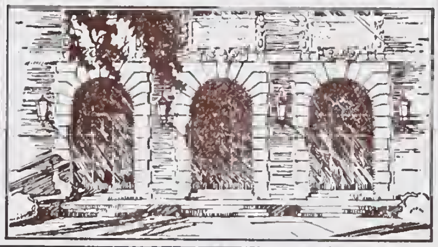


UIUCDCS-R-75-754

math

COO-2383-0025

510.84
Il6r
no.754

AUTOMATIC INTEGRATION OF THE HEAT EQUATION

by

MICHAEL H. OSTRAR

October 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-754 COO-2383-0025

AUTOMATIC INTEGRATION OF THE HEAT EQUATION

by

MICHAEL H. OSTRAR

October 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

* Supported in part by the Energy Research and Development Authority under contract US AEC AT(11-1)2383 and submitted in partial fulfillment of the requirements of the Graduate College for the degree of Master of Science in Computer Science.

ACKNOWLEDGMENT

My deepest thanks and appreciation are due Professor Daniel Watanabe for his continuous guidance and encouragement, Professor C. W. Gear for his support of this work, and Ms. Pamela Farr for her extremely quick and efficient job in typing this thesis.

ABSTRACT

A scheme for automatically integrating the heat equation is discussed. A program based on this scheme is explained, and the results of integrating the heat equation for a number of different initial conditions are presented and interpreted.

TABLE OF CONTENTS

1. Introduction

The method of lines is a technique for numerically solving partial differential equations by replacing the partial derivatives of all but one independent variable by finite difference approximations. This results in a system of ordinary differential equations in the remaining independent variable which can then be solved by using a standard numerical integration scheme. Such a technique is very flexible, since it can be used with almost any type of finite difference approximations and with almost any standard method for numerically solving ordinary differential equations. This suggests that the method of lines might be well-suited for use in an automatic program for controlling the errors in the solution of partial differential equations. By employing the method of lines with a fixed mesh, an automatic program could vary one or more of the following as it deems it necessary in order to control the error in accordance with any particular set of criteria:

1. The size of the integration step in the remaining independent variable.
2. The numerical integration scheme used.
3. The order of the finite difference approximations used.

Automatic programs for ordinary differential equations attempt to control the global error by bounding the local error introduced at each integration step. Since the theory behind this procedure is not yet fully developed, it is unreasonable to expect the theory for automatically solving partial differential equations to be any further advanced. However, in the following section, it is shown that by using the method of lines and making several reasonable assumptions, it is possible to employ an heuristic scheme to control the error automatically in the solution of the heat equation. Section 3 describes a program that was written to test experimentally the validity of the analysis in section one and describes the numerical tests run with the program. The final section discusses the results of these tests.

2. Estimating the Global Error

Suppose that we wish to solve numerically the heat equation

$$u_t = \lambda u_{xx} \tag{1}$$

using the method of lines. In order to use this technique, at $t_0 = 0$ we choose a fixed set of equispaced points $\{x_i\}$ at which we seek approximations to the solution $\underline{u}(t) = (u(t, x_i))^T$ at the times $\{t_i\}$. The t_i's are functions of the local error tolerance ε, since the integration stepsize must be controlled in order to keep the local error below ε. We approximate $\underline{u}_{xx} = (u_{xx}(x_i, t))^T$ by the centered difference operator $A(T, \varepsilon)$. Suppose that $A(t, \varepsilon)$ is controlled such that the following conditions always hold:

i. $A(t, \varepsilon)$ does not change on $[t_j, t_{j+1})$. It may change only at the points $\{t_j\}$. The elements $\{a_{ij}(t, \varepsilon)\}$ of A are thus piecewise constant.

ii. $A(t, \varepsilon)$ is symmetric and negative definite for all $t \geq 0$.

iii. The error $\underline{\sigma}(t, \varepsilon) = \underline{u}_{xx} - A\underline{u}$ in approximating the partial derivatives by finite differences is controlled at t_j such that $\lambda \Delta t_j ||\underline{\sigma}(t_j, \varepsilon)||_2 \leq \varepsilon/2$ where $\Delta t_j = t_{j+1} - t_j$. In practice, this is too much to expect. At the end of this section, we discuss the effect of relaxing this requirement to make it more realistic.

Let A_j denote $A(t_j, \varepsilon)$. In the event that $A(t_j, \varepsilon) \neq A(t_{j+1}, \varepsilon)$ define $A(t_{j+1}^-, \varepsilon) = A(t_j, \varepsilon) = A_j$

Let $\underline{v}$ denote the exact solution to

$$\underline{v}_t = \lambda A(t, \varepsilon)\underline{v} \tag{2}$$

We obtain approximations $\underline{w}_j$ to $\underline{v}_j = \underline{v}(t_j, \varepsilon)$ by numerical integration using the trapezoidal rule

$$\underline{w}_{j+1} = \underline{w}_j + 1/2\ \lambda\Delta t_j[A(t_j)\underline{w}_j + A(t_{j+1}^-)\underline{w}_{j+1}]$$

$$= \underline{w}_j + 1/2\ \lambda\Delta t_j A(t_j)[\underline{w}_j + \underline{w}_{j+1}] \qquad (3)$$

Define $\underline{\tau}_j$ to be the local truncation error of the trapezoidal method in (3). That is, if $\underline{w}_j = \underline{v}_j$, then $\underline{\tau}_j \equiv \underline{v}_{j+1} - \underline{w}_{j+1}$. Assume for the moment that we can control the stepsizes $\{\Delta t_j\}$ such that for all $j \geq 0$, $||\underline{\tau}_j||_2 = \varepsilon/2$. Later, we will examine the effect of slightly weakening this assumption. Let $\underline{z} = \underline{v} - \underline{w}$ be the difference between the exact and numerical solutions to (2). We will now derive a bound for $||\underline{z}_n||_2$, then address ourselves to the question of bounding $||\underline{u} - \underline{v}||_2$, and lastly combine the result of the two analyses to arrive at an estimate of the global error $||\underline{u} - \underline{w}||_2$ for the program.

In solving (2) by the trapezoidal rule, each time a step is taken, the current global error is multiplied by $\exp(\Delta t_j \lambda A_j)$, and a new local truncation error is added. We thus have

$$\underline{z}_{j+1} = \exp(\Delta t_j \lambda A_j)\underline{z}_j + \underline{\tau}_j \qquad (4)$$

Assuming that $\underline{w}_0 = \underline{v}_0$, then $\underline{z}_0 = \underline{0}$, and applying (4), we have

$$\underline{z}_n = \sum_{j=0}^{n-1} [\prod_{i=j+1}^{n-1} \exp(\Delta t_j \lambda A_i)]\underline{\tau}_j$$

Since the A_i's are symmetric negative definite, $||\exp(\Delta t_j \lambda A_i)||_2 \leq 1$. Using this fact and the assumption that $||\underline{\tau}_j||_2 = \varepsilon/2$, we obtain

$$||\underline{z}_n||_2 \leq \sum_{j=0}^{n-1} \varepsilon/2 = n\varepsilon/2 \qquad (5)$$

We now examine $||\underline{u} - \underline{v}||_2$. Using the above definition of $\underline{\sigma}(t, \varepsilon)$, we can rewrite (1) as

$$\underline{u}_t = \lambda A(t, \varepsilon)\underline{u} + \lambda\underline{\sigma}(t, \varepsilon) \qquad (6)$$

Subtracting (2) and integrating, we have

$$||\underline{u}(t_n) - \underline{v}(t_n)||_2 \leq ||\int_0^{t_n} \exp[\lambda \int_s^{t_n} A(s', \varepsilon)ds']\lambda\underline{\sigma}(s, \varepsilon)ds||_2$$

We again use the fact that $A(s', \varepsilon)$ being symmetric negative definite implies $||\exp[\lambda\int_s^{t_n} A(s', \varepsilon)ds']||_2 \leq 1$ to get

$$||\underline{u}(t_n) - \underline{v}(t_n)||_2 \leq \int_0^{t_n} ||\exp[\lambda\int_s^{t_n} A(s', \varepsilon)ds']||_2 ||\lambda\underline{\sigma}(s, \varepsilon)||_2 ds$$

$$\leq \int_0^{t_n} ||\lambda\underline{\sigma}(s, \varepsilon)||_2 ds$$

We approximate $\int_{t_j}^{\bar{t}_{j+1}} ||\lambda\underline{\sigma}(s, \varepsilon)||_2 ds$ by $\Delta t_j \lambda ||\underline{\sigma}(t_j, \varepsilon)||_2$. Assuming that the error in this approximation is small enough to be neglected, we can use the assumption that $\lambda\Delta t_j ||\underline{\sigma}(t, \varepsilon)||_2 \leq \varepsilon/2$ to get

$$||\underline{u}(t_n) - \underline{v}(t_n)||_2 \leq \varepsilon/2 \sum_{j=0}^{n-1} \int_{t_j}^{\bar{t}_{j+1}} ds/\Delta t_j = n\varepsilon/2 \tag{7}$$

Finally, we combine (5) and (7) to get a bound for $||\underline{u} - \underline{w}||_2$. Using the triangle inequality, we get

$$||\underline{u} - \underline{w}||_2 \leq n\varepsilon \tag{8}$$

Since we are using the trapezoidal rule to integrate (2) numerically, it is reasonable to expect that $||\underline{\tau}_j||_2 = O(\Delta t_j^3)$. But, since $||\underline{\tau}_j||_2 = \varepsilon/2$, it follows that $\Delta t_j^3 = O(\varepsilon)$ and hence, that $\Delta t_j = O(\varepsilon^{1/3})$. If we now assume that $\Delta t_j = O(1/n)$ (as it ought to be), then we have $n = O(\varepsilon^{-1/3})$. Substituting this relation in (8), we get

$$||\underline{u} - \underline{w}||_2 \leq k\varepsilon^{2/3} \tag{9}$$

for some constant k.

In practice, we cannot expect a program to actually control $\underline{\sigma}(t_j, \varepsilon)$ and $\underline{\tau}_j$ as strictly as was assumed in the above analysis. We will now look at the consequences of reducing these expectations to a realizable level. Since $\underline{u}$ is smooth and $A(t, \varepsilon)$ is piecewise smooth in t, we can express the truncation error of the trapezoidal method at $\bar{t}_{j+1}$ as a power series in Δt_j

$$\underline{\tau}_j = \sum_{i=3}^{\infty} \underline{\tau}_{ij} \Delta t_j^i$$

Instead of requiring that $||\underline{\tau}_j||_2 = \varepsilon/2$, we require that $||\underline{\tau}_{3j} \Delta t_j^3||_2 \leq \varepsilon/2$. In practice, we do not know $\underline{\tau}_{3j}$ exactly, but we will assume that we can estimate it accurately enough so that the following conclusions are not altered. Since $||\underline{\tau}_{3j} \Delta t_j^3||_2 \leq \varepsilon/2$, we may infer that $\Delta t_j = 0(\varepsilon\ 1/3)$, and hence

$$||\underline{\tau}_j||_2 = ||\underline{\tau}_{3j} \Delta t_j^3||_2 + o(\Delta t_j^3) \leq \varepsilon/2 + o(\varepsilon) \tag{10}$$

The (at least) piecewise smoothness of $\underline{u}$ and $A(t, \varepsilon)$ allows us to similarly write $\underline{\sigma}(t, \varepsilon)$ on $[t_j, t_{j+1})$ as $\underline{\sigma}(t, \varepsilon) = \sum_{i=p_j}^{\infty} \underline{\sigma}_i(t, \varepsilon) \Delta x^i$. We will no longer purport to control $\underline{\sigma}$ such that $\lambda \Delta t_j ||\underline{\sigma}(t_j, \varepsilon)||_2 \leq \varepsilon/2$. Instead, we only require that

$$\lambda \Delta t_j ||\underline{\sigma}_{p_j}(t_j, \varepsilon) \Delta x^{p_j}||_2 \leq \varepsilon/2 \tag{11}$$

Once again, in practice, we only have an estimate of $\underline{\sigma}_{p_j}(t_j, \varepsilon)$, but we will assume that the estimate is sufficiently accurate for the discussion to remain unaltered. Since $\Delta t_j = 0(\varepsilon^{1/3})$, it follows from (11) that $\Delta x = 0(\varepsilon^{2/3p_j})$ and hence,

$$\Delta t_j ||\underline{\sigma}(t, \varepsilon)|| \leq \varepsilon/2 + o(\varepsilon^{1+2/3p_j}) \tag{12}$$

We can therefore, still expect the results of the previous analysis to hold, even when the requirements are less stringent.

3. Programming Application

A version of the numerical procedure analyzed in the previous section was implemented in a FORTRAN IV program. This section begins by giving some of the particulars of its operation. For a more detailed account of its functioning, see the block diagram which has been included in the appendix. The program was written specifically to solve the heat equation for various initial conditions. It approximates the second spatial derivatives using centered differences and uses the trapezoidal rule to integrate with respect to time. It was numerically verified that all of the centered difference formulas used give rise to symmetric matrices $A(t_j, \varepsilon)$ which are negative definite.

Prior to attempting an integration step, an estimate of the local spatial truncation error is made using centered differences on the current numerical solution to estimate the magnitude of the higher order derivative which appears in the leading term of the Taylor expansion for the error. If the estimated error exceeds $\varepsilon/2$, the current centered difference formula for the second derivatives is replaced by another centered difference formula which is two orders of accuracy in Δx higher. The local spatial truncation error for the new formula is estimated, and if its magnitude still exceeds $\varepsilon/2$, the process repeats with the centered difference formula that is higher in order by yet two more powers of Δx. This continues until either a formula is found for which the estimated error is less than $\varepsilon/2$ in magnitude or the order of the formula reaches a limit specified in the initialization section of the program.

If, on the other hand, the estimated spatial truncation error is much less than $\varepsilon/2$, an estimate is made of what the spatial truncation error would be if the current centered difference formula were replaced by a centered difference formula whose accuracy is two orders lower. If the estimated error for this lower order formula still is less than $\varepsilon/2$ in magnitude, the lower order formula replaces the current formula.

An integration step in t is then taken, and the temporal truncation error is estimated using step-doubling. This procedure involves taking a step of size Δt and comparing the result with the solution obtained by taking two steps of size $\Delta t/2$. If the estimated error exceeds $\varepsilon/2$, Δt is adjusted downward, and the integration step is reattempted with the adjusted Δt. If, instead the estimated error is less than $\varepsilon/2$, the step is accepted, and Δt may possibly be increased for the next step.

The program was used to solve the heat equation

$$u_t = \lambda u_{xx}$$

with the diffusivity constant $\lambda = 1/10$ subject to the initial condition

$$u(x, 0) = f(x)$$

We assumed f(x) was periodic with period 2 and antisymmetric with respect to $x = 0, \pm 1, \pm 2, \ldots$. Hence, we can restrict our attention to the interval [0, 1]. This reduces the run time and also permits us to use centered differences at all mesh points in [0,1]. Under these assumptions, with boundary conditions

$$u(0, t) = u(1, t) = 0,$$

the solution is

$$u(x, t) = \sum_{n=1}^{\infty} b_n \exp(-n^2\pi^2\lambda t) \sin n\pi x$$

where
$$b_n = 2 \int_0^1 f(x) \sin n\pi x \, dx$$

The following initial conditions were used:

Case 1. $f(x) = \sin \pi x$,

$b_n = 1, \quad n = 1,$

$\quad\;\; 0, \quad n > 1.$

Case 2. $f(x) = 1 \quad 0 < x < 1.$

$b_n = 4/(n\pi), \quad n$ odd,

$\quad\;\; 0, \quad n$ even.

Here $u(x, 0)$ is a square-wave.

Case 3. $f(x) = 4x, \quad 0 \leq x \leq .25,$

$\quad 1, \quad .25 \leq x \leq .75,$

$\quad 4 - 4x, \quad .75 \leq x \leq 1.$

$b_n = \frac{8}{n\pi} (\sin n\pi/4 + \sin 3n\pi/4), \quad n$ odd,

$\quad\;\; 0, \quad n$ even.

Here $u(x, 0)$ is an isoceles trapezoid on $[0, 1]$.

Case 4. $f(x) = x/a, \quad 0 \leq x \leq a,$

$\quad = 1 - (x - a)/(1 - a), \quad a \leq x \leq 1.$

$b_n = 2 \sin (n\pi\mu)/[n^2\pi^2\mu(1 - a)].$

Case 4a. $\mu = 1/2$

Case 4b. $\mu = 7/10$

Case 4c. $\mu = 9/10$

Case 4d. $\mu = 999/1000$

Here $u(x, 0)$ on $[0, 1]$ is a peak with its summit at $x = a$. As a approaches 1, the slope to the left of μ slowly decreases, while the slope to the right of μ becomes steeper at an ever increasing rate. For μ close to 1, $u(x, 0)$ is very nearly a sawtooth wave.

The program was run for each of the above cases with ε's of 10^{-2}, 10^{-3}, ..., 10^{-8} except for CASE 1 where it was also run for $\varepsilon = 10^{-9}$ and $\varepsilon = 10^{-10}$. For all cases except for CASE 1, the integration was begun at $t = 1/10$ rather than at $t = 0$ in order to eliminate the difficulties arising from the discontinuities at zero. The initial number of points in the centered difference approximation to the second derivative was three. The initial settings for Δt and the number of internal mesh points were .01 and 19 respectively. Integration was halted as soon as t exceeded 2.

4. Results

The results of the tests described above are summarized in the tables which appear in this section. An explanation of each column that appears in the table is given below:

<u>ε</u> - On each integration step, the magnitude of the estimated spatial and temporal truncation errors are both required to be less than or equal to $\varepsilon/2$.

<u># CALLS to RKSTEP</u> - Each time that the subroutine RKSTEP is called, an integration is performed using the trapezoidal rule. Since the program uses step doubling, each time a step is attempted, the number of calls to RKSTEP is increased by three.

<u># CALLS times # POINTS</u> - Each time RKSTEP is called, the current number of internal mesh points is added to this number. It thus provides a rough measure of the work done by the program.

<u>LOG 10 DIFFERENCE in WORK</u> - The difference between the log (base 10) of the entry from the current row of the # CALLS TIMES # POINTS column and the log (base 10) of the entry from the preceding row in that column. This may give a better idea of the rate at which the work done by this program increases for each factor of ten decrease in ε.

<u>INTERNAL MESH POINTS</u> - The number of equispaced points $\{x_i\}$ (exclusive of $x_0 = 0$ and $x_{n+1} = 1$) at which the solution is computed. If the program has difficulty in taking the first step successfully, it may attempt to raise the number of points.

<u>INITIAL COUPLING</u> - The number of points used in the centered difference approximation to the second derivative when the first successful step was taken.

FINAL COUPLING - The number of points used in the centered difference approximation to the second derivative when the upper limit of integration was reached.

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	153	1957		19	3	3	.11	.73E-3	.52E-3	
.1E-2	36	475	-.385	19	5	5	.24	.40E-3	.24E-3	-.336
.1E-3	60	779	.215	19	5	5	.18	.94E-4	.93E-4	-.412
.1E-4	102	1311	.226	19	7	7	.078	.24E-4	.23E-4	-.606
.1E-5	213	2717	.317	19	7	7	.037	.52E-5	.50E-5	-.663
.1E-6	456	5795	.329	19	7	7	.020	.11E-5	.11E-5	-.658
.1E-7	1005	12749	.342	19	9	7	.0079	.23E-6	.21E-6	-.719
.1E-8	2097	26581	.319	19	9	9	.0037	.53E-7	.50E-7	-.623
.1E-9	4521	57285	.333	19	11	9	.0017	.11E-7	.11E-7	-.658

TABLE 1. RESULTS FOR CASE 1 (SINE WAVE).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	45	589		19	5	5	.250	.398E-3	.281E-3	
.1E-2	51	665	.053	19	7	5	.221	.263E-3	.227E-3	-.093
.1E-3	84	1083	.212	19	11	5	.188	.946E-4	.946E-4	-.380
.1E-4	174	2593	.379	22	11	7	.0826	.221E-4	.221E-4	-.632
.1E-5	369	6935	.427	28	11	7	.0406	.459E-5	.459E-5	-.682
.1E-6	789	18991	.438	36	11	7	.0163	.100E-5	.100E-5	-.662
.1E-7	1707	51274	.431	45	11	7	.00737	.212E-6	.212E-6	-.674

TABLE 2. RESULTS FOR CASE 2 (SQUARE WAVE).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	36	475		19	5	5	.250	.399E-3	.384E-3	
.1E-2	42	551	.064	19	7	5	.233	.292E-3	.244E-3	-.033
.1E-3	66	855	.191	19	9	5	.167	.961E-4	.945E-4	-.412
.1E-4	132	1691	.296	19	11	7	.0740	.215E-4	.210E-4	-.653
.1E-5	282	4177	.393	22	11	7	.0399	.470E-5	.470E-5	-.650
.1E-6	594	11135	.426	28	11	7	.0166	.102E-5	.101E-5	-.668
.1E-7	1299	31231	.445	36	11	7	.00901	.216E-6	.216E-6	-.670

TABLE 3. RESULTS FOR CASE 3 (TRAPEZOID).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	36	475		19	5	3	.25	.40E-3	.15E-3	
.1E-2	39	513	.034	19	7	5	.25	.22E-3	.18E-3	+.083
.1E-3	60	779	.182	19	9	5	.19	.84E-4	.84E-4	-.341
.1E-4	123	1577	.306	19	11	7	.090	.19E-4	.19E-4	-.652
.1E-5	261	3869	.390	22	11	7	.047	.40E-5	.40E-5	-.672
.1E-6	555	10407	.419	28	11	7	.021	.88E-6	.88E-6	-.656
.1E-7	1200	28855	.457	36	11	7	.00884	.19E-6	.19E-6	-.671

TABLE 4. RESULTS FOR CASE 4a (PEAK with $\mu = .5$).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	36	475		19	5	3	.250	.750E-3	.147E-3	
.1E-2	42	551	.064	19	7	5	.224	.234E-3	.141E-3	-.018
.1E-3	69	893	.210	19	9	5	.207	.728E-4	.728E-4	-.287
.1E-4	135	1729	.287	19	11	7	.0900	.159E-4	.159E-4	-.661
.1E-5	291	4504	.416	23	11	7	.0432	.349E-5	.349E-5	-.658
.1E-6	618	11996	.425	29	11	7	.0196	.746E-6	.746E-6	-.670
.1E-7	1335	32095	.427	36	11	7	.0104	.158E-6	.158E-6	-.674

TABLE 5. RESULTS FOR CASE 4b (PEAK with μ = .7).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	42	551		19	5	3	.250	.725E-3	.960E-4	
.1E-2	48	627	.056	19	7	5	.241	.222E-3	.130E-3	+.142
.1E-3	84	1083	.238	19	9	5	.222	.602E-4	.602E-4	-.344
.1E-4	171	2319	.330	20	11	7	.0906	.137E-4	.137E-4	-.643
.1E-5	360	6044	.416	25	11	7	.0474	.301E-5	.301E-5	-.658
.1E-6	783	16755	.443	32	11	7	.0187	.624E-6	.624E-6	-.684
.1E-7	1662	44379	.423	40	11	7	.0106	.130E-6	.130E-6	-.681

TABLE 6. RESULTS FOR CASE 4c (PEAK with $\mu = .9$).

ε	# CALLS TO RKSTEP	# CALLS TIMES # POINTS	LOG 10 DIFFERENCE IN WORK	INTERNAL MESH POINTS	INITIAL COUPLING	FINAL COUPLING	FINAL Δt	MAX ERROR EVER	MAX FINAL ERROR	LOG 10 DIFFERENCE IN MAX FINAL ERROR
.1E-1	45	589		19	5	3	.250	.694E-3	.868E-4	
.1E-2	51	665	.053	19	7	5	.247	.221E-3	.125E-3	+.158
.1E-3	81	1045	.196	19	11	5	.188	.568E-4	.568E-4	-.343
.1E-4	177	2637	.402	22	11	5	.0871	.110E-4	.108E-4	-.721
.1E-5	372	6991	.424	28	11	7	.0415	.278E-5	.278E-5	-.597
.1E-6	792	19063	.435	36	11	7	.0204	.621E-6	.621E-6	-.651
.1E-7	1692	50824	.426	45	11	7	.00909	.130E-6	.130E-6	-.679

TABLE 7. RESULTS FOR CASE 4d (PEAK with μ = .999, "SAWTOOTH").

<u>FINAL</u> <u>Δt</u> - The stepsize when the upper limit of integration was reached.

<u>MAX</u> <u>ERROR</u> <u>EVER</u> - The magnitude of the worst error in any individual component of the numerical solution vector that was ever found. The numerical solution was compared to the exact solution on each step that t equalled or exceeded an integral multiple of 2/10.

<u>MAX</u> <u>FINAL</u> <u>ERROR</u> - The magnitude of the worst error among the components of the final numerical solution vector.

<u>LOG</u> <u>10</u> <u>DIFFERENCE</u> <u>IN</u> <u>MAX</u> <u>FINAL</u> <u>ERROR</u> - This column is to MAX FINAL ERROR as LOG 10 DIFFERENCE IN WORK is to # CALLS TIMES # POINTS.

Results indicate that the idea of controlling the global error through separate control of the local spatial and temporal discretization errors is indeed quite feasible. Of particular interest is the LOG 10 DIFFERENCE IN MAX FINAL ERROR columns. These show that for each factor of ten decrease in ε, the global error decreased by a factor which appeared to approach $(10)^{2/3}$. This is as predicted in the previous section.

The LOG 10 DIFFERENCE IN WORK columns show that as ε was successively decreased by factors of ten, the amount of work done increased by factors which tended to stabilize in the range of $(10)^{1/3}$ to $(10)^{1/2}$ with the exact number depending on the initial condition. For CASE 1, where it was never found necessary to increase the number of internal mesh points, the number was precisely $(10)^{1/3}$. This is what would be expected with the trapezoidal rule. Since the local truncation error is proportional to the cube of the stepsize, and the program attempts to keep the local errors on the order of ε, if the mesh is fixed, the number of integration steps performed should be $O(\varepsilon^{-1/3})$.

Combining the relationships between ε and the global error and between ε and the amount of work done by the program, we can get a relationship between the global error and the work. Denoting the global error by e and the work by q, we can write

$$\lim_{\varepsilon \to 0} e(\varepsilon) = c_1 \varepsilon^{2/3}$$

$$\lim_{\varepsilon \to 0} q(\varepsilon) = c_2 \varepsilon^{-m} \qquad 1/3 \leq m \leq 1/2$$

and hence,

$$\lim_{\varepsilon \to 0} e(q) = c_3 q^{p} \qquad -2 \leq p \leq -4/3$$

where c_1, c_2, and c_3 are constants.

While this may not seem like a particularly good return of accuracy for the computational effort expended, it compares quite favorably with most of the methods commonly used to solve partial differential equations. The well-known Crank-Nicolson method which for the heat equation gives rise to the system

$$u_i^{n+1} = u_i^n + 1/2\ \lambda r(\delta_x^2\, u_i^n + \delta_x^2\, u_i^{n+1})$$

where $u_i^n = u(x_i, t)$, $r = \Delta t/\Delta x^2$, and δ_x^2 is the standard 3-point centered difference, has a local truncation error of $O(\Delta t^3 + \Delta t\ \Delta x^2)$. The Douglas formula is the most accurate formula involving the same six points to approximate $u_{xx}(x_i, t_{n+1})$. It generates the system

$$u_i^{n+1} \equiv u_i^n + 1/2\ \lambda r(\delta_x^2\, u_i^n + \delta_x^2\, u_i^{n+1}) + \frac{1}{12}\lambda(\delta_x^2\, u_i^n - \delta_x^2\, u_i^{n+1})$$

for the heat equation which results in a local error of $O(\Delta t^3 + \Delta t\ \Delta x^4)$. If, as Δt approaches zero, r is kept fixed, the local error is $O(\Delta t^3)$. The global error should therefore be $O(\Delta t^2)$. This means that

$$\lim_{\Delta t \to 0} e(\Delta t) = O(\Delta t^2)$$

just as for the trapezoidal rule.

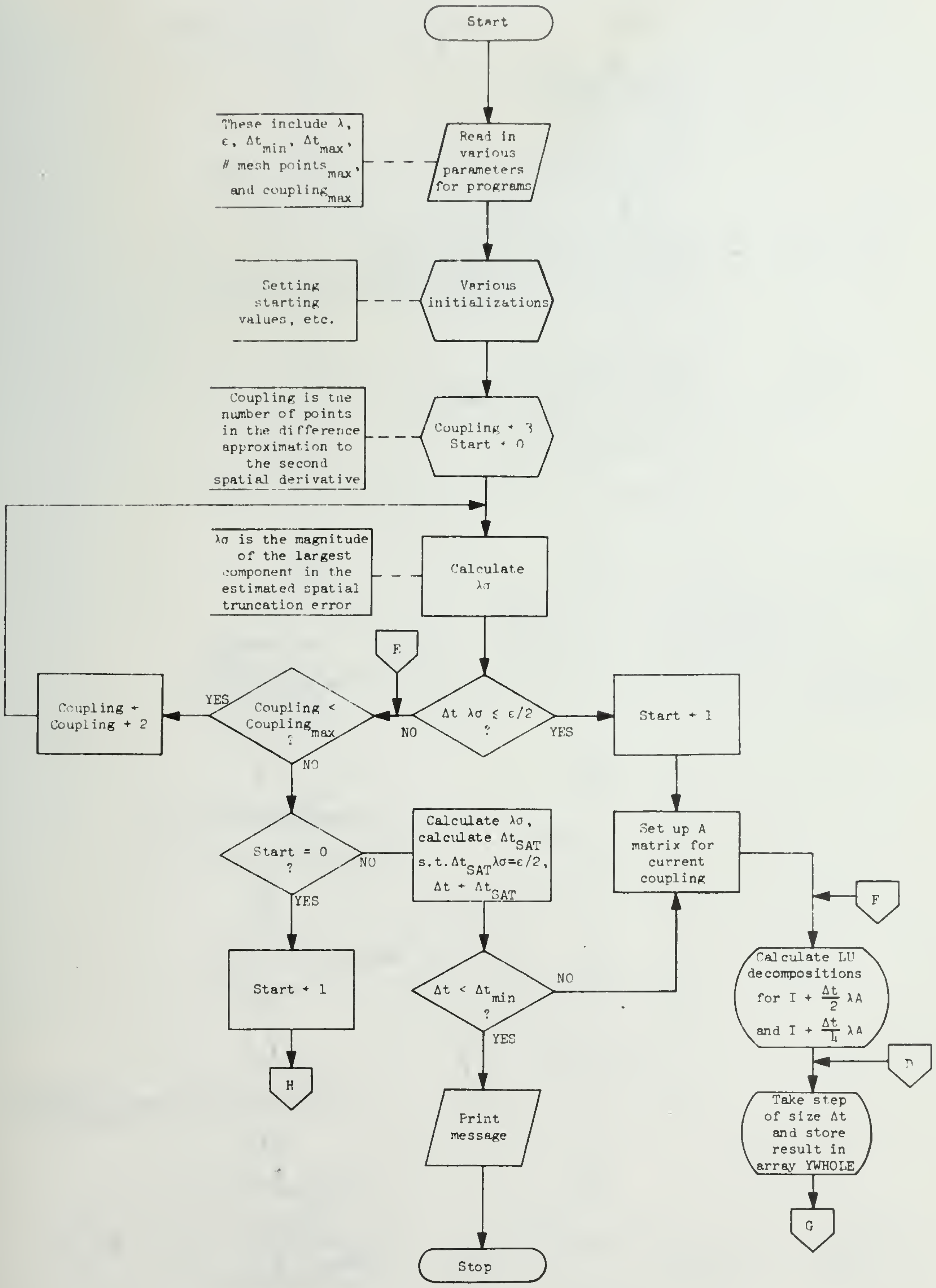
In advancing the solution one time step, Douglas' method must solve a tridiagonal system. The amount of work necessary to solve such a system is proportional to the number of equations in the system and hence to the number of internal mesh points. But since the number of internal mesh points is always within one of Δx^{-1}, we may take the work proportional to Δx^{-1}. The number of steps that must be taken and hence, the number of times the system must be solved is proportional to Δt^{-1}. Hence, for Douglas' method, $\Delta x^{-1}\ \Delta t^{-1}$ may be taken as a measure of work analogous to the measure # CALLS TIMES # POINTS described above for the program. If, as before, r is assumed to remain fixed as Δt varies, for the Douglas method we have

$$\lim_{\Delta t \to 0} q(\Delta t) = O(\Delta x^{-1}\ \Delta t^{-1}) = O(\Delta t^{-3/2})$$

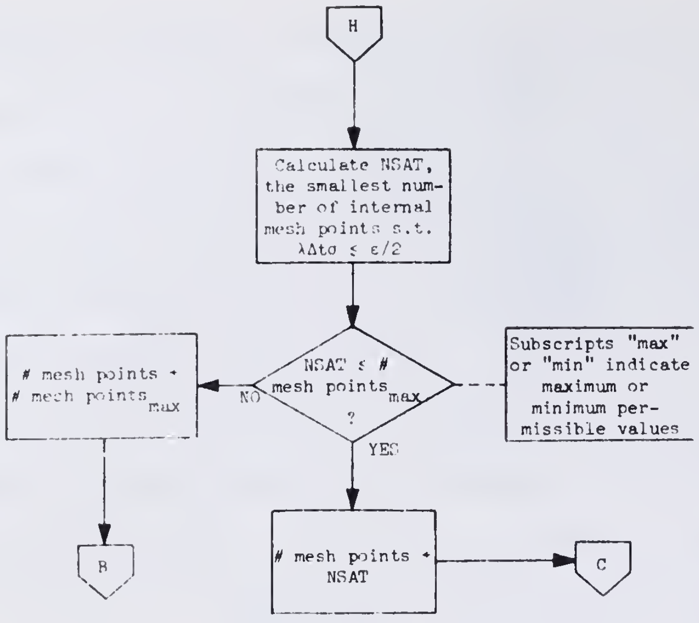
and hence,

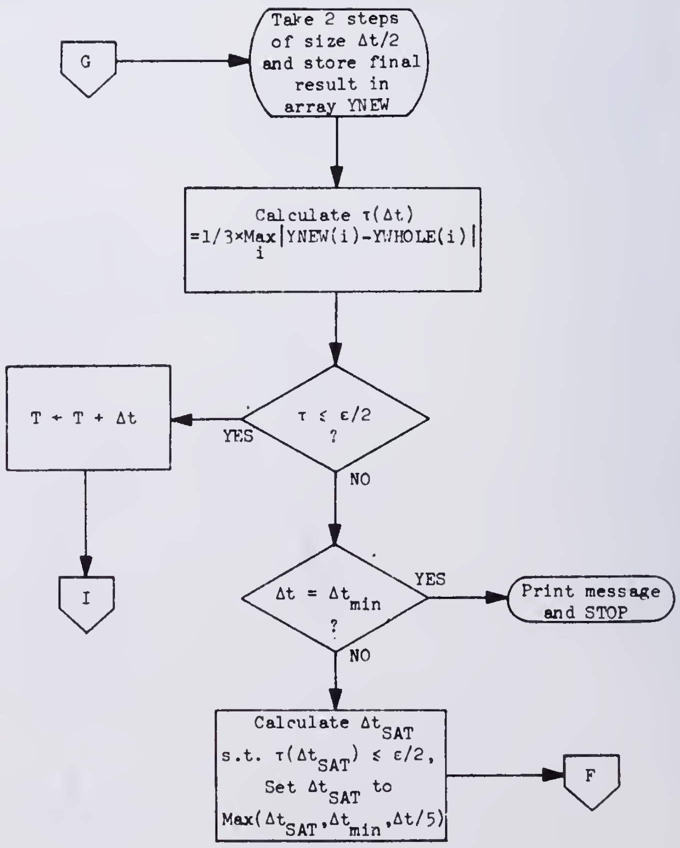
$$\lim_{\Delta t \to 0} e(q) = O(q^{-4/3})$$

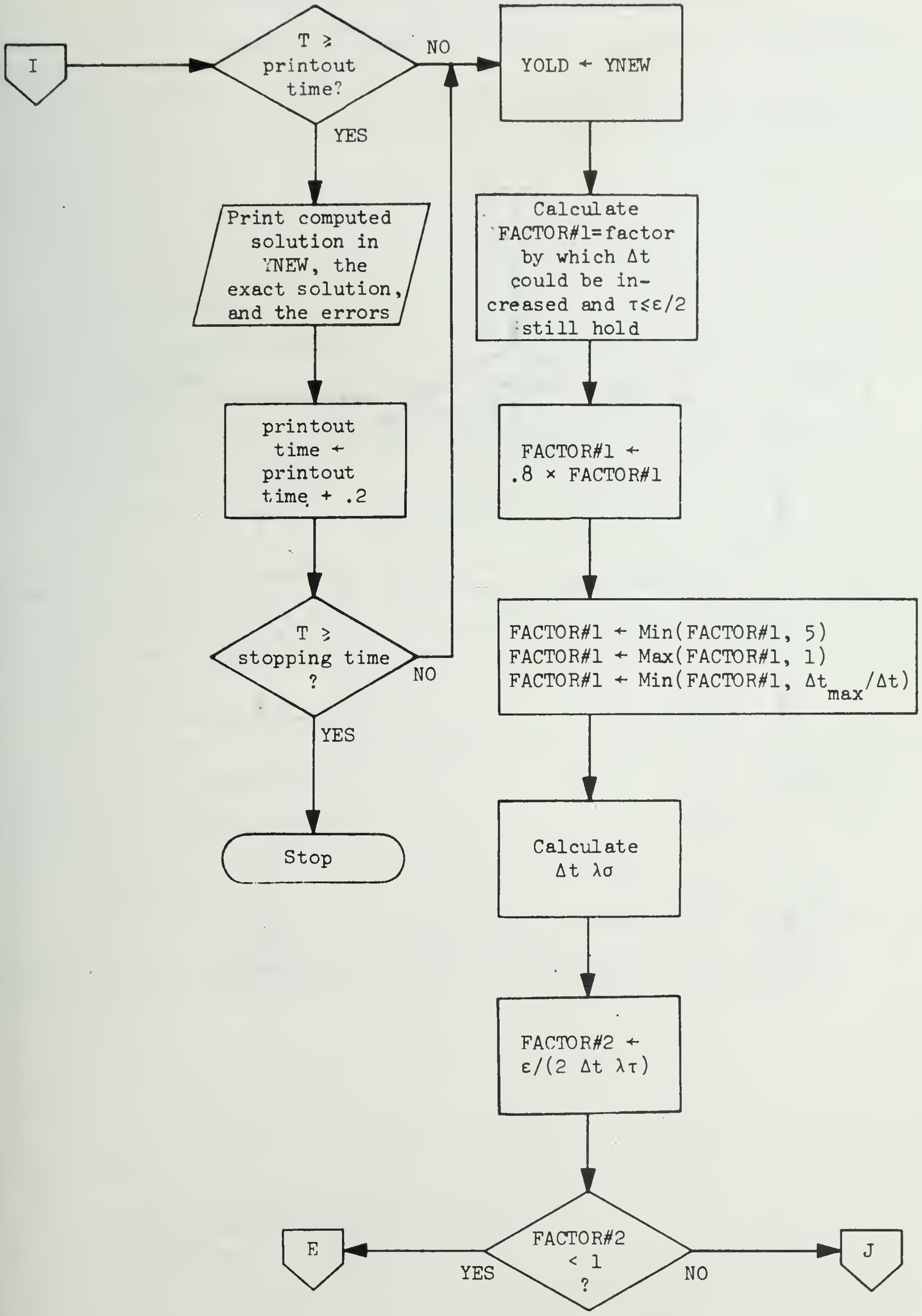
which is no better than the worst case observed for the program.

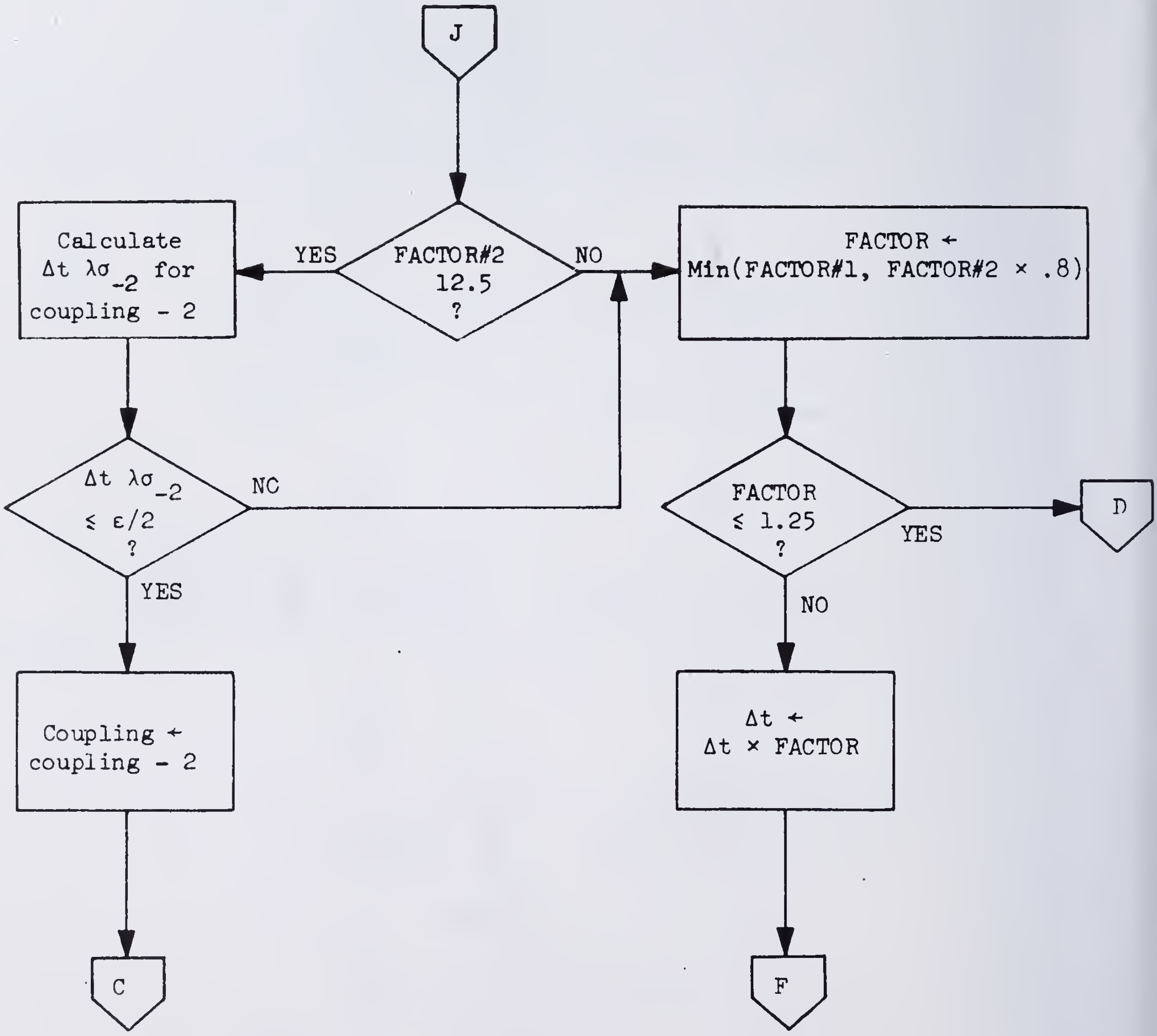


Start
Read in various parameters for programs
These include λ, ϵ, Δt_{min}, Δt_{max}, # mesh points$_{max}$, and coupling$_{max}$
Various initializations
Setting starting values, etc.
Coupling ← 3
Start ← 0
Coupling is the number of points in the difference approximation to the second spatial derivative
Calculate $\lambda\sigma$
$\lambda\sigma$ is the magnitude of the largest component in the estimated spatial truncation error
E
$\Delta t\ \lambda\sigma \le \epsilon/2$?
YES
NO
Start ← 1
Coupling < Coupling$_{max}$?
YES
NO
Coupling ← Coupling + 2
Start = 0 ?
YES
NO
Start ← 1
H
Calculate $\lambda\sigma$, calculate Δt_{SAT} s.t. $\Delta t_{SAT}\lambda\sigma=\epsilon/2$, $\Delta t \leftarrow \Delta t_{SAT}$
$\Delta t < \Delta t_{min}$?
NO
YES
Print message
Stop
Set up A matrix for current coupling
F
Calculate LU decompositions for $I + \frac{\Delta t}{2}\lambda A$ and $I + \frac{\Delta t}{4}\lambda A$
D
Take step of size Δt and store result in array YWHOLE
G


H
Calculate NSAT, the smallest number of internal mesh points s.t. λΔtσ ≤ ε/2
NSAT ≤ # mesh points max ?
NO
YES
# mesh points ← # mesh points max
Subscripts "max" or "min" indicate maximum or minimum permissible values
# mesh points ← NSAT
B
C


G
Take 2 steps of size Δt/2 and store final result in array YNEW
Calculate τ(Δt) =1/3×Max_i |YNEW(i)-YWHOLE(i)|
τ ≤ ε/2 ?
YES
NO
T ← T + Δt
I
Δt = Δt min ?
YES
NO
Print message and STOP
Calculate Δt SAT s.t. τ(Δt SAT) ≤ ε/2, Set Δt SAT to Max(Δt SAT, Δt min, Δt/5)
F


I
T ≥ printout time?
NO
YES
YOLD ← YNEW
Print computed solution in YNEW, the exact solution, and the errors
Calculate FACTOR#1=factor by which Δt could be increased and τ≤ε/2 still hold
printout time ← printout time + .2
FACTOR#1 ← .8 × FACTOR#1
T ≥ stopping time ?
NO
YES
FACTOR#1 ← Min(FACTOR#1, 5)
FACTOR#1 ← Max(FACTOR#1, 1)
FACTOR#1 ← Min(FACTOR#1, Δt_max/Δt)
Stop
Calculate Δt λσ
FACTOR#2 ← ε/(2 Δt λτ)
E
YES
FACTOR#2 < 1 ?
NO
J


J
YES
FACTOR#2 12.5 ?
NO
Calculate $\Delta t\ \lambda\sigma_{-2}$ for coupling - 2
FACTOR ← Min(FACTOR#1, FACTOR#2 × .8)
$\Delta t\ \lambda\sigma_{-2} \le \varepsilon/2$?
NC
FACTOR $\le$ 1.25 ?
YES
D
YES
NO
Coupling ← coupling - 2
$\Delta t \leftarrow \Delta t \times$ FACTOR
C
F

Form AEC–427
(6/68)
AECM 3201

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY–TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.

COO-2383-0025

2. TITLE

AUTOMATIC INTEGRATION OF THE HEAT EQUATION

3. TYPE OF DOCUMENT (Check one):

[✓] a. Scientific and technical report
☐ b. Conference paper not to be published in a journal:
Title of conference ______
Date of conference ______
Exact location of conference ______
Sponsoring organization ______
☐ c. Other (Specify) ______

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

[✓] a. AEC's normal announcement and distribution procedures may be followed.
☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
☐ c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization

University of Illinois
Department of Computer Science
Urbana, IL 61801

Signature [illegible]

Date October 3, 1975

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

☐ a. AEC patent clearance has been granted by responsible AEC patent group.
☐ b. Report has been sent to responsible AEC patent group for clearance.
☐ c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-754	2.	3. Recipient's Accession No.
4. Title and Subtitle AUTOMATIC INTEGRATION OF THE HEAT EQUATION			5. Report Date October 1975
			6.
7. Author(s) Michael H. Ostrar			8. Performing Organization Rept. No. UIUCDCS-R-75-754
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. US AEC AT(11-1)2383
12. Sponsoring Organization Name and Address US AEC Chicago Operations Office 9800 South Cass Avenue Argonne, IL 60439			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts

A scheme for automatically integrating the heat equation is discussed. A program based on this scheme is explained, and the results of integrating the heat equation for a number of different initial conditions are presented and interpreted.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 33
	20. Security Class (This Page UNCLASSIFIED	22. Price