

UCRL-JC-132809

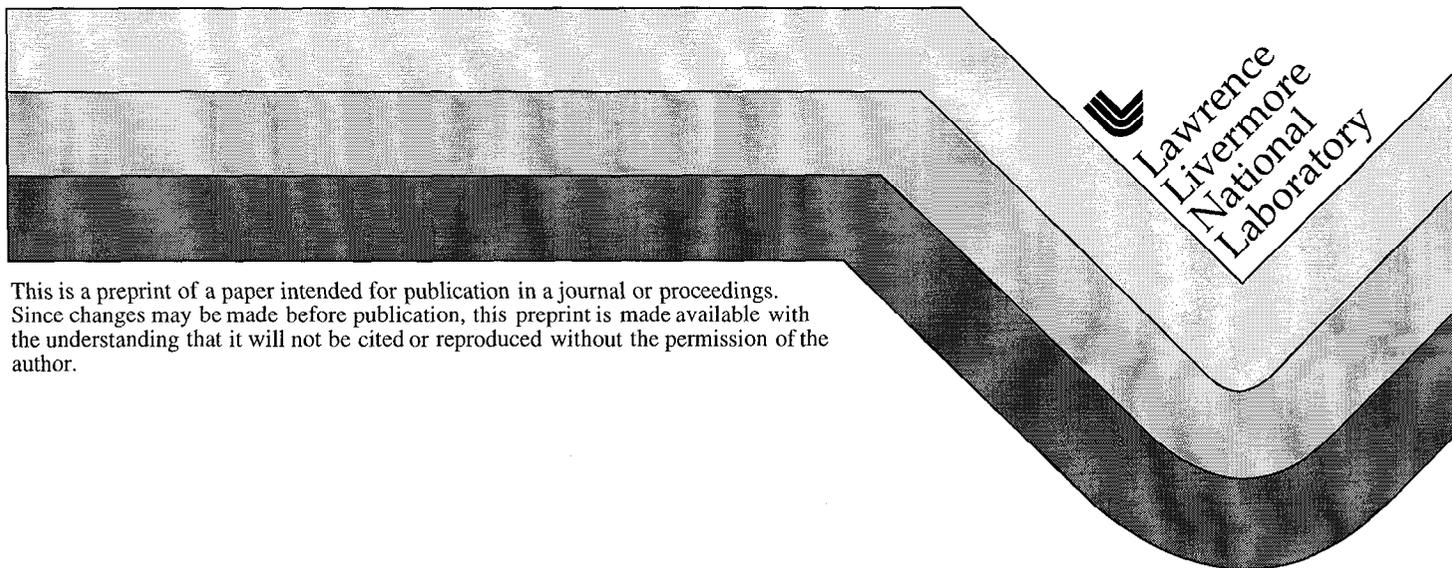
PREPRINT

# A Multigrid Strategy for Accelerating Steady-State Computations of Waves Propagating with Curvature Dependent Speeds

Jonathan Rochez

This paper was prepared for submittal to the  
*Copper Mountain Conference on Multigrid*  
*Copper Mountain, CO*  
*April 11-16, 1999*

December 16, 1998



This is a preprint of a paper intended for publication in a journal or proceedings.  
Since changes may be made before publication, this preprint is made available with  
the understanding that it will not be cited or reproduced without the permission of the  
author.

#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of 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 or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

# A Multigrid Strategy for Accelerating Steady-State Computations of Waves Propagating with Curvature Dependent Speeds

Jonathan Rochez<sup>1</sup>

**Abstract.** A multigrid strategy is developed for accelerating the steady state computations of waves propagating with curvature dependent speeds. This will allow the rapid computation of a "burn table". In a high explosive material, the creation of a burn table will allow the elimination of solving chemical reaction ODEs and feed in source terms to the reactive flow equations for solution of the system of ignition of the high explosive material. Standard iterative methods show a quick reduction of the residual followed by a slow final convergence to the solution at high iterations. Such systems are excellent choices for the use of multigrid methods to speed up convergence, even on a nonlinear system such as this. Numerical steady-state solutions to the eikonal equation on a rectangular grid are conducted. Results are presented for a square grid in 2D and a cubic grid in 3D using a Runge-Kutta time iteration for the smoothing operator until steady-state is reached.

## 1.0 Introduction.

The simulation of the detonation of a block of high explosive material involves many factors to consider. To accurately predict the outcome of a particular detonation, the conservation equations of reactive flow need to be solved. The detonation front, as it propagates through the material, is a quickly moving zone of chemical reactions. The solution of equations of state needs to be calculated for temperature and pressure terms for the conservation equations. The nature of these chemical reactions is very fast in reaction time. This in turn leads to very small time step sizes for the overall numerical simulation, which even on today's computers, can take an extremely long time to attain a solution.

To avoid the complications associated with small time steps, an infinitely thin reaction zone (i.e. a line interface) is used and a shock wave or "burn front" is allowed to propagate outwards in its normal direction to simulate the resulting explosion. Using experimental data for speeds of the wave in various media, the burn front can be propagated out from one or many detonation points and a "burn table" can be created from the resulting

---

1. Department of Applied Science, University of California at Davis and Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, California 94551 (rochez1@llnl.gov). This research was supported under the auspices of the United States Department of Energy by Lawrence Livermore National Laboratory contract W-7405-Eng-48.

simulation. The burn table is simply the time difference at which the burn front crosses a particular point in the geometry from the detonation beginning at some initial time  $t_0$ . By creating the burn table, the exact position of the burn front is then known at any time, and jumps in temperature and pressure can then be used at those locations as source terms in the conservation equations. This approach allows the solution of the differential equations associated with the chemical reactions and slow time steps to be sidestepped, yet gives a more accurate solution than if the chemistry of the problem were ignored altogether. As will be seen, the burn front propagation can be modeled by the eikonal equation.

## 2.0 Fronts Propagating with Curvature-Dependent Speed.

Let  $\Gamma(0) = x(s)$ ,  $0 \leq s \leq S$  be a simple closed curve in  $R^2$  and let  $\Gamma(t)$  be a one parameter family of surfaces generated by moving  $\Gamma(0)$  along its normal vector field with speed  $\sigma = \sigma(x, t)$ . Here,  $\sigma$  is a given scalar function. Thus, if we let  $x(s, t)$  be the position vector of the surface  $\Gamma(t)$ , then the equations of motion of the burn front is given by

$$\frac{\partial}{\partial t} x(s, t) = \sigma n(s, t) \quad (\text{EQ 1})$$

A tangent vector to the curve  $\Gamma(t)$  is given by  $[x_s, y_s]^t$ . Hence the unit normal vector to  $\Gamma(t)$  is given by  $n(s, t) = [y_s, -x_s] / (\sqrt{x_s^2 + y_s^2})$ . The equation of motion is now given by  $x_t = \sigma y_s / (\sqrt{x_s^2 + y_s^2})$   $y_t = -\sigma x_s / (\sqrt{x_s^2 + y_s^2})$ . Formally,  $x(s, t)$  is a mapping from  $[0, S] \times [0, \infty]$  to  $R^2$  generated by the moving curve.

Let  $J = x'(s, t) = \begin{bmatrix} x_s & y_s \\ x_t & y_t \end{bmatrix}$  be the Jacobian matrix of  $x$ . If  $x'(s^*, t^*)$  is non-singular, then by the inverse function theorem, there exists near  $(s^*, t^*)$  an inverse map  $\chi(x, y) : R^2 \rightarrow [0, S] \times [0, \infty]$ . That is there exists functions  $s = \psi(x, y)$ ,  $t = \phi(x, y)$ . Now by the earlier equation of motion,

$$\text{Det}[J] = x_s y_t - y_s x_t = -\sigma \sqrt{x_s^2 + y_s^2}. \quad (\text{EQ 2})$$

Looking at the chain rule shows  $\begin{bmatrix} x_s & y_s \\ x_t & y_t \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} \partial_s \\ \partial_t \end{bmatrix}$  or rearranging gives

$$\begin{bmatrix} \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} x_s & y_s \\ x_t & y_t \end{bmatrix}^{-1} \begin{bmatrix} \partial_s \\ \partial_t \end{bmatrix} = \frac{1}{\text{Det}[J]} \begin{bmatrix} y_t & -y_s \\ -x_t & x_s \end{bmatrix} \begin{bmatrix} \partial_s \\ \partial_t \end{bmatrix}$$

It follows that

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \frac{1}{\text{Det}[J]} \begin{bmatrix} y_t & -y_s \\ -x_t & x_s \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial s} \\ \frac{\partial \phi}{\partial t} \end{bmatrix} = \frac{1}{\text{Det}[J]} \begin{bmatrix} y_t & -y_s \\ -x_t & x_s \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial s} \\ \frac{\partial \phi}{\partial t} \end{bmatrix} = \frac{1}{\text{Det}[J]} \begin{bmatrix} y_t & -y_s \\ -x_t & x_s \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\text{Det}[J]} \begin{bmatrix} -y_s \\ x_s \end{bmatrix}$$

Using this result with the value for  $\text{Det}[J]$  shown in EQ 2, the classical eikonal equation is obtained :

$$\phi_x^2 + \phi_y^2 = \frac{1}{(\text{Det}[J])^2} (y_s^2 + x_s^2) = \frac{1}{\sigma^2} \quad (\text{EQ 3})$$

A common way of solving EQ 3 is to introduce a time variable and solve the equation

$$\phi_t + \sigma |\nabla \phi| = 1 \quad (\text{EQ 4})$$

to steady state. If this equation is time-stepped until steady-state is achieved (i.e.  $\phi_t = 0$ ), then the solution to EQ 3 is obtained. The front propagation speed,  $\sigma$ , can be a function of position, curvature, etc. If desired to include curvature, a common profile that is used is  $\sigma = \sigma_o (1 - \varepsilon \kappa)$ , where  $\kappa$  is the curvature defined by  $\kappa = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$  and  $\varepsilon$  is a constant usually  $\ll 1.0$ .

### 3.0 Multigrid/Runge-Kutta Method

A nonlinear system of equations

$$Q(u) = f \quad (\text{EQ 5})$$

is obtained when EQ 3 is discretized. EQ 4 then yields a system of ordinary differential equations which can be integrated to steady state by a Runge-Kutta integration scheme. The Runge-Kutta method can be regarded as an iterative method for solving the nonlinear system of equations in EQ 5 where the time-step is an acceleration parameter. In this way, the method can be further accelerated to steady-state by multigrid. This idea was used to accelerate convergence to steady-state solutions of shock problems in [2].

Specifically, the algorithm for the fine level at the  $n^{\text{th}}$  iteration is

$$\begin{aligned}
u^{(i)} &= u^{(0)} - \alpha_i \Delta t_0 (Q_0(u^{(i-1)}) - f_0), i = 1, \dots, m \\
\bar{u} &= u^{(m)} \\
u^{(0)} &= u^n
\end{aligned}$$

Here,  $\alpha_i$  ( $i = 1, \dots, m$ ) and  $m$  are constants which can be manipulated to gain better convergence. Using some restriction operator to bring the solution from the fine grid to the coarse grid,  $R_u$ , and a restriction operator for the residual from the fine grid to the coarse grid,  $R_r$ , the coarse grid operations are defined by

$$\begin{aligned}
v^{(0)} &= R_u(\bar{u}) \\
f_1 &= Q_1(v^{(0)}) - R_r(Q(\bar{u}) - f_0) \\
v^{(i)} &= v^{(0)} - \alpha_i \Delta t_1 (Q_1(v^{(i-1)}) - f_1), i = 1, \dots, m \\
\tilde{v} &= v^{(m)}
\end{aligned} \tag{EQ 6}$$

$Q_1$  is the nonlinear operator that operates on the coarse grid while  $f_1$  is the right-hand side of the coarse grid equation. The correction calculated by the coarse grid operation is

$$u^{n+1} = \bar{u} + P(\tilde{v} - v^{(0)})$$

Here,  $P$  is a suitable prolongation operator to take the solution from the coarse grid back to the fine grid.

The above algorithm can be applied recursively for additional grid levels. Application to the third grid level could be viewed as a correction to the correction to the original iteration. The only note to take care of in additional levels is the prolongation step difference which will be the correction between the original iteration at a particular level minus the final iteration result at the same level. For example, if a third grid were being computed, the additional coarsening step would be as follows

$$\begin{aligned}
w^{(0)} &= R_u(\tilde{v}) \\
f_2 &= Q_2(w^{(0)}) - R_r(Q_1(\tilde{v}) - f_1) \\
w^{(i)} &= w^{(0)} - \alpha_i \Delta t_2 (Q_2(w^{(i-1)}) - f_2), i = 1, \dots, m \\
\tilde{w} &= w^{(m)}
\end{aligned}$$

The prolongation step back to the first coarse level would then be

$$vtemp^{n+1} = \tilde{v} + P(\tilde{w} - w^{(0)})$$

A smoothing iteration would then occur as in EQ with  $v^{(0)}$  replaced with  $vtemp$ . The final output after this operation,  $\tilde{v}$ , would then be used with the original estimation for the solution on the coarse grid,  $v^{(0)}$ , to yield

$$u^{n+1} = \bar{u} + P(\tilde{v} - v^{(0)})$$

The restriction operators and prolongation operator should be chosen suitably for the particular discretized domain that is being used. For restriction, weighted averages of neighboring points is often acceptable. Conversely, for prolongation, it is common to inject the solution for a similar point and create weighted diminished injections for neighboring points.

## 4.0 Single Point Detonation

To see how well the proposed algorithm would speed up the solution to the eikonal equation, the system was first computed on a 2D square cartesian grid. A single detonation point was introduced into the system at the origin (bottom left of the grid) and allowed to propagate outward until steady-state was reached with unit speed and no curvature initially. The grid size in both  $x$  and  $y$  directions is a constant,  $h$ , and the total length of a side is 1. Zero Neumann boundary conditions were specified. This setup was chosen for its simplicity and the analytic solution is well known. Calling the unknown burn time (the time at which the burn front crosses a particular point in the domain since ignition), the solution is  $\phi = \sqrt{x^2 + y^2}$ .

On a square grid, the restriction operators have been set at the following :

$$R_{i,j} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

The following equation above reads as

$$v_{i,j} = \frac{1}{4}u_{i,j} + \frac{1}{8}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) \\ + \frac{1}{16}(u_{i+1,j+1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i-1,j-1})$$

where from Section 3.0 above,  $u$  is the solution on the fine grid and  $v$  is the solution on the coarse grid. For the problem here, the restriction of the solution and the restriction of the residual are both using this same operator ( $R_u = R_r$ ).

The prolongation operator that is chosen is essentially the inverse operation of the restrictions with an additional factor of 4. For like grid points on the fine and coarse grid, the value, as is seen in the operator, is simply injected in.

$$P = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

Reading the above equation would give, for example, that  $u_{i,j+1} = \frac{1}{2}v_{i,j}$  for the data value in position (1,2).

To compare how well this multigrid scheme performs, a simple finite differencing scheme is also run on the same problem where derivative terms in the eikonal equation are approximated by finite differences. The time derivative term uses a forward differencing scheme while the first derivative term uses a backward one. For various size problems, a direct clock time of the algorithms can be compared to show any speed-ups. The time step used for both algorithms is set at  $\Delta t = \Delta x^2/4$ . This time step is suboptimal for both cases, yet the purpose here is simply to choose a similar time step for comparison and one that converges. The actual optimal time step that still yields a stable solution is difficult to calculate due to the nonlinearity of the problem.

## 5.0 Numerical Results.

To check the speed-up for the algorithm over a simple finite differencing approximation, with like time steps, wall clock time is compared for both algorithms. Table 1 shows the 2-D time in seconds until steady-state is reached of the regular finite difference algorithm on various sizes of problems,  $n$  (the number of grid points in 1 direction). The stopping criterion used was a mean squared error of  $1.0e^{-10}$ . Table 2 shows the corresponding data for the multigrid algorithm presented for various levels of  $v$ -cycles. For example, 3 levels would be  $v$ -cycles calculating the finest grid and using the next two coarsest grids in the multigrid algorithm. Obviously due to some problem sizes, certain levels could not be obtained due to too few grid points.

**TABLE 1. 2-D Finite Differencing Computation Time to Steady-State**

<b>n</b>	<b>1 Level</b>
5	0.17
9	1.37
17	9.45
33	102.21
65	1291.29
129	18893.55

**TABLE 2. 2-D Multigrid Computation Time to Steady-State**

<b>n</b>	<b>1 Level</b>	<b>2 Levels</b>	<b>3 Levels</b>	<b>4 Levels</b>	<b>5 Levels</b>	<b>6 Levels</b>	<b>7 Levels</b>
5	0.06	0.05	---	---	---	---	---
9	0.49	0.61	0.66	---	---	---	---
17	4.94	4.28	4.83	5.00	---	---	---
33	52.51	34.17	29.55	45.48	29.99	---	---
65	623.49	429.95	388.92	376.22	442.91	585.76	---
129	9221.45	6375.44	5589.67	5499.83	5291.56	6875.44	9924.37

Speed-ups of factors of 4 are seen from some of the larger size problems. 1 level computation times for the multigrid algorithm shown throughout Table 2 are less than the comparable finite differencing scheme used for a same size problem. This is due to the fact that the multigrid Runge-Kutta algorithm employed uses a scaling factor,  $\alpha$ , which yields the difference in the two algorithms and accounts for the speed-up for the 1 level method. To take advantage of this speed-up and the quick elimination of the high frequency error components, the best level for each size is taken and several function evaluations at each level are performed (i.e.  $m$  is varied when  $i = 1, \dots, m$  in our multigrid algorithm). The results show even greater speed-ups and are presented in Table 3.

**TABLE 3. 2-D Computation Time for Various Number of Iterations at Select Multigrid Level**

<b>m</b>	<b>n = 17 2 Levels</b>	<b>n = 33 3 Levels</b>	<b>n = 65 4 Levels</b>	<b>n = 129 5 Levels</b>
2	3.55	24.56	179.27	3124.55
3	2.87	19.41	135.49	2497.02
4	2.29	14.43	91.84	1998.33
5	1.75	10.56	62.76	1523.49
6	1.38	7.53	41.21	884.77
7	0.96	5.35	25.90	314.29
8	0.90	6.42	---	---
9	0.64	---	---	---

From comparing values from Table 3 with those in Table 1, the true speed-up of the multigrid algorithm for this test problem on various levels is seen. For 129 x 129 size problems, the greatest speed-up is seen. The finite difference method took 18,893.55 seconds to reach steady-state while the multigrid algorithm took only 314.29, a factor of 60.

The same problem after being run in 3-D shows similar results in lessened computation time. Table 4 shows the time to reach steady state for the finite difference algorithm on the unit cube in 3-D, and Table 5 shows the corresponding times for Multigrid.

**TABLE 4. 3-D Finite Differencing Computation Time to Steady-State**

<b>n</b>	<b>1 Level</b>
5	2.69
9	73.27
17	2899.19
33	66722.40

**TABLE 5. 3-D Multigrid Computation Time to Steady-State**

<b>n</b>	<b>1 Level</b>	<b>2 Levels</b>	<b>3 Levels</b>	<b>4 Levels</b>	<b>5 Levels</b>
5	1.92	1.92	---	---	---
9	48.12	43.50	46.90	---	---
17	1952.54	1527.44	1603.89	2250.51	---
33	40295.36	32787.66	29550.67	30678.11	45998.03

As seen from comparing the two tables, speed-ups of factors of 4 are observed again. Table 6 shows similar data as Table 3, but in the corresponding 3-D case.

**TABLE 6. 3-D Computation Time for Various Number of Iterations at Select Multigrid Level**

<b>m</b>	<b>n = 17 2 Levels</b>	<b>n = 33 3 Levels</b>
2	2545.88	22590.46
3	2014.15	16877.74
4	1650.98	12491.57
5	1304.61	8829.55
6	956.42	5526.37
7	500.52	2249.15
8	200.49	1023.49
9	51.07	---

For the best case in a size  $n = 33$  problem, a speed-up of a factor of roughly 65 is achieved reducing the original time of 66,72.40 seconds down to merely 1,023.49 seconds.

## 6.0 Conclusions

A multigrid strategy was developed to solve the eikonal equation on orthogonal grids. Speed up factors of 65 were obtained for a problem with a single point initial value. These encouraging results motivate the extension of this algorithm to the eikonal equation on unstructured hexahedral grids as well as to other Hamilton-Jacobi type equations.

## 7.0 Acknowledgments

The author would like to thank the Institute for Scientific Computing Research (ISCR) at the Lawrence Livermore National Laboratory (LLNL) for support of this research.

## 8.0 References

- [1] L. Ferm and P. Lotstedt, *Two-Grid Solution of Shock Problems*, SIAM J. Sci. Comput., Vol. 18, No. 6, pp1533-1552, November 1997.
- [2] A. Jameson, W. Schmidt, and E. Turkel, *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*, AIAA paper 81-1259, 1981.
- [3] W. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, Pennsylvania 1987.
- [4] G. Kreiss and H. Kreiss, *Convergence to Steady-State of Solutions of Burgers' Equation*, Appl. Numer. Math., 2 (1986), pp. 161-179.
- [5] S. Osher and F. Solomon, *Upwind Schemes for Hyperbolic Systems of Conservation Laws*, Math. Comp., 38 (1982), pp. 339-377.
- [6] R. LeVeque, *Numerical Methods for Conservation Laws*, 2<sup>nd</sup> Edition, Birkhauser Verlag, Berlin, 1992.
- [7] J. Sethian, *Level Set Methods*, Cambridge University Press, New York, NY, 1996.