JOURNAL OF COMPUTATIONAL PHYSICS 39, 418–432 (1981)

A Numerical Method for the Calculation of Traveling Wave Solutions of a Quench Front Problem*,†

J. E. DENDY, JR., AND BURTON WENDROFF

Theoretical Division, University of California,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico 87545

Received February 15, 1980; revised June 3, 1980

A numerical method is described for the solution of the following quench front problem: Find $u(x, y)$ and v such that

$$\frac{\partial}{\partial x} k(u) \frac{\partial u}{\partial x} + vq(u) \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} k(u) \frac{\partial u}{\partial y} = 0,$$

$$\left. \frac{\partial u}{\partial y} \right|_{y=0} = f(u),$$

$$\left. \frac{\partial u}{\partial y} \right|_{y=1} = 0,$$

$$u(-\infty, y) = 0,$$

$$u(+\infty, y) = 1.$$

The method is based on the idea of isotherm migration. The resulting problem is an eigenvalue problem for a system of nonlinear Cauchy–Riemann equations. The method is very efficient in comparison with previous methods for this problem.

1. INTRODUCTION

Emergency core cooling systems for water-cooled reactors bring about a cooling of the hot reactor core by flooding with water from the bottom or spraying with water from the top. The surface to be cooled will be at a temperature well above that of boiling water. At this temperature water cannot stay in contact with the surface; a film of steam forms which insulates the surface from further cooling. The surface does cool down, however; a front forms which moves into the hot region. Behind the front the surface is cool and in contact with the water; ahead of the front it is hot and dry. This front is called a quench front or rewetting front. The mechanism for the

* This work was supported by the United States Department of Energy and the United States Nuclear Regulatory Commission.

0021-9991/81/020418–15$02.00/0

propagation of this front is the flow of heat along the temperature gradient on the surface.

The mathematical model [8] of this process, in the case of a liquid film rewetting a hot plate, is, after nondimensionalization, the following initial boundary value problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \qquad u(x, y, 0) \text{ given.}$$

The boundary conditions are

$$\left.\frac{\partial u}{\partial y}\right|_{y=1} = 0$$

at the insulated boundary, $y = 1$, and

$$\left.\frac{\partial u}{\partial y}\right|_{y=0} = f(u)$$

at the conducting boundary, $y = 0$. The function $f(u)$ is to be specified. In addition

$$u(-\infty, y, t) = 0,$$
$$u(+\infty, y, t) = 1.$$

What are of interest in this problem are plane traveling wave solutions, solutions of the form

$$u = u(x - vt, y).$$

If we set $x' = x - vt$ and look for steady state solutions, we obtain

$$\begin{aligned} u &= u(x, y), \\ \frac{\partial^2 u}{\partial x^2} + v\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial y^2} &= 0, \\ \left.\frac{\partial u}{\partial y}\right|_{y=0} &= f(u), \\ \left.\frac{\partial u}{\partial y}\right|_{y=1} &= 0, \\ u(-\infty, y) &= 0, \\ u(+\infty, y) &= 1, \end{aligned} \tag{1.1}$$

where we have written x instead of x' for simplicity.

As in [4] a one-dimensional approximation can be obtained by integrating (1.1) from $y = 0$ to $y = 1$, obtaining

$$\begin{aligned} u &= u(x), \\ \frac{\partial^2 u}{\partial x^2} + v\frac{\partial u}{\partial x} &= -f(u), \\ u(-\infty) &= 0, \\ u(+\infty) &= 1. \end{aligned} \tag{1.2}$$

The numerical method described in this paper is designed to solve numerically both (1.1) and (1.2). A computer code implementing the method is described in [2]. This code has the name QUENCH, and for ease of reference we refer to the numerical method itself with this name. The method is extremely efficient. For example, one problem which required 15 min of CDC 7600 time with the method in [5] requires only 1 sec with QUENCH for equivalent accuracy.

2. The Numerical Method

The numerical method uses the idea of isotherm migration [5, 3], which is to change variables in (1.1), writing $w(u, y) = (\partial u/\partial x)(x(u, y), y)$ and $z(u, y) = (\partial u/\partial y)(x(u, y), y)$. Then (1.1) transforms to

$$w\frac{\partial w}{\partial u} + vw + z\frac{\partial z}{\partial u} + \frac{\partial z}{\partial y} = 0. \tag{2.1a}$$

A second equation can be obtained from the condition that $\partial^2 u/\partial x\,\partial y = \partial^2 u/\partial y\,\partial x$ which holds in the interior; this yields

$$z\frac{\partial w}{\partial u} - w\frac{\partial z}{\partial u} + \frac{\partial w}{\partial y} = 0. \tag{2.1b}$$

The boundary conditions transform to

$$\begin{aligned} z(u, 1) &= 0, \\ z(u, 0) &= f(u), \end{aligned}$$

and under the assumption that $(\partial u/\partial x)(-\infty, y) = 0 = (\partial u/\partial x)(+\infty, y)$ we also have

$$\begin{aligned} w(0, y) &= 0, \\ w(1, y) &= 0. \end{aligned}$$

The motivation behind this change of variables is threefold. First, the infinite region is traded for a finite region. Second, the nonlinear boundary condition is traded for a linear boundary condition. Third, dealing with u as an independent variable instead of x provides the possibility of using a much smaller number of mesh points for equivalent accuracy. The price for these simplifications is greatly increased complexity in the differential equation.

Before proceeding with the description of QUENCH, we wish to describe how this use of isotherm migration differs from that in [5]. In [5] the following equations were derived

$$-x_t = \{w[1 + (x_y)^2]_u\} - x_{yy}. \tag{2.2}$$

The relationship

$$-u_y(x, y, t) = \frac{x_y(u, y, t)}{x_u(u, y, t)}$$

was exploited to alow (2.2) to be written in terms solely of derivatives of x. In [5], (2.2) was solved to steady state by an explicit method in time; at steady state $x_t = v$. This partially explains the inefficiency of [5]. One could presumably set $x_t = v$ in (2.2) and solve the resulting equation efficiently. We prefer the equations (2.1), however, because they are written directly in terms of w and z. The boundary conditions for w and z are natural; such was not the case for the boundary conditions for x in [5]. The resemblance of (2.1) to the Cauchy–Riemann equations also seems attractive.

Since (2.1) resembles the Cauchy–Riemann equations it seems reasonable to approximate it by writing difference equations on a staggered grid. The grid is shown in Fig. 1. Note that it has a border of fictitious cells. A typical cell with the location of the W's and Z's is shown in Fig. 2. At the left and right boundaries, this storage is modified somewhat. For example, at the left boundary, it is as is shown in Fig. 3. For the purposes of exposition we assume to begin with a uniform mesh with mesh spacing Δu and Δy. A difference scheme approximating (2.1a) on the (i, j)th cell is

$$\begin{aligned}\frac{1}{2}(W_{i,j} + W_{i-1,j})(W_{i,j} - W_{i-1,j})/\Delta u + V\frac{1}{2}(W_{i,j} + W_{i-1,j}) \\ + \frac{1}{\Delta y}((\theta^r_{i,j}(Z)Z_{i,j} + (1 - \theta^r_{i,j}(Z))Z_{i+1,j}) - (\theta^l_{i,j-1}(Z)Z_{i,j-1} \\ + (1 - \theta_{i-1,j-1}(Z))Z_{i-1,j-1})) = 0,\end{aligned} \tag{2.3a}$$

where

$$\begin{aligned}\theta^r_{i,j}(Z) &= 1.-.125\Delta y(Z_{i,j} + Z_{i,j-1} + Z_{i+1,j} + Z_{i+1,j+1})/\Delta u \quad \text{if } j < J1 \\ &= 1.-.25\Delta y(Z_{i,j} + Z_{i,j-1})/\Delta u \quad \text{if } j = J1\end{aligned} \tag{2.4a}$$

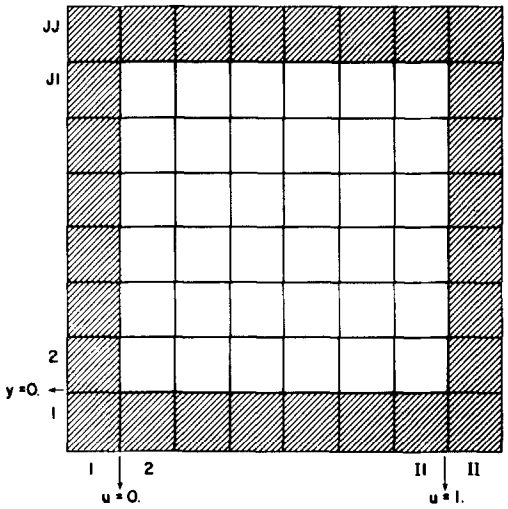

FIGURE 1

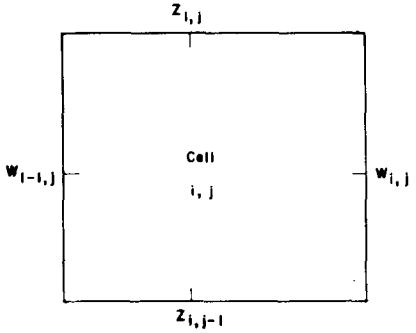

FIGURE 2

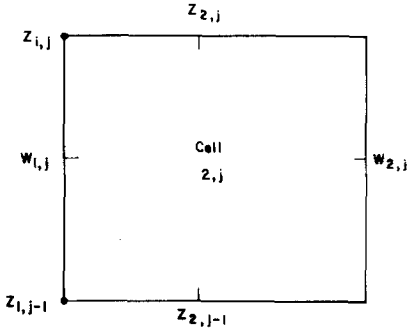

FIGURE 3

and

$$\begin{aligned}\theta^I_{i,j}(Z) &= 1.-.125\Delta y(Z_{i,j}+Z_{i,j-1}+Z_{i-1,j}+Z_{i-1,j-1})/\Delta u \qquad \text{if } j>2\\ &= 1.-.25\Delta y(Z_{i,j}+Z_{i,j-1})/\Delta u \qquad \text{if } j=2.\end{aligned} \tag{2.4b}$$

The differencing of $w(\partial w/\partial u)+vw$ is obvious. The differencing of $z(\partial z/\partial u)+(\partial z/\partial y)$ is less obvious and is explained below. The second equation (2.1b) is approximated by a difference equation at the interior cell corners:

$$\frac{1}{2}(Z_{i+1,j}+Z_{i,j})\frac{1}{4\Delta u}(W_{i+1,j+1}+W_{i+1,j}-W_{i-1,j+1}-W_{i-1,j})$$

$$-\frac{1}{2}(W_{i,j}+W_{i,j+1})\frac{1}{\Delta u}(Z_{i+1,j}-Z_{i,j})+\frac{1}{\Delta y}(W_{i,j+1}-W_{i,j})=0. \tag{2.3b}$$

The boundary conditions are

$$\begin{aligned}Z_{i,J1} &= 0, && 2\leqslant i\leqslant I1,\\ Z_{i,1} &= f(\Delta u(i-\tfrac{3}{2})), && 2\leqslant i\leqslant I1,\\ W_{1,j} &= 0 = W_{I1,j}, && 2\leqslant j\leqslant J1.\end{aligned}$$

It is straighforward to check that the number of equations at cell centers and interior cell corners is the same as the number of unknowns, including V. In this regard it should be noted that Z is zero along the lines $u=0$ and $u=1$ since $z=\partial u/\partial y$.

Let us give first a brief description of the way (2.3) is solved by QUENCH. The method always begins by attempting to solve the following one-dimensional difference equation approximating (1.2):

$$\begin{gathered}\tfrac{1}{2}(W_i+W_{i-1})(W_i-W_{i-1})/\Delta u+V\tfrac{1}{2}(W_i+W_{i-1})=-f((i-\tfrac{1}{2})\Delta u),\\ W_1=0=W_{I1}.\end{gathered} \tag{2.5}$$

If (2.5) is solved successfully, then one chooses a Δy according to a criterion described below, scales W and V by $W\leftarrow W/(\Delta y)^{1/2}$, $V\leftarrow V/(\Delta y)^{1/2}$ and then attempts to solve (2.3a) with only one y interval. The method then starts adding on y intervals and solves a sequence of two-dimensional problems. The method stops under one of four conditions: (i) too many iterations have been performed, (ii) the V obtained on the current mesh is "close" to the V obtained on the last mesh, (iii) storage requirements have been exceeded, or (iv) the current mesh has enough y intervals to reach $y=1$.

We now discuss the above in more detail. For the one-dimensional problem (1.2), both (1.2) and (2.5) can be solved exactly if

$$\begin{aligned}f(u) &= Bu, && 0\leqslant u\leqslant u_c,\\ &= 0, && u_c<u\leqslant 1;\end{aligned} \tag{2.6}$$

$v > 0$ is given by

$$v^2 = B\frac{u_c^2}{1-u_c} \tag{2.7}$$

and w is given by

$$\begin{aligned} w(u) &= \frac{v(1-u_c)}{u_c}u \qquad \text{if} \quad 0 \leqslant u \leqslant u_c \\ &= v(1-u) \qquad\quad \text{if} \quad u_c < u \leqslant 1. \end{aligned} \tag{2.8}$$

In its simplest mode, QUENCH assumes f to be piecewise linear and the last piece to be identically zero; the latter can be shown to be sufficient for existence of a unique solution of (1.2). One specifies f by specifying the number of breakpoints $NBKP$ of f, counting 0 and 1; the breakpoints $BKP(I)$, $I = 1,..., NBKP$ of f; the slopes $B(I)$ of f, $I = 1,..., NBKP1 = NBKP - 1$; and $FLP(I) = f(BKP(I^+))$, $I = 1,..., NBKP1$, i.e., the limits of f as $BKP(I)$ is approached from the right. One also specifies V and $NIND$, which specifies u_c in (2.8) to be $BKP(NIND)$. The method then guesses W according to (2.8). As noted earlier if f satisfies (2.6) and V satisfies (2.7), then W is the exact answer. For more general piecewise linear f's, one can only hope that this procedure provides a good guess for W and V; at least for the test problems we have tried it always has. For f's more general than piecewise linear, one needs to use enough intervals that f is well approximated by a piecewise linear function on those intervals.

The importance of the initial guess, of course, is that (2.5) is to be solved by Newton's method. If we linearize (2.5), then given W and V we need to solve for WP and VP given by

$$\begin{aligned} &\tfrac{1}{2}(WP_i + WP_{i-1})(W_i - W_{i-1})/\Delta u + \tfrac{1}{2}(W_i + W_{i-1})(WP_i - WP_{i-1})/\Delta u \\ &\quad + VP\tfrac{1}{2}(W_i + W_{i-1}) + V\tfrac{1}{2}(WP_i + WP_{i-1}) \\ &= \tfrac{1}{2}(W_i + W_{i-1})(W_i - W_{i-1})/\Delta u + \tfrac{1}{2}V(W_i + W_{i-1}). \end{aligned}$$

This system can be solved directly for $(WP_2,..., WP_{I1-1}, VP)$; in fact, the solution is efficient since the matrix is a lower triangular bidiagonal matrix except for the last column.

For the model f given by (2.6) it has been observed that the larger B is the more the solution of (2.3) tends to be a boundary layer near $y = 0$. That is, for large B's, W and Z are effectively zero outside a narrow band around $y = 0$ [5]. Such a problem needs a Δy small enough to resolve this layer. The appropriate Δy can be so small that to solve (2.3) on the whole region $(0, 1) \times (0, 1)$ would be prohibitively expensive. This consideration motivates the idea of solving (2.3) on a sequence of regions $(0, 1) \times (0, y_i)$, obtaining a sequence of V's, say V^i, and accepting V^i as the answer when $|V^i - V^{i-1}|\,(1 - y_i)/(y_i - y_{i-1}) \leqslant .1\,V^i$. Thus QUENCH is designed to solve to 10% accuracy in V.

The question remains of how to choose Δy. In [5] a similar procedure was employed, and the Δy derived there had to satisfy

$$\Delta y \leqslant \Delta u(2/\max f(u)); \tag{2.9}$$

this was a natural condition arising from the difference scheme employed in [5]. The first difference scheme employed in QUENCH to approximate $z\,\partial z/\partial u + \partial z/\partial y$ was a straightforward conservative differencing and gave good results aside from the fact that there was no obvious criterion to choose Δy. This latter failure motivated using the scheme (2.3) with

$$\theta_{i,j} = \theta^r_{i,j} = \theta^l_{i,j} = 1.-.25\Delta y(Z_{i,j} + Z_{i,j-1})/\Delta u;$$

this scheme is in the spirit of [5] and in effect differences $z\,\partial z/\partial u + \partial z/\partial y$ by differencing u_{yy} in the (x, y) coordinate system. Thus the $\theta_{i,j}$'s are interpolation coefficients for interpolating between $Z_{i,j}$ and $Z_{i+1,j}$ and between $Z_{i,j-1}$ and $Z_{i-1,j-1}$. The condition that $0 \leqslant \theta_{i,j} \leqslant 1$ yields the preceding restriction on Δy since in the continuous problem (1.1), z assumes its maximum value at $y = 0$. With the $\theta_{i,j}$'s defined this way, the differencing is not conservative; however, by averaging as in the definition of $\theta^r_{i,j}$ and $\theta^l_{i,j}$ in (2.4), the differencing becomes conservative.

If the method solves the one-dimensional problem (2.5) successfully, it chooses Δy according to (2.9); it then scales the answers by $W \leftarrow W/(\Delta y)^{1/2}$, $V \leftarrow V/(\Delta y)^{1/2}$ so as to provide the answer to (2.5) with f replaced by $f/\Delta y$. Then the method solves (2.3a) with one y interval with the same algorithm as for (2.5). If this is successful, the method begins adding on y intervals, guessing values for the new W's and Z's and attempting to solve (2.3) by Newton's method. The linearized systems are now no longer efficient to solve directly and one must have recourse to an iterative method.

The iterative method is a variant of distributed relaxation [6]. For a general system $Ax = b$, point distributed relaxation relaxes equation i, $\sum a_{ij}x_j = b_i$ by forming $x_j^{n+1} = x_j^n + a_{ij}\,\delta$, where one solves for δ by requiring that the residual is zero when x^{n+1} is substituted for x in the ith equation. One can generalize this notion by relaxing several equations simultaneously (in the spirit of block relaxation). If I is the set of indices of equations to be relaxed simultaneously, then one forms $x_j^{n+1} = x_j^n + \sum_{i\in I} a_{ij}\,\delta_i$ so that x^{n+1} substituted for x satisfies the ith equation for all $i \in I$.

Since Δy is usually much smaller than Δu, it is appropriate to use line distributed relaxation by lines in y, that is, to satisfy (2.3a) for a whole column of cells (i, j), $j = 2,...,J1$ at once. This gives rise to a linear system to solve for each i. The linear system is of the form

$$(T + A)\delta = r, \tag{2.10}$$

where T is tridiagonal, $\delta = (\delta_2,..., \delta_{J-1})^T$ and $(A)_{j,k} = \mu_j\mu_k$, where $\mu_j = \frac{1}{2}(W^n_{i,j} + W^n_{i-1,j})$. This system can be solved efficiently by defining $\delta_{JJ} = \sum_{j=2}^{J1} \mu_j\,\delta_j$ and rewriting (2.10) as

$$\begin{pmatrix} & & \mu_2 \\ & T & \vdots \\ & & \mu_{J1} \\ \mu_2 \cdots & \mu_{J1} & -1 \end{pmatrix} \begin{pmatrix} \delta_2 \\ \vdots \\ \delta_{J1} \\ \delta_{JJ} \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_{J1} \\ 0 \end{pmatrix}. \tag{2.11}$$

The matrix in (2.11) is a bordered tridiagonal matrix and can be solved efficiently.

A similar line relaxation scheme is used for (2.3b), giving rise in this case to only a tridiagonal matrix. Since only 10% accuracy in V is asked for by QUENCH, these relaxation schemes are relatively inexpensive. For more accuracy in V it is probably desirable to convert QUENCH into a multigrid code.

A final detail concerning the relaxation schemes is that a slight cheat is used for the relaxation on the linearized systems of (2.3a). In the distributed relaxation, the contribution $\theta^r_{i,j}(Z+\delta)Z_{i,j}$, etc., is ignored, since its inclusion would give rise to a bordered pentadiagonal matrix to solve, the assumption being that any improvement in convergence rate would be more than offset by the additional work required.

Let us comment on why conservative differencing is desirable for this problem. Note from (2.1a) and integration by parts that

$$v = \frac{\int f(u)\, du}{\iint w\, du\, dy}. \tag{2.12}$$

Since the difference scheme (2.3) is conservative the V and W obtained from it satisfy a discrete version of (2.12). Because of the way $zz_u + z_y$ is differenced, however, the discrete approximation to $\int f(u)\, du$ is a nonstandard one. In fact, we have found the quantity

$$VINT = \frac{\sum_{i=2}^{I1} f(\Delta u(i-\frac{3}{2}))\, \Delta u}{\sum_i \sum_j W_{ij}\, \Delta u\, \Delta y} \tag{2.13}$$

to be a more accurate approximate to v.

Let us remark on the unequal interval provision of QUENCH. This provision is provided for flexibility; since best results are obtained when there are meshpoints at the breakpoints of f, it is clear that unequal intervals are necessary to satisfy this constraint and yet keep the number of intervals small. When unequal intervals are allowed to change length sufficiently slowly, QUENCH apparently performs as well as in the equal interval case if the interior breakpoints of f are greater than about .1 and less than about .9. Otherwise, performance is unfortunately degraded. An example is given in Section 4.

Finally, let us remark that QUENCH will handle an equation more general than (1.1); indeed, it allows for the equation

$$\nabla \cdot (k(u)\, \nabla u) + vq(u)u = 0.$$

4. Numerical Examples

In this section we present several examples of problems worked with QUENCH. To describe the problems, we list the input variables, the definitions of which are as follows:

NBKP Number of breakpoints in piecewise linear f, counting 0. and 1.

ISW $ISW = 0$ for unequal intervals. $ISW = 1$ for equal intervals.

TOL *TOL* is the relative tolerance in the solution of the linearized systems. See also *WMAX*. $TOL = .01$ is recommended unless a negative Z or W is obtained as an answer. One case where this can happen is when there is an interior breakpoint $BKP(I)$ of f satisfying 0. less than $BKP(I)$ less than .1 or .9 less than $BKP(I)$ less than 1., in which case $TOL = .001$ is recommended.

WMAX *WMAX* is the maximum number of iterations. For each linearized problem the code will iterate until the discrete $L2$ norm of the residuals is less than $V * TOL$ or until the number of iterations exceeds *WMAX*.

D $D = 0.$ means solve only the one-dimensional problem. $D = 1.$ means solve the two-dimensional problem.

V V is the initial guess for the velocity. See remarks below on how to choose V.

BKP *BKP* is the array of breakpoints of f. $BKP(1) = 0.$ and $BKP(NBKP) = 1.$ must be specified.

B B is the array of slopes of f. $B(NBKP\text{-}1) = 0.$ must be specified.

FLP *FLP* is the array of values of $f(BKP(I)^+)$, i.e., the limits of f as $BKP(I)$ is approached from the right. $FLP(NBKP\text{-}1) = 0.$ must be specified.

NUO NUO = number of u intervals between 0. and 1.

NIND $NIND$ = index of breakpoint to be used in initial guess routine.

UMP $UMP(I)$ = values of U at Ith meshpoint. For best results it is recommended that there be a meshpoint at each of the breakpoints of f. In the equal interval case this array is computed by QUENCH and is $UMP(I) = (I - 1)/NUO$. In the unequal interval case the array must be input.

The procedure we use for choosing an initial guess for V is to exploit the fact that for f of the form (2.6), the one-dimensional problem can be solved exactly. For a general f, the procedure is to try to fit f with an f of the form (2.6) and guess V using the formula (2.7), where in this case u_c is taken to be $UMP(NIND)$. The initial guess routine then guesses W as in (2.8) and then attempts to solve the one-dimensional problem. QUENCH then chooses a mesh spacing in y, *HY*, which is printed, and proceeds to solve (2.3) on a sequence of regions $(0, 1) \times (0, y_l)$ as described in Section 2. For each member of the sequence, QUENCH arrives at a V and a *VINT*, the latter being obtained by (2.13). For problems in which we have

answers by other methods, *VINT* appears to be more accurate. When $|V^i - V^{i-1}|(1 - y_i)/(y_i - y_{i-1}) \leqslant .1V^i$, QUENCH terminates and accepts the last *VINT* as the velocity. The final *W*'s and *Z*'s are printed out. A warning is printed out if any of these are negative, in which case the answers should be viewed suspiciously since w and z can be proved to be nonnegative. A possible remedy is to run the problem again with a smaller *TOL* or a refined u mesh.

For each problem we give a history of the run, giving data for each region $(0, 1) \times (0, y_i)$. (Actually, to save space, a complete history is given only for the first example.) *JJ* is the number of y intervals, *WU* is the number of iterations, *ERR*1 and *ERR*2 are the discrete L-2 norms of the residuals of (2.3a) and (2.3b), respectively. *V* is the computed velocity and *VINT* is the integral approximation (2.12) to the velocity. $JJ = 3$ is the one-dimensional problem and is given twice, once with the vertical mesh spacing $HY = 1.$ and once with *HY* equal to what the code chooses. We also give CDC 7600 C.P.U. times with the caution that since these problems were run in a time-sharing environment, they should be interpreted in only a relative sense. We also use the shorthand 3.2, −1 to indicate, e.g., 3.2×10^{-1}.

EXAMPLE 1.

$$f(u) = 22.05u, \quad \text{if } u \leqslant .5,$$
$$= 0, \quad \text{if } u > .5.$$

Input data: $NBKP = 3$, $ISW = 1$, $TOL = .01$, $WMAX = 100.$, $D = 1.$, $V = 3.32$, $BKP(1) = 0.$, $BKP(2) = .5$, $BKP(3) = 1.$, $B(1) = 22.05$, $B(2) = 0.$, $FLP(1) = 0.$, $FLP(2) = 0.$, $NUO = 6$, $NIND = 2$.

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	1.	1., −3	0.	3.32	3.32
3	2.	1.78, −1	0.	17.4	20.8
4	14.	1.41, −1	2.46, −2	12.9	15.5
5	7.	1.28, −1	2.96, −2	10.9	13.1
6	6.	6.26, −2	5.05, −2	9.72	11.8
8	9.	8.53, −2	3.99, −2	8.47	10.3
10	10.	7.36, −2	4.08, −2	7.85	9.57
13	13.	5.98, −2	3.78, −2	7.39	9.01
17	14.	6.92, −2	4.46, −2	7.16	8.74
22	12.	6.78, −2	3.58, −2	7.11	8.68

CPU time = .578 sec, $HY = 3.0303..., -2$. Note that *VINT* agrees favorably with the value given in [1] of 8.409.

Also in this example we consider

$$f(u) = 1024.0u, \quad \text{if } u \leqslant .5,$$
$$= 0, \quad \text{if } u > .5.$$

The input data is the same except that $B(1) = 1024.$, $V = 22.62$.

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	1.	1.35, −1	0.	22.6	22.6
3	3.	1.16, −1	0.	8.10, 2	9.71, 2
4	21.	6.97, 0	1.22, 0	5.97, 2	7.19, 2
⋮					
66	66.	3.37, 0	1.72, 0	3.38, 2	4.13, 2
88	87.	3.37, 0	1.55, 0	3.38, 2	4.13, 2

CPU time = 7.074 sec, $HY = 6.5104, -3$. Asymptotic analysis [9] gives 390. for this problem, while [7], which works only for f's of this specific form, gives 380. ± .035. Thus QUENCH solves to 10% accuracy in v with six uniform u intervals even for this harder problem.

EXAMPLE 2.

$$\begin{aligned} f(u) &= 22.05u, && \text{if } u \leqslant .5, \\ &= 11.025 - 36.75\,(u - .5), && \text{if } .5 \leqslant u \leqslant .8, \\ &= 0, && \text{if } u > .8. \end{aligned}$$

Input data: $NBKP = 4$, $ISW = 0$, $TOL = .01$, $WMAX = 100.$, $D = 1.$, $V = 3.32$, $BKP(1) = 0.$, $BKP(2) = .5$, $BKP(3) = .8$, $BKP(4) = 1.$, $B(1) = 22.05$, $B(2) = -36.75$, $B(3) = 0.$, $FLP(1) = 0.$, $FLP(2) = 11.025$, $FLP(3) = 0.$, $NUO = 7$, $NIND = 2$, $UMP(1) = 0.$, $UMP(2) = .1666$, $UMP(2) = .3333$, $UMP(3) = .5$, $UMP(4) = .65$, $UMP(5) = .8$, $UMP(6) = 1.$

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	4.	1.96, −6	0.	5.17	5.7
3	3.	5.22, −4	0.	28.3	30.6
4	63.	2.58, −3	3.05, −4	22.0	23.9
⋮					
8	44.	1.72, −3	6.55, −4	17.7	19.4
10	46.	1.68, −3	7.05, −4	17.6	19.2

CPU time = .811 sec, $HY = 3.448276, -2$. This example illustrates two features: a general piecewise linear f and the nonuniform mesh option.

EXAMPLE 3.

$$\begin{aligned} f(u) &= 22.05u, && \text{if } u \leqslant .8333, \\ &= 0, && \text{if } u > .8333. \end{aligned}$$

Input data: $NBKP = 3$, $ISW = 1$, $TOL = .01$, $WMAX = 100.$, $D = 1.$, $V = 9.58$, $BKP(1) = 0.$, $BKP(2) = .8333$, $BKP(3) = 1.$, $B(1) = 22.05$, $B(2) = 0.$, $FLP(1) = 0.$, $FLP(2) = 0.$, $NUO = 12$, $NIND = 2$.

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	1.	9.3, −3	0.	9.59	9.59
3	2.	2.00, 0	6.37, −2	98.0	103.
4	54.	9.69, −1	1.64, −1	76.2	79.6
⋮					
10	19.	5.84, −1	1.65, −1	57.7	60.9
13	23.	5.64, −1	1.82, −1	57.8	60.9

CPU time = .910 sec, $HY = 9.0909..., -3$. It is interesting to compare this with a run using a nonuniform grid for which the input data is the same as above except that $ISW = 0$, $NUO = 7$, and $UMP(1) = 0.$, $UMP(2) = .2$, $UMP(3) = .4$, $UMP(4) = .6$, $UMP(5) = .75$, $UMP(6) = .8333$, $UMP(7) = .9166$, $UMP(8) = 1.$

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	1.	7.83, −3	0.	9.58	9.59
3	2.	7.07, −2	0.	95.1	100.
4	51.	9.38, −1	1.49, −1	74.4	77.7
⋮					
10	7.	5.65, −1	8.69, −1	56.7	60.0
13	18.	5.39, −1	1.41, −1	56.9	60.1

CPU time = .547 sec, $HY = 9.615385, -3$. Note that the *VINT*'s agree well with each other and with the value of 63.22 in [1].

EXAMPLE 4.

$$f(u) = 22.05u, \quad \text{if } u \leqslant .98,$$
$$= 0, \quad \text{if } u > .98.$$

Input data: $NBKP = 3$, $ISW = 0$, $TOL = .01$, $WMAX = 10{,}000.$, $D = 1$, $V = 32.54$, $BKP(1) = 0.$, $BKP(2) = .98$, $BKP(3) = 1.$, $B(1) = 22.05$, $B(2) = 0.$, $FLP(1) = 0.$, $FLP(2) = 1.$, $NUO = 14$, $NIND = 2$, $UMP(1) = 0.$, $UMP(2) = .2$, $UMP(3) = .4$, $UMP(4) = .6$, $UMP(5) = .7$, $UMP(6) = .76$, $UMP(7) = .82$, $UMP(8) = .88$, $UMP(9) = .92$, $UMP(10) = .94$, $UMP(11) = .96$, $UMP(12) = .97$, $UMP(13) = .98$, $UMP(14) = .99$, $UMP(15) = 1.$

For this problem QUENCH goes out to $JJ = 13$ with $HY = 9.310987, -4$ and obtains an answer of 8.48, 2 in 4.5 secs. However, the last Z obtained is negative, for all y, giving a warning that *TOL* is too large. When *TOL* is decreased to .005, this is still the case, V now being 7.29, 2, obtained in 10. sec. (In this case QUENCH goes

out to $JJ = 17$.) When TOL is decreased further to .001, the following results are obtained.

JJ	WU	$ERR1$	$ERR2$	V	$VINT$
3	1.	1.36, −2	0.	32.5	32.5
3	2.	2.94, −3	0.	1060.	1070.
4	5140.	1.06, −1	1.49, −2	823.	827.
17	30.	6.61, −1	6.11, −1	666.	669.
22	47.	6.55, −1	5.44, −1	665.	670.

CPU time = 25.8 sec, $HY = 9.310987, -4$. Hence, although the calculation is quite costly, the agreement with the value of 649.1 in [1] is good.

EXAMPLE 5.

$$f(u) = 156\,u^4, \quad \text{if } u \leqslant .666,$$
$$= 0, \quad \text{if } u > .666.$$

Input data: $NBKP = 3$, $ISW = 1$, $TOL = .01$, $WMAX = 100.$, $D = 1.$, $V = 4.92$, $BKP(1) = 0.$, $BKP(2) = .666$, $BKP(3) = 1.$, $B(1) = 156.$, $B(2) = 0.$, $FLP(1) = 0.$, $FLP(2) = 0.$, $NUO = 24$, $NIND = 2$.

This, of course, is an example of an f which is not piecewise linear.

JJ	WU	$ERR1$	$ERR2$	V	$VINT$
3	3.	3.61, −2	0.	6.1	6.1
3	2.	4.77, −2	0.	213.	217.
4	77.	2.08, −0	9.52, −2	153.	155.
66	32.	3.35, −1	8.38, −2	33.3	34.0
88	37.	3.21, −1	7.44, −2	30.9	31.5

In [8] a value of 25. is obtained for a nearby problem ($BKP(2) = .677$) and in [9] a value of 30. is obtained.

EXAMPLE 6.

$$f(u) = 22.05u, \quad \text{if } u \leqslant .5,$$
$$= 0, \quad \text{if } u > .5.$$

$k(u) = 1 + u$, $q(u) = 1 + u$. Input data: $NBKP = 3$, $ISW = 1$, $TOL = .01$, $WMAX = 100.$, $D = 1.$, $V = 3.32$, $BKP(1) = 0.$, $BKP(2) = .5$, $BKP(3) = 1.$, $B(1) = 22.05$, $B(2) = 0.$, $FLP(1) = 0.$, $FLP(2) = 0.$, $NUO = 6$, $NIND = 2$.

JJ	*WU*	*ERR*1	*ERR*2	*V*	*VINT*
3	16.	3.17, −2	0.	3.88	3.89
3	8.	2.13, −1	0.	20.	24.8
4	12.	1.96, −1	3.24, −2	14.7	18.4
17	14.	7.28, −2	4.69, −2	7.63	10.3
22	17.	6.78, −2	4.21, −2	7.46	10.2

CPU time = .63 sec, $HY = 3.0303..., -2$.

ACKNOWLEDGMENTS

We wish to thank Achi Brandt for advice concerning this problem when we first began work on it. In particular, he is responsible for the advice to use point or line distributed relaxation.

REFERENCES

1. J. G. M. ANDERSEN AND P. HANSEN, Report NORHAV-D-6, Danish Atomic Energy Comm. Res. Estab., Risø, Denmark, 1974.
2. M. CHENEY AND J. E. DENDY, JR., "QUENCH—A Code for the Calculation of Travelling Wave Solutions of a Quench Front Problem," Los Alamos Scientific Laboratory Report LA-UR 79-2150, 1979.
3. J. CRANK AND R. S. GUPTA, *Int. J. Heat Mass Transfer* **18** (1975), 1101–1107.
4. J. E. DENDY, JR., B. SWARTZ, AND B. WENDROFF, Computing travelling wave solutions of a nonlinear heat equation, *in* "Topics in Numerical Analysis III" (J. Miller, Ed.), Academic Press, New York, 1977.
5. D. DURACK AND B. WENDROFF, *Nucl. Sci. Eng.* **64** (1977), 187–191.
6. S. KACZMARZ, Angenäherte Auflösung von Systemen linearer Gleichungen, *Bull. Acad. Polon. Sci. Lett. A* (1937), 355–357.
7. H. T. LAQUER AND B. WENDROFF, Bounds for the model quench front, *SIAM J. Numer. Anal.*, in press.
8. T. S. THOMPSON, *Nucl. Eng. Des.* **31** (1974), 234–245.
9. B. WENDROFF, *Nucl. Eng. Des.* **52** (1979), 219–223.