5/91 *gmc* ①

SANDIA REPORT

SAND90-8251 • UC-705

Unlimited Release

Printed December 1990

An Implicit Integration Scheme For First Order Differential Equations

J. F. Lathrop

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94551
for the United States Department of Energy
under Contract DE-ACO4-76DP00789

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of the contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

This report has been reproduced from the best available copy.

Available to DOE and DOE contractors from:

Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge TN 37831

Prices available from (615) 576-8401, FTS 626-8401.

Available to the public from:

National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd.
Springfield, VA 22161.

SAND90-8251
Unlimited Distribution
Printed December 1990

DE92 003343

**An Implicit Integration Scheme
For First Order Differential Equations**

J. F. Lathrop
Scientific Computing and
Applied Math Division
Sandia National Laboratories
Livermore, California 94551

ABSTRACT

An unusual iterative technique is presented to solve the first order implicit integration equations. This method retains much of the simplicity of the first order explicit method, does not require the development of the usual Jacobian matrix, does not need to invert a matrix, and does not require the large amount of memory usually needed by implicit methods. A numerical example is also provided.

MASTER

CONTENTS

	<u>Page</u>
Introduction	7
A Different Way of Looking at the Problem	9
Advantage for Finite Element Codes	11
Numerical Example	13
Questions to be Answered	14
References	15

An Implicit Integration Scheme For First Order Differential Equations

Introduction

The development of this method was driven by problems associated with finite element codes that solve the heat conduction problem. The development of the finite element approximation leads to a system of first order differential equations of the form

$$K\Theta' = F(t, \Theta)$$

where K is the capacitance matrix, Θ a vector of nodal temperatures, and prime denotes time derivative. Boundary conditions may specify temperatures as time dependent functions at specified points.

Many finite element codes (in fact most explicit codes) approximate the capacitance matrix by a diagonal matrix. This corresponds to associating all the mass with the nodes. The obvious advantage of this is that the inverse is trivial to compute and hence we have the system of equations:

$$\Theta' = G(t, \Theta).$$

There are two simple first order integration methods that can be used to solve this system. The explicit formula is based on approximating the derivative by a forward difference,

$$\Theta(t+h) = \Theta(t) + h\Theta'(t), \quad (1)$$

and the implicit formula is based on approximating the derivative by a backward difference

$$\Theta(t+h) = \Theta(t) + h\Theta'(t+h). \quad (2)$$

The first formula has the obvious advantage of being easy to implement. At each step we evaluate the derivatives at each point, multiply by the time step, add it to the current temperatures and after applying the boundary conditions, we obtain the temperatures at the new time. The disadvantage is that the method is not unconditionally stable, and the stable time step is proportional to the spatial subdivision size. In practice this means that for a 3-D problem, cutting each spatial subdivision in half (for increased accuracy) cuts the stable time step by nearly an order of magnitude!

The second formula is unconditionally stable, and therefore allows larger time steps than the other formula. (Of course larger time steps means less accuracy, but the errors are bounded) The problem with the second formula is how to solve it. Most implicit codes

use Newton's method (or a variant of it) to solve the equation. At any time t , given an estimate w for $\Theta(t+h)$ we evaluate the residual $w - \Theta(t) - hG(t+h, w)$. An improved estimate can then be calculated from

$$\Delta w = -(I - hJ)^{-1}[w - \Theta(t) - hG(t+h, w)]$$

where I is the identity matrix and J the Jacobian of the function G

$$J_{ij} = \frac{\partial G_i(t+h, w)}{\partial w_j}.$$

This requires building the matrix J , and inverting $(I - hJ)$ at each step!

A lot of very clever work has gone into solving this problem efficiently. In many cases the Jacobian does not change very rapidly and so most codes only calculate the Jacobian and the associated inverse when needed. Where "need" is usually determined by convergence rate. The matrix J is sparse and various schemes have been developed for ordering the elements and nodes in the spatial discretization to minimize the bandwidth of the matrix. Other schemes have been developed to invert the matrix efficiently or to approximate the inverse with something that is easier to calculate. All of these methods have assumed that the best way to solve the problem is with a Newton's method iteration! We propose an iterative method that works well but is not based on Newton's method and does not require the Jacobian or an estimate of the Jacobian and does not need to invert any matrices!

A Different Way of Looking at the Problem

We still need to solve equation 2. Let's see what we might do if we had a "linear" problem (ie. the function G is a linear function of Θ). If

$$G(t, \Theta) = A\Theta + f(t),$$

where A is a matrix and $f(t)$ a vector valued function, then the problem to solve could be written as

$$(I - hA)\Theta(t + h) = \Theta(t) + hf(t + h). \quad (3)$$

We still need to solve a linear system of equations and the matrix A is really just the Jacobian matrix again. It looks like we haven't done any better than before! However, we will show that we can solve this problem without having to build the matrix A .

Our iterative solution is based on using the conjugate gradient method to solve equation 3. So let's review how the conjugate gradient method works.

To solve a system of equations $Mu = B$, we start with an approximate solution $u^{(0)}$.

0)	given an estimate $u^{(0)}$	
1)	find the residual vector	$R = B - Mu^{(i)}$
2)	calculate adjustment	$\alpha_i = -\frac{(R^{(i)}, MP^{(i-1)})}{(P^{(i-1)}, MP^{(i-1)})}$ if $i > 0$ $\alpha_i = 0$ if $i = 0$
3)	find new correction vector	$P^{(i)} = R^{(i)} + \alpha_i P^{(i)}$
4)	find correction amount	$\lambda_i = \frac{(P^{(i)}, P^{(i)})}{(P^{(i)}, MP^{(i)})}$
5)	calculate new estimate	$u^{(i+1)} = u^{(i)} + \lambda_i P^{(i)}$
6)	go to step 1	

In the above, (P, P) denotes a vector dot product.

Now, let's apply the conjugate gradient iteration to equation 3, remembering that A and $f(t)$ are from the approximation:

$$G(t, \Theta) = A\Theta + f(t). \quad (4)$$

We start with an estimate $u^{(0)}$ to $\Theta(t + h)$. An easy choice is to use formula 1 (the explicit rule)

$$u^{(0)} = \Theta(t) + hG(t, \Theta(t)).$$

The residual for this iteration is then given by

$$R = \Theta(t) - u^{(i)} + h(Au^{(i)} - f(t + h)).$$

However, if we substitute (4) into this we have

$$R^{(i)} = \Theta(t) - u^{(i)} + hG(t + h, u^{(i)}).$$

The first time through the iteration $\alpha = 0$ so $P = R$. To find λ we need to evaluate

$$MP = (I - hA)P = P - h[A(u + P) + f(t + h) - Au - f(t + h)].$$

Using relation (4) again we have,

$$MP = P - h[G(t + h, u + P) - G(t + h, u)].$$

We have now put together everything we need for our iterative method which we outline below.

0)	given $\Theta(t)$ and $\Theta' = G(t, \Theta)$	
a)	estimate $\Theta(t + h)$	$u^{(0)} = \Theta(t) + h\Theta'(t)$
b)	enforce boundary conditions on $u^{(0)}$	
1)	first evaluate	$v^{(i)} = \Theta(t) + hG(t + h, u^{(i)})$
a)	enforce boundary conditions on $v^{(i)}$	
b)	calculate the residual	$R^{(i)} = v^{(i)} - u^{(i)}$
2)	calculate adjustment if $i > 0$	$\alpha_i = \frac{(R^{(i)}, MP)}{(P^{(i-1)}, MP)}$
3)	get new adjustment vector	$P^{(i)} = R^{(i)} + \alpha_i P^{(i-1)}$
4) a)	evaluate MP	$MP = P^{(i)} - h[G(t + h, u^{(i)} + P^{(i)}) - G(t + h, u^{(i)})]$
b)	calculate correction amount	$\lambda_i = \frac{(P^{(i)}, P^{(i)})}{(P^{(i)}, MP)}$
5)	get the new estimate	$u^{(i+1)} = u^{(i)} + \lambda_i P^{(i)}$
6)	go to step 1	

Note that the estimate will always satisfy the boundary conditions since both $u^{(0)}$ and $v^{(i)}$ do, and therefore $R^{(i)}$ will be zero at points where the boundary values are specified. Also, note that since $G(t + h, u^{(i)})$ is calculated in step 1 it may be saved and need not be recalculated in step 4.

Advantage for Finite Element Codes

You will recall that finite element codes need to solve the system of differential equations

$$K\Theta' = F(t, \Theta).$$

In practice, both K and F are built up by evaluating the contributions from each element and adding them up. When K is approximated by a diagonal matrix, the code can calculate the two vectors K and $F(t, \Theta)$ element by element and then it is a simple task to calculate $G_i(t, \Theta) = \frac{F_i(t, \Theta)}{K_i}$.

Given routines to calculate the derivative and apply the boundary conditions then, an explicit code can be written to look like:

```

comment    (at time t, we have  $\Theta$ ,  $\Theta'$ , and  $h$ )
timeloop:  $\Theta_{est} = \Theta + h * \Theta'$ 
           call boundary( $t + h, \Theta_{est}$ )
           estprime = tderiv( $t + h, \Theta_{est}$ )
doneit:  $t = t + h$ 
         $\Theta = \Theta_{est}$ 
         $\Theta' = estprime$ 
process: Print & Plot
        go to timeloop

```

Since our method does not require calculation of a Jacobian matrix, we can use the same routines developed for the explicit program and write the following implicit code

```

comment    (at time t, we have  $\Theta$ ,  $\Theta'$ , and  $h$ )
timeloop:  $\Theta_{est} = \Theta + h * \Theta'$ 
           call boundary( $t + h, \Theta_{est}$ )
           ITER=0
step1: estprime = tderiv( $t + h, \Theta_{est}$ )
         $r = \Theta + h * estprime$ 
        call boundary( $t + h, r$ )
         $r = r - \Theta_{est}$ 
        if ( $|r| < \epsilon_1$ ) go to doneit
steps2-3: if (iter > 0) then
           alpha=dotp( $r, mp$ )/pmp
            $p = r - alpha * p$ 
        else
            $p = r$ 
        endif
step4: cor =  $\Theta_{est} + p$ 
        pprime = tderiv( $t + h, cor$ )
        mp =  $p - h * (pprime - estprime)$ 

```

```

      pmp = dotp(p,mp)
      lambda=dotp(p,p)/pmp
step5:  $\Theta_{est} = \Theta_{est} + \textit{lambda} * p$ 
      go to step1
doneit:  $\Theta = \Theta_{est}$ 
       $\Theta' = \textit{estprime}$ 
      Print & Plot
      go to timeloop

```

This allows one to develop and debug a code using the simpler explicit integration scheme and then easily convert it to an implicit code. Notice that the only additional memory required is three (or four) vectors for r , p , mp (and $pprime$ unless you store it over mp). It is also a simple matter to create a switchable explicit - implicit code. This would also be a good tool to use in experimenting with domain decomposition in which part of the problem is solved explicitly while other parts are solved implicitly.

Numerical Example

The method has been incorporated into a modified version of the 3-D finite element heat transfer code TACO3D (1). To evaluate its performance I ran the following problem.

Consider a piece of material insulated on five sides and raised to a high temperature on the remaining side. This is really a one dimensional problem whose solution is given in (3) and is usually derived in an elementary course in partial differential equations.

The material was .5M long, with a density of 7900, specific heat of 380, and thermal conductivity of 15. The material was at an initial temperature of 300 and the surface was given a temperature of 600. The code was run with elements of a constant thickness of .025M.

The results are summarized in table 1.

fig. 1 Test Problem

	At $x = 0$		At $x = .2$	
Δt	Explicit	Implicit	Explicit	Implicit
20	315.10	315.45	336.20	336.43
50	314.70	315.58	335.84	336.43
100	-22×10^{15}	315.78	-1.45×10^{16}	336.43
200	1.8×10^{12}	316.20	4.58×10^{15}	336.43
500	300.00	317.32	300.00	336.47
exact solution	315.17		336.23	

Table 1 Solution at time = 5000

It should be noted that in the explicit calculation with $\Delta t = 500$, the instability of the explicit method is not noticeable because after ten time steps the instability had not propagated that far.

Questions To Be Answered

The method works, and experiments to date indicate that it works well even for non-linear problems. For reasonably small integration step sizes the iteration converges, however it would be useful to develop some conditions to guarantee convergence.

Most iterative methods work better with better initial estimates. Hughes (4) has introduced an "element by element" method that generates an approximate solution. It might be advantageous to use this as a starting point.

Most people recommend some sort of preconditioning with the conjugate gradient method. I have not found anything that helps this iteration, although again using Hughes' techniques might prove useful. Since the problem matrix has the form of $I = hA$, as long as h is 'small enough' this is well conditioned and simple diagonal preconditioning wouldn't be expected to do much.

This method was based on the conjugate gradient iteration, it would be interesting to see if the conjugate residual method would be any better, or any of the class of methods described by Grcar (7).

References

- [1] W. E. Mason, "TACO3D - A Three-Dimensional Finite Element Heat Transfer Code" SAND 83-8212 (1983)
- [2] A. B. Shapiro, "TOPAZ - A Finite Element Heat Conduction Code For Analyzing 2-D Solids" UCID - 20045 (1984)
- [3] J. H. VanSant "Conduction Heat Transfer Solutions" UCRL-52863 (1980)
- [4] T. J. Hughes, I. Levit, J. Winget "Element-by-Element Implicit Algorithms for Heat Conduction" *Journal of the Engineering Mechanics Division, ASCE* Vol.109 No.2, 576-585 (1983)
- [5] H. S. Carslaw, J. C. Jaeger *Conduction of Heat in Solids* Oxford Univ. Press, London (1980)
- [6] T. Belytschko "An Overview of Semidiscretization and Time Integration Procedures" in *Computational Methods for Transient Analysis* Belytschko and Hughes eds. Elsevier, Amsterdam (1983)
- [7] J. F. Grcar "Operator Coefficient Methods for Linear Equations" to appear in *SIAM Journal on Matrix Analysis Applications* also available as SAND89-8691 (1989)

INITIAL DISTRIBUTION

UNLIMITED RELEASE

N. K. Madsen, LLNL, L-156
A. C. Hindmarsh, LLNL, L-316
L. R. Petzold, LLNL, L-316
R. L. Berger, LLNL, L-156

8000 J. C. Crawford
Attn: E. E. Ives, 8100
R. J. Detry, 8200
P. L. Mattern, 8300
R. C. Wayne, 8400
P. E. Brewer, 8500

8210 J. C. Meza
8245 J. F. Grcar
8243 J. F. Lathrop (10)
8245 W. E. Mason Jr
8240 C. W. Robinson
8243 L. A. Bertram
8243 M. L. Callabresi
8245 R. J. Kee
8245 W. S. Winters Jr
8312 M. I. Baskes
8312 K. W. Mahin

8535 Publications for OSTI (10)
8535 Publications/Central Technical Files 8524-2
3141 Technical Library Processes Division (2)
8524-2 Central Technical Files (2)

END

**DATE
FILMED**

12/19/91