JOURNAL OF COMPUTATIONAL PHYSICS 13, 45–58 (1973)

Use of Symbolic Computation to Generate Evolution Equations and Asymptotic Solutions to Elliptic Equations

ROBERT W. ATHERTON AND GEORGE M. HOMSY

Department of Chemical Engineering,
Stanford University, Stanford, California 94305

Received January 24, 1973

An algorithm for the symbolic computation of wave evolution equations using REDUCE is developed. Its use is demonstrated for the class of long waves at the interface between immiscible fluids. Extensions to other problems are indicated.

INTRODUCTION

In fluid mechanics as well as other areas of applied mathematics the need to obtain analytic solutions to systems of partial differential equations often requires large amounts of symbolic manipulation. Such manipulation is often tedious and unrewarding; in addition there is always the danger of introducing a random error that would invalidate the final result. When these manipulations are of a repetitive nature as with asymptotic expansions, it would seem efficient to program them for machine computation.

There have long existed precise algorithms, consistent with the definition of Knuth [11], to solve analytically certain classes of partial differential equations. An algorithm is a finite set of rules which give a sequence of operations for solving a specific type of problem, and which possesses the characteristics of finiteness definiteness, input, output, and effectiveness. Because these algorithms involve symbolic rather than numeric computation, they have not been easily implemented as machine computations.

Recently, however, new computer languages which primarily manipulate logical expressions rather than perform only numeric operations have appeared. These languages, such as REDUCE and LISP, may be used to implement computationally complex algorithms to achieve analytic solutions to systems of partial differential equations.

In this paper we shall discuss the development and programming of an algorithm in REDUCE to solve a moving boundary problem in fluid mechanics. The solution technique [2] in this case involves expressing the solution to an elliptic equation, in which there exist disparate length scales, in terms of the function describing the moving boundary. The equation describing the moving boundary is then determined by substituting the expression for the interior function into a constraint applied at the surface. This procedure results in an equation for the behavior of the surface. The algorithm thus allows generation of a wave evolution equation for the surface.

1. Problem Statement

The class of physical problems under discussion is that of long waves at the interface of two fluids [9, 13, 18]. Such waves are known to occur under circumstances typified by the absence or relative unimportance of body forces such as gravity which tend to hydrostatically damp such waves. We have chosen to treat one of the simplest of these physical problems, that of two-dimensional flow of a viscous liquid film down an inclined plane. The liquid flow is taken to be very viscous, and the adjoining gas phase is assumed to exert no traction on the fluid. If y is the distance normal to the bounding solid surface, we denote the interfacial position by $y = h(x, t)$ (see Fig. 1). The aim of the calculation is thus to develop an algorithm to generate the evolution equation for $h(x, t)$.

The key to the successful description of the class of waves under discussion is the recognition that for long waves, there exist two length scales for the motions which are widely separated. The first scale is that of the film thickness h, and *if the motion is highly viscous*, it is the correct uniform scale for the motion in the y direction. The second scale is the wave length, λ, and the assumption for the problem in this paper is that $\lambda \gg h$. This relation is an extremely well-justified condition for interfacial waves, since (in the language of linear stability theory) the neutral curve for self-excited waves bifurcates at or near the origin in the wave number, Reynolds number plane. Thus, long waves represent the most dangerous mode of disturbance and are expected to be the ones observed experimentally.

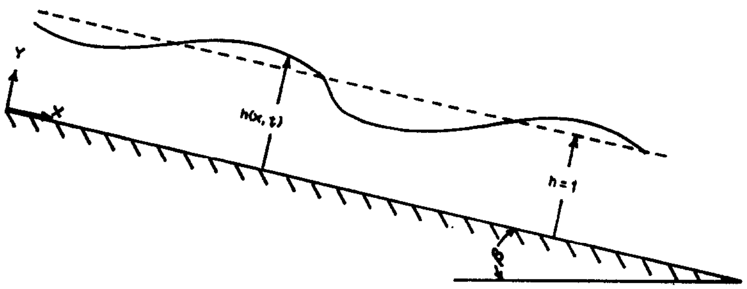

FIG. 1. Geometry of falling film.

For two-dimensional flow the Navier–Stokes equations may be reduced to one equation for the stream function ψ defined such that $(u, v) = (\psi_y, -\psi_x)$. We further introduce a stretching of the x and t dependence so that derivatives of any function with respect to any independent variable are $O(1)$. The scalings for dimensionless variables are given by

$$\begin{aligned}(x, y, t) &= (\alpha x'/h_0, y'/h_0, \alpha t'\bar{u}_0/h_0)\\ (\psi, p, h) &= (\psi' h_0/\bar{u}_0, p' h_0/\mu\bar{u}_0, h'/h_0).\end{aligned}$$

Here primed quantities are dimensional, h_0 is the Nusselt film thickness, $\bar{u}_0$ the average velocity. The quantity α is a dimensionless wave number, i.e., $\alpha = 2\pi h_0/\lambda$, and for long waves becomes a small parameter. The complete nonlinear dynamical equation for $\psi(x, t, y)$ is

$$\begin{aligned}\psi_{yyyy} = {} & \alpha R(\psi_{tyy} + \psi_y\psi_{xyy} - \psi_x\psi_{yyy}) - 2\alpha^2\psi_{xxyy}\\ & + \alpha^3 R(\psi_{txx} - \psi_x\psi_{xxy} + \psi_y\psi_{xxx}) - \alpha^4\psi_{xxxx}\,. \qquad (1.1)\end{aligned}$$

R is the film Reynolds number consistent with this scaling, and in this context is taken as an $O(1)$ parameter.

The equation must be solved subject to the following five boundary conditions [1]: The no slip conditions aply at the wall

$$\psi_x = 0 \quad \text{at} \quad y = 0, \qquad (1.2)$$

$$\psi_y = 0 \quad \text{at} \quad y = 0. \qquad (1.3)$$

The stress conditions apply at the interface.

$$(\psi_{yy} - \alpha^2\psi_{xx})(1 - \alpha^2 h_x^{\,2}) - 4\alpha^2 h_x\psi_{xy} = 0 \quad \text{at} \quad y = h(x, t), \qquad (1.4)$$

$$\begin{aligned}& \psi_{yyy} + 3 + \alpha^2\psi_{xxy} - \alpha R(\psi_{ty} + \psi_y\psi_{xy} - \psi_x\psi_{yy})\\ & \quad - \alpha^2 h_x\psi_{yyx} - 3\alpha h_x \cot\beta - \alpha^4 h_x\psi_{xxx} + \alpha^3 R h_x(\psi_{tx} + \psi_y\psi_{xx} - \psi_x\psi_{yx})\\ & \quad + 2\alpha^2(\psi_{yxx} + \psi_{yyx}h_x) + 8\alpha^4\psi_{xy}(h_x h_{xx} + 2\alpha^2(h_x^{\,3}h_{xx}) + 3\alpha^4(h_x^{\,5}h_{xx}))\\ & \quad + \alpha RP[h_{xxx} + 3\alpha^2(-1/2h_x^{\,2}h_{xxx} - h_{xx}^2 h_x)\\ & \quad + 15\alpha^4(1/8h_{xxx}h_x^{\,4} + 1/2h_{xx}^2 h_x^{\,3}) + 35\alpha^6(-1/16h_x^{\,6}h_{xxx} - 3/8h_x^{\,5}h_{xx}^2)]\\ & \quad + 4\alpha^4 h_x^{\,2}(\psi_{yxx} + \psi_{xyy}h_x)(1 + \alpha^2 h_x^{\,2} + \alpha^4 h_x^{\,4}) + o(\alpha^9) = 0\\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{at} \quad y = h(x, t). \qquad (1.5)\end{aligned}$$

Here $P = \alpha^2\mathrm{We}$, where We is the usual Weber number; $\mathrm{We} = \sigma/h_0\bar{u}_0^{\,2}\rho$. P is a convenient surface tension parameter since it is at most $O(1)$ for liquids with high surface tension such as water and alcohol, but has somewhat smaller values for viscous oils [1]. β is the angle of inclination of the plane (c.f. Fig. 1). Finally we have the kinematic condition at the interface

$$h_t + h_x\psi_y + \psi_x = 0 \quad \text{at} \quad y = h(x, t). \qquad (1.6)$$

The method of solution will develop the stream function ψ as a function of h and its derivatives. Thus, we write $\psi = \psi(h(x, t), y)$, and $\psi(h, h)$ represents ψ evaluated at $y = h(x, t)$. Some analysis shows that the kinematic condition may be written more compactly as below [1].

$$h_t + [\psi(h, h)]_x = 0. \tag{1.7}$$

It is thus in the general form of a conservation law, and expresses the fact that the interface is a material surface.

The five boundary conditions thus provide sufficient conditions for the determination of ψ and the interfacial position $h(x, t)$. Following the technique described in [2], the first four boundary conditions are used to solve for the stream function in terms of the unknown function $h(x, t)$. The last boundary condition serves as a constraint which gives an equation to be solved for $h(x, t)$. Solution of this last equation then completes the solution to the problem.

2. The Long Wave Expansion

In order to solve for ψ, we utilize a regular perturbation technique known as long wave expansions. Benney [4] first proposed the use of long wave expansions in this context, and they have been applied by Gjevik [5, 6] and Krantz and Goren [12] to the present problem, and by Roskes [15] to the related three-dimensional problem.

We expand ψ asymptotically as

$$\psi = \sum_{k=0}^{N} \alpha^k \psi^{(k)}. \tag{2.1}$$

The equations describing each successive $\psi^{(n)}$ are determined by substitution of (2.1) into (1.1–1.5) and grouping like powers of α. The sequence of $\psi^{(n)}$ are then determined by successive integrations. The first two differential systems were given by Krantz and Goren [12]. Because they clarify the treatment of more complicated higher orders, we present them here.

$$\begin{aligned} &\psi^{(0)}_{yyyy} = 0, & \\ &\text{at } y = h(x, t), & \psi^{(0)}_{yyy} + 3 = 0, \\ &\text{at } y = h(x, t), & \psi^{(0)}_{yy} = 0, \\ &\text{at } y = 0, & \psi^{(0)}_{y} = 0, \\ &\text{at } y = 0, & \psi^{(0)}_{x} = 0, \end{aligned} \tag{2.2}$$

with solution

$$\psi^{(0)} = 3/2(y^2 h - 1/3 y^3). \tag{2.3}$$

The differential system of order α is

$$\begin{aligned}
\psi^{(1)}_{yyyy} &= R(\psi^{(0)}_{tyy} + \psi^{(0)}_y \psi^{(0)}_{xyy} - \psi^{(0)}_x \psi^{(0)}_{yyy}), \\
&\text{at } y = h(x,t), \quad \psi^{(1)}_{yyy} - R(\psi^{(0)}_{ty} + \psi^{(0)}_y \psi^{(0)}_{xy} - \psi^{(0)}_x \psi^{(0)}_{xy}) \\
&\qquad\qquad - 3 \cot \beta h_x + PRh_{xxx} = 0 \\
&\text{at } y = h(x,t), \quad \psi^{(1)}_{yy} = 0, \\
&\text{at } y = 0, \quad \psi^{(1)}_y = 0, \\
&\text{at } y = 0, \quad \psi^{(1)}_x = 0,
\end{aligned} \tag{2.4}$$

with solution

$$\begin{aligned}
\psi^{(1)} &= -y^2\{R(3/4h^2h_t + 3/4h^4h_x) + 3/2 \cot\beta\, hh_x - 1/2PRhh_{xxx}\} \\
&\quad + y^3\{1/2 \cot\beta\, h_x - 1/6PRh_{xxx}\} + R\{1/8h_t y^4\} + 3/40Rhh_x y^5.
\end{aligned} \tag{2.5}$$

As might be expected, the complexity of the expressions increases with the order of the approximation. Hand computation beyond $\psi^{(1)}$ has all the pitfalls mentioned in the introduction. Unfortunately, understanding the physics of the problem will require higher order terms [1], and machine computation becomes very appealing.

3. Algorithm for Machine Computation

The first step is to obtain all the necessary equations and boundary conditions to solve for each function. The necessary manipulations described in the previous section are easily performed using the capabilities of REDUCE. Only a very few program instructions are required, and the resulting equations are in a format suitable for further machine manipulation. The sets of equations for functions through $\psi^{(8)}$ have been generated. The flowsheet for the generation of these sets is given in Fig. 2.

Once the perturbation equations have been generated, straightforward integration and substitution would be the usual procedure. Because the computations are to be done via computer; however, careful analysis of the properties of the system is necessary for computational tractability and efficiency.

Inspection of the differential equations generated reveals that each is of the form of a fourth order derivative with respect to y, with a forcing function in terms of the known $\psi^{(n-1)}, \psi^{(n-2)}, \ldots, \psi^{(0)}$.

$$\psi^{(n)}_{yyyy} = f(\psi^{(n-1)}, \psi^{(n-2)}, \ldots, \psi^{(0)}). \tag{3.1}$$

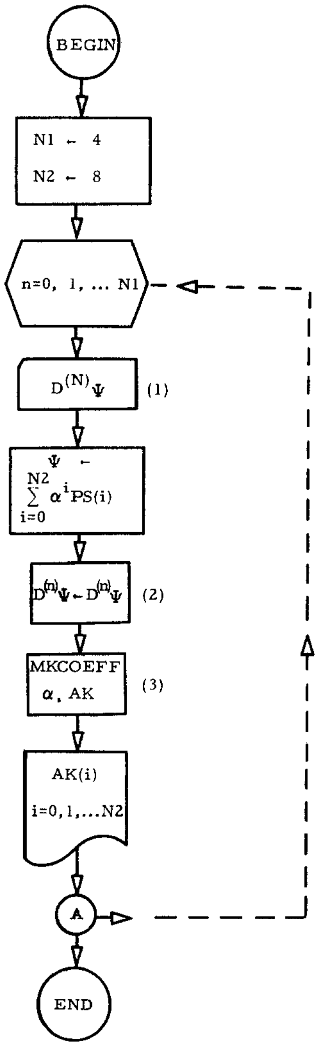

FIG. 2. Flow chart of program for the computation of the perturbation equations. (1) $D^{(n)}\Psi$ is read as normal algebraic expression. For example $D^{(2)}\Psi = (\Psi_{yy} - \alpha^2\Psi_{yy})(1 - \alpha^2 h_y^2) - 4\alpha^2 h_x \Psi_{xy}$. The expression must be given in REDUCE syntax. (2) This statement causes all declared substitutions for expressions in the rhs to be performed. (3) This statement causes the coefficients of the powers of α to be assigned to the array AK.

Experience with $\psi^{(0)}$ and $\psi^{(1)}$ reveals that if the indicated substitutions are performed the differential equations take the following form:

$$\psi^{(n)}_{yyyy} = \sum_{i=0}^{t_n} \bar{g}_i^{(n)} y^i, \qquad n = 0, 1, \ldots, \tag{3.2}$$

$$\{t_n\} = \{0, 1, 5, 9, \ldots, t_{n-1} + 4 \cdots\},$$

where the $\bar{g}_i^{(n)}$ are functions of x and t through explicit dependence on $h(x, t)$ and its derivatives h_x, h_t, h_{xt}, etc.

$$\bar{g}_i^{(n)} = \bar{g}_i^{(n)}(h(x, t), h_t, h_x, h_{xt}, \ldots). \tag{3.3}$$

Inspection of Eqs. (3.2) and (3.3) reveals that a separation of variables has been effected. This makes solution for $\psi^{(n)}$ a fairly straightforward procedure. Equation (3.2) may now be trivially integrated to give

$$\psi^{(n)} = \sum_{i=0}^{t_n} \frac{\bar{g}_i^{(n)} y^{i+4}}{(i+4)(i+3)(i+2)(i+1)} + C_3^{(n)} y^3 + C_2^{(n)} y^2 + C_1^{(n)} y + C_0^{(n)}. \tag{3.4a}$$

The constants $C_j^{(n)}$ are to be determined by the boundary conditions: $D^{(0)}\psi^{(n)}$, $D^{(1)}\psi^{(n)}$, $D^{(2)}\psi^{(n)}$, $D^{(3)}\psi^{(n)}$. Since the stream function is determined to within an additive constant, we define the solid surface to be represented by the streamline $\psi = 0$ and replace the boundary condition $\psi_x^{(n)}(0) = 0$ by

$$D^{(0)}\psi^{(n)} \equiv \psi^{(n)}(0) = 0. \tag{3.5}$$

Thus $C_0^{(n)}$ is zero for all n. A similar general result comes from application of $D^{(1)}\psi^{(n)}$.

$$D^{(1)}\psi^{(n)} \equiv \psi_y^{(n)}(0) = 0. \tag{3.6}$$

Thus $C_1^{(n)}$ is zero for all n. Equation (3.4) may be simplified to

$$\psi^{(n)} = \sum_{i=0}^{t_n} \frac{\bar{g}_i^{(n)} y^{i+4}}{(i+4)(i+3)(i+2)(i+1)} + C_3^{(n)} y^3 + C_2^{(n)} y^2. \tag{3.4b}$$

The boundary conditions at the interface do not yield $C_2^{(n)}$ and $C_3^{(n)}$ so easily. They are of the following form at $y = h(x, t)$

$$D^{(3)}\psi^{(n)} = \psi^{(n)}_{yyy} - f_3(\psi^{(n-1)}, \psi^{(n-2)}, \ldots, \psi^{(0)}) = 0 \tag{3.7}$$

$$D^{(2)}\psi^{(n)} = \psi^{(n)}_{yy} - f_2(\psi^{(n-1)}, \psi^{(n-2)}, \ldots, \psi^{(0)}) = 0 \tag{3.8}$$

First the substitutions are made for the lower order functions which results in an expression similar to the right side of (3.2). Then (3.4b) is substituted into (3.7), and it is solved for $C_3^{(n)}$; $C_3^{(n)}$ and (3.4) are substituted into (3.8), and it is solved for $C_2^{(n)}$.

Our results to this point may be summarized by the following set of equations.

$$\bar{g}_i^{(n)} = \bar{g}_i^{(n)}(g_j^{(n-1)}, g_j^{(n-2)},\ldots, g_j^{(0)}), \qquad i = 0,\ldots, t_n{}^j; \tag{3.9a}$$

$$g_{i+4}^{(n)} = \bar{g}_i^{(n)}/(i+4)(i+3)(i+2)(i+1), \qquad i = 0,\ldots, t_n; \tag{3.9b}$$

$$g_i^{(n)} = C_i^{(n)} = C_i^{(n)}(g_j^{(n-1)},\ldots, g_j^{(0)}, \bar{g}_j^{(n)}), \qquad i = 2, 3; \tag{3.9c}$$

$$\psi^{(n)} = \sum_{i=2}^{t_n} g_i^{(n)} y^i, \qquad \text{where} \quad \tau_n = t_n + 4. \tag{3.9d}$$

It is convenient to express the coefficients of the differential equation $\bar{g}_i^{(n)}$ in terms of the coefficients of the previous functions, $g_i^{(n-j)}$, as in (3.9a). (3.9b, c, d) follow from the previous discussion. The computation of $\psi^{(n)}$ is completed by performing the sequential substitutions (3.9a, b, c, d) to arrive at $\psi^{(n)}(h(x, t), y)$ as desired. The algorithm is summarized in Table I, and the corresponding flowsheet is presented in Fig. 3.

TABLE I
Algorithm for the Computation of Regular Perturbation of the Stream Function to Order α^N

Step 0:	$n \to 0$, Input N
Step 1:	Input $D^{(3)}\psi^{(n)}$, $D^{(2)}\psi^{(n)}$
Step 2:	If $n = 0$, go to 4
Step 3:	$\psi^{(n-j)} \to \sum g_i^{(n)} y^i$ into 3.1, 7, 8 for $j = 1,\ldots, n$
Step 4:	Substitute 3.4b into 3.7, $y \to h$, Solve for $C_3^{(n)}$
Step 5:	Substitute 3.4b, $C_3^{(n)}$ into 3.8, $y \to h$, Solve for $C_2^{(n)}$
Step 6:	Substitute $g_j^{(n-j)}$, $j = 1,\ldots, n$ into 3.9a, b, c
Step 7:	Output $g_j^{(n-j)}$, $j = 0,\ldots, \tau_n$
Step 8:	$n \to n + 1$
Step 9:	If $n \leqslant N$, go to 1; if not, stop.

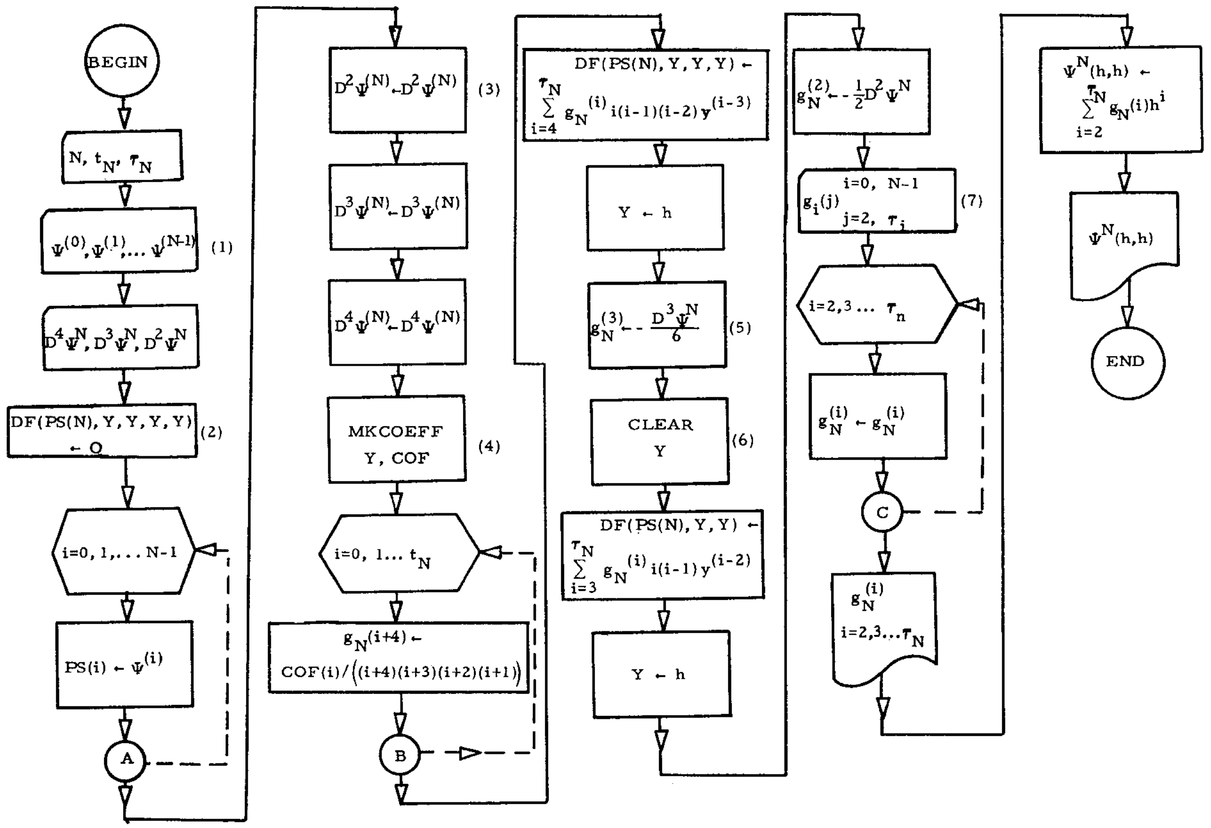

FIG. 3. Flow chart of program for the integration of the expansion functions at any order α^N. (1) The perturbation functions are read as power series in y (see Eq. 3.9d). (2) Ψ_{yyyy} is set equal to zero since no substitutions will be made for it. (3) Statements of this type cause all substitutions defined by preceding statements to be performed in the rhs. (4) This statement assigns the coefficients of the powers of y to the array COF. It operates on the last expression in the work space; this is $D^4\Psi$ in the previous statement. (5) Limitations in the size of available core may require this step to be performed in stages. (6) This statement nullifies all substitutions for y. (7) These coefficients are now given in terms of partial derivatives of h (see Eq. 3.3).

4. Implementation of the Algorithm

The algorithm of the previous section was implemented using programs in REDUCE [3]. REDUCE is a language designed for general algebraic computation. It involves the manipulation of logical expressions, not numbers. Its capabilities include expansion and ordering of rational functions of polynomials, symbolic differentiation, substitution and pattern matching, simplification of expressions, and other capabilities not germane to this work. Complete details on REDUCE are provided by Hearn [7, 8].

The computations were carried out in batch runs using an IBM 360/67. At present REDUCE is imbedded in LISP 1.5, and thus, both the REDUCE translator and LISP compiler must be present for execution of a program. In addition very large amounts of core are required to represent symbolic expressions, especially the complex derivative expressions that are common in this computation. The net result is to restrict the amount of computation that can be done with any one program.

A long sequence of computations, thus, cannot be performed with only one run. Instead computations must be carried out until the expressions generated have filled the available core. The results are written on a disk in a format compatible with REDUCE by very simple commands, and become statements in the program for the next computation. The file handling facilities are very convenient.

Despite the drawbacks presented by the storage problem, REDUCE is a very useful tool for symbolic computation. It is fairly easy to learn and concurrent knowledge of LISP is not essential. Once one become familiar with the language and its limitations, it is fairly easy to carry out long, detailed computations for analytic results.

5. Results

To date functions up through $\psi^{(3)}$ have been computed and appear in print for the first time.

$$\psi^{(2)} = \sum_{i=2}^{9} g_i^{(2)}(h, h_t, h_x, h_{xt}, \ldots)\, y^i$$

$$\psi^{(3)} = \sum_{i=2}^{13} g_i^{(3)}(h, h_t, h_x, h_{xt}, \ldots)\, y^i$$

The expressions for $g_i^{(2)}$ and $g_i^{(3)}$ are available from the authors upon request.

The terms $\psi^{(2)}(h, h)$ and $\psi^{(3)}(h, h)$ necessary for the kinematic condition are given in the Appendix.

Gjevik [6] and Benney [4] have published forms of $\psi^{(2)}(h, h)$ with the time derivatives eliminated; Benney also neglected surface tension. The corresponding forms of our result do not agree with theirs. Neither author gives full details of the development but it is likely that the discrepancies are due to algebraic errors.

This method should be contrasted with a technique of Van Dyke [17] for machine computation of perturbation solutions. Van Dyke uses numeric computation to generate the coefficients and exponents of high order terms from a general expression for such a term. This technique is not applicable to this problem because the x- and t-derivatives in the boundary conditions cause continued expansion of coefficients of y through $\partial g_i^{(n-j)}/\partial x$, $\partial g_i^{(n-j)}/\partial t$ terms in going from expressions like (3.1) to expressions like (3.2). No general expressions for $\psi^{(n)}$ of the type required by Van Dyke may be found, because symbolic differentiation of an unknown expression, generated later, may not be done numerically.

It should also be noted that singular perturbation or matched asymptotic expansions could also be computed in this manner. Sequences of functions would be generated and steps to implement matching would be added. The algorithm would be more complex however, since one would have to ensure that at each level of matching, solutions of proper order had been computed in each region. It is well known that in such problems, the asymptotic expansions are seldom simple power series as was the case here.

6. Generation of Evolution Equations

The next step in solving the flow problem is to substitute the computed expressions for $\psi^{(n)}$ into the kinematic condition.

$$(\partial h/\partial t) + [\psi^{(0)}(h, h) + \alpha\psi^{(1)}(h, h) + \alpha^2\psi^{(2)}(h, h) + \alpha^3\psi^{(3)}(h, h)\cdots]_x + O(\alpha^4) = 0.$$

In so doing we obtain a parabolic evolution equation describing the wavelike behavior of the liquid surface.

There is a great deal of interest in the study of model nonlinear wave equations such as the Korteweg–de Vries equation [16, 19]. An important problem is determining the solutions and behavior of the relevant wave equation. Just as important is the derivation and physical relevance of the model equation.

The method of derivation used in this work [2] has two advantages over the usual method of derivation such as that of Mei [14] and Su and Gardner [16]. The first is theoretical. The evolution equation presented above is valid for any amplitude wave motion and to any wave number consistent with the approximation chosen. The usual methods derive equations restricted in amplitude as well as wave number. This point is not to say, however, that a further simplification of the surface equation will not be required in order to effect a solution.

The second advantage of this method is that it is computationally compact. Only one sequence of functions must be found in the process of deriving the evolution equation. As many as three sequences of functions may be sought in other methods. This compactness is very important considering the difficulty of storing the complex, higher order expressions.

7. Conclusion

The use of symbolic computation languages such as REDUCE or LISP in computing asymptotic expansions and in deriving evolution equations for an important class of long interfacial waves has been successfully demonstrated. The results are attractive on two counts. The evolution equations so derived are compact and are capable of describing long waves of arbitrary amplitude. The algorithm is conceptually simple and may be easily applied to any related problem of interest.

Previous explicit results for viscous film flow down an inclined plane have been extended to fourth order in wave number. Expansions to this order are thought to be capable of providing quantitatively accurate descriptions of wave evolution and equilibration for this problem and the solution to these equations will be the subject of a later publication.

Appendix

$$\begin{aligned}\psi^{(2)}(h, h) = {} & \frac{1}{8} PR^2h^7h_{xxxx} + \frac{1}{20} PR^2h^6h_{xxx}h_x - \frac{3}{10} PR^2h^5h_{txxx} - \frac{5}{24} PR^2h^4h_th_{xxx} \\ & + \frac{1569}{869} R^2h^{10}h_{xx} - \frac{435}{448} R^2h^9h_x^2 + \frac{2027}{869} R^2h^8h_{tx} - \frac{3}{4} R^2h^7h_th_x \\ & + \frac{181}{240} R^2h^6h_{tt} + \frac{9}{8} R^2h^5h_t^2 - \frac{3}{8} R \cot \beta h^7h_{xx} - \frac{3}{20} R \cot \beta h^6h_x^2 \\ & + \frac{9}{10} R \cot \beta h^5h_{tx} + \frac{5}{8} R \cot \beta h^4h_th_x + 3h^4h_{xx} + 7h^3h_x^2\end{aligned}$$

$$
\begin{aligned}
\psi^3(h, h) = &- \frac{83}{168} P^2R^3h^7h_{xxxx}h_{xxx} - \frac{3}{40} P^2R^3h^6h_xh_{xxx}^2 + \frac{68797}{443520} PR^3h^{11}h_{xxxxx} \\
&+ \frac{1171}{40320} PR^3h^{10}h_xh_{xxxx} + \frac{573}{4480} PR^3h^{10}h_{xxx}h_{xx} + \frac{19381}{15120} PR^3h^9h_{txxxx} \\
&+ \frac{147}{320} PR^3h^9h_x^2h_{xxx} + \frac{13}{224} PR^3h^8h_{txxx}h_x + \frac{9659}{6720} PR^3h^8h_{tx}h_{xxx} \\
&+ \frac{8425}{4480} PR^3h^8h_th_{xxxx} + \frac{1529}{2520} PR^3h^7h_{ttxxx} + \frac{103}{112} PR^3h^7h_th_{xxx}h_x \\
&+ \frac{131}{90} PR^3h^6h_th_{txxx} + \frac{181}{720} PR^3h^6h_{tt}h_{xxx} + \frac{3}{8} PR^3h^5h_th_{xxx} \\
&+ \frac{37}{280} PR^2 \cot \beta h^7h_{xx}h_{xxx} + \frac{37}{280} PR^2 \cot \beta h^7h_{xxxx}h_x \\
&+ \frac{9}{20} PR \cot \beta h^6h_x^2h_{xxx} + \frac{3}{5} PRh^5h_{xxxxx} + 3PRh^4h_{xxxx}h_x \\
&+ PRh^4h_{xxx}h_{xx} + \frac{11}{6} PRh^3h_x^2h_{xxx} + \frac{7}{2} PRh^3h_xh_{xx}^2 \\
&- \frac{514407}{732160} R^3h^{14}h_{xxx} + \frac{2189247}{12812800} R^3h^{13}h_xh_{xx} - \frac{38789509}{5913600} R^3h^{12}h_{txx} \\
&- \frac{3424047}{985600} R^3h^{12}h_x^3 - \frac{3032489}{70400} R^3h^{11}h_th_{xx} - \frac{177397}{22400} R^3h^{11}h_{tx}h_x \\
&- \frac{3127987}{40320} R^3h^{10}h_{ttx} + \frac{13283}{3200} R^3h^{10}h_th_x^2 - \frac{515101}{40320} R^3h^9h_{tt}h_x \\
&- \frac{18824}{315} R^3h^9h_{tx}h_t - \frac{24233}{13440} R^3h^8h_{ttt} - \frac{1155461}{40320} R^3h^8h_t^2h_x \\
&- \frac{343}{40} R^3h^7h_th_{tt} - \frac{177}{40} R^3h^6h_t^3 - \frac{114661}{246400} R^2 \cot \beta h^{11}h_{xxx} \\
&- \frac{195497}{16800} R^2 \cot \beta h^{10}h_xh_{xx} - \frac{3479}{1440} R^2 \cot \beta h^9h_{txx} \\
&- \frac{441}{320} R^2 \cot \beta h^9h_x^3 - \frac{1711}{2240} R^2 \cot \beta h^8h_{tx}h_x - \frac{1711}{2240} R^2 \cot \beta h^8h_th_{xx} \\
&- \frac{523}{280} R^2 \cot \beta h^7h_{ttx} + \frac{1689}{560} R^2 \cot \beta h^7h_th_x^2 - \frac{131}{30} R^2 \cot \beta h^6h_th_{tx} \\
&- \frac{91}{120} R^2 \cot \beta h^6h_{tt}h_x - \frac{9}{8} R^2 \cot \beta h^5h_t^2h_x + \frac{169}{280} R^2 \cot^2 \beta h^7h_{xx}h_x \\
&- \frac{27}{40} R \cot^2 \beta h^6h_x^3 - \frac{213}{80} Rh^8h_{xxx} + \frac{2711}{280} Rh_{xx}h_x
\end{aligned}
$$

$$- \frac{137}{20} Rh^6 h_{txx} - 39 Rh^6 h_x{}^3 - \frac{291}{40} Rh^5 h_t h_{xx} - \frac{44}{5} Rh^5 h_{tx} h_x$$

$$- \frac{105}{8} Rh^4 h_t h_x{}^2 - \frac{4}{5} \cot \beta h^5 h_{xxx} - \frac{507}{40} \cot \beta h^4 h_x h_{xx} - 7 \cot \beta h^3 h_x{}^3$$

ACKNOWLEDGMENTS

We would like to thank R. I. Berns of the Stanford Computation Center and Professor A. C. Hearn of the University of Utah for aid with REDUCE, and the School of Engineering and Department of Chemical Engineering, Stanford University, for computer time and financial support of this research.

REFERENCES

1. R. W. ATHERTON, Engineer's Thesis, Stanford University, 1972.
2. R. W. ATHERTON, to be published (1973).
3. R. W. ATHERTON, unpublished programs and results (1971, 1972).
4. D. J. BENNEY, *J. Math. and Phys.* **45** (1966), 150.
5. B. GJEVIK, *Phys. Fluids* **13** (1970), 1918.
6. B. GJEVIK, report, Oslo University, 1971.
7. A. C. HEARN, SHARE XXXIV, Denver, Colorado, March, 1970.
8. A. C. HEARN, "REDUCE 2 User's Manual," Stanford Artificial Intelligence Project, Memo AIM-133, October, 1970.
9. C. E. HICKOX, *Phys. Fluids* **14** (1971), 251.
10. K. JAVDANI MILANI, Ph.D. Thesis, University of California, Berkeley, 1971.
11. D. E. KNUTH, "Fundamental Algorithms, The Art of Computer Programming 1," Addison-Wesley, Reading, Massachusetts, 1968.
12. W. B. KRANTZ AND S. L. GOREN, *I & EC Fund.* **9** (1970), 107.
13. S. P. LIN, *Phys. Fluids* **14** (1971), 263.
14. C. C. MEI, *J. Math. and Phys.* **45** (1966), 266.
15. G. ROSKES, *Phys. Fluids* **13** (1970), 1440.
16. C. H. SU AND C. S. GARDNER, *J. Math. and Phys.* **10** (1969), 536.
17. M. VAN DYKE, *J. Fluid Mech.* **44** (1970), 365.
18. C. S. YIH, *J. Fluid Mech.* **27** (1967), 337.
19. N. J. ZABUSKY, "Nonlinear Partial Differential Equations," (W. F. Ames, Ed.) Academic Press, New York, 1967.