User guide for shock and blast simulation with the OctVCE code (version 3.5+)

Mechanical Engineering Report 2007/13
Joseph Tang
Department of Mechanical Engineering
The University of Queensland
Brisbane QLD 4072

August 9, 2007

Abstract

OctVCE is a cartesian cell CFD code produced especially for numerical simulations of shock and blast wave interactions with complex geometries. Virtual Cell Embedding (VCE) was chosen as its cartesian cell kernel as it is simple to code and sufficient for practical engineering design problems. This also makes the code much more 'user-friendly' than structured grid approaches as the gridding process is done automatically. The CFD methodology relies on a finite-volume formulation of the unsteady Euler equations and is solved using a standard explicit Godonov (MUSCL) scheme. Both octree-based adaptive mesh refinement and shared-memory parallel processing capability have also been incorporated. For further details on the theory behind the code, see the companion report [20].

Contents

List of Figures

Chapter 1

Introduction

This user guide to the OctVCE code is written with the intent of helping new users easily set up and run simulations for shock and blast simulation. It therefore does not describe in detail the source code files or the underlying theory. A more extensive coverage of the theory can be found in the companion report [20]. But inevitably, some further discussion on the numerical methodology is necessary (particularly for new users) to understand how simulations are set up. This will be the subject of §1.1 to §1.5.

1.1 Scope and formulation

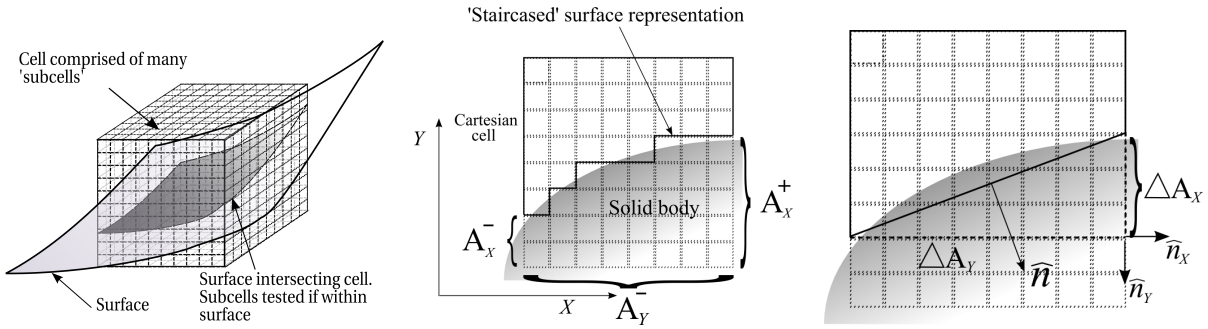
OctVCE is a cartesian cell code written in C designed especially for modelling shock and blast effects (in particular from bomb explosions) in complex geomtries. It aims to reduce problem set-up time and enable users to focus more on design aspects by providing an automatic mesh generation capability. The gridding technique chosen is Virtual Cell Embedding (VCE) [9], a particularly simple method. Very small cells are merged with larger neighbours to give acceptable timesteps. OctVCE also implements an isotropic h-refinement (octree-based) procedure for CPU and memory efficiency.

OctVCE adpots a finite-volume formulation of the unsteady Euler equations with a second order explicit Runge-Kutta Godonov (MUSCL) scheme. The gradients are calculated with a least-squares method [8] and the limiter chosen is of the min-mod variety [5]. Flux solvers used are AUSM [11], AUSMDV [23] and EFM [13]. No fluid-structure coupling or chemical reactions are assumed, although gas models can be perfect gas, JWL [10] or JWLB [3] for the explosive products. The numerical methodology is consistent with OctVCE's scope, which is limited to problems of practical engineering design where very high resolution or realism is unnecessary (e.g. for determining blast overpressures or impulse).

1.2 VCE method

The basic idea of the VCE method [9] is to subdivide a body-intersected cell into a lattice of 'subcells' (fig 1.1(a)). Each subcell is tested if it is obstructed by a body. Summation of the obstructed subcells give the approximate obstructed volume and interface areas. The surface is

then approximated using this information, either with a staircased representation (fig 1.1(b)) or smooth planar representation (fig 1.1(c)). Normally more subcells are used on the face areas than cell volume as the body representation really depends on the obstructed interface areas.



(a) 'Subcells' illustration (b) Staircased surface (c) Planar surface

Figure 1.1: VCE method

1.3 Octree domains

OctVCE implements isotropic subdivision of cubical cells as its h-refinement procedure. This approach lends itself easily to an 'octree' structure where a parent cell is refined to give 8 children (and similarly for the coarsening process), as in fig 1.3. Thus OctVCE starts with a *root* cell which **corresponds to the numerical domain**; this root cell is then refined a certain number of *levels* (the root is level 0) to give the initial mesh.

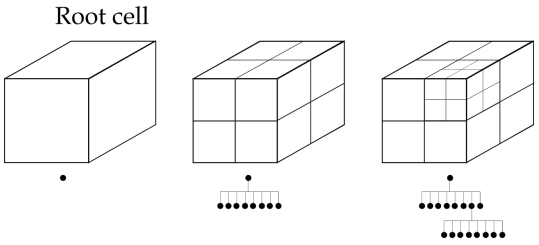


Figure 1.2: Octree illustration

When the user enters the size and location of the root cell the numerical domain's extent is fully specified. The root cell's dimensions are chosen to immerse the mesh appropriately within an environment of solid objects (whose locations are usually known beforehand). Whilst solid wall boundary conditions are implemented automatically by the cartesian cell method by the presence of solid objects, the user is required to specify an appropriate *domain boundary condition* for each of the 6 faces of the root cell as they represent the borders of the numerical domain.

1.4 Initializing explosions

The initial explosion or 'bomb' is represented by a volume or region of high pressure and density explosive gas (this approach is also adopted by Timofeev et al [22]). These explosive gases can

be modelled with the perfect gas, JWL [10] or JWLB [3] equation of state (described in further detail in §1.5). It has been found that matching the required total explosive energy is much more important than the initial shape or density of the explosion volume.

Usually cells at the explosion volume are refined to the highest permissible level to track the developing blast wave accurately from the start. This method of mapping the bomb volume to a cartesian cell representation 'staircases' the charge and thus will not produce very accurate solutions in the near-field when the actual charge shape is similarly not staircased. See [16] for a 'remapping' technique between 1D spherical, 2D axisymmetric and finally 3D solutions is employed that gives higher accuracy; the approach here is simpler and should hopefully be adequate in the mid- to far-field.

OctVCE also supports a 'detonation-like' finite-energy release scheme where the explosion volume is treated as a solid explosive i.e. all the cells within the explosion volume are not part of the computational domain. Ignition points are specified at locations within the explosive with a 'detonation' wave moving outward at constant velocity which 'activates' the cells, initializing them to a specified high pressure and density. This not an actual detonation process, but is useful for more detailed modelling of non-uniform blast in the near-field as the location of charge initiation can sometimes have an effect on both peak overpressure and impulse [1].

1.4.1 Initial conditions for JWL or JWLB equations of state

As OctVCE cannot simulate an actual detonation process, one might ask to what pressures or densities the initial bomb should be set. Typically the density of the bomb ρ_b chosen is the undetonated density ρ_0 (or loading density) of the explosive. This is to get the correct initial mass and volume. The initial JWL or JWLB explosive or chemical energy E_0 for the bomb is also assumed (this value is reported for various explosives in [10, 4]). The pressure is then calculated using the equation of state i.e. $p = p(\rho_b, E_0)$.

The mapping of a bomb shape to the cartesian grid means that it will be difficult to match exactly the same volume occupied by the actual explosive. Depending on the grid resolution, this can result in a considerable mismatch of initial explosive energy if the same explosive density ρ_0 and energy E_0 are used as the initial condition. However, as will be seen in §B.2.3 OctVCE can also conveniently output the total volume of cells representing the bomb volume V_b. This information can be used to alter the initial explosion condition so that the initial explosive energy matches exactly the actual energy.

For example, suppose exact matching of the the blast energy density E_0 (in J/kg) and bomb mass m is desired. The total blast energy is then $E_b = mE_0$. The match is ensured by simply adjusting that $\rho_b = m/V_b$. It is also possible to assume that the undetonated density ρ_0 in the JWL equation of state (eqn 1.1) is equal to this value of ρ_b instead of the figure reported in the literature.

1.5 The JWL and JWLB equations of state

It is useful to describe the actual JWL and JWLB equations of state here for later reference. The JWLB equation of state is

$$p = p(\rho, e) = \sum_{i=1}^{n} A_i \left(1 - \frac{\lambda(\widetilde{v})}{R_i \widetilde{v}}\right) e^{-R_i \widetilde{v}} + \frac{\lambda(\widetilde{v})\, e}{v} + C\left(1 - \frac{\lambda(\widetilde{v})}{\omega}\right) \widetilde{v}^{-(\omega+1)} \tag{1.1}$$

where the Grüneisen coefficient λ given by

$$\lambda(\widetilde{v}) = \sum_{i=1}^{n} \left(A_{\lambda_i} \widetilde{v} + B_{\lambda_i}\right) e^{-R_{\lambda_i} \widetilde{v}} + \omega \tag{1.2}$$

and $\widetilde{v} = v/v_0$.

$v = 1/\rho$ is the specific volume of explosion products whilst $\rho_0 = 1/v_0$ is the undetonated (loading) density of the solid explosive. The constants A_i, R_i, A_{λ_i}, B_{λ_i}, R_{λ_i}, C, ω and v_0 can be found for various explosives in [4]. At most $n = 5$. Note that A_i and C are dimensional (in units of pressure), whilst R_i, A_{λ_i}, B_{λ_i}, R_{λ_i} and ω are non-dimensional (thus $\lambda(\widetilde{v})$ is also non-dimensional).

Note that the JWL equation of state is basically the JWLB equation with $n = 2$, $C = 0$ and $\lambda = \omega$ (so with all A_{λ_i} and B_{λ_i} zero). Constants for the JWL equation are provided in [10].

It is also important for later purposes to note the temperature-dependent form of these equations of state i.e. $p = p(\rho, T)$. The temperature dependent form for the JWLB equation is quite lengthy but is reported in [3]. However the JWL form is much simpler and can be written here as

$$p = A_1 e^{-R_1 \widetilde{v}} + A_2 e^{-R_2 \widetilde{v}} + \omega \rho C_v T \tag{1.3}$$

where C_v is the specific heat at constant volume for the explosion products.

Chapter 2

Code compilation

OctVCE works best under a Unix or Linux system. To compile OctVCE, both the `Octvce3.5.⋆/` (where ⋆ is the subkernel version) and `Geomio/` directories are required. The latest subkernel version as of April 2007 is `4`. These directories should be placed within the same level directory.

The code is compiled using the *makefile* in the `Octvce3.5.⋆/unix/` directory. Near the top of the *makefile* there are 3 variables which need to be set by the user. Their function will be described below, along with some advice on how they can be set.

1. `CC` – the compiler name e.g. `icc` or `pgcc` (an OpenMP compiler [6] is necessary for parallel processing)
2. `CFLAGS` – the compiling options (which are compiler dependent) e.g. `-O3`.

   Note – at present a `-static` flag (especially under the Intel compiler) is not recommended as it causes the code to consume exorbitant system memory. This author is still working on the cause for this (though this flag is not really necessary anyway).
3. `OUT_DIR` – where the executable `octvce.exe` is placed.

   By default this is in `Octvce3.5.⋆/tests/`

In the past the author has typically compiled with the Intel compiler with flags like `-O3`, `-openmp`, `-ipo` and `-ip`. With the Portland compiler the flags were `-O3`, `-mp`, `-fastsse` and `-Mipa=fast`.

The other variables `OV_SRC_DIR` and `GM_SRC_DIR` need not be altered, as they are already set to where the source files are located. If it is absolutely necessary to move `Geomio/` to some other location, both the *makefile* variable `GM_SRC_DIR` and the header file locations in `Octvce3.5.⋆/source/ov_kernel.h` need to be altered correspondingly. After setting these variables appropriately just type `make` and the executable should be produced.

Chapter 3

Setting up a simulation

OctVCE requires several *input files* for running simulations (assumed to be in the same directory that the executable `octvce.exe` is located). These files describe the solid geometry, gas models, domain boundary conditions (see §1.3), initial conditions, and other important parameters like domain size, maximum refinement level, output frequency etc. These input parameters will be described below.

3.1 General simulation parameters input file

The general simulation parameters input file specifies to OctVCE such data as the root cell size and location, run parameters, adaptation parameters, subcelling information and output frequency information. It must be written **in exactly the same format** as the template in fig 3.1 (a template can also be found in `Octvce3.5.*/tests/ov.par`). All words or spacings between lines in the file must follow the template exactly. This lack of flexibility isn't as bad as it seems as templates can be reused and modified repeatedly.

A description of each input section (along with advice on what to input) will be described below. Note that **any dimensional parameters are assumed to be in SI units** e.g. metres, seconds, pascals, joules, kilograms etc.

1. `Octree root spatial properties`

   Description :– Specifies location and size of the cubical root cell i.e. computational domain.

   Parameters to input :–

   (a) `Centroid location:` – enter where in physical space the root cell should be centered

   (b) `Edge length scale:` – enter the edge length of the root cell

2. `Run parameters`

   Description :– Some parameters on how the flow is solved numerically

   Parameters to input :–

(a) `Type of flux solver:` – options include `AUSM`, `AUSMDV`, `EFM`, `ADAPTIVE`

The `ADAPTIVE` option uses a combined solver where EFM is used at shocks and AUSMDV elsewhere, which is handy for eliminating odd-even decopuling issues [14].

(b) `Use multiple limiters?` – enter in `y` or `n`

Under the present methodology the extreme initial conditions at an explosion's beginning can sometimes cause the higher-order reconstruction scheme to fail (especially if the JWL or JWLB equation is used). Using a single global limiter can prevent this instability, though multiple limiters give a more accurate solution. More discussion on the failure of OctVCE's flow solver is also found in §4.

(c) `Global timestep (set to 0 if don't want it fixed):` – usually this is set to 0 to allow for varying timesteps.

(d) `Max CFL (must be between 0 and 1):` – enter in the *maximum allowable* CFL number. Usually this is set to 0.5. Strong explosions may require such small timesteps initially that the CFL may need to be quite low at the beginning of a simulation. Eventually this CFL number ramps back up to the maximum allowable value.

(e) `Finish time (choose one and make the other '0')` – simulation finishes either when a set number of timesteps is exceeded or flow time exceeded. If finish time is measured by timesteps/flow time, a `0` must be put in the other field.

3. `Parallel processing`

Parameter to input :–

`Adapt in parallel?` – enter in `y` or `n`

Usually this is `y`. If parallel processing isn't employed, this line is ignored. OctVCE can parallelize the adaptation procedure, which generally gives faster code for simulations involving many cells. But sometimes the extra work in forking and joining threads might be too time-intensive for small simulations.

4. `Adaptation parameters`

Description :– Specifies adaptation option, and thresholds for refinement and coarsening around flow features like shocks or contact discontinuities

Parameters to input :–

(a) `Adapt every how many time steps (enter '0' for no adaptation):` – if adaptation is desired, '5' is a recommended number, else '0' for no adaptation

(b) `Type of error indicator:` – options are `1`, `2`, `3`

Please consult [20] for further documentation regarding adaptation indicators. Indicator `1` corresponds to the shock-detection scheme based on velocity gradients. Indicator `2` uses density differences (good for detecting contact discontinuities). Indicator `3` simply means indicators `1` and `2` being used together.

(c) `If error indicator type is 1 or 3 - compression threshold:` – a value from 0.05 to even 0.005 is acceptable. The smaller the number, the greater the chance that weak compressions (not shocks) will also be refined about.

(d) `If error indicator type is 2 or 3 ...`

i. `Refinement threshold:` – the user must experiment with this, but 0.3 seems like a good starting figure for many calculations. If flow features are quite weak, this number may need to be reduced.

ii. `Coarsening threshold:` – the user must experiment with this, but it must be smaller than the refinement threshold. 0.1 seems like a good starting figure, though the author has used values as low as 0.01. The lower the value, the better chance the weaker features will still be refined about.

iii. `Noise filter value:` – this can vary between 0.001 to 0.1. The larger the value, the less chance for refinement around compressions. The user must experiment with this.

5. `Octree refinement parameters`

Description :– Specifies subdivision of root cell to form initial grid, minimum and maximum refinement levels when adaptation is used.

Note – the root cell is at level 0; if it is refined n times the leaf cells will be at level n.

Parameters to input :–

(a) `No. times to refine root initially:` – this refines the root cell *uniformly* to create the initial grid (which would also be the final grid if adaptation isn't employed). However if any cells are known to be 'solid' i.e. completely immersed in a body, it won't be refined any further. Usually the initial grid can be fairly coarse, so this number is typically smaller than 8.

(b) `Max refinement level:` – this obviously has to be a higher level than the initial grid level. The highest level of refinement used thus far by the author has been 11.

(c) `Min refinement level:` – usually this is the same as the initial grid level.

(d) `Min intersect refinement level:` – specifies the minimum level that *partially obstructed* cells should be at. All cells with objects cutting through their *volumes* will be refined to this level. Sometimes handy when higher resolution is required at surfaces, but generally this has been set to the initial grid level.

(e) `Level of cell IC intersection:` – specifies the level that cells intersecting the initial bomb volume (see §1.4) should be at. Usually this is set to the maximum refinement level.

6. `Geometry engine parameters`

Description :– Provides data for the geometry engine to complete its process of approximating surfaces with VCE in obstructed cells

Parameters to input :–

(a) `Interrogate geometry engine only after what level:` – specifies for what cell levels the actual point-in-polyhedron queries will be performed. A higher figure means more computation (but more accuracy). It's generally set to the same level as the initial grid level. But if the initial grid level is very high (say $\geq$ level 7), a lower number will give substantially faster gridding time.

(b) `No. area subcells along edge (must be even):` – the number of subcells ns_f *along* a single cell face edge. The total number of subcells on one cell face is thus ${ns_f}^2$. The higher the number, the more accurate the surface representation. Usually between 20 and 64 (but for 2D simulations, see §3.7).

(c) `No. volume subcells along edge (must be even):` – the number of subcells ns_v along a cell volume edge. The total number of volume subcells is thus ${ns_f}^3$. A number of about 10 to 16 should be sufficient.

(d) `Wall representation ('0' for staircased, '1' for smoothed):` – as written, only `0` or `1` can be entered. See §1.2 for explanation. Generally a smooth wall (option `1`) is preferrable.

7. `Visualization`

Description :– Specifies what data to output and how often

Parameters to input :–

(a) `Write flow soln of only intersected cells?` – options are `y` or `n`.
Generally `n` is entered as the whole flowfield is desired. If `y` is used then only the output from cells intersected or flush with solid surfaces will be given (which gives a much smaller solution file).

(b) `Write data frequency (choose one and make the other '0')` – usage is the same as with the finish time (see above). Either output every some number of timesteps or at a specified time interval. If write data frequency measured by timesteps/flow time, a `0` must be put in the other field.
The solution files will be in the VTK XML format and have a `.vtu` and/or `.pvtu` extension. See §5 for further information.

(c) `Base solution file name:` – enter in a name for the solution files. Other information e.g. the timestep where the file is output and finally the `.vtu` extension will be appended to this.

8. `History files`

Description :– Specifies at which locations the pressure history should be recorded. See §5.3 for further information on the output format of the pressure traces.

Parameters to input :–

(a) `No. history locations:` – how many pressure traces are desired? If this is 0, all further lines will be ignored.

(b) `Dump history locations every how many time steps:` – how often should the pressure at a point be recorded. Typically recording every 2 to 5 timesteps is enough.

(c) `Base history file name:` – enter in a name for the trace recording. Other information e.g trace location is appended.

9. `History locations`

Parameters to input :– At each line enter in the gauge location where the pressure trace is recorded. Must be a 3D point.

```
INPUT PARAMETERS FOR OCTVCE SIMULATION
--------------------------------------


Octree root spatial properties
------------------------------

Centroid location: 2 2 2
Edge length scale: 4


Run parameters
--------------

Type of flux solver: ADAPTIVE
Use multiple limiters? n
Global timestep (set to 0 if don't want it fixed): 0
Max CFL (must be between 0 and 1): 0.5
Finish time (choose one and make the other '0') -
  By timestep: 0
  By flow time: 1e-2


Parallel processing
-------------------

Adapt in parallel? n


Adaptation parameters
---------------------

Adapt every how many time steps (enter '0' for no adaptation): 5
Type of error indicator: 2

  If error indicator type is 1 or 3 - compression threshold: 0.05

  If error indicator type is 2 or 3 ...
  -------------------------------------
    Refinement threshold: 0.3
    Coarsening threshold: 0.1
    Noise filter value: 0.05


Octree refinement parameters
----------------------------

No. times to refine root initially: 5
Max refinement level: 9
Min refinement level: 5
Min intersect refinement level: 5
Level of cell IC intersection: 9


Geometry engine parameters
--------------------------

Interrogate geometry engine only after what level: 5
No. area subcells along edge (must be even): 64
No. volume subcells along edge (must be even): 10
Wall representation ('0' for staircased, '1' for smoothed): 0


Visualization
-------------

Write flow soln of only intersected cells? n
Write data frequency (choose one and make the other '0') -
  By timestep: 0
  By flow time: 1e-3
Base solution file name: soln


History files
-------------

No. history locations: 1
Dump history locations every how many time steps: 5
Base history file name: hist_

History locations
-----------------
1 0 0
```

Figure 3.1: General parameters file

3.2 Solid geometry definition

Each solid object in the computational domain must be 3D object and defined with the VTK XML version 4.2+ ASCII file format [2] definition[1] (commonly with a `.vtp` extension). However it is not necessary for VTK to be installed on the system as OctVCE contains its own VTK file parser.

It is important all these objects are 'watertight' i.e. no 'holes' which may confound the geometry engine, and that face definitions are defined in *counterclockwise* fashion (using the right hand rule) so that the surface normal points *outward* from the surface.

For example, fig 3.2 is a VTK file describing a unit cube (centered at $(0.5, 0.5, 0.5)$). The `<Polys>` dataset describes how the cube's verticies are joined to make the individual faces. Note that the connectivity data joins the verticies in the required counterclockwise order. Only the `Points`, `connectivity` and `offsets` data arrays are required to fully define a solid body.

```
<?xml version="1.0"?>
<VTKFile type="PolyData" version="0.1" byte_order="LittleEndian">
<PolyData>
<Piece NumberOfPoints="8" NumberOfVerts="0" NumberofLines="0" NumberOfStrips="0" NumberOfPolys="6">
<Points>
<DataArray type="Float32" NumberOfComponents="3" format="ascii">
0 0 0
1 0 0
1 1 0
0 1 0
0 0 1
1 0 1
1 1 1
0 1 1
</DataArray>
</Points>
<Polys>
<DataArray type="Int32" Name="connectivity" format="ascii">
0 3 2 1
4 5 6 7
0 1 5 4
2 3 7 6
0 4 7 3
1 2 6 5
</DataArray>
<DataArray type="Int32" Name="offsets" format="ascii">
4
8
12
16
20
24
</DataArray>
</Polys>
</Piece>
</PolyData>
</VTKFile>
```

Figure 3.2: Example VTP file of a cube

With a each solid object described by its own VTK file, the collection of solid objects is then placed in a 'parallel' VTK file (with a `.pvtp` extension) to describe the physical *environment* of all objects. Even if there is only 1 body it must be still entered into this file. An example parallel VTK file where 3 solid objects exist in the environment is provided in fig 3.3.

[1] Or see www.vtk.org/pdf/file-formats.pdf

```
<?xml version="1.0"?>
<VTKFile type="PPolyData" version="0.1" byte_order="LittleEndian">
<PPolyData GhostLevel="0">
<PPoints>
<PDataArray type="Float32" NumberOfComponents="3"/>
</PPoints>
<Piece Source="body1.vtp"/>
<Piece Source="body2.vtp"/>
<Piece Source="body3.vtp"/>
</PPolyData>
</VTKFile>
```

Figure 3.3: Example PVTP environment definition

3.2.1 A note on flushed surfaces

In many situations there could exist solid bodies which are exactly flush with a grid line/plane e.g. in creating a solid domain boundary (as in fig 3.4). This can sometimes present a problem of geometric tolerance as OctVCE's geometry engine may on occasion compute the cell faces flush with the solid surfaces as unobstructed still.

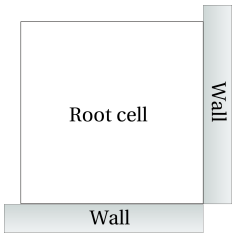


Figure 3.4: Boundary-aligned wall

The easiest solution is to have a small overlap between cells and the solid surfaces. Two ways to do this are –

1. Modify the body VTK file so that the solid surface overlaps very slightly with the cell surface (increasing or decreasing a dimension by 10^{-10} is enough)
2. Very slightly increase/decrease the root cell's edge (i.e. domain's) length to cause this overlap (again an increase of 10^{-10} should be sufficient)

3.3 IC input file

The initial condition input file specifies to OctVCE what the initial conditions in the computational domain should be. When modelling explosions, OctVCE requires 2 initial conditions – (a) the initial flow state for the ambient (commonly atmospheric) gas, and (b) the conditions for the initial explosion products (recall §1.4).

As with §3.1, this file must be written in **exactly the same format** as the template in fig 3.5 (a template can also be found in `Octvce3.5.*/tests/ov_IC.par`). Note that **any dimensional parameters are assumed to be in SI units**. A description of this file will be given below.

1. `Ambient conditions` – initial conditions for the ambient gas

   Parameters to input :–

   (a) `Pressure:` – enter the pressure (in Pa) for the ambient gas (typically atmospheric conditions)

   (b) `Density:` – enter the density (in kg/m^3) for the ambient gas (typically atmospheric conditions)

   (c) `U:` – enter in the initial velocity (in m/s) along the x axis

   (d) `V:` – enter in the initial velocity (in m/s) along the y axis

   (e) `W:` – enter in the initial velocity (in m/s) along the z axis

2. `Products conditions` – initial conditions for the explosive products gas

   Note – this whole section can be omitted if only ambient conditions are needed e.g. solving steady state flow over a wedge where the initial conditions are the constant freestream conditions.

   Parameters to input :–

   (a) `Pressure:` – enter the pressure (in Pa) for the initial explosion volume. If the JWL/JWLB equation is used see §1.4.1 for an idea of what pressure to use.

   (b) `Density:` – enter the density (in kg/m^3) for the initial explosion volume (see §1.4.1 if the JWL/JWLB equation is used)

   (c) `U:` – enter in the initial velocity along the x axis (for high explosive modelling, is typically 0)

   (d) `V:` – enter in the initial velocity along the y axis (for high explosive modelling, is typically 0)

   (e) `W:` – enter in the initial velocity along the z axis (for high explosive modelling, is typically 0)

   (f) `File:` – the name of the file describing the initial bomb geometry in the computational domain.

   The file is assumed to be in the same directory as the executable `octvce.exe`. It is exactly in the same format as those `.vtp` files describing individual solid objects as discussed in §3.2.

```
INITIAL CONDITION DATA
----------------------

Ambient conditions
------------------
Pressure: 101325
Density: 1.18472745130136
U: 0
V: 0
W: 0

Products conditions
-------------------
Pressure: 1.18623322568660e+10
Density: 1.65058278219220e+03
U: 0
V: 0
W: 0
File: ic.vtp
```

Figure 3.5: Initial condition file (2 species)

3.4 Gas model input file

The gas model input file specifies to OctVCE what gas models to use for the ambient and explosive products gases (see §3.3). The ambient gas is always modelled with ideal gas equation of state $p = \rho e\,(\gamma - 1)$, but the explosive products can be modelled with the ideal gas, JWL or JWLB equations of state (see §1.5).

As with §3.3, this file must be written in **exactly the same format** as the template in fig 3.6 (a template can also be found in `Octvce3.5.*/tests/ov_gas.par`). Note that **any dimensional parameters are assumed to be in SI units**. A description of this file (along with some advice on what to input) will be given below.

1. `No. species:` – options are `1`, `2`

   The number of gas species in the simulation. Typically for high explosive modelling it's 2 (the ambient and explosive products gases).

2. `Ambient gas Cv:` – enter in the specific heat at constant volume for the ambient gas

3. `Ambient gas gamma:` – enter in the ratio of specific heats for the ambient gas

   Note – if only 1 species (i.e. the ambient gas) is used all following lines are ignored as this information is sufficient

4. `EOS type for products:` – options are `Ideal`, `JWLB`

   Recall from §1.5 that the JWL equation of state is a simplified form of the JWLB one

5. `Products gas Cv:` – enter in the constant specific heat at constant volume for the explosive products

   Usually C_v is calculated so as to match a known pressure and temperature condition. For example, if the JWL equation of state is used, eqn 1.3 can be used to back-calculate C_v if the initial temperature and pressure of the explosion is known (recall §1.4 and §3.3). Mader [12] provides initial detonation temperatures for some explosives.

6. `Products gas gamma:` – enter in the ratio of specific heats of the explosion products (if the ideal gas equation of state is used)

7. `If EOS is JWLB - want to treat low products density as air?` – options are `y` or `n`

   To save computation time, regions where density of the explosive products is very low can be just regarded as air if `y` is selected. This is usually the preferred option.

8. `If EOS is JWLB - EOS coefficients are` – enter in all JWL/JWLB constants in the following lines

   Using eqn 1.1 as reference, `A1` for example corresponds to A_1, `R1` to R_1, `AL1` to A_{λ_1}, `BL1` to B_{λ_1}, `RL1` to R_{λ_1}, `OMEGA` is ω and `v0` is $1/\rho_0$. Recall from §1.5 that A_i and C are dimensional and here are in units of pascals, and v_0 is in units of m^3/kg.

   The example in fig 3.6 is the JWL equation of state where appropriately all A_i and R_i (for $i \geq 3$) and all A_{λ_i}, B_{λ_i} and R_{λ_i} must be set to 0. In general the JWL should be more than sufficient for most high explosive simulations since the focus is not really on near-field modelling anyway.

```
GAS PROPERTIES
--------------

No. species: 2
Ambient gas Cv: 717.5
Ambient gas gamma: 1.4

  If no. species > 1 ...
  ----------------------
  EOS type for products: JWLB
  Products gas Cv: 858.4798936

    If EOS is perfect gas ...
    -------------------------
    Products gas gamma: 1.66

    If EOS is JWLB - want to treat low products density as air? y
    -------------------------------------------------------------
    If EOS is JWLB - EOS coefficients are
    -------------------------------------
    A1: 5.24229e11
    A2: 0.076783e11
    A3: 0
    A4: 0
    A5: 0
    R1: 4.2
    R2: 1.1
    R3: 0
    R4: 0
    R5: 0
    AL1: 0
    AL2: 0
    AL3: 0
    AL4: 0
    AL5: 0
    BL1: 0
    BL2: 0
    BL3: 0
    BL4: 0
    BL5: 0
    RL1: 0
    RL2: 0
    RL3: 0
    RL4: 0
    RL5: 0
    C: 0.006651e11
    OMEGA: 0.34
    v0: 6.05846620229409e-04
```

Figure 3.6: Gas model file

3.5 Domain BC input file

The domain boundary condition file specifies to OctVCE the boundary conditions that need to be applied to the 6 faces of the root cell (recall the discussion in §1.3). This file must **follow the format** (with **correct spelling, captials where necessary** etc) of the template in fig 3.7 (a template can also be found in `Octvce3.5.*/tests/ov_BC.par`). Note that **any dimensional parameters are assumed to be in SI units**.

Recall that reflecting (or wall) boundary conditions are **implemented by the presence of a solid body**, so this file is really for all other boundary conditions. The manner of input to this file is somewhat different from other input files described so far, so it might be best to show how this file is used with the examples in fig 3.7 and fig 3.8. These two examples are probably the only ones that most users need to know.

3.5.1 Non-reflecting BCs

In fig 3.7 note that a `Non-reflecting` boundary condition in the "`BC type:`" field is applied to all the faces of the root cell (east/west boundaries are perpendicular to the x axis, north/south to the y axis and upper/lower to the z axis). For explosions modelling where boundaries remain 'open' to air, the non-reflecting option (based on Thompson's formulation [21]) is probably the most useful. The non-reflecting BCs nevertheless do cause weak reflections if post-shock flow is subsonic, so a perhaps a better strategy would just be to extend the domain far enough for the BCs to be a non-issue.

If a solid body obstructs a cell face on a domain boundary, then a solid wall boundary condition is applied there instead of whatever is written in this file for that boundary.

```
BORDER BOUNDARY CONDITIONS
--------------------------

East boundary
-------------
BC type: Non-reflecting

West boundary
-------------
BC type: Non-reflecting

North boundary
--------------
BC type: Non-reflecting

South boundary
--------------
BC type: Non-reflecting

Upper boundary
--------------
BC type: Non-reflecting

Lower boundary
--------------
BC type: Non-reflecting
```

Figure 3.7: Example domain boundary condition file

3.5.2 Inflow and outflow BCs

Fig 3.8 specifies a supersonic inflow coming from the west domain boundary with an extrapolated outflow at the east domain boundary (i.e. flow from left to right). The "`BC type:`" is now `Specific` as flow variables need to be set to a numerical value or to an `Extrapolated` value. The flow variables that need to be set are the pressure, density and velocity. A similar input can be made on the north, south, upper and lower domain boundaries.

As noted in §3.5.2 if a solid body obstructs a cell face on a domain boundary, then a solid wall boundary condition is applied there instead of whatever is written in this file.

```
BORDER BOUNDARY CONDITIONS
--------------------------

East boundary
-------------
BC type: Specific
Pressure: Extrapolated
Density: Extraplated
U: Extrapolated
V: Extrapolated
W: Extrapolated

West boundary
-------------
BC type: Specific
Pressure: 1e5
Density: 1.2
U: 1000
V: 0
W: 0

.
.
.
```

Figure 3.8: Fields for inlet and outlet BCs

3.6 'Detonation' data input file

As mentioned in §1.4 it is possible for 'detonation-like' modelling where ignition points are specified within a solid explosive for detonation waves moving outward from these locations to consume the whole explosive. The data for the ignition points and detonation velocities is specified in the detonation data file. It must be follow **exactly the same format** as the template in fig 3.9 (a template can also be found in `Octvce3.5.*/tests/detonations.par`).

Note that **any dimensional parameters are assumed to be in SI units**. The example in fig 3.9 should not be difficult to understand. The user inputs the number of ignition points in the mandatory "`No. detonation points:`" field. Then for each ignition point its 3D location and detonation wave radial velocity is written in the "`Detonation point:`" and "`Detonation velocity:`" field.

```
MORE SPECIFIC DETONATION INFO
-----------------------------

No. detonation points: 3

Enter in detonation points and velocities below
-----------------------------------------------

Detonation point: 1.924 0 0
Detonation velocity: 7980

Detonation point: 2 0 0
Detonation velocity: 7980

Detonation point: 2.076 0 0
Detonation velocity: 7980
```

Figure 3.9: Example detonation input file

3.7 Two-dimensional simulation

Although OctVCE is a 3D code, 2D planar and axisymmetric simulations in the xy plane can be performed by immersing most of the root cell in an environment of solid bodies (larger than the root cell) such that only one layer of cells in the z plane is not completely immersed. How this is accomplished is discussed in §3.7.1 and §3.7.3.

3.7.1 Making gaps and axes

Referring to fig 3.10, a very thick wall (thicker than the root cell edge) is needed and placed at $z = t$. The 'lower' z face of the root cell (see §3.5) **must be at** $z = 0$. Later on some command line arguments to OctVCE will be given to let it know a 2D simulation is desired, so the user need not worry about flow in the z direction. For axisymmetric calculations, OctVCE assumes the x axis is the symmetric axis and the y axis is the radial axis.

t is set equal to *half* the length of the highest refined cell, making all other leaf cells except those flush with the wall completely immersed, thus giving only one layer of computational cells in the xy plane. t must be set to this value so that all cells including the smallest ones can have at most two neighbours at a face (consistent with the 2D *quadtree* equivalent of 3D octree adaptation).

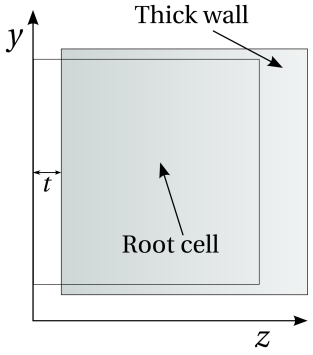


Figure 3.10: View from yz plane

3.7.2 Caution on axisymmetric computation

The inherent nature of VCE means that in axisymmetric geometry there will be some source of conserved quantities produced at intersected cells. This is because the approximate surface representation gives a cell volume (per radian) and area (per radian) and surface normals that are not completely correct. It might be good to run the simulation in quiescent flow with the same geometry to assess the influence of these source terms. However, it has been found in practice that these effects are typically quite minor compared to stronger flowfield features, especially that resulting from blast. These effects have also been shown to be small in simple conical supersonic flow [19] where derivation of key quantities of pressure and force are concerned.

3.7.3 Appropriate area subcell number

It is also important to ensure that the number of area subcells along a cell edge (ns_f in §3.1) are high enough for t to be computed as non-zero (at least one area subcell for the coarsest cell must exist in this gap). It is recommended that the ns_f not be lower than 32 for 2D simulations. If there are 5 levels of refinement (the difference between maximum and minimum refinement levels), ns_f must be 64, and if 6 levels of refinement, ns_f must be 128 etc.

Chapter 4

Running a simulation

Remembering that all input files have to be in the same directory as the executable `octvce.exe`, the code is then run using the arguments –

`./octvce.exe -gas` <*gas model file*> `-bc` <*boundary condition file*> `-ic` <*initial condition file*> `-par` <*general parameters file*> `-geom` <*VTK file of all bodies*> `-ncpus` <*no. cpus*> [`-deton` <*detonations file*>] [`-2D`] [`-2D=axisymmetric`] [`-switch-order` <*switch time*>] [`-output-grid`] [`-output-grid=last`] [`-continue-soln` <*mesh file*>] [`-count=energy`]

The arguments **in square brackets are optional**; others are mandatory. Their description will be given below.

1. `-gas` <*gas model file*>

   Enter in the filename of the gas model input file of §3.4

2. `-bc` <*boundary condition file*>

   Enter in the filename of the domain BC input file of §3.5

3. `-ic` <*initial condition file*>

   Enter in the filename of the IC input file of §3.3

4. `-par` <*general parameters file*>

   Enter in the filename of the general simulation parameters input file of §3.1

5. `-geom` <*VTK file of all bodies*>

   Enter in the filename of the `.pvtp` file describing the environment of all solid objects (see §3.2 and the example in fig 3.3)

6. `-ncpus` <*no. cpus*>

   Enter in the number of CPUs/threads for parallel processing (code must be compiled with an OpenMP compiler). For no parallel processing enter in '1' (obviously).

7. `-deton` <*detonations file*>

   An optional argument. If 'detonation-like' modelling desired, enter in the filename of the detonation data input file of §3.6.

8. `-2D`

   An optional argument, but use if 2D planar flow is being modelled. Refer to §3.7 for details on how to set up the simulation for general 2D simulations.

9. `-2D=axisymmetric`

   An optional argument, but use if 2D axisymmetric flow is being modelled. Refer to §3.7 for details on how to set up the simulation for general 2D simulations.

10. `-switch-order` <*switch time*>

    An optional argument. Sometimes even using a single limiter in a higher-order scheme (as discussed in §3.1) is not enough to keep the solution stable (from experience, this might happen if the bomb volume is resolved with only a few cells). As a last resort, this option tells OctVCE to employ a first-order scheme until *switch time*, after which the second-order scheme is resumed.

    Rose [16, p.g. 61] recommends a scaled switching time of around 1.2×10^{-3} s/kg$^{1/3}$ for the explosive mass.

11. `-output-grid`

    An optional argument. Will output the whole mesh (tree structure, flow state etc) at the same frequency as solution files are being output. Mesh files have a `.ov_out` extension. The simulation can be continued using these solution files using the `-continue-soln` argument (see below).

12. `-output-grid=last`

    An optional argument. Like `-output-grid` of above, but only outputs mesh file at the end of the simulation.

13. `-continue-soln` <*mesh file*>

    An optional argument. Continue the simulation from the mesh file with the `.ov_out` extension generated using the `-output-grid` option. The initial condition data from `ov.par` is ignored.

14. `-count=energy`

    An optional argument. Used for high explosives modelling, where the total energy and volume of all cells in the initial 'bomb' volume (see §1.4) is counted and output. The program then exits.

    This is handy as the cartesian cell 'staircasing' representation of the initial bomb volume when the initial grid is generated may give total explosion energies that are slightly too high or low. The information output can be used to ensure a better energy match by adjusting bomb volume and/or gas condition.

As the simulation is run some data (like timestep, step number, CFL, no. cells refined/coarsened etc) will also be output on the screen.

4.1 Memory usage

OctVCE is quite a memory-intensive code, with total memory used per cell during a simulation estimated at around 4 kb. For this reason in large simulations it's necessary to run this code on shared-memory platforms, but this is recommended anyway for obtaining solutions in a reasonable timeframe.

4.2 Grid convergence and/or error estimation

As with any CFD method where discretization errors exist, it is generally recommended to do a grid convergence study involving a minimum of two different grids (three preferrably). In this case, it can typically involve monitoring behaviour at a pressure trace or group of traces as the grid is refined. A converging series of traces can be taken as indicative of a converging solution in general.

This allows an estimation of error based on Richardson-extrapolation (see [15, 17]) and possibly a more accurate estimate of the true *numerical* solution, provided the grids are fine enough for convergence of the solution to be observed. How good the true numerical solution corresponds to the *actual*, or real-world (experimental) solution is a separate validation exercise (several of which are currently being undertaken by the author).

Generalized Richardson-extrapolation predicts the exact solution f_{exact} to be a function of the finer grid solution f_1, coarser grid solution f_2, grid refinement factor r, and order of the scheme p as follows in eqn 4.1.

$$f_{exact} = f_1 + \frac{f_1 - f_2}{r^p - 1} \tag{4.1}$$

It can be also used to predict quantites derived from the solution e.g. surface force. The second term in eqn 4.1 can be thought of as the 'error' as it represents an additive correction to f_1, and is a reasonable approximation to the true numerical error when it is much smaller than f_1.

As OctVCE only allows integer halvings of cell sizes, $r = 2$ nominally. This inherently assumes that the solution on an adapted mesh is just as good as the solution on a uniform mesh with the same minimum cell size, which is not always true in general. p can be between 1 and 2 depending on what quantity is being measured e.g. peak overpressure would be best with $p = 1$ as it's a shock-dependent quantity and the present scheme is first order at shocks. In smoother flow regions, p can be nominally 2 as OctVCE is a nominally second-order method (though a value between 1 and 2 might be more reasonable due to the presence of limiters and other numerical effects that can hamper the solution order).

Chapter 5

Post-processing

5.1 Solution file format

Solution files are also in the VTK XML version 4.2+ ASCII file format (see §3.2) but will have a name of *basename*`.`$\star$`.vtu` and/or *basename*`.`$\star$`.pvtu`, where *basename* is the base solution file name (see §3.1) and $\star$ the timestep at which the solution file was output.

If the code was run in serial the solution files will only have the `.vtu` extension. But if the code was run in parallel on n CPUs, there will be n `.vtu` files corresponding to the mesh portions allocated to each CPU. All these files are listed in the `.pvtu` file to comprise the entire mesh domain (recall the discussion on parallel VTK files in §3.2). For later visualization it is thus the `.pvtu` file that needs to be opened.

5.2 Visualizing solution files

The solution files can only be opened with a VTK file visualizer (obviously). Two currently existing VTK visualizers are the Mayavi[1] and Paraview[2] applications. Paraview seems to be more efficient for visualizing very large solution files. Both these applications can display the solution on a 2D plane (which is usually desired).

5.2.1 A note on displaying 2D solutions

Recall the disussion on simulation set-up for 2D simulations in §3.7, where the rest of the root cell is 'trimmed' so that only one layer of cells on the z plane are actually in the flow solution. However these cells are nonetheless still three-dimensional.

To fully display the 2D simulation, a planar cut (in the visualizer) must be made such that the plane passes through *all* cells in this layer (even the finest ones). If this does not happen there will appear 'gaps' in the solution. So if the lower boundary of the root is at $z = 0$, a z-plane cut

[1] http://mayavi.sourceforge.net/
[2] www.paraview.org

at some very small number (say 10^{-10}) should pass through all cells and the whole 2D solution on this plane will be displayed.

5.3 Pressure trace format

When pressure traces/histories at a gauge location are recorded (see §3.1), the pressure trace file for each location records 2 quantities – (i) the time and (ii) the *overpressure* (pressure minus ambient pressure). It has the name of *basename*$\star$`.dat`, where *basename* is the base history filename and $\star$ the trace number (starting at 0), corresponding to the order in which the gauge locations were entered in general simulation parameteers input file (§3.1).

The file can be read and plotted by gnuplot[3]. The trace can yield impulse data by for example placing the columns in an array and writing an Octave[4] script to integrate the pressure trace $\int Pdt$. However as trace files can get quite large, it might be better to write an integration routine in a more efficient language like C or FORTRAN.

[3]http://www.gnuplot.info/
[4]http://www.gnu.org/software/octave/

Appendix A

A useful 1D code

This section describes a useful stand-alone 1D code that has been developed alongside OctVCE for general blast modelling work. The 1D code `blast_1D` is much like OctVCE but much simpler to use and faster to run. It can be run assuming planar, cylindrical or spherical symmetry. The 1D assumption means that the bomb can be thought of as a *driver gas* (as in a shock tube) which is initialized to the proper explosive conditions. Like OctVCE, it is a nominally 2nd-order code that uses much of the same underlying numerical methodology.

The code is located in the `blast_1D` directory. The code is compiled in the same way as OctVCE is (see §2) using the *makefile* in the `blast_1D/unix/` directory. The default directory where the executable `b1d.exe` is placed is in `blast_1D/tests/`.

Like OctVCE, several input files are required for `blast_1D` to run. Templates for these input files can be found in `blast_1D/tests/`. They are `b1d_gas.par` (gas model input file), `b1d_ic.par` (IC input file) and `b1d.par` (general simulation parameters input file). Like OctVCE's input files, the formats of these files cannot be altered (only the parameters) and they must be in the same directory that the executable `b1d.exe` is in.

The gas model input file is identical to OctVCE's gas model input file (see §3.4). The initial condition input file is very simple and simply requires one to specify the initial ambient and driver gas conditions (fig A.1), like with OctVCE's input file.

```
Ambient IC
----------
Pressure: 101325
Density: 1.2

Driver IC
---------
Pressure: 9e9
Density: 1650
```

Figure A.1: IC input file for 1D code

The general simulation parameters input file format can be seen in fig A.2. A description of each input line is given below.

```
INPUT FILE FOR BLAST_1D
-----------------------

Type of geometry = 3
Domain length = 2
Contact surface at x = 5e-2
No. cells = 2000
Run until time = 2e-3
Max CFL = 0.5
Num timesteps dump data = 1000

No. history locations = 2
Dump history locations every how many time steps: 2

History locations
-----------------
0.1
0.2
```

Figure A.2: General parameters file for 1D code

1. `Type of geometry =` – enter `1` for planar geometry, `2` for cylindrical, and `3` for spherical geometry.

2. `Domain length =` – enter in length of domain. Bomb is located at the left end; the right end is set to nonreflecting outlet.

3. `Contact surface at x =` – enter in location of interface between driver and ambient gas. For a 1D spherical bomb calculation, this is just the radius of the charge.

4. `No. cells =` – enter in the number of cells.

   Note that it's important to specify a domain length and number of cells so that there are an integer number of cells within the charge, thus matching bomb energy exactly.

5. `Run until time =` – enter in time where simulation finishes

6. `Max CFL =` – enter in maximum CFL; usually 0.5 is a good number.

7. `Write data frequency =` – output frequency (with respect to solution time) of solution.

   The solution files will have a `s.`$\star$`.dat` name where $\star$ is the timestep at which the solution was output. The columns of this `.dat` file consist of cell location, density, pressure, products mass fraction and flow velocity, and it can be easily plotted by gnuplot[1].

8. `No. history locations =` – the number of pressure gauges desired

9. `Dump history locations every how many time steps =` – how often should the pressure at a point be recorded. Typically recording every 2 to 5 timesteps is enough.

   At each line below the `History locations` line, enter in the gauge location (along the x-axis, as this is a 1D case) where the pressure trace is recorded. Each trace file has a `hist_`$\star$`.dat` name where $\star$ is the trace number (starting at 0) corresponding to the order in which the gauge locations were entered.

[1] http://www.gnuplot.info/

Appendix B

Example simulations

This section will cover some example simulations which are included with the source code in `Octvce3.5.⋆/tests/` (all geometry and input files are provided). Additional explanations are provided where appropriate. It is hoped that these examples will help the user get started easily and relatively quickly with the OctVCE code (though reading the material in §3 to §4 is also recommended, of course). Some results will also be provided.

B.1 Supersonic conical flow

Here supersonic flow over a cone with a 20° half-angle is simulated (post-shock $M_\infty = 1.5$, whilst shock $M = 3.66$). This is a 2D axisymmetric simulation. The geometry and input files are in `Octvce3.5.⋆/tests/cone_eg/`.

B.1.1 Cone geometry

As this is an axisymmetric simulation, the geometry must be constructed in the manner described in §3.7. The x axis is the axis of symmetry and y the radial co-ordinate.

The geometry files for individual bodies are `ceiling.vtp` (to prevent flow out the top y boundary), `cone_20.vtp` (the actual cone geometry) and `floor.vtp` (to prevent flow out the bottom y boundary). Note how the ceiling and floor very slightly overlap the domain (recall the discussion in §3.2.1).

Finally the required 'thick wall' (to ensure only 1 layer of cells in the z-plane – see §3.7.1) is file `right_wall.vtp`. All these bodies are listed in the parallel VTK file `bodies.pvtp` describing the environment. The whole geometry is displayed in fig B.1.

B.1.2 Input files for cone simulation

All input files are located in `Octvce3.5.⋆/tests/cone_eg/*.par`.

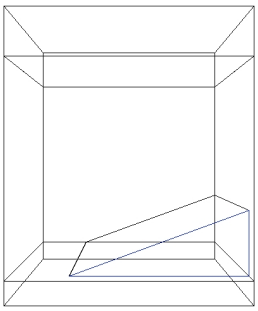


Figure B.1: Geometry for cone simulation

General simulation parameters file

This file has been called `ov.par`.

IC input file

This file has been called `ov_IC.par`.

Note the IC input file has 2 initial conditions for the ambient and 'explosive products' gases. However in this case it is better to think of the 'explosive products' gas as simply the gas at *post-shock* conditions. Rather than staring the simulation with flow coming in from the left boundary, some computation time can be saved if the initial conditions are setup so that the shock has already entered the domain but not passed the cone.

This approach is also advantageous if mesh refinement at the shock is desired (though this isn't done here), as if the simulation starts by having the shock enter the domain, it isn't possible under the current input method for the mesh to be refined to the maximum level around the shock if no post-shock initial condition already exists *in the domain*. As discussed in §3.3 it is necessary to have a file describing the geometry of the bomb volume (or in this case, the post-shock flow), and this is given in `ic.vtp`.

Gas model input file

This file has been called `ov_gas.par`. Note only 1 (ideal-gas) species is necessary (though there are 2 regions where the initial pressures, densities and velocities are different – see above).

Domain BC input file

This file has been called `ov_BC.par`. As this simulation has supersonic inflow coming from the west (or left) with extrapolated outflow at the east (or right) boundary, the domain BC input file follows nearly exactly format as discussed in §3.5.2 (with only the values different). It is necessary to enter in 'boundary conditions' for the other root cell faces, even though solid wall boundary conditions are set for them.

B.1.3 Running the cone simulation

Now that all input files have been appropriately created, the code is run with the command line arguments as shown below (recall the discussion in §4). As this code was run on a 2 CPU SMP, 2 CPUs have been used.

```
time ./octvce.exe -gas ov_gas.par -bc ov_BC.par -ic ov_IC.par
-par ov.par -geom bodies.pvtp -ncpus 2 -2D=axisymmetric
```

Figure B.2: Running the cone simulation

Some screen output from the first 2 timesteps is shown below. Note that at present adaptation in parallel for 2D simulations isn't done as total numbers of cells aren't very large, giving relatively small benefit from the extra work (recall the discussion in §3.1). The number of computational cells is 52555.

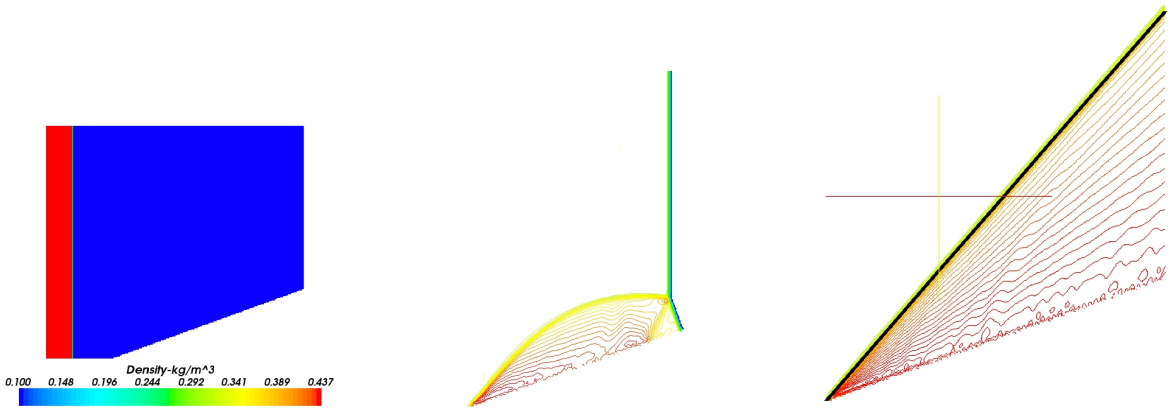
```
Beginning time integration...
Now entering 2D axisymmetric mode.  Please ensure gap thickness = 0.5 of length of
highest refined cell, and please ensure you are working in XY plane and X is symmetry
axis and Y is radial coordinate
For 2D modes, your gap thickness should thus be 1.953125e-03.  Please ensure this
is correct
read_ov_param(): Warning.  As you're not doing adaptive, level of IC cell
intersected = no. times to refine root initially
octvce(): Warning - in 2D modes, can't adapt in parallel because I can't be bothered.
But you still get parallel flow solution and I/O
Step 1, time 5.320191e-08 s, dt 5.320191e-08, CFL 7.150448e-02, num_cells 52555
Step 2, time 1.333144e-07 s, dt 8.011254e-08, CFL 1.076750e-01, num_cells 52555
.
.
.
```

Figure B.3: Some screen output from cone simulation

B.1.4 Some results for cone simulation

The initial solution is shown in fig B.4(a) where the 2 different initial conditions (for ambient and post-shock flow conditions) can be easily seen. Fig B.4(b) shows the density contours at 0.5 ms before the shock has exited the right boundary. It can be seen that the reflected shock, Mach stem and contact discontinuity are captured quite well.

In the steady state limit (fig B.4(c)) one can see some 'noise' existing at the surface. This arises from the inherent nature of VCE in approximating surfaces (a more detailed disucssion is given in [19]). However the solution should be sufficient for practical engineering purposes, with the shock angle agreeing very well with the theoretical result (the black line).



(a) Initial solution (b) Density contours at 0.5 ms (c) Steady-state result

Figure B.4: Density contours

B.2 Shock diffraction over blast wall

This test case attempts to duplicate Chapman's [7] axisymmetric simulation of a blast wave propagating over a blast wall or barrier. The charge of 60 g of TNT is detonated at a 0.15 m height, 1.05 m from the the target structure (behind the blast wall) where the pressure trace at 0.15 m height will be recorded (see the initial grid in fig B.8(c)). In this simulation the domain is identically 1.05 m in length. This size should be sufficient to produce the main rise and decay of the overpressure before effects from the non-reflecting boundary affect the solution.

The geometry and input files are in `Octvce3.5.⋆/tests/smith_barrier_eg/`.

B.2.1 Blast barrier geometry

As this is an axisymmetric simulation, the geometry must be constructed in the manner described in §3.7. As the x axis is the axis of symmetry (in this case the height) and y the radial co-ordinate, the whole geometry must be constructed such that the *west* boundary at $x = 0$ is the true 'ground'.

A number of the geometry files have names corresponding to those of cone simulation (§B.1.1) and their function is similar. The file `back_wall.vtp` corresponds to the true 'ground' (preventing flow out the west boundary). Note the slight overlap in geometry and domain as discussed in §3.2.1 and §B.1.1. The whole geometry is displayed in fig B.5.

B.2.2 Input files for blast barrier simulation

All input files are located in `Octvce3.5⋆/tests/smith_barrier_eg/*.par`.

General simulation parameters file

This file has been called `ov.par`.

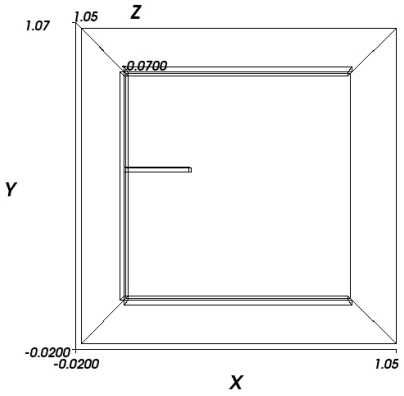


Figure B.5: Geometry for blast barrier simulation

IC input file

This file is called `ov_IC.par`.

As this is a high explosives simulation the initial condition for the explosive products has been initalized in the manner described in §1.4, using Lee's JWL values for TNT [10]. The geometry file for the spherical charge is `sphere.vtp` (centered at $x = 0.15$ m, $y = 0$). Because of the axisymmetry, the actual bomb volume is a semi-circle.

Gas model input file

This file is called `ov_gas.par`. 2 species are necessary for the ambient and explosive products gases. The JWL values from Lee's paper [10] for TNT have been entered (in SI units).

Domain BC input file

This file is called `ov_BC.par`. It is identical to the one in fig 3.7 as non-reflecting conditions need to be placed on all boundaries (where they aren't blocked by walls).

B.2.3 Running the blast barrier simulation

Before running the simulation it is important to count the total blast energy to ensure it matches the theoretical value as close as possible (recall the discussion in §1.4). If the charge consisted of solid TNT then using according to the JWL parameters it would have a radius of 2.0636×10^{-2} m and the energy for 60 g would be 4.1×10^4 J/rad (for a semi-circle). So the code is run and the output shown in fig B.7.

```
time ./octvce.exe -gas ov_gas.par -bc ov_BC.par -ic ov_IC.par
-par ov.par -geom bodies.pvtp -2D=axisymmetric -ncpus 1
-count=energy
```

Figure B.6: Running the blast barrier simulation

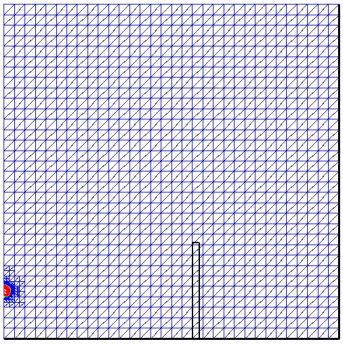
```
Now entering 2D axisymmetric mode.  Please ensure gap
thickness = 0.5 of length of highest refined cell, and
please ensure you are working in XY plane and X is symmetry
axis and Y is radial coordinate
For 2D modes, your gap thickness should thus be 5.126953e-04.
Please ensure this is correct

octvce(): Here are the results from energy counting ...
For IC region 0, total no. cells is 160, total energy
is 4.11379124e+04 J/rad, tot vol. is 5.87684463e-06 m^3/rad
```

Figure B.7: Energy counting results for blast wall simulation

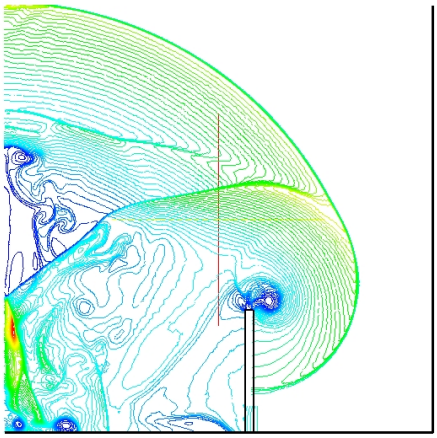
The energy count gives 4.114×10^4 J/rad (there were 160 cells representing the bomb volume). The sphere radius used in `sphere.vtp` was 2.018×10^{-2} m, which is slightly smaller than theoretical whilst preserving the same shape. As discussed in §1.4.1 it is possible to use the total volume of cells within the bomb (here it is 5.877×10^{-6} m^3/rad) to adjust the initial bomb condition for an exact energy match, but this strategy isn't implemented here. When one is satisfied with the energy match, the code can be run without the `-count=energy` argument.

B.2.4 Some results for blast wall simulation

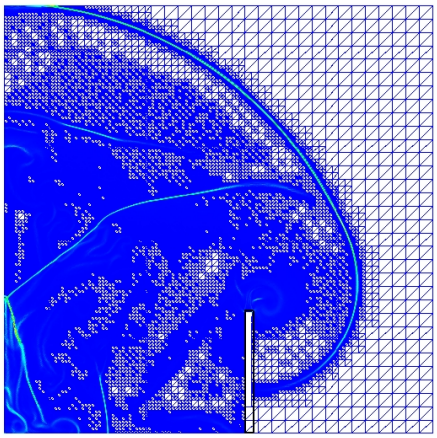
The initial grid is seen in fig B.8(a) (the figure has been inverted so that the ground is at the 'bottom'). The density contours and grid at 1 ms are shown in figs B.8(b) and B.8(c). Note in fig B.8(d) that at 2 ms the primary shock has exited the domain, but no reflections exist because of the non-reflecting boundary condition. The pressure trace at the gauge location is shown in fig B.8(e) and compared to Chapman's values.



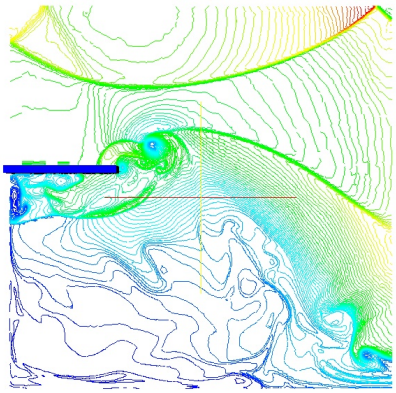
(a) Initial grid



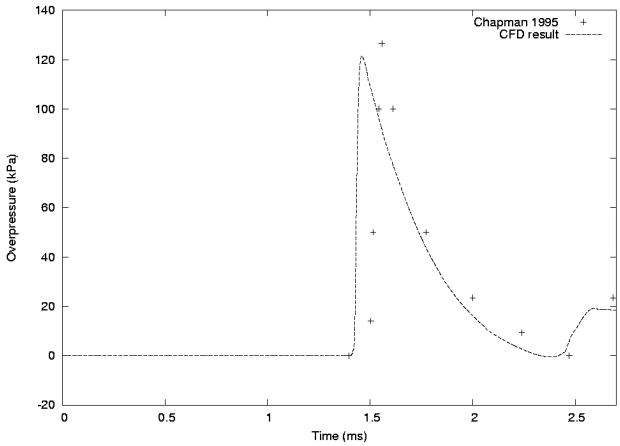
(b) Density contours at 1 ms



(c) Grid and schlieren at 1 ms



(d) Density contours at 2 ms



(e) Pressure trace

Figure B.8: Blast wall results

B.3 Blast over 3D obstructions

This 3D test case attempts to duplicate Sklavounos' investigation [18] on blast wave effects with concrete blocks. Sklavounos appears to model the blast using a finite energy release rate which releases a total energy of 1.908×10^6 J, or 444 g TNT equivalent. This simulation takes advantage of the symmetry plane through the center of the blocks. This simulation was terminated before any waves passed out of the domain. A root cell length of 10 m was used (this should be enough before any boundary effects to intefere the trace readings).

The geometry and input files are in `Octvce3.5.*/tests/rigas_eg/`.

B.3.1 Obstruction geometries

Explanation on the nature of some VTK files is necessary. As the explosion takes place at the lower left corner of the domain, the file `right_wall.vtp` places a wall at $y = 0$ to give the symmetry plane throught the center of the blocks. However the file `back_wall.vtp` also places a symmetry plane at $x = 0$ to save computation time (the reflection from this boundary won't affect the initial gauge overpressure readings).

The `ceiling.vtp` and `left_blocker.vtp` files reduce the domain size along the y and z axes as reflections from them will arrive far too late to affect initial gauge readings. However the domain length along the x axis is kept at 10 m and a non-reflection boundary condition is active at the east face of the root cell.

B.3.2 Input files for 3D obstruction simulation

The input files are in `Octvce3.5*/tests/rigas_eg/*.par`.

General simulation parameters file

This file is called `ov.par`.

Note that adaptation indicator type 3 is used (which is just indicators 1 and 2 used together). From §3.1 it was recommended that indicator type 1 vary from 0.01 to 0.05, but here it is 0.005. This value was chosen as the domain is relatively large compared to the initial explosive, and particularly after the blast diffracts over the obstacles an indicator value greater 0.005 won't refine any cells around the shock.

IC input file

This file is called `ov_IC.par`. TNT has been used as the explosive, so this file is very similar to the one in §B.2.2. Note that because there are 2 symmetry planes and the assumption that the charge is at ground level, the bomb volume is actually a quarter hemi-sphere (i.e. a sphere octant). The theoretical energy in this volume should be 4.77×10^5 J, but at the given resolution the best match is 5.215×10^5 J (with a smaller sphere radius).

Gas model input file

This file is called `ov_gas.par`. It is also exactly identical to gas model input file in §B.2.2.

Domain BC input file

This file is called `ov_BC.par`. It is also exactly identical to gas model input file in §B.2.2.

B.3.3 Running the 3D obstruction simulation

Recall §B.2.3 where the total energy and bomb volume are first counted in an attempt to match best the theoretical total energy. Once an appropriate energy match is achieved the simulation is finally run (here it was using 8 threads) with the command line arguments

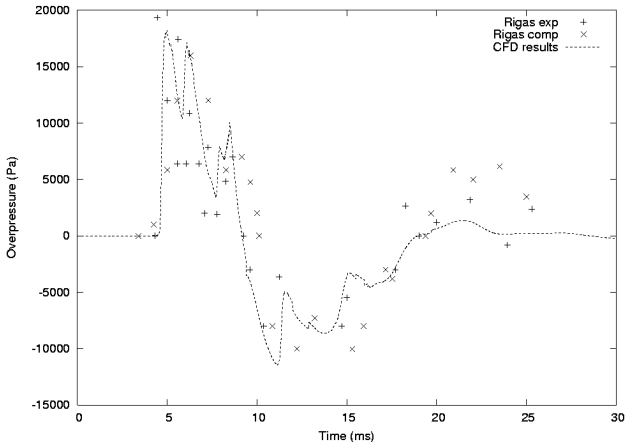
```
time ./octvce.exe -gas ov_gas.par -bc ov_BC.par -ic ov_IC.par
-par ov.par -geom bodies.pvtp -ncpus 8
```

Figure B.9: Running the 3D obstruction simulation

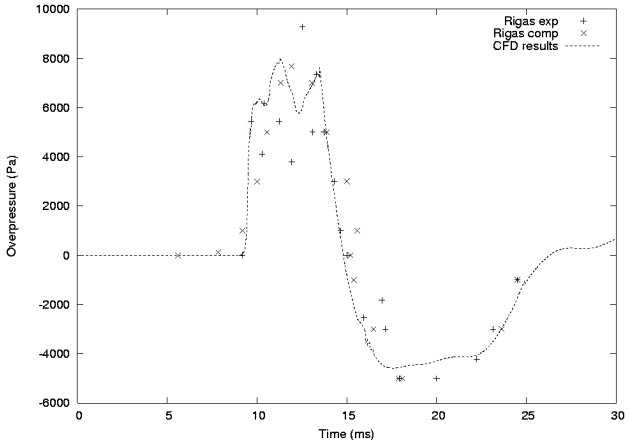
B.3.4 Some results for 3D obstruction simulation

Fig B.10(a) shows the pressure trace at gauge 1 of Sklavounos' simulation (this gauge is positioned between the first and second concrete blocks). His computational and experimental results are also shown for comparison (they had to be estimated by eye). It can be seen that agreement is quite good with the expectd first three peaks being the incident shock and the reflected shocks from the ground and the second concrete block.

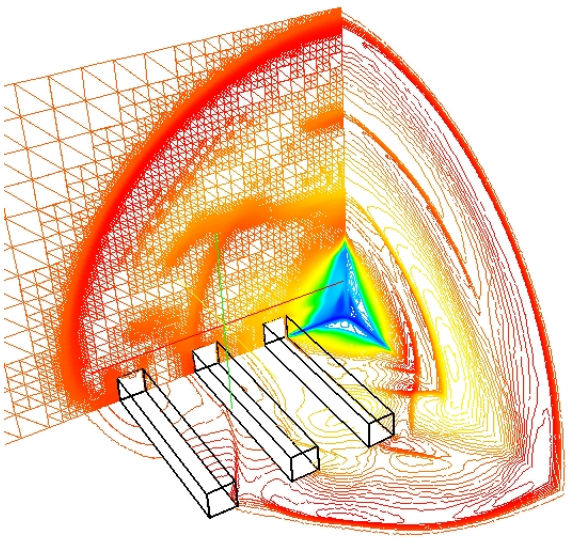
Fig B.10(b) is the pressure trace at gauge 2 (between the second and third concrete blocks). As with the gauge 1 trace the agreement in peak overpressure and general waveform profile is good, although here the waveform is more complex given the reflections that have occured before the blast reached this gauge. Fig B.10(c) shows the mesh and density contours on planes intersecting the blast centre.



(a) Trace at gauge 1



(b) Trace at gauge 2



(c) Density contours

Figure B.10: 3D obstruction results

Bibliography

[1] J G Anderson, G Katselis, and C Caputo. Analysis of a Generic Warhead Part I: Experimental and Computational Assessment of Free Field Overpressure, 2002. DSTO Weapons Systems Division, Australia, DSTO-TR-1313.

[2] L S Avila. *The VTK user's guide*. Kitware, Inc., Clifton Park, NY, 2004.

[3] E L Baker and D L Littlefield. Implementation of a High Explosive Equation of State into an Eulerian Hydrocode. In *AIP Conference Proceedings*, volume 706, pages 375–378, 2004.

[4] E L Baker and L I Stiel. Improved Quantitative Explosive Performance Prediction Using Jaguar. In *Insensitive Munitions and Energetic Materials Technology Symposium, Tampa, FL*, 1997.

[5] T J Barth. On Unstructured Grids and Solvers, 1990. von Karman Institute for Fluid Dynamics, Lecture Series 1990–03.

[6] OpenMP Architecture Review Board. OpenMP C and C++ Application Program Interface, 2002. Version 2.0.

[7] T C Chapman, T A Rose, and P D Smith. Blast wave simulation using AUTODYN2D: a parametric study. *International Journal of Impact Engineering*, 16(5/6):777–787, 1995.

[8] E F Charlton. *An Octree Solution to Conservation–laws over Arbitrary Regions (OSCAR) with Applications to Aircraft Aerodynamics*. PhD thesis, The University of Michigan, 1997.

[9] A M Landsberg and J P Boris. The Virtual Cell Embedding method: a simple approach for gridding complex geometries, 1997. AIAA–97–1982.

[10] E L Lee, H C Horning, and J W Kury. Adiabatic Expansion of High Explosive Detonation Products, 1968. University of California Report No. UCRL–50422.

[11] M S Liou and C J Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107:23–39, 1993.

[12] C L Mader. *Numerical modeling of explosives and propellants*. CRC Press, Boca Raton, 1998.

[13] D I Pullin. Direct Simulation Methods for Compressible Inviscid Ideal-Gas Flow. *Journal of Computational Physics*, 34(2):231–244, 1980.

[14] J J Quirk. A Contribution to the Great Riemann Solver Debate. *International Journal for Numerical Methods in Fluids*, 18(6):555–574, 1994.

[15] P J Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico, 1998.

[16] T A Rose. *An Approach to the Evaluation of Blast Loads on Finite and Semi-Infinite Structures*. PhD thesis, Cranfield University, Engineering Systems Department, 2001.

[17] C J Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205:131–156, 2005.

[18] S Sklavounos and F Rigas. Computer simulation of shock waves transmission in obstructed terrains. *Journal of Loss Prevention in the Process Industries*, 17:407–417, 2004.

[19] J Tang. A simple axisymmetric extension to Virtual Cell Embedding, 2007. Submitted to the International Journal for Numerical Methods in Fluids.

[20] J Tang. Theory manual to OctVCE – a cartesian cell CFD code with special application to blast wave problems, 2007. University of Queensland Mechanical Engineering Report 2007/12.

[21] K W Thompson. Time-Dependent Boundary Conditions for Hyperbolic Systems, II. *Journal of Computational Physics*, 89(2):439–461, 1990.

[22] E Timofeev, P Voinovich, and K Takayama. Adaptive Unstructured Simulation of Three–Dimensional Blast waves with Ground Surface Effect. In *36th Aerospace Sciences Meeting and Exhibit*, 1998. AIAA 98–0544.

[23] Y Wada and M S Liou. A Flux Splitting Scheme with High-Resolution and Robustness for Discontinuities, 1994. AIAA 94–0083.