## General Disclaimer

## One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

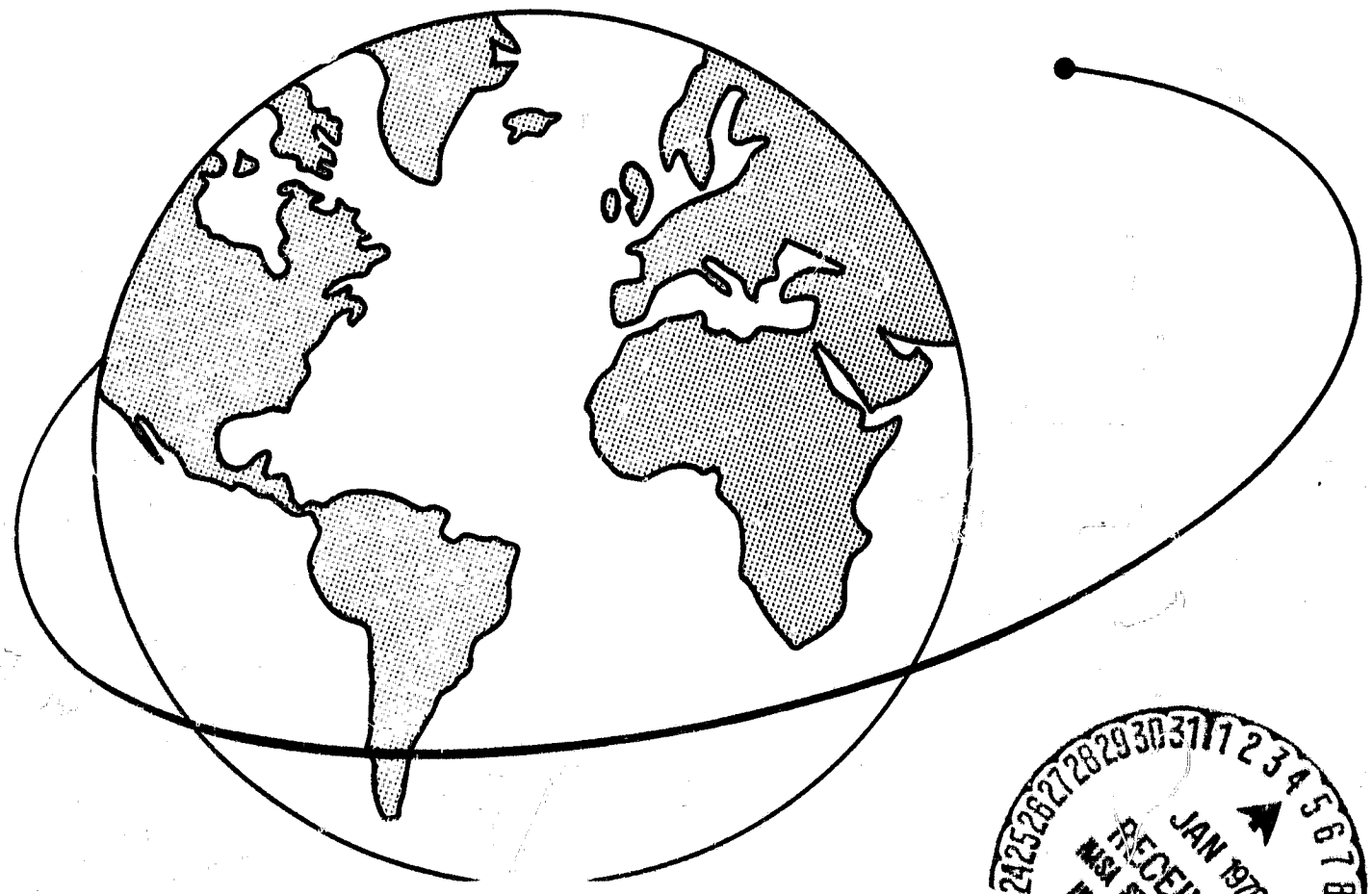
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

ELEA

# TECHNIQUES FOR MANIPULATION OF LONG POISSON SERIES

J. R. CHERNIACK



N71-14039
(ACCESSION NUMBER)
16
(PAGES)
CR-111763
(NASA CR OR TMX OR AD NUMBER)
(THRU)
G3
(CODE)
19
(CATEGORY)
FACILITY FORM 602

RECEIVED
JAN 1970
NASA STI FACILITY
INPUT BRANCH

Smithsonian Astrophysical Observatory
SPECIAL REPORT 328

NASA Grant

Research in Space Science
SAO Special Report No. 328

# TECHNIQUES FOR MANIPULATION OF LONG POISSON SERIES

J. R. Cherniack

October 29, 1970

Smithsonian Institution
Astrophysical Observatory
Cambridge, Massachusetts 02138

TABLE OF CONTENTS

## ABSTRACT

We consider some algorithms that manipulate long trigonometric series of the form $\sum_i A_i \text{ scn } (B_i)$, where $A_i$ and $B_i$ are symbolic expressions and scn means sine or cosine. Section 1 describes an efficient algorithm for converting trigonometric polynomials to trigonometric series. The results of Section 2 are independent of the order of terms of trigonometric series. Physically interesting trigonometric series are often very long. Section 3 describes efficient algorithms for Section 2 and the multiplication, addition, and differentiation of long series. These techniques are used extensively in SPASM (Hall and Cherniack, 1969).

## RÉSUMÉ

Nous considérons quelques algorithmes qui manipulent de longues séries trigonométriques de la forme $\sum_i A_i \text{ scn } (B_i)$, où $A_i$ et $B_i$ sont les expressions symboliques et scn signifie sinus ou cosinus. La première partie décrit un algorithme efficace pour convertir des polynômes trigonométriques à des séries trigonométriques. Les résultats de la deuxième partie sont indépendants de l'ordre des termes des séries trigonométriques. Les séries trigonométriques intéressantes physiquement sont souvent très longues. La troisième partie décrit des algorithmes efficaces pour la deuxième partie et la multiplication, l'addition et la différenciation des séries longues. SPASM (Hall et Cherniack, 1969) fait un usage considérable de ces méthodes.

## КОНСПЕКТ

Рассматриваются несколько алгорифмов которые манипулируют длинные тригонометрические ряды вида $\sum_i A_i \text{ scn } (B_i)$, где $A_i$ и $B_i$ являются операторными выражениями и scn означает синус или косинус. Часть 1 описывает действенный алгорифм для преобразования тригонометрического полинома в тригонометрический ряд. Результаты Части 2 являются независимыми от порядка членов в тригонометрических рядах. Физически интересные тригонометрические ряды часто бывают очень длинны. Часть 3 описывает действенные алгорифмы для Части 2 и умножение, сложение и дифференцирование длинных рядов. Эти приемы широко употребляются в СПАЗМ'е (Халл и Черниак, 1969 г.).

# TECHNIQUES FOR MANIPULATION OF LONG POISSON SERIES

J. R. Cherniack

## 1. PRELIMINARIES

A trigonometric series (TS) is a finite series of the form $\sum_i A_i \text{ scn } (B_i)$. We assume that $A_i$, $B_i$ are polynomials, although our remarks generalize quite readily to such esoteric constructs as noncommutative fields. We will always write polynomials the same way according to any fixed ordering rule, which we need not describe. We say a polynomial is 0 if it is null, or is $> 0$ or $< 0$ if it has first coefficient $> 0$ or $< 0$, respectively.

The set of Poisson series (PS) is a proper subset of the set of trigonometric series. We call a TS a PS if

1) Terms with zero coefficients do not appear.

2) No sin (or cos) terms have the same arguments.

3) Every argument is positive, except for at most one null argument for a cos term.

4) The symbol $\pi$ does not appear in any arguments (thus excluding $\sin(x) + \cos(\pi/2 - x)$ from the set of PS).

For example, $(x^2 + y^2) \sin(xy^3) - (x^2 + y^2) \sin(-xy^3)$ is not a PS; $(2x^2 + 2y^2) \sin(xy^3)$ is.

The set of PS is noteworthy because it is uniquely closed, aside from term order, under the elementary operations and because any trigonometric polynomial can be uniquely represented as a PS. Moreover, PS are prevelant in much of classical analysis, particularly in celestial mechanics.

This research was supported in part by grant NGR 09-015-002 from the National Aeronautics and Space Administration.

# 2. CONVERSION OF TRIGONOMETRIC POLYNOMIALS TO POISSON SERIES

## 2.1 Introduction

We consider some problems involved in extending symbolic polynomial manipulating systems like ALPAK (Brown, 1963), PM (Collins, 1966), or SPASM (Hall and Cherniack, 1969) tc systems that manipulate finite linear trigonometric series of the form $\sum_i A_i \text{ scn } (B_i)$, where all $A_i$ and $B_i$ are symbolic polynomials not involving sine or cosine.

## 2.2 Reduction to Canonical Form

We proceed to map polynomials $P[\text{scn } (a_1), \text{ scn } (a), \ldots, \text{ scn } (a_n)]$ into their PS equivalents. The a's are polynomials.

Let T be any term of P. By repeated application of the identities, $\cos x \cos y = [\cos (x-y) + \cos (x+y)]/2$, $\sin x \sin y = [\cos (x-y) - \cos (x+y)]/2$, $\sin x \cos y = [\sin (x-y) + \sin (x+y)]/2$, $\sin (-x) = -\sin (x)$, and $\cos (-x) = \cos (x)$. We can reduce T to a PS. Moreover, since T is the product of powers of odd and even functions, the reduced form of T must be odd or even and therefore must consist of all sines or all cosines. This reduction is not computationally efficient.

We can reduce P to a PS by applying the identities above to each term in turn, but this method is not computationally feasible without elaborate bookkeeping to minimize duplicate computation. We shall see that we can arrange for the polynomial manipulator to do the bookkeeping if we use the complex exponential representation of sine and cosine.

We recall that $\sin(x) = (e^{ix} - e^{-ix})/2i$ and $\cos(x) = (e^{ix} + e^{-ix})/2$. If we write X for $e^{ix}$, we can write $\sin(x) = (X - X^{-1})/2i$ and $\cos(x) = (X + X^{-1})/2$. We choose SPASM for the polynomial arithmetic because it allows negative exponents. In the other systems, we can manipulate pairs of polynomials representing numerators and denominators. (For example, cos (z) corresponds to the pair $(Z^2 + 1, 2Z)$.)

## 2.3 An Example

In SPASM, where variable names can be names of polynomials, we let X be the pointer to the argument of scn (x) and we do our initial computations in terms of polynomials of pointer names. An example should make things clearer. Let us find the PS equivalent of

$$\sin(a^2 + bc)\cos^2(a^2 - bc) = \sin(x_1)\cos^2(x_2)$$

$$= (1/2i)\left(e^{ix_1} - e^{-ix_1}\right)(1/2)^2\left(e^{ix_2} + e^{-ix_2}\right)^2$$

$$= \frac{1}{8i}\left[X_1 - X_1^{-1}\right]\left[X_2 + X_2^{-1}\right]^2 \qquad (1)$$

$$= \frac{1}{8i}\left[X_1X_2^2 + 2X_1 + X_1X_2^{-2} - X_1^{-1}X_2^2 - 2X_1^{-1} - X_1^{-1}X_2^{-1}\right] \qquad (2)$$

$$= \frac{1}{2i}\left[\frac{X_1X_2^2 - X_1^{-1}X_2^{-2}}{4} + \frac{X_1 - X_1^{-1}}{2} + \frac{X_1X_2^{-2} - X_1^{-1}X_2^2}{4}\right] \qquad (3)$$

$$= \frac{1}{4}\sin(x_1 + 2x_2) + \frac{1}{2}\sin(x_1) + \frac{1}{4}\sin(x_1 - 2x_2) \qquad (4)$$

$$= \frac{1}{4}\sin(3a^2 - bc) + \frac{1}{2}\sin(a^2 + bc) - \frac{1}{4}\sin(3bc - a^2)\ . \qquad (5)$$

## 2.4 Details

This procedure* generalizes readily. For each term of P, in general, the transition from step (1) to step (2) requires one polynomial multiplication for each factor, since the expansion of $(X_j \pm X_j^{-1})^n$ is known explicitly by the

*A similar method developed independently by S. R. Bourne (private communication, 1970) is described in his doctoral dissertation.

binomial theorem. The coefficient of each term is collected separately and the exponent 1/i is noted modulo 4.

The transition from step (2) to step (4) is more interesting. By inspecting the exponent of 1/i (we multiply the coefficient by -1, if appropriate), we can tell immediately whether the result is a sine or a cosine series. It is not necessary to rearrange the terms of the product $P = \Pi(X_j + X_j^{-1})^{n_j}$ as we did in step (3). A typical term of P is of the form $C\, X_1^{j_1} X_2^{j_2} \ldots X_n^{j_n}$, where the C, $j_1$, $j_2$, ..., $j_n$ are nonzero signed integers. It follows from the symmetry of P that there must exist another term of P of the form $\pm C\, X_1^{-j_1} X_2^{-j_2} \ldots X_n^{-j_n}$.

We can thus get to step (4) by ignoring each term of P whose leading factor has an exponent $\leq 0$. In our example, all the information required to recreate step (4) is contained in the terms $X_1X_2^2$, $2X_1$, and $X_1X_2^{-2}$ (P can also contain constant terms that eventually correspond to cos (0), which are treated analogously).

To complete the transition to (5), we must interpret terms of the form $c_j \prod_k X_k^{j_k} = c_j \exp(i \sum_k j_k x_k)$ with $j_1 > 0$ as scn terms. We can tell from the exponent of 1/i whether we have a sine or a cosine. The coefficient of the scn term is $2c_j$. Finally, the argument of the scn term is $\sum_k j_k x_k$, which can be computed by use of the host polynomial manipulating system.

Alternatively, we could defer step (5) and keep two running sums of the sine or cosine contributions from each term of P; then, we could perform step (5) once for each running sum when all terms of the polynomial had been processed. This is especially easy if complex polynomial arithmetic is available.

It is clear from the construction that PS afford a canonical representation for trigonometric polynomials.

# 3. PRACTICAL CONSIDERATIONS

## 3.1 Introduction

The reduction by means of both the complex exponential and the straightforward application of the addition formula have been programed in SPASM. For the conversion of eight terms given by $[\cos^2(x) + \sin^2(x)]^7 \cos(xy)$, the first method is 100 times faster than the second, and it is clear that the relative speeds diverge exponentially with the complexity of the expression.

We consider algorithms for the computer manipulation of large Poisson series — in particular, the elementary operations for multiplication, addition, and differentiation of such series. After some preliminary examples, we develop multiplication algorithms of increasing complexity. The last algorithm is significantly faster than its predecessors. Addition, differentiation, and conversion of trigonometric polynomials follow simply.

When this procedure was first used, we observed that the PS equivalent of physically interesting trigonometric series, such as those for the geopotential, were large compared with the memory of the host computer. Indeed, real problems fit into memory with difficulty, and we were forced to use secondary storage. Efficient use of secondary storage required imposition of a hash-based canonical term ordering on PS.

## 3.2 Two Examples

Let

$$P = \left[\sum_{i=0}^{N} \cos(X_i)\right]\left[\sum_{j=0}^{N} \cos(Y_j)\right] ,$$

where $X_i$, $Y_j$ are names of polynomials. If we choose the polynomials $x^i$ and $y^j$ as $X_i$ and $Y_j$, respectively $(i, j = 0, \ldots, N)$, we obtain

Example I

$$P_I = \sum_{i=0}^{N} \sum_{j=0}^{N} \cos(x^i) \cos(y^j)$$

$$= \frac{1}{2} \sum_{i=0}^{N} \sum_{j=0}^{N} [\cos(x^i + y^j) + \cos(x^i - y^j)] \quad .$$

If we choose the polynomials ix and jx as $X_i$ and $Y_j$ $(i, j = 0, \ldots, N)$, we obtain

Example II

$$P_{II} = \sum_{j=0}^{N} \sum_{i=0}^{N} \cos(ix) \cos(jx)$$

$$= \frac{1}{2} \sum_{j=0}^{N} \sum_{i=0}^{N} \cos[(i + j)x] + \cos[(i - j)x]$$

$$= \sum_{k=0}^{2N} a_k \cos(kx) \quad ,$$

where the exact values of the $a_k$ do not concern us.

Examples I and II represent two poles of the problem of collecting like terms. In example II, $2(N + 1)^2$ terms collapse to a final result of $2N + 1$ terms. In example I, there is no collection of like terms. As we develop multiplication algorithms, we shall see how they fare with $P_I$ and $P_{II}$.

## 3.3 The Basic Algorithm

We state algorithms in the spirit of Knuth (1968):

Algorithm M (Multiply Poisson series). Given two PS A and B, form their product.

M1 [Initialize]. Set RS ← 0 (running sum on secondary storage). Set CS ← 0 (core sum of terms most recently accumulated).

M2 [Loop]. For each term of A and B, perform steps M3 through M5 (A and B are on secondary storage so, for example, we make a pass over B for each term of A, buffering as appropriate, etc.).

M3 [Multiply terms]. If the A and B terms are $a_1$ scn $(a_2)$ and $b_1$ scn $(b_2)$, form their product—the two terms $P = (a_1 b_1)/2$ scn $(a_2 - b_2)$ and $Q = \pm (a_1 b_1)/2$ scn $(a_2 + b_2)$. If the arguments are not positive, negate them and change the sign of the coefficients, if necessary, while invoking the identities sin $(-x)$ = -sin x and cos $(-x)$ = cos x.

M4 [Update core sum]. Combine P and Q with CS to update CS.

M5 [Check for core overflow]. If CS is full, combine CS with RS to update RS, and set CS ← 0.

M6 [Mopup]. Combine CS with RS to update RS.

M7 [Terminate]. Collect like terms in RS if necessary. Product is RS. End.

We have begged the question of when to collect like terms. The simplest strategy is to postpone collection until step M7, merely adjoining new terms to the bottom of CS in step M4. This strategy is efficient (with respect to time) for example I, but is terrible for example II.

We can improve algorithm M's performance on example B if we collect like terms in step M4, for then we will be sorting fewer terms to do the collection of M7.

Another way to improve performance is to impose some order on terms of Poisson series. Any transitive ordering will do, so long as equality holds only if the terms are of the same trigonometric type (sin or cos) and have identical arguments (e.g., lexigraphically an argument with sin preceding cos for identical arguments). We shall write a $\textcircled{<}$ b and a $\textcircled{=}$ b for a precedes b and a equals b, respectively. Then, change step M5 to sort CS in $\bigcirc$-order on overflow and collect like terms, which must be adjacent. If we merge the sorted CS with RS, combining like terms on the fly, the resulting RS will be in order. The collection of like terms is then unnecessary in step M7. This method does a much better job on example II. However, it is not so good for example I, because a lot of time is spent in step M4 or the modified M5 in discovering that there are no like terms.

Our dilemma is that we want to collect like terms as soon as possible but we do not want to check for them unless there are some. Problems like these have been solved by hash techniques for years; we shall borrow the methods freely (Flores, 1969, pp. 85-97).

First, we associate an integer, or hash code h(P), with each polynomial P, $0 \le h(P) \le N$. Now, using h and the $\bigcirc$-ordering previously defined, we define a new $\square$-ordering on terms as follows:

If $A \equiv a_1\ \mathrm{scn}\ (a_2)$ , $B \equiv b_1\ \mathrm{scn}\ (b_2)$ ,

$A \boxed{=} B$ if $A \textcircled{=} B$ ,

$A \boxed{<} B$ if $h(a_2) < h(b_2)$

or if $h(a_2) = h(b_2)$ and $A \textcircled{<} B$ ;

i.e., terms in $\square$-order are grouped in ascending order of hash codes, and those with equal hash codes are $\bigcirc$-ordered.

Our final strategy is

Algorithm F (Multiply Poisson series A and B using hash techniques).

F1 [Initialize]. Set RS ← 0, and replace all elements of CS by Λ (nil).

F2 [Loop]. For each term of A and B, do steps F3 through F4.

F3 [Multiply terms]. Form P and Q as in step M3.

F4 [Process each term of product]. Hash (P). Hash (Q). ("Hash" is a function defined below, which hash sorts its argument into CS, merging CS with RS on overflow.)

F5 [Mopup]. Flush. (Flush combines what is left of CS with RS.) End (result is RS).

Algorithm Hash (Z) ("Hash" hash sorts its argument into CS and processes overflows of CS).

H1 [Hash the argument]. Set i ← h[arg (Z)]. (h computes hash codes for polynomial arguments.)

H2 [Check for Λ]. If $CS_i = \Lambda$, set $CS_i \leftarrow Z$ and go to H6.

H3 [Check for like terms]. If arg ($CS_i$) $\textcircled{=}$ arg (Z) and type (Z) = type ($CS_i$), go to H5 (type has a value sin or cos).

H4 [No match, no Λ]. Set i ← i + 1 (mod N) (a circular buffer), and go to H2.

H5 [Collect like terms]. Replace coef ($CS_i$) ← coef ($CS_i$) + coef (Z).

H6 [Check for overflow]. If CS is full, Flush. Return. ("Full" means some preassigned ratio (say, 1/2) of CS cells are in use.)

Algorithm Flush (Combine CS with RS).

F1 [Sort CS]. Sort. (Rearrange CS in $\square$-order.)

F2 [Merge]. Merge RS with CS, collecting like terms as in step H5. Delete terms with zero coefficients as encountered or formed. (Since RS and CS are both in $\square$-order, this step is very simple.)

F3 [Finish]. Set RS to result of merge. Replace all elements of CS by Λ. Return.

Algorithm Sort (CS consists of runs of terms separated by Λ's. To sort CS, we need only sort individual runs; indeed, that is the point of the method.)

S1 [Loop control]. For each run in CS, do S2.

S2 [Sort]. Sort the run in $\square$-order. (The first run is a special case because it may contain overflows from the last runs in circular CS; we ignore the details.)

S3 [Finished]. Return.

## 3.4 Extensions to Other Operations

Algorithm F is, in our experience, reasonably efficient for both extremes of multiplication represented by examples I and II. If we adopt the convention that $\square$-order applies to all PS, then addition is merely a merge, in the sense of F2. Conversion of trigonometric polynomials is straightforward. Since differentiation can transform sin $\longleftrightarrow$ cos, efficient conversion of results to $\square$-order requires that sin and cos of the same argument be adjacent. Our example $\bigcirc$-order of Section 3.3 accomplishes this. Conversion of arbitrarily ordered PS to $\square$-order can be done, somewhat inefficiently, by F-multiplication of the given PS by 1 cos (0).

## 4. ACKNOWLEDGMENT

The author thanks D. G. M. Anderson of Harvard for encouragement and valuable suggestions.

## 5. REFERENCES


BROWN, W. S.

1963. The ALPAK system for non-numerical algebra on a digital computer – I. Polynomials in several variables and truncated power series with polynomial coefficients. Bell Sys. Tech. Journ., vol. 42, pp. 2081-2119.

COLLINS, G. E.

1966. PM, a system for polynomial manipulation. Comm. Assoc. Com. Mach., vol. 9, no. 8, pp. 578-589.

FLORES, I.

1969. Computer Sorting. Prentice-Hall, Englewood Cliffs, N. J., 237 pp.

HALL, N. M., and CHERNIACK, J. R.

1969. Smithsonian Package for Algebraic Symbolic Manipulation – users manual. Smithsonian Astrophys. Obs. Spec. Rep. No. 291, 49 pp.

KNUTH, D. E.

1968. The Art of Computer Programming, vol. 1, Fundamental Algorithms. Addision-Wesley, Reading, Mass., 634 pp.

## BIOGRAPHICAL NOTE

JEROME R. CHERNIACK received the B.S. degree from City College of New York in 1958.

Mr. Cherniack has been with the Smithsonian Astrophysical Observatory since 1959. He is currently working with the geophysics and geodetics group, concentrating on computer sciences and applied mathematics.

## NOTICE

This series of Special Reports was instituted under the supervision of Dr. F. L. Whipple, Director of the Astrophysical Observatory of the Smithsonian Institution, shortly after the launching of the first artificial earth satellite on October 4, 1957. Contributions come from the Staff of the Observatory.

First issued to ensure the immediate dissemination of data for satellite tracking, the reports have continued to provide a rapid distribution of catalogs of satellite observations, orbital information, and preliminary results of data analyses prior to formal publication in the appropriate journals. The Reports are also used extensively for the rapid publication of preliminary or special results in other fields of astrophysics.

The Reports are regularly distributed to all institutions participating in the U. S. space research program and to individual scientists who request them from the Publications Division, Distribution Section, Smithsonian Astrophysical Observatory, Cambridge, Massachusetts 02138.