SHORT COMMUNICATIONS

ANALYSIS OF STATIC NON-LINEAR RESPONSE
BY EXPLICIT TIME INTEGRATION

J. T. ODEN AND J. E. KEY
University of Alabama in Huntsville, Huntsville, Alabama, U.S.A.

In general, a system of linear (or non-linear) algebraic equations is solved on an analogue computer by integrating an appropriately defined system of associated first-order differential equations, the steady-state solution of which is the desired solution of algebraic system. This approach usually works very well so long as the associated dynamical system is stable. The idea of using the same approach numerically is also not altogether new. One can, 'analogously', construct a dynamical system associated with a given system of non-linear algebraic equations and solve it numerically by any of a number of explicit time integration schemes. In fact, similar ideas provide the basis for certain 'dynamic relaxation' methods, incremental loading methods and the widely acclaimed methods of invariant imbedding.

For example, to solve a system of \( n \) non-linear equilibrium equations in \( n \) unknowns \( X \) and a 'load parameter' \( p \)

\[
f(X, p) = 0
\]

the incremental loading methods (also called imbedding or continuation methods\(^1,2\)) involve assuming that \( X \) and \( p \) are functions of a parameter \( s \). Then, if \( \ddot{f} = df/ds \), (1) can be replaced by the first-order system (see Reference 2)

\[
\ddot{f} = [\partial f/\partial X] \dddot{X} + [\partial f/\partial p] \dddot{p} = 0
\]

Variants of (2) of the form \( A\dddot{f} + B\dddot{f} + C = 0 \), etc., have been explored by Stricklin and co-workers.\(^3\)

Most practical methods of incremental loading involve integrating (2) or one of its variants numerically. Unfortunately, integration of (2) and its variants lead to implicit integration schemes owing to the fact that the Jacobian matrix \( [\partial f/\partial X] \) at \( s = s \) depends on \( X(s) \). Consequently, a system of linear equations (or in some cases non-linear equations) must be solved at each increment of the integration process.

It would seem that a more direct method of obtaining an appropriate dynamical system associated with (1) is to use simply

\[
\dddot{X} - f(X, p) = 0
\]

(or some variant such as \( \dddot{X} - Cf = 0 \)) by using explicit integrations. Then the entire process is one of only direct substitution, no equation solving of any kind is involved. Of course, such a procedure can only work when (3) and the integration scheme used for its corresponding discrete model are stable.

In this communication, we briefly describe some recent results obtained in applying this technique to a highly non-linear system of equations encountered in our work on finite element studies in finite elasticity. More specifically, we consider a non-linear system of the type

\[
C_{NM} \dddot{u}^M + 2\nu_{\alpha\beta} \left( \left( \delta \alpha \beta - \lambda^4 f^{\alpha\beta} \right) \partial W/\partial I_4 \right) + \left( \lambda^2 \delta \alpha \beta + f^{\alpha\beta} (1 + 2\lambda^4 + 2\lambda^4 \chi_{uv}) \right) \left( \partial W/\partial I_3 \right) \times b_{\alpha\beta} \left( \partial \rho_1 + b_{\rho\beta} u^\rho_1 \right) = P_{N}(t)
\]
All the terms in this equation are defined in Reference 2, p. 305 except $C_{NM}$, which is simply a 'damping' matrix, $C_{NM} = c_D(\psi_N, \psi_M)$. In matrix form, (4) can be written
\[ \dot{u} = -C^{-1}f(u, t) \] (5)

In our calculations, we use a diagonal $C$ and replace (5) by the simple difference scheme
\[ u^{(n+1)} = u^{(n)} - \Delta t\dot{u}^{(n)}; \quad \dot{u}^{(n)} = -C^{-1}f(u^n, n\Delta t) \] (6)

This scheme was used to solve the biaxial strip problem discussed in Reference 2 (for specific geometrical features and material properties see Reference 2, p. 304). A diagonal damping matrix $C = c_D I$ was used, and various degrees of damping were achieved by adjusting the damping coefficient $c_D$. For this example the time step $\Delta t$ was set to 0.00001, and the equations were integrated from $t = 0$ to $t = 0.003$ sec with various coefficients of damping. The velocity of the loaded edge of the strip and the displacement of the loaded edge are shown as functions of time for various damping coefficients in Figures 1(a) and 1(b), respectively. For damping coefficients of 0.0012 lb-sec/in and 0.0003 lb-sec/in the solution converged to the static solution within 3 msec. However, for the increased damping of 0.0005 lb-sec/in and 0.0010 lb-sec/in a longer integration period would be required for convergence. For a damping coefficient of 0.0001 lb-sec/in, the sample problem failed to converge, due to an instability of the numerical integration scheme.
This problem can be overcome by using a higher-order integration scheme or by decreasing the time step size. The explicit time integration approach presented here was developed to test an alternative approach to the incremental loading techniques with or without Newton–Raphson corrections, described in Reference 2. Both the standard incremental loading method, without N–R corrections, and the explicit integration scheme tend to produce accumulative numerical errors. Success of the integration approach described here is strongly dependent on the choice of a proper damping coefficient. For a good choice, the method can be much faster than standard incremental loading procedures. In general, if \( c_n \) is chosen arbitrarily, the method can require an unreasonable amount of solution time or, of course, be unstable. For certain highly non-linear problems, the method does have one significant advantage in that the solution process can be initiated from states in which the initial stiffness \( k_0 \) is zero: e.g. initially flat membranes.

ACKNOWLEDGEMENT

We acknowledge the support of our work under AFOSR Contract F44620–69–C–0124.

REFERENCES


SOME COMMENTS ON THE EQUATION BLOCK SOLVER

M. SVOBODA AND M. SAHM

Daimler-Benz A.G., Stuttgart, West Germany

The very large system of linear equations

\[ KX = Y \]

where the matrix \( K \) is symmetric and positive definite is nowadays frequently solved by block elimination. In this case, the matrix \( K \) is partitioned into square submatrices \( K_{ij} \), called blocks, and then the Cholesky, Crout or the Gauss algorithm applied at submatrix level. Such a solution method is described by Cantin\(^1\) and von Fuchs and Roy.\(^2\)

A similar algorithm was developed by the Daimler-Benz A.G. Company and used for the solution of the equation systems resulting from the finite element method. It was found that this routine took about twice as much IBM 370/155 computer* CPU-time as the previously used version of the Jensen solver.\(^3\)

There are, however, some advantages to block elimination and, therefore, the authors have searched for some acceleration possibilities. A new version of the block elimination algorithm was developed, which is much faster than the first. The principal characteristics of the new programme are as follows.

In the Daimler-Benz finite element programme the matrix is formed by rows (only the non-zero elements and their location information are stored), and stored sequentially on a disc. A

* The average execution times for multiplication, division and addition are 15·2, 23·3 and 2·2 \( \mu \)sec for double-precision words.

Received 12 December 1972
Revised 6 March 1973
new subroutine was written, which forms the submatrices $K_{ij}$ and prepares information for following computations.

During the formulation of $K_{ij}$ the non-zero elements are counted. Only if there are more than $2N^2/3$ non-zero elements—$N$ is the order of the submatrix $K_{ij}$—is $K_{ij}$ stored as a fully populated matrix; actually, it was found that the submatrices $K_{ij}$ are very poorly populated. The mean value of the number of non-zero elements was about $0.15N^2$ for various two- and three-dimensional problems. Such submatrices are stored in a compressed form, i.e. only the non-zero elements and their location information are stored.

A fully stored matrix $K_{ij}$ requires $N^2$ of 8-byte words in double precision; the compressed matrix requires maximally $2N^2/3$ of 8-byte words and twice $2N^2/3$ of 2-byte words of the IBM 370/155 computer. Therefore, the same storage space is needed for the compressed and uncompressed matrix.

New subroutines were written for operating with compressed matrices. These run very quickly because only the operations with non-zero elements are performed. With $N = 48$ a finite element programme for 2,000 elements and 6,000 unknowns runs on the IBM 370/155 computer in a 140K partition. Some CPU-times of this solver are shown in Table I. The object codes were produced by the H-compiler of the IBM 370/155 computer.

Table I. Timing tests

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of blocks per column $n$</th>
<th>No. of blocks per row $m$</th>
<th>No. of equations $N$</th>
<th>Half bandwidth $M$</th>
<th>CPU-time in sec. (equation formulation and solution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-dimensional</td>
<td>16</td>
<td>4</td>
<td>724</td>
<td>112</td>
<td>335</td>
</tr>
<tr>
<td>3-dimensional</td>
<td>37</td>
<td>6</td>
<td>1,744</td>
<td>198</td>
<td>1,755</td>
</tr>
<tr>
<td>3-dimensional</td>
<td>52</td>
<td>6</td>
<td>2,468</td>
<td>218</td>
<td>3,028</td>
</tr>
<tr>
<td>2-dimensional</td>
<td>5</td>
<td>2</td>
<td>220</td>
<td>23</td>
<td>21</td>
</tr>
<tr>
<td>2-dimensional</td>
<td>10</td>
<td>2</td>
<td>440</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>2-dimensional</td>
<td>37</td>
<td>3</td>
<td>1,740</td>
<td>61</td>
<td>313</td>
</tr>
<tr>
<td>2-dimensional</td>
<td>65</td>
<td>3</td>
<td>3,120</td>
<td>81</td>
<td>866</td>
</tr>
</tbody>
</table>

† The formulation of the equation system takes about 5–10 per cent of the solution time.

REFERENCES

NOTE ON A MODIFIED CONJUGATE GRADIENT METHOD

WINIFRED L. WOOD

Department of Mathematics, University of Reading, Reading, England

SUMMARY

Mr. Bruce Irons recently proposed a modification of the Conjugate Gradient Method. This note shows that if the matrix of the original set of equations is symmetric this method will have the same terminal property as the usual Conjugate Gradient Method except for a particular case when it will not converge at all. If the matrix is not symmetric we cannot deduce the usual terminal property.

Received 1 January 1973
Mr. Bruce Irons recently proposed a modification of the Conjugate Gradient Method to solve the set of $n$ equations in $n$ unknowns (all quantities real)

$$K_Y \delta = F$$  \hspace{1cm} (1)

namely the scheme:

At the $k$th step, given $\delta_k, p_{k-1},$

form the residual

$$r_k = F - K_Y \delta_k$$  \hspace{1cm} (2)

and the direction

$$p_k = K^{-1} r_k + \beta_{k-1} p_{k-1}$$  \hspace{1cm} (3)

where $K$ is a symmetric positive definite matrix "not too unlike" $K_Y$, which happens to be convenient to use, and $\beta_{k-1}$ is chosen to make

$$\langle p_{k-1}, K_Y p_k \rangle = 0$$  \hspace{1cm} (4)

that is,

$$\beta_{k-1} = -\frac{\langle p_{k-1}, K_Y K^{-1} r_k \rangle}{\langle p_{k-1}, K_Y p_{k-1} \rangle}$$  \hspace{1cm} (5)

Then

$$\delta_{k+1} = \delta_k + \alpha_k p_k$$  \hspace{1cm} (6)

where

$$\alpha_k = \frac{\langle r_k, p_k \rangle}{\langle p_k, K_Y p_k \rangle}$$  \hspace{1cm} (7)

Since from equations (2) and (6) we have

$$r_{k+1} = r_k - \alpha_k K_Y p_k$$  \hspace{1cm} (8)

the $\alpha_k$ are such as to make

$$\langle p_k, r_{k+1} \rangle = 0$$  \hspace{1cm} (9)

If $K$ is also a symmetric positive definite matrix we can regard the $\alpha_k$ as chosen to minimize the error expression

$$\langle r_{k+1}, K_Y^{-1} r_{k+1} \rangle = \langle r_k, K_Y^{-1} r_k \rangle - 2\alpha_k \langle r_k, p_k \rangle + \alpha_k^2 \langle p_k, K_Y p_k \rangle$$  \hspace{1cm} (10)

which is minimized by $\alpha_k$ as given by equation (7).

Now this algorithm will not converge if $K \equiv K_Y$ because then we have from equation (5)

$$\beta_{k-1} = -\frac{\langle p_{k-1}, r_k \rangle}{\langle p_{k-1}, K_Y p_{k-1} \rangle}$$  \hspace{1cm} (10)

Hence

$$p_k = K^{-1} r_k \text{ } \forall k$$

and from equation (8)

$$r_{k+1} = (1 - \alpha_k) r_k$$

Hence

$$p_{k+1} = (1 - \alpha_k) (1 - \alpha_{k-1}) \ldots (1 - \alpha_0) r_0$$

and all steps are made in the same direction. Thus there is no convergence (unless we have $r_0 = 0$).
If however $K$ is any symmetric positive definite matrix $\neq K_T$ and $K_T$ is also a symmetric matrix we can prove convergence by a method similar to that for the usual Conjugate Gradient Method. Suppose at the start we have

\[
\begin{align*}
p_0 &= 0, \quad \delta_1 = \delta_0, \quad r_1 = r_0 \\
p_1 &= K^{-1}r_1 = s_1, \quad \text{putting } K^{-1}r_i = s_i \\
\delta_2 &= \delta_1 + \alpha_1 p_1 \\
(p_1, r_2) &= 0 \\
r_2 &= r_1 - \alpha_1 K_T p_1 \\
p_2 &= K^{-1}r_2 + \beta_1 p_1 = s_2 + \beta_1 s_1 \\
(p_1, K_T p_2) &= 0
\end{align*}
\]  

Also

\[
\begin{align*}
(r_2, s_2) &= (r_2, p_1) = 0 \\
(K s_2, s_2) &= 0
\end{align*}
\]

Further

\[
\begin{align*}
r_3 &= r_2 - \alpha_2 K_T p_2 \\
(r_3, p_2) &= 0 \\
p_3 &= K^{-1}r_3 + \beta_2 p_2 \\
&= s_3 + \beta_2 s_1 \\
(p_2, K_T p_3) &= 0
\end{align*}
\]

and

\[
(r_3, s_3) = (r_3, p_1) = 0 \quad \text{from equation (13)}
\]

Hence we have the following results:

(i)

\[
(r_3, p_1) = (r_2 - \alpha_2 K_T p_2, p_1) = 0
\]

(ii)

\[
(r_3, s_2) = (r_3, p_2 - \beta_1 p_1) = 0 \quad \text{from equations (12) and (13)}
\]

Also

\[
(r_3, s_1) = (r_3, p_1) = 0 \quad \text{from equation (13)}
\]

and since

\[
(r_3, K^{-1}r_1) = 0 = (r_3, K^{-1}r_2),
\]

we have

\[
(r_3, K^{-1}K_T p_1) = 0 \quad \text{from equation (10a)}
\]

(iii)

\[
(p_1, K_T p_3) = (p_1, K_T K^{-1}r_2) + \beta_2 (p_1, K_T p_2) = 0 \quad \text{from equations (14) and (10a)}
\]

provided $K_T$ is symmetric.
Thus by an induction argument, following the sequence (i), (ii) and (iii), we can show that

\[
\begin{align*}
(r_i, p_j) &= 0, \quad i > j \\
(r_i, s_j) &= 0, \quad i \neq j
\end{align*}
\]

that is,

\[
\begin{align*}
(s_i, Ks_j) &= 0, \quad i \neq j \\
(p_i, K_T p_j) &= 0, \quad i \neq j
\end{align*}
\]  

(15)

and

\[
\begin{align*}
(P_i', K'r_P) &= 0, \quad i \neq j
\end{align*}
\]

Hence \( p_k \) is a linear combination of \( s_1, s_2, \ldots, s_k \) and the \( s_i \) are mutually conjugate with respect to \( K \).

\( r_{n+1} \) is orthogonal to all the \( s_i, \ i = 1, 2, \ldots, n \) and hence is zero.

This result is independent of the choice of \( K \), provided \( K \neq K_T \).

Steps (i) and (ii) as they stand above do not need the assumption that \( K_T \) is symmetric but the induction argument cannot be continued without step (iii) which does.

If \( K \) is not a symmetric matrix we can only get part of the way. We again have

\[
p_3 = s_3 + \beta_3 s_2 + \beta_2 s_1
\]

where \( s_1, s_2, s_3 \) are conjugate with respect to \( K \).

Now

\[
(r_4, s_3) = (r_4, p_3 - \beta_2 p_2)
\]

\[
= -\beta_2 (r_4, p_2) \quad \text{from equation (9)}
\]

\[
= -\beta_2 (r_3 - \alpha_3 K_T p_3, p_2) \quad \text{from equation (8)}
\]

\[
= \beta_2 \alpha_3 (K_T p_3, p_2) \quad \text{from equation (9)}
\]

\[
= 0 \quad \text{from equation (4)}
\]

But

\[
(r_4, s_2) = (r_3 - \alpha_3 K_T p_3, p_2 - \beta_1 p_1)
\]

\[
= \alpha_3 \beta_1 (K_T p_3, p_1) \quad \text{from equations (3) to (9)}
\]

and because

\[
(r_3, p_1) = (Ks_3, s_1) = 0
\]

and

\[
(K_T p_3, p_1) = (K_T s_3 + \beta_2 K_T p_2, p_1) \quad \text{from equation (3)}
\]

\[
= (K_T s_3, p_1) \quad \text{from equation (4)}
\]

(16)

Now we know that

\[
(s_3, K_T p_1) = \frac{1}{\alpha_1} (s_3, r_1 - r_2) \quad \text{from equation (8)}
\]

\[
= 0
\]

because

\[
(s_3, r_1) = (s_3, Ks_1) = 0
\]

and

\[
(s_3, r_2) = (s_3, Ks_2) = 0
\]

But we cannot deduce that \( (K_T s_3, p_1) = 0 \) unless \( K_T \) is symmetric, i.e. we cannot deduce \( (r_4, s_2) = 0 \).
Similarly, we cannot deduce

\[(r_k, s_1) = 0\]

unless \(K_\tau\) is symmetric.

Hence \(r_k\) is not necessarily orthogonal to \(s_1, s_2, \ldots, s_{k-1}, k > 3\) and although the method at the start will appear to behave like the Conjugate Gradient Method we can no longer expect it to have the same terminal property.

REFERENCE


ACCELERATING THE CONVERGENCE OF ELASTIC–PLASTIC STRESS ANALYSIS

E. F. BOYLE

Sulzer Brothers Limited, 8401 Winterthur, Switzerland

AND

A. JENNINGS†

Civil Engineering Department, Queen’s University, Belfast, Northern Ireland

SUMMARY

Some results are given in which a modified Aitken acceleration is applied to elastic–plastic stress analysis.

THE RESIDUAL FORCE METHOD

The residual force method, formerly called the initial stress method, for non-linear structural analysis\(^1,2\) avoids the continual modification and decomposition of the stiffness matrix required if a Newton–Raphson procedure is adopted. Instead, the stiffness matrix associated with the initial unloaded state is used throughout the analysis. For elastic–plastic analysis the use of the initial stiffness matrix after some of the material has yielded results in incorrect displacements which have a general tendency to underestimate the correct values. Residual forces associated with the set of incorrect displacements are calculated and become a right-hand side in a re-solution of the stiffness equations. As the resulting displacements will still be incorrect iteration is continued until convergence is obtained. Whereas convergence may be slow, particularly when the plastic region is extensive, the avoidance of decomposition of the stiffness matrix gives a large saving in the case where the stiffness matrix has a large average bandwidth (which will normally be so when the number of variables is large).

The basic rate-of-convergence of the iterative process is governed primarily by the material characteristics and secondarily by the geometrical properties of the problem and its finite element idealization. Although these may not easily be changed the efficiency of the method may be improved by extrapolation techniques which have the effect of accelerating convergence.

† Reader.

Received 30 November 1972
MODIFIED AITKEN ACCELERATION

Whereas Aitken's $S^2$ process is difficult to use in conjunction with computer methods, recent modifications by Jennings overcome this weakness. Aitken's method can be considered as an extrapolation from three consecutive results on the assumption that the error decays exponentially.

Figure 1. Finite element map for an edge notched specimen with crack length equal to half width. (a) Outer map. (b) Inner map (region A)
Its unreliability for computer use stems from the possibility of a zero or near zero denominator producing either an infinite or gross prediction. Jennings' modification for multivariable problems requires only one division as opposed to one for each variable, and allows the divisor to be chosen so as to avoid the possibility of a zero.

Suppose the vector $x$ defines a converged solution and, at the $k$th iteration, the predicted variables are $x^{(k)}$. The corresponding error $e^{(k)}$ satisfies the equation

$$x^{(k)} = x + e^{(k)} \quad (1)$$

If convergence is governed by the equation

$$e^{(k+1)} = Me^{(k)} \quad (2)$$

where $M$ is an iteration matrix which is independent of $k$ but not necessarily symmetric, then the recommended procedure is to adopt the modified variables

$$\tilde{x} = x^{(k+2)} + s(x^{(k+2)} - x^{(k+1)}) \quad (3)$$

where

$$s = \frac{(x^{(k)} - 2x^{(k+1)} + x^{(k+2)})^T (x^{(k+1)} - x^{(k+2)})}{(x^{(k)} - 2x^{(k+1)} + x^{(k+2)})^T (x^{(k)} - 2x^{(k+1)} + x^{(k+2)})}$$

APPLICATION TO RESIDUAL FORCE ITERATION

Figure 1 shows the finite element map for an edge notched specimen under pure end moment in which 892 constant strain elements were used, and which was loaded in increments beyond the moment to cause unconstrained yield.

Three analyses were carried out, one with normal iteration, one with an over-relaxation parameter of 1.85 and one with modified Aitken acceleration applied as frequently as possible, i.e. every second normal iteration. Not only was the modified Aitken acceleration consistently effective in speeding convergence but decreased the total number of iterations for the whole analysis by a factor in excess of 4.5 (Figure 2). Modified Aitken acceleration has been successfully
applied to many subsequent elastic-plastic analyses. The values of $s$ automatically chosen have tended to lie between 0.5 and 5 for small load levels and to rise to much larger values nearer to general yield. Values of up to 40 have been obtained.

DISCUSSION OF CONVERGENCE

It is unlikely that the iteration process of the residual force method is actually governed by an iteration matrix. However, it is possible that the predicted variables behave very much as though they are so governed. Consider that equation (2) holds over $n$ iterations where $n$ is the number of variables. Compounding and transposing the iteration equations give

$$[e^{(k)} e^{(k+1)} \ldots e^{(k+n)} ]^T M^T = [e^{(k+1)} e^{(k+2)} \ldots e^{(k+n)}]^T$$

which may be solved for $M$ providing that the matrix $[e^{(k)} e^{(k+1)} \ldots e^{(k+n)}]$ is non-singular. Hence it appears likely that any general multivariable iterative process will behave as though the error decay is governed by an iteration matrix if only a limited number of iterations are considered and if certain special cases do not arise, i.e. $e^{(k+1)}$ being proportional to $e^{(k)}$. Whereas this amounts to only a weak theoretical justification of the modified Aitken acceleration in the context of general multivariable iteration processes it does tend to support the practical results obtained.

REFERENCES


A METHOD FOR NUMERICALLY SOLVING SECOND-ORDER NON-HOMOGENEOUS LINEAR DIFFERENTIAL EQUATIONS WITH VARIABLE COEFFICIENTS

D. B. WILSON, H. Y. TSAO AND A. R. SHOUMAN

*College of Engineering, New Mexico State University, Las Cruces, New Mexico, U.S.A.*

INTRODUCTION

The general second-order non-homogeneous linear differential equation with variable coefficients can be represented by equation (1)

$$\frac{d^2 y}{dx^2} + P(x) \frac{dy}{dx} + Q(x) y = R(x)$$

where $P(x)$, $Q(x)$ and $R(x)$ are general functions of $x$. There are many systems of importance and interest to engineers that are modelled by this equation. Many of these model equations are of a form for which there is a method of analytical solution in terms of a specific class of functions, e.g. Legendre, Bessel, Gauss, Mathieu, etc.\(^1\) In addition, there are numerical methods which are applicable to the numerical integration of any differential equations. By far the most frequently

Received 4 December 1972
used numerical integration methods are either the Euler method or the fourth-order Runge-Kutta method. A new method for the numerical integration of the class of equations shown in equation (1) has been developed. It is compared with the fourth-order Runge-Kutta method.

THEORY OF SOLUTION METHOD

Multiplying equation (1) by \( \exp \int P(x) \, dx \) gives

\[
\frac{d}{dx} \left( \exp \int P(x) \, dx \cdot \frac{dy}{dx} \right) + \exp \int P(x) \, dx \cdot Q(x) \, y = \exp \int P(x) \, dx \cdot R(x)
\]  

(2)

Again, multiplying equation (2) by \( \exp \int P(x) \, dx \) and introducing the transform

\[
dz = \exp \left( \int P(x) \, dx \right) \, dx
\]

or

\[
z = \int \exp \left( \int P(x) \, dx \right) \, dx
\]

gives

\[
\frac{d^2 y}{dz^2} + \Phi_0(z) \, y = U_0(z)
\]  

(3)

where \( \Phi_0(z) \) and \( U_0(z) \) are functions of \( z \).

For the homogeneous equation

\[
\frac{d^2 y}{dz^2} + \Phi_0(z) \, y = 0
\]  

(4)

integrating by parts gives

\[
\frac{dy}{dz} = -\Phi_1(z) \, y + \int \Phi_1(z) \, \frac{dy}{dz} \, dz
\]  

(5)

where \( \Phi_1(z) = \int \Phi_0(z) \, dz \). Integrating the second term on the right-hand side of equation (5) by parts again and substituting for \( \frac{d^2 y}{dz^2} \) from equation (4) gives

\[
\frac{dy}{dz} = -\Phi_1(z) \, y + \Phi_4(z) \, \frac{dy}{dz} + \int \Phi_4(z) \, \Phi_0(z) \, y \, dz
\]  

(6)

Repeating this procedure leads to the following equation

\[
\frac{dy}{dz} + \Phi_1(z) - \Phi_3(z) + \Phi_5(z) - \Phi_7(z) + \ldots \frac{C_2}{1 - \Phi_2(z) + \Phi_4(z) - \Phi_6(z) + \ldots} \, y = \frac{C_2}{1 - \Phi_2(z) + \Phi_4(z) - \Phi_6(z) + \ldots}
\]  

(7)

where \( C_2 \) = a constant and where the first term is

\[
\Phi_1(z) = \int \Phi_0(z) \, dz
\]

and for \( j \geq 1 \)

\[
\Phi_{2j}(z) = \int \Phi_{2j-1}(z) \, dz
\]

\[
\Phi_{2j+1}(z) = \int \Phi_0(z) \, \Phi_{2j}(z) \, dz
\]

Equation (7) can be integrated using the integrating factor

\[
\exp \left[ \int \frac{\Phi_1(z) - \Phi_3(z) + \Phi_5(z) - \ldots \, dz}{1 - \Phi_2(z) + \Phi_4(z) - \Phi_6(z) + \ldots} \right] = \frac{1}{1 - \Phi_2(z) + \Phi_4(z) - \Phi_6(z) + \ldots}
\]  

(8)
Thus the solution to equation (3) can be written as
\[ y = C_1 S_1(z) + C_2 S_2(z) \] (9)
with \( C_1 \) and \( C_2 \) as arbitrary constants. From equation (9)
\[ S_1(z) = 1 - \Phi_4(z) + \Phi_6(z) - \ldots = 1 + \sum_{j=1}^{\infty} (-1)^j \Phi_{2j}(z) \] (10)
and
\[ S_2(z) = S_1(z) \int \frac{dz}{S_1^2(z)} \] (11)

It should be mentioned that equation (11) gives \( S_2(z) \) in general regardless of whether \( S_1(z) \)
is obtained from equation (10) or whether \( S_1(z) \) is any known solution of the original equation.
Further, from equation (11)
\[ \frac{S_2(z)}{S_1(z)} = \int \frac{dz}{S_1^2(z)} \] (11a)
and differentiating with respect to \( z \) gives
\[ \frac{S_2'(z)}{S_1'(z)} - \frac{S_2(z)S_1'(z)}{S_1^2(z)} = \frac{1}{S_1^2(z)} \] (12)
This reduces to
\[ S_2'(z)S_1(z) - S_2(z)S_1'(z) = 1 \] (13)
which is the Wronskian relationship, and proves that \( S_1(z) \) and \( S_2(z) \) are solutions.

It can be shown, using this procedure, that the particular integral for equation (3) is the following
\[ S_3(z) = S_1(z) \int \left[ \sum_{j=0}^{\infty} (-1)^j U_{2j+1}(z) \right] dz / S_1^2(z) \] (14)
where
\[ U_1(z) = \int U_0(z) dz \]
and for \( j \geq 1 \)
\[ U_{2j+1}(z) = \int U_0(z) \Phi_{2j}(z) dz \]
The general solution to equation (3) is
\[ y = C_1 S_1(z) + C_2 S_2(z) + S_3(z) \] (15)
The evaluation of \( C_1 \) and \( C_2 \) in equation (15) depend upon the nature of the initial or boundary
conditions for the given problem. The derivation of four cases for the evaluation of \( C_1 \) and \( C_2 \)
are given in Reference 4.

Program description and results
For the general non-homogeneous problem the computational algorithm evaluates equation (15). The definitions for \( S_1(z), S_2(z) \) and \( S_3(z) \), as given in equations (10), (11) and (14), involve a
nested set of integral functions. Each integral is evaluated by a modified Euler's method. Each
evaluation of the infinite series in \( S_1(z) \) and \( S_2(z) \) is terminated after a finite number of terms.
This truncation error can be estimated by doubling the number of terms evaluated. Any error
accumulated from the computer round-off can be estimated in the standard manner using double
precision calculation. The computational formulas used and the complete computer program
listing are given in Reference 4.
The solution method transforms the independent variable from \( x \) to \( z \). In that transformation two explicit functions \( \Phi_0(z) \) and \( U_0(z) \) are obtained. These functions are computed as statement functions in the computer program and must be changed for each new problem. As shown in equations (10), (11) and (14) values of \( S_1(z) \), \( S_2(z) \) and \( S_3(z) \) can be computed independently of an initial value of \( y \). The initial values or boundary values appear only in the evaluation of \( C_1 \) and \( C_2 \). This numerical algorithm is thus independent of a starting value for the function. Further \( S_1(z) \), \( S_2(z) \) and \( S_3(z) \) do not need to be re-evaluated for a change of boundary or initial conditions when the interval of interest in \( x \) has not been changed.

One final detail of the computational algorithm involves the value of \( S_1(z) \). \( S_1(z) \) appears in the denominator of both functions \( S_2(z) \) and \( S_3(z) \). Since \( S_1(z) \) is an alternating series of \( \pm \) terms it may have a value of zero. At such a point, \( S_1(z) \) and \( S_2(z) \) are undetermined. For such a problem, i.e. a singularity, the usual procedure is to try a variable transformation; however, with this new algorithm it is only required to reverse the direction of computation across the interval of interest, e.g. integrate from \( b \) to \( a \) rather than from \( a \) to \( b \).

The following example problem is demonstrated and the computation is compared to the fourth-order Runge–Kutta method.

Non-homogeneous type model equation, initial-value problem

\[
\frac{d^2y}{dx^2} + \frac{2}{x} \frac{dy}{dx} + \frac{2}{x^2} y = 4x
\]  

(16)

<table>
<thead>
<tr>
<th>( x ) value</th>
<th>Analytical solution</th>
<th>This program*</th>
<th>Computed from</th>
<th>Computed from</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9999819</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>1.1051840</td>
<td>1.013391</td>
<td>1.013391</td>
<td>1.013336</td>
<td>1.013336</td>
</tr>
<tr>
<td>1.1934490</td>
<td>1.051901</td>
<td>1.05191</td>
<td>1.051811</td>
<td>1.051811</td>
</tr>
<tr>
<td>1.2985730</td>
<td>1.142379</td>
<td>1.14242</td>
<td>1.142269</td>
<td>1.142269</td>
</tr>
<tr>
<td>1.4009630</td>
<td>1.289700</td>
<td>1.28979</td>
<td>1.289590</td>
<td>1.289590</td>
</tr>
<tr>
<td>1.5006710</td>
<td>1.501679</td>
<td>1.50188</td>
<td>1.501598</td>
<td>1.501598</td>
</tr>
<tr>
<td>1.6069180</td>
<td>1.815465</td>
<td>1.81582</td>
<td>1.815441</td>
<td>1.815441</td>
</tr>
<tr>
<td>1.7007110</td>
<td>2.179088</td>
<td>2.17969</td>
<td>2.179148</td>
<td>2.179148</td>
</tr>
<tr>
<td>1.7996850</td>
<td>2.662286</td>
<td>2.66340</td>
<td>2.662477</td>
<td>2.662477</td>
</tr>
<tr>
<td>1.9018130</td>
<td>3.280095</td>
<td>3.28336</td>
<td>3.280478</td>
<td>3.280478</td>
</tr>
<tr>
<td>1.9998610</td>
<td>3.998888</td>
<td>(0.185676)†</td>
<td>3.999510</td>
<td>0.0010104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(−0.001006)†</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The transformed equation is

\[
\frac{d^2y}{dz^2} + \frac{2}{9z^2} y = \frac{4}{3z}
\]

Therefore

\[
\Phi_0(z) = \frac{2}{9z^2} \quad \text{and} \quad U_0(z) = \frac{4}{3z}
\]

and

\[
C_1 = y_0, \quad C_2 = \left( \frac{dy}{dx} \right)_{x_0} f(x_0)
\]

† Values in parentheses are \( S_2(z) \) showing that it has a zero value. The last good value of \( y \) was at \( x = 1.982 \).
with initial conditions

\[ \text{at } x_0 = 1, \ y_0 = 1 \]

and

\[ \text{at } x_0 = 1, \ f(x_0) = \frac{dy}{dz} = 0 \]

The results are shown in Table I.

The values shown in Table I were computed using ten terms in the series for \( S_1(z) \) and \( S_2(z) \). In addition, Table I shows that the algorithm can easily handle values of \( S_1(z) \) going to zero by reversing the direction of computation. Tables II and III show other characteristics of this algorithm.

### Table II. Comparison of execution times

<table>
<thead>
<tr>
<th>This algorithm</th>
<th>Execution time</th>
<th>Compile time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fourth-order Runge–Kutta</td>
<td>1.27 sec</td>
<td>1.67 sec</td>
</tr>
<tr>
<td>5 terms in series</td>
<td>1.74 sec</td>
<td>7.17 sec</td>
</tr>
<tr>
<td>10 terms in series</td>
<td>1.95 sec</td>
<td>6.94 sec</td>
</tr>
<tr>
<td>15 terms in series</td>
<td>2.02 sec</td>
<td>6.82 sec</td>
</tr>
<tr>
<td>20 terms in series</td>
<td>2.80 sec</td>
<td>7.47 sec</td>
</tr>
</tbody>
</table>

### Table III. Analysis of truncation error

<table>
<thead>
<tr>
<th>Number of terms in series</th>
<th>( x = 1.1 )</th>
<th>( x = 1.5 )</th>
<th>( x = 2.0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>1.013337</td>
<td>1.501589</td>
<td>3.999475</td>
</tr>
<tr>
<td>15</td>
<td>1.013337</td>
<td>1.501589</td>
<td>3.999475</td>
</tr>
<tr>
<td>10</td>
<td>1.013337</td>
<td>1.501589</td>
<td>3.999375</td>
</tr>
<tr>
<td>5</td>
<td>1.021926</td>
<td>1.500860</td>
<td>3.998927</td>
</tr>
<tr>
<td>Analytical value</td>
<td>1.013300*</td>
<td>1.500000</td>
<td>4.000000</td>
</tr>
</tbody>
</table>

* Rounded-off after calculating by interpolation.

This program has comparable accuracy (using at least ten terms in the series) to the fourth-order Runge–Kutta method. It is slower than the fourth-order Runge-Kutta as shown in Table II; however, if additional sets of initial conditions are to be evaluated, or if the problem is a boundary-value problem, this algorithm becomes comparable to the fourth-order Runge–Kutta in total computer time while maintaining the same accuracy.

One final property of this method should be noted, and that is the algorithm would work even if an analytical transformation was not available for \( z(x) \). This has not been demonstrated in this paper.

It has been shown that the utility of the present solution program for solving second-order non-homogeneous linear differential equations with variable coefficients makes it an interesting addition to numerical computation.

### REFERENCES

ANNOUNCEMENT

AN INVITATION TO PARTICIPATE IN THE INTERNATIONAL SYMPOSIUM
ON STRUCTURAL MECHANICS SOFTWARE

An International Symposium on Structural Mechanics Software, sponsored by the Office of Naval Research, will be held on 12–14 June 1974 at the University of Maryland, College Park, Maryland.

The Symposium will focus on the current state-of-the-art and future areas of research in structural mechanics software. Several key areas of software development and data management will be considered. Session topics will include the following:

- Current Systems for Structural Software Dissemination.
- Remote System Software Sharing: Interactive and Batch Modes.
- Unified Data Bases and Modular Software Concepts.
- Pre-processors and Post-processors for Remotely Accessible Batch Programs.
- Mini-computer and Network Systems.
- Requirements for Future Developments in Software Sharing.
- Survey of Current Analysis and Design Structures Software.

A group of remotely accessible user-oriented programs will be assembled as a part of the Symposium and will serve both as an active example of the current state-of-the-art in several key areas of structural mechanics and as a possible means of making software more easily available to the technical community. Modified versions of the programs will be accessible by Symposium participants by means of teletype units and user reactions will be invited during the Symposium.

The Symposium solicits structural mechanics software for inclusion in the Symposium in the general areas of non-linear analysis, dynamic analysis, stability, plasticity, viscoelasticity and general shock and vibration. Programs submitted should address an analysis or design problem of current interest to the user community. Timesharing or batch mode programs, including special purpose, general purpose and pre- and post-processors for large structural programs such as NASTRAN, will be considered. To be eligible for consideration, 1,000 word abstracts of programs must be received by 1 October 1973. Abstracts will be reviewed for topic suitability; programs with supporting documentation will be required by 1 December 1973. Notice of final acceptance will be made by 1 March 1974.

For further information, prospective contributors are invited to contact either:

- Dr. H. G. Schaeffer
  Department of Aerospace Engineering
  University of Maryland
  College Park, Maryland 20742

  or

- Dr. Walter Pilkey
  Department of Engineering Science and Systems
  University of Virginia
  Charlottesville, Virginia 22901