Chapter 3

Flow Integration

We now describe a time accurate method for integrating the discretized flow solution contained by an arbitrary set of meshes which conform to our hierarchical grid system. The methodology behind this integration process has been developed specifically to circumvent three problems that affect many mesh refinement schemes. Briefly, these problems are: one, for time accurate calculations the computational effort grows exponentially with increasing mesh refinement, thus making very high resolution calculations prohibitively expensive; two, the method of integration is inextricably linked to the method of spatial refinement, thus making it difficult to take advantage of any basic *flow-algorithm* developments that appear in the literature; three, the scheme is not formally conservative, thus making it unsuitable for the computation of flows containing strong shocks. The presentation of material in this chapter reflects the *bottom-up* design procedure used to develop the integration process. First, a separate description is given for each of the major components that form the integration process. These descriptions are then followed by details of how the separate components must be combined so as to form a general procedure capable of integrating any given set of meshes, provided of course that they are *properly nested*. Finally, the results for a series of validation problems are presented, these results demonstrate the suitability of our scheme for computing shock hydrodynamic flows.

3.1 Temporal Refinement

The majority of mesh refinement schemes give the impression of having been designed to minimize the number of grid cells that are required to compute a solution of a given resolution. This design philosophy is presumably based on the notion that the effort required to integrate the numerical flow solution decreases as the number of grid cells decreases. Applied to the design of schemes that compute solutions to steady flows this approach is not unreasonable. But such a philosophy is not necessarily suited to the

design of schemes that compute solutions to unsteady flows. Indeed, for many mesh refinement schemes the computational effort required to compute time accurate simulations grows exponentially with increasing mesh refinement, thus making very high resolution calculations prohibitively expensive.

The root cause of this inefficiency is the fact that in order to achieve time accuracy such schemes must integrate every cell with the same time step. Furthermore, to prevent non-linear wave interactions affecting the accuracy of the integration process, in general, no one cell can be integrated with a Courant number greater than one. Consequently, the rate at which the solution can be evolved is likely to depend on the finest mesh spacing. The smaller the finest mesh cell the more time steps that must be taken for every mesh cell in order to evolve the solution over some fixed period of time. Now, a better philosophy for the design of a time accurate scheme would be to minimize the total number of time steps, taken as a sum over all grid cells, required to integrate a flow solution over some fixed period of time. Adopting this philosophy leads to a scheme that refines in time as well as space. More, but smaller time steps are taken for fine cells than for coarse cells.

Berger & Oliger[9] were the first to recognize the need to refine in both time and space. But, the importance of the temporal refinement was understated. Indeed, relative to unstructured grids the method of spatial refinement used by the AMR algorithm is inefficient. However, the hierarchical grid system facilitates the adoption of a temporal refinement strategy and it is the saving in computational effort resulting from this strategy that primarily accounts for the success of the AMR algorithm. The following example demonstrates the size of these savings.

Consider a one-dimensional grid conforming to our hierarchical grid system. Suppose the grid at level 0 consists of m equally spaced cells and the grid at level l covers p cells of the grid at level $l-1$. If the spatial refinement factor is the same for all grid levels, r say, then the complete grid contains some $m + r.p.l_{max}$ cells. Now, such a grid could be used to track an isolated discontinuity. Using a uniform time step the computational effort, $\mathcal{W}_u$, to follow the discontinuity over n cells of the grid G_0 is roughly proportional to,

$$n.r^{l_{max}} \left(m + r.p.l_{max}\right). \tag{3.1}$$

Alternatively, adopting a temporal refinement strategy where the time step at level l is r times smaller than that at level $l-1$ the computational effort, $\mathcal{W}_{tr}$, is

$$n\left[m + p.r\left(r + r^2 \ldots + r^{l_{max}}\right)\right],$$

which may be simplified to,

$$n\left[m + p.r^2\left(\frac{r^{l_{max}} - 1}{r - 1}\right)\right]. \tag{3.2}$$

By alternatively considering the extremes of large m and large l_{max} it may be shown that,

$$\left(\frac{r-1}{r}\right) l_{max} < \frac{\mathcal{W}_u}{\mathcal{W}_{tr}} < r^{l_{max}}. \tag{3.3}$$

So, the more ambitious the calculation the greater the computational saving. The ratios of computational effort, $\mathcal{W}_u/\mathcal{W}_{tr}$, for various values of m, r, l_{max} are shown in figure 3.1. Again, these show that the computational saving increases with increasing m, r or l_{max}.

$\mathcal{W}_u/\mathcal{W}_{tr}$ for $p = 2, r = 2$					
$m \backslash l_{max}$	1	2	3	4	5
10	1.6	2.1	2.7	3.2	3.7
100	1.9	3.5	5.7	8.4	11.0
1000	2.0	3.9	7.7	14.5	26.2

$\mathcal{W}_u/\mathcal{W}_{tr}$ for $p = 2, r = 4$					
$m \backslash l_{max}$	1	2	3	4	5
10	1.7	2.4	3.2	3.9	4.7
100	3.3	7.1	10.3	12.0	13.0
1000	3.9	14.0	39.2	71.0	89.4

Figure 3.1: Tabulated values of $\mathcal{W}_u/\mathcal{W}_{tr}$ for various m, r, l_{max}.

Now, if more, but smaller time steps are taken for fine mesh cells than for coarse mesh cells, how are the different rates of advance synchronized such that the overall evolution of the flow is time accurate? For many types of grid the short answer to this question is, with extreme difficulty! The degree of difficulty is reflected by the paucity of schemes that combine both spatial and temporal refinement. But, for the hierarchical grid system used by the AMR algorithm this synchronization is easily accomplished.

3.1.1 A Practical Scheme

The method of temporal refinement employed by the AMR algorithm is simple, but effective. A different time step is used for each level within the grid hierarchy. The time step used for the integration of the grid G_l, Δt_l, is related to that used for the coarser grid G_{l-1} by,

$$\Delta t_l = \frac{\Delta t_{l-1}}{\max(rI_l, rJ_l)}. \tag{3.4}$$

Choosing the time step in this manner ensures that the Courant number associated with the grid G_l is comparable to that associated with the grid G_{l-1}. The advance of the solutions at different grid levels are synchronized by recursively interleaving the integrations at different grid levels; details of why this ordering is desirable will be given later. For example, the set of grids shown in figure 3.2 would necessitate the sequence of integrations shown in figure 3.3. At first sight this ordering looks complicated, but the simple recursive procedure shown in figure 3.4 will perform the correct sequencing of grid integrations for an arbitrary number of grid levels. The procedure call,

$$\textbf{Sequence\_Grid\_Integrations}(0, \Delta t_0)$$

will perform a complete sequence of time steps such as that shown in figure 3.3.

l	rI_l	rJ_l
1	2	2
2	4	2

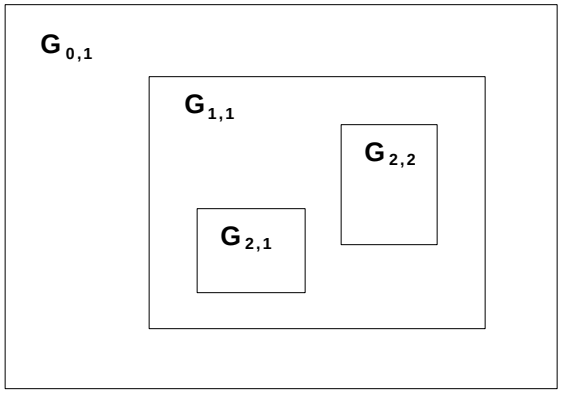


Figure 3.2: An example set of meshes.

GRID INTEGRATED	TIME STEP
G_0	Δt_0
$\quad G_1$	$\quad \Delta t_0/2$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad G_1$	$\quad \Delta t_0/2$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$
$\quad\quad G_2$	$\quad\quad \Delta t_0/8$

Figure 3.3: The sequence of grid integrations necessitated by the example set of meshes.

```
Procedure Sequence_Grid_Integrations(l,Δt_l)
  Integrate_Grid(l,Δt_l)
  if l < lmax {
    for time_step = 1 to max(rI_{l+1}, rJ_{l+1}) {
      Sequence_Grid_Integrations(l + 1,Δt_l/max(rI_{l+1}, rJ_{l+1}))
    }
  }
End Procedure
```

Figure 3.4: A Procedure to recursively interleave the grid integrations.

3.2 Boundary Procedures

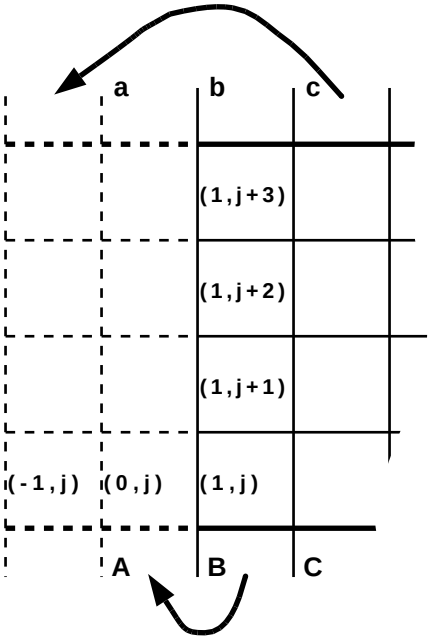
The procedure to integrate a grid must be able to process an arbitrary set of *properly nested* meshes. To this end, it is advantageous to split the integration process into two distinct parts. It being easier to design separate, foolproof procedures for these two parts rather than design a single, foolproof procedure to tackle the integration process directly. The dummy cells which surround each mesh are the key to this *divide and conquer* strategy. They effectively turn cell interfaces along mesh boundaries into internal interfaces. Prior to integrating a grid, the dummy cells for every mesh contained by the grid are primed with data. Each mesh is then processed independently of every other mesh by some mesh integrator which never actually sees a mesh boundary. This is possible because the data used to prime dummy cells is chosen such that the resultant numerical fluxes along mesh boundaries are consistent with the various boundary conditions that have to be met. Separating the basic numerical scheme that integrates the flow solution from the machinery that looks after mesh boundaries greatly improves the flexibility of the AMR algorithm. In principle, any *cell-centred* scheme developed for a single quasi-rectangular mesh can form the basis for a mesh integrator. To date, we have used three different integrators for the Euler equations[21, 52, 65] and one for detonation flows[17]. In this section we shall limit ourselves to describing the procedures that are used to process dummy cells, in chapter 5 we shall describe one method that we have used as the basis for a mesh integrator, Roe's flux-difference splitting scheme.

The information that the AMR algorithm gathers along mesh boundaries, see §2.3, enables all dummy cells to be processed using just three simple procedures. One procedure for each of the three possible types of boundary interface. We now give descriptions for each of these procedures in turn. For simplicity we shall only consider dummy cells which border the western edge of a mesh, the procedures for the other three sides follow trivially. In §2.4.3 it was noted that it is only necessary to store one set of boundary information for every rJ_l interfaces along the western boundary of a mesh at grid level l. Consequently

each of the following procedures process some $2rJ_l$ dummy cells at a time.

3.2.1 *External* Boundaries

An interface tagged as an *external* boundary may be further categorized as one of several types: *reflecting, transmissive, no slip, viscous inflow* etc. A separate procedure is required for each type. These *external* procedures require no knowledge of the mesh connectivity, and therefore they are no different from procedures that are often used by schemes which employ just a single quasi-rectangular mesh. For example, to maintain a solid wall boundary condition, suitable for inviscid flows, simply reflect the normal component of momentum across the boundary as shown in figure 3.5. Now to implement a second-order Roe scheme *via* flux limiting it is necessary to use two dummy cells, for limiter functions are evaluated from the ratio of the upwind to local wave strength[58]. And the wave upwind of one entering the flow domain across the interface Bb comes from Aa rather than from Cc.



$$\mathbf{W}_{i,j} = \begin{pmatrix} \rho_{i,j} \\ \rho_{i,j} V n_{i,j} \\ \rho_{i,j} V t_{i,j} \\ E_{i,j} \end{pmatrix}$$

$$\begin{aligned} \mathbf{W}_{0,j} &= \mathbf{R}\,\mathbf{W}_{1,j} \\ \mathbf{W}_{-1,j} &= \mathbf{R}\,\mathbf{W}_{2,j} \end{aligned}$$

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.5: Solid wall boundary procedure for inviscid flows.

At present, the major limitation of the AMR algorithm is the relatively poor geometric packing capabilities offered by its grid. This problem afflicts all schemes which use quasi-rectangular, body-fitted meshes to discretize the flow domain. To overcome this geometric limitation many workers, most notably Löhner *et al*[37], have turned to schemes which use unstructured meshes. But these schemes are inferior to the AMR algorithm in other

respects. Their storage overheads are larger, they require more effort to dynamically adapt the grid, and to date they have not been able to take advantage of a temporal refinement strategy. Now, there is an alternative approach to removing this limitation that could be incorporated into the AMR algorithm. Schemes have been devised by Le Veque[35], and by Priestley[47] which remove any geometric limitation at the expense of having to use more complicated solid wall boundary procedures. The computational grid is allowed to intersect solid surfaces, thus a single cartesian mesh can discretize the flow domain around the most complicated of shapes. A special set of integration procedures are then applied to those cells intersected by solid boundaries. However it should be noted that whilst this cartesian boundary approach shows promise, it has not yet been shown to work for the sorts of complicated geometries that can be tackled routinely using unstructured grids.

3.2.2 *Fine-Fine* Boundaries

Figure 3.6 shows an example where some of the boundary interfaces along the western edge of the mesh $G_{l,a}$ would be tagged as *fine-fine*. This type of boundary is easy to deal with because there is a one to one correspondence between a dummy cell bordering the mesh $G_{l,a}$ and a cell contained by the mesh $G_{l,b}$. For each group of $2rJ_l$ dummy cells that need to be primed, the decoded connectivity information will point to the location within $G_{l,b}$ of one of the required cells. The location of the other cells follows trivially, see §2.4.3, and the solution vectors contained by the $2rJ_l$ mesh cells of $G_{l,b}$ are simply shovelled to the corresponding $2rJ_l$ dummy cells of $G_{l,a}$. For a second-order explicit mesh integrator this procedure results in a seamless join between the two meshes.

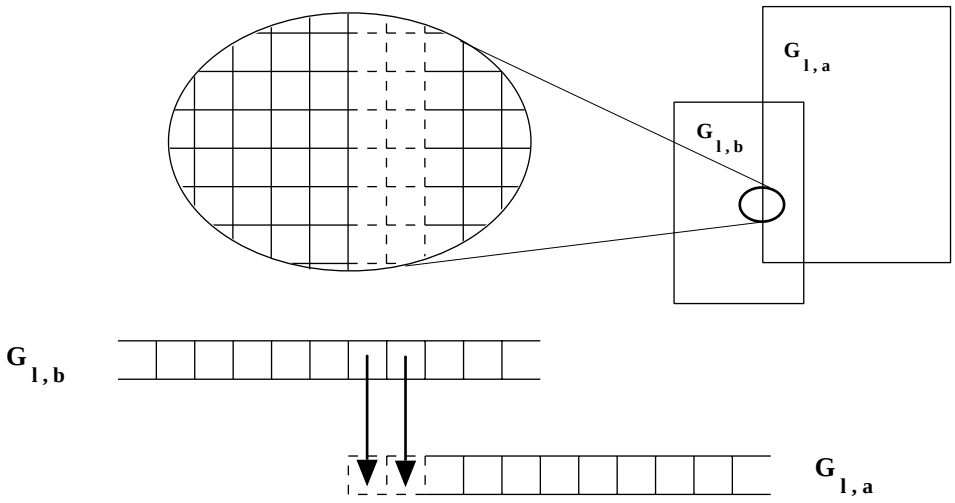


Figure 3.6: A *fine-fine* boundary.

3.2.3 *Fine-Coarse* Boundaries

The ramifications stemming from the procedure used to prime dummy cells along *fine-coarse* boundaries are widespread. To avoid clouding the description for this procedure, here we shall just give the bald details. In the next section we shall attempt to justify our treatment of *fine-coarse* boundaries, in particular we shall address the issues of conservation, accuracy and stability.

Figure 3.7 shows an example where all the boundary interfaces for a mesh $G_{l,a}$ would be tagged as *fine-coarse*, for simplicity we have assumed both rI_l and rJ_l have the value 4. Now consider the implications of the temporal refinement strategy described in §3.1. Following each integration of the coarse mesh $G_{l-1,b}$ from time t to time $t+\Delta t_{l-1}$, the fine mesh $G_{l,a}$ will be integrated four times with a time step of $\Delta t_{l-1}/4$ as shown in figure 3.8. For each of these four integrations, the *fine-coarse* boundary procedure primes a dummy cell contained by the mesh $G_{l,a}$ with an estimate for the solution vector that would have been available if this cell had in fact covered some mesh at level l. These estimates are found by interpolating the coarse grid solutions that are known at times t and $t+\Delta t_{l-1}$.

First, linear interpolation in time is used to estimate the coarse grid solution at the desired time level: t, $t+\frac{1}{4}\Delta t_{l-1}$, $t+\frac{1}{2}\Delta t_{l-1}$ or $t+\frac{3}{4}\Delta t_{l-1}$. Then, a MUSCL[66] procedure is used to interpolate this solution in space to find values with which to prime the dummy cells. The coarse grid solution is assumed to be piecewise planar, the slopes for each planar element are found by applying a *MinMod* limiter function to the forward and backward slopes between cell centres. So, for the coarse cell (i,j),

$$\begin{aligned} \phi_{i+\frac{1}{2},j} - \phi_{i-\frac{1}{2},j} &= MinMod(\phi_{i+1,j} - \phi_{i,j}, \phi_{i,j} - \phi_{i-1,j}) \\ \phi_{i,j+\frac{1}{2}} - \phi_{i,j-\frac{1}{2}} &= MinMod(\phi_{i,j+1} - \phi_{i,j}, \phi_{i,j} - \phi_{i,j-1}). \end{aligned} \tag{3.5}$$

Where,

$$MinMod(a,b) = \begin{cases} 0 & \text{if } ab < 0 \\ sgn(a).min(|a|,|b|) & \text{otherwise.} \end{cases} \tag{3.6}$$

Note the decoded connectivity information for each group of $2rJ_l$ dummy cells

$$\{G_{l,a:i,j_f+j-1} : -1 \le i \le 0, 1 \le j \le rJ_l\},$$

provides the mesh index, $Gp_l + b$, and the mesh co-ordinates, (i_c, j_c), of the coarse cell that they overlap. Furthermore, the eight cells which surround this cell must also belong to the mesh $G_{l-1,b}$, but note some of these cells may be dummy cells. Therefore it is straightforward to locate the coarse grid solution for a nine point stencil centred on (i_c, j_c).

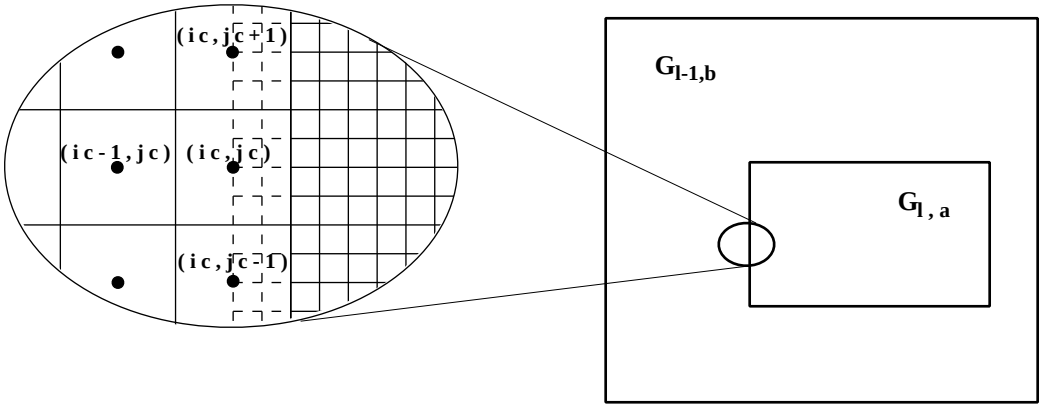


Figure 3.7: A *fine-coarse* boundary.

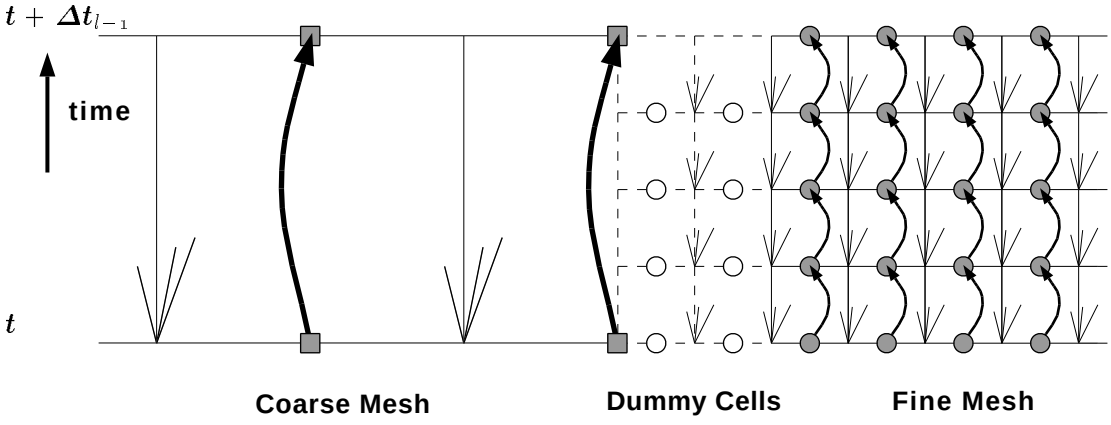


Figure 3.8: Time evolution of the flow along a *fine-coarse* boundary.

3.3 Treatment of *Fine-Coarse* Boundaries

There is much evidence to suggest that discontinuities in mesh spacing exasperate the numerical difficulties associated with modelling shock waves. Indeed, it has become part of CFD folklore that spurious reflections are likely to occur whenever a shock wave crosses such a discontinuity. Several workers, principally Rai[49], claim to have developed procedures which enable shock waves to run uninterrupted across discontinuous grids. In practice, the performance of such remedial procedures is problem dependent; provided that the shock waves are not too strong and that the grid discontinuities are not too severe then satisfactory results may be obtained, otherwise problems remain. The following thought experiment suggests that preventative measures are a better policy than curative measures.

Consider the composite grid formed from abutting two uniform rectangular meshes, one mesh being r times finer than the other, and suppose a planar shock wave is allowed to propagate in a direction which runs parallel to this join. All things being well, one would expect the flow to remain one-dimensional. But, if r is very large then it is difficult to see how such a two-dimensional simulation could maintain, indefinitely, a one-dimensional flow. Note for a given *shock-capturing* scheme the numerical representation of a shock is self-similar with mesh spacing. Therefore, as shown in figure 3.9, the shock wave on the coarse mesh would be much wider than that on the fine mesh. Consequently, at the foot of the shock wave there would be a pressure gradient which acts across the grid discontinuity from the coarse mesh to the fine mesh, and at the head of the shock there would be a pressure gradient which acts in the opposite direction. Thus, the grid discontinuity would cause the supposedly planar shock wave to act as a vorticity generator! Even if the rate of production were small, the accumulation of vorticity would be relentless. So, sooner or later the two-dimensional numerical flow solution would differ markedly from the expected one-dimensional solution.

Although the above experiment is somewhat artificial, it does nevertheless illustrate an insidious problem — how can a general *fine-coarse* boundary procedure possibly distinguish between numerical effects and physical effects. If r is large, the smeared shock seen by the fine mesh would extend over many mesh cells. Consequently, the vorticity generator would inevitably appear as a genuine flow feature rather than as a localized numerical phenomena. In effect, remedial procedures only work because they place a restriction on the size of r, thus it becomes possible to distinguish between physical features and numerical features. Despite their short comings, remedial procedures serve a laudable purpose, and to some extent our treatment of *fine-coarse* boundaries follows such an approach. But the best form of cure remains prevention. As described in the next chapter, under normal circumstances the AMR algorithm ensures that strong discontinuities are wholly encased within a single embedded grid which shadows their every movement. Thus a strong discontinuity is never given the chance to cross a *fine-coarse* mesh boundary.

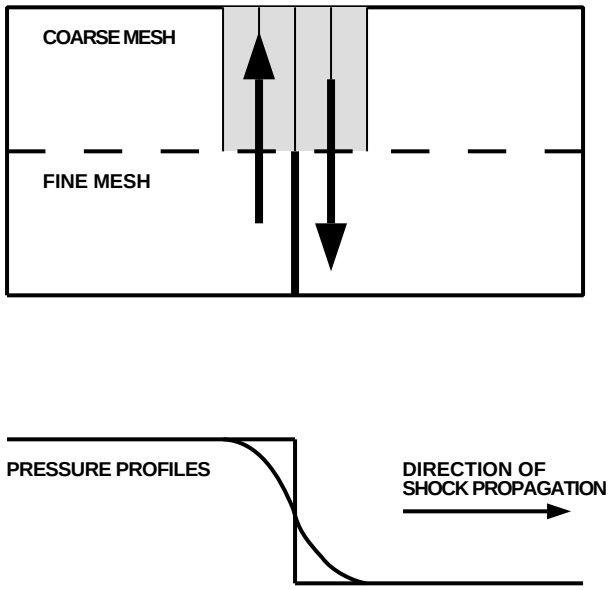


Figure 3.9: A grid discontinuity can cause a plane shock to act as a vorticity generator.

3.3.1 Conservation

It is well known that provided a *shock-capturing* scheme is conservative, a converged numerical solution is guaranteed to represent a weak solution to the equations which govern the fluid motion, and so will yield accurate predictions for the strengths and speeds of shock waves. Although conservation *per se* is not a sufficient condition to ensure reliable results[1], it is almost universally accepted as being a pre-requisite to produce such results. Consequently, the overriding constraint on our treatment of *fine-coarse* boundaries is that it be conservative. Now, the easiest way to ensure that a numerical scheme be conservative is to enforce conservation rigidly by adopting a finite-volume methodology. For a finite-volume scheme, the flow solution is updated by applying a flux balance to each mesh cell. If all the separate flux balances are added together, fluxes across cell interfaces internal to the computational domain cancel. Therefore, any net changes in the so-called conserved variables are entirely due to the fluxes which cross the boundaries of the computational domain. So, for example, if there are no fluxes across this boundary then the total mass of the flow solution must remain constant. Note, the *fine-coarse* boundary procedure described in §3.2.3 is not conservative; with reference to figures 3.7 & 3.8, the net flow across the eastern interface of the cell $G_{l-1,b:i_c,j_c}$ does not, in general, match the net flow across the four adjoining cell interfaces on the western edge of the mesh $G_{l,a}$. However, it is possible to enforce conservation by applying a simple fix up procedure to the updated coarse solution.

[1] A weak solution is not necessarily a unique solution. Consequently, not all weak solutions are physically correct. Indeed, some *shock-capturing* schemes admit spurious solutions in the form of expansion shocks. Note such solutions violate the 2^{nd} law of Thermodynamics. Also, a conservative scheme may not be stable and so will not be able to produce a convergent solution.

Before giving a description for this fix up procedure it is convenient to introduce the following notation,

$Vol_{l,k:i,j}$	Volume of cell $G_{l,k:i,j}$
$Af_{l,k:i,j}$	Area of face between cells $G_{l,k:i,j}$ and $G_{l,k:i+1,j}$
$Ag_{l,k:i,j}$	Area of face between cells $G_{l,k:i,j}$ and $G_{l,k:i,j+1}$
$\boldsymbol{\mathcal{F}}^t_{l,k:i,j}$	Flux vector normal to cell interface $Af_{l,k:i,j}$ at time t
$\boldsymbol{\mathcal{G}}^t_{l,k:i,j}$	Flux normal to cell interface $Ag_{l,k:i,j}$ at time t.

We assume that the integration of the mesh cell $G_{l,k:i,j}$ may be cast in the form,

$$\begin{aligned}\mathbf{W}^{t+\Delta t_l}_{l,k:i,j} = \mathbf{W}^t_{l,k:i,j} \quad & -\tfrac{\Delta t_l}{Vol_{l,k:i,j}} \left[\boldsymbol{\mathcal{F}}^t_{l,k:i,j} Af_{l,k:i,j} - \boldsymbol{\mathcal{F}}^t_{l,k:i-1,j} Af_{l,k:i-1,j}\right] \\ & -\tfrac{\Delta t_l}{Vol_{l,k:i,j}} \left[\boldsymbol{\mathcal{G}}^t_{l,k:i,j} Ag_{l,k:i,j} - \boldsymbol{\mathcal{G}}^t_{l,k:i,j-1} Ag_{l,k:i,j-1}\right].\end{aligned} \tag{3.7}$$

For brevity we can write this as,

$$\mathbf{W}^{t+\Delta t_l}_{l,k:i,j} = \mathbf{W}^t_{l,k:i,j} + \mathbf{DW}^t_{l,k:i,j}. \tag{3.8}$$

However the actual integration process need not be a single stage scheme nor does it need to be a finite-volume scheme. All that we assume is, that the update procedure may be interpreted along the lines of equation (3.7).

Now, we return to the example shown in figures 3.7 & 3.8. The net flow through the interface $Af_{l-1,b:i_c,j_c}$ during the integration of the cell $G_{l-1,b:i_c,j_c}$ from time $t \rightarrow t + \Delta t_{l-1}$ is,

$$\Delta t_{l-1} \boldsymbol{\mathcal{F}}^t_{l-1,b:i_c,j_c} Af_{l-1,b:i_c,j_c}. \tag{3.9}$$

But the net flow through this interface during the rt_l integrations of the mesh $G_{l,a}$ is,

$$\Delta t_l \sum_{m=1}^{rt_l} \sum_{j=1}^{rJ_l} \boldsymbol{\mathcal{F}}^{t+(m-1)\Delta t_l}_{l,a:0,j_f+j-1} Af_{l,a:0,j_f+j-1}. \tag{3.10}$$

In essence, the conservative fix up procedure uses the difference between these two net flows to adjust the updated coarse solution. Since

$$\Delta t_l = \frac{1}{rt_l} \Delta t_{l-1}, \tag{3.11}$$

and

$$Af_{l,a:0,j_f+j-1} = \frac{1}{rJ_l} Af_{l-1,b:i_c,j_c}, \qquad 1 \le j \le rJ_l \tag{3.12}$$

equation (3.10) may be re-written in the form,

$$\Delta t_{l-1} \bar{\boldsymbol{\mathcal{F}}} Af_{l-1,b:i_c,j_c}. \tag{3.13}$$

Where,

$$\bar{\boldsymbol{\mathcal{F}}} = \frac{1}{rt_l rJ_l} \sum_{m=1}^{rt_l} \sum_{j=1}^{rJ_l} \boldsymbol{\mathcal{F}}^{t+(m-1)\Delta t_l}_{l,a:0,j_f+j-1}. \tag{3.14}$$

So, after having found $\bar{\boldsymbol{\mathcal{F}}}$, one way to enforce conservation would be to go back and re-integrate the coarse cell $G_{l-1,b:i_c,j_c}$ replacing the coarse mesh flux $\boldsymbol{\mathcal{F}}^t_{l-1,b:i_c,j_c}$ by $\bar{\boldsymbol{\mathcal{F}}}$. If this were done, after some manipulation we could write,

$$\begin{aligned} \mathbf{W}^{t+\Delta t_l}_{l-1,b:i_c,j_c} &= \mathbf{W}^t_{l-1,b:i_c,j_c} + \mathbf{DW}^t_{l-1,b:i_c,j_c} \\ &+ \frac{\Delta t_{l-1}}{Vol_{l-1,b:i_c,j_c}} \left[\boldsymbol{\mathcal{F}}^t_{l-1,b:i_c,j_c} - \bar{\boldsymbol{\mathcal{F}}} \right] Af_{l-1,b:i_c,j_c} \end{aligned} \tag{3.15}$$

Consequently, a more convenient way to enforce conservation is to simply correct the updated coarse cell solution using the term

$$\frac{\Delta t_{l-1}}{Vol_{l-1,b:i_c,j_c}} \left[\boldsymbol{\mathcal{F}}^t_{l-1,b:i_c,j_c} - \bar{\boldsymbol{\mathcal{F}}} \right] Af_{l-1,b:i_c,j_c}. \tag{3.16}$$

This correction term is applicable to western boundaries only. A similar line of reasoning may be used to find the correction terms for coarse cells along the northern, southern and eastern edges. These terms are,

Northern Edge : (3.17)

$$\frac{\Delta t_{l-1}}{Vol_{l-1,b:i_c,j_c}} \left[\frac{1}{rt_l rI_l} \sum_{m=1}^{rt_l} \sum_{i=1}^{rI_l} \boldsymbol{\mathcal{G}}^{t+(m-1)\Delta t_l}_{l,a:i_f+i-1,JM_{l,a}} - \boldsymbol{\mathcal{G}}^t_{l-1,b:i_c,j_c-1} \right] Ag_{l-1,b:i_c,j_c-1},$$

Southern Edge : (3.18)

$$\frac{\Delta t_{l-1}}{Vol_{l-1,b:i_c,j_c}} \left[\boldsymbol{\mathcal{G}}^t_{l-1,b:i_c,j_c} - \frac{1}{rt_l rI_l} \sum_{m=1}^{rt_l} \sum_{i=1}^{rI_l} \boldsymbol{\mathcal{G}}^{t+(m-1)\Delta t_l}_{l,a:i_f+i-1,0} \right] Ag_{l-1,b:i_c,j_c},$$

Eastern Edge : (3.19)

$$\frac{\Delta t_{l-1}}{Vol_{l-1,b:i_c,j_c}} \left[\frac{1}{rt_l rJ_l} \sum_{m=1}^{rt_l} \sum_{j=1}^{rj_l} \boldsymbol{\mathcal{F}}^{t+(m-1)\Delta t_l}_{l,a:IM_{l,a},j_f+i-1} - \boldsymbol{\mathcal{F}}^t_{l-1,b:i_c-1,j_c} \right] Af_{l-1,b:i_c-1,j_c}.$$

In essence, our fix up procedure represents a generalization of the method used by Berger & Collela[8]. Enforcing conservation in this manner places little constraint on the internal workings of the mesh integrator and is therefore preferable to methods which attempt to effect a conservative treatment of *fine-coarse* boundaries through the direct manipulation of the computational stencil. Such methods might well work for specific flux functions, but they lack generality and are therefore unsuited to a general purpose scheme such as the AMR algorithm. Finally, a word of warning, if two or more meshes at level l have boundary interfaces which abut the same coarse cell at level $l-1$ then care should be taken to ensure that this coarse cell receives only one fix up.

3.4 Stability and Accuracy

It is inevitable, that both the accuracy and the stability of our *fine-coarse* boundary procedure will vary with the choice of numerical scheme used to integrate the flow solution. However, many *shock-capturing* schemes can be viewed as comprising two parts, an upwind part which provides stability and a Lax-Wendroff part which provides accuracy. So, despite the fact that these two parts are combined non-linearly[2], we can gauge the performance of the *fine-coarse* boundary procedure by considering these two component schemes, separately. To this end, we now examine solutions to the one-dimensional linear advection equation

$$\frac{\partial w}{\partial t} + a\frac{\partial w}{\partial x} = 0, \qquad a > 0.$$

Specifically, we consider calculations where the computational grid is formed from two uniform meshes, one mesh being r times finer than the other. However, before we can proceed with this examination, we need to introduce the following notation together with formulations for the upwind and Lax-Wendroff mesh integrators.

If we denote the coarse mesh spacing by Δx_c and the coarse mesh solution at time $n\Delta t_c$ by $\{u_i^n\}$, away from the *fine-coarse* boundary, the flow solution could be updated using the finite-volume formulation

$$u_i^{n+1} = u_i^n + \nu(u_{i-\frac{1}{2}}^n - u_{i+\frac{1}{2}}^n).$$

Where ν is the Courant number $\frac{a\Delta t_c}{\Delta x_c}$, and the interface values $u_{i-\frac{1}{2}}^n$ & $u_{i+\frac{1}{2}}^n$ are, depending on the choice of scheme, given by

$$\begin{array}{lll} \text{Upwind scheme :} & u_{i+\frac{1}{2}}^n = & u_i^n \\ \text{Lax} - \text{Wendroff scheme :} & u_{i+\frac{1}{2}}^n = & \frac{1}{2}(1+\nu)u_i^n + \frac{1}{2}(1-\nu)u_{i+1}^n . \end{array} \tag{3.20}$$

Similarly, if we denote the fine mesh solution at time $n\Delta t_c + m\frac{\Delta t_c}{r}$ by $\{v_j^m\}$, away from the *fine-coarse* boundary, we may write

$$v_j^{m+1} = v_j^m + \nu(v_{j-\frac{1}{2}}^m - v_{j+\frac{1}{2}}^m).$$

Where the interface values $v_{j-\frac{1}{2}}^m$ & $v_{j+\frac{1}{2}}^m$ are identical in form to those given by equation (3.20). Note the fine mesh Courant number is the same as the coarse mesh Courant number, for $\Delta t_f = \frac{\Delta t_c}{r}$.

Now, from the point of view of stability, it is desirable that our treatment of *fine-coarse* boundaries be monotonicity preserving. If this is the case, *fine-coarse* boundaries cannot introduce new oscillations into the flow solution and so the flow integration process will inherit the stability characteristics of the numerical scheme used as the basis of the mesh integration process. Therefore, we now use simple upwinding to investigate the case where

[2]Godunov's theorem precludes second-order schemes with constant coefficients.

information is propagating from the coarse mesh to the fine mesh, the coarse boundary being situated at the interface $i+\frac{1}{2}$, and we suppose that the initial data across this boundary is monotonic such that

$$u_{i-1}^n \geq u_i^n \geq v_1^n.$$

After integrating the coarse mesh *via* the upwinding scheme, u_i^{n+1} will be given by

$$u_i^n + \nu(u_{i-1}^n - u_i^n).$$

Therefore, during the subsequent r integrations of the fine mesh, the fine mesh dummy cell will take values[3]

$$v_0^m = u_i^n + \frac{\nu m}{r}(u_{i-1}^n - u_i^n) \qquad 0 \leq m \leq r-1. \tag{3.21}$$

Consequently, the average value of $v_{\frac{1}{2}}$ over these r integrations is

$$u_i^n + \frac{\nu(r-1)}{2r}(u_{i-1}^n - u_i^n),$$

and so after the application of the conservative fix up pass

$$u_i^{n+1} = u_i^n + \nu\left(1 - \frac{\nu(r-1)}{2r}\right)(u_{i-1}^n - u_i^n). \tag{3.22}$$

During the r integrations of the fine mesh we have,

$$\begin{array}{rcll} v_1^1 & = & v_1^0 + \nu(v_0^0 - v_1^0) & \\ v_1^2 & = & v_1^1 + \nu(v_0^1 - v_1^1) & = \quad v_1^0 + \nu(1-\nu)(v_0^0 - v_1^0) + \nu(v_0^1 - v_1^0) \\ v_1^3 & = & v_1^2 + \nu(v_0^2 - v_1^2) & = \quad v_1^0 + \nu(1-\nu)^2(v_0^0 - v_1^0) + \nu(1-\nu)(v_0^1 - v_1^0) + \nu(v_0^2 - v_1^0) \end{array}$$

So,

$$v_1^m = v_1^0 + \sum_{p=0}^{m-1}\left[\nu(1-\nu)^{m-p-1}(v_0^p - v_1^0)\right].$$

After substituting for v_0^p, we may write this equation as

$$v_1^m = v_1^0 + (u_i^n - v_1^0)\sum_{p=0}^{m-1}\left[\nu(1-\nu)^{m-p-1}\right] + (u_{i-1}^n - u_i^n)\sum_{p=0}^{m-1}\left[\frac{\nu^2 p}{r}(1-\nu)^{m-p-1}\right],$$

which may be simplified to give

$$v_1^m = v_1^0 + (1-\phi^m)(u_i^n - v_1^0) + \frac{\nu}{r}\left[(m-1) - \frac{\phi(1-\phi^{m-1})}{1-\phi}\right](u_{i-1}^n - u_i^n).$$

Where $\phi = (1-\nu)$. Now, the *fine-coarse* boundary procedure will be monotonicity preserving provided that $u_1^{n+1} \geq v_1^r$. Therefore, we require

$$\begin{aligned} u_i^n + \nu\left[1 - \frac{\nu(r-1)}{2r}\right](u_{i-1}^n - u_i^n) \geq \\ v_1^0 + (1-\phi^r)(u_i^n - v_1^0) + \frac{\nu}{r}\left[(r-1) - \frac{\phi(1-\phi^{r-1})}{1-\phi}\right](u_{i-1}^n - u_i^n). \end{aligned}$$

[3]Assumes zeroth order spatial interpolation, thus $v_0^m = u_i^m$.

After rearranging, we may write this condition as

$$u_i^n\left[1-(1-\phi^r)\right]+\nu\left[1-\frac{\nu(r-1)}{2r}\right](u_{i-1}^n-u_i^n)\geq$$
$$v_1^0\left[1-(1-\phi^r)\right]+\frac{\nu}{r}\left[(r-1)-\frac{\phi(1-\phi^{r-1})}{1-\phi}\right](u_{i-1}^n-u_i^n).$$

But, from our original proposition $u_i^n \geq v_1^0$, therefore if we assume that the CFL condition holds, resulting in $0 \leq 1-\phi^r \leq 1$, this monotonicity condition reduces to

$$\left[1-\frac{\nu(r-1)}{2r}\right]\geq\frac{1}{r}\left[(r-1)-\frac{\phi(1-\phi^{r-1})}{1-\phi}\right].$$

Finally, after replacing ϕ by $(1-\nu)$, we arrive at the condition

$$\nu^2 \leq \frac{2}{r-1}[1-(1-\nu)^r]. \tag{3.23}$$

which must be satisfied if our boundary treatment is to preserve the monotonicity of the original data. This same condition would apply if we had started with data $u_{i-1}^n \leq u_i^n \leq v_1^n$. But, interestingly, no such condition exists[4] if the direction of propagation is reversed resulting in information travelling from the fine mesh to the coarse mesh; this case is trivial to analyse and so is not included here. Therefore equation (3.23) is a sufficient condition for our *fine-coarse* boundary procedure in conjunction with upwinding to be monotonicity preserving. Also note, since $\phi^r \ll 1$, a reasonable approximation to this monotonicity condition is given by

$$\nu \leq \sqrt{\frac{2}{r-1}}, \qquad r>2. \tag{3.24}$$

For example, if $r=4$, equation (3.24) predicts $\nu_{max}=0.8165$ whereas equation (3.23) predicts $\nu_{max}=0.8160$.

Although the above analysis is fairly simplistic it does convey an important message. Namely, unless some restriction, over and above the CFL condition, is placed on the size of the Courant number used to integrate the flow solution, a monotonicity preserving mesh integrator will not necessarily yield monotone profiles across *fine-coarse* boundaries. Unfortunately, we are unable to repeat the above analysis for specific second-order monotonicity preserving schemes, for such schemes have non-linear coefficients and it simply becomes too difficult to keep track of these coefficients during the r integrations of the fine mesh. However, it is a simple matter to consider the worst possible case from a certain class of non-linear flux functions, thereby finding an upper bound[5] on the Courant number restriction for common flux functions such as that defined by equation (5.21).

We now consider a general monotonicity preserving interface flux such that the cell interface values, $u_{i+\frac{1}{2}}^n$ or $v_{j+\frac{1}{2}}^m$, cannot lie outside the bounds set by their neighbouring

[4] Other than the CFL condition.

[5] Upper bound in the sense of being most restrictive.

cell-centre values, u_i^n & u_{i+1}^n or v_j^m & v_{j+1}^m. Again we only examine the case where information propagates from the coarse mesh to the fine mesh, starting with initial data $u_{i-1}^n \geq u_i^n \geq v_1^m$. In essence, the monotonicity restriction on the Courant number arises because the net flow across the *fine-coarse* boundary during the r integrations of the fine mesh is greater than that during the single integration of the coarse mesh. Therefore, we can find the worst case simply by maximizing the difference between these two net flows.

Now, after the integration of the coarse mesh, u_i^{n+1} will be given by

$$u_i^n + \nu \Delta u_i,$$

where $\Delta u_i = u_{i-\frac{1}{2}}^n - u_{i+\frac{1}{2}}^n$. Therefore, during the subsequent r integrations of the fine mesh, the fine mesh dummy cell will take values

$$v_0^m = u_i^n + \frac{\nu m}{r}\Delta u - |\delta|, \qquad 0 \leq m \leq r-1.$$

Where $|\delta|$ is a positive adjustment which allows for the MUSCL interpolation process. So, since we assume a convex flux function, the interface value $v_{\frac{1}{2}}^m$ cannot exceed

$$u_i^n + \frac{\nu m}{r}\Delta u_i,$$

and so the average value for $v_{\frac{1}{2}}$ cannot exceed

$$u_i^n + \frac{\nu(r-1)}{2r}\Delta u.$$

Therefore, after the application of the conservative fix up pass, at worst

$$u_i^{n+1} = u_i^n + \nu\left(1 - \frac{\nu(r-1)}{2r}\right)\Delta u_i.$$

Moreover, since we assume that the interface fluxes are monotonicity preserving, we must have

$$v_1^{m+1} \leq v_0^m.$$

Therefore, v_1^r cannot exceed

$$u_i^n + \frac{\nu(r-1)}{r}\Delta u_i.$$

So, considering this worst possible case, to maintain monotonicity across the *fine-coarse* boundary we require

$$u_i^n + \nu\left(1 - \frac{\nu(r-1)}{2r}\right)\Delta u_i \geq u_i^n + \frac{\nu(r-1)}{r}\Delta u_i,$$

or simply

$$\nu \leq \frac{2}{r-1}. \tag{3.25}$$

Again, we emphasize that this condition is overly restrictive. Indeed, numerical experiments using the flux function defined by equation (5.21) suggest that $\nu_{max} = \sqrt{\frac{2}{r-1}}$

remains a reliable guide to the cut off point where the treatment of *fine-coarse* boundaries starts to compromise the monotonicity characteristics of the mesh integrator. Now, in certain ways it is desirable to employ large refinement ratios, but large values of r necessitate small values of ν_{max}. So, a compromise needs to be struck. Hence, the majority of our calculations are performed using $r = 4$ and with ν_{max} set to 0.8 so as to satisfy the condition given by equation (3.24).

Like much of the theory upon which *shock-capturing* schemes are based, the results from the above analysis are only formally valid for the linear advection equation. So, while the study of this model equation has proved to be remarkably fruitful in helping to develop robust schemes for integrating the Euler equations there is no guarantee that any specific result will in fact carry through from the linear scalar case to the non-linear system case. Indeed, in §3.6 we present some numerical results which clearly suggest that a linear scalar analysis is inadequate for investigating the problems that arise whenever a shock wave crosses a grid discontinuity. However, since we do not allow shock waves to cross *fine-coarse boundaries* the above analysis remains a fair indication of why our *fine-coarse* boundary procedure proves to be stable.

To end this section we note, that from a formal viewpoint our *fine-coarse* boundary procedure results in a flow integration process that is only first-order accurate; dummy cells along *fine-coarse* boundaries are primed using linear interpolation, so a high-order mesh integrator which uses this first-order accurate data cannot produce results that are better than first-order accurate. However, this observation says little about the absolute accuracy of the flow integration process. In § 4.4.3 we show that the AMR algorithm can produce results that are comparable in accuracy to those obtained using a single uniformly fine mesh. We attribute this performance to two things. First, the recursive sequencing for the grid integrations minimizes the time span over which it is necessary to interpolate. Second, for second-order fluxes of the form given by equation (3.20), the *fine-coarse* boundary procedure will preserve linear data exactly. But note, simple upwinding will only preserve linear data for a Courant number of 1.

3.5 Co-ordinating the Flow Integration Process

Thus far, we have given separate descriptions for the major components of the flow integration process. Here we describe how these components are combined so as to form a means of integrating the flow solution contained by an arbitrary set of *properly-nested* meshes.

The flow integration process is co-ordinated by the procedure shown in figure 3.10. The basic structure for this procedure stems from the need to recursively interleave the integration of the grids at different levels within the grid hierarchy. This recursive sequencing was discussed in §3.1.1 and so need not be considered further. Other than to

note, that the procedure **Integrate_Flow** cannot be translated literally into FORTRAN, for FORTRAN does not support recursion. But this does not present a serious problem — in appendix C we describe a general method by which FORTRAN may be used to code recursive procedures.

The flow solution contained by the grid G_l is integrated *via* the call **Integrate_Grid**; a breakdown of the procedure **Integrate_Grid** is shown in figure 3.11. The call **Set_BC** primes the dummy cells for each mesh contained by G_l using the procedures described in §3.2. Note, the *fine-coarse* boundary procedure interpolates between t and $t+\Delta t_{l-1}$, hence the parameter N_t. This parameter keeps track of which integration is being performed, namely, $t + (N_t - 1)\Delta t_l \rightarrow t + N_t \Delta t_l$. Once all the dummy cells have been primed, each mesh $G_{l,k}$ may be integrated, separately, using some convenient mesh integrator. The inner workings of the procedure **Integrate_Mesh** are unimportant. Suffice it to say, this procedure calculates cell interface fluxes which are then used to update the solution vectors contained by the set of mesh cells,

$$\{G_{l,k:i,j} : 1 \leq i \leq IM_{l,k}, 1 \leq j \leq JM_{l,k}\}.$$

After a mesh $G_{l,k}$ has been integrated, two actions must be performed so as to allow the conservative fix up to take place. First, the procedure **Save_Coarse_Fluxes** interrogates the mesh boundary information for the grid G_{l+1} to see which, if any, interface fluxes from G_{l+1} are used to fix up the solution $\mathbf{W}_l$. Second, the procedure **Integrate_Fine_Fluxes** updates the running totals for the fluxes across the boundary interfaces of the mesh $G_{l,k}$. These totals will be used to apply the conservative fix up to the solution $\mathbf{W}_{l-1}$. Note, at any one time, a complete set of interface fluxes need only be stored for the single mesh which is in the process of being integrated. Both the coarse fluxes and the fine flux totals used by the fix up procedure are only required along mesh boundaries, and so these quantities may be stored using *type2* storage heaps. Thus, the conservative fix up procedure does not use an excessive amount of storage.

Following the integration of the grid G_l, the call **Reset_Fine_Flux_Totals** simply initialises the running totals which will be calculated during the subsequent rt_{l+1} integrations of the grid G_{l+1}. Once these rt_{l+1} integrations have been performed conservation is enforced using the method described in §3.3.1. Finally, to ensure that the flow solutions at different grid levels remain consistent, the solution on the mesh G_{l+1} is projected onto the underlying coarse mesh G_l; groups of rI_{l+1} by rJ_{l+1} solution vectors at level $l+1$ are simply area weighted to find the equivalent solution vector for the underlying coarse cell at level l.

```
Procedure Integrate_Flow(l,Nt)
  Integrate_Grid(l,Nt)
  if l < l_max {
    Reset_Fine_Flux_Totals(l + 1)
    for Nft = 1 to rt_{l+1} {
      Integrate_Flow(l + 1,Nft)
    }
    Apply_Conservative_Fixup(l + 1)
    Project_Solution(l + 1)
  }
End Procedure
```

Figure 3.10: *Pseudo-code* procedure to co-ordinate the flow integration process.

```
Procedure Integrate_Grid(l,Nt)
  Set_BC(l,Nt)
  for k = 1 to nG_l {
    Integrate_Mesh(Gp_l + k)
    Save_Coarse_Fluxes(Gp_l + k)
    Integrate_Fine_Fluxes(Gp_l + k)
  }
End Procedure
```

Figure 3.11: *Pseudo-code* procedure to integrate a single grid.