Numerical Computation of Three Dimensional Detonation Waves on Parallel Computers

by

Björn Sjögreen

Report No. 162/1994

UPPSALA UNIVERSITET
Institutionen för
teknisk databehandling

UPPSALA UNIVERSITY
Department of
Scientific Computing

Uppsala University
Department of Scientific Computing
Report No. 162
July 1994

Abstract The inviscid Euler equations for compressible fluid flow in three space dimensions are solved. One chemical reaction is taking place inside the fluid. A detonation wave solution, propagating in a square channel is considered. The equations are solved using a modern shock capturing difference method on a parallel computer. A new class of split Runge-Kutta methods for time integration are introduced. Computed solutions in two and three space dimensions are presented. The implementation on a parallel computer is discussed.

1. Introduction

The simulation of flows with chemical reactions is of importance in many applications in hypersonic aerodynamics. Combustion problems are described by systems of partial differential equations containing a complicated reaction mechanism. The complexity of the equations implies that there are many aspects on the numerical solution of such problems. One particular problem is the simulation of a detonation wave i.e., a shock wave followed by a thin zone of reaction. In order to obtain a reasonably simple model to investigate with respect to numerical effects, we neglect diffusion and simplify the reaction mechanism to a single one step reaction $R \rightarrow P$. From such a model we obtain the well known ZND detonation waves, which have been analyzed in e.g., [2].

It is well known that the thin reaction zone in the detonation wave must be well resolved numerically. This requires a large number of grid points and consequently it is necessary to use very fast computers.

In the early days of parallel computing, parallel computers were experimental machines with poor software and fairly weak processors. Parallel computers are becoming more mature every year, and today we have reached the situation where they are fully competitive with conventional one processor machines, both in terms of performance and in terms of software support. It is natural to use parallel computers for the computation of detonation waves, not in order to do research on parallel algorithms, but just because we can solve our problem in shorter time on such a machine.

In this paper we generalize our previous work [3] to three space dimensions. We describe the implementation on a parallel computer. We develop and analyze new Runge-Kutta time discretization methods which permit unconditional stability for the source term, without using operator splitting.

2. Equations and Spatial Discretization

The inviscid Euler equations of gas dynamics in three space dimensions with one chemical reaction are

$$\begin{pmatrix} \rho \\ m \\ n \\ o \\ e \\ \rho z \end{pmatrix}_t + \begin{pmatrix} m \\ m^2/\rho + p \\ mn/\rho \\ mo/\rho \\ m(e+p)/\rho \\ mz \end{pmatrix}_x + \begin{pmatrix} n \\ mn/\rho \\ n^2/\rho + p \\ no/\rho \\ n(e+p)/\rho \\ nz \end{pmatrix}_y + \begin{pmatrix} o \\ mo/\rho \\ no/\rho \\ o^2/\rho + p \\ o(e+p)/\rho \\ oz \end{pmatrix}_z = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -K\rho z e^{-T_i/T} \end{pmatrix}. \tag{2.1}$$

These equations are derived from the conservation of mass, momentum and energy, and from the equation for the chemical reaction rate. Initial data are added to the problem. The dependent variables $(\rho, m, n, o, e, \rho z)$ are functions of (t, x, y, z), and the additional variables p, T are found from the relations

$$\begin{aligned} p &= (\gamma - 1)(e - \frac{1}{2\rho}(m^2 + n^2 + o^2) - q_0 \rho z) \\ T &= p/\rho \end{aligned}.$$

The constants γ, K, T_i, q_0 are assumed to be given.

This problem is a partial differential equation on the form

$$u_t + f_1(u)_x + f_2(u)_y + f_3(u)_z = s(u).$$

In order to solve (2.1) numerically, we make the following discretization of the problem. We first introduce the grid points

$$\{(x_i, y_j, z_k), i, j, k \in \ldots, -2, -1, 0, 1, 2, \ldots\}$$

in $\mathbf{R}^3$. The approximate solution at (x_i, y_j, z_k) at time t is denoted $u(t)_{i,j,k}$, and the step sizes $\Delta x = x_{i+1} - x_i, \Delta y = y_{j+1} - y_j, \Delta z = z_{k+1} - z_k$ are constants. We approximate the equation (2.1) semi discretely by

$$\frac{du(t)_{i,j,k}}{dt} = -\Delta_{+i} f_{i-1/2,j,k}/\Delta x - \Delta_{+j} g_{i,j-1/2,k}/\Delta y - \Delta_{+k} h_{i,j,k-1/2}/\Delta z + s(u_{i,j,k})$$

where $\Delta_{+i} u_{i,j,k} = u_{i+1,j,k} - u_{i,j,k}$, and similarly for the other dimensions. Here, $\Delta_{+i} f_{i-1/2,j,k}/\Delta x$ is an approximation of $f_1(u)_x$, with $f_{i-1/2,j,k}$ a numerical flux function. The y and z-derivatives are approximated similarly. Discontinuities will occur in the solution to this problem. We have therefore chosen the numerical flux functions corresponding to a modern shock capturing method. More specifically, we have used Roe's method, but with the accuracy increased to two by a piecewise linear reconstruction, using slope limiters. We describe this method in one space dimension. Roe's method consists of approximating the flux derivative $f(u)_x$ at x_j by

$$(h(u_{j+1}, u_j) - h(u_j, u_{j-1}))/\Delta x.$$

The numerical flux function $h_{j-1/2} = h(u_j, u_{j-1})$ is given by

$$h_{j-1/2} = \frac{1}{2}(f(u_j) + f(u_{j-1})) - \frac{1}{2}\sum_{k=1}^{m} |a^k_{j-1/2}|\alpha^k_{j-1/2}\mathbf{r}^k_{j-1/2} \tag{2.2}$$

$a^k_{j-1/2}$ and $\mathbf{r}^k_{j-1/2}$ are the eigenvectors and eigenvalues of a matrix $A_{j-1/2}$ satisfying $\Delta_- f(u_j) = A_{j-1/2}\Delta_- u_j$. $\alpha^k_{j-1/2}$ are the representation of $\Delta_- u_j$ in the eigenbasis i.e.,

$$\Delta_- u_j = \sum_{k=1}^{m} \alpha^k_{j-1/2}\mathbf{r}^k_{j-1/2}$$

The eigenvalues and eigenvectors of the equations (2.1) are given in Appendix A. This method is first order accurate. Second order accuracy is obtained by making the piecewise linear reconstruction

$$u(x) = u_j + s_j(x - x_j)/\Delta x \qquad x_{j-1/2} < x < x_{j+1/2}$$

within each computational cell $[x_{j-1/2}, x_{j+1/2}]$. The slopes are limited in some way in order to obtain good shock resolution. Two commonly used limiters are

$$s_j = \mathrm{minmod}(\Delta_+ u_j, \Delta_- u_j)$$

and

$$s_j = \begin{cases} 2\Delta_+ u_j \Delta_- u_j/(\Delta_+ u_j + \Delta_- u_j) & \text{if } \Delta_+ u_j \Delta_- u_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

The second order approximation of $f(u)_x$ is

$$(h(u_{j+1} - s_{j+1}/2, u_j + s_j/2) - h(u_j - s_j/2, u_{j-1} + s_{j-1}/2))/\Delta x.$$

If this second order method is applied to the equation

$$u_t + f(u)_x = 0, \tag{2.3}$$

we obtain, with forward Euler time discretization

$$u_j^{n+1} = u_j^n - \lambda\Delta_+ h(u_j - s_j/2, u_{j-1} + s_{j-1}/2) \tag{2.4}$$

where $\lambda = \Delta t/\Delta x$. If (2.3) is a scalar equation, it is well known that (2.4) has the TVD property, which guarantees that there will be no unphysical oscillations near discontinuities in the solution.

3. Split Runge-Kutta Methods

We now have a spatial discretization of TVD type. We next make the time discretization, using a Runge-Kutta method. For simplicity the formulas in this section are given for one space dimension. The extension to three dimensions is straightforward. In [5] a family of so called TVD Runge-Kutta methods were developed. These are methods such that if the semi discrete method is TVD, the time discretized method is TVD too. If we are given the semi discrete problem

$$\frac{du_j(t)}{dt} = -\Delta_+ h_{j-1/2}/\Delta x$$

approximating

$$u_t + f(u)_x = 0$$

then the second order accurate method in the TVD Runge-Kutta family is

$$\begin{aligned} u^{(1)} &= u^n - \lambda \Delta_+ h^n_{j-1/2} \\ u^{(2)} &= u^{(1)} - \lambda \Delta_+ h^{(1)}_{j-1/2} \\ u^{n+1} &= (u^n + u^{(2)})/2 \end{aligned} \tag{3.1}$$

with $\lambda = \Delta t/\Delta x$. The third order method is

$$\begin{aligned} u^{(1)} &= u^n - \lambda \Delta_+ h^n_{j-1/2} \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}(u^{(1)} - \lambda \Delta_+ h^{(1)}_{j-1/2}) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}(u^{(2)} - \lambda \Delta_+ h^{(2)}_{j-1/2}) \end{aligned} \tag{3.2}$$

These are explicit methods, and they are stable under some restriction on the time step. It turns out that both (3.1) and (3.2) are TVD under the same CFL condition as obtained with the forward Euler time discretization [5].

Assume now that the source term is added to the problem. We then have a semi discrete equation on the form

$$\frac{du_j(t)}{dt} = -\Delta_+ h_{j-1/2}/\Delta x + s(u_j). \tag{3.3}$$

The most straightforward time discretization would be to apply (3.1) or (3.2) to (3.3) directly by putting the source term at the same stage as the convective term. However, there is a danger that the source term will impose further restrictions on the time step due to stability constraints. This is the case when K is very large, which sometimes occurs in practical problems.

The most commonly used approach to source term discretization, is to use operator splitting. Operator splitting limits the order of accuracy to two. We will therefore consider split Runge-Kutta methods, which we obtain from approximating the ODE

$$y_t = f_1(y) + f_2(y) \tag{3.4}$$

by

$$
\begin{aligned}
y^{(0)} &= y_n \\
y^{(k)} &= y^{(0)} + \Delta t \sum_{j=0}^{k-1} a_{k,j} f_1(y^{(j)}) + \Delta t \sum_{j=0}^{k} b_{k,j} f_2(y^{(j)}) \quad k = 1, \ldots, m \\
y_{n+1} &= y^{(m)}
\end{aligned}
\tag{3.5}
$$

In this m stage method, we assume that we can afford to solve for $f_2(y)$ implicitly, but not for $f_1(y)$.

These methods have the advantage that higher order than two is possible, and it incorporates the source term into the Runge-Kutta framework in a straightforward way.

We write (3.5) as

$$y_{n+1} = F(y_n)$$

and we define the order of accuracy as usual.

Definition 3.1. *The method (3.5) has order of accuracy p if*

$$y(t + \Delta t) - F(y(t)) = O(\Delta t^{p+1})$$

where $y(t)$ is a solution of (3.4).

To define stability, we apply the method (3.5) to the linear equation

$$y_t = \lambda_1 y + \lambda_2 y. \tag{3.6}$$

The method can then be written

$$y_{n+1} = p(z_1, z_2) y_n$$

with $z_1 = \Delta t \lambda_1$ and $z_2 = \Delta t \lambda_2$, and p some rational function of z_1 and z_2.

Definition 3.2. *The domain of stability for the method (3.5) is the set*

$$\{(z_1, z_2) : |p(z_1, z_2)| \leq 1\}.$$

If all z_2 with $\Re(z_2) \leq 0$ are inside the stability domain when $z_1 \in \Omega$, for some domain Ω, we say that the method is unconditionally stable in z_2 when $z_1 \in \Omega$.

We consider the methods (3.1) and (3.2) with source term added. First we add a source term, $g(u)$, to the second order accurate (3.1) and obtain

$$
\begin{aligned}
u^{(1)} &= u^n - \lambda \Delta_+ h^n_{j-1/2} + a \Delta t g(u^n) + b \Delta t g(u^{(1)}) \\
u^{n+1} &= (u^{(1)} + u^n)/2 - \lambda \Delta_+ h^{(1)}_{j-1/2}/2 + c \Delta t g(u^{(1)}) + d \Delta t g(u^{n+1})
\end{aligned}
\tag{3.7}
$$

We next determine the coefficients a, b, c, d such that we obtain unconditional stability in z_2. This can be done, as seen from the following theorem

Theorem 3.1. *When we impose conditions for second order accuracy on the method (3.7) we obtain the following two families of methods*

$$\begin{aligned} &a = 1-2d \quad b=0 \quad c=1/2 \quad d\neq 0\\ &a=1 \quad b=0 \quad c=1/2-d \end{aligned}$$

with d as parameter. There are only two solutions which give unconditional stability in z_2, they are obtained for $d=1/2$ i.e.,

$$\begin{aligned} &a=0 \quad b=0 \quad c=1/2 \quad d=1/2\\ &a=1 \quad b=0 \quad c=0 \quad d=1/2 \end{aligned}$$

The unconditional stability in z_2 is valid if the following CFL *condition in z_1 holds*

$$|1+z_1|+|z_1^2/2|\leq 1 \tag{3.8}$$

Remark: (3.8) gives a non empty domain of stability as seen in Fig. 3.1.

Proof: The second order accurate family is found by straightforward Taylor expansion. It is standard and we do not give it here. Next, when the second order method is applied to the test problem (3.6), we obtain the amplification factor

$$p(z_1,z_2)=\frac{1+z_1+(1-d)z_2+z_1^2/2+(1-d)z_1z_2+(1/2-d)z_2^2}{1-dz_2}$$

The only possibility is to take $d=1/2$, otherwise the z_2^2 term in the numerator will increase quicker than the denominator as $|z_2|\to\infty$ in the left hand plane.

For $d=1/2$, we can write

$$p=\frac{(1+z_1)(1+z_2/2)}{1-z_2/2}+\frac{z_1^2/2}{1-z_2/2}$$

which, by using the stability for the trapezoidal rule and the backward Euler methods, can be estimated

$$|p|\leq|1+z_1|\frac{|1+z_2/2|}{|1-z_2/2|}+\frac{|z_1^2|/2}{|1-z_2/2|}\leq|1+z_1|+|z_1^2|/2$$

and the stability result follows.

We next carry out the same analysis for the third order method

$$\begin{aligned} u^{(1)} &= u^n-\lambda\Delta_+h^n_{j-1/2}+a\Delta t g(u^n)+b\Delta t g(u^{(1)})\\ u^{(2)} &= \frac{3}{4}u^n+\frac{1}{4}(u^{(1)}-\lambda\Delta_+h^{(1)}_{j-1/2})+c\Delta t g(u^{(1)})+d\Delta t g(u^{(2)})\\ u^{n+1} &= \frac{1}{3}u^n+\frac{2}{3}(u^{(2)}-\lambda\Delta_+h^{(2)}_{j-1/2})+e\Delta t g(u^{(2)})+f\Delta t g(u^{n+1}) \end{aligned} \tag{3.9}$$

As previously, we determine a,b,c,d,e,f, such that third order and unconditional stability in z_2 is obtained.

Theorem 3.2. *When we impose conditions for third order accuracy on the method (3.9) we obtain the following solution*

$$a = 1 \quad b = 0 \quad c = 1/4 - 3f/2 \quad d = 3f/2 \quad e = 2/3 - f$$

with f as parameter. There is only one solution which gives unconditional stability in z_2 under a CFL *condition in z_1, obtained for $f = 2/3$ i.e.,*

$$a = 1 \quad b = 0 \quad c = -3/4 \quad d = 1 \quad e = 0 \quad f = 2/3$$

The unconditional stability in z_2 is valid if the following CFL *condition in z_1 holds*

$$|1 + z_1| + |z_1^2/2| + |z_1^3/6| \leq 1 \tag{3.10}$$

Remark: The stability domain (3.10) is non empty as seen in Fig. 3.2.

Proof: The proof is similar to that of Theorem 3.1. The third order family is obtained from straightforward Taylor expansion. Next, we apply the method to the problem (3.6), and as in Theorem 3.1 we find, by inspection of the polynomial p, that $f = 2/3$ is necessary for unconditional stability in z_2. After $f = 2/3$ has been inserted p becomes

$$p = (1 + z_1)\frac{1 - 2/3z_2 - z_2^2/2}{(1 - z_2)(1 - 2z_2/3)} + \frac{z_1/2}{1 - z_2} + \frac{z_1^3/6}{(1 - z_2)(1 - 2z_2/3)}$$

The inequality

$$\frac{1}{|1 - az|} \leq 1 \quad a \geq 0, \Re(z) \leq 0$$

is used on the second and third terms. For the first term, we use

$$|\frac{1 - 2/3z_2 - z_2^2/2}{(1 - z_2)(1 - 2z_2/3)}| \leq 1 \quad \Re(z_2) \leq 0. \tag{3.11}$$

This is easily proved by considering the contour in Fig. 3.3. It is straightforward by elementary calculus to see that the quotient (3.11) is ≤ 1 on this contour for the radius, R, sufficiently large. Since the poles are in the right hand plane, (3.11) follows from the maximum principle for analytic functions. Finally, we obtain

$$|p| \leq |1 + z_1| + |z_1^2/2| + |z_1^3/6|$$

from which (3.10) follows.

We have here obtained a split method of order of accuracy three. In standard operator splitting, the highest possible order is two.

In the computations in Section 5, we have used the second order Runge-Kutta method (3.7), with $d = 1/2$ as in Theorem 3.1. This choice was made because the spatial accuracy was two. If we had implemented a third order ENO spatial discretization, we would have used the third order time integration method.

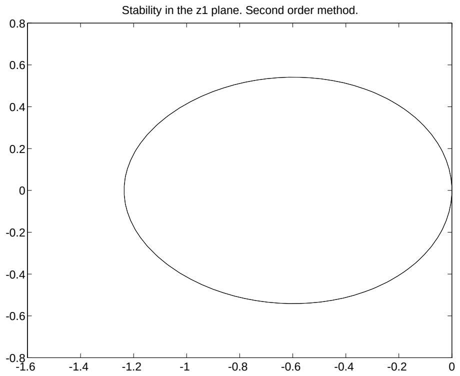

FIG. 3.1. *Stability domain (3.8).*

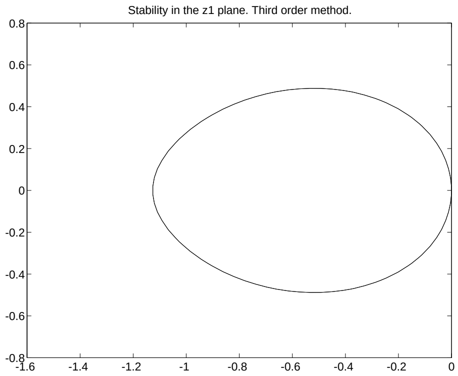

FIG. 3.2. *Stability domain (3.10).*

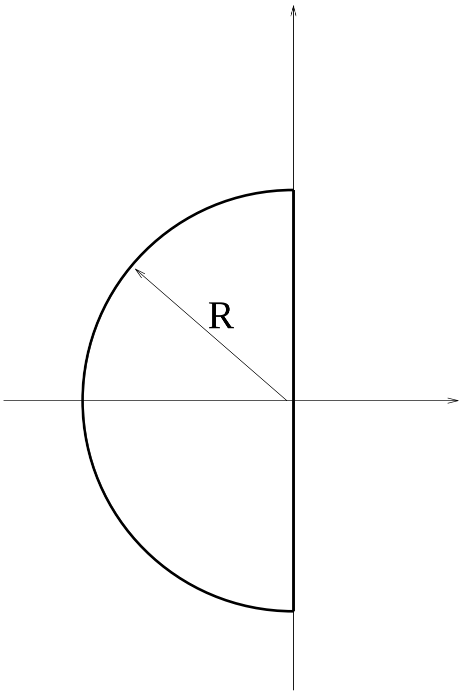

FIG. 3.3. *Contour in complex plane used to prove (3.11).*

4. Implementation on Parallel Computers

We have implemented the method on several distributed memory parallel computers, using message passing communication routines. An original version was implemented on an Intel iPSC/2 hypercube computer some years ago. The performance figures presented below show a dramatic improvement in computational speed compared with this old machine. The iPSC/2 is included in this comparison to illustrate the development of parallel computers in recent years.

The computational domain was divided into a number of subcubes and each subcube was assigned to one processor. For the older hypercube computers, it was important to map the cube into a three dimensional grid by a Grey code mapping which minimizes the distance between neighbors in the cube. For modern parallel computers it is not necessary to worry about this. Communication is fast enough between all nodes.

For reference we list the formulas used for the subdivision of the domain. Consider a one dimensional index domain $l \leq i \leq n$.

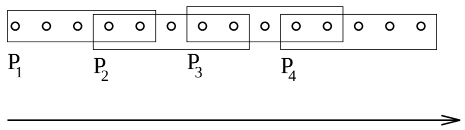

FIG. 4.1. *Distribution of a one dimensional domain into 4 processors.*

Divide the domain into p pieces, one for each processor. Let s be the number of points in one piece, and let a be the amount of overlap between neighbors, the total number of grid points is $N = n - b + 1$. In Fig. 4.1, $p = 4, a = 2, s = 5, l = 1, N = 14$. We have the following formula

$$\frac{N-a}{p} + a = s$$

The required overlap for a difference method with a $2q + 1$ point stencil is

$$a = 2q$$

In the computer program $N, 2q + 1$, and p are given by the user, the program adjusts N so that p divides $N - a$, and compute s. In this way the number of grid points will be equal in all processors. The formula for translation between the local index in the piece k, $1 \leq i_k \leq s$ and the global index $l \leq i \leq n$ is

$$i = i_k + (k-1)(s-a)$$

Given a program on a scalar computer, the work to port it to a distributed memory computer consist of a) new boundary conditions and b) reorganize some of the I/O through a host processor or to change the way input and output routines adress files. The rest of the code, solver in the interior, time marching, etc. can be kept unchanged. The largest part of the porting work is under point b) above.

To the existing boundary condition we add a new communication boundary condition. It consists of the following steps, with i local index in the processor

1. send $q + 1 \leq i \leq 2q$ to the left
2. send $s - 2q + 1 \leq i \leq s - q$ to the right

3. receive $1 \le i \le q$ from the left
4. receive $s - q + 1 \le i \le s$ from the right

For the processors 1 and p, which do not have neighbors to the left and right respectively, the original boundary conditions are used in place of some of the points 1–4 above.

Table 4.1 shows some typical Mflop rates attained on various machines. All computations were made in 64-bit precision arithmetic, using the programming language FORTRAN77, except for the Connection Machine CM200, where program was rewritten in CM-Fortran, and where the computation was made with only two space dimensions.

TABLE 4.1
Mflop rates for various parallel computers.

Computer	Number of Processors	Mflop Rate
Intel iPSC/2	8	2
IBM SP1	8	125
Intel Paragon	8	42
Cray T3D	8	96
CM200	4096	96
Sun SPARC 10	8	50

For all the other machines the code was unchanged except for the calls to communication subroutines.

The IBM/SP1 consists of RS6000 processors, connected in a switched network. We used the machine specific communication library MPL.

For the Intel Paragon we used exactly the same communication subroutine calls as for the Intel iPSC/2.

We also run the code on a cluster of Sun SPARCstation 10/40 computers with the communication library PVM.

The performance figures are given to serve as a guidance for what is reasonable to expect from these modern parallel computers. They can, of course, not be understood as a complete evaluation of the machines. We will give details about implementation and performance in a forthcoming report, where we also will include additional machines.

The time to do communication to impose boundary conditions etc. was, as usual with explicit difference methods, neglectable ($< 5\%$) compared with the time to do arithmetic work [6]. The only exeption was the PVM library on the workstation cluster, where communication overhead can sometimes be considerably larger than on specialized parallel machines. Nevertheless, the computational speed is far from the peak performance given by the vendors. The main cause of this performance degradation can be found in the one processor performance.

One source of trouble is that on some machines single precision arithmetic is faster than double precision arithmetic. The peak performance is given in single, but we are using double in the program. A second source of trouble is that the maximum speed is obtained only for very long operations, where intermediate result can be kept in fast cache memory. Our computation was not organized to profit from long operations, loop unrolling, etc. Performance figures for LINPACK benchmarks, show usually higher flop rates since LINPACK consists of purer vector operations.

Experiments on one processor on the Intel hypercube, and a Sun workstation showed for double precision the Mflop rates given in Table 4.2. The arithmetic expressions are inside a long loop over i.

TABLE 4.2
Mflop rates inside a long loop.

Computer	$x_i + y_i$	$x_i + a * y_i$	$a_1 x_{i-3} + a_2 x_{i-2} + \ldots$ (21 terms)
iPSC/2	0.17	0.28	0.60
Sun SPARC10	7.0	10	23

Our code is thus placed near the triad operation. This seems to be the speed one can expect in most practical computations which are arithmetically intense, but where the arithmetic is not structured into nice vector operations.

5. Results

In our computations, we choose a problem where the solution consists of a detonation front, propagating in a square channel with cross section dimensions 0.5×0.5. The computational domain is

$$0 \le x \le 1 \quad 0 \le y \le 0.5 \quad 0 \le z \le 0.5.$$

The channel was truncated in the x direction to have length one. During the computation, the computational domain was translated along with the wave in the x direction. The detonation wave will exhibit transverse wave structure, and oscillate in various modes as it propagates down the channel. This is similar to the two dimensional computation in [3].

We used the following parameter values, $\gamma = 1.2$, $K = 10^4$, $T_i = 50$, $q_0 = 50$. These are the same as used in [3]. They correspond to a half reaction length, $x_{1/2}$, of 0.0285 and half reaction time of 0.0204. The channel width is thus approximately $17.5x_{1/2}$.

As initial data we first compute a one dimensional ZND profile, which connects the left hand state $\rho = 1$, $p = 3$, $m = n = o = 0$, $z = 1$ by a Chapman-Jouguet detonation to a right hand state $\rho = 1.731379$, $p = 24.52073$, $m/\rho = 3.015113$, $n = o = z = 0$. To start the transverse wave motion, we translate one dimensional ZND profile by translating it a few grid points in the x-direction, according to a sinusoidal profile in the $y - z$ plane i.e.,

$$u(0, x, y, z) = u_1(x + a \sin^2 2\pi y \sin^2 2\pi z)$$

with a a small constant, and u_1 the one dimensional ZND profile.

At the walls we imposed Euler boundary condition i.e., normal velocity equal to zero. The other variables were extrapolated. At the open ends $x = 0$ and $x = 1$, the left and right states of the initial traveling wave solution were imposed.

We start with a comparison of a two dimensional and a three dimensional computation. The solution develops triple points on the wave, which move up and down in the transverse direction. In Fig. 5.1 we display iso density contour lines from a two dimensional wave at seven different times.

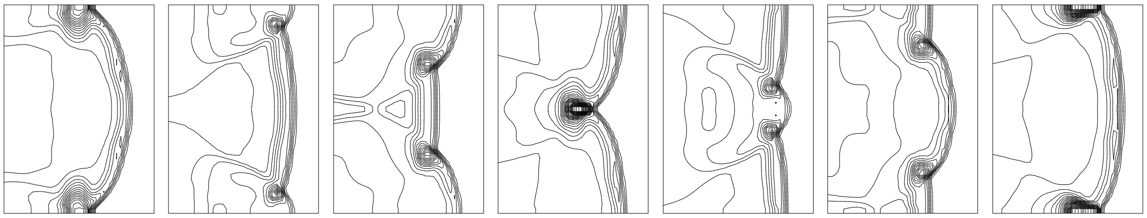

FIG. 5.1. *Iso density contours at seven times, two dimensional computation.*

In Fig. 5.2 we display the density in the cut plane $y = 0$ in a three dimensional solution for six different times. The solutions were computed with 100×50 and $100 \times 50 \times 50$ grid points respectively.

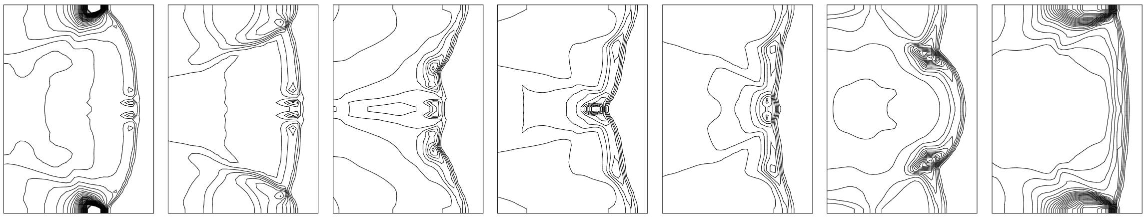

FIG. 5.2. *Density contours at seven times, three dimensional computation.*

In experiments with detonations, a soot-coated plate is inserted as a wall, and the triple point will trace out a pattern [1]. We obtain a similar picture from a numerical computation, by plotting iso density contours in the $y - t$ plane.

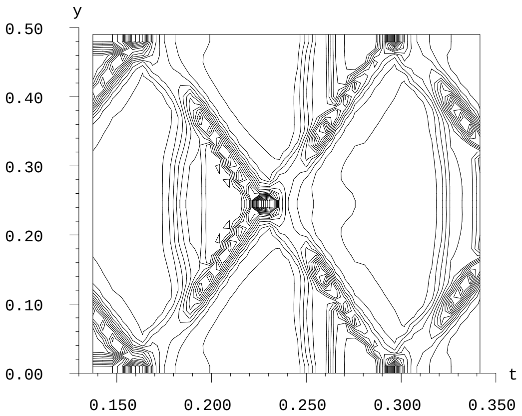

FIG. 5.3. *Density contours in the $y - t$ plane, two dimensional computation.*

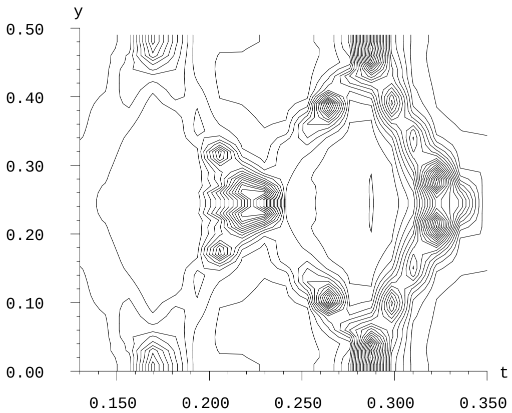

FIG. 5.4. *Density contours in the $y - t$ plane, three dimensional computation.*

In Fig. 5.3 we show iso density contours in the $y - t$ plane for the two dimensional computation, and in Fig. 5.4 the two dimensional coordinate plane $j = 1$ in a three dimensional solution. We see that the detonation cells are of almost the same size. In [4] the disagreement between cell sizes computed in two space dimensions, and sizes obtained in experiments, were explained by the two dimensionality in the computations. We conclude that for the simple model reaction $R \rightarrow P$ considered here, cell sizes do not depend on the dimension.

In Fig. 5.5 we show plots of one iso mass fraction surface at the value $z = 0.5$ for the three dimensional wave. The iso surfaces show the transverse modes of the wave. Note that the two dimensional triple points are seen along the edges of the iso surfaces.

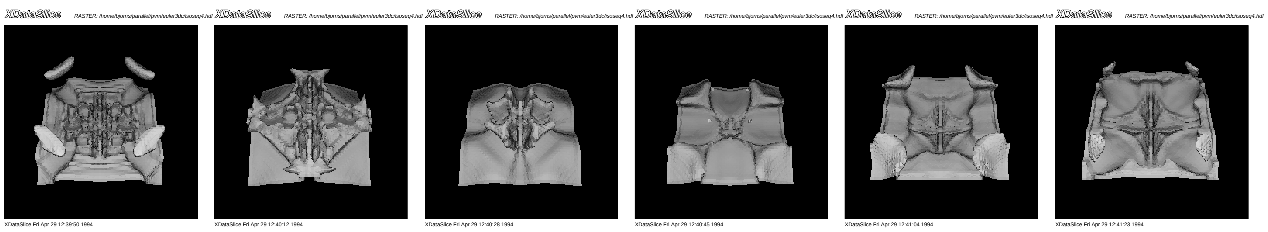

FIG. 5.5. *Iso surfaces of the mass fraction at the value $z = 1/2$, three dimensional detonation.*

In Fig. 5.6 we display a two dimensional computation, now computed by a first order accurate method in space, instead of the second order method which was used in Fig. 5.1.

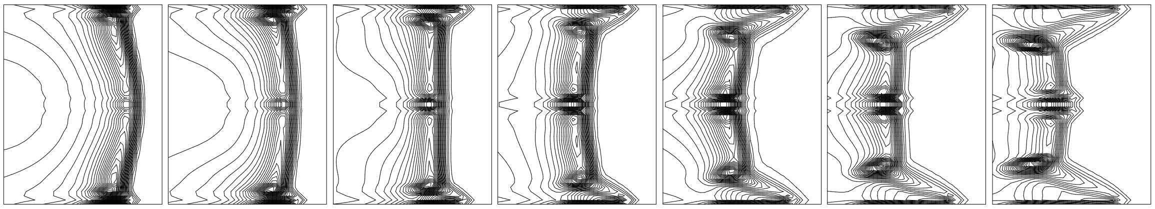

FIG. 5.6. *Density contours at seven times, two dimensional computation, first order accurate.*

We see that the wave pattern becomes steady, and that discontinuities diagonal to the grid develop. This is a numerical effect, and it should not be in the solution. In [3], we found such solutions in two dimensions, they occur because K is large, and the time step is not small enough to resolve the chemical time scale. These solutions are related to the well known unphysical one dimensional numerical solutions [2].

In Fig. 5.7, we show how this effect appears in three dimensions. The computation is the same as in Fig. 5.5, but with a first order accurate method in space.

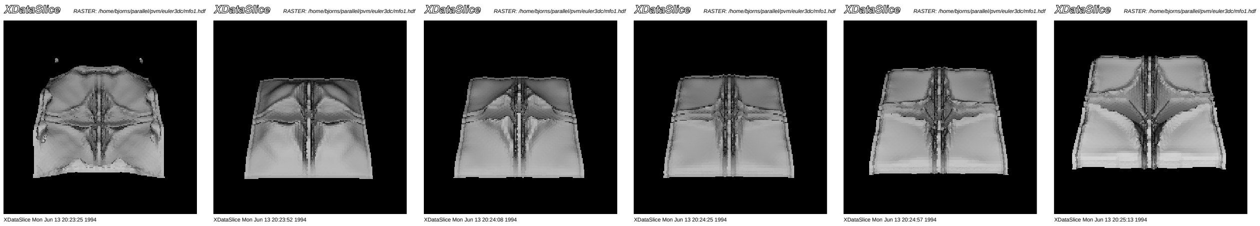

FIG. 5.7. *Mass fraction iso surfaces, first order accurate method.*

We conclude that the order of accuracy in the numerical method is of extreme importance. The numerical difficulties previously observed in one [2] and two [3] dimensional computations are present in three dimensions as well.

Acknowledgements

I am grateful to Per Hammarlund at the Royal Institute of Technology, Stockholm for discussions on implementation, and help to access various parallel computes.

University of California at Los Angeles supplied CPU time on their IBM/SP1 computer.

The iso surface plots were made with the program XDataSlice from the National Center for Supercomputing Applications, (NCSA).

References

[1] D. C. Bull, J. E. Elsworth, P. J. Shuff, and E. Metcalfe, *Detonation Cell Structures in Fuel/Air Mixtures*, Combust. Flame, **45** (1982), pp. 7–22.

[2] P. Colella, A. Majda, and V. Roytburd, *Theoretical and Numerical Structures for Reacting Shock Waves*, SIAM J. Sci. Stat. Comput., **7** (1986), pp. 1059–1080.

[3] B. Engquist and B. Sjögreen, *Robust Difference Approximations of Stiff Inviscid Detonation Waves*, UCLA, CAM Report 91-03.

[4] K. Kailasanath, E. S. Oran, J. P. Boris, and T. R. Young, *Determination of Detonation Cell Size and the Role of Transverse Waves in Two-Dimensional Detonations*, Combust. Flame, **61** (1985), pp. 199–209.

[5] C-W. Shu and S. Osher, *Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes*, J. Comput. Phys., **77** (1988), pp. 439–471.

[6] B. Sjögreen, *Shock Computations on Parallel Computers*, UCLA, CAM Report 87-27.

Appendix A

We shall give the eigendecomposition necessary for Roe's approximate Riemann solver. We assume that γ is constant. We consider the system (2.1) without source term. On a variable grid, metric coefficients enter into the equations and we are led to consider the flux function

$$\alpha\mathbf{F}+\beta\mathbf{G}+\delta H=\alpha\begin{pmatrix} m \\ m^2/\rho+p \\ mn/\rho \\ mo/\rho \\ m(e+p)/\rho \\ mz \end{pmatrix}+\beta\begin{pmatrix} n \\ mn/\rho \\ n^2/\rho+p \\ no/\rho \\ n(e+p)/\rho \\ nz \end{pmatrix}+\delta\begin{pmatrix} o \\ mo/\rho \\ no/\rho \\ o^2/\rho+p \\ o(e+p)/\rho \\ oz \end{pmatrix}$$

with $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ the x, y and z direction fluxes in (2.1) respectively and α,β and δ represents the metric. We define

$$u=m/\rho,\quad v=n/\rho,\quad w=o/\rho,\quad p=(\gamma-1)(e-\frac{1}{2}\rho(u^2+v^2+w^2)-q_0\rho z)$$

The eigenvalues of the jacobian of $\alpha\mathbf{F}+\beta\mathbf{G}+\delta\mathbf{H}$ are

$$\lambda_1=\bar{u}-sc,\quad \lambda_2=\lambda_3=\lambda_4=\lambda_5=\bar{u},\quad \lambda_6=\bar{u}+sc$$

with $\bar{u}=\alpha u+\beta v+\delta w$ and $s=\sqrt{\alpha^2+\beta^2+\delta^2}$. The eigenvectors are

$$\mathbf{r}_1=\begin{pmatrix} 1 \\ u-k_1c \\ v-k_2c \\ w-k_3c \\ h-\hat{u}c \\ z \end{pmatrix}\quad \mathbf{r}_2=\begin{pmatrix} 1 \\ u \\ v \\ w \\ q^2/2 \\ 0 \end{pmatrix}\quad \mathbf{r}_3=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ q_0 \\ 1 \end{pmatrix}$$

$$\mathbf{r}_4=\begin{pmatrix} 0 \\ a_1 \\ a_2 \\ a_3 \\ \hat{v} \\ 0 \end{pmatrix}\mathbf{r}_5=\begin{pmatrix} 0 \\ b_1 \\ b_2 \\ b_3 \\ \hat{w} \\ 0 \end{pmatrix}\mathbf{r}_6=\begin{pmatrix} 1 \\ u+k_1c \\ v+k_2c \\ w+k_3c \\ h+\hat{u}c \\ z \end{pmatrix}$$

with the definitions

$$\begin{aligned}
&k_1=\alpha/\sqrt{\alpha^2+\beta^2+\delta^2},\quad k_2=\beta/\sqrt{\alpha^2+\beta^2+\delta^2},\quad k_3=\delta/\sqrt{\alpha^2+\beta^2+\delta^2}\\
&\hat{u}=k_1u+k_2v+k_3w,\quad \hat{v}=a_1u+a_2v+a_3w,\quad \hat{w}=b_1u+b_2v+b_3w\\
&q^2=u^2+v^2+w^2,\quad c^2=\gamma p/\rho
\end{aligned}$$

The vectors $\mathbf{a}=(a_1\ \ a_2\ \ a_3)$, $\mathbf{b}=(b_1\ \ b_2\ \ b_3)$, are such that they are orthogonal to the vector $\mathbf{k}=(k_1\ \ k_2\ \ k_3)$, and such that $\mathbf{a},\mathbf{b},\mathbf{k}$ spans $\mathbf{R}^3$, but otherwise arbitrary. We can use the equation $\mathbf{k}=\mathbf{a}\times\mathbf{b}$ to determine $\mathbf{a}$ and $\mathbf{b}$. The determination of $\mathbf{a}$ and $\mathbf{b}$ turns out to cost a fairly large number of arithmetic operations. Fortunately, in Roe's method, it is not necessary to explicitly compute $\mathbf{a}$ and $\mathbf{b}$. The eigenvectors are linearly independent for c positive. Thus the system of equations

$$(\mathbf{r}_1 \ \ \mathbf{r}_2 \ \ \mathbf{r}_3 \ \ \mathbf{r}_4 \ \ \mathbf{r}_5 \ \ \mathbf{r}_6)\begin{pmatrix} x_1 \\ \ldots \\ x_6 \end{pmatrix} = \begin{pmatrix} d\rho \\ dm \\ dn \\ do \\ de \\ dy \end{pmatrix} \tag{A.1}$$

has a unique solution. The solution is needed in Roe's method to find the coefficients α. To solve it, define

$$\begin{aligned} d_1 &= \frac{\gamma-1}{c^2}(de + \frac{q^2}{2}d\rho - udm - vdn - wdo - q_0 dy) \\ d_2 &= (k_1 dm + k_2 dn + k_3 do - \hat{u} d\rho)/c \end{aligned}$$

then the solution is

$$x_1 = (d_1 - d_2)/2 \quad x_2 = d\rho - d_1 \quad x_3 = dy - zd_1 \quad x_6 = (d_1 + d_2)/2$$

From the numerical flux (2.x) for Roe's method, we infer that it is enough to compute

$$x_4 \mathbf{r}_4 + x_5 \mathbf{r}_5$$

instead of evaluating x_4 and x_5 separately. This expression turns out to be

$$\begin{aligned} x_4 a_1 + x_5 b_1 &= dm - d_2 k_1 c - ud\rho \\ x_4 a_2 + x_5 b_2 &= dn - d_2 k_2 c - vd\rho \\ x_4 a_3 + x_5 b_3 &= do - d_2 k_3 c - wd\rho \\ x_4 \hat{v} + x_5 \hat{w} &= udm + vdn + wdo - q^2 d\rho - d_2 \hat{u} c \end{aligned}$$

The left eigenvectors can easily be found by letting the right hand side in (A.1) be unit vectors. To impose characteristic boundary conditions, we need the expressions

$$l_k^T \frac{d\mathbf{u}}{ds_k} = 0, \quad k = 1, \ldots, 6$$

where l_k is the kth left eigenvector, and s_k the parameter changing along the kth characteristic. These six equations are equivalent with

$$\begin{aligned} &\frac{d\hat{u}}{ds} - \frac{1}{\rho c}\frac{dp}{ds} = 0 \quad k = 1 \\ &\frac{d\rho}{ds} - \frac{1}{c^2}\frac{dp}{ds} = 0 \ \ \frac{dz}{ds} = 0 \ \ \frac{d\hat{v}}{ds} = 0 \ \ \frac{d\hat{w}}{ds} = 0 \quad k = 2,3,4,5 \\ &\frac{d\hat{u}}{ds} + \frac{1}{\rho c}\frac{dp}{ds} = 0 \quad k = 6 \end{aligned}$$