12

DAHLGREN DIVISION
NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5100

NSWCDD/TR-94/217

PARALLEL BLOCK IMPLICIT INTEGRATION TECHNIQUE FOR TRAJECTORY PARALLELISM

BY ALAN E. RUFTY
STRATEGIC AND SPACE SYSTEMS DEPARTMENT

DECEMBER 1994

FOREWORD

This report describes the evaluation of a Parallel Block Implicit (PBI) integration technique in a simplified missile trajectory. This project was carried out to ascertain the suitability of PBI techniques when modest amounts of parallelism are available; that is, when 3 to 10 processors are allocated per missile trajectory. This work was performed in the SLBM Research and Analysis Division as a Systems Engineering Enhancement (SEE) project in fiscal year 1992.

The author would like to thank Mr. Alvin Good, SLBM Software Development Division (K50), for his help with the test results obtained from the TRANSPUTER. The author would also like to thank his research supervisor, Mr. Davis Owen, for his general encouragement and support.

This report has been reviewed by Mrs. Carol A. Rose, Head, Fire Control Formulation Branch and Dr. D. W. Lando, Head, SLBM Research and Analysis Division.

Approved by:

[illegible]

R. L. SCHMIDT, Head
Strategic and Space Systems Department

ABSTRACT

This report describes the evaluation of a Parallel Block Implicit (PBI) integration technique in a simplified missile trajectory. This project was carried out to ascertain the suitability of PBI techniques when modest amounts of parallelism are available; that is, when 3 to 10 processors are allocated per missile trajectory. The PBI technique was first evaluated on a serial mainframe computer before it was implemented in parallel on an INMOS TRANSPUTER with four parallel central processing units. While the serial implementation of the four-node PBI technique indicated that a speedup of a factor of three to four was possible with ideal hardware, in practice only a modest gain (approximately 30 percent) was obtained because of systems-related overhead.

CONTENTS

INTRODUCTION

From a hardware perspective parallel processing provides a natural way to greatly expand limited computational resources. This has given rise to a strong trend toward hardware parallelism; however, software is frequently unable to make efficient use of this parallelism. There are numerous reasons for this. First, as of early 1993, only rudimentary compilers exist for converting standard software into a suitable parallelized assembly language—there are, however, development efforts under way in this area for the Cray[1] as well as other environments. Second, standards are somewhat lacking for parallelism in both the hardware and compiler arenas. This further compounds matters by tending to make parallel coding efforts nontransportable. On this standardization front there are, however, several bright prospects with Fortran 90 being a notable example.[2] Third, for many hardware architectures parallel implementations have steep overhead requirements; consequently, if software is not well suited to the associated operating system and hardware environment then some parallel implementations can, in fact, consume more execution time than consumed by purely serial implementations. Finally, physical problems and engineering systems are quite frequently not amenable to parallelism and, if this is the case, 'smart' compilers and proper hardware can have only limited benefits.

There are two standard software strategies for implementing parallelism. In the first strategy, one attempts to have large blocks of high-level code that run independently. This approach is labeled 'high-level parallelism.' In the second strategy, parallelism is worked directly into the problem definition at some basic level. This second approach is known as 'low-level parallelism.' The potential applicability of both these approaches will depend on the combination of the hardware at hand, the problem under consideration, and the software engineering itself. If high-level parallelism is a viable strategy, it can frequently be implemented in a direct and straightforward way. For example, if one wishes to implement missile trajectory simulations (which is the main problem of interest here) and there are four completely independent processors available, one could directly implement one missile trajectory on each of the processors and obtain an approximate speedup of a factor of four with a minimal amount of effort—provided, of course, that neither data nor memory addressing problems arise. Given that such straightforward steps have been taken where appropriate, the following question arises: Can a better scheme be found or can additional parallelism be made use of? Low-level parallelism arises immediately in this context. For trajectory software the problem of how to attempt low-level parallelism is, however, enigmatic. This is not too surprising since trajectories themselves are serial in nature—the current 'position' and 'velocity' depend directly on the 'position' and 'velocity' at any given past instant. Stated differently: All dynamical variables have a 'Christmas tree light' or serial-like dependence on their past values. Any successful low-level parallel trajectory integration scheme must be robust enough to overcome the inherent penalty imposed by its nonserial nature. This report, then, analyzes what can be accomplished by applying low-level parallelism to trajectory modeling.

Just as there are two basic software strategies for implementing parallelism, there are two basic strategies on the hardware front. Parallel hardware can thus be characterized by two extremes: 'small-scale' parallelism where four or so processors are involved and 'large-scale' or massive parallelism where 1000 or more processors are frequently used. It is perhaps worth noting that while massive parallelism is rapidly gaining favor as a preferred way to engineer up scale performance,[3,4,5] the actual realized performance is frequently rather disappointing.

The particular evaluation effort at hand is restricted to 'small-scale' parallelism. The reasons for this restriction are twofold: first, available hardware for testing was very restricted and, in fact, a TRANSPUTER board with four individual processing units mounted on a 286 personal computer was used*. Second, it is clear that the problem under study (individual missile trajectories) is manifestly unsuited for massive parallelism. One relatively straightforward way to make use of low-level parallelism is to simply use the vector nature of trajectory variables by assigning one computation per register; however, this approach has severe limitations and tends to over utilize overhead resources. The most promising approach to low-level trajectory parallelism that emerged in this study was the Parallel Block Implicit (PBI) technique. This technique was first implemented on a serial computer (CDC 875) for extensive test and evaluation. Since it performed as hoped by displaying considerable promise in the standard mainframe environment, it was then implemented on a 286-based INMOS TRANSPUTER with four processing elements—as mentioned earlier. Mr. Alvin Good of the SLBM Software Development Division (K50) performed the TRANSPUTER implementation based on the supplied formulation—this formulation is included as Appendix A. He subsequently performed a thorough timing evaluation. Overhead proved to be a problem for the PBI technique just as it has for all other attempts at low-level parallelism (including in-house attempts based on COGENT machines). There are strong indications that if hardware without undue overhead constraints becomes available then the PBI technique will perform as desired.

This report contains a brief overview of parallel trajectory integration strategies, the merits of the PBI approach, a summary of procedures carried out in the present study, a synopsis of the test results obtained on the TRANSPUTER by Mr. Alvin Good of K50, and a summary of the findings. Appendix A contains mathematical implementation details for the PBI, as well as the Runge/Kutta (R/K), approach while Appendix B contains a derivation of the oblate gravity model used. It is worth noting that the PBI approach is not mentioned in standard text books and all the relevant PBI journal articles seem to predate the current trend toward parallelism.[6,7,8] Thus part of the intent of this report is to draw wider attention to a noteworthy but somewhat underutilized integration technique.

* When the TRANSPUTER boards are mounted on a 386 similar performance results, but there are many alternative transputer architectures.[5]

TRAJECTORY NUMERICAL INTEGRATION SCHEMES

This section gives an overview of various relevant trajectory integration strategies and then discusses the background studies leading up to the choice of the PBI technique as the leading candidate for low-level parallelism. For missile trajectories of interest, a complicated engineering system is being simulated and thus large numbers of so called 'critical events' can be expected. Critical events tend to strongly reduce the efficiency of standard 'predictor-corrector' techniques[9] and therefore R/K techniques naturally come under consideration. R/K techniques are 'self-starting' and thus can be easily adapted to problems where a large number of 'start-ups' arise from critical events. Unfortunately, R/K techniques are strongly serial in nature so they are not easily adapted to parallelism; moreover, the efficiency of R/K techniques in trajectory applications stems largely from this serial nature. Within certain specialized niches, existing R/K techniques do, however, have parallelization promise. For example, iterated implicit R/K techniques[10,11] are, in general, worth examining in connection with six degree of freedom (6–D) trajectory applications. Implicit R/K techniques are useful in situations where integration stability must be controlled; i.e., for a class of systems governed by so called 'stiff differential equations,'[12] which includes 6–D trajectory models.

Given the intrinsic difficulty of successfully parallelizing R/K approaches, alternative integration schemes came under consideration. Many new ideas surfaced as potentially interesting. Most of the approaches attempted to overcome the serial dependence associated with numerical integration midpoints. (For an example of this serial dependence see the fourth-order R/K procedure shown in Appendix A, Section II.) One alternative idea for an integration step procedure was to use polynomial fitting techniques. In this approach, one first uses a polynomial fit or other suitable basis set over a small interval in order to predict values for midpoint step locations. The resulting evaluations are then used in a functionalization integration step update procedure so as to complete the current integration step. Another alternative approach was based on the fact that geophysical collocation can be used to develop optimal integrators for geophysical quantities and these 'collocation integrators' are easily parallelized. In their realm of applicability, 'collocation integrators' are frequently more efficient than standard approaches because the covariance information used by 'collocation integrators' is more complete than the information used by standard quadrature integrators. Given this inherent efficiency, the hope was that such approaches could be adapted to replace standard trajectory numerical integrators. In particular, the idea was to use (geophysical) collocation-like techniques with covariance functions developed specifically for the integration task at hand as well as specific missile types under consideration so as to decouple midpoint step evaluations in the integration schemes. During the time these and a number of other ideas (such as Encke-based approaches and variation of parameters[13]) were under preliminary consideration, a thorough literature review turned up the PBI technique and none of these alternatives were followed up. The PBI technique has several of the best features of the alternative approaches already directly built into it. Moreover, PBI techniques have the great advantage of having a proven efficiency that is comparable to existing R/K techniques for serial implementations—something that could hardly be expected of any of the novel techniques listed previously (with the possible exception of geophysical collocation-like approaches). Perhaps the most significant aspect of the PBI technique is that it can be parallelized naturally. Further advantages of the PBI approach are discussed in the next section.

PARALLEL BLOCK IMPLICIT APPROACH

PBI techniques are discussed at length in the original literature[7,8] and a complete implementation description is given in Appendix A, Section III (which is based on a synopsis found in Reference 11). Further, since the main body of this report is introductory in nature, only a descriptive treatment is given in this section. PBI techniques contain a certain number of so-called nodes which are analogous to the midpoint evaluations of ordinary integration schemes except that they are completely independent and so can be evaluated in parallel. By subjecting these node points to a form of iteration, or iterative improvement, solutions of high order result. (The order of an integration scheme and other standard numerical analysis terms are defined in standard textbook references.[9]) When PBI techniques are compared with R/K techniques they generally involve more computations per integration step, but because of their greatly increased order of integration one can take much larger integration steps. In general for serial applications, because of these two offsetting factors, PBI techniques are roughly as efficient as R/K techniques. Any gain in PBI efficiency due to parallelism will thus result in direct improvements in comparison with R/K techniques. A mathematical description of four–node PBI techniques can be found in Section III of Appendix A. Parallelism arises in the PBI approach from the fact that, for each iteration of equations (A-13), (A-14), (A-15) and (A-16), the main trajectory evaluations are independent; that is, $\vec{\mathbf{F}}(1,s)$, $\vec{\mathbf{F}}(2,s)$, $\vec{\mathbf{F}}(3,s)$, and $\vec{\mathbf{F}}(4,s)$ can be evaluated on independent processors for each given value of s.

PBI techniques were judged as worth exploring, in part because of the following noteworthy features: (1) PBI techniques are inherently parallel. For example, the four–node iterative PBI system implemented uses four concurrent 'function' evaluations and thus is naturally suited to the use of four simultaneous, more or less equally loaded, processors. Furthermore, if each of the 3 vector components of position and velocity are assigned dedicated processors then 12 processors can, in theory, be efficiently utilized. In addition, PBI techniques exist for various numbers of nodes. (2) PBI techniques are of high order. The four-node PBI technique implemented is of order seven while most standard in-house R/K methods are of order four. (3) All the final node points are also accurate trajectory evaluation points so one can easily use them as interpolation points or as trajectory output points. (4) When implemented in serial, PBI techniques are roughly as efficient as R/K techniques. There is thus only a nominal penalty to overcome in using a PBI technique in place of a R/K technique. (5) PBI techniques are compatible with Encke techniques which are frequently used in conjunction with R/K techniques to enhance their efficiency. (6) The 'step size' can be varied at will from one integration cycle to the next. (7) PBI techniques are relatively easy to implement.

From this list it is clear that PBI techniques have a number of desirable traits. Thus, for example, points (3) and (6) when taken in conjunction mean that one can easily handle interrupts for critical events without greatly hampering the algorithm's efficiency—this is an important point since many otherwise efficient integration schemes are of rather limited SLBM utility because of it. PBI techniques may be tentatively considered the 'bench mark standard' by which other attempts at low-level trajectory numerical integration parallelism should be judged. The relative merits of the PBI approach are not, however, completely clear-cut. For example, while the chosen PBI implementation is seventh-order, there are only five usable interpolation points per integration cycle. In addition to limiting the accuracy of critical event predictions (i.e., item number (3)), this could make the algorithm somewhat less accurate when time-dependent forces are included. Time-dependent forces occur in practice when thrust tables are introduced. Thus, if the PBI technique is to be put to practical use it may have to be extended to handle explicit time dependence better. A preliminary examination suggests such modifications can be readily implemented when and if required.

The two primary issues in evaluating the merits of the PBI approach are the relative efficiency and accuracy of the technique and the suitability of the technique to available hardware.

TESTING RESULTS

To properly evaluate numerical algorithm efficiency and accuracy one must do comparisons relative to other schemes; consequently, the first step in this study was to compare the PBI techniques to a standard fourth-order R/K technique. This part of the study was performed with a test bed trajectory model implemented on a serial mainframe computer. After this mainframe testing phase was carried out, the second part of the testing phase involving a parallel TRANSPUTER implementation was performed. The results of both phases of testing should be born in mind when forming an assessment of the suitability of the PBI technique in the context of other uses. Available hardware was somewhat limited in the TRANSPUTER testing phase.

PROTOTYPE TRAJECTORY TESTING STAGE

First a 'test bed trajectory model' was developed. This test bed trajectory model was a 'synthetic' trajectory model implemented on a standard serial computer and served as a simplified test platform for parallel numerical integration techniques. This model was developed with a simplified vacuum and atmospheric trajectory reentry phase and also included the ability to model either tesseral or oblate gravity for all phases of flight. In addition, a simple thrust boost model was included but was not used in this study. Both a standard fourth-order R/K and four-node PBI integration technique were then implemented in this model. Comparison studies of these two integration schemes were carried out for both the vacuum and reentry phases of flight with both the oblate and tesseral gravity models. The PBI technique was found to be capable of taking integration steps that are between three and four times as large as the R/K technique. For an ordinary serial computer this means that the PBI technique is roughly as efficient as the R/K technique since, for example, the four-node PBI technique involves four times as much computation per step. However, on a parallel computer of the right architecture (i.e., a four-element system with low overhead), the four-node PBI technique should run four times as fast. Similar types of ratios should hold for PBI schemes with higher numbers of nodes if operating system overhead is low; i.e., a six–node PBI scheme should be roughly six times as fast with the proper architecture. Testing on serial computers thus indicated that PBI performance was as indicated in the technical literature.[7,8,11] The next step was to determine if the speedup of a factor of three and a half could actually be realized for available parallel computer systems. Toward that end, the test bed model was simplified even further and a corresponding formulation was written for Fortran code (see Appendix A). These products were then passed to K50 personnel who then performed the implementation and timing studies on the TRANSPUTER.

TRANSPUTER TESTING STAGE

While the theoretical limit of efficiency of the PBI technique as implemented is a speedup of a factor of four, there is a hardware-imposed theoretical speedup limit of a factor of three due to limitations of the four-element TRANSPUTER architecture employed. Moreover, with a four-element TRANSPUTER architecture even this limit of three is very hard to approach in low-level parallelism experiments because of hardware overhead considerations. In fact, any part of the trajectory that is not in parallel remains in full-time residence on at least one element and ties up system resources. Since the PBI integration technique is tied to trajectory architecture and a general-purpose study was intended, it was decided early on that K50 would implement only an oblate gravity force module and that more sophisticated force model aspects (i.e., tesseral gravity, thrust tables, aerodynamic forces, etc.) were to be 'emulated' (at least for timing study purposes) by looping through the oblate gravity routine repeatedly. More precisely, the requisite 'timing emulation loops' were placed internally to Section II.d (or III.d) of Appendix A, but outside all the computations in that section so as to perform the same computation repeatedly. For such comparison studies, typically 25 to 2000 loops might be expected to emulate various levels of complexity found in standard trajectory models.

Timing studies were conducted on the complete trajectories including the parts that were outside the integration loop and so not in parallel. Because of overhead, the PBI technique actually took more time than a serial implementation for 1 to 10 'timing emulation loops.' This result was expected. At 35 loops the four-node TRANSPUTER implementation took 24 percent less time than the serial implementation. This 24 percent less time amounts to a speedup of a factor of 1.31, which was an encouraging result. However, not much greater efficiency was obtained as the number of loops was increased—at 2000 loops there was a 29 percent savings. This corresponds to a speedup of a factor of 1.42. To check the hypothesis that TRANSPUTER overhead was hampering the PBI efficiency timing marks were set around just that part of the code which was explicitly parallelized (i.e., the oblate gravity routine). This experiment confirmed this hypothesis—for example, with 2000 loops around the oblate gravity computations a speedup of a factor of 2.85 was obtained. Here the factor of 2.85 is close to the theoretical limit of a factor of 3 previously mentioned. Given the overhead requirements of available in-house hardware, it seems that only modest gains can be expected from any form of low-level parallelism with the TRANSPUTER architecture used. The main question arising here is whether a new TRANSPUTER architecture (i.e., one employing more processing elements) could be employed so as to greatly lessen overhead problems—unfortunately such hardware-specific issues were outside the scope of the present study.

SUMMARY AND CONCLUSIONS

In summary, while the serial mainframe implementation indicated that a speedup of a factor of around 3.5 was quite possible with ideal parallel hardware, only a modest speedup of about 1.3 was obtained with the hardware at hand because of systems-related overhead. The problem of systems-related overhead is by no means limited to the hardware and software tested in the present low-level parallelism study, but has been found to be pervasive. By using redundant computational resources, it should be possible to circumvent overhead problems. The speedup of 2.85 obtained in the special experiment mentioned in the last section seems to lend strong support to this hypothesis. Given the current hardware overhead situation, it seems that the most efficient available strategy is to stick with high-level parallelism whenever possible. The PBI approach itself is, however, worth considering as a substitute for certain R/K applications.

REFERENCES

1. Comerford, R., "Software on the brink," *IEE Spectrum*, Vol. 29, No. 9, pp. 34–38, 1992.

2. Metcalf, M., "Still programming after all these years," *New Scientist*, Vol. 135, No. 1838, pp. 30–33, 1992.

3. Zorpette, G., "The power of parallelism," *IEE Spectrum*, Vol. 29, No. 9, pp. 28–33, 1992.

4. Cybenko, G. and Kuck, D. J., "Revolution or evolution?," *IEE Spectrum*, Vol. 29, No. 9, pp. 39–41, 1992.

5. Jacob, R. and Anderson, J., "Do-it-Yourself Massively Parallel Supercomputer Does Useful Physics," *Computers in Physics*, Vol. 6, No. 3, pp. 244–251, 1992.

6. Rosser, J. B., "A Runge–Kutta for all seasons," *SIAM Review*, Vol. 9, pp. 417–452, 1967.

7. Shampine, L. F. and Watts, A. H., "Block implicit one-step methods," *Math. Comput.*, Vol. 23, pp. 731–740, 1969.

8. Worland, P. B., "Parallel Methods for the numerical solution of ordinary differential equations," *IEE Trans. Comput.*, Vol. C-25, pp. 1045–1048, 1976.

9. Johnson, L. W., and Riess, R. D., *Numerical Analysis*, Second Edition, Addison–Wesley Publishing Co., Inc., Reading, MA, 1982.

10. Miranker, M. L. and Liniger, W., "Parallel Methods for numerical integration of ordinary differential equations," *J. Math. Comput.*, Vol. 21, pp. 303–319, 1967.

11. Hutchinson, D. and Khalaf, B. M. S., "Parallel algorithms for solving initial value problems: front broadening and embedded parallelism," *Parallel Computing*, Vol. 17, pp. 957–968, 1992.

12. Shampine, L. F. and Gear, C. W., "A User's View of Solving Stiff Ordinary Differential Equations," *SIAM Review*, Vol. 21, No. 1, pp. 1–17, 1979.

REFERENCES (continued)

13. Battin, Richard H., *An Introduction to the Mathematics and Methods of Astrodynamics*, American Institute of Aeronautics and Astronautics, Inc., New York, NY, 1987.

APPENDIX A

PARALLEL BLOCK IMPLICIT FORMULATION

CONTENTS

I. INTRODUCTION/GENERAL PROGRAM STRUCTURE

This formulation is intended for implementation on a parallel computer and is designed to test out the efficiencies of a candidate integration procedure for trajectory parallelism (the Parallel Block Implicit (PBI) integration algorithm) against a commonly used serial procedure (fourth-order Runge/Kutta (R/K)). Since the relative efficiencies of the two schemes are to be compared, both must be independently implemented and timing runs must then be conducted on both implementations. This formulation thus, in reality, consists of two independent trajectory formulations. The (serial) R/K consists of Sections II.a through II.d of the formulation, while the PBI formulation consists of Sections III.a through III.d. (Here Section III.d is identical to Section II.d and as such is not reproduced.) Sections II.a and III.a share 'common code,' which can be reused as noted explicitly in Section III.a. Both 'major sections' of the formulation (Section II and Section III) are sufficiently straightforward as to warrant only several minor additional comments.

With regards to the 'R/K formulation' (Section II) the 'Initial Condition (I.C.) Module' (Section II.a) and the 'R/K Integration Module' (Section II.b) are the two major modules with the I.C. Module being called first. Also the 'Three Force Module' (Section II.c) and the 'Oblate Gravity Module' (Section II.d) are called from the 'R/K Integration Module' (Section II.b). These last two modules are formulated as 'function calls' for convenience (i.e., Section II.c is formulated as $\mathbf{F}_{Three}(\cdots)$ while Section II.d is formulated as $\mathbf{G}_{Oblt}(\cdots)$). The R/K used is a standard text book fourth-order one—a mathematical description of the equations used can be found in the main introduction to the 'R/K formulation' (Section II).

With regards to the 'PBI formulation' (Section III) the 'I.C. Module' (Section III.a) and the 'PBI Integration Module' (Section III.b) are the two major modules with the I.C. Module being called first. The 'Six Force Module' (Section III.c) and the 'Oblate Gravity Module' (Section III.d or Section II.d) are called from the 'PBI Integration Module' (Section III.b). These last two modules are formulated as 'function calls' for convenience [i.e. Section III.c is formulated as $\vec{\mathbf{F}}_{six}(\cdots)$ while, as noted above, Section III.d or II.d is formulated as $\mathbf{G}_{Oblt}(\cdots)$]. A mathematical description of the equations used for the PBI technique can be found in the main introduction to the 'PBI formulation' (Section III).

II. RUNGE/KUTTA (R/K) IMPLEMENTATION

The second order differential equation to be integrated is

$$\ddot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{v}) , \tag{A-1}$$

where three-vectors are shown in boldface type (i.e., **x** and **v** are three-vectors while **f** is a vector valued function). Equation (A-1) must be integrated in a step-by-step fashion. It is standard practice to introduce a 'running index,' say i to indicate the present step of the integration process. (It is not necessary to dimension variables for this index—all that is required is to simply make sure that the 'new' or 'updated values' [i.e., the $(i+1)$'th values] are distinguished from the 'current values' of the appropriate registers [i.e., the i'th values].) The standard fourth-order R/K update equations are

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \tfrac{h}{6}(\mathbf{m}_0 + \mathbf{m}_1 + \mathbf{m}_2) \tag{A-2}$$
$$\mathbf{v}_{i+1} = \mathbf{v}_i + \tfrac{1}{6}(\mathbf{m}_0 + 2\mathbf{m}_1 + 2\mathbf{m}_2 + \mathbf{m}_3), \tag{A-3}$$

where

$$\mathbf{m}_0 = h\mathbf{f}(t_i, \mathbf{x}_i, \mathbf{v}_i) \tag{A-4}$$
$$\mathbf{m}_1 = h\mathbf{f}(t_i + \tfrac{h}{2}, \mathbf{x}_i + \tfrac{h}{2}\mathbf{v}_i, \mathbf{v}_i + \tfrac{1}{2}\mathbf{m}_0) \tag{A-5}$$
$$\mathbf{m}_2 = h\mathbf{f}(t_i + \tfrac{h}{2}, \mathbf{x}_i + \tfrac{h}{2}\mathbf{v}_i + \tfrac{h}{4}\mathbf{m}_0, \mathbf{v}_i + \tfrac{1}{2}\mathbf{m}_1) \tag{A-6}$$
$$\mathbf{m}_3 = h\mathbf{f}(t_{i+1}, \mathbf{x}_i + h\mathbf{v}_i + \tfrac{h}{2}\mathbf{m}_1, \mathbf{v}_i + \mathbf{m}_2) \tag{A-7}$$

and

$$t_{i+1} = t_i + 1. \tag{A-8}$$

a. R/K INITIALIZATION MODULE

This module (or 'SUBROUTINE') sets up initial conditions or 'input' values for the R/K integration software. In all the sections that follow, the dimensions of variables or constants are denoted by appending subscripts and square brackets. The subscripts denote the size of dimension required explicitly. (Thus $[\mathbf{x}_3]$ means that $\mathbf{x}$ is a 'three-vector' or array of dimension three.) Undimensioned variables are shown in the input-output list without brackets. All variables in the input-output lists are of type REAL unless otherwise noted.

PURPOSE OF MODULE: This module sets up initial conditions for R/K variables and constants.

INPUT: None
OUTPUT (CONSTANTS): $[\mathbf{X}_{0_3}], [\mathbf{V}_{0_3}], \alpha_0, \alpha_2, T_0, T_{end}, H_{Step}^{R/K}$

The following values (in engineering units) are set up when this module is called (other units or more exact values can be substituted if required [see Appendix B]):

$$\alpha_0 = -.1407643 \times 10^{17}$$
$$\alpha_2 = -.71110 \times 10^{12}.$$

Reasonable test values must also be supplied for the following variables (it is assumed that the origin of the coordinate system is at the earth's center and that the z-axis is along the polar direction):

$$\mathbf{X}_0 = \left\{ \begin{array}{c} \text{appropriate} \\ \text{input values} \\ \text{(in feet)} \end{array} \right\}, \qquad \mathbf{V}_0 = \left\{ \begin{array}{c} \text{appropriate} \\ \text{input values} \\ \text{(feet per second)} \end{array} \right\}$$

$$T_0 = \text{input (in seconds)}$$
$$T_{end} = \text{input (in seconds)}$$

$$H_{Step}^{R/K} = 1 \text{ or input (in seconds).}$$

{ End of Module}

b. R/K TRAJECTORY INTEGRATION MODULE

This module (or 'SUBROUTINE') is the main R/K integration module. (See the comments given in Section II.a for variable and constant sizing conventions.)

PURPOSE OF MODULE: This module integrates position and velocity vectors by applying equations (A-2) through (A-8). The governing differential equation is equation (A-1).

INPUT (CONSTANTS): $[\mathbf{X}_{0_3}]$, $[\mathbf{V}_{0_3}]$, $T_0, T_{end}, H^{R/K}_{Step}$
OUTPUT (PRINT OUT ONLY): $T, [\mathbf{X}_3], [\mathbf{V}_3]$
VARIABLES ('LOCAL'): $T, [\mathbf{X}_3], [\mathbf{V}_3], [\mathbf{F}_3], [\mathbf{m}_{0_3}], [\mathbf{m}_{1_3}], [\mathbf{m}_{2_3}], [\mathbf{m}_{3_3}]$

First initialize time, position and velocity variables:

$$T = T_0$$
$$\mathbf{X} = \mathbf{X}_0$$
$$\mathbf{V} = \mathbf{V}_0 .$$

Next set up 'main do loop':

Begin Loop: Do until $T > T_{end}$

$$h = H^{R/K}_{Step}$$
$$t_{i+1} = T + h .$$

Next calculate the 'three-force' (i.e., 'CALL' Section II.c):

$$\mathbf{F} = \mathbf{F}_{Three}(T, \mathbf{X}, \mathbf{V}) .$$

Evaluate the right-hand side of equations (A-4), (A-5) and (A-6):

$$\mathbf{m}_0 = h\mathbf{F}$$

$$\mathbf{F} = \mathbf{F}_{Three}(T + \tfrac{h}{2}, \mathbf{X} + \tfrac{h}{2}\mathbf{V}, \mathbf{V} + \tfrac{1}{2}\mathbf{m}_0)$$

$$\mathbf{m}_1 = h\mathbf{F}$$

$$\mathbf{F} = \mathbf{F}_{Three}(T + \tfrac{h}{2}, \mathbf{X} + \tfrac{h}{2}\mathbf{V} + \tfrac{h}{4}\mathbf{m}_0, \mathbf{V} + \tfrac{1}{2}\mathbf{m}_1)$$

$$\mathbf{m}_2 = h\mathbf{F}$$

$$\mathbf{F} = \mathbf{F}_{Three}(t_{i+1}, \mathbf{X} + h\mathbf{V} + \tfrac{h}{2}\mathrm{m}_1, \mathbf{V} + \mathrm{m}_2)$$

$$\mathrm{m}_3 = h\mathbf{F}\,.$$

Update $\mathbf{X}$: Replace $\mathbf{X}$ by

$$\mathbf{X} + h\mathbf{V} + \tfrac{h}{6}(\mathrm{m}_0 + \mathrm{m}_1 + \mathrm{m}_2)\,.$$

(Operationally this is a two–step process [$\mathbf{X}_{new} = \mathbf{X} + h\mathbf{V} + \frac{h}{6}(\mathrm{m}_0 + \mathrm{m}_1 + \mathrm{m}_2)$ and $\mathbf{X} = \mathbf{X}_{new}$] that can be performed in a single step in most programming languages.)

Similarly update $\mathbf{V}$: Replace $\mathbf{V}$ by

$$\mathbf{V} + \tfrac{1}{6}(\mathrm{m}_0 + 2\mathrm{m}_1 + 2\mathrm{m}_2 + \mathrm{m}_3)\,.$$

Update T: Replace T by $T + h$ (i.e., $T = T + h$).

Next print the output variables:

PRINT OUT: T, $\mathbf{X}$ and $\mathbf{V}$.

Finally branch back to the 'main do loop entry point.'

End test on T.

{ End of Module}

c. R/K THREE-FORCE MODULE ($\mathbf{F}_{Three}$)

This module (or 'SUBROUTINE') calculates three-force.

PURPOSE OF MODULE: In the governing differential equation (equation (A-1)) a force evaluation occurs on the right-hand-side. This module evaluates that force as required by R/K numerical integration.

INPUT: $T, [\mathbf{X}_3], [\mathbf{V}_3]$
OUTPUT: $\mathbf{F}_{Three}$
VARIABLES ('LOCAL'): $[\mathbf{G}_3]$

First call the oblate gravity evaluator (i.e., 'CALL' Section II.d):

$$\mathbf{G} = \mathbf{G}_{Oblt}(\mathbf{X})\,.$$

Next set up the value of $\mathbf{F}_{Three}$ and return:

$$\mathbf{F}_{Three} = \mathbf{G}\,.$$

{ End of Module}

d. R/K OBLATE GRAVITY MODULE ($\mathbf{G}_{Oblt}$)

This module (or 'SUBROUTINE') calculates oblate gravity at a given position (i.e., the specified position vector $\mathbf{R}$). Appendix B provides a derivation of the equations implemented in this section.

PURPOSE OF MODULE: In the governing differential equation (equation (A-1)) a force evaluation occurs on the right-hand-side. This module evaluates the gravitational part of that force as required by R/K (or PBI) numerical integration.

INPUT: $[\mathbf{R}_3]$
OUTPUT: $\mathbf{G}_{Oblt}$
INPUT (CONSTANTS): α_0, α_2
VARIABLES ('LOCAL'): $[\mathbf{G}_3], R, G_A, G_B$

First calculate the magnitude of the vector $\mathbf{R}$:

$$R = \sqrt{R_1^2 + R_2^2 + R_3^2}$$

where

$$\mathbf{R} = \begin{Bmatrix} R_1 \\ R_2 \\ R_3 \end{Bmatrix}.$$

Next calculate intermediates:

$$G_A = \frac{\alpha_0 \cdot \alpha_2}{R^5}$$
$$G_B = \frac{\alpha_0}{R^3} + G_A\left(\frac{5R_3^2}{R^2} - 1\right).$$

Calculate the gravity components:

$$G_1 = R_1 \cdot G_B$$
$$G_2 = R_2 \cdot G_B$$
$$G_3 = R_3 \cdot (G_B - 2G_A),$$

where

$$\mathbf{G} = \begin{Bmatrix} G_1 \\ G_2 \\ G_3 \end{Bmatrix}.$$

Next set up the values of $\mathbf{G}_{Oblt}$ and return:

$$\mathbf{G}_{Oblt} = \mathbf{G}.$$

{End of Module}

[End of R/K Formulation]

III. PARALLEL BLOCK IMPLICIT (PBI) IMPLEMENTATION

As noted in Section II, the second order differential equation to be integrated is

$$\ddot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{v}),$$

where three-vectors are shown in boldface type (i.e., **x** and **v** are three-vectors while **f** is a vector valued function). Equation (A-1) should be contrasted with the usual equation treated directly by most numerical integration procedures:

$$\frac{dx}{dt} = f(x,t). \tag{A-9}$$

Equation (A-9) is first order (instead of second order) and it treats only the 'one-dimensional' case. For standard numerical integration schemes it is a simple matter to treat higher-dimensional cases (as in equation (A-1))—in effect all that is necessary is to replace the scalar quantities in equation (A-9) by the appropriate vector-valued ones. The other 'defect' can also be overcome easily. A second order differential equation can be recast in the form of a first order differential equation by simply treating x and $\dot{x}$ as independent quantities [which doubles the number of arguments in equation (A-9)]. The above strategy is used in the PBI implementation given in this section. This means that we must deal with a 'six-vector' implementation rather than the 'three-vector' implementation given in Section II. The following paragraph should help clarify the mathematical details. (In Section III the 'running index,' which indicates the present step of the process, is implicitly understood and as such does not appear in either the mathematical equations or the formulation [except indirectly during the update to the 'state vector'].) Whereas three-vectors have been denoted simply by boldface type, six-vectors will be denoted by boldface type with top arrows.

Let the 'generalized position vector' (i.e., the 'state vector') and 'generalized force' (i.e., the 'six-force') be defined as follows:

$$\vec{\mathbf{Y}} \equiv \begin{Bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{Bmatrix} \quad \text{and} \quad \vec{\mathbf{F}} \equiv \begin{Bmatrix} v_x \\ v_y \\ v_z \\ \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_z \end{Bmatrix}$$

where (x, y, x) are the components of position, (v_x, v_y, v_z) are the components of velocity and $(\mathbf{F}_x, \mathbf{F}_y, \mathbf{F}_z)$ are the components of (three) force. Then $\mathbf{F} = m\mathbf{a} = m\ddot{\mathbf{X}}$ becomes

$$\frac{d\vec{\mathbf{Y}}}{dt} = \vec{\mathbf{F}} = \vec{\mathbf{F}}(t, \vec{\mathbf{Y}}). \tag{A-10}$$

PBI equations are given in terms of $\vec{\mathbf{Y}}(r,s)$ where r is the node number ($r = 1,2,3,$ or 4 and is a sort of 'running index' within a given integration cycle) and s is the iteration number ($s = 0$ to start, then $s = 1,2,3$ and [finally] 4). 'Position' and time depend on the iteration number: $\vec{\mathbf{Y}} = \vec{\mathbf{Y}}(r,s)$, $t = t(r)$. Let $\vec{\mathbf{F}}(r,s) \equiv \vec{\mathbf{F}}\left(t(r), \vec{\mathbf{Y}}(r,s)\right)$, then the PBI integration equations are given below (where $\vec{\mathbf{Y}}_0$ is the starting value of $\vec{\mathbf{Y}}$ for the given integration cycle):

$$\vec{\mathbf{Y}}(r,0) = \vec{\mathbf{Y}}_0 + r \cdot h \cdot \vec{\mathbf{F}}(r,0), \tag{A-11}$$

where $r = 1,2,3,4$ and

$$\vec{\mathbf{F}}(r,0) = \vec{\mathbf{F}}(t(r), \vec{\mathbf{Y}}_0).$$

Next iterate the following set of equations for $s = 1$ to 4:

$$\vec{\mathbf{Y}}(1,s+1) = \vec{\mathbf{Y}}_0 + \tfrac{h}{720}\{251\vec{\mathbf{F}}_0 + 646\vec{\mathbf{F}}(1,s) - 264\vec{\mathbf{F}}(2,s) + 106\vec{\mathbf{F}}(3,s) - 19\vec{\mathbf{F}}(4,s)\} \tag{A-12}$$

$$\vec{\mathbf{Y}}(2,s+1) = \vec{\mathbf{Y}}_0 + \tfrac{h}{90}\{29\vec{\mathbf{F}}_0 + 124\vec{\mathbf{F}}(1,s) + 24\vec{\mathbf{F}}(2,s) + 4\vec{\mathbf{F}}(3,s) - \vec{\mathbf{F}}(4,s)\} \tag{A-13}$$

$$\vec{\mathbf{Y}}(3,s+1) = \vec{\mathbf{Y}}_0 + \tfrac{3h}{80}\{9\vec{\mathbf{F}}_0 + 34\vec{\mathbf{F}}(1,s) + 24\vec{\mathbf{F}}(2,s) + 14\vec{\mathbf{F}}(3,s) - \vec{\mathbf{F}}(4,s)\} \tag{A-14}$$

$$\vec{\mathbf{Y}}(4,s+1) = \vec{\mathbf{Y}}_0 + \tfrac{2h}{45}\{7\vec{\mathbf{F}}_0 + 32\vec{\mathbf{F}}(1,s) + 12\vec{\mathbf{F}}(2,s) + 32\vec{\mathbf{F}}(3,s) + 7\vec{\mathbf{F}}(4,s)\} \tag{A-15}$$

where

$$\vec{\mathbf{F}}_0 \equiv \vec{\mathbf{F}}(0,0).$$

This completes one full integration step. To perform the next integration step set

$$\vec{\mathbf{Y}}_0 = \vec{\mathbf{Y}}(4,5) \tag{A-16}$$

and apply the same procedure again (starting with equation (A-11)).

The reader may have observed that since six-vectors are used in the above equations one can achieve parallelism by simply allocating a 'computation node' to each of the six independent components. This is indeed one strategy for implementing parallelism; however, the favored approach arises from observing that $\vec{\mathbf{F}}(1,s), \vec{\mathbf{F}}(2,s), \vec{\mathbf{F}}(3,s)$ and $\vec{\mathbf{F}}(4,s)$ are computationally independent during the iteration of equations (A-12), (A-13), (A-14) and (A-15) and, as such, each equation can be assigned to a specific (and independent) 'computational node.'

a. PBI INITIALIZATION MODULE

This module (or 'SUBROUTINE') sets up initial conditions or 'input' values for the PBI integration software. Much of the beginning of this section is identical to Section II.a. (As in Sections II.a through II.d, the dimensions of variables or constants are denoted by appending subscripts and square brackets within this and the following sections. The subscripts indicate the size of dimension required explicitly. {Thus $[\mathbf{x}_3]$ means that $\mathbf{x}$ is a 'three-vector' or array of dimension three.} Undimensioned variables are shown in the input-output list without brackets. All variables in the input-output lists are of type REAL unless otherwise noted.)

PURPOSE OF MODULE: This module sets up initial conditions for PBI variables and constants.

INPUT: None

OUTPUT (CONSTANTS): $[\mathbf{X}_{0_3}]$, $[\mathbf{V}_{0_3}]$, $\alpha_0, \alpha_2, T_0, T_{end}, H^{PBI}_{Step}, [B_4], [C_{4\times4}]$

The following values (in engineering units) are set up when this module is called (other units or more exact values can be substituted if required [see Appendix B]):

$$\alpha_0 = -.1407643 \times 10^{17}$$
$$\alpha_2 = -.71110 \times 10^{12}.$$

Reasonable test values must also be supplied for the following variables (it is assumed that the origin of the coordinate system is at the earth's center and that the z-axis is along the polar direction):

$$\mathbf{X}_0 = \left\{ \begin{array}{c} \text{appropriate} \\ \text{input values} \\ \text{(in feet)} \end{array} \right\}, \qquad \mathbf{V}_0 = \left\{ \begin{array}{c} \text{appropriate} \\ \text{input values} \\ \text{(feet per second)} \end{array} \right\}$$

$$T_0 = \text{input (in seconds)}$$
$$T_{end} = \text{input (in seconds)}$$

Most of the above is identical to Section II.a (of course $H^{R/K}_{Step}$ is not set since it is not used in this module).

Next set up constants indigenous to the PBI integration technique itself:

$$H^{PBI}_{Step} = 1.$$

(Notice that the PBI 'step size' [H^{PBI}_{Step}] here is basically four times the R/K step size since the node number used is four here [i.e., Section II].)

$$B_1 = \tfrac{251}{720}, \quad B_2 = \tfrac{29}{90}, \quad B_3 = 3\left(\tfrac{9}{80}\right), \quad B_4 = 2\left(\tfrac{7}{45}\right)$$

$$\begin{array}{llll}
C_{1,1} = \tfrac{646}{720}, & C_{1,2} = -\left(\tfrac{264}{720}\right), & C_{1,3} = \tfrac{106}{720}, & C_{1,4} = -\left(\tfrac{19}{720}\right) \\
C_{2,1} = \tfrac{124}{90}, & C_{2,2} = \tfrac{24}{90}, & C_{2,3} = \tfrac{4}{90}, & C_{2,4} = -\left(\tfrac{1}{90}\right) \\
C_{3,1} = 3\left(\tfrac{34}{80}\right), & C_{3,2} = 3\left(\tfrac{24}{80}\right), & C_{3,3} = 3\left(\tfrac{14}{80}\right), & C_{3,4} = -3\left(\tfrac{1}{80}\right) \\
C_{4,1} = 2\left(\tfrac{22}{45}\right), & C_{4,2} = 2\left(\tfrac{12}{45}\right), & C_{4,3} = 2\left(\tfrac{32}{45}\right), & C_{4,4} = 2\left(\tfrac{7}{45}\right)
\end{array}$$

Next perform a 'consistency check:'

$$S_0 = \sum_{k=1}^{4} B_k$$

$$S_j = \sum_{k=1}^{4} C_{kj} \; ; \quad \text{for } j = 1, 2, 3, 4.$$

Print the results and exit this module:

Print S_0, S_1, S_2, S_3, S_4.

{ End of Module}

b. PBI TRAJECTORY INTEGRATION MODULE

This module (or 'SUBROUTINE') is the main PBI integration module.

PURPOSE OF MODULE: This module integrates position and velocity vectors by applying equations (A-2) through (A-7). The governing differential equation is equation (A-1).

INPUT (CONSTANTS): $[\mathbf{X}_{0_3}]$, $[\mathbf{V}_{0_3}]$, $T_0, T_{end}, [B_4], [C_{4\times4}]$

OUTPUT (PRINT OUT ONLY): T, $\left[\vec{\mathbf{Y}}_{R_{6\times4}}\right]$

VARIABLES ('LOCAL'): $[t_{R_4}]$, $\left[\vec{\mathbf{Y}}_{R_{6\times4}}\right]$, $\left[\vec{\mathbf{Y}}_{n6}\right]$, $\left[\vec{\mathbf{F}}_{n6}\right]$, $\left[\vec{\mathbf{F}}_{R_{6\times4}}\right]$, $[S_4], [\vec{\mathbf{Y}}_6], [\vec{\mathbf{F}}_6]$

First initialize time and the 'state vector' (position and velocity variables):

$$t_p = T_0$$

$$\vec{\mathbf{Y}} = \begin{Bmatrix} \mathbf{X}_0(1) \\ \mathbf{X}_0(2) \\ \mathbf{X}_0(3) \\ \mathbf{V}_0(1) \\ \mathbf{V}_0(2) \\ \mathbf{V}_0(3) \end{Bmatrix}$$

where $\mathbf{X}_0(k) \equiv [\mathbf{X}_0]_k$ for $k = 1,2,3$ (and similarly for $\mathbf{V}_0$).

Next set up 'main do loop':

Begin Loop: Do until $T > T_{end}$

$$T = t_p$$

$$h = H^{PBI}_{Step} \,.$$

Next calculate the 'six-force' (i.e., 'CALL' Section III.c)

$$\vec{\mathbf{F}} = \vec{\mathbf{F}}_{six}(T, \vec{\mathbf{Y}}) \,.$$

Set

$$t_R(k) = t_p + k \cdot h \qquad \text{for} \quad k = 1,2,3,4.$$

Set

$$\vec{\mathbf{Y}}_R(i,k) = \vec{\mathbf{Y}}(i) + k \cdot h \cdot \vec{\mathbf{F}}(i)$$

for $k = 1,2,3,4$ and $i = 1,2,3,4$.

```
Set
        $\vec{\mathbf{Y}}_n = \vec{\mathbf{Y}}\,, \qquad \vec{\mathbf{F}}_n = \vec{\mathbf{F}}.$
Next perform the iterative improvement (built into the PBI technique):
Begin loop $l = 1$ to 4 (this is an implicit loop—i.e., $l$ is not used explicitly):
    Begin loop $K = 1$ to 4:
        Set
            $\vec{\mathbf{Y}}(I) = \vec{\mathbf{Y}}(I,K)\ ; I = 1, 2, \cdots, 6$
            $T = t_R(K)$
            $\vec{\mathbf{F}} = \vec{\mathbf{F}}_{six}(T, \vec{\mathbf{Y}})$
            $\vec{\mathbf{F}}_R(I,K) = \vec{\mathbf{F}}(I)\ ; I = 1, 2, \cdots, 6.$
    End loop $K$.
    Calculate $\vec{\mathbf{Y}}_R(I,K) \equiv [\vec{\mathbf{Y}}_R]_{I,K}$ :
        $\vec{\mathbf{Y}}_R(I,K) = \vec{\mathbf{Y}}_n(I) + h \cdot B_K \cdot \vec{\mathbf{F}}_n(I) + h \sum_{J=1}^{4} C_{K,J} \vec{\mathbf{F}}_R(I,J)$
    for $I = 1, 2, 3, \cdots, 6$; $K = 1, 2, 3, 4$.
End loop $l$.
Update time and the 'state vector' (i.e., $\vec{\mathbf{Y}}$):
        $t_p = t_R(4)$
        $\vec{\mathbf{Y}}(I) = \vec{\mathbf{Y}}(I,4) \quad , I = 1, 2, 3, \cdots, 6.$
Next print the output variables:
        PRINT OUT:  $T$ and $\vec{\mathbf{Y}}_R$ .
Finally branch back to the 'main do loop entry point:'
End test on T.
```

{ End of Module}

c. PBI SIX-FORCE MODULE ($\vec{\mathbf{F}}_{six}$)

This module (or 'SUBROUTINE') calculates 'six-force.'

PURPOSE OF MODULE: If the governing set of second order differential equations (i.e., equation (A-1) in vector form) is recast as a set of first order differential equations (i.e., equation (A-9) in vector form) a 'six-dimensional' force (or 'six-force') evaluation occurs on the right-hand side. This module evaluates that force as required by the PBI numerical integration procedure.

INPUT: $[\mathbf{X}_3]$

OUTPUT: $\left[\vec{\mathbf{F}}_{six_6}\right]$

VARIABLES ('LOCAL'): $[\vec{\mathbf{Y}}_6], [\mathbf{G}_3], [\mathbf{Z}_3]$

First set up the call variables for the oblate gravity evaluator

$$\begin{Bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \mathbf{X}_3 \end{Bmatrix} = \begin{Bmatrix} \vec{\mathbf{Y}}_1 \\ \vec{\mathbf{Y}}_2 \\ \vec{\mathbf{Y}}_3 \end{Bmatrix}$$

and then actually 'CALL' Section III.d or II.d:

$$\mathbf{G} = \mathbf{G}_{Oblt}(\mathbf{X}, \mathbf{G}) \, .$$

Next set up the value of $\vec{\mathbf{F}}_{six}$ and return:

$$\vec{\mathbf{F}}_{six} = \begin{Bmatrix} \vec{\mathbf{Y}}_4 \\ \vec{\mathbf{Y}}_5 \\ \vec{\mathbf{Y}}_6 \\ \mathbf{G}_1 \\ \mathbf{G}_2 \\ \mathbf{G}_3 \end{Bmatrix} \, .$$

{ End of Module}

d. PBI OBLATE GRAVITY MODULE (G_{Oblt}—same as Section II.d)

This module (or 'SUBROUTINE') calculates oblate gravity. This section is identical to Section II.d and is not reproduced here to eliminate redundancy—reproduce the coding for Section II.d and insert it here.

[End of PBI Formulation]

APPENDIX B

OBLATE GRAVITATIONAL MODEL DERIVATION

OBLATE GRAVITATIONAL MODEL DERIVATION

This appendix derives the oblate gravitational model and constants found in Appendix A. Toward that end, first consider a spherical harmonic series expansion for the Earth's external gravitational potential.[B-1,B-2] The gravitational acceleration, **G**, can then found by taking the gradient of the potential (V):

$$\mathbf{G} = \nabla V\,. \tag{B-1}$$

Notice that 'gravitational acceleration' does not include centrifugal force terms.[B-1] (One would expect centrifugal terms not to be included in the gravitational model here since inertial coordinate frames are implicitly assumed throughout this report.) Physicists do not commonly make the distinction between gravity and gravitation that geophysicists make,[B-1] since it is always obvious from the physical context whether centrifugal forces are to be included or not. In this appendix the distinction between gravitation and gravity is maintained; however, elsewhere in the text the more customary phrase 'oblate gravity model' is used in place of the (geophysically) correct term 'oblate gravitational model.' It is also worth noting that the sign convention on the right-hand side of equation (B-1) is the one commonly used by geophysicists[B-1,B-2] and it differs from the one commonly used by physicists. The sign convention of equation (B-1) is adhered to throughout the report.

For an oblate gravitational model, it is only necessary to retain the first two dominant terms in a spherical harmonic expansion:[B-1,B-2]

$$V = -\,\frac{kM}{R}\left[1 + \frac{a^2 J_2}{R^2} P_2(\cos\phi)\right]\,, \tag{B-2}$$

where R is the magnitude of the position vector **R** (as in Section II.d of Appendix A); ϕ is the angle between the polar vector (R_3 or z direction) and the position vector **R**; P_2 is the second-order Legendre polynomial of the first kind; a, J_2 and kM are earth-related constants; and the origin of the coordinate system is at the Earth's center. The following (approximate) values of these constants will be used in the sequel:[B-2]

$$kM = 3.986 \times 10^5 \tfrac{km^3}{s^2}$$
$$a = 6378\,km$$
$$J_2 = .0010827\ .$$

B-1 Heiskanen, Weikko A. and Moritz, Helmut, *Physical Geodesy*, W. H. Freeman and Co., San Francisco, CA, 1967.

B-2 Kaplan, Marshall H., *Modern Spacecraft Dynamics & Control*, John Wiley & Sons, Inc., New York, NY, 1976.

In what follows let $x = R_1$, $y = R_2$, $z = R_3$ and $r = R$. Then

$$P_2(\cos\phi) = \tfrac{1}{4}(3\cos 2\phi + 1) = \tfrac{3}{4}(2\cos^2\phi - 1) + \tfrac{1}{4}$$

and since $\cos\phi = z/r$ it follows that

$$\frac{P_2(\cos\phi)}{r^2} = \frac{3}{2}\frac{z^2}{r^4} - \frac{1}{2r^2}\,.$$

Equation (B-2) can thus be rewritten as

$$V = kM\left\{\frac{1}{r} - a^2\,J_2\left(\frac{3}{2}\frac{z^2}{r^5} - \frac{1}{2r^3}\right)\right\}.$$

Moreover, since $\nabla r = \mathbf{R}/r$ it follows that

$$\nabla\left(\frac{1}{r^n}\right) = -\,\frac{n\mathbf{R}}{r^{n+2}}\,,$$

and thus that

$$\frac{\partial V}{\partial x} = \frac{-kMx}{r^3} - kM\,a^2\,J_2\left[\frac{3}{2}\frac{x}{r^5} - \frac{15}{2}\frac{z^2x}{r^7}\right] \tag{B-3}$$

$$\frac{\partial V}{\partial y} = \frac{-kMy}{r^3} - kM\,a^2\,J_2\left[\frac{3}{2}\frac{y}{r^5} - \frac{15}{2}\frac{z^2y}{r^7}\right] \tag{B-4}$$

$$\frac{\partial V}{\partial z} = \frac{-kMz}{r^3} - kM\,a^2\,J_2\left[\frac{3z}{r^5} + \frac{3}{2}\frac{z}{r^5} - \frac{15}{2}\frac{z^3}{r^7}\right]. \tag{B-5}$$

Equations (B-3), (B-4), and (B-5) can be simplified by introducing the following variables

$$G_A = \frac{\alpha_0\cdot\alpha_2}{r^5}$$

$$G_B = \frac{\alpha_0}{r^3} + G_A\left(\frac{5z^2}{r^2} - 1\right)$$

and set of constants

$$\alpha_0 = -kM \tag{B-6}$$

$$\alpha_2 = -\tfrac{3}{2}a^2J_2\,. \tag{B-7}$$

Thus equations (B-3), (B-4), and (B-5) can be rewritten as

$$\frac{\partial V}{\partial x} = G_1 = x\cdot G_B$$

$$\frac{\partial V}{\partial y} = G_2 = y\cdot G_B$$

$$\frac{\partial V}{\partial z} = G_3 = z\cdot(G_B - 2G_A)\,,$$

in agreement with Section II.d of Appendix A, where

$$\mathbf{G}_{Oblt} = \begin{Bmatrix} G_1 \\ G_2 \\ G_3 \end{Bmatrix} .$$

Finally, using the values of the Earth-related constants given previously in equations (B-6) and (B-7) and switching to engineering units (i.e., feet, feet per second, etc.) yields:

$$\alpha_0 = -.1407643 \times 10^{17}$$
$$\alpha_2 = -.71110 \times 10^{12} ,$$

in agreement with the values found in Sections II.a and III.a of Appendix A.

DISTRIBUTION

	Copies
DOD ACTIVITIES (CONUS)	
ATTN CODE 411 CHIEF OF NAVAL RESEARCH BALLSTON TOWER 1 800 NORTH QUINCY ST ARLINGTON VA 22217-5660	1
ATTN MATH AND SCIENCES DIV LIB OFFICE OF NAVAL RESEARCH WASHINGTON DC 20360	1
ATTN SP 00	1
SP 20	1
SP 23	1
STRATEGIC SYSTEMS PROGRAM OFFICE DEPARTMENT OF THE NAVY CRYSTAL MALL NO 3 1931 JEFFERSON DAVIS HIGHWAY ARLINGTON VA 22202	
DEFENSE TECHNICAL INFORMATION CENTER CAMERON STATION ALEXANDRIA VA 22304-6145	12
ATTN E29L (TECHNICAL LIBRARY) COMMANDER CSSDD NSWC 6703 W HIGHWAY 98 PANAMA CITY FL 32407-7001	1

DISTRIBUTION (Continued)

	Copies
NON-DOD ACTIVITIES (CONUS)	
ATTN DUDLEY KNOX LIBRARY SUPERINTENDENT U S NAVAL POSTGRADUATE SCHOOL MONTEREY CA 93943	1
PRESIDENT NAVAL WAR COLLEGE NEWPORT RI 02841	1
ATTN GIFT AND EXCHANGE DIVISION LIBRARY OF CONGRESS WASHINGTON DC 20540	4
THE CNA CORPORATION PO BOX 16268 ALEXANDRIA VA 222302-0268	1
INTERNAL	
B	1
C	1
C1	1
D	1
D1	1
D2	1
D4	1
E231	3
E282 (SWANSBURG)	1
F	1
G	1
J	1
K	1
K10	1
K104 (A R DIDONATO)	1
K12	1

DISTRIBUTION (Continued)

		Copies
INTERNAL		
K12	(CARR)	1
K12	(CUNNINGHAM)	1
K12	(OTOOLE)	1
K12	(TANNENBAUM)	1
K13		1
K13	(LAWTON)	1
K40		1
K407	(GATES)	1
K41		1
K41	(BOYLES)	1
K41	(DAVIS)	1
K41	(THOMPSON)	1
K41	(OWEN)	1
K41	(RUFTY)	5
K41	(WILKERSON)	1
K42		1
K42	(HUGHES)	1
K42	(ROBINSON)	1
K43		1
K43	(BROWN)	1
K43	(DRESHER)	1
K44		1
K44	(DAVAILUS)	1
K44	(GODIN)	1
K44	(PHAM)	1
K50		1
K505		1
K51		1
K52		1
K53		1
K54		1
K54	(GOOD)	1
K55		1
N74	(GIDEP)	1
R		1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1994	3. REPORT TYPE AND DATES COVERED FINAL

4. TITLE AND SUBTITLE Parallel Block Implicit Integration Technique for Trajectory Parallelism	5. FUNDING NUMBERS
6. AUTHOR(S) Alan E. Rufty	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Commander Naval Surface Warfare Center, Dahlgren Division (Code K41) 17320 Dahlgren Road Dahlgren, VA 22448-5100	8. PERFORMING ORGANIZATION REPORT NUMBER NSWCDD/TR-94/217
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the evaluation of a Parallel Block Implicit (PBI) integration technique in a simplified missile trajectory. This project was carried out to ascertain the suitability of PBI techniques when modest amounts of parallelism are available; that is, when 3 to 10 processors are allocated per missile trajectory. The PBI technique was first evaluated on a serial mainframe computer before it was implemented in parallel on an INMOS TRANSPUTER with four parallel central processing units. While the serial implementation of the four-node PBI technique indicated that a speedup of a factor of three to four was possible with ideal hardware, in practice only a modest gain (approximately 30 percent) was obtained because of systems-related overhead.

14. SUBJECT TERMS Parallel Block Implicit, TRANSPUTER, Runge/Kutta, parallel processing, trajectory parallelism	15. NUMBER OF PAGES 40
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and its title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

BLOCK 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If Known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2
NTIS	- Leave blank

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief ***(Maximum 200 words)*** factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code ***(NTIS only)***

Block 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of this page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.