SECTION IV

DYNAMICS OF COMETS: NUMERICAL MODELLING

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

AN EFFICIENT INTEGRATOR THAT USES GAUSS-RADAU SPACINGS

Edgar Everhart
Physics Department and Chamberlin Observatory
University of Denver
Denver, Colorado 80208 USA

ABSTRACT. This describes our integrator RADAU, which has been used by several groups in the U.S.A., in Italy, and in the U.S.S.R. over the past 10 years in the numerical integration of orbits and other problems involving numerical solution of systems of ordinary differential equations. First- and second-order equations are solved directly, including the general second-order case. A self-starting integrator, RADAU proceeds by sequences within which the substeps are taken at Gauss-Radau spacings. This allows rather high orders of accuracy with relatively few function evaluations. After the first sequence the information from previous sequences is used to improve the accuracy. The integrator itself chooses the next sequence size. When a 64-bit double word is available in double precision, a 15th-order version is often appropriate, and the FORTRAN code for this case is included here. RADAU is at least comparable with the best of other integrators in speed and accuracy, and it is often superior, particularly at high accuracies.

1. INTRODUCTION

The first description of RADAU was by Everhart (1974a). A more full treatment is in an unpublished technical report, Everhart (1974b), which has had a wide distribution, and there have been improvements suggested by experience in the past 10 years. Our method is related to the implicit Runge-Kutta procedures of Butcher (1964), which also use Gauss-Radau spacings, but the algorithms of the two methods are quite different. RADAU can easily be written to a very high order of accuracy. In special tests we have used it in 31st order.

It has been used sucessfully in determining the original orbits for some 224 comets, Marsden, Sekanina, and Everhart (1978), Everhart and Marsden (1983). Starting with the comet in its accurate osculating orbit near perihelion, they integrated in 15th order backwards through the solar system, taking into account the perturbations of every planet on every other planet and on the comet. It was a 10-body integration near the sun, with 30 simultaneous second-order equations. As the comet moved outward in this barycentric integration, the masses of the inner planets were one by one added to the sun's mass until at large distances only

A. Carusi and G. B. Valsecchi (eds.), Dynamics of Comets: Their Origin and Evolution, 185–202.

the 5 outer planets and the comet were included. This 6-body integration was continued until the comet was 60 AU away from the sun, at which place the original barycentric orbit was determined. The formal accuracy was far more than was justified by the accuracy of the starting elements. Such effects as the close encounters of Comet Bowell with both Saturn and Jupiter were accurately handled. The perturbations of Saturn on Jupiter, which changed the position of Jupiter slightly, made a significant difference on the computed original orbit of this comet.

The results of the first paper on this, Everhart (1974a), showed the enormous advantages of using Gauss-Radau spacings. In 1973 we were developing an integrator for the equation $y'' = F(y,t)$ for applications in celestial mechanics. We expanded y" in an empirical time series through terms in t^3, and the corresponding expansion for y was to t^5. The idea was to examine a sequence of overall length T. With uniform spacing of substeps it was evident that the force function F would have to be evaluated at 0, .3333T, .6667T, and at T. This did result in a 5th-order integrator, with errors of the order of T^6 in y. Thus if one cut the sequence size in half one should expect to see 1/64th the error. Then some rather complicated algebra suggested that if the substeps were taken at 0., 0.21234T, 0.59053T, and 0.91441T, then the integrator would be 7th order, instead. When this was tried the results were astonishing. Errors dropped by a factor of 100 at the same number of function calls. It was now a 7th-order integrator. Soon thereafter, this was discussed with W. H. Goodyear, who suggested that Gaussian spacings had just been re-invented. It turned out that the spacings found here were those used in Gauss-Radau quadratures. They were camouflaged a bit because for quadratures one uses the range -1 to +1, instead of 0 to +1 as here. The special spacings of Gaussian quadratures allow one to get almost double the order of accuracy that one gets with constant spacings, comparisons being made at the same number of function calls. The present integrator extends the advantages of Gaussian spacings to the integration of differential equations.

The detailed example of the next section describes an expansion that would give 6th-order results at constant spacings, but which is a 9th-order integrator with Gauss-Radau spacings. In our most practical version we use a time expansion through t^9 in y (and through t^7 in y"), achieving 15th-order accuracy with these spacings.

The present method solves simultaneous differential equations of first or second order directly. We define several classes:

Class I	$y' = F(y,t)$
Class IIS (special)	$y'' = F(y,t)$
Class II (general)	$y'' = F(y',y,t)$

The method can be extended also to solve general 3rd- or 4th-order differential equations directly. The theory is given for solving one equation. Extension to any number of such simultaneous equations involves simple loops as seen in the examples of Sec. 4.

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

2. THE ALGORITHM

We illustrate the method in solving the general class II equation, $y'' = F(y',y,t)$, in 9th order of accuracy.

2.1 Expansions

Since y' and y are implicit functions of the independent variable t, one may expand y" (which is F) as a truncated series,

$$y'' = F = F_1 + A_1 t + A_2 t^2 + A_3 t^3 + A_4 t^4, \tag{1}$$

about the initial value $t_1 = 0$, where F has the value F_1. At the end of a sequence of length T the problem is to find y(T). Equation (1) is not a Maclaurin series expansion in t where the error is comparable with the first term neglected. With Gaussian spacings the error in the truncated series can be a millionth the size of the last term included.

Using $h = t/T$ and $B_1 = A_1 T$, $B_2 = A_2 T^2$, $B_3 = A_3 T^3 \ldots$, one has

$$y'' = F = F_1 + B_1 h + B_2 h^2 + B_3 h^3 + B_4 h^4. \tag{2}$$

After F_1 is found at $h_1 = 0$, other values $F_2, \ldots, F_5$ are developed at suitable spacings $h_2, \ldots, h_5$ within the interval of h between 0 and 1. We expand F in a way that incorporates these spacings h_n, introducing a set of coefficients G and writing

$$F(h) = F_1 + G_1 h + \\ + G_2 h(h-h_2) + G_3 h(h-h_2)(h-h_3) + G_4 h(h-h_2)(h-h_3)(h-h_4) . \tag{3}$$

This truncates at each location h_n. Thus $F_3 = F_1 + G_1 h_3 + G_2 h_3 (h_3 - h_2)$. Abbreviating with $r_{nj} = 1/(h_n - h_j)$ and $r_{n1} = 1/h_n$ one finds

$$\begin{aligned} G_1 &= (F_2-F_1)r_{21} , \\ G_2 &= ((F_3-F_1)r_{31} - G_1)r_{32} , \\ G_3 &= (((F_4-F_1)r_{41} - G_1)r_{42} - G_2)r_{43} , \\ G_4 &= ((((F_5-F_1)r_{51} - G_1)r_{52} - G_2)r_{53} - G_3)r_{54} . \end{aligned} \tag{4}$$

The B's and G's are related through Eqs. (2) and (3). This gives

$$\begin{aligned} B_1 &= c_{11}G_1 + c_{21}G_2 + c_{31}G_3 + \ldots , \\ B_2 &= \quad\; c_{22}G_2 + c_{32}G_3 + \ldots , \\ B_3 &= \quad\quad\; c_{33}G_3 + \ldots , \end{aligned} \tag{5}$$

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

where the recurrence relationships for the c's are

$$c_{jj} = 1 \quad ,$$
$$c_{j1} = -h_j c_{j-1,1} \quad , \qquad j > 1 \quad , \tag{6}$$
$$c_{jk} = c_{j-1,k-1} - h_j c_{j-1,k} \quad , \quad k < j \quad .$$

The reverse relationships are also needed. These are

$$G_1 = d_{11}B_1 + d_{21}B_2 + d_{31}B_3 + \cdots \quad ,$$
$$G_2 = \qquad d_{22}B_2 + d_{32}B_3 + \cdots \quad , \tag{7}$$
$$G_3 = \qquad\qquad d_{33}B_3 + \cdots \quad ,$$

with the recurrence relationships

$$d_{jj} = 1 \quad ,$$
$$d_{j1} = h_2 d_{j-1,1} = h_2^{j-1} \quad , \quad j > 1 \quad , \tag{8}$$
$$d_{jk} = d_{j-1,k-1} + h_{k+1} d_{j-1,k} \quad , \quad k < j \quad .$$

In this dimensionless form the c-, d-, and r-values do not depend on the sequence length T and are calculated but once for a given integration order. Evaluating these constants uses simple loops and takes much less than one second.

2.2 Predictors and Correctors

The predictors $y_n(h_n)$ and $y'_n(h_n)$ at each substep n within a sequence are found by integrating Eqs. (1) or (2). These are

$$y_n = y_1 + y'_1 h_n T +$$
$$+ h_n^2 T^2 (F_1/2 + h_n(B_1/6 + h_n(B_2/12 + h_n(B_3/20 + h_n B_4/30)))) \quad , \tag{9}$$
$$y'_n = y'_1 +$$
$$+ h_n T (F_1 + h_n(B_1/2 + h_n(B_2/3 + h_n(B_3/4 + h_n B_4/5)))) \quad . \tag{10}$$

At the end of the sequence, where h = 1 and t = T, the correctors are

$$y(T) = y_1 + y'_1 T +$$
$$+ T^2 (F_1/2 + B_1/6 + B_2/12 + B_3/20 + B_4/30) \quad , \tag{11}$$
$$y'(T) = y'_1 + T(F_1 + B_1/2 + B_2/3 + B_3/4 + B_4/5) \quad . \tag{12}$$

The system is implicit; the B-values are not known when they are first needed.

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

2.3 Procedure

After the first sequence, fairly good values of the B-coefficients are predicted for the current sequence (as will be described in Sec. 2.4 below). The corresponding G-values are known through Eqs. (7). Using the Radau spacings, the integrator steps through the sequence. At each step the position (and velocity if necessary) is obtained from Eqs. (9), (10) using the B-values. Then the force is calculated at the predicted position, and each new force value determines an improved G-value as in Eqs. (4). Every B-value that depends on this G-value is immediately upgraded using Eqs. (5). Essentially, the B-values determine the predicted positions, the forces at these positions determine the G-values, and these determine better B-values. After stepping through a sequence once, the integrator has considerably improved B- and G-values. A second pass refines these to high accuracy. The procedure becomes very clear when one follows through the listing of the 15th-order integrator in Sec. 4.

On initial starting, the initial B-values are all zero, and the predicted positions are inaccurate. Nonetheless, the process converges to high accuracy with enough iterations. It is worthwhile to take 6 iterations to start the 15th-order integrator and 10 iterations to start in 27th order.

2.4 Prediction of the B-values for the Next Sequence

Convergence is speeded greatly by using past information, and after the first sequence only 2 passes are required. Let $B_1, \ldots, B_4$ be accurate values from the previous sequence, and let $Q = T(\text{new})/T(\text{old}) = T'/T$ be the ratio of the sequence sizes. An analytic continuation of the curve for F from one sequence to the next (with a change in origin and this change in scale) requires that

$$\begin{aligned} B_1(\text{new}) &= Q\,(B_1 + 2B_2 + 3B_3 + 4B_4)\ , \\ B_2(\text{new}) &= \quad Q^2(B_2 + 3B_3 + 6B_4)\ , \\ B_3(\text{new}) &= \quad\quad Q^3(B_3 + 4B_4)\ , \\ B_4(\text{new}) &= \quad\quad\quad Q^4\,B_4\ . \end{aligned} \tag{13}$$

These are excellent values with which to start the new sequence, especially with the refinement discussed in the next section. Equations (13) can easily be extended to any order by noting the pattern of binomial coefficients in the columns. Thus, if there were a B_5 term, the constants in the added column then are 1, 5, 10, 10, 5 from bottom to top.

2.5 Refinements

In the technical report, Everhart (1974b), we advocating absorbing the constants 1/6, 1/12, 1/20, etc, that appear in Eqs. (9) and (11) into

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

the B-values. This can be done, and it does speed the computation some 10-20%, but it has the disadvantage of changing all the above equations and making the program difficult to check. We have discarded this idea.

Two other improvements are incorporated: First, we have found in some quite recent tests that if the equations are written out without loops then the integration is 25% to 30% faster! This is entirely practical through 15th order, as listed in the Sec. 4.

A second improvement has been incorporated. If one saves the values of B(new) predicted by Eqs. (13) and compares them later with the final B-values of the sequence, the difference is slowly-varying, and can be applied in advance as a correction. This simple change in the algorithm cuts the global error by a factor of 2 to 10 and is quite worthwhile.

2.6 Gauss-Radau, Gauss-Lobatto, and Gauss-Legendre Spacings

The use of Gaussian spacings for the substeps enhances the integration order for Class I, IIS, and II equations alike. For quadratures one ordinarily uses Gauss-Legendre spacings, but for solving differential equations with an implicit algorithm there is reason to use the Gauss-Radau or Gauss-Lobatto spacings. They use F_1 at $h_1=0$, and F_1 is not recalculated in the iteration.

Radau and Lobatto spacings may be found to 30 significant digits in Tables 12 and 11 of Stroud and Secrest (1966) to extremely high orders. These span the range -1 to +1, and they must be rescaled to the range 0 to +1 for the present application. The spacings so rescaled may be found in Table I of our 1974 paper for orders through 15th. There is a typographical error in h_2 for 15th order. The correct number is in the listing in Sec. 4. The reason Radau spacings, which give odd orders, are preferred to Lobatto spacings is that the order is one higher for a given number of terms. Thus Eq.(2) with terms through 4th order in time gives 9th order with Radau spacings and 8th order with Lobatto spacings.

2.7 Sequence Size Control

To control T one can monitor the last term in the y-expansion, here $B_4T^2/30$. This term is $B_4'T'^2/30$ for the next sequence (primed values), which will be controlled to have magnitude 10^{-L}. Now $B_4' = (T'/T)^4B_4$, as in the last of Eqs. (13), so the condition on T' is

$$10^{-L} = B_4'T'^2/30 = (T'/T)^4B_4T'^2/30 = H_4T'^6, \qquad (14)$$

whence $H = B_4/(30T^4)$, or rather the largest such term in absolute value in any of the equations. Upon solving for T' one finds $T' = (10^{-L}/H)^{1/6}$ is appropriate for the next sequence in a 9th-order integration.

In the case of a 15th-order integration of a system of simultaneous Class IIS or Class II equations, the result of an analogous calculation is

$$T' = (10^{-L}/H)^{1/9}, \quad \text{(15th-order case)} \tag{15}$$

where H is taken from the largest value of

$$H = B_7/(72T^7) \quad . \tag{16}$$

In 15th order values of L will run from 6 or 7 for low-accuracy calculations to 10 or more for high-accuracy problems. The global error for the entire integration is much smaller than 10^{-L}. See Sec. 3.2 below.

We can give no strong theoretical reason why the above calculation should be a good sequence size control. However, practical experience over a 10-year time span with invariably accurate results suggest that the above sequence size control is appropriate and usable.

The initial trial sequence size may be furnished by the programmer. If he makes no choice then the value +0.1 is chosen for forward integration and -0.1 for backward integration. This always gets one started. Choosing too large a starting value of T is not a serious problem. If the value of T(new) called for by the sequence size control is less than T(start), then the program itself chooses a more appropriate starting value and restarts.

The integrator can be set to accept a constant step size, and this is appropriate in some cases, as in Secs. 3.3 and 3.4 below.

3. RECENT PRACTICAL TESTS

Among the best of other integrators is DVDQ, a variable-order Adams (multistep) method by Krogh (1973) that solves equations of Class I, IIS, and II directly. There are DIFSYS and DIFSY2 by Bulirsch and Stoer (1966), extrapolation methods that solve Class I and IIS, respectively. Another is RKN8(9), a Runge-Kutta-Nystrom integrator of 8th order by Fehlberg (1972) that solves Class IIS. The first of these is accurate though not always fastest, the extrapolation methods have a low overhead per function call and are fast, and the RKN8(9) integrator is easy to program and fast for comparitively low accuracy problems. In some celestial mechanics problems, the customary integration is by a multistep method of high order. In an earlier time predictors were hand-calculated by differencing, but with machines it is often advantageous to predict with Lagrangian formulas.

Comparitive tests of RADAU and other integrators, by House, Weiss, and Weigandt (1978), have studied numerical integration of stellar orbits, while Papp, Innamen, and Patrick (1977, 1980) studied numerical orbit computations in galaxy models. These authors found RADAU to be a good compromise between speed and efficiency for their problems. It was chosen by Shefer (1982), who used it in 11th order for his study of perturbed orbits. In a review article Batrakov (1982) speaks of RADAU as being among the best of modern integrators, particularly for the Encke method. After extensive comparitive tests Carusi, Kresak, Perozzi, and Valsecchi (1985) chose RADAU in 19th order for their study of the long term evolution of all the short-period comets.

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

3.1 A particular 3-body Problem

A periodic orbit in the restricted three-body problem was described by Newton (1959). In the earth-moon system, idealized to circular orbits, a small body follows a complicated path with 3 loops passing very near the earth and the moon. Step size must be changed frequently on this orbit. The equations are:

$$y_1'' = 2y_2' + y_1 - u'(y_1 + u)/r_1^3 - u(y_1 - u')/r_2^3 ,$$
$$y_2'' = -2y_1' + y_2 - u'y_2/r_1^3 - u\,y_2/r_2^3 \quad . \tag{17}$$

Here $r_1 = ((y_1 + u)^2 + y_2^2)^{\frac{1}{2}}$, $r_2 = ((y_1 - u')^2 + y_2^2)^{\frac{1}{2}}$,

and $u = 1/82.45$, $u' = 1 - u$.

The initial conditions are:

$$y_1 = 1.2, \quad y_1' = 0 ,$$
$$y_2 = 0, \quad y_2' = -1.04935\ 75098\ 30319\ 90731\ 04104\ 34\ldots , \tag{18}$$

and $t_{final} = 6.19216\ 93313\ 19639\ 70699\ 23217\ \ldots$.

A CDC 6600 computer was used with RADAU in orders 7, 11, and 15 in single precision (60-bit word) and orders 19, 23, and 27 in double precision (120-bit word). In every case RADAU used less function calls than the other integrators. However, for the same absolute error, DVDQ was close at intermediate acccuracies. DIFSYS used twice as many function calls because it solved this system as 4 first order equations. However, its low overhead per function call made it comparable in timing. Gallaher and Perlin (1966) gave the long numbers in the problem to 21 digits. The last 5 or 6 digits were found by RADAU in 27th order.

3.2 An Elliptical Orbit Problem

Using the IBM-PC with a 64-bit double word one integrates 8 times around an ellipse of eccentricity 0.6 looking at the error of closure. Here 11th and 15th order were about comparable in timing for accuracies from 5 to about 12 digits.

Quite recently we tested this problem again using a Cray-1 computer in double precision (128-bit word). Our integrator could be set for 15th, 19th, 23rd, or 27th order (version RA27). The limiting accuracy was in the 24th or 25th decimal digit because of roundoff. The high orders reached this accuracy easily, but, surprizingly, so did 15th order at L=14. (15th order used more function calls.) This 15th order case is most interesting. The last term in the series for each sequence was held to 10^{-14} or less, but the global error was 10^{-24} in hundreds of sequences. The error in one sequence must have been very much

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

smaller than this latter figure. More than anything else, this fact shows the extraordinary properties of the truncated series developed from Gaussian spacings.

In these highly accurate calculations it is advantagous to iterate 3 times through each sequence.

This test is useful, but if one must integrate such an eccentric two-dimensional orbit, he should use Levi-Civita regularization (1903), see Bettis and Szebehely (1972). This removes the singularities from the differential equations and speeds this problem by a factor of eight.

3.3 The Outer Planets Problem

We tested the outer planet integration problem of Eckert, Brouwer, and Clemence (1951). They used a 12th-order multistep Lagrangian integrator in the predict-evaluate mode with 40-day steps. Their result was the positions of the 5 outer planets to 9 decimals for the years 1653-2060. Their calculation required 120 hours on a very early large computer. We used RADAU in 15th order, taking 320-day sequences (7 substeps/sequence) and integrating from a starting point in 1941 back to 1653 in 3.7 seconds on a CDC 7600 computer in single precision (60-bit word). The agreement for Jupiter with the 1951 paper was exact to the number of digits published. The other planets sometimes differed by 1 or 2 in the last digit. It should be noted that the multistep method would be just as fast on a modern computer. The advantage of RADAU is the it is self-starting, whereas the multistep methods are difficult to start.

Recently this same problem was done to the same accuracy with an IBM Personal Computer (equipped with the 8087 co-processor) in double-precision FORTRAN (64-bit word). Using RADAU in 15th order, the time was 8.7 minutes. It is practical to do such problems on this home computer.

3.4 A Class I Problem

Most of the problems in celestial mechanics involve 2nd-order differential equations, but the 1st-order equation is important in many fields. We tried the equation

$$y' = t(1 - y) + (1 - t)\exp(-t), \qquad y(0) = 1, \tag{19}$$

described by Krogh (1973), for which the solution is

$$y = 1 - \exp(-t) + \exp(-t^2/2). \tag{20}$$

The test is to integrate out to t=10. This equation is difficult to integrate because instability usually sets in as the integration proceeds. Indeed, RADAU became unstable and could not do this when allowed to pick its own sequence sizes, but when a constant sequence size of 0.2 or smaller was specified, then the error was in the 16th digit. For this test RADAU was in 15th order with a 64-bit double word. This is a 'stiff' equation. Often systems of stiff equations have both very large and very small eigenvalues. Though not developed for these, RADAU can handle some cases. The equations of celestial mechanics are not stiff.

This problem is somewhat unusual in that the independent variable t appears explicitly on the right. This causes no difficulty with the integrator, but does not often happen in celestial mechanics problems.

4. LISTING OF RADAU INTEGRATOR

It is not easy to program a complicated integrator, such as RADAU, even when given a full mathematical description of the algorithm. A FORTRAN listing of the subroutine is therefore given here. There was a choice: (1) Giving a version called RA7 which uses loops to calculate the series and where one may choose 7th, 11th, or 15th order. (2) Giving a 15th-order version called RA15 where the series are written out explicitly. Choice (2) was made: because it is some 30% faster, because 15th-order is often suitable, and because it is easier to compare the written-out series with the mathematical description.

It will be evident how to handle any number NV of simultaneous differential equations. In the listing will be seen loops with index K running from 1 to NV which bring all equations along together.

It should not be difficult to write (say) an 11th-order version by modification of the 15th-order listing in the appendix. Dimension H for 6 and put the Radau spacings for order 11 in the DATA statement. The dimensioning of other quantities is a little excessive, but will do no harm. Each series in written out only through terms with indices (5,K). The index 7 in most loops is changed to 5, and 8's are changed to 6's. The quantity PW becomes 1./7. Besides RA7 described already, another version available on request is RA27. This will integrate in 15th, 19th, 23rd, or 27th order. This requires a computer with a 120- or 128-bit double word and will integrate to 24 decimal-digit accuracy.

4.1 FORTRAN Listing of RA15, the 15th-order version of RADAU

```
      SUBROUTINE RA15(X,V,TF,XL,LL,NV,NCLASS,NOR)
C  Integrator by E. Everhart, Physics Department, University of Denver
C  This 15th-order version is called RA15. Order NOR is 15.
C  y'=F(y,t)    is   NCLASS=1,            y"=F(y,t) is NCLASS= -2,
C  y"=F(y',y,t) is   NCLASS=2
C  TF is t(final) - t(initial). (Negative when integrating backward.)
C  NV = the number of simultaneous differential equations.
C  Change dimensioning if NV is greater than 18.
C  LL controls accuracy. Thus SS=10.**(-LL) controls the size of
C  the last term in a series. Try LL=8 and work up or down from there.
C  However, if LL.LT.0, then XL is the constant sequence size used.
C  A non-zero XL sets the size of the first sequence regardless of
C  LL's sign.  Zero's and Oh's look alike on this printer. Use care!
C  X and V enter as the starting position-velocity vector (values of y
C  and y' at t=0) and they output as the final position-velocity vector.
C  Integration is in double precision. A 64-bit double-word is assumed.
C  In some computers IMPLICIT REAL*8 should be IMPLICIT DOUBLE PRECISION
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 TVAL,PW
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
C  The vectors X and V are dimensioned for unity below, because they
C  appear in the call. Storage for them is set out in the main program.
      DIMENSION X(1),V(1),F1(18),FJ(18),C(21),D(21),R(21),Y(18),Z(18),
     A     B(7,18),G(7,18),E(7,18),BD(7,18),H(8),W(7),U(7),NW(8)
      LOGICAL NPQ,NSF,NPER,NCL,NES
      DATA NW/0,0,1,3,6,10,15,21/
      DATA ZERO, HALF, ONE,SR/0.0D0, 0.5D0, 1.0D0,1.4D0/
C  These H values are the Gauss-Radau spacings, scaled to the range 0
C  to 1, for integrating to order 15.    H(1) = 0.D0 always.
      DATA H/         0.D0, .0562625605369221 5D0, .1802406917368923 6D0,
     A.3526247171131696 4D0, .5471536263305553 8D0, .7342101772154105 3D0,
     B.8853209468390957 7D0, .9775206135612875 0D0/
C  The sum of these H-values should be 3.7333333333333333
      NPER=.FALSE.
      NSF=.FALSE.
      NCL=NCLASS.EQ.1
      NPQ=NCLASS.LT.2
C  y'=F(y,t),NCL=.TRUE.  y"=F(y,t),NCL=.FALSE.  y"=F(y',y,t),NCL=.FALSE.
C  NCLASS=1, NPQ=.TRUE.  NCLASS=-2,NPQ=.TRUE.   NCLASS= 2,   NPQ=.FALSE.
C  NSF is .FALSE. on starting sequence, otherwise .TRUE.
C  NPER is .TRUE. only on last sequence of the integration.
C  NES is .TRUE. only if LL is negative. Then the sequence size is XL.
      DIR=ONE
      IF(TF.LT.ZERO) DIR=-ONE
      NES=LL.LT.0
      XL=DABS(XL)*DIR
      PW=1./9.
C  Evaluate the constants in the W-, U-, C-, D-, and R-vectors
      DO 14 N=2,8
      WW=N+N*N
      IF(NCL) WW=N
      W(N-1)=ONE/WW
      WW=N
   14 U(N-1)=ONE/WW
      DO 22 K=1,NV
      IF(NCL) V(K)=ZERO
      DO 22 L=1,7
      BD(L,K)=ZERO
   22 B(L,K)=ZERO
      W1=HALF
      IF(NCL) W1=ONE
      C(1)=-H(2)
      D(1)=H(2)
      R(1)=ONE/(H(3)-H(2))
      LA=1
      LC=1
      DO 73 K=3,7
      LB=LA
      LA=LC+1
      LC=NW(K+1)
      C(LA)=-H(K)*C(LB)
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
      C(LC)=C(LA-1)-H(K)
      D(LA)=H(2)*D(LB)
      D(LC)=-C(LC)
      R(LA)=ONE/(H(K+1)-H(2))
      R(LC)=ONE/(H(K+1)-H(K))
      IF(K.EQ.3) GO TO 73
      DO 72 L=4,K
      LD=LA+L-3
      LE=LB+L-4
      C(LD)=C(LE)-H(K)*C(LE+1)
      D(LD)=D(LE)+H(L-1)*D(LE+1)
   72 R(LD)=ONE/(H(K+1)-H(L-1))
   73 CONTINUE
      SS=10.**(-LL)
C  The statements above are used only once in an integration to set up
C  the constants. They use less than a second of execution time.  Next
C  set in an estimate to TP based on experience. Same sign as DIR.
      TP=0.1D0*DIR
      IF(XL.NE.ZERO) TP=XL
      IF(TP/TF.GT.HALF) TP=HALF*TF
      NCOUNT=0
      WRITE (*,3)
    3 FORMAT(/' No. of calls, Every 10th seq.X(1),X(2),T,TM,TF')
C  An * is the symbol for writing on the monitor. The printer is unit 4.
C  Line 4000 is the starting place of the first sequence.
4000  NS=0
      NF=0
      NI=6
      TM=ZERO
      CALL FORCE (X, V, ZERO, F1)
      NF=NF+1
C  Line 722 begins every sequence after the first. First find new
C  G-values from the predicted B-values, following Eqs. (7) in text.
 722  DO 58 K=1,NV
      G(1,K)=B(1,K)+D(1)*B(2,K)+
     X  D(2)*B(3,K)+D(4)*B(4,K)+D( 7)*B(5,K)+D(11)*B(6,K)+D(16)*B(7,K)
      G(2,K)=            B(2,K)+
     X  D(3)*B(3,K)+D(5)*B(4,K)+D( 8)*B(5,K)+D(12)*B(6,K)+D(17)*B(7,K)
      G(3,K)=B(3,K)+D(6)*B(4,K)+D( 9)*B(5,K)+D(13)*B(6,K)+D(18)*B(7,K)
      G(4,K)=            B(4,K)+D(10)*B(5,K)+D(14)*B(6,K)+D(19)*B(7,K)
      G(5,K)=                        B(5,K)+D(15)*B(6,K)+D(20)*B(7,K)
      G(6,K)=                                    B(6,K)+D(21)*B(7,K)
   58 G(7,K)=                                                B(7,K)
      T=TP
      T2=T*T
      IF(NCL) T2=T
      TVAL=DABS(T)
C  Writing to the screen during the integration lets one monitor the
C  progress. Values are shown at every 10th sequence.
      IF(NS/10*10.EQ.NS) WRITE(*,7) NF,NS,X(1),X(2),T,TM,TF
    7 FORMAT(1X,2I6,5F12.5)
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
C  Loop 175 is 6 iterations on first sequence and 2 iterations therafter
      DO 175 M=1,NI
C  Loop 174 is for each substep within a sequence.
      DO 174 J=2,8
      JD=J-1
      JDM=J-2
      S=H(J)
      Q=S
      IF(NCL) Q=ONE
C  Here Y is used for the value of y at substep n. We use Eq. (9).
C  These collapsed series are broken into two parts because an otherwise
C  excellent  compiler could not handle the complicated expression.
      DO 130 K=1,NV
      A=W(3)*B(3,K)+S*(W(4)*B(4,K)+S*(W(5)*B(5,K)+S*(W(6)*B(6,K)+
     V   S*W(7)*B(7,K))))
      Y(K)=X(K)+Q*(T*V(K)+T2*S*(F1(K)*W1+S*(W(1)*B(1,K)+S*(W(2)*B(2,K)
     X  +S*A))))
      IF(NPQ) GO TO 130
C  Next are calculated the velocity predictors if needed for general
C  Class II. Here Z is used as the value of y' at substep n. (Eq. (10))
      A=U(3)*B(3,K)+S*(U(4)*B(4,K)+S*(U(5)*B(5,K)+S*(U(6)*B(6,K)+
     T    S*U(7)*B(7,K))))
      Z(K)=V(K)+S*T*(F1(K)+S*(U(1)*B(1,K)+S*(U(2)*B(2,K)+S*A)))
 130  CONTINUE
C  Find forces at each substep.
      CALL FORCE(Y,Z,TM+S*T,FJ)
      NF=NF+1
      DO 171 K=1,NV
C  Find G-values from the force FJ found at the current substep. This
C  section, including the many-branched GOTO, uses Eqs. (4) of text.
      TEMP=G(JD,K)
      GK=(FJ(K)-F1(K))/S
      GO TO (102,102,103,104,105,106,107,108),J
 102  G(1,K)=      GK
      GO TO 160
 103  G(2,K)=     (GK-G(1,K))*R(1)
      GO TO 160
 104  G(3,K)=    ((GK-G(1,K))*R(2)-G(2,K))*R(3)
      GO TO 160
 105  G(4,K)=   (((GK-G(1,K))*R(4)-G(2,K))*R(5)-G(3,K))*R(6)
      GO TO 160
 106  G(5,K)=  ((((GK-G(1,K))*R(7)-G(2,K))*R(8)-G(3,K))*R(9)-
     X     G(4,K))*R(10)
      GO TO 160
 107  G(6,K)= (((((GK-G(1,K))*R(11)-G(2,K))*R(12)-G(3,K))*R(13)-
     X     G(4,K))*R(14)-G(5,K))*R(15)
      GO TO 160
 108  G(7,K)=((((((GK-G(1,K))*R(16)-G(2,K))*R(17)-G(3,K))*R(18)-
     X     G(4,K))*R(19)-G(5,K))*R(20)-G(6,K))*R(21)
C  TEMP is now the improvement on G(JD,K) over its former value.
C  Now we upgrade the B-value using this difference in the one term.
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
C  This section is based on Eqs. (5).
 160  TEMP=G(JD,K)-TEMP
      B(JD,K)=B(JD,K)+TEMP
      GO TO (171,171,203,204,205,206,207,208),J
 203  B(1,K)=B(1,K)+C(1)*TEMP
      GO TO 171
 204  B(1,K)=B(1,K)+C(2)*TEMP
      B(2,K)=B(2,K)+C(3)*TEMP
      GO TO 171
 205  B(1,K)=B(1,K)+C(4)*TEMP
      B(2,K)=B(2,K)+C(5)*TEMP
      B(3,K)=B(3,K)+C(6)*TEMP
      GO TO 171
 206  B(1,K)=B(1,K)+C(7)*TEMP
      B(2,K)=B(2,K)+C(8)*TEMP
      B(3,K)=B(3,K)+C(9)*TEMP
      B(4,K)=B(4,K)+C(10)*TEMP
      GO TO 171
 207  B(1,K)=B(1,K)+C(11)*TEMP
      B(2,K)=B(2,K)+C(12)*TEMP
      B(3,K)=B(3,K)+C(13)*TEMP
      B(4,K)=B(4,K)+C(14)*TEMP
      B(5,K)=B(5,K)+C(15)*TEMP
      GO TO 171
 208  B(1,K)=B(1,K)+C(16)*TEMP
      B(2,K)=B(2,K)+C(17)*TEMP
      B(3,K)=B(3,K)+C(18)*TEMP
      B(4,K)=B(4,K)+C(19)*TEMP
      B(5,K)=B(5,K)+C(20)*TEMP
      B(6,K)=B(6,K)+C(21)*TEMP
 171  CONTINUE
 174  CONTINUE
      IF(NES.OR.M.LT.NI) GO TO 175
C  Integration of sequence is over. Next is sequence size control.
      HV=ZERO
      DO 635 K=1,NV
 635  HV=DMAX1(HV,DABS(B(7,K)))
      HV=HV*W(7)/TVAL**7
 175  CONTINUE
      IF (NSF) GO TO 180
      IF(.NOT.NES) TP=(SS/HV)**PW*DIR
      IF(NES) TP=XL
      IF(NES) GO TO 170
      IF(TP/T.GT.ONE) GO TO 170
   8  FORMAT (2X,2I2,2D18.10)
      TP=.8D0*TP
      NCOUNT=NCOUNT+1
      IF(NCOUNT.GT.10) RETURN
      IF(NCOUNT.GT.1) WRITE (4,8) NOR,NCOUNT,T,TP
C  Restart with TP=0.8*T if new TP is smaller than original T on 1st seq
      GO TO 4000
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
 170  NSF=.TRUE.
C Loop 35 finds new X and V values at end of sequence. Eqs. (11),(12).
 180  DO 35 K=1,NV
      X(K)=X(K)+V(K)*T+T2*(F1(K)*W1+B(1,K)*W(1)+B(2,K)*W(2)+B(3,K)*W(3)
     X     +B(4,K)*W(4)+B(5,K)*W(5)+B(6,K)*W(6)+B(7,K)*W(7))
      IF(NCL) GO TO 35
      V(K)=V(K)+T*(F1(K)+B(1,K)*U(1)+B(2,K)*U(2)+B(3,K)*U(3)
     V      +B(4,K)*U(4)+B(5,K)*U(5)+B(6,K)*U(6)+B(7,K)*U(7))
  35  CONTINUE
      TM=TM+T
      NS=NS+1
C  Return if done.
      IF(.NOT.NPER) GO TO 78
      WRITE(*,7) NF,NS,X(1),X(2),T,TM,TF
      WRITE(4,7) NF,NS
      RETURN
C  Control on size of next sequence and adjust last sequence to exactly
C  cover the integration span. NPER=.TRUE. set on last sequence.
78    CALL FORCE (X,V,TM,F1)
      NF=NF+1
      IF(NES) GO TO 341
      TP=DIR*(SS/HV)**PW
      IF(TP/T.GT.SR) TP=T*SR
 341  IF(NES) TP=XL
      IF(DIR*(TM+TP).LT.DIR*TF-1.D-8) GO TO 77
      TP=TF-TM
      NPER=.TRUE.
C  Now predict B-values for next step using Eqs. (13). Values from the
C  preceding sequence were saved in the E-matrix. The correction BD
C  is applied in loop 39 as described in Sec. 2.5.
  77  Q=TP/T
      DO 39 K=1,NV
      IF(NS.EQ.1) GO TO 31
      DO 20 J=1,7
  20  BD(J,K)=B(J,K)-E(J,K)
  31  E(1,K)=      Q*(B(1,K)+ 2.D0*B(2,K)+ 3.D0*B(3,K)+
     X          4.D0*B(4,K)+ 5.D0*B(5,K)+ 6.D0*B(6,K)+ 7.D0*B(7,K))
      E(2,K)=               Q**2*(B(2,K)+ 3.D0*B(3,K)+
     Y          6.D0*B(4,K)+10.D0*B(5,K)+15.D0*B(6,K)+21.D0*B(7,K))
      E(3,K)=                           Q**3*(B(3,K)+
     Z          4.D0*B(4,K)+10.D0*B(5,K)+20.D0*B(6,K)+35.D0*B(7,K))
      E(4,K)=   Q**4*(B(4,K)+ 5.D0*B(5,K)+15.D0*B(6,K)+35.D0*B(7,K))
      E(5,K)=               Q**5*(B(5,K)+ 6.D0*B(6,K)+21.D0*B(7,K))
      E(6,K)=                           Q**6*(B(6,K)+ 7.D0*B(7,K))
      E(7,K)=                                       Q**7*B(7,K)
      DO 39 L=1,7
  39  B(L,K)=E(L,K)+BD(L,K)
C  Two iterations for every sequence. (Use 3 for 23rd and 27th order.)
      NI=2
      GO TO 722
      END
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

4.2 The main program

Whereas the integrating subroutine RA15 of the previous section retains the same form for different problems, the main program is specific to the particular problem. Besides providing the parameters in the call to RA15, it must give inital conditions, final value of the independent variable, and display or use the final values. A listing of JSUNP.FOR that controls the outer planet program will be sent on request. This gives initial positions and velocities of the 5 planets at the start. When combined with the programs in Sec. 4.1 and 4.3 all parts will be in hand for doing the outer planet problem of Sec. 3.3

4.3 The FORCE subroutine

This example of a force subroutine is rather complicated, but instructive and useful.

```
      SUBROUTINE FORCE(X,V,TM,F)
C  The FORCE subroutine for the 5 outer planet integration.
      IMPLICIT REAL*8 (A-H,O-Z)
C The above Implicit statement assumes an 8-byte double word (64 bits).
C X, V. and F are dimensioned unity because they appear in the call.
      DIMENSION X(1),V(1),F(1),RM(5),PM(5),R(5),RH(5,5)
C  The reciprocal masses of the 5 planets, units of reciprocal sun.
      DATA RM/1047.355D0, 3501.6D0, 22869.D0, 19314.D0, 360000.D0/
      DATA SC,SCZ,KSA/0.D0, 0.D0, 0/
      IF(KSA.EQ.1) GO TO 5
      KSA=1
      SCZ=-(1.720209895D-2**2)*(800.D0**2)
      SC=-1.8938494521574133D2
C  SCZ is the Gaussian constant for an 800-day time unit, and SC is the
C  same except the mass of the sun is augmented by masses of inner
C  planets, Mercury through Mars.
C  X, V, and F are dimensioned for 15 in calling programs. Indices 1,2,3
C  are for x,y,z for Jupiter, 4,5,6 are for x,y,z Saturn, 7,8,9 are for
C  x,y,z  Uranus, 10,11,12 for x,y,z Neptune, and 13,14,15 for Pluto.
      DO 4 I=1,5
   4  PM(I)=-SCZ/RM(I)
   5  DO 10 N=1,5
      J=(N-1)*3+1
      R(N)=1.D0/DSQRT(X(J)**2+X(J+1)**2+X(J+2)**2)**3
      IF(N.EQ.5)  GO TO 10
      NA=N+1
      DO 9 L=NA,5
      K=(L-1)*3+1
      RH(N,L)=1.D0/DSQRT((X(J)-X(K))**2+(X(J+1)-X(K+1))**2+(X(J+2)
     A      -X(K+2))**2)**3
   9  RH(L,N)=RH(N,L)
C  indices K and L run 1-15, indices N and L for the planets run 1-5.
C  The mass factors are in PM, the distance from the sun of each planet
C  contribute to R, and the planet-to-planet distances contribute to RH.
  10  CONTINUE
```

https://doi.org/10.1017/S0252921100083913 Published online by Cambridge University Press

```
      DO 20 N=1,5
      J=(N-1)*3+1
      SCM=(SC-PM(N))*R(N)
      F(J  )=SCM*X(J  )
      F(J+1)=SCM*X(J+1)
      F(J+2)=SCM*X(J+2)
C  The F-values above are for the sun-planet forces/unit mass.
      DO 20 L=1,5
      IF(L.EQ.N) GO TO 20
      K=(L-1)*3+1
      F(J  )=F(J  )+PM(L)*((X(K  )-X(J  ))*RH(N,L)-X(K  )*R(L))
      F(J+1)=F(J+1)+PM(L)*((X(K+1)-X(J+1))*RH(N,L)-X(K+1)*R(L))
      F(J+2)=F(J+2)+PM(L)*((X(K+2)-X(J+2))*RH(N,L)-X(K+2)*R(L))
C  The mutual planetary perturbation forces/unit mass are added on. The
C  first part of the second term is due to the planet-to-planet force,
C  and the second part is the indirect term because the sun at the
C  origin is not at the center of mass of the system.
   20 CONTINUE
      RETURN
      END
```

ACKNOWLEDGEMENTS

I thank the National Science Foundation, who sponsored the work of visiting scientists at the National Center for Atmospheric Research, where the original testing of the integrator was carried out. To Dr. W. H. Goodyear, Dr. Paul R. Beaudet. Dr. Paul N. Swarztrauber, Dr. J. M. A. Danby, and Dr. Fred Fernald I owe a note of appreciation for their help. The use and testing of the integrator by Dr. B. G. Marsden, Dr. A. Carusi, Dr. G. B. Valsecchi, Dr. E. Perozzi, Mr. Shio Oikawa, and many others has been encouraging.

REFERENCES

Batrakov, Yu. N. :1982, 'Methods of Computation of the Perturbed Motion of Small Bodies in the Solar System' in Sun and Planetary System, (Proceedings of the VI European Meeting in Astronomy), W. Fricke and G. Teleki, editors, D. Reidel Publishing Co., Dordrecht. pp415-419.

Bettis, D. G. and Szebehely, V. :1972, 'Treatment of Close Approaches in the Numerical Integration of the Gravitational Problem of N-bodies' Gravitational N-Body Problem, M. Lecar, editor, D. Reidel Publishing Co., Dordrecht, pp388-405. See p395.

Bulirsch, R. and Stoer, J. :1966. 'Numerical Treatment of Ordinary Differential Equations by Extrapolatiom Methods' Num. Math., **8**, pp1-13.

Butcher, J. C. :1964, 'Integration Processes Based on Radau Quadrature Formulas', Math. Comp. **18**, pp233-344.

Carusi, A., Kresak, L., Perozzi, E., and Valsecchi, G. B. :1985 'The Long-Term Evolution Project' in Dynamics of Comets: Their Origin and Evolution, A. Carusi and G. B. Valsecchi, editors, D. Reidel Publishing Co.,Dordrecht. (IAU Colloq. 83, Rome, June 1984) This volume.

Eckert, W. J., Brouwer, D., and Clemence, G. M. :1951 'Coordinates of the Five Outer Planets, 1653-2060' Astron. Papers Am. Ephemeris, **12**.
Everhart, E. :1974a, 'Implicit Single-Sequence Methods for Integrating Orbits', Celest. Mech. **10**, pp35-55.
Everhart, E. :1974b 'An Efficient Integrator of Very High Order and Accuracy' Denver Res. Inst. Tech. Report, 1 July 1974 (unpublished).
Everhart E. and Marsden, B. G. :1983, 'New Original and Future Cometary Orbits' Astron. J., **88**, pp135-137.
Fehlberg, E. :1972, 'Classical Eighth- and Lower Order Runge-Kutta-Nystrom Formulas with Stepsize Control for Special Second-Order Differential Equations', NASA Tech. Rept., NASA TR R-381.
Gallaher, L. J. and Perlin, I. E. :1966, 'A comparison of Several Methods of Numerical Integration of Nonlinear Differential Equations'. Presented at the SIAM meeting, Univ. of Iowa, March. 1966 (unpublished). See Krogh, 1973.
House, F., Weiss, G., and Weigandt, R. :1978, 'Numerical Integration of Stellar Orbits', Celest. Mech., **18**, pp311-318.
Krogh, F. T. :1973, 'On Testing a Subroutine for Numerical Integration of Ordinary Differential Equations', J. Assoc. Comput. Mach., **20**, pp545-562.
Levi-Civita, T. :1903, 'Traiettorie singulari ed urti nel problema ristretto del tre corp', Annali di Mat. **9**, pp1-32.
Marsden, B. G., Sekanina, Z., and Everhart, E. :1978, 'New Osculating Orbits for 110 Comets and Analysis of Original Orbits for 200 Comets', Astron. J., **83**, pp64-71.
Newton, R. R. :1959, 'Periodic Orbits of a Planetoid', Smithson. Contrib. Astrophys. **3**, No. 7, pp69-78.
Papp, K. A., Innanen, K. A., and Patrick, A. T. :1977, 1980, 'A Comparison of Five Algolrithms for Numerical Orbit Computation in Galaxy Models', Celest. Mech., **18**, pp277-286, and **21**, 337-349.
Shefer, V. A. :1982, 'Variational Equations of the Perturbed Two Body Problem in Regularized Form', Institute of Theoretical Astronomy, Leningrad. Publication No. **37**. An 11th-order version of RADAU is used.
Stroud, A. H. and Secrest, D. :1966, Gaussian Quadrature Formulas Prentice Hall, Inc. Englewood Cliffs, N.J., See Table 12 (Radau spacings).

QUESTION: (A. Milani) At the University of Pisa we have been using an integrator similar to yours of the implicit Runge-Kutta type which uses information from a previous step to start the iteration. However, we use the Gauss-Legendre spacings because we get order 2n with n sub-steps. Why do you think the Gauss-Radau spacing is better?

ANSWER: (E.Everhart) When one uses Gauss-Radau spacings there is a force found at the starting point of a sequence, and this is not repeated when one iterates. The trial expression of Eq. (9) is already correct through the quadratic terms, and the iteration should converge quicker. None-the-less, a very fine integerator can surely be built on Gauss-Legendre spacings and I am delighted to hear of yours. Please send me a listing and description when you can.