JOURNAL OF COMPUTATIONAL PHYSICS 67, 19 27 (1986)

Computer Implementation of an Algorithm for the Quadratic Analytical Solution of Hamiltonian Systems

R. A. HOWLAND*

Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, Indiana 46556

AND

D. L. RICHARDSON

Department of Aerospace Engineering, University of Cincinnati, Cincinnati, Ohio 45221

Received June 1, 1983; revised July 5, 1985

A software package for the analytical solution of Hamiltonians defining perturbed periodic motion is described. Features of the algorithm are reviewed briefly, and options and system requirements of the program are indicated. Finally, Duffing's equation illustrates a typical run in which frequencies through seventh order and solutions through third both expressed in terms of canonical variables, and then in terms of the original amplitude—are found in only two algorithm steps. © 1986 Academic Press, Inc.

For some time the first author has been involved in the development of a Lie transform technique to solve Hamiltonians describing near-linear periodic systems [1, 2]. If rapid [formal] convergence to a high-order analytic solution is a major feature of this method, so, too, are the accompanying algebraic complexities, which soon become overwhelming. Thus, to realize fully the potential of the technique, computer implementation was recommended, and it was decided to devise a general-purpose, user-oriented routine.

The results, a package of FORTRAN programs [3] using rational fraction or floating point coefficients, and utilizing SAP [4], a subroutine package for the algebraic manipulation of Poisson series written in part by the second author, are described in the present paper. To aid in understanding the quadratic transformation properties of the program, the basic algorithm is first reviewed briefly, then the program itself and its optional features are detailed. Finally, Duffing's equation is used as an example illustrating the method and the capabilities of the package.

* Research supported in part by NSF Grant MCS-8003592.

0021-9991/86 $3.00

1. The Algorithm

1.1. *The Basic Algorithm*

Given a continuous function $f(\mathbf{x}, \mathbf{X}; \kappa, \varepsilon)$, analytic in the conjugate n-vector coordinates $\mathbf{x}$ and momenta $\mathbf{X}$ about $\kappa = 0$ and $\varepsilon = 0$, and a C^∞ function $W(\mathbf{x}, \mathbf{X}; \varepsilon)$, we define [1, 5, 6] the *Lie operator* (the Poisson bracket)

$$L_W f \equiv \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \frac{\partial W}{\partial X_i} - \frac{\partial f}{\partial X_i} \frac{\partial W}{\partial x_i} \right),$$

and its iterates

$$L_W^i f \equiv L_W (L_W^{i-1} f), \qquad L_W^0 f \equiv f,$$

and thus the *Lie transform of f under* (or *generated by*) W

$$\exp(\kappa L_W) f \equiv \sum_{i \geq 0} \frac{\kappa^i}{i!} L_W^i f.$$

It can be shown [1, 6] that, under the coordinate transformation

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{X} \end{pmatrix} = \begin{pmatrix} \mathbf{x}(\mathbf{y}, \mathbf{Y}; \kappa, \varepsilon) \\ \mathbf{X}(\mathbf{y}, \mathbf{Y}; \kappa, \varepsilon) \end{pmatrix} = \exp(\kappa L_{W(\mathbf{y}, \mathbf{Y}; \varepsilon)}) \begin{pmatrix} \mathbf{y} \\ \mathbf{Y} \end{pmatrix}, \tag{1.1.1}$$

an arbitrary function, $f(\mathbf{x}, \mathbf{X}; \varepsilon)$, transforms according to the relation

$$\begin{aligned} f^*(\mathbf{y}, \mathbf{Y}; \kappa, \varepsilon) &\equiv f(\mathbf{x}(\mathbf{y}, \mathbf{Y}; \kappa, \varepsilon), \mathbf{X}(\mathbf{y}, \mathbf{Y}; \kappa, \varepsilon); \varepsilon) \\ &= \exp(\kappa L_{W(\mathbf{y}, \mathbf{Y}; \varepsilon)}) f(\mathbf{y}, \mathbf{Y}; \varepsilon); \end{aligned} \tag{1.1.2}$$

further, the inverse transformation is given by

$$\begin{aligned} f(\mathbf{x}, \mathbf{X}; \kappa, \varepsilon) &\equiv f^*(\mathbf{y}(\mathbf{x}, \mathbf{X}; \kappa, \varepsilon), \mathbf{Y}(\mathbf{x}, \mathbf{X}; \kappa, \varepsilon); \kappa, \varepsilon) \\ &= \exp(\kappa L_{-W(\mathbf{x}, \mathbf{X}; \varepsilon)}) f^*(\mathbf{x}, \mathbf{X}; \kappa, \varepsilon). \end{aligned} \tag{1.1.3}$$

The transformation (1.1.1) is canonical and has the appealing feature of giving solutions *explicitly* in terms of *all* old or *all* new variables, rather than only implicitly as with transformations generated by a classical function of mixed variables. Thus the solution can be recovered directly from (1.1.1), obviating the inversion necessary with a traditional Hamiltonian approach; furthermore, since (1.1.3) expresses the transformed variables entirely in terms of the original ones, it is an easy matter to specify initial conditions on the new variables from the conditions on the old variables.

For functions of the form

$$H(\mathbf{x}, \mathbf{X}; \varepsilon) = H_0(\mathbf{x}, \mathbf{X}; \varepsilon) + \varepsilon^m H_1(\mathbf{x}, \mathbf{X}; \varepsilon),$$

(1.1.2) can be written

$$H^* = H_0 + \varepsilon^m H_1 + \kappa L_W H_0 + O(\kappa^2); \tag{1.1.4}$$

thus the choice of W governs the form of the transformed function, particularly its lowest order (in κ) — an observation which forms the basis for application of the method detailed below.

In certain cases, $L_{W(\mathbf{x},\mathbf{X};\varepsilon)} H_0(\mathbf{x}, \mathbf{X}; \varepsilon)$ may be factored by positive powers of the parameter ε:

$$L_W H_0 = O(\varepsilon^q).$$

But if W satisfies the *determining equation* [2]

$$L_W H_0 + H_1 = \kappa\varepsilon^{-q} R, \qquad R \equiv O(1), \tag{1.1.5}$$

the transformation (1.1.2) assumes the form [7]

$$H^* = \sum_{i \geq 0} \frac{\kappa^i}{i!} L_W^i (1 - i)\, H_0 + \exp(\kappa L_W)(\kappa^2 \varepsilon^{-q} R). \tag{1.1.6}$$

W satisfying (1.1.5) will generally have terms factored by ε^{-q}; but (1.1.6) demonstrates that, to $O(\kappa^2 \varepsilon^{-q})$, the transformed function is just H_0 itself.

Various choices of H_0 and H_1 in (1.1.5) result in different forms of transformation algorithm (1.1.6); as might be expected of a method operating on several orders of perturbation simultaneously, the "efficiency," measured by counting the number of interterm multiplications, varies considerably with the form of specific problem [7]. Clearly the attempt to predict a priori the most efficient choice of algorithm is an uncertain enterprise, but the final program implements a rudimentary search for the one appropriate in a given case.

Note that this algorithm, (1.1.6), is only applied to functions used to determine the generating function in (1.1.5); other functions are still transformed with (1.1.2).

The final Hamiltonian algorithm, then, consists of the following equations, summarized here for convenience:

Determining Equation: $L_W H_0 + H_1 = \kappa\varepsilon^{-q} R$

(General) Transformation: $f^* = \exp(\kappa L_W) f$

(Hamiltonian) Transformation: $H^* = \sum_{i \geq 0} \frac{\kappa^i}{i!} (1 - i)\, L_W^i H_0 + \exp(\kappa L_W)(\kappa^2 \varepsilon^{-q} R)$

Inverse Transformation: $f = \exp(-\kappa L_W) f^*$.

Application of the technique to nonlinear periodic Hamiltonian systems consists in first separating the perturbation (terms depending on the angles, $\mathbf{x}$) from that part depending only on the actions, $\mathbf{X}$:

$$H(\mathbf{x}, \mathbf{X}; \varepsilon) = H_0(\_, \mathbf{X}; \varepsilon) + \varepsilon^m H_1(\mathbf{x}, \mathbf{X}; \varepsilon).$$

The generating function is then determined from (1.1.5), and the resulting transformations carried out using (1.1.6) and, as desired, (1.1.2). Setting $\kappa = \varepsilon^m$ "in magnitude" then eliminates the perturbation (see (1.1.4)), and the function reverts to one depending on the single small parameter, ε. In certain cases a linear combination of frequencies (or even a single frequency) may be of order greater than zero. In this case of *linear independence of order q*,—i.e., the frequencies,

$$\omega_i = \frac{\partial(\sum_{j=0}^{m} \varepsilon^j H_j)}{\partial X_i},$$

are linearly *de*pendent for $m = 0, \ldots, q-1$—

$$L_W H_0 = \sum \omega_i \frac{\partial W}{\partial x_i}$$

in the determining equation (1.1.5) is factored by ε^q, so W by ε^{-q}; but (1.1.6) demonstrates that the perturbation is still eliminated to $O(\kappa^2 \varepsilon^{-q}) = 2m - q$.

The above describes a typical transformation step. Having calculated the transformed Hamiltonian to the ultimately desired order using (1.1.6), one can determine an *entirely new* transformation, generated by an appropriate [*new*] function and expanded in powers of the small parameter $\kappa^* = \varepsilon^{2m-q}$, to bring the Hamiltonian closer to the desired form. This subsequent step eliminates angles to order $2(2m-q) - q = 4m - 3q$. Iteration of the process results in the quadratic transformation for $q = 0$; degeneracy reduces the speed of transformation somewhat, but it is still continually accelerated. The [sequence of] transformations (1.1.1) then give the original variables in terms of the final (soluble) ones—i.e., the solution; the inverse transformation (1.1.3) can be used to match initial conditions on the new variables to those of the original.

There are several benefits of this approach. Lie transforms admit of easy transformation, and the use of a second small parameter, κ, measuring the magnitude of the present perturbation (as opposed to the original small parameter, ε, measuring the perturbation at the first step only) also allows easy inversion (1.1.3). In addition to the inherent simplicity, greater efficiency might be expected, and this is indeed the case for all but the simplest Hamiltonians [2].

But independent of the efficiency, there is an additional feature important to application: the quadratic property of the transformation itself. Inaccuracies in the frequencies limit the time interval over which asymptotic solutions to nonlinearly vibrating systems can be expected to be valid, since such errors generate time-factored periodic terms producing a "mixed secular" drift of the formal solution away

from the "actual" one. Thus it is common practice at the last step in the asymptotic solution of a nonlinear problem to select terms giving the frequencies, without determining explicitly the highest order of transformation [in which the solution is implicit]. So although the solution is known to one order (in the original small parameter) less than the frequencies, this solution can be expected to remain valid for times on the order of the reciprocal of the small parameter before the mixed secular error begins to encroach on the highest order of solution. But virtually all perturbation techniques are "linear," operating on in some sense "controlling"—only a single order at a time; thus *only* a single such higher order of frequencies can be found, independent of the order to which the solution itself is determined. Contrast this with the present method, in which multiple orders of the Hamiltonian are transformed (or "controlled"), allowing determination at the last step of *multiple* orders of frequency above the accuracy of the [explicit] solution—a number of higher orders which *increases* with the order to which the solution is found. And it is *this* number to which power the reciprocal of the small parameter is raised to give the order of time over which the solution can be expected to remain valid.

2. The Computer Program

2.1. *The Basic Program*

The algorithm described above is implemented through a set of general-purpose software packages consisting of a main program driving selected subroutines in *SAP* [4], a subroutine package written in part by the second author for the analytical manipulation of "Poisson series"—series with terms of the form

$$(\text{coef}) \cdot X_1^{k_1} \cdots X_n^{k_n} \cdot \begin{Bmatrix} \sin \\ \cos \end{Bmatrix} (l_1\theta_1 + \cdots + l_m\theta_m).$$

These are available in three versions, in which the coefficients are either rational fractions, double precision numbers, or "extended precision rational fractions." (In the last, both numerator and denominator are extended precision floating point numbers. Integer-like arithmetic is simulated by limiting the results of rational fraction calculations to numbers which can be expressed exactly with this variable type. The maximum absolute value of any integer in this [IBM] representation is $2^{112} - 1$, corresponding to integers of 33 significant places. A result larger than this halts execution.)

The package requires, as its minimum input, the definition of the relevant algebraic ("parametric") variables—the small parameter, action variables, and any others appropriate to the specific problem—and the trigonometric canonical angle variables; this, along with the identification of any fractional exponents for the parametric variables, initializes the algebraic manipulation package. Also required are the starting Hamiltonian (the floating point version of the program allows for

TABLE I

Program Options

"E(VALUATE)": Evaluates parametric variables in starting Hamiltonian.
"T(RANSFORMATION)": Gives coordinates in new variables (solutions).
"F(ULL TRANSFORMATION)": Calculates solutions to maximum accuracy.
"I(NVERT)": Finds inverse transformations of variables and frequencies (to match initial conditions).
"M(INIMUM ELIMINATION)": Disables search for greater-than-expected elimination.
"R(ESONANCE)": Overrides 0-denominator test in GENFUN; retains resonant terms in Hamiltonian and calculates degeneracy at each order.
"S(OLUTIONS)": Calls user-defined subroutines to initialize, manipulate, solutions.
"G(ENERATING FUNCTION)": Prints each generating function.
"O(VERRIDE)": Overrides GENFUN test checking elimination
"P(OISSON COUNT)": Counts Poisson brackets, multiplications.
"C(HANGE TOLERANCE)":[a] Sets new floating point tolerance from default 10^{-13}.
"A(LL ROUNDOFF TERMS ELIMINATED)":[a] Prints all terms ignored in GENFUN test; measures roundoff.

[a] Floating point program only.

either double precision or rational fraction coefficients on input) and the order to which the frequencies are desired. Minimum output consists of the transformed Hamiltonian and frequency expressions at each step, as well as the final Hamiltonian.

The program allows a selection of additional independent options, controlled by key letters on data cards added after the basic input; these are listed in Table I. ("GENFUN" refers to the subroutine which calculates and checks the generating function.)

At present there is one restriction on the input Hamiltonians, due to the fact that the current algebra program cannot treat series as denominators. Each term of the generating function has a linear combination of the frequencies—which *are* generally power series (in the original small parameter)—in its denominator. Thus the program finds the reciprocal of this quantity in a binomial expansion using a subroutine in the algebra package. But as implemented there (and again to avoid division by series), this procedure requires that the highest-order term of each denominator consists of a single term. If this condition is violated, the program halts.

For details of the technical considerations in implementation of the algorithm, see [8]. One point which might be mentioned, however, deals with internal checks on the transformation itself: at each step the generating function is substituted into the determining equation to ascertain *it* is satisfied; the Hamiltonian is then transformed *independently* and the predicted elimination verified. The extra time required is justified by the desire to maintain the integrity of the program package.

2.2. *System Requirements*

Both versions of the program have been implemented on three systems: the IBM 370/168 at the University of Notre Dame, the Amdahl 470-V7 at the University of Cincinnati, and the PDP 11/70 at the Rose-Hulman Institute of Technology (which served the campus while the first author was there). The former default to 4-byte integer values and have 16-byte entended precision capabilities; memory is a secondary consideration, and the program can deal with up to 30,000 individual terms allocated among the various series (Hamiltonian, generating function, transformed variables, etc.) The PDP 11/70, however, was exceptionally stringent in its memory requirements, due primarily to the [RSTS V7.0-07] operating system's maximum program size of 27K 16-bit words; in addition, the FORTRAN-IV [V2.5] implementation there admitted of only 2-byte integer and 8-byte floating point variables. Despite the space limitations, the use of 1-byte integer values for certain variables, and space and buffer optimization in compilation, allowed approximately 500 terms (slightly fewer in the floating point version) to be stored. Experience has indicated that, even were the PDP FORTRAN-PLUS (which implements 4-byte integers and 16-byte double precision) available at Rose, the extra storage required for such variables would not warrant their use: the modest memory available balanced well, with the rate at which integer overflow occurs in the rational fraction version on the one hand, and with roundoff error in the double precision program on the other.

3. Example Duffing's Equation

Duffing's equation is a simple nondegenerate system with one degree of freedom:

$$H(Q, P; \varepsilon) = \omega P + \varepsilon\left(\frac{3}{8}\frac{P^2}{\omega^2} - \frac{1}{2}\frac{P^2}{\omega^2}\cos 2Q + \frac{1}{8}\frac{P^2}{\omega^2}\cos 4Q\right). \tag{3.0.1}$$

The input is in Hamiltonian form, but in this case it was decided to express the output in terms of the unperturbed amplitude of vibration. This could not be done with a simple transformation, since the original amplitude had to be identified with the coefficient of a particular trigonometric term in the solution. Thus the "solution" option was used to initialize a new set of variables and to perform a Lagrange inversion of the amplitude at each step. No dislocation of the original routine was required to do this; the desired operations were merely programmed to be called (by this option) at the already-provided entry points in the main program.

The results, giving the solution and frequency in terms of the [third-order] amplitude of unperturbed oscillation, a, are

$$\text{Solution} = a\sin\theta - \varepsilon\frac{a^3}{32\omega^2}(\sin 3\theta) + \varepsilon^2\frac{a^5}{1024\omega^4}(21\sin 3\theta + \sin 5\theta)$$

$$-\varepsilon^3\frac{a^7}{3268\omega^6}(417\sin 3\theta + 43\sin 5\theta + \sin 7\theta)$$

$$\text{Frequency} = \omega + \varepsilon \frac{3a^2}{8\omega} - \varepsilon^2 \frac{15a^4}{256\omega^3} + \varepsilon^3 \frac{123a^6}{8192\omega^5}$$
$$- \varepsilon^4 \frac{921a^8}{262144\omega^7} + \varepsilon^5 \frac{4593249a^{10}}{33554432\omega^9}$$
$$+ \varepsilon^6 \frac{7930839a^{12}}{536870912\omega^{11}} - \varepsilon^7 \frac{68744007a^{14}}{8589934592\omega^{13}}.$$

The job was initially run with the standard rational fraction version of the program, but 4-byte integer overflow occurred in the seventh order of the Lagrange inversion of the frequency (though *not* in the frequency or solution expressed in the original canonical variables). Thus it was decided to use the extended precision rational fraction routine. The entire job took 19 seconds on the IBM 370—a factor of $4\frac{1}{2}$ times longer than with the standard rational fraction version. It is significant that three-quarters of the run time was involved with the Lagrange inversion; the purely canonical part of the program—which completed successfully using the 4-byte rational fraction version, recall—took less than 6 seconds with the extended precision variables, and less than $1\frac{1}{2}$ seconds with the standard program. It should be noted that the Lagrange inversion would have been a necessary part of solution had Lie transformations not been used. (For sake of comparison, the same program run on the PDP 11/70—it had to be done using the double precision version, due to 2-byte integer overflow in the rational fraction version—took about 15 times longer.)

4. Conclusion

This paper has described in some detail an algorithm for the quadratic analytical solution of Hamiltonian systems and a computer program implementing it. Copies are available by contacting either of the authors.

Several extensions of the basic program are planned. One follows a suggestion of Dr. André Deprit and would forestall the effects of floating point roundoff error by calculating solutions in rational fraction coefficients until integer overflow occurs; the results of he previous successfully completed step would then be stored automatically for input to the double precision routine. Another attempts to overcome the central memory limitation (which would become critical going to high orders) by utilizing temporary disc storage for all series not currently being manipulated. While continual disc access would be expected to increase run time significantly, it would allow essentially the entire memory allocation to be devoted to operations on the present series.

Although the program is designed to be of use on its own, it is also intended to serve as a tool to investigate accelerated perturbation techniques in general. In particular, the "Poisson count" option (Section 2.1), giving a running total of separate term-by-term multiplications at each step, indicates a substantial saving in "real

effort" over linear algorithms [7]. As mentioned above, reducing this number of multiplications also reduces the memory required to carry them out, suggesting the possibility of obtaining solutions to a higher accuracy (before exceeding memory limitations) than previously possible with linear methods.

ACKNOWLEDGMENTS

Although major development of this program by the first author was done under NSF Grant MCS-8003592, initial phases were carried out with the encouragement and support of the Rose-Hulman Institute of Technology, Terre Haute, Indiana, during his appointment there in the Department of Mechanical Engineering. He acknowledges both sources gratefully. The authors are also indebted to Dr. André Deprit of the National Bureau of Standards for his many helpful comments and suggestions, as well as to Dr. Allen Jupp of the University of Liverpool for his.

REFERENCES

1. R. A. HOWLAND, *Celestial Mech.* **15**, 327 (1977).
2. R. A. HOWLAND, *Celestial. Mech.* **19**, 95 (1979).
3. R. A. HOWLAND, "User's Guide for QUAD," Publ. Department of Aerospace and Mechanical Engineering (University of Notre Dame, 1983).
4. D. RICHARDSON, "S.A.P. User's Guide," Publ. Department of Aerospace Engineering (University of Cincinnati, 1980).
5. G. HORI, *Publ. Astron. Soc. Jpn.* **18**, 287 (1966).
6. A. DEPRIT, *Celestial Mech.* **1**, 12 (1969).
7. R. A. HOWLAND AND D. L. RICHARDSON, *Celestial Mech.* **32**, 99 (1984).
8. R. A. HOWLAND AND D. L. RICHARDSON, "Programming Considerations in Computer Implementation of an Algorithm for the Quadratic Analytical Solution of Hamiltonian Systems," *Modeling and Simulation in Engineering*, edited by W. F. Ames and R. Vichnevetsky (Reidel, Dordrecht, 1983), Vol. 3, p. 279.