AN ABSTRACT OF THE THESIS OF

JAMES ELLIOT ROUFF for the MASTER OF SCIENCE
(Name) (Degree)

in MATHEMATICS presented on March 3, 1971
(Major) (Date)

Title: ON ERROR-BOUND CONNECTED OPTIMIZED TRIGONOMETRIC SUBROUTINES

Redacted for privacy

Abstract approved:
Harry Goheen

Polynomial evaluation algorithms proposed by E. Belaga and V. Ya Pan are applied here in designing some optimum trigonometric evaluation routines.

On Error-Bound Connected Optimized Trigonometric Subroutines

by

James Elliot Rouff

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

June 1971

APPROVED:

Redacted for privacy

Professor of Mathematics

in charge of major

Redacted for privacy

Acting Chairman of Department of Mathematics

Redacted for privacy

Dean of Graduate School

Date thesis is presented March 3, 1971

Typed by Clover Redfern for James Elliot Rouff

TABLE OF CONTENTS

ON ERROR-BOUND CONNECTED OPTIMIZED TRIGONOMETRIC SUBROUTINES

I. INTRODUCTION

A library of standard functions is an important part of any scientific programming system. Programmers working on scientific applications constantly use standard programs to evaluate simple transcendental functions like e^x, $\cos x$, and $\sqrt{x}$. Because of such widespread use high performance standards are required of these routines. The most critical requirements to consider in the design of these routines are accuracy and speed.

Accuracy levels are directly dependent on the mode and precision of the arithmetic used in the routine. In some libraries several evaluation routines are available for a single function so the function can be evaluated using single, double, and sometimes higher precision arithmetic. The user selects the appropriate routine depending on the accuracy desired in the returned value.

The execution time of any function evaluation routine can be reduced by increasing its storage requirements, and vice versa. In the past programmers attempted to achieve a good balance between speed and core requirements. However with the availability of larger and larger storage capabilities in present day machines, interest has shifted to writing very fast routines requiring extensive storage allocations.

In this paper we discuss the design of SINCOS, a computer program which evaluates the sine and cosine functions in a fast and accurate manner. The program has a variable precision feature which allows object program execution in different precisions giving it a variable accuracy capability. SINCOS is written in COMPASS for the Control Data Corporation's 3300 Computer. Use of an assembly level language assures efficient code generation.

Since any simple transcendental function can be approximated by a polynomial

$$P(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n \tag{1.1}$$

of finite degree n, the faster this approximation can be evaluated the faster the program will run. Obviously low degree polynomial approximations are desired, as every term saved in the approximation (1.1) means time save in program execution. So polynomial approximation techniques are of considerable importance in designing any evaluation routines for $P(x)$.

Chapter II is devoted to a discussion of the powerful technique of polynomial approximation using Chebyshev Polynomials. As we will see in Chapter IV, these techniques are employed extensively in the design of SINCOS. Much of the material there is from Lanczos

[10] and Stiefel [17].

The simplest method of evaluating (1.1) is to raise x to the second, third, . . . , and nth power, then multiply each power of x by its corresponding coefficient and add up all the terms. In this way, for each x_i we get $P(x_i)$ after carrying out n additions and $2n-1$ multiplications. To denote operation counts like these we will use the convention $\alpha M + \beta A$ to mean α multiplications and β additions.

A much more efficient method is the widely used recurrence scheme credited to Horner given as follows:

$$P_0(x) = a_0, \quad P_{r+1}(x) = xP_r(x) + a_{r+1}. \tag{1.2}$$

So

$$P_n(x) = (\ldots((a_0x+a_1)x+a_2)x+\ldots+a_n). \tag{1.3}$$

This technique requires $nM + nA$ operations for the evaluation of an arbitrary polynomial of degree n.

Within the last ten years algorithms have been developed by which an arbitrary polynomial of nth degree can be evaluated with less than n multiplications, that is more economically than Horner's scheme. Here we proceed, for the time being at least, on the assumption that evaluation methods requiring a decreased number of arithmetic operations over Horner's method (or at least replacing a multiplication by an addition) are faster to execute.[1]

Consider the polynomial

$$P(x) = x^7 + x^6 + \ldots + 1 = \frac{x^8 - 1}{x - 1}, \quad (x \neq 1). \qquad (1.4)$$

We can compute this polynomial using only $3M + 2A + 1D$ operations as follows:

$$p_1 = x\,x = x^2, \quad p_2 = p_1\,p_1 = x^4, \quad p_3 = p_2\,p_2 = x^8$$

$$p_4 = x - 1, \quad p_5 = p_3 - 1, \quad P(x) = p_5/p_4.$$

We expect of course to find other polynomials like (1.4) which can be handled in a very efficient manner. However these 'easy' polynomials (and corresponding computing schemes) will not concern us here. Instead, we are interested in algorithms which can be applied to more general polynomials. In fact any method will work if certain conditions are imposed on the coefficients.

We will consider computation schemes which evaluate a polynomial $P(x)$ in two stages. In the first stage, by means of operations confined to the coefficients, the polynomial is transferred to a special form. In the second stage this transformed polynomial is evaluated for specific values of its argument. It can happen that the

---

[1] Of course on machines with no floating point hardware an addition often takes about as long to execute as an average multiplication.

number of operations required by the second stage is less than that for Horner's scheme. This technique of initially conditioning the coefficients is especially useful in evaluating polynomials used again and again in function evaluation routines, since the first stage can be done before hand. From now on when we speak of the number of operations required to compute a polynomial, we refer to the number of operations in the second stage only.

In article [1] E. Belaga gives a proof of the impossibility of constructing a scheme for computing arbitrary nth degree polynomials in which the second stage requires less than $[\frac{n+1}{2}] + 1$ multiplications and n additions. We illustrate these techniques with two examples both of which require a number of operations close to this minimum bound.

Example 1: (Pan) The polynomial

$$P_5(x) = a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 \tag{1.5}$$

can be expressed in the equivalent form

$$P_5(x) = a_0\{(x+\lambda_1)[(x^2+\lambda_3)(x^2+x+\lambda_2) + \lambda_4]+\lambda_5\} \tag{1.6}$$

Equating coefficients of the corresponding powers of x, we get

$$\lambda_1 = \frac{a_1}{a_0} - 1$$

$$\lambda_2 = \frac{a_1(a_2-a_1)}{a_0^2} + \frac{a_1}{a_0} - \frac{a_3}{a_0}$$

$$\lambda_3 = \frac{a_2}{a_0} - \lambda_1 - \lambda_2$$

$$\lambda_4 = \frac{a_4}{a_0} - \lambda_2\lambda_3 - \lambda_1\lambda_3$$

$$\lambda_5 = \frac{a_5}{a_0} - \lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_4$$

Taking a simple case, let

$$P_5(x) = 2x^5 + 4x^4 + 4x^3 + 2x^2 + 2x + 2 \tag{1.7}$$

then,

$$\lambda_1 = 1, \quad \lambda_2 = 1, \quad \lambda_3 = 0, \quad \lambda_4 = 1, \quad \lambda_5 = 0$$

and

$$P_5(1) = 2[2(3+1)+0] = 16$$

$$P_5(1/2) = 2\{(3/2)[(1/4)(7/4)+1]+0\} = 69/16$$

etc. Thus if $\lambda_1, \ldots, \lambda_5$ are computed before hand (1.6) can be evaluated at the expense of only $4M + 6A$ operations.

Example 2: The 11th degree polynomial given below was used in the IBM 7090 Arcsin routine as an approximation to Arcsin x for $0 \le x \le \frac{1}{2}$ [4].

$$P(x) = x + .166667820x^{3} + .0749469671x^{5} + .0455206330x^{7} + .0239940153x^{9} + .0424173419x^{11}. \qquad (1.8)$$

Observe that $Q(x) = P(x)/x$ is a 5th degree polynomial in x^2 and can be written

$$Q(x) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5$$

$$= \{[(Az+B)^2+C](Az+B)^2+D\}(Az+E) + F, \quad (z = x^2) \qquad (1.9)$$

where $A = \sqrt[5]{a_5}$, and $C_k = a_k/A^k$ for $k = 0, 1, 2, 3, 4$.

B can be determined by solving the cubic equation

$$40B^3 - 24C_4B^2 + s(C_3+C_4^2)B + (C_2-C_3C_4) = 0. \qquad (1.10)$$

Then

$$E = C_4 - 4B$$

$$C = C_3 + 10B^2 - 4C_4B$$

$$D = C_1 - B^4 - 4B^3E - B^2C - 2BCE$$

$$F = C_0 - B^4E - B^2CE - DE$$

The algebra is a little messy but we get finally

$$P_{11}(x) = x\{[(\lambda_2+.4918761283)\lambda_2+.3697723067](\lambda_1+.6599526040)$$

$$+ .7533057075\}, \qquad (1.11)$$

where

$$z = x^2$$

$$\lambda_1 = .5315066345z$$

$$\lambda_2 = (\lambda_1-.0898244458)^2$$

Thus $P_{11}(x)$ is evaluated in $6M + 5A$ operations.

Before 'economical' methods like these can be used in a function evaluation routine the conditioning of their respective polynomial forms must be investigated. If a form is well conditioned the numerical process of evaluation gives results with accuracies close to that of normal arithemetic. We are concerned here with the capability of a particular functional form to define a function with an accuracy comparable to that of the coefficients rather than with a systematic growth of rounding error stimulated by some numerical instability of a particular function.

If the number of arithmetic operations in an evaluation routine is denoted by n, we can define a functional form $f(x)$ as badly conditioned in $[a,b]$ if kth digit floating point arithmetic gives a value for $f(x)$ with more than $\sqrt{n}$ units of error in the kth significant digit of the norm $\|f\|$ of $f(x)$ on $[a,b]$.

Using this definition with

$$\|f\| = \int_a^b [f(x)]^2 dx \tag{1.12}$$

Rice [15] investigated several economical methods applied to over a 100 polynomials (of degree 6 and 7). He found Pan's form to be ill-conditioned about 50% of the time. In comparison Horner's form was ill-conditioned in less than 10% of the cases. In fact, Rice finds a "definite positive correlation between computational efficiency and ill-conditioning".

Of course coding extended precision or fixed point arithmetic in some of the intermediate calculations will maintain accuracy. However, this technique often costs more in time than Horner's scheme and thus is unsuited for our purposes. In Chapter IV these practical problems are considered in detail.

Well conditioned economical schemes have been implemented on the Soviet Computers "Strela", BESM, M-2, M-3, and "Ural", and have resulted in noteworthy computational savings. For example the following algorithm is used to reduce the time envolved in computing a 9th degree polynomial approximation to arc tan z on the Soviet Computer BESM [11].

$$\text{arc tan } z = z\{[(Az^2+B)^2+C+Az^2][(Az^2+B)^2+D]-E\},$$

where

$$A = 0.55505873374$$

$$B = -0.65760729852$$

$$C = 0.24882437998$$

$$D = 0.17504500622$$

$$E = -0.58613261827$$

Chapter III gives some motivation behind schemes like the one above, which returns values of arc tan x correct to ten decimal places.

II. CHEBYSHEV APPROXIMATION

Consider a sequence of polynomials of increasing degree

$$P_0(x), P_1(x), P_2(x), \ldots, P_n(x), \ldots$$

where the polynomial $P_r(x)$ has exactly the degree r, for all r. Now suppose we expand a given polynomial $Q_n(x)$ in terms of the polynomials in the polynomial system.

$$Q_n(x) = C_0P_0(x) + C_1P_1(x) + \ldots + C_nP_n(x) \qquad (2.1)$$

Truncating this expansion to form the partial sum

$$C_0P_0(x) + C_1P_1(x) + \ldots + C_mP_m(x), \quad \text{with } m < n \qquad (2.2)$$

We get a polynomial of mth degree which may or may not be a good approximation of the polynomial $Q_n(x)$. The success depends on the proper choice of a polynomial system.

Unfortunately it is in general true that polynomial approximations for arbitrary functions (e.g., those for Taylor Series expansions) will exhibit an uneven error distribution. Since any arbitrary interval can be transformed (normalized) to the interval $[-1, 1]$, it suffices to examine the behavior of functions on the interval $[-1, 1]$

to determine their potential value as approximation tools.

We would like a related set of functions which has its maximum values well distributed on $[-1, 1]$. Then if we approximate an arbitrary polynomial using a linear combination of these functions the error will be distributed evenly over the interval. The cosine functions, $\cos \phi, \cos 2\phi, \ldots$ are the best candidates.

The Chebyshev Polynomials

We can transform the function $\cos n\phi$ on $[0, \pi]$ into an nth degree polynomial in x on $[-1, 1]$ using the transformation

$$\phi = \cos^{-1} x \tag{2.3}$$

The polynomials

$$T_n(x) = \cos n\phi, \quad n = 0, 1, 2, \ldots \tag{2.4}$$

expressed in the variable (2.3) are called the Chebyshev Polynomials. It is clear that,

$$T_0(x) = \cos 0 = 1$$

$$T_1(x) = \cos \phi = \cos(\cos^{-1} x) = x$$

$$T_2(x) = \cos 2\phi = \cos^2 \phi - \sin^2 \phi = 2x^2 - 1$$

In general,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \tag{2.5}$$

which follows from the well known trigonometric identity

$$\cos(n+1)\phi + \cos(n-1)\phi = 2\cos(n\phi)\cos\phi.$$

The renormalization to the range $[0,1]$ is often useful. This renormalization can be accomplished by setting

$$\cos\phi = 2x - 1 \tag{2.6}$$

or

$$x = (1+\cos\phi)/2 = \cos^2(\phi/2) \tag{2.7}$$

Then as ϕ varies from 0 to π, x varies from 0 to 1. The shifted Chebyshev Polynomials $T_k^*(x)$ (Chebyshev Polynomials adjusted to the range $[0,1]$) are still defined by

$$T_k^*(x) = \cos k\phi \tag{2.8}$$

but expressed now in the variable (2.7). The recursion relation corresponding to (2.5) is

$$T_{n+1}^*(x) = 2T_n^*(x)T_1^*(x) - T_{n-1}^*(x) \tag{2.9}$$

To develop a general formula for the powers x^n in terms of T^* polynomials use the identity

$$x^n = \cos^{2n}(\phi/2) = \left(\frac{e^{i\phi/2}+e^{-i\phi/2}}{2}\right)^{2n}$$

$$= 2/4^n[\cos n\phi + \binom{2n}{1}\cos(n-1)\phi + \ldots + \binom{2n}{n} 1/2]$$

$$= 2/4^n[T_n^*(x) + \binom{2n}{n} T_{n-1}^*(x) + \ldots + \binom{2n}{n} T_0^*] \qquad (2.11)$$

The following discussion will be limited to the shifted Chebyshev Polynomials $T_k^*(x)$ over the interval $[0,1]$. The Chebyshev Polynomials $T_k(x)$ have analogous properties over the interval $[-1,1]$.

Minimax Properties of the Chebyshev Polynomials

The following 'minimax' property accounts more than any other for the usefulness of Chebyshev Polynomials in approximation work.

Lemma. Among polynomials of degree n with leading coefficient one, $Q_n(x) = 2^{1-2n}T_n^*(x)$ has the least possible maximum value on $[0,1]$.

Proof: (by contradiction) It can be seen that $Q_n(x)$ attains its maximum value (alternately $\pm 1/2^{2n-1}$) $n+1$ times on $[0,1]$. Call these points x_i the T^* abcissas.

Assume there exists a polynomial $R_n(x)$ of nth degree with leading coefficient 1, which has a smaller maximum magnitude than $Q_n(x)$ on $[0, 1]$. Let x_i be a T^* abscissa for which $Q_n(x) = 1/2^{2n-1}$. Then

$$R_n(x_i) - Q_n(x_i) < 0$$

Similarly if x_j is a T^* abscissa for which $Q_n(x_j) = -1/2^{2n-1}$. Then,

$$R_n(x_j) - Q_n(x_j) > 0$$

It follows that $R_n(x) - Q_n(x)$ is alternately positive and negative and since there are $(n+1)$ T^* abscissa points over the interval $[0, 1]$, the function $R_n(x) - Q_n(x)$ has at least n zeros. But since $P_n(x)$ and $Q_n(x)$ both have highest coefficient equal to 1, $R_n(x) - Q_n(x)$ is a polynomial of degree $(n-1)$, and thus can have n zeros only if it is identically zero. This contradicts our assumption, therefore $R_n(x)$ does not exist.

<u>Corollary</u>. In the interval $[0, 1]$ a polynomial of nth degree with leading coefficient a_n has to assume at least once an absolute value $a_n/2^{2n-1}$.

Suppose now that we want to approximate a polynomial P_n of nth degree in some interval $[a, b]$. There exists in general a unique polynomial P_m of degree $m < n$ such that

$$\max_{[a,b]} |P_m - P_n| \tag{2.12}$$

is as small as possible. Then P_m is called a minimax approximation of P_n.

It can be shown that the T^* expansion has the property that every partial sum is the minimax approximation of the next partial sum on $[0, 1]$. Consider a polynomial of nth degree

$$Q_n(x) = a_0 + a_1 x + \ldots + a_n x^n \tag{2.13}$$

Using (2.11) expand $Q_n(x)$ in terms of the T^* polynomials to get

$$Q_n(x) = C_0 T_0^*(x) + C_1 T_1^*(x) + \ldots + C_n T_n^*(x). \tag{2.14}$$

The partial sums of this expansion are given by

$$S_m(x) = \sum_{k=1}^{m} C_k T_k^*(x), \quad m = 0, 1, 2, \ldots, n. \tag{2.15}$$

In particular,

$$S_{n-1}(x) = C_0 T_0^*(x) + C_1 T_1^*(x) + \ldots + C_{n-1} T_{n-1}^*(x) \tag{2.16}$$

and

$$Q_n(x) - S_{n-1}(x) = C_n T_n^*(x) \tag{2.17}$$

So in the interval $[0, 1]$

$$Q_n(x) - S_{n-1}(x) \leq |C_n| \qquad (2.18)$$

That is, if $Q_n(x)$ is approximated by $S_{n-1}(x)$ the error is at most equal to $|C_n|$. From (2.14) it can be seen that $Q_n(x)$ has highest coefficient $2^{2n-1}|C_n|$. Now consider $P_{n-1}(x)$ a polynomial of degree n-1. $Q_n(x) - P_{n-1}(x)$ is a polynomial of nth degree with highest coefficient $2^{2n-1}C_n$, and according to the corollary above, there exists at least one point y on $[0,1]$ such that

$$Q_n(y) - P_{n-1}(y) \geq \frac{2^{2n-1}C_n}{2^{2n-1}} = C_n \qquad (2.19)$$

Comparison of (2.18) and (2.19) implies that S_{n-1} is a minimax approximation of $Q_n(x)$ by a polynomial of degree n-1 on $[0,1]$.

Similarly $S_{n-2}(x)$ is the minimax approximation of $S_{n-1}(x)$ by a polynomial of degree n-2, and so on. We should add here that S_{n-2} is certainly not the best approximation of $S_n(x)$ by a polynomial of degree n-2 on $[0.1]$. In fact it can be shown, by the same reasoning as above, that for any polynomial P_{n-2} of degree n-2, there exists a point y in $[0,1]$ such that

$$|Q_n(y) - P_{n-2}(y)| \geq |C_{n-1}| \qquad (2.21)$$

However while $S_{n-2}(x)$ is not the minimax approximation it is a very good one, especially if we assume that the coefficients decrease rapidly in the T^* expansion (2.11). And in general if the expansion is truncated after the mth term, $S_m(x)$ will be a near minimax approximation of $Q_n(x)$ by a polynomial of degree m on [0, 1].

A comparison of the Taylor Series expansion and the Chebyshev expansion shows that the latter reduces the error of the Taylor expansion by a factor of 2^{2n-1} in the range [0, 1] (and by a factor of 2^{n-1} in the range [-1, 1]) making great accuracy possible even with a small number of terms.

Chebyshev Economization--Telescoping a Power Series

The procedure of telescoping a power series can best be illustrated by an example. If we truncate the power series expansion of $\sin\frac{\pi}{2}x$ after the 9th degree term, we get

$$P_9(x) = \frac{\pi}{2}x - \frac{(\pi/2)^3}{3!}x^3 + \frac{(\pi/2)^5}{5!}x^5 - \frac{(\pi/2)^7}{7!}x^7 + \frac{(\pi/2)^9}{9!}x^9 \qquad (2.22)$$

The maximum absolute error in the approximation of $\sin\frac{\pi}{2}x$ by $P_9(x)$ in [-1, 1] is about 3×10^{-5}. Using (2.11) we can expand (2.22) into the following Chebyshev series

$$P_9(x) = 1.13364816T_1 - .13804176T_3 + .00448040T_5 - .00006770T_7 + .00000059T_9 \qquad (2.23)$$

Here we have exactly the same function but rearranged to increase the convergence. This more rapid convergence allows us to drop terms. Since $|T_k(x)| \leq 1$ in $[-1, 1]$ the error caused by dropping a term is simply the coefficient of that term. We get an upper bound on the error at any point in the range by adding up the absolute values of the dropped terms. So that terms T_9 and T_7 can be dropped without incurring an error greater than

$$.6770 \times 10^{-4} + .59 \times 10^{-6} = 6.8 \times 10^{-5}.$$

Rearranging this truncated expansion back into a polynomial in x, we get

$$\sin \frac{1}{2}\pi x = 0.07185143x^5 - .64210139x^3 + 1.57431708x \qquad (2.24)$$

a fifth degree polynomial approximation with a maximum absolute error of 9.8×10^{-5}.

A Computer Program to Perform Chebyshev Economization

CHEBY is a computer program designed to perform Chebyshev economization (i.e., carry out the process outlined above) on $[0, 1]$. The program will accept a polynomial of arbitrary degree and a specification of the maximum tolerable error to be permitted in the economized polynomial. It outputs the coefficients of the T^*-expansion (like (2.23) for $[0, 1]$) and the coefficients of the economized polynomial in x (2.24).

CHEBY proceeds in a way completely analogous to the example worked above, arranging the numerical work to take advantage of the cumulative capabilities of the machine.

We can trace through a simple example to see how CHEBY and its main subroutines (indicated in parenthesis) proceed. Consider the fourth degree polynomial

$$P(x) = 1 + x + 2x^2 + 3x^3 + 4x^4 \quad (\text{note: } 2^{2n-1} = 128)$$

First to cancel out the dinominators in the expressions for the powers of x in terms of the T^* polynomials, weigh the coefficients (WEIGH) from the lowest to highest by the vector

$$(256/128,\ 64/128,\ 16/128,\ 4/128\ \ 1/128)$$

this gives

$$(2,\ 1/2,\ 1/4,\ 3/32,\ 1/32).$$

Now multiply (MULTIPLY) this row by the coefficients of the explicit expressions of the powers x^n $(n = 0, \ldots, 4)$ in the following manner (these coefficients are previously computed by POWERS using (2.11) and stored contiguously into memory).

$$\begin{pmatrix} 2 \\ 1/2 \\ 1/4 \\ 3/32 \\ 1/32 \end{pmatrix} \times \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ 6 & 4 & 1 & & \\ 20 & 15 & 6 & 1 & \\ 70 & 56 & 28 & 8 & 1 \end{pmatrix} = \begin{pmatrix} 2 & & & & \\ 1 & 1/2 & & & \\ 3/2 & 1 & 1/4 & & \\ 15/8 & 45/32 & 9/32 & 3/32 & \\ 35/16 & 28/16 & 7/8 & 1/4 & 1/32 \end{pmatrix}$$

And add up the columns to get (ADD)

$$P(x) = 8.56250T_0^* + 4.65625T_1^* + 1.68750T_2^* + .34375T_3^* + .03125T_4^* .$$

Now start adding absolute values of the coefficients from right to left until the sum is as near as possible without exceeding the allowable error bound, say 1.0 (TRUNCATE). Using (2.10) compute the coefficients of the T_n^* polynomials for $n = 0, 1, 2$ (COEFF) and rearrange the truncated T^* expansion back into a polynomial in x as follows

$$\begin{pmatrix} 8.56 \\ 4.66 \\ 1.69 \end{pmatrix} \times \begin{pmatrix} 1/2 & & \\ -1 & 2 & \\ 1 & -8 & 8 \end{pmatrix} = \begin{pmatrix} 4.28 & & \\ -4.66 & 9.32 & \\ 1.69 & -13.52 & 13.52 \end{pmatrix}$$

Adding the columns, we get

$$P(x) = 1.31 - 4.20x + 13.52x^2$$

Determining the Chebyshev Series Coefficients

It is not always possible to use the methods of the previous two sections to obtain numerical values of the Chebyshev series coefficients. For example if we start with a Taylor expansion the numerical work becomes unwieldy near the convergence radius of the Taylor series, since a very large number of terms is needed at the start. However other numerical methods are available, two of which are given below.

If $F(x) = \sum a_k T_k$, and we want to compute approximate values for the coefficients a_k. Let n be a positive integer and consider for $r = 0, 1, \ldots$ the numbers

$$a'_r = \frac{2}{n} \sum_{k=0}^{n} F\left(\cos\frac{\pi k}{n}\right) \cos\frac{\pi k}{n} \qquad (2.25)$$

Fike [5] shows that for n sufficiently larger than r, the difference $a'_r - a_r$ is negligibly small (approximately a_{2n-r}). So a'_r can be used to approximate a_r. Of course to use this method we must have some means of evaluating $F(x)$.

The Bessel functions can often be used to evaluate Chebyshev series coefficients. For example if $\sin\frac{\pi}{4}x$ is given by the Chebyshev series expansion

$$\sum a_{2k+1} T_{2k+1}(x),$$

it has been shown (Snyder [16]) that

$$a_{2k+1} = (-1)^k J_{2k+1}(\pi/4),$$

where J_{2k+1} is the Bessel function of the first kind of order $2k+1$, and

$$J_n(x) = \sum_{r=0}^{\infty} (-1)^r \frac{(\frac{1}{2}x)^{2r+n}}{r!(r+n)!}$$

Most Tables of Chebyshev series coefficients are compiled using these methods (e.g., Vionnet [19]).

Determining Minimax Approximations Using the Chebyshev Series Coefficients

As was shown, the polynomial $P_n(x)$ obtained by truncating the power series

$$F(x) = \sum a_k T_k(x) \tag{2.26}$$

after the term $a_n T_n(x)$ is usually a near minimax approximation to $F(x)$ in $[-1,1]$ (for polynomials of degree n). We would like to obtain $P_n^*(x)$ a polynomial of degree n that approximates $F(x)$

with a minimum absolute error in $[-1, 1]$.

The coefficients of the Chebyshev series can be used to compute accurate values for the coefficients of $P_n^*(x)$ (see Hornecker [7]). We illustrate with an example [5]. The polynomial

$$P_6(x) = C_0T_0(x) + C_2T_2(x) + C_4T_4(x) + C_6T_6(x) \tag{2.27}$$

is an approximation to $\cos\frac{1}{4}\pi x$ in $[-1, 1]$ with

$$C_0 = 1.7032638274$$

$$C_2 = -.1464366444$$

$$C_4 = .0019214493$$

$$C_6 = -.0000099650$$

Now let

$$P_6^*(x) = \frac{1}{2}C_0^*T_0(x) + C_2^*T_2(x) + C_4^*T_4(x) + C_6^*T_6(x) \tag{2.28}$$

denote the Chebyshev representation of $P_6^*(x)$. Very accurate approximations for C_0^*, C_2^*, C_4^*, C_6^* can be expressed in terms of the coefficients C_k of the series (2.27) as follows:

$$\frac{1}{2}C_0^* = \frac{1}{2}C_0 - \frac{C_{10}^4}{C_8^3} - \frac{C_{12}^2}{C_8} + \frac{3C_{10}^2C_{12}}{C_8^2} - \frac{2C_{10}C_{14}}{C_8} + C_{16}$$

$$C_2^* = C_2 + \frac{C_{10}^3}{C_8^2} - \frac{2C_{10}C_{12}}{C_8} + C_{14}$$

$$C_4^* = C_4 - \frac{C_{10}^2}{C_8} + C_{12} + \frac{2C_{10}^4}{C_8^3} - \frac{4C^2{}_{10}C_{12}}{C_8^2} + \frac{2C_{10}C_{14}}{C_8}$$

$$C_6^* = C_6 + C_{10} - \frac{C_{10}^3}{C_8^2} + \frac{2C_{10}C_{12}}{C_8} \qquad (2.29)$$

Substituting these approximations into (2.28) and rearranging terms, we get

$$P_6^*(x) = .9999999724 - .308424253x^2 + .01548499153x^4 - .0003188805x^6 \qquad (2.30)$$

and

$$|P_6^*(x) - \cos\tfrac{1}{4}\pi x| \leq .27577 \times 10^{-7}$$

It is worthwhile to note that the accuracy of the best polynomial in this case is little better than the truncated Chebyshev series. In fact

$$|P_6(x) - \cos\tfrac{1}{4}\pi x| \leq |C_8| + |C_{10}| + \ldots$$
$$\leq .27625 \times 10^{-7}$$

Hornecker shows that if C_{2r} is small compared with C_8 for $2r < 8$, then the maximum error of the best polynomial of degree 6 is given approximately by

$$|C_8| \left\{ 1 + \frac{C_{10}^2}{C_8^2} + \frac{C_{10}^2 C_{12}}{C_8^2 C_8} - \frac{C_{10}^2}{C_8^4} + \frac{C_{12}^2}{C_8^2} \right\} \cong .276 \times 10^{-7} \qquad (2.31)$$

So the accuracy of the best polynomial often is little better than that of the truncated Chebyshev series. In fact, far greater gain would come from the simple inclusion of just one more term in the series.

III. EVALUATION OF POLYNOMIALS WITH INITIAL CONDITIONING OF THE COEFFICIENTS

We would like to construct computation schemes which evaluate general polynomials $P_n(x)$ of degree n in two stages. In the first stage the coefficients of $P_n(x)$ are manipulated (conditioned), using operations confined to the coefficients, transferring the polynomial to a special form. In the second stage this special polynomial is evaluated for specific values of its argument.

It can be shown that for polynomials of general form, it is impossible to construct a scheme without initial conditioning of the coefficients, which is more efficient than Horner's scheme. So we are interested in schemes where the number of operations required to evaluate the special polynomial in the second stage is less than that for Horner's scheme. Two examples of this were cited in the introduction.

Ostrowski[2] has shown that Horner's scheme minimizes the total number of operations, counting additions equally with multiplications, for the evaluation of polynomials of degree <4. So we will only consider polynomials of degree ≥ 4 in this discussion, and it will be seen that Horner's scheme is never optimal for $n \geq 4$.

---

[2] Ostrowski, A. M. On Two Problems in Abstract Algebra Connected with Horner's Rule, Studies presented to R. von Mises, Academic Press, New York 1954, 40-48.

First we cite two general schemes by which an arbitrary polynomial of degree n with coefficients on the real axis, can be computed using only real numbers. The first and most elementary algorithm (Knuth [8]) gives best results for a large class of polynomials of degree 6, 7, or 8 and some of higher order. The second requires $[\frac{n+4}{2}]$ multiplications and $n+1$ additions for any polynomial of degree n.

Consider the polynomial

$$P_n(x) = x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n \tag{3.1}$$

where $n = 2m$. This can be written as

$$z^n + z^{n-1} + b_2z^{n-2} + \ldots + b_n \tag{3.2}$$

with $z = y+t$, $t = (a_1-1)/n$. Now if the equation

$$\alpha^{m-1} + b_3\alpha^{m-2} + b_5\alpha^{m-3} + \ldots + b_{n-1} = 0 \tag{3.3}$$

has a real root α_n, then (applying synthetic division by the quadratic factor $(z^2-\alpha_n)$)

$$P_n(x) = (z^{n-2}+z^{n-3}+C_2z^{n-4}+\ldots+C_{n-2})(z^2-\alpha_n) + \lambda_n \tag{3.4}$$

If no real root exists, apply Horner's rule to get

$$P_n(x) = ((z^{n-2}+z^{n-3}+b_2 z^{n-4}+\dots+b_{n-2})z+b_{n-1})z + b_n \qquad (3.5)$$

This process can be repeated on the reduced polynomial until finally we come to the following

$$P^*(x) = (z^2+z+C)(z^2-\alpha_4)+\lambda_4 \qquad (3.6)$$

which can be easily computed. Knuth [8] describes this algorithm with the auxiliary polynomials $g(z)$ and $p_l(z)$ $(l = 1, 2, \dots, \frac{1}{2}n)$, connected by the relations:

$$\begin{aligned} z &= y + t \\ g &= z\,z = z^2 \\ p_1 &= g + z + C \\ p_2 &= p_1\,(g-\alpha_4) + \lambda_4 \\ p_3 &= p_2\,(g-\alpha_6) + \lambda_6 \quad \text{or} \quad (p_2\,z+\alpha_6)z + \lambda_6 \end{aligned} \qquad (3.7)$$

etc, the 'or' depending on whether the reduction equations (3.3) have real roots or not. The fact that reduction equations of odd degree always have real roots implies that Horner's method is never optimal for polynomials of degree ≥ 4. In general this scheme requires $n+1$ additions and $n-r-1$ multiplications, where r is the number of reduction equations having a real root. We take the following obviously contrived example

$$P_8(x) = x^8 + x^7 + 3x^6 + 4x^5 - x^4 + 2x^3 - 3x^2 - 4x + 2 \qquad (3.1')$$

$t = 0$ which simplifies things and the first reduction equation is

$$\alpha^3 + 4\alpha^2 + 2\alpha - 4 = 0$$

Taking the root $\alpha_8 = -2$ and applying the squaring rule

$$P_8(z) = (z^6+z^5+z^4+2z^3-3z^2-2z+3)(z^2+2) - 4 \qquad (3.4')$$

The next reduction equation

$$\alpha^2 + 2\alpha - 2 = 0$$

has real roots but let's apply Horner's rule twice

$$P_3(z) = ((z^4+z^3+z^2+2z-3)z-2)z + 3 \qquad (3.5')$$

The last reduction equation is

$$\alpha + 2 = 0, \quad \text{so} \quad \alpha_4 = -2 \quad \text{and}$$

$$P_2(z) = (z^2+z-1)(z^2+2) - 1 \qquad (3.6')$$

So we have

$$P_8(z) = ((((z^2+z-1)(z^2+2)-1)z-2)z+3)(z^2+2) - 4$$

which can be evaluated using 5 multiplications and 9 additions (counting $z = y+0$).

In order to compute a polynomial $P_n(x)$ of degree $n \geq 5$, Pan [14] constructs the auxiliary polynomials $g(x)$, $h(x)$, and $p_r(x)$ where

$$g_2 = x\,x = x^2$$

$$h_2 = g_2 + x = x^2 + x$$

$$p_1 = x + \lambda_1$$

$$\left.\begin{aligned} g_4^{(s)} &= (g_2+\lambda_{4s-1})(h_2+\lambda_{4s-2}) + \lambda_{4s} \\ p_{4s+1} &= p_{4s-3}g_4^{(s)} + \lambda_{4s+1} \end{aligned}\right\} \quad (s = 1, 2, \ldots, k)$$

$$p_{4k+3} = p_{4k+1}(g_2+\lambda_{4k+2}) + \lambda_{4k+3} \tag{3.8}$$

$$P_n(x) = \sum_{i=0}^{n} a_i x^{n-i} \equiv \begin{cases} a_0 p_n & \text{for } n = 4k+1,\ 4k+3 \\ a_0 x p_{n-1} + a_n & \text{for } n = 4k+2,\ 4k+4 \end{cases}$$

As stated above the second stage of this scheme requires $[\frac{n+4}{2}]$ multiplications and $n+1$ additions. The first stage consists of determining the parameters λ_i. Pan proves the existence of real values $\lambda_1, \ldots, \lambda_n$ satisfying the Equations (3.8) when $n \geq 5$ and the coefficients $a_0, \ldots, a_n$ lie on the real axis, using the following result--stated without proof. If

$$Q_4(x) = x^4 + x^3 + \beta_1 x^2 + \beta_2 x + \beta_3 \tag{3.9}$$

then there exist real numbers λ_1, λ_2, and λ_3 such that

$$Q_4(x) = (x^2+x+\lambda_2)(x^2+\lambda_1) + \lambda_3 .$$

Using this set

$$(x^2+x+\lambda_{4s-2})(x^2+\lambda_{4s-1}) + \lambda_{4s} = x^4 + x^3 + \beta_2^{(s)}x^2 + \beta_3^{(s)} + \beta_4^{(s)} \qquad (s = 1, 2, \ldots, k) \qquad (3.10)$$

It suffices to consider the case $n = 4k+1$

$$P_{4k+1}(x) = x^{4k+1} + a_1^{(4k+1)}x^{4k} + \ldots + a_{4k+1}^{(4k+1)} \qquad (3.11)$$

Real numbers $a_1^{(4k-3)}, \ldots, a_{4k-3}^{(4k-3)}, \beta_2^{(k)}, \beta_3^{(k)}, \beta_4^{(k)}, \lambda_{4k+1}$ can always be found such that

$$P_{4k+1}(x) = P_{4k-3}(x)(x^4+x^3+\beta_2^{(k)}x^2 + \beta_3^{(k)}x + \beta_4^{(k)}) + \lambda_{4k+1} \qquad (3.12)$$

where

$$P_{4k-3}(x) = x^{4k-3} + a_1^{(4k-3)}x^{4k-4} + \ldots + a_{4k-3}^{(4k-3)}$$

Repeating this process for $s = k, k-1, \ldots, 1$ Pan constructs an iterative process for obtaining the unknown parameters $a_q^{(4s+1)}$ $(q = 1, \ldots, 4s+1)$ from the known coefficients $\lambda_{4s+1}, \beta_2^{(s)}, \beta_3^{(s)}, \beta_4^{(s)}, a_\ell^{(4s-3)}$ $(\ell = 1, 2, \ldots, 4s-3)$. Now for each $s = 1, 2, \ldots, k$ determine the real variables $\lambda_{4s-2}, \lambda_{4s-3}, \lambda_{4s}$ using Equation (3.10). Thus we have the required real functions $\lambda_i = \lambda_i(a_0, \ldots, a_n)$.

We can follow through the above algorithm with an example. Take the polynomial

$$P_9(x) = x^9 + 3x^8 + 5x^7 + 7x^6 + 9x^5 + 9x^4 + 7x^3 + 5x^2 + 3x + 1 \tag{3.11'}$$

This can be written in the form

$$P_9(x) = P_5(x)(x^4+x^3+\beta_2^{(2)}x^2 + \beta_3^{(2)}x + \beta_4^{(2)}) + \lambda_9$$

$$= (x^5+2x^4+2x^3+2x^2+2x+1)(x^4+x^3+x^2+x+1) \tag{3.12'}$$

$$=> \quad \beta_2^{(2)} = 1, \quad \beta_3^{(2)} = 1, \quad \beta_4^{(2)} = 1, \quad \text{and} \quad \lambda_9 = 0.$$

Similarly

$$P_5(x) = P_1(x)(x^4+x^3+\beta_2^{(1)}x^2+\beta_3^{(1)}x+\beta_4^{(1)}) + \lambda_5$$

$$= (x+1)(x^4+x^3+x^2+x+1) \tag{3.12''}$$

$$=> \quad \beta_2^{(1)} = 1, \quad \beta_3^{(1)} = 1, \quad \beta_4^{(1)} = 1, \quad \text{and} \quad \lambda_5 = 0.$$

To solve for $\lambda_2, \lambda_3, \lambda_4$ we get from (3.10) with $s = 1$

$$x^4 + x^3 + x^2 + x + 1 = (x^2+\lambda_3)(x^2+x+\lambda_2) + \lambda_4 \tag{3.10'}$$

$$=> \quad \lambda_2 = 0, \quad \lambda_3 = 1, \quad \lambda_4 = 1$$

and (3.10) with $s = 2$ gives

$$x^4 + x^3 + x^2 + x + 1 = (x^2+\lambda_7)(x^2+x+\lambda_6) + \lambda_8 \qquad (3.10'')$$

$$=> \qquad \lambda_6 = 0, \quad \lambda_7 = 1, \quad \lambda_8 = 1.$$

Thus we have found real functions $\lambda_i = \lambda_i (a_0, \ldots, a_n)$ for which all the equations of scheme (3.8) are satisfied. (Note: $\lambda_1 = a_1^{(1)}$.) Now for stage 2, i.e., evaluating the polynomial at specific values of its argument. For $x = 1$, we get

$$p_0 = 1$$

$$p_1 = x + \lambda_1 = 2$$

$$p_4^{(1)} = (p_0+x+\lambda_2)(p_0+\lambda_3) + \lambda_4 = 5$$

$$p_4^{(2)} = (p_0+x+\lambda_6)(p_0+\lambda_7) + \lambda_8 = 5$$

$$p_5 = p_1\, p_4^{(1)} + \lambda_5 = 10$$

$$p_9 = p_5\, p_4^{(2)} + \lambda_9 = 50$$

Therefore

$$P_9(1) = a_0\, p_9 = 50$$

a result arrived at after $6M + 10A$ operations.

The cases $n = 4k+2$, $n = 4k+3$, and $n = 4k+4$ follow easily from the basic case $n = 4k+1$. The above k-step procedure of determining four of the λ_i's at a time is not usually as practical as direct evaluation of the n nonlinear equations obtained by equating the coefficients in all of the equations (3.8). For example with $n = 5$.

$$P_5(x) = a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5$$

$$p_1 = x + \lambda_1$$

$$g_4^{(1)} = (x^2+\lambda_3)(x^2+x+\lambda_2) + \lambda_4$$

$$p_5 = p_1\, g_4^{(1)} - \lambda_5 \qquad (3.14)$$

$$\begin{aligned} P_5 = p_5\, a_0 &= a_0(p_1\, g_4^{(1)}-\lambda_5) \\ &= a_0\{(x+\lambda_1)[(x^2+\lambda_3)(x^2+x+\lambda_2)+\lambda_4]+\lambda_5\} \\ &= a_0x(\lambda_2\lambda_3+\lambda_1\lambda_3+\lambda_4) + a_0x^2(\lambda_1\lambda_2+\lambda_3+\lambda_1\lambda_3) + a_0x^3(\lambda_1+\lambda_2+\lambda_3) \\ &\quad + a_0x^4(1+\lambda_1) + a_0x^5 + a_0(\lambda_1\lambda_2\lambda_3+\lambda_1\lambda_4+\lambda_5) \end{aligned} \qquad (3.15)$$

Equating like coefficients of (3.14) and (3.15) we get

$$\lambda_1 = \frac{a_1}{a_0} - 1$$

$$\lambda_2 = \frac{a_1(a_2-a_1)}{a_0^2} + \frac{a_1}{a_0} - \frac{a_3}{a_0}$$

$$\lambda_3 = \frac{a_2}{a_0} - \lambda_1 - \lambda_2$$

$$\lambda_4 = \frac{a_4}{a_0} - \lambda_2\lambda_3 - \lambda_1\lambda_3, \qquad \lambda_5 = \frac{a_5}{a_0} - \lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_4 \qquad (3.16)$$

Belaga [1] has shown that any scheme for evaluating an arbitrary nth degree polynomial must contain at least $[n/2] + 1$ multiplications and n additions. The above scheme requires $n+1$ additions and one more than the minimum attainable number of multiplications for even n. Lifting the restriction on real parameters, Pan [13] constructs a scheme in which the lower bound on the number of operations is attained to within one addition (for $n = 2r$).

$$p_0 = 1,$$

$$p_2 = z(z+\lambda_1),$$

$$p_4 = (p_2+\lambda_2)(p_2+z+\lambda_3) + \lambda_4$$

$$p_{2s+2} = p_{2s}(p_2+\lambda_{2s+1}) + \lambda_{2s+2} \qquad (s = 2, 3, \ldots, r-1)$$

$$P_n(z) = \begin{cases} a_0 p_{2r} & \text{for } n = 2r \\ a_0 z p_{2r} + a_n & \text{for } n = 2r-1 \end{cases} \tag{3.17}$$

In Chapter IV we make considerable use of this scheme and will need to be able to determine real parameters $\lambda_1, \ldots, \lambda_n$ satisfying all the equations (3.17) for given values of the coefficients $a_0, \ldots, a_n$.

In article [13] Pan presents an operator program for the automatic determination of sets of real parameters λ_i. He lets $M^r_{i_r, i_{r+2}, \ldots, i_{n-2}}$ denote the schemes to be constructed, each requiring $\frac{1}{2}(n+1) + 1$ multiplications and from n to $n+2$ additions for a given polynomial of power n. It seems natural to select

---

$$a_j^{(s)} = a_j^{(s+2)} + (-\mu_s)a_{j-1}^{(s)} + (-\lambda_{s+1})a_{i-2}^{(s)}, \qquad j = 1, 2, \ldots, s$$

the scheme with the least number of additions. However if the preliminary manipulations of the coefficients leads to complex numbers this scheme is not suitable for computation, and one of the other schemes $M^r_{i_r, i_{r+2}, \ldots, i_{n-2}}$ must be used.

The following equations are used in Pan's operator program to determine the parameters λ_i.

$$\alpha_j^{(n)} = \frac{a_j}{a_0}, \quad j = 0, 1, \ldots, n \qquad (3.18)$$

$$\lambda_1 = \frac{\alpha_1^{(n)} - \sum_{j=0}^{(n-r-2)/2} i_{r+2j} - k_r}{[n/2]}, \qquad (3.19)$$

where

$$k_r = \begin{cases} 0 & \text{for} \quad r = 2, 3 \\ 1 & \text{for} \quad r = 5 \end{cases}$$

$$\mu_s = \lambda_i + i_s, \quad s = r, r+2, \ldots, n-2 \qquad (3.20)$$

$$\sum_{m=0}^{[(s+1)/2]} (-\lambda_{s+1})^m \sum_{i=m}^{s+1-m} C_i^m \alpha^{s+2}_{s+1-i-m} (-\mu_s)^{i-m} = 0, \qquad (3.21$$

$$s = n-2, n-4, \ldots, r,$$

where

$$\alpha_0^{(s+1)} = 1, \quad C_i^m = i!/m!(i-m)!$$

$$\alpha_j^{(s)} = \alpha_j^{(s+2)} + (-\mu_s)\alpha_{j-1}^{(s)} + (-\lambda_{s+1})\alpha_{i-2}^{(s)}, \quad j = 1, 2, \ldots, s$$

$$\lambda_{s+1}\alpha_s^{(s)} + \lambda_{s+2} = \alpha_{s+2}^{(s+2)}, \tag{3.22}$$

where

$$\alpha_0^{(s)} = \alpha_0^{(s+1)} = 1; \quad \alpha_{-1}^{(s)} = 0; \quad r \le s \le n-2; \quad s-r \text{ even};$$

$$P_r = x^r + \sum_{j=1}^{r} \alpha_j^{(r)} x^{r-j}$$

$$= \begin{cases} x^2 + \lambda_1 x + \lambda_2 & \text{for } r = 2, \\ x^3 + \lambda_1 x^2 + \lambda_2 x & \text{for } r = 3, \\ ((x^2+\lambda_1 x+\lambda_2)(x^2+(\lambda_1+1)x+\lambda_3)+\lambda_4)x + \lambda_5 & \text{for } r = 5 \end{cases} \tag{3.23}$$

Using the operator notation of Keetov the algorithm is:

$$A_1 \overset{4}{\downarrow} A_2 \overset{6}{\downarrow} A_3\, P_4 \overset{2}{\uparrow} A_5\, P_6 \overset{3}{\uparrow} A_7\, \mathscr{K} \tag{3.24}$$

where the operators proceed as follows:

A_1 - Determine $\alpha_j^{(n)}$, $(j = 0, 1, 2, \ldots, n)$

A_2 - Set the value of r; $i_r, i_{r+2}, \ldots, i_{n-2}$ (if A_2 is repeated, different sets of r; $i_r, i_{r+2}, \ldots, i_{n-2}$ are used); determine λ_1 from formula (3.19); determine $\mu_r, \mu_{r+2}, \ldots, \mu_{n-2}$ from (3.20); Set $s = n$.

A_3 - $s = s/2$; determine

$$b_m^{(s)} = \sum_{i=m}^{s+1-m} C_i^m a_{s+1-m-i}^{(s+2)} (-\mu_s)^{i-m}, \quad m = 0, 1, \ldots, [\tfrac{s+1}{2}],$$

$$C_i^m = \frac{i!}{m!(i-m)!};$$

P_4 - Go to A_2 if Equation (3.21) has no real roots λ_{s+1}; otherwise go to A_5.

A_5 - Determine the real solution λ_{s+1} of Equation (3.21); determine $a_j(s)$, $j = 1, 2, \ldots, s$; λ_{s+2} from the system of Equations (3.23).

P_6 - Go to A_3 if $s \geq r+2$; Go to A_7 if $s = r$.

A_7 - Determine $\lambda_1, \lambda_2, \ldots, \lambda_r$ from (3.23).

$\mathcal{H}$ - Stop.

We give a simple example with $n = 4$.

$$P_4(x) = x^4 + x^3 + x^2 + x + 1 \tag{3.25}$$

Choose the computing scheme M_1^2 (i.e., $r = 2$, $i_2 = 1$):

A_1 - $a_0^{(4)} = 1$, $a_1^{(4)} = 1, \ldots, a_4^{(4)} = 1$.

A_2 - $i_2 = 1$, $\lambda_1 = a_1^{(4)} - \sum_{j=0}^{0} i_2 = 1 - 1 = 0$, and $\mu_2 = \lambda + i_2 = 1$.

A_3 - $b_0^{(2)} = \sum_{i=0}^{3} C_2^0 a_{3-i}^{(4)} (-\mu_2)^i$

$= a_3^{(4)} + a_2^{(4)}(-\mu_2) + a_1^{(4)}(-\mu_2)^2 + a_0^{(4)}(-\mu_2)^3$

$= 1 + 1(-1) + 1(+1) + 1(-1) = 0$

$b_1^{(2)} = \sum_{i=1}^{2} C_i^1 a_{2-i}^{(4)} (-\mu_2)^{i-1} = a_1^{(4)} + 2a_0^{(4)}(-\mu_2) = 1 + 2(1)(-1) = -1$

P_4 - $\sum_{m=0}^{1} (-\lambda_3)^m b_m^{(2)} = 0$

$\Rightarrow b_0^{(2)} - \lambda_3 b_1^{(2)} = 0 \quad \Rightarrow \quad \lambda_3 = b_0^{(2)}/b_1^{(2)} = 0$

A_5 - Determine $a_1^{(2)}$, $a_2^{(2)}$, and λ_4 from (3.23)

$a_1^{(2)} = a_1^{(4)} + (-\mu_2)a_0^{(2)} + (-\lambda_3)a_{-1}^{(2)}$

$a_2^{(2)} = a_2^{(4)} + (-\mu_2)a_1^2 + (-\lambda_3)a_0^{(2)}$

$\Rightarrow \quad a_1^{(2)} = 1 - a_0^{(2)} = 1 - 1 = 0$

$a_2^{(2)} = 1 - a_1^{(2)} = 1 - 0 = 1,$ and

$\lambda_3 a_2^{(2)} + \lambda_4 = a_4^{(4)} \quad \Rightarrow \quad \lambda_4 = 1$

A_7 - $P_2 = x^2 + \sum_{i=1}^{2} a_i^{(2)} x^{2-i} = x^2 + a_1^{(2)}x + a_2^{(2)} = x^2 + \lambda_1 x + \lambda_2$

$\Rightarrow \quad \lambda_2 = 1$

So,

$$\lambda_1 = 0, \quad \lambda_2 = 1, \quad \lambda_3 = 0, \quad \lambda_4 = 1$$

and

$$P_4(x) = (x^2+x\lambda_1+\lambda_2)(x^2+x\lambda_1+x+\lambda_3) + \lambda_4$$

=> $$P_4(1) = (1+0+1)(1+0+1+0) + 1 = 5$$

$$P_4(2) = (4+0+1)(4+0+2+0) + 1 = 31,$$

etc.

IV. IMPLEMENTING SOME SINE AND COSINE EVALUATION ROUTINES

SINCOS is a computer program designed to evaluate the sine and cosine functions in a fast and efficient manner. The program has a variable precision feature which allows object program execution in different precisions giving it a variable accuracy capability. The absolute error of the returned value is guaranteed to be within some maximum bound passed as a parameter by the calling program. SINCOS is divided into four main subroutines; SIN, COS, SIN3, and COS3. SIN and COS are double precision routines while SIN3 and COS3 use triple precision arithmetic.

Using double precision (36 bit mantissias) arithmetic we can obtain the sine function with a maximum absolute error of 5.5×10^{-9}, and the cosine accurate to 7.0×10^{-9}. To achieve greater accuracies higher precision routines are needed. Triple precision (48 bit mantissias) arithmetic gives results with maximum absolute errors 1.6×10^{-15} and 5.7×10^{-14} for the sine and cosine respectively. SIN4 and COS4, quadriple precision (72 bit mantissias) routines, are described here but have not been implemented in the present version of the program. Ideally this precision has the capability of achieving twenty decimal places of accuracy. It will be convenient to divide our discussion into two parts--first considering SIN and COS with the higher precision routines coming later.

SIN and COS

Since SIN and COS proceed in an analogous fashion it can be assumed, unless noted otherwise, that any process described for SIN(COS) has a counterpart in COS(SIN). Our first task is the reduce the argument range to the interval $[-1, 1]$ making possible the use of Chebyshev approximation techniques.

It can be shown that having a polynomial approximation to SIN $\frac{1}{2}\pi z$ for $-1 \leq z \leq 1$ permits the evaluation of SIN x for any x. Proceed as follows:

Let $u = (2/\pi)x$, and $v = u - 4[(\frac{1}{4})(u+1)]$. Then, if $v \leq 1$, set $z = v$; otherwise, set $z = 2 - v$.

We claim, (1) $-1 \leq z \leq 1$ and (2) $\sin \frac{1}{2}\pi z = \sin x$.

Proof: $\sin x = \sin \frac{1}{2}\pi u = \sin \frac{1}{2}\pi(v+4[(\frac{1}{4})(u+1)])$

$$= \sin(\frac{1}{2}\pi v + 2\pi[(\frac{1}{4})(u+1)])$$

$$= \sin \frac{1}{2}\pi v \tag{4.1}$$

It is easily seen that $-1 \leq v \leq 3$, therefore

1) If $v \leq 1$, then $z = v$ and $-1 \leq z \leq 1$.

2) If $v > 1$, then $z = 2 - v$ and $-1 \leq z \leq 1$. And

$$\sin\frac{1}{2}\pi z = \sin\frac{1}{2}\pi(2-v) = \sin(\pi-\frac{1}{2}\pi v)$$

$$= \sin\frac{1}{2}\pi v = \sin x. \qquad (4.2)$$

In the subroutine SIN, z is calculated directly using

$$z = 1 - |4|\{((2/\pi)x+1)/4\}|-2|. \qquad (4.3)$$

Similarly having a polynomial approximation to $\cos\frac{1}{2}\pi z$ for $-1 \leq z \leq 1$ permits the evaluation of $\cos x$ for any x. Where

$$z = |4|\{((2/\pi)x+1)/4\}|-2| - 1. \qquad (4.4)$$

Now in order to find minimax polynomial approximations of the sine and cosine functions on $[-\infty, +\infty]$ we start with the Maclaurin expansions

$$\sin\frac{1}{2}\pi z = \frac{\pi}{2}z - \frac{(\pi/2)^3 z^3}{3!} + \frac{(\pi/2)^5 z^5}{5!} - \frac{(\pi/2)^7 z^7}{7!} + \dots \qquad (4.5)$$

$$\cos\frac{1}{2}\pi z = \frac{\pi}{2} - \frac{(\pi/2)^2 z^2}{2} + \frac{(\pi/2)^4 z^4}{4!} - \frac{(\pi/2)^6 z^6}{6!} + \dots \qquad (4.6)$$

On $[-1, 1]$ the series (4.5) truncated after the term $\frac{(\pi/2)^{11} z^{11}}{11!}$ will approximate $\sin\frac{1}{2}\pi z$ (and thus $\sin x$ for any x) with an absolute error no greater than 2.0×10^{-7}. Telescoping this truncated series in terms of the Chebyshev polynomials $T_k(z)$ gives

$$\sin \frac{1}{2}\pi z = 1.13364817782T_1 - 0.13807177658T_3 + 0.00449071424T_5$$
$$- 0.00006770128T_7 + 0.00000058914T_9 - 0.00000000334T_{11} \qquad (4.7)$$

The terms in T_{11}, T_9, and T_7 can be dropped causing an additional error no greater than 7.0×10^{-5}. Rearranging back into a polynomial in z we get

$$P_5(z) = .07185143z^5 - .64210139z^3 + 1.57431708z \qquad (4.8)$$

This is a very good approximation to $\sin \frac{1}{2}\pi z$ by a fifth degree polynomial. However, we are looking for $P_5^*(z)$ the best (minimax) approximation of $\sin \frac{1}{2}\pi z$ by a polynomial of degree 5.

As demonstrated in Chapter II, formulas have been developed (Hornecker [7]) by means of which very accurate numerical values for the coefficients of $P_5^*(z)$ can be obtained from the coefficients C_1, C_3, C_5 of the Chebyshev series (4.7). If we let

$$P_5^*(z) = C_1^*T_1 + C_3^*T_3 + C_5^*T_5 \qquad (4.9)$$

denote the Chebyshev series approximation of $P_5^*(z)$. Then

$$C_1^* = C_1 + \frac{C_9^3}{C_7^2} - 2\frac{C_9C_{11}}{C_7} + C_{13} - \frac{C_9^4}{C_7^3} + 3\frac{C_9^2C_{11}}{C_7^2} - \frac{C_{11}}{C_7} - 2\frac{C_9C_{13}}{C_7} + C_{15} \qquad (4.10)$$

$$C_3^* = C_3 - \frac{C_9^2}{C_7} + C_{11} + 2\,\frac{C_9^4}{C_7^3} - 4\,\frac{C_9^2 C_{11}}{C_7^2} + 2\,\frac{C_9 C_{13}}{C_7}$$

$$C_5^* = C_5 + C_9 - \frac{C_9^3}{C_7^2} + 2\,\frac{C_9 C_{11}}{C_7} \tag{4.10}$$

Substituting the known values C_k $(k = 1, \ldots, 15)$ we can solve (4.10) for the C_k^*'s and rearranging (4.9) we get

$$P_5^*(z) = .0727102z^5 - .6432292z^3 + 1.5706268z, \tag{4.11}$$

the best approximation to $\sin \frac{1}{2}\pi z$ by a polynomial of fifth degree, for $|z| \leq 1$.

These methods and others have been used in the past to find best approximating polynomials to $\sin \frac{1}{2}\pi z$ and $\cos \frac{1}{2}\pi z$. Lyusternik, et al. [11] have compiled and systematized many of the approximating formulas appearing in the literature, some of which are reproduced in Tables I and II. The coefficients there are used in the numerical algorithms which make up SIN and COS.

Now,

$$P_5^*(z)/z = .0727102z^4 - .6432294z^2 + 1.570794352 \tag{4.12}$$

is a second degree polynomial in z^2. Using Horner's rule we get

$$P_5^*(z) = ((.0727102z^2 - .6432292)z^2 + 1.570794352)z. \tag{4.13}$$

Table I. $(f(x) = \sin \frac{1}{2} \pi x)$.

Power n of Polynomial Approximation	Approximation Error	Computing Scheme	Values of Coefficients $a_0, \ldots, a_n$	Values of Parameters $\lambda_1, \ldots, \lambda_n$ (if any)	Number of Operations in Scheme	
					M	A
2	1.1×10^{-4}	Horner	0.0727102 -0.6432292 1.5706268		2	2
3	1.1×10^{-6}	Horner	-0.004362476 0.079487663 -0.645920978 1.570794852		3	3
4	5.5×10^{-9}	M_1^2 (3.17)	0.00015148419 -0.00467376557 0.07968967928 -0.64596371106 1.57079631847	-15.9265787407 327.8793687059 -39.54931896929 23336.78037853	3	5
5	1.3×10^{-8}	(3.8)	-0.00000341817225 0.0001602171343O -0.00468162023910 0.07969958728630 -0.64596409264401 1.57079632662143	-47.871774617 -43123.906284 44541.393704 192312014 1.3 110662978.0	4	6

Table II. $(f(x) = \cos \frac{1}{2} \pi x)$.

Power n of Polynomial Approximation	Approximation Error	Computing Scheme	Values of Coefficients $a_0, \ldots, a_n$	Values of Parameters $\lambda_1, \ldots, \lambda_n$ (if any)	Number of Operations in Scheme	
					M	A
2	5.9×10^{-4}	Horner	0.2239903 -1.2227967 0.9994032		2	2
4	5.0×10^{-8}	M_2^1 (3.17)	0.00085811 -0.02081046 0.25365065 -1.23369819 0.99999995	-12.62575311 441.1248973 -292.3165656 130113.4667	3	5
5	7.0×10^{-9}	(3.8)	-0.0000237888 0.0009176703 -0.0208626564 0.2536693147 -1.2337005336 0.9999999998	-39.575728914447 -24693.987676860 25610.558310990 633492228.40278 42122620.748651	4	6

And P_5^* is evaluated at the expense of $4M + 2A$ operations. Similarly $P_7^*(z)$ is computed with $5M + 3A$ operations.

Consider

$$P_9^*(z)/z = .00015148419z^8 - .004673765572z^6 + .079689679282z^4$$
$$- .64596371106z^2 + 1.57079631847 \tag{4.14}$$

a fourth degree polynomial in $z^2 = y$, which approximates $\sin \frac{1}{2}\pi z$ $(-1 \leq z \leq 1)$ with a maximum absolute error of 55×10^{-10}. Using the computing scheme (3.17) we can evaluate this polynomial in $6M + 5A$ operations as follows:

$$P_9^*(z) = .00015148419z\{(y^2+y\lambda_1+\lambda_2)(y^2+y\lambda_1+y+\lambda_3) + \lambda_4\} \tag{4.15}$$

where

$$\lambda_1 = -15.9265787407$$
$$\lambda_2 = 327.7983687059$$
$$\lambda_3 = -39.54931896929$$
$$\lambda_4 = 23336.78037853.$$

The parameters $\lambda_1, \ldots, \lambda_4$ were determined using the operator program (3.24) with $r = 2$ and $i_2 = 1$. (M_1^2).

Finally scheme (3.8) is used to evaluate

$$P^*_{11}(z)/z = -.00000341817225z^{10} + .00016021713430z^{8} - .00468162023910z^{6} + .07969958728630z^{4} - .64596409264401z^{2} + 1.57079632662143 \qquad (4.16)$$

a fifth degree polynomial in $z^2 = y$. So we write

$$P^*_{11}(z) = -.00000341817225z\{(y+\lambda_1)[(y^2+\lambda_3)(y^2+y+\lambda_2)+\lambda_4]+\lambda_5\} \qquad (4.17)$$

The parameters $\lambda_1, \ldots, \lambda_5$ are determined using the formulas (3.16).

$\lambda_1 = -47.871774617$

$\lambda_2 = -43123.906284$

$\lambda_3 = 44541.393704$

$\lambda_4 = 1923120141.3$

$\lambda_5 = 110662998.0$

Tables I and II summarize the approximating polynomials and corresponding solution schemes of SIN and COS. The operation counts refer to the requirements of the particular scheme only. In each case, SIN requires two additional multiplications to perform the transformation

$$P(z) = z\,Q(y), \quad \text{where} \quad y = z^2 \qquad (4.18)$$

while COS needs one additional multiplication to perform the change of variable $y = z^2$.

The program considers the accuracy requested by the user and finds the lowest degree approximating polynomial capable of achieving that accuracy. A variety of factors determine the error produced in evaluating a particular polynomial on a computer.

The CDC 3300 represents a double precision floating point number internally with 36 bits of fraction. Thus an ideal function evaluation routine should give results with a maximum absolute error of magnitude 2^{-36}. That is if $C(x)$ denotes for any value of x the correct value of $\cos x$ and $C^*(x)$ is the returned value of an ideal COS routine, the following should hold:

$$|C(x) - C^*(x)| \leq 2^{-36} \approx 10^{-10}. \tag{4.19}$$

Since the computations are performed in the quasi-arithmetic of the computer, results are often contaminated with rounding error in excess of the bound (4.19)[3]. Propagation of roundoff error is most damaging at points where the function is unstable. In COS the first step is to compute $y = (2/\pi)x$. Actually what we get is y^* an approximation to y. Let

$$\delta = \frac{y^*-y}{y} \tag{4.20}$$

be the relative error in y^*. Then

---

[3] On the CDC 3300 roundoff error is approximately 1×10^{-10}.

$$y^* = y(\delta+1) = (2/\pi)x(\delta+1). \tag{4.21}$$

So to find cos x, COS evaluates

$$\cos\frac{1}{2}\pi y^* = \cos x(\delta+1). \tag{4.22}$$

Thus the rounding error in y^* has the same effect as a small change in the original argument x. This change is most significant at points where $\cos x$ is unstable (e.g., at $\frac{1}{2}\pi$).

Consider the computation of

$$\cos 1.570790000 = \cos\left(\frac{1}{2}\pi - .0000063267\ldots\right) \tag{4.23}$$

using ten digit arithmetic. The first step is to find

$$y^* = (2/\pi)1.570790000.$$

Suppose that the relative error in y^* is $.1 \times 10^{-10}$, that is a change of 1 in the last digit. Then the machine evaluates cos 1.570790001 instead of 1.570790000 and we see

$$\cos 1.570790001 = .0000063257887$$

whereas,

$$\cos 1.570790000 = .0000063267949$$

So trying to compute cos 1.570790000 accurate to ten significant digits using strictly ten digit (floating point) arithmetic is not possible.

Naturally if the range-reduction calculations are performed in higher precision arithmetic the rounding error can be restrained considerably. However, for our purposes, it is impractical to do so as it would require a difficult transition from double to triple precision arithmetic. The cost in time would wipe out tenfold any gains from the economical evaluation methods. This technique is most useful in single precision routines with range reduction calculations easily done in double precision.

The economical methods used here were chosen because they are well conditioned for the approximating polynomials in SIN and COS. That is the polynomial forms in each scheme define the function with an accuracy comparable to the coefficients. Considering our remarks in the introduction, we take comparable to mean no more than $\sqrt{n} + r$ units of error in the kth significant digit of the norm $\|f\|$ of $f(x)$ on $[-1, 1]$. Where a polynomial approximation with an absolute error of r units in the kth digit is used, and n is the number of arithmetic operations performed (assuming kth digit floating point arithmetic capability).

To see this consider Pan's scheme as used to evaluate the 9th degree approximating polynomial (4.14) for $\sin \frac{1}{2}\pi z$ at $z = .1$.

$$P(z) = .00015148419z\{(y^2+y\lambda_1+\lambda_2)(y^2+y\lambda_1+y+\lambda_3)+\lambda_4\} \tag{4.24}$$

where $y = z^2 = .01$, and $\lambda_1, \ldots, \lambda_4$ are as given in (4.15). The calculation proceeds as follows:

$$\begin{array}{lcl}
y\,\lambda_1 & & -.159265787407 \\
\underline{+\;y^2} & & \underline{+.0001} \\
T_1 & = & -.159165787407 \\
 & & \\
\underline{+\;\lambda_2} & & \underline{+327.7983687059} \\
T_2 & = & 327.6391029185 \\
 & & \\
T_1 + y & & -.149165787407 \\
 & & \\
\underline{+\;\lambda_3} & & \underline{-39.54931896929} \\
T_3 & = & -39.69848475669 \\
 & & \\
T_2\,T_3 & & -13006.8524027669 \\
 & & \\
\underline{+\;\lambda_4} & & \underline{+23336.18037853} \\
T_4 & = & 10329.92797577
\end{array} \qquad (4.25)$$

$$P(.1) = .000015148419\;T_4 = .1564344652 \quad \text{(rounded to 10 digits)}$$

whereas the true value is,

$$P(.1) = \sin \frac{1}{2}\pi(.1) = \sin 9 = .1564344650\ldots$$

So we have an error of 2 units in the tenth decimal place.

The scheme requires $5A + 5M = 10$ operations and the approximating polynomial (4.14) gives $\sin \frac{1}{2}\pi z$ with a maximum absolute error of 5.5×10^{-9}. However at $z = .1$ (4.14) is accurate to ten decimal digits. Thus we would accept any value with an error of less than $\sqrt{10} = 3.16$ units in the tenth decimal place.

Clearly error suppression occurs above as the least significant digits of each intermediate product are ignored in an immediate summing with a larger number. The fact that $-1 \leq z \leq 1$ assures these products will always be the smaller of the two. Horner's scheme when applied to our approximating polynomials has a similar effect (in fact values are often accurate to the full precision of the calculations) since the polynomial coefficients decrease rapidly in magnitude and the range is restricted to $[-1, 1]$.

There can remain one important drawback to implementing even these well conditioned methods. On machines with only one floating point arithmetic register the time saved in executing a reduced number of arithmetic operations often is wiped out by the necessity of storing and retrieving intermediate results. The CDC 3300, the machine on which this program is run, is a case in point. Store instructions (3.8 μsec) take about one third as long to execute as an average floating point addition (11 μsec). Thus counts of arithmetic operations alone are not at all sufficient in estimating the relative speeds of different evaluation methods. In fact the 'economical' methods described thus far are (at least for low degree polynomials) almost never as fast as Horner's method on the CDC 3300. The following table, based on the execution times published by the respective companies, compares several of the methods used here (times given in μsec).

Degree	Method	CDC 3300	IBM 360/75
4	Horner	108.	12.48
4	(3.17), M_1^2	118.2	11.23
5	Horner	135.0	15.50
5	(3.17), M_0^2	138.0	13.44
5	(3.8)	154.0	14.37
6	Horner	162.	18.52
6	(3.17), M_{10}^2	163.8	15.31
6	(3.8)	181.	17.5

And it can be seen that 'economical' methods are only advantages on machines with multiple arithmetic registers.

SIN3 and COS3

SIN3 and COS3 are evaluation routines using triple precision arithmetic. It is clear that the user sacrifices speed here in order to obtain greater accuracies. This is due mainly to the limitations of the machine being used (i.e., single arithmetic register with no extended precision hardware).

It can be shown that having a polynomial approximation to $\sin \frac{1}{4}\pi z$ and $\cos \frac{1}{4}\pi z$ for $-1 \leq z \leq 1$ permits the evaluation of $\sin x$ and $\cos x$ for any x. For example to find $\cos 240$, let $z = .666\ldots$ and

$$\cos 240 = - \sin \frac{1}{4}\pi z \cong - \sin 30 = - \frac{1}{2} .$$

The first ten coefficients of the Chebyshev series expansions for $\cos \frac{1}{4}\pi z$ and $\sin \frac{1}{4}\pi z$ were calculated to 24 decimal places by Vionnet [19]. From these minimax polynomial approximation for $\sin \frac{1}{4}\pi z$ and $\cos \frac{1}{4}\pi z$ are found as before for $\sin \frac{1}{2}\pi z$ and $\cos \frac{1}{2}\pi z$.

$$\sin \frac{1}{4}\pi z = 0.7263756767T_1 - 0.0194200290T_3$$

$$+ 0.0001516929T_5 - 0.0000005606T_7 \qquad (4.26)$$

Let

$$P_7^*(z) = C_1^*T + C_3^*T + C_5^*T + C_7^*T \qquad (4.27)$$

be a Chebyshev series approximation to $P_7^*(z)$ a polynomial which approximates $\sin \frac{1}{4}\pi z$ with minimax absolute error in $[-1, 1]$ (for polynomials of degree ≤ 7). Now using the formulas of Hornecker [7].

$$C_1^* = C_1 - \frac{C_{11}^4}{C_9^3} + 3 \frac{C_{11}^2 C_{13}}{C_9^2} - \frac{C_{13}^2}{C_9} - 2\frac{C_{11} C_{15}}{C}$$

$$C_3^* = C_3 + \frac{C_{11}^3}{C_9^2} - 2\frac{C_{11} C_{13}}{C_9} + C_{15}$$

$$C_5^* = C_5 - \frac{C_{11}^2}{C_9} + C_{13} + 2 \frac{C_{11}^4}{C_9^3} - 4 \frac{C_{11}^2 C_{13}}{C_9^2} + 2\frac{C_{11} C_{15}}{C_9} \qquad (4.28)$$

$$C_7^* = C_7 + C_{11} - \frac{C_{11}^3}{C_9^2} + 2\,\frac{C_{11}C_{13}}{C_9} \qquad (4.28)$$

And rounding to 10 decimal places

$$C_1^* = 0.7263756767$$

$$C_3^* = -0.0194200290$$

$$C_5^* = 0.0001516929$$

$$C_7^* = -0.0000005606$$

This example is interesting because to 10 decimal places $C_1^* = C_1, \ldots, C_7^* = C_7$. So the accuracy of the best possible polynomial approximation is no better than the truncated Chebshev series (4.26) unless extended precision is used to represent the coefficients and perform the calculations.

This follows from the remarks made at the end of Chapter II. In fact the maximum error of the best polynomial approximation of degree 7 is given approximately by

$$|C_9|\{1+ \frac{C_{11}^2}{C_9^2} + \frac{C_{11}^2 C_{13}}{C_9^3} - \frac{C_4^4}{C_9^4} + \frac{C_{13}^2}{C_9^2}\} \cong .12 \times 10^{-8} \qquad (4.29)$$

whereas the absolute value of the truncation error in (4.26) cannot be greater than

$$|C_9| + |C_{11}| + |C_{13}| + \ldots = .120701\ldots \times 10^{-8}.$$

Tables III and IV summarize the approximating polynomials and corresponding solution schemes of SIN3 and COS3, as well as some proposed methods which could be implemented in higher precision routines.

In the case of 6th degree polynomials (P_6) the scheme (3.8) is actually applied to a polynomial p_5 where

$$P_6 = a_0 x\, p_5 + a_6. \tag{4.30}$$

If we denote

$$p_5 = a_0' x^5 + a_1' x^4 + a_2' x^3 + a_3' x^2 + a_4' x + a_5' \tag{4.31}$$

then clearly $a_0' = 1$ and the equations (3.16) can be simplified as follows:

$$\begin{aligned}
\lambda_1 &= a_1' - 1 \\
\lambda_2 &= a_1'(a_2'-a_1') + a_1' - a_3' \\
\lambda_3 &= a_2' - \lambda_1 - \lambda_2 \\
\lambda_4 &= a_4' - \lambda_1\lambda_3 - \lambda_2\lambda_3 \\
\lambda_5 &= a_5' - \lambda_1\lambda_2\lambda_3 - \lambda_1\lambda_4
\end{aligned} \tag{4.32}$$

For example we have

Table III. $(f(x) = \sin \frac{1}{4} \pi x)$.

Power n of Polynomial Approximation	Approximation Error	Computing Scheme	Values of Coefficients $a_0, \ldots, a_n$	Values of Parameters $\lambda_1, \ldots, \lambda_n$	Number of Operations in Scheme	
					M	A
4	1.7×10^{-12}	M_2^1	0.0000003085630	-59.760858722530	3	5
			-0.0000365714167	10787.503271213		
		(3.17)	0.0024903924781	-6228.1658273871		
			-0.0807455118150	6.9731700587889		
			0.7853981633788			
5	1.6×10^{-15}	(3.8)	-0.0000000017347987	-181.6167385875952063	4	6
			0.0000003133336833	-2405341.757322552383		
			-0.0000365761873953	2426607.196290946814		
			0.0024903945652995	5837306874943.196260		
			-0.0807455121876694	88041311079.7904		
			0.7853981633974265			
6	1.2×10^{-18}	(3.8)	0.0000000000068723070		5	7
			-0.0000000017571336497	-256.6832297654921411		
			0.0000003133616020111	-6401944.657074794301		
			-0.0000365762041465937	6447799.070733526369		
			0.0024903945701852502	41280470233822.88046		
			-0.0807455121882800902	506089259785.87		
			0.7853981633974482911			

Table IV. $(f(x) = \cos \frac{1}{4} \pi x)$.

Power n of Polynomial Approximation	Approximation Error	Computing Scheme	Values of Coefficients $a_0, \ldots, a_n$	Values of Parameters $\lambda_1, \ldots, \lambda_n$	Number of Operations in Scheme	
					M	A
4	4.7×10^{-11}	M_2^1 (3.17)	0.000003529804 -0.000325938600 0.015854325237 -0.308425135160 0.999999999953	-46.66950402911 22771.3247524 -20411.13819954 4.65071958357	3	5
5	5.7×10^{-14}	(3.8)	-0.000000024268543 0.000003590475595 -0.000325991687588 0.015854344197125 -0.308425137530042 0.999999999999944	-148.94771960558159 -1356083.7065525327 1369665.3390273788 1857597567058.0802 32238352946.95482	4	6
6	4.8×10^{-17}	(3.8)	0.000000000113654754 -0.000000024609507280 0.000003590859180060 -0.000325991886483649 0.015854344243741571 -0.308425137534037837 0.999999999999999953	-217.528622111113492006 -4019938.982748709913 4051750.959061588123 1628881249617 8.450338 219354189306.77765431	5	7

$$
\begin{aligned}
\cos \frac{1}{4}\pi z \cong\ & 0.0000000001136547 54y^6 - 0.000000024609507280y^5 \\
& + 0.000003590859180060y^4 - 0.000325991886483649y^3 \\
& + 0.015854344243741571y^2 - 0.308425137534037837y \\
& + 0.999999999999999953, \qquad (y = z^2)
\end{aligned}
$$

Setting the right hand side above equal to $a_0 y\, p_5 + a_6$ gives

$$
\begin{aligned}
p_5 = y^5 & - 216.5286221111349 2006y^4 + 31594.447690767075172y^3 \\
& - 2868264.414910871216 2y^2 + 139495654.03785547765y \\
& - 2713702037.7875072168
\end{aligned}
$$

And using (4.32), with these coefficients, we get the parameters $\lambda_1, \ldots, \lambda_5$ given in Table IV. The approximating polynomial is calculated in the following manner:

$$\cos \frac{1}{4}\pi z \cong a_0 y\{(y+\lambda_1)[(y+\lambda_3)(y^2+y+\lambda_2)+\lambda_4]+\lambda_5\} + a_6$$

which requires 6M + 7A operations (counting the change of variable $z = y^2$) and gives results with a maximum absolute error of 4.8×10^{-17}.

BIBLIOGRAPHY

1. Belaga, E.G. 1958. Some problems involved in the calculation of polynomials. Dok. Akad. Nauk SSSR 123:775-777.

2. Clenshaw, C. 1962. Chebyshev series for mathematical functions. NPL Math. Tables, Vol. 5. H.M. Stationery Office, London.

3. Cody, W.J., Jr. 1967. Letter to the editor. Communications of the ACM. 10:531.

4. Fike, C.T. 1967. Methods of evaluating polynomial approximations in function evaluation routines. Communications of the ACM 10:175-178.

5. Fike, C.T. 1968. Computer evaluation of mathematical functions. Englewood Cliffs, Prentice-Hall.

6. Hastings, C., Jr. 1955. Approximations for digital computers. Princeton, Princeton University Press.

7. Hornecker, G. 1958. Evaluation approchee de la meilleure approximation polynomiale d'ordre n de f(x) sur un segment fini [a,b]. Chiffres 1:157-169.

8. Knuth, D.E. 1962. Evaluation of polynomials by computer. Communications of the ACM 5:595-599.

9. Lanczos, C. 1952. Tables of Chebyshev polynomials $S_n(x)$ and $C_n(x)$. N.B.S. Applied Mathematics Series, Vol. 9. U.S. Government Printing Office, Washington D.C.

10. Lanczos, C. 1956. Applied Analysis. Englewood Cliffs, Prentice-Hall. 530.

11. Lyusternik, L.A., Chervonenkis, O.A. and Yanpol'skii, A.R. 1965. Handbook for computing elementary functions. New York, Pergamon Press. English translation by G.J. Tee.

12. Motzkin, T.S. 1955. Evaluation of polynomials. Bulletin of the American Mathematical Society 61:163.

13. Pan, V. Ya. 1962. Computation of polynomials by schemes with initial conditioning of the coefficients and a program for automatic determination of the parameters. Zh. Vychisl. Mat. i Mat. Fiz. 2:133-140 (Translated in USSR Computational Mathematics and Physics 137-146. 1963)

14. Pan, V. Ya. 1966. Methods of computing values of polynomials. Russian Mathematical Surveys 21:105-136.

15. Rice, J. 1965. On the conditioning of polynomial and rational forms. Numerische Mathematik 7:426-435.

16. Snyder, M.A. 1966. Chebyshev methods in numerical approximation. Englewood Cliffs, Prnetice-Hall.

17. Stiefel, E.L. 1963. An introduction to numerical mathematics. New York, Academic Press. English translation by W.C. Rheinboldt and C.J. Rheinboldt.

18. Todd, J. 1955. Motivation for working in numerical analysis. Communications on Pure and Applied Mathematics 8:96-116.

19. Vionnet, M. 1959. Approximation de Tchebycheff d'ordre n des functions sin x, sin x/x, cos x, et exp x. Chiffres 2:77-96.

20. Control Data Corporation 60057600C. 1968. 3100/3200/3300/3500 Computer systems FORTRAN reference manual. Control Data Corporation.

21. Control Data Corporation 60184200. 1967. 3100/3200/3300/3500 Computer systems COMPASS training manual. Control Data Corporation.

22. International Business Machines Corporation C28-6583-0. 1965. System/360 basic programming support, FORTRAN IV, 360P-F0-031, Programmers guide. International Business Machines Corporation.

23. International Business Machines Corporation C28-6956-0. 1965. System/360 operating system FORTRAN IV (E) library subprograms. International Business Machines Corporation.

APPENDICES

```
INPUT POLYNOMIAL
   1+0X1-0.5X2+0X3+0.0416666666X4+0X5-0.0013888888X6+0X7+0.0000248015X8


COEFFICIENTS OF THE CHEBYSHEV EXPANSION
+1.647169571
-.2322992839
-.05371504891
+.002458275682
+.0002821392458
-.000007714166552
-5.873425537E-007
+1.211010742E-008
+7.568817138E-010


COEFFICIENTS OF THE REARRANGED POLYNOMIAL
+1.000000000
-1.015042217E-011
-.4999999999
+1.030286966E-012
+.04166666660
-1.172395514E-013
-.001388888800
+0.000000000
+0.00002480150000


MAXIMUM ABSOLUTE ERROR = 0.0000000001


INPUT POLYNOMIAL
   1+0X1-0.5X2+0X3+0.0416666666X4+0X5-0.0013888888X6+0X7+0.0000248015X8


COEFFICIENTS OF THE CHEBYSHEV EXPANSION
+1.647169571
-.2322992839
-.05371504891
+.002458275682
+.0002821392458
-.000007714166552
-5.873425537E-007
+1.211010742E-008
+7.568817138E-010


COEFFICIENTS OF THE REARRANGED POLYNOMIAL
+1.000000598
-.00004337858368
-.4994896781
-.002202691617
+.003949653275


MAXIMUM ABSOLUTE ERROR = 0.0000009
```

```
INPUT POLYNOMIAL
   1-1X1+1X2-1X3+1X4-1X5+1X6


COEFFICIENTS OF THE CHEBYSHEV EXPANSION
+1.630859375
-.05468750000
+.1635742187
+.05078125000
+.02050781250
+.003906250000
+.0004882812500


COEFFICIENTS OF THE REARRANGED POLYNOMIAL
+1.000000000
-1.000000000
+1.000000000
-1.000000000
+1.000000000
-1.000000000
+1.000000000


MAXIMUM ABSOLUTE ERROR =  0.0002


INPUT POLYNOMIAL
   1-1X1+1X2-1X3+1X4-1X5+1X6


COEFFICIENTS OF THE CHEBYSHEV EXPANSION
+1.630859375
-.05468750000
+.1635742187
+.05078125000
+.02050781250
+.003906250000
+.0004882812500


COEFFICIENTS OF THE REARRANGED POLYNOMIAL
+.9995117187
-.9648437500
+.5898437500
+.7500000000
-2.375000000
+2.000000000


MAXIMUM ABSOLUTE ERROR = 0.0005
```

```
INPUT POLYNOMIAL
.125+.0125X+.00125X2+.000125X3+.0000125X4+.00000125X5+.000000125X6
+.0000000125X7+.00000000125X8


COEFFICIENTS OF THE CHEBYSHEV EXPANSION
+.2635231382
+.006939628554
+.0001827484164
+.000004812485046
+1.267234802E-007
+3.334045410E-009
+8.697509766E-011
+2.136230468E-012
+3.814697265E-014


COEFFICIENTS OF THE REARRANGED POLYNOMIAL
+.1250000032
+.01249983935
+.001251263806
+.0001215583105
+.00001622060547


MAXIMUM ABSOLUTE ERROR = 0.000000009



COS 90, +-.000000009

    +5.995389074E-009


COS 90, +-.0000000001

    +7.862943860E-011


COS 89, +-.00000009

    +.01745241187


COS 89, +-.000000000001

    +.0174524064371980


COS 91, +-.000000000001

    -.0174524064370416
```

```
 SIN 30, +-.0001

     +.50006270115


SIN 150, +-.0000001

     +.50000002862


 SIN -30, +-.00000000000001

     -.5000000000000041


 SIN 390, +-.000000001

     +.5000000000072790


 COS 30, +-.0005

     +.86602379417


 COS 150, +-.000001

     -.86602541603


 COS -30, +-.0000000000001

     -.8660254037844290



 COS 0, +-.0000001

     +.99999999970


 COS 0, +-.0000000000001

     +.9999999999999981


 SIN 45, +-.000001

     +.70710663143


 SIN 225, +-.000001

     -.70710663143


 SIN -45, +-.00000000001

     -.7071067811865487
LOGOFF
   TIME 8.047 SECONDS  MFBLKS 24  COST $1.95
```