Printing Requisition/Graphic Services

Requisition No. 35600

1. Please complete unshaded areas on form as applicable. (4-part no carbon required)
2. Distribute copies as follows: White, Canary and Pink — Printing/Arts Library or applicable Copy Centre. Goldenrod — Retain.
3. On completion of order, pink copy will be returned with printed material. Canary copy will be costed and returned to requisitioner. Retain as a record of your charges.
4. Please direct enquiries, quoting requisition number, to Printing/Graphic Services, Extension [illegible]

Title or Description

Date | Date Required | [illegible]

Signature | Signing Authority

Department | Room | Phone | Delivery: ☐ Mail ☐ Pick-up ☐ Via Stores ☐ Other

Reproduction Requirements: ☐ Offset ☐ Signs/Repro's ☐ Xerox | Number of Pages | Number of Copies

Type of Paper Stock: ☐ Bond ☐ Book ☐ Cover ☐ Bristol ☐ Supplied

Paper Size: ☐ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17

Paper Colour: ☐ White ☐ Other | Ink: ☐ Black

Printing: ☐ 1 Side ☐ 2 Sides | Numbering: to

Binding/Finishing Operations: ☐ Collating ☐ Corner Stitching ☐ 3 Ring ☐ Tape ☐ Plastic Ring ☐ Perforating

Folding: Finished Size | Cutting: Finished Size

Special Instructions

Cost, Time/Materials		Set Up	Run	Oper. No.	Mins.	Total
Signs, Repro's	1					
Camera	2					
Correction & Masking Negatives	3					
Platemaking	4					
Printing	5					
Bindery	6					
Sub. Total Time						
Sub. Total Materials						
Prov. Tax						
Total						

Film: Qty | Size | Plates: Qty | Size & Type

Paper: Qty | Size | Plastic Rings: Qty | Size

Outside Services

Graphic Services May 88 [illegible]

AN ALTRAN IMPLEMENTATION OF
THE FRACTION-FREE PADÉ ALGORITHM

K.O. Geddes

Department of Computer Science
University of Waterloo

RESEARCH REPORT CS-78-10

February 1978

Faculty

of

Mathematics

University of Waterloo
Waterloo, Ontario, Canada

AN ALTRAN IMPLEMENTATION OF THE FRACTION-FREE PADÉ ALGORITHM

K.O. Geddes

Department of Computer Science
University of Waterloo

RESEARCH REPORT CS-78-10

February 1978

This research was supported by the National Research Council of Canada under Grant A8967.

ABSTRACT

The Fraction-Free Padé algorithm for the symbolic computation of Padé approximants is described in a pseudo-Algol algorithmic notation and also as a set of ALTRAN procedures. The results of applying the ALTRAN implementation to some sample problems are presented. The algorithm is based on a new one-step/two-step fraction-free symmetric triangularization algorithm and it exploits the block structure of the Padé table in case of singularity.

TABLE OF CONTENTS

1. INTRODUCTION.

The Padé method for deriving rational approximations to a power series is a classical tool of analysis. The analytical properties of Padé approximants have been expounded by several authors, including Wall [12] and Gragg [9]. Perhaps the most extensive application of the method has been in theoretical physics (see [1]).

The traditional framework for the method assumes that the power series coefficients lie in the field of complex numbers and that the power series represents a function in the complex plane. Although Padé approximants can be computed via solving a linear system of equations, algorithms for efficient floating-point computation are usually based on recurrence relations which reduce the computational complexity from $O(n^3)$ to $O(n^2)$. (See [13] for an extensive comparison of floating-point algorithms.) Typically, one computes a sequence of Padé approximants rather than a single approximant. The floating-point algorithms assume a restrictive normality condition which guarantees that the linear system corresponding to each approximantis nonsingular; for example, it is assumed that every coefficient in the series is nonzero.

In this paper, the Padé method is viewed as an algebraic manipulation algorithm which may be applied to a power series over an arbitrary integral domain. In particular, it may be applied to a power series whose coefficients contain indeterminates (i.e. the coefficients are multivariate polynomials) such as arises naturally from an initial-value problem with indeterminate initial conditions. The normality condition is not assumed because singular systems can be identified in exact computation and the block structure of the Padé table can be exploited to determine the corresponding approximant. The algorithm may be interpreted as a method for summing an

arbitrary power series which has a rational representation, or as a method for deriving rational approximations for a power series in contexts where the concept of "approximation" is valid.

1.1. Power Series over an Integral Domain D.

Following Birkhoff and Bartee [3], let D[[z]] denote the set of all power series in the symbol z with coefficients lying in the integral domain D, called the *coefficient domain*. The set D[[z]] itself comprises an integral domain, called a *power series domain*. The subset D[z] of all polynomials (finite power series) in z over D is also an integral domain. The quotient field of D[z] will be denoted by D(z) and it consists of all rational forms (rational functions) in z over D; i.e. D(z) contains quotients of the form $p(z)/q(z)$ where $p(z), q(z) \in D[z]$ with $q(z) \neq 0$ (the zero polynomial). The standard equivalence relation is assumed in the quotient field D(z):

$$p_1(z)/q_1(z) \equiv p_2(z)/q_2(z) \quad \text{if} \quad p_1(z)\, q_2(z) = q_1(z)\, p_2(z) \ .$$

The conditions under which a power series $C(z) \in D[[z]]$ can be represented by a rational function $p(z)/q(z) \in D(z)$ are discussed in [8]. The classical Padé theory requires that D[[z]] be embedded in the larger domain $F_D[[z]]$, where F_D denotes the quotient field of the coefficient domain D. The linear system defining a Padé approximant for C(z) can then be solved over the field F_D, leading to a rational function lying in the new quotient field $F_D(z)$. However, the fields D(z) and $F_D(z)$ are isomorphic since a rational function can always be normalized to eliminate fractions in the coefficients. Computationally, it is advantageous to pass directly from $C(z) \in D[[z]]$ to a Padé approximant $p(z)/q(z) \in D(z)$ without ever forming any fractions in F_D.

1.2. Avoiding the Quotient Field F_D.

If a rational function $p(z)/q(z)$ is to represent a power series in $F_D[[z]]$ then $q(z)$ must be invertible as a power series in $F_D[[z]]$. This will be true if and only if the first coefficient ("constant term") of $q(z)$ is nonzero (see [3], Chapter 13). It is therefore traditional to represent a Padé approximant in the normalized form

$$(1)\quad \frac{p(z)}{q(z)} = \frac{v_0 + v_1 z + \ldots + v_m z^m}{1 - u_1 z - \ldots - u_n z^n} \in F_D(z)$$

(the negative signs are convenient in the sequel). The power series expansion of a rational function in the form (1) is

$$p(z)/q(z) = \sum_{k=0}^{\infty} c_k z^k$$

where

$$(2)\quad c_k = v_k + \sum_{i=1}^{k} u_i c_{k-i},\ 0 \le k \le m;$$

$$(3)\quad c_k = \sum_{i=1}^{n} u_i c_{k-i},\ k > m;$$

with the convention that $u_i = 0$ for $i > n$ and $c_j = 0$ for $j < 0$.

Conversely, given a power series

$$C(z) = \sum_{k=0}^{\infty} c_k z^k \in F_D[[z]],$$

the denominator fractions $u_i (1 \le i \le n)$ of the (m,n) Padé approximant for $C(z)$ are normally determined by satisfying the first n equations of (3) (the proper handling of singularity is disucssed in [8]), and the numerator fractions are then determined by solving (2) for v_k $(0 \le k \le m)$.

The linear system of order n which defines the denominator fractions is called a *Hankel system* (see [6]) and is of the form

$$(4)\quad H_{m,n}\ \underline{u} = \underline{c}_{m,n}$$

where $\underline{u} = [u_n, \ldots, u_1]^T$ and where the augmented *Hankel matrix* is

$$[H_{m,n} \mid \underline{c}_{m,n}] = \left[\begin{array}{cccc|c} c_{m-n+1} & c_{m-n+2} & \cdots & c_m & c_{m+1} \\ c_{m-n+2} & c_{m-n+3} & \cdots & c_{m+1} & c_{m+2} \\ \vdots & \vdots & & \vdots & \vdots \\ c_m & c_{m+1} & \cdots & c_{m+n-1} & c_{m+n} \end{array}\right] .$$

If all of the coefficients c_k lie in the integral domain D (i.e. $C(z) \in D[[z]]$) then, rather than derive the (m,n) Padé approximant in the form (1), we choose to derive it in the re-normalized form

$$(5)\quad \frac{p(z)}{q(z)} = \frac{a_0 + a_1 z + \ldots + a_m z^m}{b_0 + b_1 z + \ldots + b_n z^n} \in D(z)$$

where: $b_0 = |H_{m,n}|$;

$b_i = -b_0\, u_i$ $(1 \le i \le n)$;

$a_k = b_0\, v_k$ $(0 \le k \le m)$.

Evidently, $b_0 \in D$ by the definition of a determinant, and $b_i \in D$ $(1 \le i \le n)$ by Cramer's rule. To see that $a_k \in D$, multiply through in equation (2) by b_0 and find:

$$(6)\quad a_k = \sum_{i=0}^{k} b_i\, c_{k-i} \qquad (0 \le k \le m)\ ,$$

which is simply a linear combination of elements in D. It is therefore possible

to compute the Padé approximant (5) for $C(z) \in D[[z]]$ with the computation entirely in the integral domain D. In practice, we do not solve the linear system (4) by the "division-free" Cramer's rule but rather we use a "fraction-free" symmetric elimination algorithm which requires some exact divisions. The latter algorithm produces the same results as Cramer's rule; namely, in the above notation, $b_0 u_i$ $(1 \le i \le n)$ and the determinant b_0. Equation (6) is then used to compute a_k $(0 \le k \le m)$.

.3. Fraction-Free Symmetric Triangularization

A new algorithm is presented in [8] for solving the Hankel system (4) over an integral domain D yielding, via (6), the (m,n) Padé approximant (5). The new algorithm exploits two special properties of the Hankel system (4). Firstly, the matrix $H_{m,n}$ is symmetric (but indefinite). Secondly, the augmented matrix $[H_{m,n} \mid \underline{c}_{m,n}]$ preserves this "symmetry" and has the interesting property that the $(n-r) \times (n-r+1)$ leading principal submatrix is precisely the Hankel system for the $(m-r, n-r)$ Padé approximant $(0 \le r \le \min\{m, n\})$. The new algorithm combines the concept of "block diagonal pivoting" used in numerical algorithms for symmetric indefinite triangularization with the concept of "two-step fraction-free elimination" used in symbolic computation. With the new algorithm, a diagonal sequence of Padé approximants can be computed efficiently using only one triangularization (and several back substitutions).

In section 2, the fraction-free Padé algorithm is presented in a pseudo-Algol algorithm notation. Section 3 describes some sample problems which were solved using the ALTRAN implementation of the algorithm. Section 4 contains the source listing of the ALTRAN procedures implementing the fraction-free Padé algorithm.

2. FORMAL DESCRIPTION OF THE ALGORITHM

In this section we give a formal description of the fraction-free Padé algorithm. Subsection 2.1 contains some general remarks about the algorithm and its implementation. The basic Padé procedure is presented in subsection 2.2 and the procedures implementing the one-step/two-step fraction-free method (see [8]) for solving symmetric systems of the linear equations are presented in subsection 2.3. Some sample problems and timing statistics are presented in section 3.

Section 4 contains the source code listing of an implementation of the fraction-free Padé algorithm in ALTRAN. Section 3 illustrates the application of the algorithm to power series generated by differential equations with indeterminate initial conditions. In the latter application, the relevant power series domain is $\underline{Q}[\underline{x}][[z]]$, where the coefficient domain is the multivariate polynomial domain $\underline{Q}[\underline{x}]$ over the field Q of rational numbers with indeterminates $\underline{x} = (x_1,\ldots,x_k)$. For example, the initial-value problem

$$(7)\quad y'(z) = [y(z)]^2,\ y(0) = B,$$

generates the power series

$$y(z) = B + B^2z + B^3z^2 + B^4z^3 + \ldots$$

which lies in the power series domain $\underline{Q}[B][[z]]$. In fact, this power series lies in the domain $\underline{Z}[B][[z]]$ where $\underline{Z}$ denotes the integral domain of integers, and it is significant that all computations in the fraction-free Padé algorithm would remain entirely within $\underline{Z}[B]$ rather than the larger domain $\underline{Q}[B]$ since $\underline{Z}[B]$ is also an integral domain.

2.1 The Fraction-Free Padé Algorithm

Given m,n and the coefficients $c_0,\ldots,c_{m+n}$ of the power series

$$C(z) = \sum_{k=0}^{\infty} c_k z^k,$$

the Padé algorithm computes the sequence of Padé approximants:

$$R_{m-r,n-r}(z),\ r = \ell,\ \ell-1,\ldots,0,$$

where if c_σ is the first nonzero coefficient then $\ell = \min\{m-\sigma,n\}$. By the Padé theory (see [8]), $R_{m-\ell,n-\ell}(z)$ is well-defined by the Hankel system (or else it is a Taylor polynomial in case $n-\ell=0$). For each succeeding approximant, either the Hankel system is nonsingular or the approximant is the same as its predecessor in the sequence. For example, if the power series is the one generated by problem (7) and if m=n=10 then the sequence of approximants is

$$R_{0,0}(z), R_{1,1}(z),\ldots,R_{10,10}(z)$$

where

$$R_{0,0}(z) = B;$$

$$R_{1,1}(z) = B/(1-Bz);$$

$$R_{k,k}(z) = R_{1,1}(z) \text{ for } 2 \le k \le 10.$$

Note that the exact solution of (7) is

$$y(z) = B/(1-Bz)$$

and this exact solution is obtained.

The procedures in subsections 2.2 and 2.3 are presented in a pseudo-Algol algorithmic notation which can be readily translated into any of the languages for symbolic computation. It should be noted that all divisions appearing in these procedures are exact divisions in the integral domain of power series coefficients, except of course for the final

formation of the Padé approximant as a rational function (step 4.4 of procedure PADE). In a particular application, it may or may not be desirable to have each rational approximant reduced to lowest terms (by appropriate GCD computations). As an example, the solution for problem (7) is obtained in the form

$$R_{1,1}(z) = B^3/(B^2-B^3z)$$

(the constant term in the denominator is always the determinant of the corresponding Hankel matrix) and this rational function will be reduced only if a GCD computation is performed.[1] In some applications, the extra cost of such GCD computations may not be warranted.

There is no mention here of algorithms for generating Taylor series coefficients. In the sample problems presented in this paper, the Taylor series have been generated from initial-value problems by the (expensive) method of repeated differentiation (see [7]). Efficient methods for generating power series expansions are discussed by Norman [11] and by Zippel [14].

As a final remark on implementation of the procedures, note that the procedures in subsection 2.3 for solving symmetric linear systems operate only on the upper triangular portion of the augmented matrix. The ALTRAN implementation uses a one-dimensional array to store the upper triangular augmented matrix, row by row, and employs appropriate indexing operations.

---

[1] In ALTRAN, the denominator of $R_{1,1}(z)$ is represented in the factored form $-B^2(Bz-1)$ so that in this example the reduction does take place even if the "non-canonical" option is specified. But the point holds in general.

2.2 Procedure PADE

procedure PADE (c,m,n)

Purpose: To compute the sequence of Padé approximants $R_{m-r,n-r}(z)$, $r = \ell, \ell-1,\ldots,0$.
for the power series with coefficients given by array c, where if c_σ is the first nonzero coefficient then
$$\ell = \min \{m-\sigma,n\}.$$

Input Parameters:
c - array of power series coefficients $c_0,\ldots,c_{m+n}$;

m,n - degrees of numerator and denominator, respectively, of the highest-degree Padé approximant desired.

Output:
The sequence of Padé approximants (see above) has been defined.

1. [Define upper triangular portion of n-by-(n+1) augmented Hankel matrix]

   $h_{ij} \leftarrow c_{m-n+i+j-1}$ $(1 \le i \le n,\ i \le j \le n+1)$ where $c_k = 0$ for $k < 0$

2. [Triangularize the augmented Hankel matrix]

   SYMTRI (h, n, det, perm, block)

3. [Define ℓ]

   $\sigma \leftarrow$ index of first nonzero coefficient in c

   $\ell \leftarrow \min\{m-\sigma, n\}$

4. [Compute the sequence of Padé approximants]

for r = ℓ step - 1 until 0 do

if n-r = 0 then [approximant is a Taylor polynomial]

$$R_{m-r,n-r}(z) \leftarrow \sum_{k=0}^{m-r} c_k z^k$$

else if det(n-r) = 0 then [new approximant is preceeding approximant]

$$R_{m-r,n-r}(z) \leftarrow R_{m-r-1,n-r-1}(z)$$

else

4.1. [Invoke back-substitution procedure]

SYMSOL (h, n-r, perm, block, det(n-r), xnum)

4.2. [Define numerator polynomial]

for k = 0 step 1 until m-r do

$$a_k \leftarrow \det(n-r)*c_k - \sum_{i=1}^{\min\{k,n-r\}} \text{xnum}(n-r-i+1)*c_{k-i}$$

doend

$$p(z) \leftarrow \sum_{k=0}^{m-r} a_k z^k$$

4.3. [Define denominator polynomial]

$$q(z) \leftarrow \det(n-r) - \sum_{k=1}^{n-r} \text{xnum}(n-r-k+1) z^k$$

4.4. [Define new approximant]

$$R_{m-r,n-r}(z) \leftarrow p(z) / q(z)$$

doend

end of procedure PADE.

2.3 Procedures SYMTRI, SYMINT, and SYMSOL

Remark: The three procedures of this subsection constitute a fraction-free package for the symbolic solution of symmetric indefinite systems of linear equations. The elimination algorithm used is the one-step/two-step algorithm described in [8].

procedure SYMTRI (a, n, det, perm, block)

Purpose: To perform symmetric triangularization in the upper triangular portion of the n-by-(n+1) augmented matrix a. The method used is one-step/two-step fraction-free Gaussian elimination.

Input Parameters:

a - upper triangular portion of the n-by-(n+1) augmented matrix to be triangularized;

n - the order of the matrix to be triangularized.

Output Parameters:

a - the triangularized augmented matrix (note that symmetric 2-by-2 blocks may appear on the diagonal as indicated by Boolean vector block);

det - vector of length n whose i-th entry is the determinant of the i-by-i leading principal submatrix of the input matrix (thus det(n) is the determinant of the input matrix);

perm - permutation vector of length n specifying the order of the unknowns in the triangularized linear system;

block - Boolean vector of length n such that if block(i) is true then the i-th diagonal element belongs to a symmetric 2-by-2 block (note that the upper left element of a 2-by-2 block will always be zero).

1. [Initialization]

   det(i) ← 1 $(1 \le i \le n)$ [could use any nonzero value]

   perm(i) ← i $(1 \le i \le n)$

   block(i) ← false $(1 \le i \le n)$

2. [Elimination Loop]

   divisor ← 1

   k ← 0

   while $k \le n-2$ do

      determinant ← $a_{k+1,k+1}\ a_{k+2,k+2} - a^2_{k+1,k+2}$

      if determinant = 0 & $a_{k+1,k+1} \ne 0$

         then [One-step elimination]

            k ← k+1

            $a_{ij} \leftarrow (a_{kk}\ a_{ij} - a_{ki}\ a_{kj})/\text{divisor}$ $(k+1 \le i \le n,\ i \le j \le n+1)$

            divisor ← a_{kk}

            if det(k) ≠ 0 then det(k) ← a_{kk}

         else [Two-step elimination]

            k ← k+2

2.1. [Ensure nonsingular 2-by-2 pivot]

2.2. [Update rows below row k]

2.3. [Update row k if possible]

divisor ← determinant

if det(k-1) ≠ 0 then det(k-1) ← $a_{k-1,k-1}$

if det(k) ≠ 0 then det(k) ← determinant

doend

3. [If final value of k is n-1, det(n) was not assigned]

if k = n-1 then det(n) ← a_{nn}

end of procedure SYMTRI

where steps 2.1, 2.2, 2.3 are refined as follows:

2.1. [Ensure nonsingular 2-by-2 pivot]

if determinant = 0 then [Note that $a_{k-1,k-1} = a_{k-1,k} = 0$]

det(k-1) ← det(k) ← 0

if $a_{k-1,j} = 0$ for all $k+1 \le j \le n$

then [All remaining subsystems singular]

det(j) ← 0 ($k+1 \le j \le n$)

return [Exit from procedure]

else [Interchange and compute new determinant]

i ← the smallest j ($k+1 \le j \le n$) such that $a_{k-1,j} \neq 0$

SYMINT(a, n, k, i)

perm(k) ⇔ perm(i) [Interchange in vector perm]

det(j) ← 0 ($k+1 \le j \le i-1$)

determinant ← - $a^2_{k-1,k}$

2.2. [Update rows below row k]

determinant ← determinant/divisor

for i = k+1 step 1 until n do

$m_{i1} \leftarrow (a_{k-1,i}\ a_{kk} - a_{ki}\ a_{k-1,k})$/divisor

$m_{i2} \leftarrow (a_{ki}\ a_{k-1,k-1} - a_{k-1,i}\ a_{k-1,k})$/divisor

$a_{ij} \leftarrow$ (determinant * $a_{ij} - m_{i1}\ a_{k-1,j} - m_{i2}\ a_{kj}$)/divisor $(i \le j \le n+1)$

doend

2.3. [Update row k if possible]

if $a_{k-1,k-1} \neq 0$

then [One-step update for row k]

$a_{kk} \leftarrow$ determinant

$a_{kj} \leftarrow (a_{k-1,k-1}\ a_{kj} - a_{k-1,k}\ a_{k-1,j})$/divisor $(k+1 \le j \le n+1)$

else [Signal 2-by-2 block on diagonal]

block(k-1) ← block(k) ← true .

procedure SYMINT(a, n, k, i)

Purpose: To perform a symmetric interchange of rows and columns k and i in the upper triangular portion of the n-by-(n+1) augmented matrix a, where k < i.

$a_{jk} \leftrightarrow a_{ji}$ $(1 \le j \le k-1)$

$a_{kk} \leftrightarrow a_{ii}$

$a_{kj} \leftrightarrow a_{ji}$ $(k+1 \le j \le i-1)$

$a_{kj} \leftrightarrow a_{ij}$ $(i+1 \le j \le n+1)$

end of procedure SYMINT.

procedure SYMSOL(a, n, perm, block, determinant, xnum)

Purpose: To solve the n-by-(n+1) linear system which has been triangularized by procedure SYMTRI, using fraction-free back-substitution.

Input Parameters:

a - the n-by-(n+1) augmented matrix as triangularized by procedure SYMTRI;

n - the order of the linear system;

perm - permutation vector from procedure SYMTRI;

block - Boolean vector from procedure SYMTRI;

determinant - the value of the determinant of the n-by-n coefficient matrix.

Output Parameters:

xnum - vector of length n containing the "numerators" of the solution vector -- the complete solution is obtained by dividing each element of xnum by determinant.

1. [Solve bottom equation or bottom two equations, as required]

if block(n)

then [Solve 2-by-2 block on bottom diagonal]

xnum(perm(n)) ← determinant * $a_{n-1,n+1}/a_{n-1,n}$

xnum(perm(n-1)) ← (determinant * $a_{n,n+1}$ - a_{nn} * xnum(perm(n)))/$a_{n-1,n}$

k ← n-2

else [Solve bottom equation]

xnum(perm(n)) ← $a_{n,n+1}$

k ← n-1

2. [Back-substitution loop]

   while k > 0 do

      if block(k)

         then [Diagonal contains 2-by-2 block]

$$\text{xnum(perm(k))} \leftarrow \left(\text{determinant} * a_{k-1,n+1} - \sum_{j=k+1}^{n} a_{k-1,j} * \text{xnum(perm(j))} \right) / a_{k-1,k}$$

$$\text{xnum(perm(k-1))} \leftarrow \left(\text{determinant} * a_{k,n+1} - \sum_{j=k}^{n} a_{kj} * \text{xnum(perm(j))} \right) / a_{k-1,k}$$

            k ← k-2

         else [Normal back-substitution]

$$\text{xnum(perm(k))} \leftarrow \left(\text{determinant} * a_{k,n+1} - \sum_{j=k+1}^{n} a_{kj} * \text{xnum(perm(j))} \right) / a_{kk}$$

            k ← k-1

   doend

end of procedure SYMSOL.

3. SAMPLE PROBLEMS

In this section, the output from the ALTRAN program which implements the algorithm of section 2 is presented for the following sample problems. For each problem, the power series has been derived from an ordinary differential equation of initial-value type but the method of deriving the power series is not part of the implementation described in this report.

Problem 1: (Problem with rational solution)

$y' = y^2$; $y(0) = \mu_1$.

Series Solution: $y(x) = \mu_1 + \mu_1^2 x + \mu_1^3 x^2 + \mu_1^4 x^3 + \ldots.$

Exact Solution: $y(x) = \mu_1 / (1 - \mu_1 x)$

Problem 2: (Exponential function with parameters)

$y' = \mu_1 y$; $y(0) = \mu_2$

Series Solution: $y(x) = \mu_2 + \mu_2\mu_1 x + \mu_2\mu_1^2/2!\ x^2 + \mu_2\mu_1^3/3!\ x^3 \ldots$

Exact Solution: $y(x) = \mu_2\ e^{\mu_1 x}$.

Problem 3: (Tangent function)

$y' = 1 + y^2$; $y(0) = 0$.

Series Solution: $y(x) = x + 1/3\ x^3 + 2/15\ x^5 + 17/315\ x^7 + \ldots$

Exact Solution: $y(x) = \tan(x)$

Problem 4: (Ordinary Thomas-Fermi function)

$$y''' = 8\, y^{3/2} + 6\, x\, y^{1/2}\, y';$$

$$y(0) = 1;\ y'(0) = 0;\ y''(0) = 2\,\mu_1.$$

Series Solution:
$$\begin{aligned} y(x) = 1 &+ \mu_1 x^2 + 4/3\, x^3 + 2/5\, \mu_1 x^5 \\ &+ 1/3\, x^6 + 3/70\, \mu_1^2 x^7 + 2/15\, \mu_1 x^8 \\ &+ (-1/252\, \mu_1^3 + 2/27)\, x^9 + 1/175\, \mu_1^2 x^{10} \\ &+ (1/1056\, \mu_1^4 + 31/1485\, \mu_1)\, x^{11} \\ &+ (4/1575\, \mu_1^3 + 4/405)\, x^{12} \\ &+ (-3/9152\, \mu_1^5 + 557/100100\, \mu_1^2)\, x^{13} \\ &+ (-29/24255\, \mu_1^4 + 4/693\, \mu_1)\, x^{14} \\ &+ \ldots \end{aligned}$$

Remark: This problem is discussed in [7].

3.1. Remarks on the Output

The first part of the output lists the specified values of M and N, the maximum degree of numerator and denominator, respectively. Then follows a listing of the specified power series coefficients. The timing statistics presented next include the time for triangularization of the N-by-N Hankel system and the time to solve the order-N system by back-substitution. (Note that, in general, there will be back-substitutions performed in all N subsystems and only the timing for the largest subsystem is presented here. Also note that if the order-N system is singular then no order-N back-substitution is performed.)

The total time required to compute the Padé approximants is listed next. This is the total time for execution of procedure PADE to compute the full sequence of approximants

$$R_{M-r,N-r}(x),\ r = \ell,\ \ell-1,\ldots,0,$$

where if c_σ is the first nonzero power series coefficient then

$$\ell = \min\{M-\sigma,\ N\}\ .$$

The latter timing statistic thus includes the cost of one triangularization, all back-substitutions, and the formation of the numerators and denominators of the approximants (in particular, this timing statistic includes the previous two).

The Padé approximants are then listed in rational function form. The index I appearing in the output is related to the indices used above by

$$I = \ell - r.$$

Thus I ranges from 0 to ℓ and if M=N and $c_0 \neq 0$ then the approximant corresponding to the index I in the output is the (I,I) Padé approximant. Finally, the total elapsed time is listed. The difference between the total elapsed time and the time spent in procedure PADE is mainly due to

the formation of the approximants in rational function form (i.e., dividing the numerator polynomial by the denominator polynomial for each approximant). In the sample problems here, these approximants were formed with the "non-canonical" option specified so that no GCD computations were performed and the cost was thus minimized. If the standard "canonical" option was in effect, the difference in time could be very significant. It should be noted that the Padé theory guarantees that the numerator and denominator of each approximant, as formed here, are relatively prime in the independent variable (i.e. the variable X of the output) so that the only possible common factors would be polynomials in the parameters of the problem (i.e. the variables MU(1), MU(2), etc. of the output).

3.2. Further Options in the ALTRAN Program

There are two parameters to procedure PADE which are not basic to the algorithm but provide options in the use of the algorithm. These are the LOGICAL parameters FRCTNS and SEQUEN (read "fractions" and "sequence"). In the sample problems and timing statistics presented here, the values of these two parameters were both .TRUE. .

The parameter FRCTNS specifies if fractions are to be carried during the computation. This refers to the rational coefficients of the polynomials in the indeterminates of the problem (or to the rational power series coefficients if no indeterminates are present). For example, if the (2,2) Padé approximant for Problem 2 is to be computed, the truncated powerseries passed into procedure PADE would be

$$C(x) = \mu_2 + \mu_2\mu_1 x + \mu_2\mu_1^2/2\, x^2 + \mu_2\mu_1^3/6\, x^3 + \mu_2\mu_1^4/24\, x^4$$

(i.e. the array of power series coefficients is passed to procedure PADE).

The power series coefficients are bivariate polynomials with rational coefficients. The computation can be made to work entirely in the integral domain of bivariate polynomials with integer coefficients (and thus avoid the GCD computations required when working with rational numbers) by noting that

$$C(x) = D(x)/24$$

where

$$D(x) = 24\ \mu_2 + 24\ \mu_2\mu_1 x + 12\ \mu_2\mu_1^2\ x^2 + 4\ \mu_2\mu_1^3\ x^3 + \mu_2\mu_1^4\ x^4.$$

The Padé approximants for D(x) can be computed and then the constant factor 24 can be replaced at the end of the computation.

The algorithm will carry fractions (i.e. work directly with C(x) in the example above) if FRCTNS is .TRUE. and it will convert to integer coefficients (i.e. work with D(x) in the example above) if FRCTNS is .FALSE. . This option was put into the program in the belief that avoiding fractions might decrease the cost of the computation. This option has not been thoroughly tested and all of the sample problems presented here were run with FRCTNS set to .TRUE. . Preliminary tests with this option have shown that the increased size of the integers carried when FRCTNS is .FALSE. may offset the savings to be gained by avoiding GCD computations on rational numbers.

The second LOGICAL parameter providing an option in procedure PADE is the parameter SEQUEN. The algorithm described in section 2 computes a diagonal sequence of Padé approximants

$$R_{M-\ell,N-\ell}(x),\ldots,R_{M,N}(x).$$

In the ALTRAN implementation of section 4, this sequence will be computed if SEQUEN is set to .TRUE. . However, if it is desired to compute only the approximant $R_{M,N}(x)$ then the parameter SEQUEN may be set to .FALSE. .

In the sample problems presented here, the value of SEQUEN was always .TRUE. . Table 2 in the following subsection indicates that, at least for Problem 4, the cost of computing the single approximant $R_{M,N}(x)$ is about equal to the cost of computing the entire sequence up to and including $R_{M-1,N-1}(x)$.

3.3. Summary of Timing Statistics

Table 1 presents the timing statistics for the four problems of this section. The problems were run on a Honeywell 66/60 and all times are in seconds. The memory requirements for these problems was minimal except for Problem 4 (see Table 2). The meanings of the headings in Table 1 are as follows.

(M,N) - the highest-degree Padé approximant in the sequence computed;

SYMTRI (order N) - time to triangularize the order-N Hankel matrix;

SYMSOL (order N) - time to do the order-N back-substitution (note that a back-substitution is done for each non-singular subsystem and this column is timing only the largest subsystem);

PADE (full sequence) - total time spent in procedure PADE (includes one triangularization, all back-substitutions, and formation of the numerator and denominator polynomials);

TOTAL (PADE plus output) - this time includes the time specified in the preceeding column plus the time to form the approximants as rational functions (in non-canonical mode - see

subsection 3.1) and the time to print them out.

Table 1: Execution Times for Problems 1-4.
(Times are in seconds; System: ALTRAN on Honeywell 66/60)

Problem Number	(M,N)	SYMTRI (order N)	SYMSOL (order N)	PADE (full sequence)	TOTAL (PADE plus output)
1	(10,10)	5.2	-	9.5	17.9
2	(5,5)	4.1	1.7	17.5	20.8
3	(7,7)	3.6	1.1	17.5	21.4
4	(7,7)	15.6	9.9	64.4	70.3

Table 2 illustrates the effect of increasing the maximum degree of approximant to be computed. In this table, only Problem 4 (the "hardest" problem) is considered. All of the headings are as described for Table 1 with the addition of the final heading:

MEMORY (Workspace) - the number of words of memory required in the workspace (note that the total memory requirements are 45K words plus the workspace, with a minimum of 4K words required in the workspace).

Table 2: Execution Times for Problem 4 -
The Effect of Increasing Degree.

(Times are in seconds; System: ALTRAN on Honeywell 66/60)

(M,N)	SYMTRI (order N)	SYMSOL (order N)	PADE (full sequence)	MEMORY* (Workspace)
(3,3)	.6	.4	3.8	4K words
(4,4)	1.5	1.1	7.9	4K words
(5,5)	3.6	2.1	16.2	4K words
(6,6)	7.5	5.0	32.0	6K words
(7,7)	15.6	9.9	64.4	8K words

*Total Memory Requirements = 45K words + Workspace

(Min. 4K words in Workspace)

Output for Problem 1:

```
# M
    10
# N
    10
# THE POWER SERIES COEFFICIENTS ARE
# C
    ( MU(1) ,
    MU(1)**2 ,
    MU(1)**3 ,
    MU(1)**4 ,
    MU(1)**5 ,
    MU(1)**6 ,
    MU(1)**7 ,
    MU(1)**8 ,
    MU(1)**9 ,
    MU(1)**10 ,
    MU(1)**11 ,
    MU(1)**12 ,
    MU(1)**13 ,
    MU(1)**14 ,
    MU(1)**15 ,
    MU(1)**16 ,
    MU(1)**17 ,
    MU(1)**18 ,
    MU(1)**19 ,
    MU(1)**20 ,
    MU(1)**21 )
```

```
# TIME (SEC.) FOR TRIANGULARIZATION WAS
# TNEW
   5.209312
# TIME (SEC.) TO COMPUTE PADE APPROXIMANTS WAS
# TNEW
   9.530109
# THE PADE APPROXIMANTS ARE
# I
   0
# RI
   MU(1)
# I
   1
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   2
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   3
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   4
# RI
   - MU(1) / ( X*MU(1) - 1 )
```

```
# I
   5
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   6
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   7
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   8
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   9
# RI
   - MU(1) / ( X*MU(1) - 1 )
# I
   10
# RI
   - MU(1) / ( X*MU(1) - 1 )
# TOTAL ELAPSED TIME (SEC.) WAS
# TNEW
   1.790972E1
```

Output for Problem 2:

```
# M
   5
# N
   5
# THE POWER SERIES COEFFICIENTS ARE
# C
   ( MU(2) ,
   MU(1)*MU(2) ,
   MU(1)**2*MU(2) / 2 ,
   MU(1)**3*MU(2) / 6 ,
   MU(1)**4*MU(2) / 24 ,
   MU(1)**5*MU(2) / 120 ,
   MU(1)**6*MU(2) / 720 ,
   MU(1)**7*MU(2) / 5040 ,
   MU(1)**8*MU(2) / 40320 ,
   MU(1)**9*MU(2) / 362880 ,
   MU(1)**10*MU(2) / 3628800 )
# TIME (SEC.) FOR TRIANGULARIZATION WAS
# TNEW
   4.090094
# TIME (SEC.) TO SOLVE ORDER-N SYSTEM WAS
# TNEW
   1.720328
# TIME (SEC.) TO COMPUTE PADE APPROXIMANTS WAS
# TNEW
   1.753778E1
```

```
# THE PADE APPROXIMANTS ARE
# I
   0
# RI
   MU(2)
# I
   1
# RI
   - MU(2) * ( X*MU(1) + 2 ) / ( X*MU(1) - 2 )
# I
   2
# RI
   MU(2) * ( X**2*MU(1)**2 + 6*X*MU(1) + 12 ) /
   ( X**2*MU(1)**2 - 6*X*MU(1) + 12 )
# I
   3
# RI
   - MU(2) * ( X**3*MU(1)**3 + 12*X**2*MU(1)**2 + 60*X*MU(1) + 120 ) /
   ( X**3*MU(1)**3 - 12*X**2*MU(1)**2 + 60*X*MU(1) - 120 )
# I
   4
# RI
   MU(2) * ( X**4*MU(1)**4 + 20*X**3*MU(1)**3 + 180*X**2*MU(1)**2 +
   840*X*MU(1) + 1680 ) / ( X**4*MU(1)**4 - 20*X**3*MU(1)**3 +
   180*X**2*MU(1)**2 - 840*X*MU(1) + 1680 )
```

```
# I
   5
# RI
   - MU(2) * ( X**5*MU(1)**5 + 30*X**4*MU(1)**4 + 420*X**3*MU(1)**3 +
   3360*X**2*MU(1)**2 + 15120*X*MU(1) + 30240 ) /
   ( X**5*MU(1)**5 - 30*X**4*MU(1)**4 + 420*X**3*MU(1)**3 - 3360*X**2*
   MU(1)**2 + 15120*X*MU(1) - 30240 )
# TOTAL ELAPSED TIME (SEC.) WAS
# TNEW
   2.084383E1
```

Output for Problem 3:

```
# M
   7
# N
   7
# THE POWER SERIES COEFFICIENTS ARE
# C
   ( 0 ,
   1 ,
   0 ,
   1 / 3 ,
   0 ,
   2 / 15 ,
   0 ,
   17 / 315 ,
   0 ,
   62 / 2835 ,
   0 ,
   1382 / 155925 ,
   0 ,
   21844 / 6081075 ,
   0 )
# TIME (SEC.) FOR TRIANGULARIZATION WAS
# TNEW
   3.589547
# TIME (SEC.) TO SOLVE ORDER-N SYSTEM WAS
# TNEW
   1.125734
```

```
# TIME (SEC.) TO COMPUTE PADE APPROXIMANTS WAS
# TNEW
   1.745994E1
# THE PADE APPROXIMANTS ARE
# I
   0
# RI
   X
# I
   1
# RI
   - 3*X / ( X**2 - 3 )
# I
   2
# RI
   X * ( X**2 - 15 ) / ( 3 * ( 2*X**2 - 5 ) )
# I
   3
# RI
   - 5*X * ( 2*X**2 - 21 )./ ( X**4 - 45*X**2 + 105 )
# I
   4
# RI
   X * ( X**4 - 105*X**2 + 945 ) / ( 15 * ( X**4 - 28*X**2 + 63 ) )
```

```
# I
   5
# RI
   - 21*X * ( X**4 - 60*X**2 + 495 ) /
   ( X**6 - 210*X**4 + 4725*X**2 - 10395 )
# I
   6
# RI
   X * ( X**6 - 378*X**4 + 17325*X**2 - 135135 ) /
   ( 7 * ( 4*X**6 - 450*X**4 + 8910*X**2 - 19305 ) )
# TOTAL ELAPSED TIME (SEC.) WAS
# TNEW
   2.136477E1
```

Output for Problem 4:

```
# M
   7
# N
   7
# THE POWER SERIES COEFFICIENTS ARE
# C
   ( 1 ,
   0 ,
   MU(1) ,
   4 / 3 ,
   0 ,
   2*MU(1) / 5 ,
   1 / 3 ,
   3*MU(1)**2 / 70 ,
   2*MU(1) / 15 ,
   - ( 3*MU(1)**3 - 56 ) / 756 ,
   MU(1)**2 / 175 ,
   MU(1) * ( 45*MU(1)**3 + 992 ) / 47520 ,
   4 * ( 9*MU(1)**3 + 35 ) / 14175 ,
   - MU(1)**2 * ( 525*MU(1)**3 - 8912 ) / 1601600 ,
   - MU(1) * ( 29*MU(1)**3 - 140 ) / 24255 )
# TIME (SEC.) FOR TRIANGULARIZATION WAS
# TNEW
   1.558583E1
# TIME (SEC.) TO SOLVE ORDER-N SYSTEM WAS
# TNEW
   9.917313
```

```
# TIME (SEC.) TO COMPUTE PADE APPROXIMANTS WAS
# TNEW
   6.438642E1
# THE PADE APPROXIMANTS ARE
# I
   0
# RI
   1
# I

   1
# RI
   1
# I
   2
# RI
   ( 9*X**2*MU(1)**3 + 16*X**2 - 12*X*MU(1) + 9*MU(1)**2 ) /
   ( 16*X**2 - 12*X*MU(1) + 9*MU(1)**2 )
# I
   3
# RI
   - ( 1161*X**3*MU(1)**3 + 5200*X**3 + 810*X**2*MU(1)**4 +
   4260*X**2*MU(1) + 405*X*MU(1)**2 + 810*MU(1)**3 + 4800 ) /
   ( 3 * ( 108*X**3*MU(1)**3 + 400*X**3 + 180*X**2*MU(1) -
   135*X*MU(1)**2 - 270*MU(1)**3 - 1600 ) )
```

```
# I
   4
# RI
   ( 54675*X**4*MU(1)**6 + 118368*X**4*MU(1)**3 - 318500*X**4 -
   611550*X**3*MU(1)**4 - 2247000*X**3*MU(1) - 455625*X**2*MU(1)**5 -
   1651860*X**2*MU(1)**2 - 62370*X*MU(1)**3 - 294000*X - 510300*
   MU(1)**4 - 1833300*MU(1) ) / ( 3 * ( 6696*X**4*MU(1)**3 + 24500*X**4 +
   43740*X**3*MU(1)**4 + 163800*X**3*MU(1) + 18225*X**2*MU(1)**5 +
   60480*X**2*MU(1)**2 - 20790*X*MU(1)**3 - 98000*X - 170100*MU(1)**4 -
   611100*MU(1) ) )
# I
   5
# RI
   - ( 2864192400*X**5*MU(1)**9 + 1_3375989144*X**5*MU(1)**6 +
   9670542000*X**5*MU(1)**3 + 7803250000*X**5 + 481595625*X**4*MU(1)**10
   9377735850*X**4*MU(1)**7 - 4_4130561300*X**4*MU(1)**4 + 2881200000*
   X**4*MU(1) - 4706879625*X**3*MU(1)**8 - 1_2868212420*X**3*MU(1)**5 +
   2_1207690000*X**3*MU(1)**2 + 5682828375*X**2*MU(1)**9 +
   2_9144181780*X**2*MU(1)**6 + 2_7372429000*X**2*MU(1)**3 +
   7203000000*X**2 - 1_2198976650*X*MU(1)**7 - 4_5268826400*X*MU(1)**4 -
   3241350000*X*MU(1) + 5201232750*MU(1)**8 + 2_4659635140*MU(1)**5 +
   2_1670740000*MU(1)**2 ) / ( 3 * ( 138516075*X**5*MU(1)**9 + 662786964*
   X**5*MU(1)**6 + 718242000*X**5*MU(1)**3 + 600250000*X**5 - 801006570*
   X**4*MU(1)**7 - 3508728300*X**4*MU(1)**4 - 185706675*X**3*MU(1)**8 +
   159633180*X**3*MU(1)**5 + 1481760000*X**3*MU(1)**2 - 160531875*X**2*
   MU(1)**9 - 1494848880*X**2*MU(1)**6 - 1900563000*X**2*MU(1)**3 -
   2401000000*X**2 + 4066325550*X*MU(1)**7 + 1_5089608800*X*MU(1)**4 +
   1080450000*X*MU(1) - 1733744250*MU(1)**8 - 8219878380*MU(1)**5 -
   7223580000*MU(1)**2 ) )
```

```
# I
   6
# RI
   3 * ( 2240_1718171875*X**6*MU(1)**15 - 1119327_2284957200*X**6*
   MU(1)**12 - 5704238_7216210564*X**6*MU(1)**9 + 8180274_1159789440*X**6
   MU(1)**6 + 50528246_4081920000*X**6*MU(1)**3 - 5010906_2080000000*X**6
   410274_9993588750*X**5*MU(1)**13 - 4746820_4559168120*X**5*MU(1)**10 -
   17461113_2455939200*X**5*MU(1)**7 - 24341449_7636688000*X**5*MU(1)**4
   16838301_3568000000*X**5*MU(1) - 47263_1062640625*X**4*MU(1)**14 +
   304912_7462965500*X**4*MU(1)**11 + 8880514_5845999520*X**4*MU(1)**8 +
   18702078_6010060800*X**4*MU(1)**5 - 35827920_2265600000*X**4*MU(1)**2
   2261913_7862648250*X**3*MU(1)**12 - 7619427_7375240080*X**3*MU(1)**9 +
   25331345_8492348800*X**3*MU(1)**6 + 79127058_0455040000*X**3*MU(1)**3
   2733221_5680000000*X**3 - 358329_3621862500*X**2*MU(1)**13 -
   9510694_5202636500*X**2*MU(1)**10 - 36482565_1826865600*X**2*MU(1)**7
   17860635_6473664000*X**2*MU(1)**4 - 5640375_4176000000*X**2*MU(1) -
   1599355_6601022000*X*MU(1)**11 + 12340739_8835804160*X*MU(1)**8 +

   69299950_9996886400*X*MU(1)**5 - 22429810_0857600000*X*MU(1)**2 -

   308826_0841050000*MU(1)**12 - 12176809_0291663200*MU(1)**9 -
   31828870_2584160000*MU(1)**6 + 49517455_6800000000*MU(1)**3 +
   6833053_9200000000 ) / ( 559413_8942405400*X**6*MU(1)**12 +
   3913225_0072474356*X**6*MU(1)**9 - 1710508_4893733760*X**6*MU(1)**6 -
   32807087_5937280000*X**6*MU(1)**3 + 3644295_4240000000*X**6 +
   90149_4572422500*X**5*MU(1)**13 + 880518_1782864600*X**5*MU(1)**10 +
   9016418_6880969600*X**5*MU(1)**7 + 30466287_5286672000*X**5*MU(1)**4 +
   10311699_5520000000*X**5*MU(1) + 6720_5154515625*X**4*MU(1)**14 -
```

```
   686182_6474095600*X**4*MU(1)**11 - 8760331_0077101280*X**4*MU(1)**8 -
   18959294_2136371200*X**4*MU(1)**5 + 19655767_6761600000*X**4*MU(1)**2
   752370_0420678750*X**3*MU(1)**12 - 11173266_7466479920*X**3*MU(1)**9 -
   4590334_4176972800*X**3*MU(1)**6 + 106600781_6737920000*X**3*MU(1)**3
   19132550_9760000000*X**3 - 148509_8342437500*X**2*MU(1)**13 +
   7998343_5267080100*X**2*MU(1)**10 - 13961084_7728116800*X**2*MU(1)**7
   202134273_9820992000*X**2*MU(1)**4 - 37420288_0128000000*X**2*MU(1) -
   4798066_9803066000*X*MU(1)**11 + 37022219_6507412480*X*MU(1)**8 +
   207899852_9990659200*X*MU(1)**5 - 67289430_2572800000*X*MU(1)**2 -
   926478_2523150000*MU(1)**12 - 36530427_0874989600*MU(1)**9 -
   95486610_7752480000*MU(1)**6 + 148552367_0400000000*MU(1)**3 +
   20499161_7600000000 )
# I
   7
# RI
   - 3 * ( 578_7735638404_7368546875*X**7*MU(1)**18 +
   32613_0978210095_1537097200*X**7*MU(1)**15 +
   601784_0573066657_6921180928*X**7*MU(1)**12 +
   4487295_7227303470_7620082432*X**7*MU(1)**9 +
   15099632_8360751379_5627212800*X**7*MU(1)**6 +
   21261328_7596337351_2761344000*X**7*MU(1)**3 +
   7606653_5520844591_9232000000*X**7 + 38_4163350599_4391406250*X**6*
   MU(1)**19 - 18284_0582370920_9702737500*X**6*MU(1)**16 -
   330347_1903563211_8269676400*X**6*MU(1)**13 -
   2128063_5147627304_1272767744*X**6*MU(1)**10 -
   5555484_3862909288_3740917760*X**6*MU(1)**7 -
   3112862_8325527691_1396864000*X**6*MU(1)**4 +
```

```
5930168_8605404147_3024000000*X**6*MU(1) - 2646_6419981889_0450140625*
X**5*MU(1)**17 + 152860_9229541139_8114511200*X**5*MU(1)**14 +
1610552_2199253679_2050137728*X**5*MU(1)**11 +
4041418_7001497967_4338593280*X**5*MU(1)**8 -
2037818_9284562375_1016243200*X**5*MU(1)**5 -
11578164_0112406276_4564480000*X**5*MU(1)**2 +
1016_8772657574_4223906250*X**4*MU(1)**18 +
101430_1824392553_5123190000*X**4*MU(1)**15 +
1177879_4027895409_8879801600*X**4*MU(1)**12 +
6901630_5680667621_2645544960*X**4*MU(1)**9 +
23494981_3918350391_8257356800*X**4*MU(1)**6 +
32977791_2409458944_3276800000*X**4*MU(1)**3 -
4149083_7556824322_8672000000*X**4 - 23846_2501016710_9987200000*X**3*
MU(1)**16 - 588055_0219073510_6789136000*X**3*MU(1)**13 -
3626841_8379763862_7851646720*X**3*MU(1)**10 -
5987852_3791309740_5095833600*X**3*MU(1)**7 +
7361667_4105990975_0542336000*X**3*MU(1)**4 +
19269855_5707331647_7337600000*X**3*MU(1) + 3340_0646002407_9150000000
X**2*MU(1)**17 + 725705_8395413983_2704682000*X**2*MU(1)**14 +
6565815_6349343019_9114631680*X**2*MU(1)**11 +
18827835_1827607302_7017553920*X**2*MU(1)**8 +
18835887_5374646599_5220992000*X**2*MU(1)**5 +
11952182_8870538353_4592000000*X**2*MU(1)**2 -
22016_2719977040_5356950000*X*MU(1)**15 - 1141129_5136673003_198827200(
X*MU(1)**12 - 6030073_4086044445_5036541440*X*MU(1)**9 -
252730_5434635568_4694425600*X*MU(1)**6 +
```

```
25483843_6498552043_2250880000*X*MU(1)**3 -
10372709_3892060807_1680000000*X + 2361_6036695432_9317500000*
MU(1)**16 + 589110_7341348048_0684492000*MU(1)**13 +
4044913_3002729587_7786900480*MU(1)**10 + 6228772_6714240832_486922240
MU(1)**7 + 3985199_0801418551_4805248000*MU(1)**4 +
26769138_6418349089_9968000000*MU(1) ) /
( 143_7873923337_1470562500*X**7*MU(1)**18 - 949_2773130362_9605950000
X**7*MU(1)**15 + 176703_5928858805_0266969600*X**7*MU(1)**12 +
2172355_4189901433_6751493888*X**7*MU(1)**9 +
8715523_3333339335_7479321600*X**7*MU(1)**6 +
13185636_3484265923_6405248000*X**7*MU(1)**3 +
5532111_6742432430_4896000000*X**7 - 6550_5609143893_2146827500*X**6*
MU(1)**16 - 183379_8004890260_5137186000*X**6*MU(1)**13 -
1738931_0934329403_2258063616*X**6*MU(1)**10 -
6624858_1013999140_7480279040*X**6*MU(1)**7 -
8389833_7057278267_4477056000*X**6*MU(1)**4 +
1376487_4995808612_5158400000*X**6*MU(1) - 248_6548692656_5299328125*
X**5*MU(1)**17 + 97511_0724664264_3418056800*X**5*MU(1)**14 +
1135890_1499891593_2911874432*X**5*MU(1)**11 +
3626064_9579075131_4739499520*X**5*MU(1)**8 -
8875_5275055831_1868620800*X**5*MU(1)**5 -
10559224_3027224204_0668160000*X**5*MU(1)**2 -
115_2490051798_3174218750*X**4*MU(1)**18 + 17429_6809111982_9263200000*
X**4*MU(1)**15 - 535449_2590537946_0609299200*X**4*MU(1)**12 -
7027997_8046081235_1637806080*X**4*MU(1)**9 -
26943800_9773909305_2302540800*X**4*MU(1)**6 -
41448866_3877600869_6954880000*X**4*MU(1)**3 -
```

```
   29043586_2897770260_0704000000*X**4 + 14936_3489900743_1160750000*X**3
   MU(1)**16 + 697219_4612593714_7140560000*X**3*MU(1)**13 +
   8969958_4892076602_9592917760*X**3*MU(1)**10 +
   42120456_1926985846_0681113600*X**3*MU(1)**7 +
   70307325_0383357410_4346624000*X**3*MU(1)**4 +
   18148859_6875218995_2819200000*X**3*MU(1) - 2935_3827920924_9497500000
   X**2*MU(1)**17 - 409785_3162197805_6060570000*X**2*MU(1)**14 -
   7562707_0039840296_3983193600*X**2*MU(1)**11 -
   37797187_5340099410_6444994560*X**2*MU(1)**8 -
   44552065_3719684144_1247232000*X**2*MU(1)**5 +
   44450867_2643432209_6128000000*X**2*MU(1)**2 +
   66048_8159931121_6070850000*X*MU(1)**15 + 3423388_5410019009_596481600(
   X*MU(1)**12 + 18090220_2258133336_5109624320*X*MU(1)**9 +
   758191_6303906705_4083276800*X*MU(1)**6 -
   76451530_9495656129_6752640000*X*MU(1)**3 +
   31118128_1676182421_5040000000*X - 7084_8110086298_7952500000*
   MU(1)**16 - 1767332_2024044144_2053476000*MU(1)**13 -
   12134739_9008188763_3360701440*MU(1)**10 -
   18686318_0142722497_4607667200*MU(1)**7 -
   11955597_2404255654_4415744000*MU(1)**4 -
   80307415_9255047269_9904000000*MU(1) )
# TOTAL ELAPSED TIME (SEC.) WAS
# TNEW
   7.031533E1
```

4. SOURCE LISTING OF ALTRAN PROCEDURES

The ALTRAN implementation of the method described in this report is given in this section.

Index of Procedures:

```
PROCEDURE MAIN

  # MAIN PROCEDURE FOR COMPUTING PADE APPROXIMANTS FOR A POWER
  #    SERIES EXPANSION.


  INTEGER  M, N, I, LAST, NUM, DEN
  REAL  TOLD, TNEW
  LONG ALGEBRAIC (X:31, MU(1:5):31)
  LONG ALGEBRAIC  RI
  LONG ALGEBRAIC ARRAY  C, R

  LONG ALGEBRAIC ARRAY ALTRAN  TAYLOR, PADE



  # IN THIS IMPLEMENTATION, PROCEDURE TAYLOR DEFINES THE VALUES OF
  #    M AND N, AND ALSO RETURNS A POWER SERIES EXPANSION AS ITS
  #    VALUE.

  C = TAYLOR(X, MU, M, N)
  WRITE  M, N
  WRITE  "THE POWER SERIES COEFFICIENTS ARE"
  WRITE  C


  # INITIALIZE TIMING VARIABLE.

  TIME(TOLD)


  # OBTAIN THE SEQUENCE OF PADE APPROXIMANTS.

  R = PADE(C, X, M, N, .TRUE., .TRUE.)


  # PRINT OUT TIMING INFORMATION.

  TNEW = TIME(TOLD)
  WRITE  "TIME (SEC.) TO COMPUTE PADE APPROXIMANTS WAS"
  WRITE  TNEW



  # PRINT OUT THE APPROXIMANTS AS RATIONAL FUNCTIONS.

  OPTS(103, 3)  # SET NON-CANONICAL FORM FOR RATIONAL FUNCTIONS.

  LAST = DBINFO(R)(1,1)
  NUM = 1;  DEN = 2

  WRITE  "THE PADE APPROXIMANTS ARE"

  DO I = 0, LAST

    RI  =  R(I,NUM) / R(I,DEN)
    WRITE  I, RI

  DOEND
```

```
   # PRINT OUT FINAL TIMING INFORMATION.

   TNEW = TIME()
   WRITE  "TOTAL ELAPSED TIME (SEC.) WAS", TNEW


END  # END OF PROCEDURE MAIN.
```

```
PROCEDURE TAYLOR(Z, INDET, M, N)
   INTEGER  M, N
   ALGEBRAIC VALUE  Z
   ALGEBRAIC ARRAY VALUE  INDET

   # THIS PROCEDURE DEFINES THE VALUES OF M AND N, AND RETURNS AN
   #    ARRAY OF POWER SERIES COEFFICIENTS AS ITS VALUE.


   LONG ALGEBRAIC ARRAY  C



   # TAYLOR SERIES FOR THE ORDINARY THOMAS-FERMI FUNCTION.

   C  =  ( 1, 0, INDET(1), 4/3, 0, 2/5*INDET(1),
             1/3, 3/70*INDET(1)**2, 2/15*INDET(1) )
   C  =  ( C, -1/252*INDET(1)**3 + 2/27, 1/175*INDET(1)**2,
             31/1485*INDET(1) + 1/1056*INDET(1)**4,
             4/1575*INDET(1)**3 + 4/405 )
   C  =  ( C, 557/100100*INDET(1)**2 - 3/9152*INDET(1)**5,
             4/693*INDET(1) - 29/24255*INDET(1)**4 )


   TPSORD(C)  # CONVERTS C TO A TPS -- I.E. INDEXED FROM 0.


   M = 7;  N = 7

   RETURN( C )


END  # END OF PROCEDURE TAYLOR.
```

```
PROCEDURE PADE(C, Z, M, N, FRCTNS, SEQUEN)
  INTEGER VALUE  M, N
  LOGICAL VALUE  FRCTNS, SEQUEN
  ALGEBRAIC VALUE  Z
  LONG ALGEBRAIC ARRAY VALUE  C

  # COMPUTE PADE APPROXIMANTS BY A FRACTION-FREE METHOD
  #
  #    PADE APPROXIMANTS ARE COMPUTED FOR THE POWER SERIES WITH
  # COEFFICIENTS C.  LET C(S) BE THE FIRST NONZERO COEFFICIENT
  # IN C.  THE SEQUENCE OF APPROXIMANTS COMPUTED IS:
  #       (M-L,N-L), . . . , (M,N) ,
  # WHERE
  #       L = MIN(M-S,N)  IF SEQUEN IS .TRUE. ;
  #       L = 0  IF SEQUEN IS .FALSE. .
  # IF THE FIRST M+1 COEFFICIENTS ARE ALL ZERO THEN NO PADE
  # APPROXIMANTS ARE COMPUTED.
  #
  # REFERENCE:  K.O. GEDDES, SYMBOLIC COMPUTATION OF PADE
  #    APPROXIMANTS, ACM TRANS. MATH. SOFTWARE, TO APPEAR.
  #
  # INPUT PARAMETERS:
  #    C - THE ARRAY CONTAINING THE POWER SERIES COEFFICIENTS
  #       C(0), ... , C(M+N) ;
  #    Z - THE NAME OF THE INDETERMINANT TO BE USED AS THE VARIABLE
  #       IN FORMING THE PADE APPROXIMANTS -- I.E. THE POWER SERIES
  #       IS THOUGHT OF AS:
  #          C(0)  +  C(1) * Z  +  . . .  +  C(M+N) * Z**(M+N) ;
  #    M, N - DEGREE OF NUMERATOR AND DENOMINATOR, RESPECTIVELY,
  #       OF THE HIGHEST-DEGREE PADE APPROXIMANT DESIRED;
  #    FRCTNS - LOGICAL VARIABLE INDICATING IF FRACTIONS ARE TO BE
  #       CARRIED.  IF FRCTNS IS .FALSE. THEN THE POWER SERIES
  #       COEFFICIENTS C WILL BE CONVERTED TO INTEGERS OR POLY-
  #       NOMIALS WITH INTEGER COEFFICIENTS BY REMOVING THE
  #       DENOMINATOR OF THE TAYLOR POLYNOMIAL.  THIS DENOMINATOR
  #       WILL THEN BE REPLACED BEFORE RETURNING FROM THE PROCEDURE;
  #    SEQUEN - LOGICAL VARIABLE INDICATING IF THE FULL DIAGONAL
  #       SEQUENCE (SEE ABOVE) IS TO BE COMPUTED.  IF SEQUEN IS
  #       .FALSE. THEN ONLY THE (M,N) APPROXIMANT IS COMPUTED.
  #
  # OUTPUT:
  #    THE VALUE RETURNED IS A TWO-DIMENSIONAL ARRAY WITH THE
  # FIRST DIMENSION INDEXED FROM 0 TO L (WHERE L IS DEFINED
  # ABOVE) AND THE SECOND DIMENSION INDEXED FROM 1 TO 2.  EACH
  # ROW OF THIS ARRAY CONTAINS THE NUMERATOR AND DENOMINATOR,
  # RESPECTIVELY, OF A PADE APPROXIMANT, AS POLYNOMIALS IN THE
  # VARIABLE Z.  THE FIRST ROW CONTAINS THE (M-L,N-L) APPROX-
  # IMANT AND THE LAST ROW CONTAINS THE (M,N) APPROXIMANT.  THE
  # RETURNED VALUE IS NULL IF EITHER OF THE FOLLOWING CONDITIONS
  # HOLDS:
  #       THE FIRST M+1 POWER SERIES COEFFICIENTS ARE ALL ZERO,
  # OR
  #       SEQUEN = .FALSE.  AND THE HANKEL MATRIX IS SINGULAR.
  #
```

```
# ASSUMPTIONS:
#    IT IS ASSUMED THAT THE LAYOUT FOR THE ALGEBRAIC ARRAY C
# CONTAINS THE INDETERMINANT Z, WITH MAXIMUM EXPONENT M+N.
#
# PROCEDURES REQUIRED:
#    SYMTRI;  SYMSOL;  ARR AND ARRMIX (TWO SYSTEM PROCEDURES).
#
#    THE FOLLOWING DECLARATION MUST APPEAR IN THE CALLING
# PROCEDURE:
#          LONG ALGEBRAIC ARRAY ALTRAN  PADE .


INTEGER  I, J, K, KSTART, LOW, L, LENGTH, II, IJ, ORDER
INTEGER ARRAY  PERM
LOGICAL ARRAY  BLOCK
REAL  TOLD, TNEW
LONG ALGEBRAIC  P, Q, DENOM, DETERM, COEF
LONG ALGEBRAIC ARRAY  H, HSUB, DET, XNUM, R

INTEGER ARRAY ALTRAN  ARR
LONG ALGEBRAIC ARRAY ALTRAN  ARRMIX


DENOM = 1

IF (.NOT. FRCTNS)  DO

   # SET UP THE DEGREE-(M+N) TAYLOR POLYNOMIAL AND REMOVE THE
   #    DENOMINATOR.  THE POWER SERIES COEFFICIENTS ARE THEN ALL
   #    INTEGERS OR POLYNOMIALS WITH INTEGER COEFFICIENTS.  THIS
   #    MAY MAKE THE ALGORITHM MORE EFFICIENT.

   P = C(0)

   DO I = 1, M+N
      P  =  P  +  C(I) * Z**I
   DOEND

   DENOM = ADEN(P)

   C = DENOM * C

DOEND
```

```
# SET UP THE AUGMENTED HANKEL MATRIX (UPPER PART) IN SYMMETRIC
#    FORM AS REQUIRED BY PROCEDURE SYMTRI.

KSTART = M-N+1

DO I = 1, N

   K = KSTART

   DO J = I, N+1

      # THE (I,J) ENTRY OF THE HANKEL MATRIX IS EITHER 0 OR C(K).

      IF (K < 0)  H = (H, 0)
         ELSE  H = (H, C(K))

      K = K + 1

   DOEND

   KSTART = KSTART + 2

DOEND


# INITIALIZE TIMING VARIABLE.

TIME(TOLD)


# TRIANGULARIZE THE HANKEL SYSTEM.

SYMTRI(H, N, DET, PERM, BLOCK)


# PRINT OUT TIMING INFORMATION.

TNEW = TIME(TOLD)
WRITE  "TIME (SEC.) FOR TRIANGULARIZATION WAS"
WRITE  TNEW


# DEFINE L DEPENDING ON THE (LOW) ORDER OF THE POWER SERIES.

TPSORD(C, LOW)

IF (LOW > M)  GO TO EXIT  # C(0), ..., C(M) ARE ALL ZERO.


IF (SEQUEN)  L = IMIN( M-LOW, N)
   ELSE  L = 0
```

```
# EXTRACT THE PADE APPROXIMANTS.

DO K = L, 0, -1

  IF (N-K == 0)  DO

    # THE (M-K, 0) APPROXIMANT IS A TAYLOR POLYNOMIAL.

      P = C(0)
      DO I = 1, M-K
        P  =  P  +  C(I) * Z**I
      DOEND


      # REPLACE FACTOR DENOM REMOVED AT BEGINNING OF PROCEDURE.

      Q = DENOM


      R  =  (R, P, Q)

    DOEND  # END OF "THEN CLAUSE".


  ELSE DO

    DETERM = DET(N-K)

    IF (DETERM == 0)  DO

      # NEW APPROXIMANT IS THE PRECEEDING APPROXIMANT.

        IF (.NOT. SEQUEN)  GO TO EXIT  # HANKEL MATRIX WAS
                                       # SINGULAR WHEN SINGLE
                                       # APPROXIMANT SPECIFIED.

        LENGTH = 2*(L-K)

        R  =  ( R, R(LENGTH-1), R(LENGTH) )

      DOEND  # END OF "THEN CLAUSE".
```

```
ELSE DO

  # SET UP (N-K)-BY-(N-K+1) HANKEL SUBMATRIX FOR
  #    PROCEDURE SYMSOL.

    IF (K == 0)  HSUB = H

    ELSE DO

      # DIRECTORY OF INDEX VARIABLES:
      #    II - POSITION OF (I,I) ELEMENT;
      #    IJ - POSITION OF (I,J) ELEMENT.

      HSUB = 0$H  # I.E. MAKE HSUB NULL.

      II = 1

      DO I = 1, N-K

        HSUB = (HSUB, H(II))
        IJ = II

        DO J = I+1, N-K+1
          IJ = IJ + 1
          HSUB = (HSUB, H(IJ))
        DOEND

        II = II + N-I+2

      DOEND

    DOEND  # END OF "ELSE CLAUSE".


    # IF LAST SUBSYSTEM, RESET TIMING VARIABLE.

    IF (K == 0)  TIME(TOLD)


    # SOLVE THE HANKEL SUBSYSTEM.

    SYMSOL(HSUB, N-K, PERM, BLOCK, DETERM, XNUM)


    # IF LAST SUBSYSTEM, PRINT OUT TIMING INFORMATION.

    IF (K == 0)  DO

      TNEW = TIME(TOLD)
      WRITE  "TIME (SEC.) TO SOLVE ORDER-N SYSTEM WAS"
      WRITE  TNEW

    DOEND
```

```
            # DEFINE NUMERATOR POLYNOMIAL.

            P = DETERM * C(0)

            DO I = 1, M-K

               COEF = DETERM * C(I)

               DO J = 1, IMIN(I, N-K)
                  COEF = COEF - XNUM(N-K-J+1) * C(I-J)
               DOEND

               P = P + COEF * Z**I

            DOEND


            # DEFINE DENOMINATOR POLYNOMIAL.

            Q = DETERM

            DO I = 1, N-K
               Q = Q - XNUM(N-K-I+1) * Z**I
            DOEND


            # REPLACE FACTOR DENOM REMOVED AT BEGINNING
            #    OF PROCEDURE.

            Q = DENOM * Q


            # ADD THE (M-K, N-K) APPROXIMANT INTO THE LIST R.

            R = (R, P, Q)

         DOEND  # END OF "ELSE CLAUSE".

      DOEND  # END OF "ELSE CLAUSE".

   DOEND  # END OF K-LOOP.


   # RESHAPE R INTO A TWO-DIMENSIONAL ARRAY.

   R = ARRMIX( ARR( (0,1), (L,2) ), R )

EXIT: RETURN( R )


END  # END OF PROCEDURE PADE.
```

```
PROCEDURE SYMTRI(A, N, DET, PERM, BLOCK)
  INTEGER VALUE  N
  INTEGER ARRAY  PERM
  LOGICAL ARRAY  BLOCK
  LONG ALGEBRAIC ARRAY  A, DET

  # SYMMETRIC TRIANGULARIZATION
  #
  # PERFORM ONE-STEP/TWO-STEP FRACTION-FREE GAUSSIAN ELIMINATION
  #    TO TRIANGULARIZE A SYMMETRIC N-BY-(N+1) AUGMENTED MATRIX
  #    STORED IN A ONE-DIMENSIONAL ARRAY AS DESCRIBED BELOW
  #    (SEE ASSUMPTIONS).
  #
  # REFERENCE:  K.O. GEDDES, SYMBOLIC COMPUTATION OF PADE
  #    APPROXIMANTS, ACM TRANS. MATH. SOFTWARE, TO APPEAR.
  #
  # INPUT PARAMETERS:
  #    A - A ONE-DIMENSIONAL ARRAY CONTAINING THE ELEMENTS OF THE
  #       N-BY-(N+1) AUGMENTED MATRIX IN SYMMETRIC FORM, AS DES-
  #       CRIBED BELOW (SEE ASSUMPTIONS);
  #    N - THE ORDER OF THE MATRIX TO BE TRIANGULARIZED.
  #
  # OUTPUT PARAMETERS:
  #    A - THE TRIANGULARIZED AUGMENTED MATRIX (NOTE THAT SYMMETRIC
  #       2-BY-2 BLOCKS MAY APPEAR ON THE DIAGONAL AS INDICATED BY
  #       BOOLEAN VECTOR BLOCK);
  #    DET - VECTOR OF LENGTH N WHOSE I-TH ENTRY IS THE DETERMINANT
  #       OF THE ORDER-I LEADING PRINCIPAL SUBMATRIX OF THE ORIGINAL
  #       INPUT MATRIX (THUS DET(N) IS THE DETERMINANT OF THE GIVEN
  #       INPUT MATRIX);
  #    PERM - PERMUTATION VECTOR OF LENGTH N SPECIFYING THE ORDER
  #       OF THE UNKNOWNS IN THE TRIANGULARIZED LINEAR SYSTEM;
  #    BLOCK - BOOLEAN VECTOR OF LENGTH N SUCH THAT IF BLOCK(I) IS
  #       .TRUE. THEN THE I-TH DIAGONAL ELEMENT BELONGS TO A SYM-
  #       METRIC BLOCK (NOTE THAT THE UPPER LEFT ELEMENT OF A 2-BY-2
  #       BLOCK WILL ALWAYS BE ZERO).
  #
  # ASSUMPTIONS:
  #    IT IS ASSUMED THAT THE UPPER TRIANGULAR PART OF THE AUG-
  # MENTED MATRIX IS STORED ROW BY ROW IN THE ONE-DIMENSIONAL
  # ARRAY A OF LENGTH  N(N+3)/2 , WITH THE I-TH ROW OCCUPYING
  # POSITIONS IBEGIN TO IEND WHERE  IBEGIN = (I-1)(2N-I+4)/2 + 1
  # AND  IEND = I(2N-I+3)/2 .  THUS THE (I,J) ELEMENT IS LOCATED
  # IN POSITION  IBEGIN+J-I  OF A,  FOR  I <= J <= N+1 .  NOTE
  # THAT THE NUMBER OF ELEMENTS STORED FOR ROW I IS  N-I+2 .
  #
  # PROCEDURES REQUIRED:
  #    SYMINT;  ARR (A SYSTEM PROCEDURE).
```

```
      INTEGER  I, J, K, KK, KI, KJ, IJ, K1I, K1J, K1K, K1K1, K1K2,
                                                    K2K2, NN, TEMP
      LONG ALGEBRAIC  DIVISR, DETERM, MI1, MI2

      INTEGER ARRAY ALTRAN  ARR



      # SET UP ARRAYS:
      #    DET - LENGTH N, EACH ELEMENT 1 (COULD BE ANY NONZERO VALUE);
      #    BLOCK - LENGTH N, EACH ELEMENT .FALSE.;
      #    PERM - LENGTH N, I-TH ELEMENT EQUALS I.

      DET = N$(1)
      BLOCK = N$(.FALSE.)
      IF (N > 0)  PERM = ARR(1,N)
      DO I = 1, N
        PERM(I) = I
      DOEND



      # ELIMINATION LOOP.

      DIVISR = 1
      K = 0

KLOOP: IF (K > N-2)  GO TO FINAL

      # DIRECTORY OF INDEX VARIABLES:
      #    KK - POSITION OF (K,K) ELEMENT;
      #    K1K1 - POSITION OF (K+1,K+1) ELEMENT;
      #    K1K2 - POSITION OF (K+1,K+2) ELEMENT;
      #    K2K2 - POSITION OF (K+2,K+2) ELEMENT.

      IF (K == 0)  K1K1 = 1
        ELSE  K1K1 = KK + N-K+2
      K1K2 = K1K1 + 1
      K2K2 = K1K1 + N-K+1
```

```
DETERM = A(K1K1)*A(K2K2) - A(K1K2)**2

IF (DETERM == 0  .AND.  A(K1K1) <> 0)  DO

  # ONE-STEP ELIMINATION.

    K = K+1
    KK = K1K1

    # DIRECTORY OF INDEX VARIABLES:
    #    KK - POSITION OF (K,K) ELEMENT;
    #    KI - POSITION OF (K,I) ELEMENT;
    #    KJ - POSITION OF (K,J) ELEMENT;
    #    IJ - POSITION OF (I,J) ELEMENT.

    KI = KK
    IJ = K2K2 - 1

    DO I = K+1, N

      KJ = KI
      KI = KI + 1

      DO J = I, N+1
        KJ = KJ + 1
        IJ = IJ + 1
        A(IJ) = ( A(KK)*A(IJ) - A(KI)*A(KJ) ) / DIVISR
      DOEND

    DOEND

    # NEXT DIVISOR IS THE (K,K) ELEMENT;  ALSO SAVE IT AS THE
    #    ORDER-K DETERMINANT UNLESS ALREADY SET TO ZERO (DUE TO
    #    INTERCHANGES).

    DIVISR = A(KK)
    IF ( DET(K) <> 0 )  DET(K) = A(KK)

  DOEND  # END OF ONE-STEP ELIMINATION.
```

```
ELSE DO

  # TWO-STEP ELIMINATION.

    K = K+2
    KK = K2K2

    # DIRECTORY OF INDEX VARIABLES:
    #    KK - POSITION OF (K,K) ELEMENT;
    #    K1K1 - POSITION OF (K-1,K-1) ELEMENT;
    #    K1I - POSITION OF (K-1,I) ELEMENT;
    #    K1J - POSITION OF (K-1,J) ELEMENT;
    #    K1K - POSITION OF (K-1,K) ELEMENT;
    #    KI - POSITION OF (K,I) ELEMENT;
    #    KJ - POSITION OF (K,J) ELEMENT;
    #    IJ - POSITION OF (I,J) ELEMENT.


    # ENSURE NONSINGULAR 2-BY-2 PIVOT.

    IF (DETERM == 0)  DO

      # NOTE: THE (K-1,K-1) AND (K-1,K) ELEMENTS MUST BE
      #    BOTH ZERO.

      DET(K-1) = 0
      DET(K) = 0

      K1I = K1K1 + 1

      DO I = K+1, N
        K1I = K1I + 1
        DETERM = - A(K1I)**2
        IF (DETERM == 0)  DET(I) = 0
          ELSE  GO TO SWITCH
      DOEND

      # ALL REMAINING SUBSYSTEMS ARE SINGULAR SO EXIT FROM
      #    PROCEDURE.

      GO TO EXIT


      # INTERCHANGE ROWS AND COLUMNS K AND I.

SWITCH: SYMINT(A, N, K, I)
      TEMP = PERM(K);  PERM(K) = PERM(I);  PERM(I) = TEMP

    DOEND  # 2-BY-2 PIVOT IS NOW NONSINGULAR.
```

```
# UPDATE ROWS BELOW ROW K.

K1K = K1K1 + 1
K1I = K1K
KI = KK
IJ = KK + N-K+1

DETERM = DETERM / DIVISR

DO I = K+1, N

   K1J = K1I
   KJ = KI
   K1I = K1I + 1
   KI = KI + 1

   MI1 = ( A(K1I)*A(KK) - A(KI)*A(K1K) ) / DIVISR
   MI2 = ( A(KI)*A(K1K1) - A(K1I)*A(K1K) ) / DIVISR

   DO J = I, N+1
      K1J = K1J + 1
      KJ = KJ + 1
      IJ = IJ + 1
      A(IJ) = ( DETERM*A(IJ) - MI1*A(K1J) - MI2*A(KJ) ) /
                                                    DIVISR
   DOEND

DOEND


# UPDATE ROW K IF POSSIBLE.

IF (A(K1K1) <> 0)  DO

   # ONE-STEP UPDATE FOR ROW K.

      A(KK) = DETERM

      K1J = K1K
      KJ = KK

      DO J = K+1, N+1
         K1J = K1J + 1
         KJ = KJ + 1
         A(KJ) = ( A(K1K1)*A(KJ) - A(K1K)*A(K1J) ) / DIVISR
      DOEND

   DOEND  # END OF "THEN CLAUSE".
```

```
            ELSE DO

               # SIGNAL 2-BY-2 BLOCK ON DIAGONAL.

                  BLOCK(K-1) = .TRUE.
                  BLOCK(K) = .TRUE.

               DOEND  # END OF "ELSE CLAUSE".


            # CURRENT DETERMINANT IS NEXT DIVISOR;  SAVE DETERMINANT
            #   VALUES UNLESS ALREADY SET TO ZERO (DUE TO INTERCHANGES).

            DIVISR = DETERM
            IF ( DET(K-1) <> 0 )  DET(K-1) = A(K1K1)
            IF ( DET(K) <> 0 )  DET(K) = DETERM


         DOEND  # END OF TWO-STEP ELIMINATION.


      GO TO KLOOP

      # END OF ELIMINATION LOOP.



      # IF FINAL VALUE OF K IS N-1 THEN DET(N) WAS NOT ASSIGNED;  ITS
      #    VALUE IS THE (N,N) ELEMENT.

FINAL: IF (K == N-1)  DO
         IF (N == 1)  NN = 1
            ELSE  NN = KK + 3
         DET(N) = A(NN)
      DOEND


 EXIT:

   END  # END OF PROCEDURE SYMTRI.
```

```
PROCEDURE SYMINT(A, N, K, I)
  INTEGER VALUE  N, K, I
  LONG ALGEBRAIC ARRAY  A

  # SYMMETRIC INTERCHANGE
  #
  # INTERCHANGE COLUMNS K AND I AND THEN ROWS K AND I IN THE SYM-
  #    METRIC N-BY-(N+1) AUGMENTED MATRIX STORED IN THE ONE-DIMEN-
  #    SIONAL ARRAY A AS DESCRIBED IN PROCEDURE SYMTRI.
  #
  # INPUT PARAMETERS:
  #    A - THE ARRAY IN WHICH THE INTERCHANGE IS TO TAKE PLACE;
  #    N - THE ORDER OF THE MATRIX;
  #    K, I - INDICES OF THE ROWS AND COLUMNS TO BE INTERCHANGED.
  #
  # OUTPUT PARAMETERS:
  #    A - THE ARRAY AFTER THE INTERCHANGES.
  #
  # ASSUMPTIONS:
  #    IT IS ASSUMED THAT K IS LESS THAN I.


  INTEGER  J, II, IJ, JI, KK, KJ, JK, ROWLEN
  LONG ALGEBRAIC  TEMP



  # DIRECTORY OF INDEX VARIABLES:
  #    II - POSITION OF (I,I) ELEMENT;
  #    IJ - POSITION OF (I,J) ELEMENT;
  #    JI - POSITION OF (J,I) ELEMENT;
  #    KK - POSITION OF (K,K) ELEMENT;
  #    KJ - POSITION OF (K,J) ELEMENT;
  #    JK - POSITION OF (J,K) ELEMENT.


  ROWLEN = N+1
  JK = K
  JI = I

  DO J = 1, K-1

    TEMP = A(JK);  A(JK) = A(JI);  A(JI) = TEMP

    ROWLEN = ROWLEN - 1
    JK = JK + ROWLEN
    JI = JI + ROWLEN

  DOEND
```

```
      KK = JK
      II = (I-1)*(2*N-I+4)/2 + 1
      TEMP = A(KK);  A(KK) = A(II);  A(II) = TEMP

      KJ = KK

      DO J = K+1, I-1

         KJ = KJ + 1
         ROWLEN = ROWLEN - 1
         JI = JI + ROWLEN

         TEMP = A(KJ);  A(KJ) = A(JI);  A(JI) = TEMP

      DOEND

      KJ = KJ + 1
      IJ = II

      DO J = I+1, N+1

         KJ = KJ + 1
         IJ = IJ + 1

         TEMP = A(KJ);  A(KJ) = A(IJ);  A(IJ) = TEMP

      DOEND

END  # END OF PROCEDURE SYMINT.
```

```
PROCEDURE SYMSOL(A, N, PERM, BLOCK, DETERM, XNUM)
  INTEGER VALUE  N
  INTEGER ARRAY VALUE  PERM
  LOGICAL ARRAY VALUE  BLOCK
  LONG ALGEBRAIC VALUE  DETERM
  LONG ALGEBRAIC ARRAY VALUE  A
  LONG ALGEBRAIC ARRAY  XNUM

  # SYMMETRIC SOLVE
  #
  # SOLVE THE N-BY-(N+1) LINEAR SYSTEM WHICH HAS BEEN TRIANGULARIZED
  #    BY PROCEDURE SYMTRI.  FRACTION-FREE BACK-SUBSTITUTION IS USED
  #    TO COMPUTE THE "NUMERATORS" OF THE SOLUTION INTO ARRAY XNUM
  #    -- THE COMPLETE SOLUTION WOULD BE XNUM/DETERM.
  #
  # INPUT PARAMETERS:
  #    A - THE ONE-DIMENSIONAL ARRAY CONTAINING THE ELEMENTS OF THE
  #       N-BY-(N+1) AUGMENTED MATRIX IN SYMMETRIC FORM, AS TRIANG-
  #       ULARIZED BY PROCEDURE SYMTRI;
  #    N - THE ORDER OF THE LINEAR SYSTEM TO BE SOLVED;
  #    PERM - PERMUTATION VECTOR OF LENGTH N SPECIFYING THE ORDER
  #       OF THE UNKNOWNS IN THE TRIANGULARIZED LINEAR SYSTEM;
  #    BLOCK - BOOLEAN VECTOR OF LENGTH N SUCH THAT IF BLOCK(I) IS
  #       .TRUE. THEN THE I-TH DIAGONAL ELEMENT BELONGS TO A SYM-
  #       METRIC 2-BY-2 BLOCK (WITH UPPER LEFT ELEMENT ZERO);
  #    DETERM - THE VALUE OF THE DETERMINANT OF THE N-BY-N COEF-
  #       FICIENT MATRIX.
  #
  # OUTPUT PARAMETERS:
  #    XNUM - VECTOR OF LENGTH N CONTAINING THE "NUMERATORS" OF THE
  #       SOLUTION VECTOR -- THE COMPLETE SOLUTION IS XNUM/DETERM.
  #
  # ASSUMPTIONS:
  #    IT IS ASSUMED THAT THE TRIANGULARIZED AUGMENTED MATRIX IS
  # STORED IN SYMMETRIC FORM AS DESCRIBED IN PROCEDURE SYMTRI.
  #
  # PROCEDURES REQUIRED:
  #    ARR (A SYSTEM PROCEDURE).


  INTEGER  K, J, NN, N1N, KK, K1K, KJ, K1J, ROWLEN
  LONG ALGEBRAIC  SUM

  INTEGER ARRAY ALTRAN  ARR


  # INITIALIZE XNUM TO BE AN ARRAY OF APPROPRIATE LENGTH.

  XNUM = ARR(1, N)
```

```
# DIRECTORY OF INDEX VARIABLES:
#     NN - POSITION OF (N,N) ELEMENT;
#     NN+1 - POSITION OF (N,N+1) ELEMENT;
#     N1N - POSITION OF (N-1,N) ELEMENT;
#     N1N+1 - POSITION OF (N-1,N+1) ELEMENT;
#     KK - POSITION OF (K,K) ELEMENT;
#     K1K - POSITION OF (K-1,K) ELEMENT;
#     KJ - POSITION OF (K,J) ELEMENT;
#     K1J - POSITION OF (K-1,J) ELEMENT;
#     ROWLEN - LENGTH OF ROW K .


NN = N*(N+3)/2 - 1
N1N = NN - 2


# SOLVE BOTTOM EQUATION OR BOTTOM TWO EQUATIONS, AS REQUIRED.

IF ( BLOCK(N) )  DO

   # SOLVE 2-BY-2 BLOCK ON BOTTOM DIAGONAL.

      XNUM(PERM(N))  =  DETERM * A(N1N+1) / A(N1N)

      XNUM(PERM(N-1))  =  ( DETERM * A(NN+1)  -
                            A(NN) * XNUM(PERM(N)) ) / A(N1N)

      K = N - 2

      ROWLEN = 4
      KK = N1N - 5
      K1K = KK - 4

   DOEND  # END OF "THEN CLAUSE".


ELSE DO

   # SOLVE BOTTOM EQUATION.

      XNUM(PERM(N))  =  A(NN+1)

      K = N - 1

      ROWLEN = 3
      KK = N1N - 1
      K1K = KK - 3

   DOEND  # END OF "ELSE CLAUSE".
```

```
      # BACK-SUBSTITUTION LOOP.

KLOOP: IF (K == 0)  GO TO EXIT

      IF ( BLOCK(K) )  DO

        # DIAGONAL CONTAINS 2-BY-2 BLOCK.

          K1J = K1K

          SUM = 0

          DO J = K+1, N
             K1J = K1J + 1
             SUM = SUM + A(K1J)*XNUM(PERM(J))
          DOEND

          K1J = K1J + 1  # K1J IS NOW POSITION OF (K-1,N+1) ELEMENT.
          XNUM(PERM(K)) = ( DETERM*A(K1J) - SUM ) / A(K1K)


          KJ = KK - 1

          SUM = 0

          DO J = K, N
             KJ = KJ + 1
             SUM = SUM + A(KJ)*XNUM(PERM(J))
          DOEND

          KJ = KJ + 1  # KJ IS NOW POSITION OF (K,N+1) ELEMENT.
          XNUM(PERM(K-1)) = ( DETERM*A(KJ) - SUM ) / A(K1K)


          K = K - 2

          ROWLEN = ROWLEN + 2
          KK = K1K - ROWLEN - 1
          K1K = KK - ROWLEN


        DOEND  # END OF "THEN CLAUSE".
```

```
      ELSE DO

        # NORMAL BACK-SUBSTITUTION.

          KJ = KK

          SUM = 0

          DO J = K+1, N
            KJ = KJ + 1
            SUM = SUM + A(KJ)*XNUM(PERM(J))
          DOEND

          KJ = KJ + 1  # KJ IS NOW POSITION OF (K,N+1) ELEMENT.
          XNUM(PERM(K)) = ( DETERM*A(KJ) - SUM ) / A(KK)


          K = K - 1

          ROWLEN = ROWLEN + 1
          KK = K1K - 1
          K1K = KK - ROWLEN


        DOEND  # END OF "ELSE CLAUSE".


      GO TO KLOOP

EXIT:

    END  # END OF PROCEDURE SYMSOL.
```

REFERENCES

1. BAKER,G.A., and GAMMEL, J.L. The Padé Approximant in Theoretical Physics. Academic Press, New York, 1970.

2. BAREISS, E.H. Sylvester's identity and multistep integer-preserving Gaussian elimination. Math. Comp. 22 (1968), 565-578.

3. BIRKHOFF, G. and BARTEE, T.C. Modern Applied Algebra. McGraw-Hill, New York, 1970.

4. BUNCH, J.R. and PARLETT, B.N. Direct methods for solving symmetric indefinite systems of linear equations. SIAM J. Numer. Anal. 8 (1971), 639-655.

5. FOX, L. An Introduction to Numerical Linear Algebra. Oxford University Press, London, 1964.

6. GANTMACHER, F.R. The Theory of Matrices, vol. 2. Chelsea, New York, 1959.

7. GEDDES, K.O. Algorithms for analytic approximation. Ph.D. Thesis, Dept. of Computer Science, University of Toronto, 1973.

8. GEDDES, K.O. Symbolic computation of Padé approximants. ACM Trans. Math. Software, to appear.

9. GRAGG, W.B. The Padé table and its relation to certain algorithms of numerical analysis. SIAM Rev. 14(1972), 1-62.

10. LIPSON, J.D. Symbolic methods for the computer solution of linear equations with applications to flowgraphs. Proc. 1968 Summer Institute on Symbolic Mathematical Computation, R.G. Tobey, ed., IBM Programming Lab. Rep. FSC69-0312, 1969, pp. 233-303.

11. NORMAN, A.C. Computing with formal power series. ACM Trans. Math. Software 1, 4 (Dec. 1975), 346-356.

12. WALL, H.S. Analytic Theory of Continued Fractions. Van Nostrand, New York, 1948.

13. WYNN, P. The rational approximation of functions which are formally defined by a power series expansion. Math. Comp. 14 (1960), 147-186.

14. ZIPPEL, R. Univariate power series expansions in algebraic manipulation. Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation, ACM, New York, 1976, pp. 198-208.