

[a] Department of Applied Physics and Electronic and Mechanical Engineering , University of Dundee , Dundee, Scotland, DD1 4HN, U K

Published online: 16 May 2007.

To cite this article: P. S. CRAWFORD (1995) Technical Note Kepler's equations in 'C', International Journal of Remote Sensing, 16:3, 549-557, DOI: 10.1080/01431169508954418

To link to this article: http://dx.doi.org/10.1080/01431169508954418

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions

INT. J. REMOTE SENSING, 1995, VOL. 16, NO. 3, 549-557

Technical Note

Kepler's equations in 'C'

P. S. CRAWFORD

Department of Applied Physics and Electronic and Mechanical Engineering, University of Dundee, Dundee DD1 4HN, Scotland, U K.

(Received 22 July 1994; in final form 7 October 1994)

Abstract. A complete method of solving Kepler's equations is presented that behaves as an analytical function. The derivative term is evaluated and an efficient and robust implementation in the 'C' language is provided in the Appendix. Simulation results illustrate the computational requirements over a range of orbital eccentricities.

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

1. Introduction

The traditional methods of computing satellite orbital problems have used various forms of solutions to Kepler's equations hard-coded into the main programmes, in FORTRAN style. However, the modern approach is for modular solutions, the slight difference in execution speed is usually a minor concern compared to the improvement in software maintainability. (In fact, depending upon the programmer's skill, optimizing compiler and application, it may be a beneficial change!)

Although it could be said that too much has been published on the subject of Kepler's equations for satellite orbits, few result in a piece of software that satisfies all of the following objectives:

1. The function is fast and efficient in evaluation.
2. The accuracy is strictly bounded by some known parameter.
3. There are no singularities to cause machine failure during computation.
4. The output follows the input in a smooth analytical manner, i.e., there are no sudden jumps of 2π radians or similar even when the input exceeds 2π radians.
5. When required, derivative terms are available for angular rate determination and Newton-Raphson root solving for larger, problems in elliptical orbits, such as the determination of satellite equator crossing time.
6. The solution is in a robust and portable form which can be recompiled on most computer systems without difficulty.

Here a concise method is presented which can be used as part of a solution to most problems involving satellite orbits, such as antenna look angle generation or image navigation.

2. Kepler's equations

Kepler's equations relate the three angles used in describing the elliptical orbit of a satellite or planet around one of its foci (an approximation to the Earth). These are known as:

0143-1161/95 $10.00 © 1995 Taylor & Francis Ltd

Mean anomaly M

This is the angle from perigree (the point of closest approach to the foci) for a fictitious satellite in a uniform circular orbit. In practical terms, this is time moving uniformly through one anomalistic period as the satellite moves 2π radians from one perigee to the next.

Eccentric anomaly E

This is an intermediate term that relates to the radial distance between the satellite and the focus.

True anomaly T

This is the angle traversed by the satellite measured from perigee. Although this angle reaches π and 2π radians at the same time as M, at other times the angles are different. This angle represents the true position of the satellite on its elliptical orbit.

An additional parameter which relates these three angles is:

Eccentricity e

The eccentricity is a measure of how elliptical the orbit is. A value of 0 represents a circular orbit and those approaching 1 represent long thin ellipses.

To provide a numerical solution to the problem of solving Kepler's equations, we can look at these equations with a view to coding them in a high level language in such a manner that all of these points are addressed.

The equations that relate these different angles can be expressed in different ways (Taff 1985, Meeus 1988), but the choice used here is based upon the following:

$$M = E - e \sin(E) \tag{1}$$

$$\tan\left(\frac{T}{2}\right) = \sqrt{\frac{1+e}{1-e}} \tan\left(\frac{E}{2}\right) \tag{2}$$

Equation (2) can be solved both ways by arranging as:

$$T = 2 \tan^{-1}\left[\sqrt{\frac{1+e}{1-e}} \tan\left(\frac{E}{2}\right)\right] \tag{3}$$

$$E = 2 \tan^{-1}\left[\sqrt{\frac{1-e}{1+e}} \tan\left(\frac{T}{2}\right)\right] \tag{4}$$

At first sight, the use of the tangent function is dangerous for computation, however, we shall code the tangent as separate sine and cosine calls and replace the arctangent function with the atan2 rectangular to polar conversion function. This results in a singularity free function for angles up to (but not including) $\pm 2\pi$ radians. This can be extended by reducing the input value modulo 2π and then restoring the remainder after solving the other terms, although a check should be made to ensure the 2π of the reduction is exactly the same as the value understood by the trigonometric functions in use.

Another important point is the term in the eccentricity. Since e is often constant for a large number of evaluations of Kepler's equation, it makes sense to compute it as an initialization call and keep a local copy as a constant in evaluating (3) or (4).

The solution of (1) is simple, however, reversing this function results in the implicit equation:

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

$$E = M + e \sin(E) \tag{5}$$

Although a considerable amount of effort has been expended on equation (5) (see Taff 1985 for most of the details), there is no analytical solution. It is possible to expand (5) in a series (Taff 1985) using powers of e and trigonometric terms in multiples of E; however, evaluating this is not computationally efficient and the truncation error is difficult to establish at higher values of e (which fails point 2 of the objectives set out above). By far the best method of solving (5) is that of Newton-Raphson root finding. In this case we re-write (5) into a function $f(E)$ that we set to zero, and find the derivative of the function:

$$f(E) = M + e \sin(E) - E = 0 \tag{6}$$

$$f'(E) = e \cos(E) - 1 \tag{7}$$

Equation (6) is then solved iteratively using the Newton-Raphson formula

$$E_{N+1} = E_N - \frac{f(E_N)}{f'(E_N)} \tag{8}$$

Where E_N is the estimate of E at iteration number N. The reason that the Newton-Raphson method is so good in this application relates to the quadratic convergence property near a root of $f(E)$ when the function is smooth and well behaved as in this application; in this case each iteration approximately doubles the number of significant figures, for example, the error term may progress something like 10^{-3}, 10^{-6}, 10^{-12} as we iterate. Of course, this is of most benefit if the evaluation of the derivative function is easy, as in this case.

The Newton-Raphson process is very good near the root but has a danger of stepping off to infinity if care is not taken, see Flannery *et al.* (1992) for an excellent discussion. For (6), this usually occurs near perigee with high eccentricities. In the case of (5) we can see that the maximum difference is bounded by the eccentricity since $\sin(E)$ is less than 1 in magnitude, that is:

$$|E - M| \leq e \tag{9}$$

So before we perform the Newton-Raphson division and correction, we should check that the intended correction is no greater than e.

There has been some discussion over what starting value of E to use in the iterative process. The usual value is $E = M$, but others have been suggested by Taff (1985); however, the bounding by e system used here works as rapidly in solving the problem illustrated in (Taff 1985, p. 55) and without any (computationally expensive) trigonometric function calls to generate the initial value.

If the type of Kepler problem involves the repeated evaluation with small changes in M for each call, it is possible to save the previous difference $(E - M)$ as a correction to the new M to get the initial value for E. It is important to zero the initializer in this scheme each time a new eccentricity is used, otherwise the iteration limit may be reached. In some cases (particularly for higher eccentricities and small increments) there can be a significant reduction in the average number of iterations; however, if the input changes frequently by a large amount, or the eccentricity changes frequently, then the $E = M$ first value is good enough and recommended for a general purpose Kepler solving object.

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

A final thought concerns the evaluation of the derivative of T with respect to M and vice-versa for the determination of the angular rate or for the derivative term in Newton-Raphson functions involving elliptical orbits. If we differentiate using the chain rule, etc., we get the following expressions:

$$\frac{dE}{dT}=\frac{\sqrt{\frac{1-e}{1+e}}}{\cos^2\left(\frac{T}{2}\right)+\frac{1-e}{1+e}\sin^2\left(\frac{T}{2}\right)} \tag{10}$$

$$\frac{dT}{dE}=\frac{\sqrt{\frac{1+e}{1-e}}}{\cos^2\left(\frac{E}{2}\right)+\frac{1+e}{1-e}\sin^2\left(\frac{E}{2}\right)} \tag{11}$$

$$\frac{dE}{dM}=\frac{1}{1-e\cos(E)} \tag{12}$$

$$\frac{dM}{dE}=1-e\cos(E) \tag{13}$$

Although these look formidable, most of the terms are used elsewhere in the evaluation and so the addition of the derivative term is not very expensive in computational terms.

3. 'C' code implementation

The source code is listed in the Appendix. The coding of these equations into the 'C' high level language at Dundee has been performed during the conversion of some older programmes (generally written in FORTRAN) to 'C' for use on Sun Microsystems' workstations. We do not suggest that 'C' is any better than FORTRAN for numerical work (see Flannery *et al.* (1992) for an extensive discussion on this subject) but that the majority of new applications are for 'C' implementation and mixed-language programmes can be difficult to maintain or support over differing systems.

For some processors, such as the Intel 80387 and 486DX, the floating point hardware has an instruction to evaluate simultaneously the sine and cosine of an argument since much of the time spent in evaluating one is common to the other. For the Sun Microsystems' 'C' compiler, there is the function:

```
void sincos (double x, double *s, double *c);
```

However not all compilers have this feature (such as the Microsoft C6.0 that we have used for IBM-PC programming). In these cases, we can change the compilation as required using the 'C' pre-processor.

At Dundee, we have used the Sun's defined constant M_PI in ⟨math.h⟩ to fix this automatically for both IBM-PC and Sun compilation. An example of this is shown in the Appendix where the `sincos (x,sx,cx)` macros is used when the M_PI constant is not found. If execution time is important and one is using a PC with an 80387 or higher coprocessor, writing an assembly language function using the FSINCOS instruction can be over 30 per cent faster than the standard library equivalent. Since virtually every new IBM-PC used for numerical work is equipped

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

with an 80387 or 486DX, we now use the assembly language option for IBM-PC programmes.

An additional feature that FORTRAN has but 'C' lacks is sign transfer, however the **SIGN (a,b)** macro will accommodate this as well.

The parameter EPS is the maximum error in radians for the final iteration. Remember that when reaching this value for the final iteration, the actual error in f(E) is approximately $(EPS)^2$ which for $EPS = 10^{-8}$ is around the machine precision for the **double** type so there is little point in iterating further unless either:

1. The function is used for numerical differencing (or similar questionable practices) and requires definite accuracy to machine precision.
2. The derivative term is required to near machine precision (at exit, the error in the derivative which uses the existing evaluation is around EPS).

In any case, the smallest value for EPS should be 2π times the machine precision, i.e., about $1{\cdot}4\ 10^{-15}$ in most cases.

The initializing of the eccentricity terms is performed by a call to the **init_kepler ( )** function and sets up the global variables (already initialized at run-time to safe sensible values). If desired, a check that e is between 0·0 and 1·0 could be performed in this function.

Next are the functions **fkepl ( )** for the *forward* solution of converting true anomaly into mean anomaly and **ikepl ( )** for the *inverse* problem of converting mean anomaly into true anomaly. In both cases the derivative is available and computed only if required, the passing of a NULL pointer will result in an early return when derivatives are not required.

The limit value for the Newton-Raphson correction is generated using the operation **−SIGN (ecc,F)** rather than the more obvious operation **SIGN (ecc,F*DF)** since **DF** is always negative for any valid eccentricity ($0 \leqslant e < 1$) and the floating point sign change is faster than a multiply on most machines (especially when an optimizing compiler has processed the SIGN macro used).

4. Simulation and testing

The efficiency of the Newton-Raphson method for solving this problem is illustrated by the graph of figure 1. This shows the number of iterations required (both the worst case and average) for a range of eccentricity from 0·001 to 0·95 evaluated over 4096 mean anomaly points between 0 and 2π radians. This used $EPS = 10^{-8}$ for the tolerance.

To verify that the software works as expected, the following values can be used from the example in (Taff 1985, p. 55):

$$e = 0{\cdot}995$$

$$M = 0{\cdot}1 \text{ (radian)}$$

After a call to **init_kepler ( )** and then **ikepl ( )** the values should be:

$$E = 0{\cdot}842731 \text{ (radian)}$$

$$T = 2{\cdot}919126 \text{ (radian)}$$

$$\text{dTdm} = 0{\cdot}874742$$

Feeding T back into **fkepl ( )** should result in the original M (to better than EPS) and dMdT should be 1·0/dTdM.

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

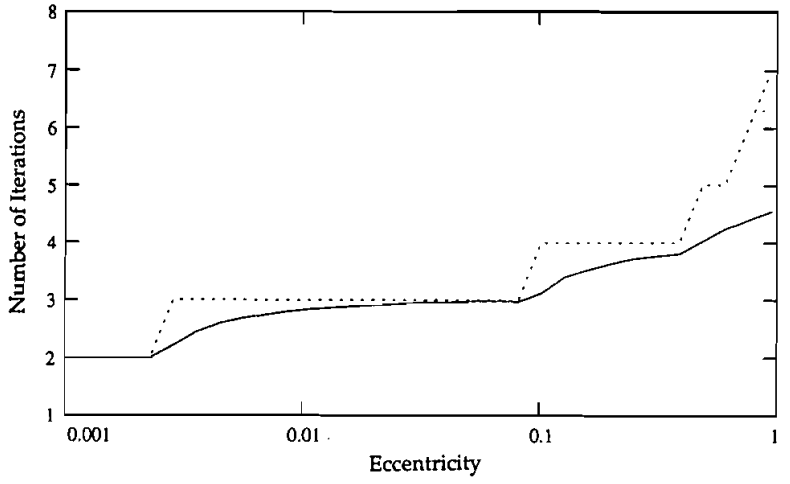


Figure 1. Number of iterations against eccentricity. —— Average interactions, --- peak interactions.

5. Conclusions

The solution of general problems in elliptical motion is eased if the solution of Kepler's equations can be relied upon to work under all reasonable conditions. We believe that this implementation of Kepler's equations achieves the original objectives outlined in the introduction in a satisfactory manner.

Appendix

This is the source code listing (with additional comments and explanations) for the 'C' implementation of Kepler's equations. A copy of this in computer readable form can be obtained by electronic mail request from psc@uk.ac.ndu.ua.

```
/* ================ File kepler.c ==================== */

#include <math.h>

#include <stdlib.h> For definition of NULL

#if defined(M_PI)

#define PI M_PI

#else

#define sincos(x, sx, cx)\

{dtmp=(x); *(sx)=sin(dtmp); *(cx)=cos(dtmp);}

#define PI 3.141592653589793238

#endif

#define TWOPI (2.0*PI)

#define EPS 1.0e-8
```

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

```
#define ITMAX 20
#define SIGN(a, b) ((b) >= 0.0 ? fabs(a) : -fabs(a))
/* ======= Function Prototypes ======================== */
void init_kepler(double);
void fkepl(double, double *, double *, double *);
void ikepl(double, double *, double *, double *);
/* ======= Global Variables =========================== */
static double ecc=0, rtpeme=1, irtpeme=1;
/* ==================================================== */
void init_kepler(double NewEcc)
{
      ecc = NewEcc; Compute 'e' terms in equations (3) and (4).
      rtpeme = sqrt((1.0+ecc)/(1.0-ecc));
      irtpeme = 1.0/rtpeme;
}
/* ==================================================== */
void fkepl(double trunom, double *ecanom, double *mnanom,
double *dMdT)
{
double T, E, baserev, KsinT2, cosT2, cosE, sinE, dEdT;
double dtmp; For the sincos macro if used
      T = fmod(trunom, TWOPI); Reduce modulo 2π
      baserev = trunom - T; Get the 2π remainder
      sincos(0.5*T, &KsinT2, &cosT2); Eqn (4)
      KsinT2 *= irtpeme;
      E = 2.0 * atan2(KsinT2, cosT2);
      *ecanom = baserev + E; Restore the 2π remainder
      if(dMdT == NULL){ Derivative required ?
            *mnanom = baserev + (E - ecc*sin(E)); No
      } else {
```

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

```
            sincos(E, &sinE, &cosE); Need both terms of E
            *mnanom = baserev + (E - ecc*sinE); Eqn (5)
            dEdT = irtpeme/(cosT2*cosT2 + KsinT2*KsinT2);
            *dMdT = (1.0 - ecc*cosE)*dEdT; Eqn (10) and (13)
        }
    }
/* ================================================ */
 void ikepl(double mnanom, double *ecanom, double *trunom,
 double *dTdM)
 {
 double M, T, E, cosE, sinE, baserev, KsinE2, cosE2;
 double F, DF, DE, dtmp;
 int jj;
        E = M = fmod(mnanom, TWOPI); Initialise E as well
        baserev = mnanom - M;
        for(jj=0; jj < ITMAX; jj++) {
            sincos(E, &sinE, &cosE);
            F = M + ecc*sinE - E; Function to zero
            DF = ecc*cosE - 1.0; Derivative function
            if(fabs(F) >= fabs(ecc*DF)) { Bound exceeded ?
                DE = -SIGN(ecc, F); Yes, limit to ecc
            } else {
                DE = F/DF; No, generate Newton-Raphson correction
            }
            E = E - DE; Apply correction, eqn (8)
            if(fabs(DE) < EPS) break; Good enough ?
        }
        sincos(0.5*E, &KsinE2, &cosE2);
        KsinE2 *= rtpeme;
        T = 2.0 * atan2(KsinE2, cosE2); Eqn (3)
        *ecanom = baserev + E;
```

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014

```
        *trunom = baserev + T;

        if(dTdM == NULL) return; Apply eqn (11) and (12) if required ?

        *dTdM = rtpeme/((KsinE2*KsinE2 + cosE2*cosE2)*(-DF));

}

/* ================ End of File ====================== */
```

References

FLANNERY, B., PRESS, W., TEUKOLSKY, S., and VETTERING, W., 1992, *Numerical Recipes in C*, 2nd Edition (Cambridge: Cambridge University Press).

MEEUS, J., 1988, *Astronomial Formulae for Calculators*, 4th Edition (Virginia: Willmann-Bell Inc.).

TAFF, L., 1985, *Celestial Mechanics* (Chichester: John Wiley & Sons).

Downloaded by [University North Carolina - Chapel Hill] at 09:02 06 December 2014