The Application of Symbolic Mathematics to a Singular Perturbation Problem

by

Ellis Horowitz[†]

TR 72 - 130

February 11, 1972
(Revised May, 1972)

Computer Science Department
Cornell University
Ithaca, New York 14850

[†] To be presented at ACM72, Boston, August 14, 1972 session on Symbolic and Algebraic Manipulation.

The Application of Symbolic Mathematics to a Singular Perturbation Problem

Ellis Horowitz

Abstract:

A basic technique for the numerical solution of ordinary differential equations is to express them as a singular perturbation problem. However, computational studies indicate that the resultant matrix equations which must be solved are often highly ill-conditioned. In this paper a particular singular perturbation problem which was shown to be ill-conditioned using 8 numerical methods is solved by symbolic techniques. These techniques lead both to an analytic proof of the solution plus to the precise knowledge of the asymptotic behavior of the solution vector as it converges. The difficulties encountered in solving the problem symbolically are disucssed. Then several conclusions are drawn about the merits of a symbolic versus a numeric approach when applied to the solution of linear systems. Finally some advice and warnings to both the user and the designer of symbol manipulation systems are given concerning their goals and expectations when large matrix equations are to be solved.

Key Words and Phrases:

Symbol manipulation, exact solution of linear systems, singular perturbation problem, Navier-Stokes equations

CR Categories: 4.22, 5.17

1. Introduction

At the SYMSAM-II Conference held in Los Angeles in March, 1971 a problem was posed to the entire membership of the conference. A visiting numerical analyst, after viewing all of the sessions and analyzing several of the symbol manipulation systems which were being demonstrated, was still skeptical about the possibility of getting his problem solved using the existing techniques. He expressed his doubts in the open discussion which was held at the end of the conference and called for the closer working together of numeral analysts and designers of symbol manipulation systems.

We attempted to solve his problem using the SAC-1 system which is currently available at Cornell, [HOR70]. In fact we were able to produce the desired solution as well as some insight into the analytical structure of his mathematical expressions. However, the solution was not obtained as a direct consequence of existing routines within the SAC-1 hierarchy of operations. Rather it was an iterative process where at each stage we encountered new difficulties and discovered new strategies which eventually did lead to a solution. Throughout we faced the classical symbol manipulation difficulty of intermediate coefficient growth as well as the time and space constraints that are perrenially present when solving large systems of linear equations. Thus we gained some invaluable experience about the usefulness and desirability of symbolic techniques versus the efficiency of standard numerical routines. For the problem dealt with in this paper 8 <u>numerical</u> methods were tried to solve it and all proved hopelessly inadequate. And yet the same difficulties which caused the failure of the numerical techniques were to cause severe strains, even on a symbol manipulation system which is tailor-made to handle polynomial and rational function manipulations.

In Section 2 the precise mathematical statement of the problem is given. In Section 3 the difficulties encountered in its solution are traced. Then the results are presented along with a proof of the solution. This proof was motivated by the computational results which were obtained from solving several subsystems of the original problem. In Section 4, some observations and conclusions are made concerning the usefulness of symbol manipulation systems in solving problems of this nature. Appendices contain more detailed descriptions of the programs and of the output.

2. Problem Statement

This problem was initially given by F. Dorr and the formulation we give is taken primarily from the one he gives in [DOR71].

Suppose we consider the following form of the steady-state Navier-Stokes equations:

$$\psi_{xx} + \psi_{yy} = -\omega \qquad \text{in } G ,$$

$$\omega_{xx} + \omega_{yy} + R(\psi_x\omega_y - \psi_y\omega_x) = 0 \qquad \text{in } G ,$$

where $G = (0,1) \times (0,1)$ and R is the Reynolds number.

The problem we wish to solve is to determine the asymptotic behavior of $\psi(x,y) = \psi(x,y,R)$ and $\omega(x,y) = \omega(x,y,R)$ as $R \to \infty$. By imposing the Dirichlet boundary conditions

$$\begin{aligned}
\psi &= 0 && \text{on } \partial G \\
\omega &= 1 && \text{on } \partial G \cap \{(x,y)\ x = 0 \text{ or } y = 1\} \\
\omega &= -1 && \text{on } \partial G \cap \{(x,y)\ x = 1 \text{ or } y = 0\}
\end{aligned}$$

we can extend the solution functions $\psi(x,y)$ and $\omega(x,y)$ across the line $x = y$ by skew symmetry. Thus, we arrive at the more general boundary value problem: let $T = \{(x,y)\,|\,0 < x < y < 1\}$, then

$$\psi_{xx} + \psi_{yy} = -\omega \qquad \text{in } T,$$

$$\omega_{xx} + \omega_{yy} + R(\psi_x\omega_y - \psi_y\omega_x) = 0 \qquad \text{in } T,$$

(1)
$$\begin{aligned}
\psi &= 0 && \text{on } \partial T \\
\omega &= 1 && \text{on } \partial T \cap \{(x,y)\,|\,x = 0 \text{ or } y = 1\} \\
\omega &= 0 && \text{on } \partial T \cap \{(x,y)\,|\,x = y\}
\end{aligned}$$

One method for obtaining approximate solutions to Equation (1) used by Greenspan, is to define a uniform grid of mesh points on T with mesh width $h > 0$. The differential equations are replaced by finite difference equations and then he examines the behavior of a set of discrete solutions, as $R \to +\infty$ with $h > 0$ held fixed. In this paper we are concerned with the one dimensional analog of equation (1), namely the 2 parameter problem:

$$\epsilon w''(t) + (1/2 - t)w'(t) = 0 \qquad (0 < t < 1)$$

$$w(0) = v_0 \ , \ w(1) = v_1 \tag{2}$$

Using the same technique as Greenspan a uniform grid of mesh points on the interval $[0,1]$ with mesh width $h > 0$ is defined. If the differential equation is replaced by difference equations defined below, then we wish to study the behavior of the solution to the difference equation, again as $\epsilon \to 0+$ with $h > 0$ held fixed. In [DOR71], Section 3 Dorr shows that the solutions to the differential equation and to the difference equation have the same asymptotic behavior. Let us now define more precisely these difference equations.

Adopting the notation in [DOR71] , we have a mesh size $h = 1/(N+1)$, N a positive integer, with mesh points $x_j = jh$. Then we define the difference operators

$$W_x(t) = \left[\frac{W(t + h) - W(t)}{h}\right],$$

$$W_{\bar{x}}(t) = \left[\frac{W(t) - W(t - h)}{h}\right],$$

$$g(t)\,\frac{\partial W(t)}{\partial x} = g(t)W_x(t) \quad \text{if} \quad g(t) \geq 0\,,$$

$$= g(t)W_{\bar{x}}(t) \quad \text{if} \quad g(t) < 0,$$

$$W_{\hat{x}}(t) = \left[\frac{W(t + h) - W(t - h)}{2h}\right]$$

$$W_{x\bar{x}}(t) = \left[\frac{W(t + h) - 2W(t) + W(t - h)}{h^2}\right].$$

Several important theoretical properties now exist between $W(t)$, $W_x(t)$, $W_{\bar{x}}(t)$, $W_{\hat{x}}(t)$ and $W_{x\bar{x}}(t)$. These properties are important for the theory behind this problem and are stated and proven in [DOR71] , Section 2.

We now show how the solution of the difference equation and Equation (2) is directly related to the solution of a class of linear systems. Let A be an $N \times N$ tri-diagonal matrix of the form

$$A = \begin{bmatrix} b_1 & C_1 & & & & & 0 \\ a_2 & b_2 & C_2 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & a_{N-1} & b_{N-1} & C_{N-1} \\ 0 & & & & & a_N & b_N \end{bmatrix}$$

Now let L_h be a three-point difference operator

$$L_h W(x_j) = a_j W(x_{j-1}) + b_j W(x_j) + C_j (W_{x_{j+1}}) \quad .$$

It now follows that the boundary-value problem

$$L_h W(x_j) = f_j \qquad (1 \leq j \leq N)$$

$$W(0) = f_0 \qquad W(1) = f_{N+1}$$

is equivalent to the matrix equation

$$AW = \bar{f}$$

where

$$W = \begin{bmatrix} W(x_1) \\ W(x_2) \\ \cdot \\ \cdot \\ \cdot \\ W(x_N) \end{bmatrix}$$

and

$$\begin{aligned} \bar{f}_j &= f_1 - a_1 f_0 && \text{if} \quad j = 1\,, \\ &= f_j && \text{if} \quad 2 \le j \le N-1\,, \\ &= f_N - c_N f_{N+1} && \text{if} \quad j = N\,. \end{aligned}$$

More particularly, we define a_j, b_j and c_j as

$$\begin{aligned} a_j &= -\frac{\epsilon}{h^2} && \text{if} \quad 1 \le j \le \left[\frac{N+1}{2}\right]\,, \\ &= -\frac{\epsilon}{h^2} + \frac{1}{h}(1/2 - x_j) && \text{if} \quad \left[\frac{N+1}{2}\right] + 1 \le j \le N\,, \end{aligned}$$

$$c_j = -\frac{\epsilon}{h^2} - \frac{1}{h}(1/2 - x_j) \qquad \text{if} \quad 1 \le j \le \frac{N+1}{2}$$

$$= -\frac{\epsilon}{h^2} \qquad \text{if} \quad \frac{N+1}{2} + 1 \le j \le N$$

and $b_j = -(a_j + c_j)$.

Thus the solution to the differential equation can be obtained by solving the matrix equation $AW = \overline{f}$. Initially this might seem like a straightforward problem. The matrix A is, in addition to being tridiagonal, irreducibly diagonally dominant and hence non-singular. However, in [DOR71] Dorr gives the computational results that were obtained using the following 8 methods:

1) Double-precision Gaussian elimination with no interchanges, in the form $W_i = d_i + (1 + e_i)W_{i+1}$.

2) Gaussian elimination with row interchanges.

3) Gaussian elimination with no interchanges, in the form $W_i = d_i + e_iW_{i-1}$.

4) Gaussian elimination with no interchanges, in the form $W_i = d_i + e_iW_{i+1}$.

5) Gaussian elimination with no interchanges, in the form $W_i = d_i + (1 + e_i)W_{i+1}$.

6) A method due to Babuska which can be derived by combining methods 3) and 4).

7) A method that symmetrizes A , and then uses Cholesky decomposition.

8) Gauss-Seidel iteration with initial guess $W_i^{(0)} \equiv 1$. The criterion for convergence is that the maximum relative difference between successive iterates be less than 10^{-12} .

Summarizing briefly for $\epsilon = .01$ the Gauss-Seidel method converged very slowly. Using $\epsilon = .001$ it actually "converged" to the wrong solution. This apparently occurred because the correction after each iteration was so small that the difference between successive iterates was small and thus it seemed as if convergence had happened. An approximate condition number was computed to be 1.1×10^{14} thus substantiating the computational instability which was observed. Thus the initial problem consisted of solving this linear system for increasing values of N and then to determine the asymptotic value of the solution vector as $N \to \infty$.

3. Problem Solution

The SAC-1 System for symbolic and algebraic calculation is a library of FØRTRAN subroutines which provide for a variety of operations on infinite precision integers, multivariate polynomials and rational functions, [COL71]. One of the SAC-1 subsystems provides for operations on linear systems whose elements are, in general, multivariate polynomials. In this section, the aspects of the solution of the singular perturbation problem which was described in the previous section will be discussed.

The initial problem called for the solution of the $N = 100 \times 100$ system. This requires $(100)^2 = 10{,}000$ cells and for SAC-1 on the IBM/360, see [HOR70], each cell occupies 3 full words of memory. In addition to these 30,000 words, the tridiagonal elements are initially polynomials of degree = 1. Each such polynomial requires 7 cells and there are 3N-2 of them. Thus, $3(N^2 + 7(3N-2)) = 36{,}258$ words are necessary just to hold the input matrix.

The next consideration is the method of solution to be used. In [McC71], McClellan shows that using either Gaussian elimination over the rationals or the exact division method over the integers gives the same computing time. For an $n \times n$ system, this time is $O(n^5 d^2)$ where d is an appropriate bound on the precision of the coefficients. A more efficient algorithm is obtainable by using the methods of modular arithmetic and this is fully described in [McC71]. Despite the increased efficiency of this method, the modular algorithms could not be used. This was due to the fact that they require two copies of the input matrix which doubles the core requirements to approximately 72K words. Since the SAC-1 library requires an additional 30K words the total storage requirements, even before we start to compute is 102K. The <u>maximum</u> amount of available core on the IBM 360/65 at Cornell is 128K words. Since the polynomials are bound to grow as the computations proceed it soon became clear that using the regularly provided linear systems package was infeasible.

In order to make use of the relative sparseness of the matrix, new programs were written which used the following recurrence relations to produce a triangularized system. Given b_i, $1 \le i \le n$, c_i, $1 \le i \le n-1$, a_i, $2 \le i \le n$, and v_i, $1 \le i \le n$ as described in Section 2 we wish to solve the system

$$\begin{bmatrix} b_1 & c_1 & 0 & & & \bigcirc \\ a_2 & b_2 & c_2 & & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot \\ & & & & & c_{n-1} \\ \bigcirc & & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix}$$

thus we have

$$(3) \qquad \begin{aligned} b_i &\leftarrow c_{i-1}\, a_i - b_{i-1}\, b_i &&, \quad 2 \le i \le n \\ c_i &\leftarrow -b_{i-1}\, c_i &&, \quad 2 \le i \le n-1 \\ v_i &\leftarrow a_i\, v_{i-1} - b_{i-1}\, v_i &&, \quad 2 \le i \le n \end{aligned}$$

which produces the triangularized system

$$\begin{bmatrix} b_1 & c_1 & & & & \bigcirc \\ & b_2 & c_2 & & & \\ & & b_3 & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & c_{n-1} \\ \bigcirc & & & & & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix}$$

Using the formulas of Eq. (3) the storage requirements are now substantially reduced. We now need to store only the 3n-2 univariate polynomials which are initially given. Then for each new (b_i, c_i, v_i) that is computed we can discard the old $(b_{i-2}, c_{i-2}, v_{i-2})$. Now let us try to bound the degree and the coefficients of (b_i, c_i, v_i). Since $\deg(b_1) = \deg(c_1) = \deg(v_1) = 1$, it follows from Equation (3) that $\deg(b_i) \le i$ for $1 \le i \le n$. Further, if d bounds in absolute value the sum of the coefficients of the given (b_i, c_i, v_i), and if we denote this by $\text{prec}(b_i)$, $\text{prec}(c_i)$, $\text{prec}(v_i) \le d$, then by Eq(3)

$$\text{prec}(b_i) = 2(\text{prec}(b_{i-1}))^2 = \ldots = 2^{2^{i-1}-1} d^{2^{i-1}} = \tfrac{1}{2}(2d)^{2^{i-1}}$$

$$\text{prec}(c_i) = \text{prec}(b_{i-1})^2 = \ldots = d^{2^{i-1}}$$

$$\mathrm{prec}(v_i) = 2(\mathrm{prec}(b_{i-1}))^2 = \ldots = \tfrac{1}{2}(2d)^{2^{i-1}}$$

Therefore, for b_{100} = determinant of A, this polynomial will in general be of degree 100 each of whose coefficients is bounded by $\tfrac{1}{2}(2^8)^{2^{99}} = 2^{2^{102}-1}$. Now since each cell will contain a number bounded by 2^{29} (this is the IBM 360 SAC-1 version of BETA), each coefficient requires $2^{2^{102}-29}$ cells. This rather shockingly high precision for the coefficients of b_i is perhaps not too surprising since it would account for the poor performance of the 8 numerical methods which were tried on this problem. It thus became clear that this problem could never be solved unless there were some special relationships which actually caused the above bounds not to be achieved. The inexact, numerical approach was unable to determine such identities and hence the problem was tried using SAC-1 for small values of N. We recall that the elements of the tridiagonal matrix defined in Section 2 have the form

$$a_j = \begin{cases} -d^2y \\ -d^2y + \frac{d}{2} - j \end{cases} \qquad c_j = \begin{cases} -d^2y - \frac{d}{2} + j & 1 \le j \le N/2 \\ -d^2y & N/2+1 \le j \le N \end{cases}$$

and

$$b_j = -(a_j + c_j)$$

where

$$d = N + 1$$

Thus the elements of the solution vector are rational functions in the variable y. Below are the results of the $N = 4x4$ case. We note that the boundary conditions for this application are $v(0) = 1$ and $v(1) = 3$ and we wish to determine the asymptotic behavior as $y \to 0$.

$$\text{determinant } (A_{4x4}) = 60000\, y^2 + 100000\, y + 60000$$

$$X_1 = 100,000\, y^2 + 180,000\, y + 120,000$$

$$X_2 = 120,000\, y^2 + 200,000\, y + 120,000$$

$$X_3 = 140,000\, y^2 + 200,000\, y + 120,000$$

$$X_4 = 160,000\, y^2 + 220,000\, y + 120,000$$

Hence we see that 20,000 is a common factor in both the determinant and the numerators of the solution vector. The solution vector $\overline{X}$ is defined to be $(X_1/\text{det},\ldots,X_4/\text{det})$.

Subsequent trial runs for $n = 6,8,10,$ and 12 revealed that in general the factor $2(2(n+1))^n$ would be a common factor in all determinants and in all of the elements of the numerators of the solution. Also, we note that the degrees of the numerator and denominator of the solution of the n x n case are n/2 rather than n as indicated by our more general bound. For example, further experimentation revealed the following results for $n = 8$ with the common factor removed:

$$\text{determinant } (A_{8x8})/2(18)^8 = 5y^4 + 30y^3 + 112y^2 + 191y + 105$$

$$X_1 = 7y^4 + 55y^3 + 198y^2 + 367y + 210$$

$$X_2 = 8y^4 + 557^3 + 221y^2 + 382y + 210$$

$$X_3 = 9y^4 + 59y^3 + 224y^2 + 382y + 210$$

$$X_4 = 10y^4 + 60y^3 + 224y^2 + 382y + 210$$

$$X_5 = 11y^4 + 60y^3 + 224y^2 + 382y + 210$$

$$X_6 = 12y^4 + 61y^3 + 224y^2 + 382y + 210$$

$$X_7 = 13y^4 + 65y^3 + 227y^2 + 382y + 210$$

$$X_8 = 14y^4 + 74y^3 + 250y^2 + 397y + 210$$

Further results were gathered for the cases $n = 10,12,14,$ and 16. In particular it was desired to determine the solution of

$$\lim_{y \to 0} \bar{X} = \lim_{y \to 0}(X_1/\text{det}, X_2/\text{det}, \ldots, X_n/\text{det})$$

where det is the determinant of A_{nxn} and the limit function is applied to each element of the vector. From the computational results the following figures for the limit were produced.

n x n	$\lim_{y\to 0} \overline{X}$
4	$X_1 = 2 \quad X_2 = 2;\ X_3 = 2\ 1/3,\ X_4 = 2\ 1/3$
6	$X_1 = X_2 = X_3 = 2;\ X_4 = X_5 = X_6 = 2\ 1/15$
8	$X_1 = \ldots = X_4 = 2;\ X_5 = \ldots = X_8 = 2\ 1/105$
10	$X_1 = \ldots = X_5 = 2;\ X_6 = \ldots = X_{10} = 2\ 1/1890$
12	$X_1 = \ldots = X_6 = 2;\ X_7 = \ldots = X_{12} = 2\ 1/10395$
14	$X_1 = \ldots = X_7 = 2;\ X_8 = \ldots = X_{14} = 2\ 1/135,135$
16	$X_1 = \ldots = X_8 = 2;\ X_9 = \ldots = X_{16} = 2\ 1/2,027,025$
20	$X_1 = \ldots = X_{10} = 2;\ X_{11} = \ldots = X_{20} = 2\ 1/654,729,075$
100	$X_1 = \ldots = X_{50} = 2;\ X_{51} = \ldots = X_{100} = 2\ 1/a$

where a is a 79 digit number.

Thus we are led immediately to the conjecture that:

$$\lim_{\substack{y\to 0\\ N\to\infty}} \overline{X} = (2,\ldots,2) .$$

Also below we list the computing times for obtaining these various solutions. After some initial experimentation which led to the removal of large scale coefficient growth it became feasible to solve the problems quite expeditiously.

Computing Times

System's Size n x n	seconds
4	.399
6	.433
8	.782
10	1.148
12	1.531
14	2.296
16	2.762
20	4.743
100	294.611

These times include the time for printing as well as the time needed to calculate the solution.

The final algorithm that was used is given below.

1) [Initialization] $b_1 \leftarrow -(x+1);\ a_1 \leftarrow x;$

2) [Forward elimination] For $j \leftarrow 2,\ldots,N/2$ do

$a_j \leftarrow -(a_{j-1} + b_{j-1})x;$

$b_j \leftarrow (-x + 1 - 2j)b_{j-1};$

3) [Determinant] $d \leftarrow a_{N/2} + b_{N/2};$

4) $x_N \leftarrow 2b_{N/2} + 3a_{N/2}$

$$x_{N-1} \leftarrow \frac{-(2x + (N-1))x_N + 3x(a_{N/2} + b_{N/2})}{-(x + N-1)}$$

5) [Back substitution] For $i \leftarrow 2,\ldots, N-1$ do

$j \leftarrow N + 1 - i$

If $j \leftarrow N/2$ Then

$b \leftarrow -2x + N + 1 - 2j$

$d \leftarrow -x + N + 1 - 2j$

$f \leftarrow -x$

else

$b \leftarrow -2x + 2j - (N + 1)$

$d \leftarrow -x$

$f \leftarrow -x + 2j - (N + 1)$

$$x_{n-i} \leftarrow \frac{bx_{n-i+1} - fx_{n-i+2}}{d}$$

6) [End] Return.

<u>Theorem:</u> Let A be the $N \times N$ tridiagonal matrix, $\bar{x} = (x_1, \ldots, x_N)$ and $v = (v_1, \ldots, v_n)$ described in Equation (3).

Then $\lim_{N \to \infty} \bar{x} = (2, \ldots, 2)$

<u>Proof:</u>

Let det be the determinant of A and $\bar{x}_i$ $1 \leq i \leq n$ the numerators of the solution vector $\bar{x}$ such that

$$\bar{x} = (\bar{x}_1/\det, \ldots, \bar{x}_N/\det)$$

It will be sufficient to examine both the leading coefficient and the constant term of det and $\bar{x}_i$ $1 \leq i \leq N$ to determine the value of the elements of the vector in the limit since each element is a rational function. From the algorithm it follows that the leading terms of a_i and b_i are $(-1)^i ix^i$ and $(-1)^i x^i$ respectively. In step (3) $det = a_{N/2} + b_{N/2}$ which implies that the leading term of det is $(-1)^N(\frac{N}{2} + 1)x^{N/2}$. For the constant term we note that $a_1 = -x$ and the smallest power of $a_i, 1 \leq i \leq N$ is 1. Hence a_i has no constant term. By induction on N, it follows that the constant term of $b_N = (-1)^N \prod_{0 \leq i \leq \frac{N-2}{2}} 2i+1$ and hence this is the constant term of det. Thus, $det(A_{NxN})$ has the form $(-1)^N(N/2 + 1)x^{N/2} + \ldots + (-1)^N \prod_{0 \leq i \leq \frac{N-2}{2}} 2i+1$.

Again by induction on N, it follows that the leading term of $\bar{x}_i = (-1)^i(\frac{3N}{} + i - N + 2)x^{N/2}$ and the constant term is $(-1)^N \prod_{0 \leq j \leq \frac{N-2}{2}} 2(2j + 1)$. Therefore, the $\bar{x}_i$ are polynomials of the form $\bar{x}_i = (-1)^i \frac{N + 2i + 4}{2} x^{N/2} + \ldots + (-1)^N \prod_{0 \leq j \leq \frac{N-2}{2}} 2(2j+1)$

Now if we divide $\bar{x}_i/det$ we can write this as

$$\bar{x}_i/det = 1 + A(x)/B(x) \quad , 1 \leq i \leq N/2$$

$$\bar{x}_i/det = 2 + C(x)/D(x) \quad , N/2 + 1 \leq i \leq N$$

where the degrees of A,B,C and D are N/2 and the constant terms of A(x),B(x) are equal. Also the constant term $C(x) = 0$.

Then, in the limit it follows that $\bar{x}_i/\det = 1 + 1 = 2$ for $1 \leq i \leq N/2$ and $\bar{x}_i/\det = 2$ for $N/2 + 1 \leq i \leq N$.

4. Observations

The symbolic solution of this singular perturbation problem, or more precisely the solution of large linear systems via exact, symbolic techniques places the worst strains on symbol manipulation systems. The difficulties of large space requirements which exist for numerical techniques are multiplied because all elements are now exactly represented. Additionally there are more severe computing time constraints since all arithmetic is being done with exact precision and polynomial entries imply a set of coefficients for each element of the matrix. Thus, the first question that is often asked by potential users is how big can the linear systems be which can be reasonably solved using the modern systems which exist today. This is a difficult question to answer without first qualifying what is meant by "solve". As we have seen in the previous sections it was possible to guess the solution to a 100 x 100 system which initially could neither be stored in the computer nor even have its solution computed if it could be stored. Thus the <u>first</u> conclusion to be drawn is that it is not always necessary to be able to produce the precise answer when it is the analytic structure that is desired. Often insight can be gained by examining just a few more cases than hand calculation permits and in this regard symbol manipulation excels.

A second point is that for special types of linear systems, special algorithms can and should be created. Though it generally takes $O(n^3)$ single precision arithmetic operations to invert a randomly generated linear system, this computing time can be significantly improved for certain well-structured matrices. For example, Toeplitz matrices can be inverted in $O(n^2)$ steps by an algorithm given in [ZOH69], pp. 592. Similarly for other $n \times n$ matrices such as Sylvester and Hilbert matrices there exist faster than $O(n^3)$ algorithms for their solution. Thus, the <u>second</u> point is that designers of linear systems packages can only hope to provide for the most commonly desired operations. Problems with special characteristics will necessitate the writing of new programs which take advantage of these features. Unfortunately, this acts as a deterrent to the use of symbolic techniques as the user must create more than simply a main program.

Turning from the time considerations let us consider the space problem. The SAC-1 Linear Systems package requires really two copies of the input matrix; the matrix, say A, is kept and additionally a representation of A modulo some single precision prime is also needed. Then, if A^{-1} is being computed two matrices are stored, one the tentative solution, A^{-1} and another which is the most recently computed answer. Thus we have a constant, 4, associated with the n^2 elements of each system and these additional copies may well mean the difference between solving a problem or not. One capability which is lacking in SAC-1 and of which I am unaware of in any other symbolic system is a sparse matrix package. In [KNU68], Knuth describes a possible data representation and gives algorithms for sparse matrix transpose and Gaussian elimination. There are a sufficient number of problems of this type, such as the one given in the previous sections, to warrant the development of such a capability. Numerical methods for solving large, sparse systems tend to be "ad hoc" and often depend upon the precise structure of the matrix to be solved. Thus, a general system using exact calculation would be invaluable.

In a recent thesis by McClellan, [McC71], we get some facts about the computational limits of our fastest algorithms. Experimenting on a Univac-1108, he gives the following solution times for solving a system, $Ax = b$, where A is $m \times m$ and b is $m \times 1$. Then, for $m = 10$ we have

Integral Entries	Time(sec)	Univariate Polynomial Entries	Time(sec)
$\lvert a_{ij}\rvert \le 2^2$	1.09	degree = 1	15.04
$\le 2^3$	1.56	2	33.96
$\le 2^4$	3.12	3	64.01
$\le 2^5$	6.58	4	89.76
		5	119.70

and for $m = 5$,

Bivariate Polynomial Entries	Time (sec)
degree = 1	24.79
2	115.95
3	263.75

Thus we see that in practical terms we can only solve systems with integral entries of roughly 1/10 the size of what we might conceivably hope to solve using numerical techniques. Perhaps this fraction seems too intolerable to the scientist with practical problems. But this brings into view a critical misunderstanding by some who try to apply symbolic tools. The application given in this paper makes the point. Initially it was hoped that the infinite precision integer arithmetic of SAC-1 could be used to get more accurate answers as $\epsilon \to 0$. The coefficients grew too fast for inexact arithmetic to work well. From the numerical analyst's point of view, he desired only more digits in hopes of understanding the asymptotic growth rate. But even though the infinite precision integer arithmetic is available, this is a gross use of symbol manipulation. Mainly, it ignores the real reason that this exact arithmetic is provided, namely to preserve the analytic structure of the formulas. In fact it was only by looking at the exact solution of the 4 x 4 and 10 x 10 cases that it led to the insight on how to compute the 100 x 100 solution. In the case of linear systems with polynomial entries our methods grow even slower. Does this make numerical techniques more attractive? In the first place there is probably no "canned" numerical program for this operation so one will have to be written. The rough procedure will be to evaluate at many points, solve each resulting system and then interpolate back to produce an answer. At each stage one may encounter truncation and round-off error. Thus either a more complicated error analysis will have to be done or simply hope that the method works. This makes the symbol manipulation approach all the more valid and necessary and asserts again the need for fast algorithms. So, my <u>third</u> point is that we should not try to compete directly with numerical methods in terms of the size of the problems that can be economically handled, but emphasize more vociferously our strong points, namely our symbolic abilities.

Bibliography

[COL71] Collins, G.E. The SAC-1 System: An Introduction and Survey, Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, ACM, Los Angeles, March 1971.

[DOR70] Dorr, Fred W. The Numerical Solution of Singular Perturbations of Boundary Value Problems, SIAM Journal on Numerical Analysis, Vol. 7, No. 2, June 1970, pp. 281-313.

[DOR71] Dorr, Fred W. An Example of Ill-Conditioning in the Numerical Solution of Singular Perturbation Problems, Mathematics of Computation, Vol. 25, No. 114, April 1971, pp. 271-283.

[HOR70] Horowitz, E. User Notes for the SAC-1 System at Cornell University, Computer Science Department, Ithaca, New York, 1970.

[KNU68] Knuth, D. E. The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Addison Wesley, 1968.

[McC71] McClellan, Michael T. The Exact Solution of Systems of Linear Equations with Polynomial Coefficients, Ph.D. Thesis, University of Wisconsin, September 1971.

[OMA68] O'Malley Jr., R. E. Topics in Singular Perturbations, Advances in Mathematics, Vol. 2, 1968, pp. 365-470.

[WAS65] Wasow, W. Asymptotic Expansions for Ordinary Differential Equations, Interscience, New York, 1965.

[ZOH69] Zohar, Shalhav "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench", JACM, Vol. 16, No. 4, October 1969.