Visualization of Astrophysical Data with AVS/Express

Jean M. Favre, Swiss Center for Scientific Computing, Manno, Switzerland, jfavre@cscs.ch, *Rolf Walder*, Institute of Astronomy ETH-Z, Zürich, Switzerland, walder@astro.phys.ethz.ch, *Doris Folini*, Institute of Astronomy and Seminar of Applied Mathematics, ETH-Z, Zürich, Switzerland, doris.folini@sam.math.ethz.ch

ABSTRACT: *In the frame of astrophysical applications, Euler equations including source terms are solved numerically in two and three dimensions. The data are computed on a J90-cluster by an adaptive mesh code with a high degree of vectorization and parallelization. These time-dependent simulations impose very high visualization constraints. We use AVS/Express to integrate custom developments required for memory, CPU, graphics resources and remote data access imposed by the application. We present the implementation of the visualization environment on SGI workstations with our 512-Tbyte storage facility.*

1 Numerical simulation

1.1 Astrophysical purpose: Structure formation in space

Observations by earth-bounded as well as by space-based telescopes strongly indicate that structure formation in a variety of astrophysical objects, ranging from stellar atmospheres to galaxies, is strongly influenced by unstable radiative shocks in colliding hypersonic flows. In nebular, wind driven structures

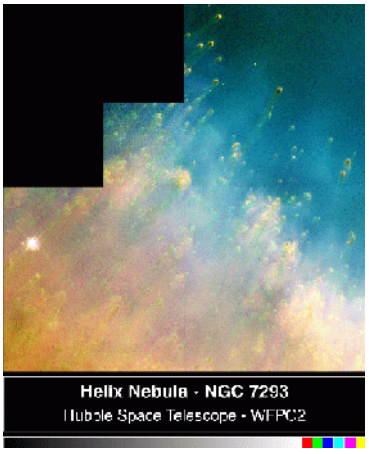

Figure 1: Observed sub-scale structures at the inner rim of the Helix nebula due to radiative shocks. (Hubble Space Telescope observations by O'Dell and Handron [12]).

for example, (see figures 1 and 2) very often sub-scale structures in the form of high-density knots and filaments are

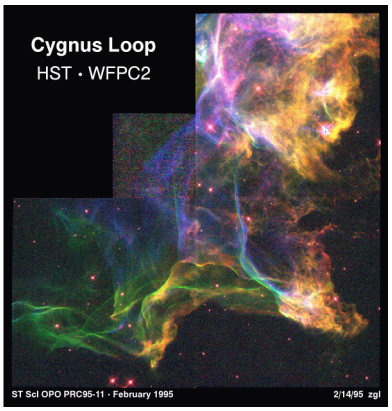

Figure 2: Typical filaments of radiative shock waves in the supernova remnant in Cygnus, the so called Cygnus loop. (Hubble Space Telescope observations by Jeff Hester, Arizona State University).

observed. Spectra taken from such structures often show a velocity dispersion clearly above the thermal one, which indicates that the nebular material is in supersonic turbulent motion. In star formation regions, high speed flows with Mach numbers of several hundred can be observed. Hunter et al.[11] pointed out that such colliding radiative flows provide a very efficient mechanism to trigger the gravitational collapse of molecular clouds, since they can lead to gas compression rates of 10^4 and more. X-ray observations indicate that radiative shocks are present in hot star winds. Numerical models suggest that these

shocks lead to clumping of the atmosphere which, in turn, is a necessary ingredient for driving hot star winds (see e.g. Owocki et al.[12], Schmutz et al.[14]) Only in recent years much attention is being paid to the role of radiative shocks in the formation of primordial galaxies (see e.g. [9], [7]).

The simulation of such astrophysical processes poses severe difficulties. Many physical effects, such as flows, magnetic fields, and radiative transfer are involved. The relevant spatial and temporal scales differ by several orders of magnitudes. Nowadays computer capabilities allow to investigate only part of the entire problem.

1.2 Model problem

We define a simplified model-problem to numerically investigate the stability properties of radiative colliding flows and their influence on structure formation. The model problem can be regarded as a two dimensional, planar Riemann-problem for radiative flows. After having been shocked, the two colliding flows in this model cool due to radiation processes. Our emphasis lies on a high spatial and temporal resolution, on the long term evolution of the model, and on the coupling of different kinds of instabilities. Meanwhile, we apply only a very simplified physical model. It consists of Euler equations with an additional sink term $N^2\,\Lambda(T)$ in the energy equation to account for radiative energy loss in parameterized form, leading to

$$\partial_t E + \nabla\left[\vec{v}\,(E+p)\right] = -N^2\Lambda(T) + Q(T) \quad (1)$$

where E is the total energy of the gas, N the particle density, Λ(T) the radiative loss function given in figure 3, and Q(T) a heating term which ensures that the temperature T remains greater than T_c.

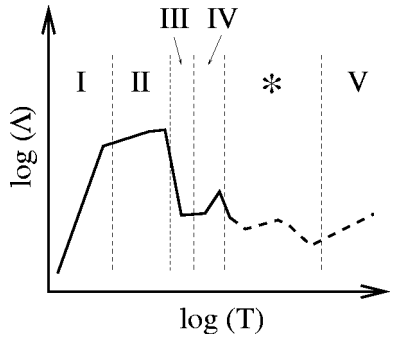

Figure 3: Radiative loss function Λ(T) with different stability regions I--V (details can be found in [17]).

We use a polytropic equation of state with $\gamma = 5/3$. Figures 4 and 5 show the flow pattern immediately after the initialization, when the interfaces are still hardly disturbed. Here, S_i denote the shocks, L_i the layers, where the post-shock gas cools, CDL the cold dense layer where the already cooled gas is collected, and I_i the interfaces between the cooling layers L_i and the CDL.

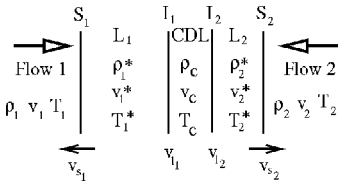

Figure 4: Schematic 2D flow geometry immediately after initialization: The hypersonic flows 1 and 2 collide. Strong shocks are building up (S_1 and S_2). The shocked material cools in the layers L_1 and L_2. The cooled gas collects, highly compressed, in the layer CDL (cold dense layer).

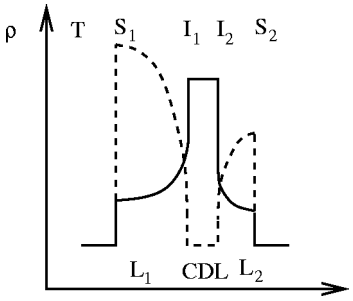

Figure 5: 1D slices shown in density (solid line) and temperature (dashed line) parallel to the undisturbed flows (right) immediately after initialization.

We have investigated two cases. First, a completely symmetric case with respect to the undisturbed flows. Both undisturbed flows have Mach numbers of 16.8. Second, an asymmetric case where the density of the left flow is decreased by a factor of 100 while the velocity is increased by a factor of 10 in order to keep the momentum of the flows balanced. The left, undisturbed flow then has a Mach number of 168, whereas the right, undisturbed flow has a Mach number of 16.8. As a result of its much lower density and much higher temperature the cooling time of the left shocked flow is such that it remains quasi-adiabatic over the time the simulation is performed. Snapshots of the time-evolution are given in figures 6 and 11 for the symmetric and the asymmetric collision cases.

1.3 Results

In astrophysics, there is a long history of investigations of the stability of radiative shocks and the structure formation associated with it (see e.g. [4], [16], [13] for analytical and [3], [10]} [5], [15] for numerical studies). It has become clear that radiative shocks in colliding hypersonic flows are -- apart from gravity -- the main source of structure formation. Cooled matter becomes highly compressed by ratios equal to the Mach number squared. Such high density structures can then cause the gravitational collapse of wider areas. The high compression rate also leads to Atwood numbers close to one across the confining interfaces of

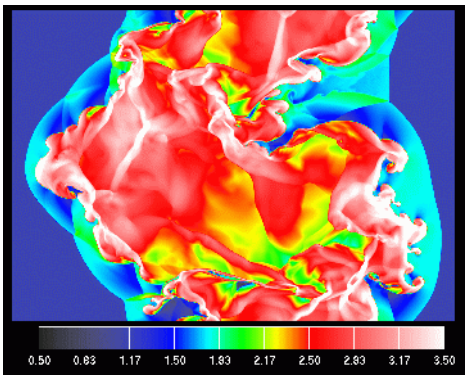

Figure 6: Density structure of the collision zone in the symmetric case. The hot shocked regions (corresponding to L_i in figures 4 and 5) have intermediate densities (green to light blue). The cooled material (CDL in figures 4 and 5), which is compressed up to 4 orders of magnitude, are shown in red to white. This cold material is in a supersonically turbulent state with shocks up to Mach 10. High density walls are formed and energy is dissipated in a series of shocks.

the cooled matter. Such high Atwood numbers, in turn, considerably increase the growth-scales of classical instabilities as Rayleigh-Taylor and Richtmyer-Meshkov. Meanwhile, the high density contrast tends to suppress the growth of Kelvin-Helmholtz instabilities. As a result, high density knots and filaments form and grow efficiently. The high density of such knots and filaments also makes them relatively easily accessible to observations. The observed scales are generally in good agreement with model predictions.

However, the quantitative aspects of such instabilities and the associated structure formation are still not very well confined. This is mainly due to the already mentioned considerable numerical effort which is necessary to investigate this type of flows. In our own studies the emphasis lies on the numerical resolution of all relevant physical scales and on the long-term evolution of the instabilities and their mutual coupling. In the frame of this paper, we can only briefly summarize our results regarding structure formation. For more information we refer the reader to other publications ([17], [18], [19],[20]).

In the asymmetric case, the structure formation is driven by Rayleigh-Taylor (RT) and Richtmyer-Meshkov (RM) instabilities, which themselves are triggered by the thermal cooling instability. Characteristic for the asymmetric case is the formation of filaments out of the CDL, accompanied by their break off and the succeeding formation of high density knots. The basic structure of the interaction zone consists of three layers (see figure 11). a) The thermally unstable cooling layer of the shock (right side). b) A layer of highly compressed, cold gas, in mildly supersonic turbulent motion (middle). RT and RM spikes grow as the confining interfaces are accelerated by the thermally unstable shock. c) A classical, Kelvin-Helmholtz (KH) excited, turbulent mixing layer (left side), where shocked, hot gas streams subsonically around broken off condensates, ablating cold material, and causing efficient X-ray emission.

In the symmetric case, (see figure 6), the structure formation is due to the extremely supersonic turbulence (up to Mach 10) in the layer of compressed gas (CDL in figures 4 and 5). It is mostly driven by unbalanced momentum transport. Characteristic for the symmetric case is the formation of a cellular pattern consisting of high density walls -- filaments -- that surround nearly evacuated cells. These walls can be considerably over-compressed compared to the compression ratio ($M^2 = 16.8^2$) achievable from the undisturbed flows alone. In the highly supersonic flow, extremely subsonic patches are embedded. Although unsteady, these patterns have a considerable lifetime and form coherent structures in the turbulent gas. Energy is dissipated by a series of shocks.

Taking a more mathematical and theoretical physics point of view, our results show that hypersonically colliding Euler flows with a specific energy sink term are unsteady and supersonically turbulent for a wide range of initial conditions.

1.4 Code, performance and data-management

The simulations have been performed by an explicit, high-resolution, finite volume integrator [6] and the adaptive mesh algorithm of Berger [2] has been applied. In this algorithm grid cells are refined whenever the estimated local truncation error is larger than a threshold to be specified. However, the refinement is not done cell by cell as some quad-tree codes are doing. Rather the algorithm fits in an optimal way rectangular (cubed) patches around the cells to be refined. Each such patch then again forms a regular, and -- if the boundary cells once are filled -- a completely independent grid. This results in a rather simple data-structure and in an integration scheme which fits very well onto a parallel shared memory machines with vector processors as CRAY provides it. The price to pay is a certain overhead in CPU and memory requirements for the processing of cells which would not necessarily need to be refined. The explicit scheme is well suited for the investigation of instabilities.

For one simulation which was the basis of the presented video (see [18]) we have integrated 4--5 millions of grid cells for 3600 coarse grid integration steps, corresponding to approximately 1 million fine grid integration steps. 260 MBytes core memory were required. The total integration time was approximately 3000 hours on CRAY J90. The application of the adaptive grid algorithm saved CPU-time and memory of approximately a factor of 10.

The process of the creation, post-processing and saving of the data is nearly completely automatized by a series of shell- and

perl-scripts. The scripts send the jobs to the scheduler of the CRAY-cluster in Zürich. Data are dumped after each time-step on the coarse grid. The data files are automatically labeled with an ID related to the current simulation, the timestep and the actual time and saved on a local disk at the institute of Astronomy. The data are then statistically analyzed on workstations. Finally, they are put on DAT- or DLT-tapes and automatically ftp-ed to the Storage Silos of the Swiss Center for Scientific Computing (SCSC), where most of the visualization takes place.

2 Visualization

AVS/Express[1] from Advanced Visual Systems was chosen for its extensive programmability and its support of multi-block data structures. It provides language primitives for manipulating arrays of typed objects and multi-block solutions can be managed inheriting from all single-grid language primitives and macros. Although our data are hierarchical, we maintain a linear array of data grids, and we use AVS/Express's ability to freely set and use memory references to any subset of data structures (a grid, a colormap, or an integer) to provide the programming flexibility we needed. Further, the dynamic dimensioning of arrays of objects of all types (grids, graphics objects, visualization primitives,...) ease the support of time-dependent simulations where the number of solution grids vary at all time steps because of the adaptive meshing.

Referencing a sub-set of grids is thus easy. The next important requirement is for multiple grids to point to global data such as shared colormaps, shared rendering properties, or shared visualization parameter settings. This is where the programming environment of AVS/Express became very handy, and allowed us to do all of our custom development.

Finally, we were able to build a custom interface to a large data storage (two StorageTek data silos) in order to automatically bring data to remote SGI workstations for rendering.

2.1 Visualizing Hierarchical Data Grids

The natural hierarchy of data grids used by the numerical simulation is an invitation to multi-level data browsing, to offer fast graphics rendering and adaptive levels of detail. With 2D problems, we have dealt with an average of 100 grids per time step. All data for a single time-step are generally read in RAM. The five levels of hierarchy are usually subdivided in less than ten grids for level 0 (the lowest level), then four to sixteen grids for levels 1, 2 and 3, and seventy to one hundred and fifty grids at level 4.

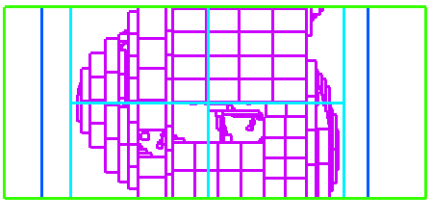

Figure 7: Bounding boxes of all 2D grids at the four highest levels.

The example of figure 7 has a high grid number (a total of 187, with 173 at the highest level). When the number of grids is high, many 2D polygons are necessary to pseudo-color the grids (1.7 million in this particular time-step). For interactive browsing, it is thus imperative to limit ourselves to the lower level grids. We accomplish this by addressing sub-ranges of the array containing all grids.

For a given level of mesh refinement, the rendering of the grids, and of the associated visualizations (isolines, isosurfaces, cross-sections, etc...) must be done by sharing all display modes, display attributes and colormap). Thanks to the ability of AVS/Express's objects to reference objects anywhere in the object hierarchy, we can easily support such requirements. We replaced all native implementations by AVS for their ARRay library of visualization tools, by our own macros, which define global - i.e. shared - objects for the Datamap, the rendering Modes and Properties. The example below shows the definition of an array of graphics primitives for the rendering of the 2d faces of an array of grids of arbitrary size. The object All_2dbounds hold the global rendering modes and properties, and the array "faces2d" references them as well as the shared colormap.

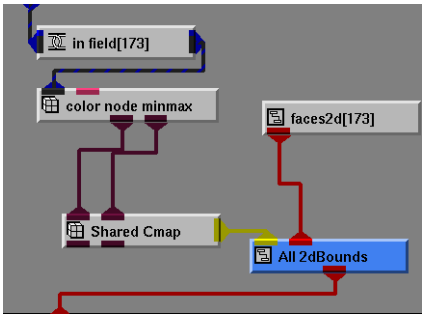

Figure 8: A Visual Editor view of the definition for the rendering of an array of arbitrary size of 2d grids.

AVS/Express two-sided programming interface has been found very effective: Applications can be built inside a Visual Editor with primitives connected in a data-flow like fashion (fig 8), or from a scripting language. Figure 9 shows the corresponding V code associated with the definition of figure 8. We have often built prototypes with the Visual Editor, further refining our implementation using the textual description of objects.

For the definition of parallel visualization objects, the V language provides several built-in commands of interest:

- Links to arrays of arbitrary size
- an *array_size* command to return the total number of elements of an array
- an *index_of* command to return the index of the current object in an array of data structures
- the ability of using the sub-elements of an array of data structures as an array itself.

They allow the automatic dimensioning of derived arrays of data structures, and the setting of each individual element based on its corresponding element in the referenced array.

Our implementation to support the multi-level and multi-grid solution files of the Institute of Astronomy differ from Advanced Visual Systems' implementation in that AVS provides "*parallel*" versions of all their visualization modules by simply duplicating most of the single-grid macros using the built-in commands mentioned above. The end-result is that private objects of all types are allocated for each grid in an array of grids, resulting in a large memory overhead, and in implementations without shared graphics resources, resulting in erroneous images.

We have thus derived new macros for the most common visualization tasks: *Iso-contour lines, Iso-surfaces, Iso-volumes, 2D Solid Textures, Glyphs, Ortho-slicing, Extract_component, Arbitrary slicing, Surf_Plot, Bounds, External faces, External edges, and the Interpolation of any field on any field.*

```
macro faces_2d_ARR {
 imlink in_field;
 node_minmax_arr color_node_minmax {
    in_field => <-.in_field;
    component = 0;
 };

 DefaultLinear Shared_Cmap {
    dataMin+nres => color_node_minmax.min;
    dataMax+nres => color_node_minmax.max;
 };

 macro All_2dBounds {
    AltObject AltObject;
    DefaultProps Shared_Props;
    DefaultModes Shared_Modes;
    GDM.DefaultObject Obj {
      dmap => Shared_Cmap;
      props => Shared_Props;
      modes => Shared_Modes;
      objects => faces2d.obj;
      use_altobj = 1;
    };
    olink obj => .Obj;
 };

macro faces2d[array_size(in_field)] {
 group &in {
    GDxform_templ &xform;
    method render;
 } => in_field[index_of(faces2d)];

 DefaultObject AltObject {
    input => in;
    props => All_2dBounds.AltObject.Props;
    modes => All_2dBounds.AltObject.AltModes;
};

 GDM.DefaultObject Obj {
    input => in;
    xform => in.xform;
    props => All_2dBounds.Shared_Props;
    modes => All_2dBounds.Shared_Modes;
    use_altobj =>All_2dBounds.Obj.use_altobj;
    cached => All_2dBounds.Obj.cached;
    altobj => AltObject;
 };
 olink obj => Obj;
};
};
```

Figure 9: V language definition for the rendering of an array of arbitrary size of 2d grids.

2.2 Other graphics issues

The ability of AVS/Express to support multi-range colormaps was found to be extremely important. Our data often cover large ranges of numerical values, and colormaps defined with a single range of hues varying linearly are often insufficient. To highlight different regions of interaction of the flows (the hot-mixing and turbulent zone, the cooling zone where matters accumulates, and the shocked regions), we defined colormaps built from piece-wise linear interpolation of sub-sets of data of varying numerical ranges. For example, the following colormap defined for data between 0 and 1, provides two-ranges, with C^0 continuity between which hues are increased linearly by AVS/Express' built-in interpolators.

```
double a = 0.38; // this is the middle-point
Datamap shared_map {
  dataMin = 0.; dataMax = 1.;
  ranges => <-.DataRange;
};

DatamapValue DValue[3] = {
{v4=.15, v3=0.,  v2=.66},
{v4=1.,  v3=1.,  v2=.66},
{v4=1.,  v3=.15, v2=0.}};

DataRange DataRange[2] = { {
DataMinValue => <-.shared_map.dataMin,
DataMaxValue => a,
controlPoints => {DValue[0], DValue[1]}},
{
DataMinValue => a,
DataMaxValue => <-.shared_map.dataMax,
controlPoints => {DValue[1], DValue[2]}}
};
```

Figure 10: V language definition for two ranges of colormapping.

For video production, we generate PAL size images at a resolution of 720x576 pixels. With a total of 414,720 pixels, it is clear that the vertices of the 2d computational cells drawn with quadrilaterals may sometimes all project to the same pixel. Thus, here we also use two types of graphics primitives (gouraud shaded quadrilaterals or shaded points), depending on the amount of zoom, the hierarchy level and the resolution of the data grids used.

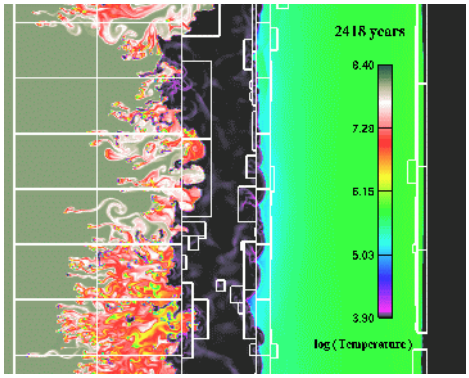

Figure 11: Temperature contours in the interaction region of the asymmetric case with the bounding boxes of the grids at the highest level of the hierarchy.

Because of the adaptive mesh refinement, the grids resolving the highest resolution data do not necessarily completely overlap the lowest level grids (see fig. 11). AVS/Express allows the *jittering* of any graphics primitives: We draw shaded grids or points in a back-to-front method, each with a jitter larger than the preceding one. With 3D data grids, another technique described by Favre[8], allows to carve away sub-domain covered by higher-resolution data, thus rendering only the regions of each level of refinement which aren't further sub-divided. Lastly, for frame-by-frame production mode, we are able to render to multiple hidden windows, and no display lists are created since the graphics data is rendered once and then thrown away.

3 Data processing

3.1 Moving from V code to C code

With large array of grids, AVS/Express' V scripting language does not always provide enough performance. Creating networks of V objects, many of them with the automatic dimensioning and indexing of arrays is very convenient for rapid prototyping. However, the overhead of object instantiation can be too much. When this is the case, one can collapse all implementations of parallel data structures into single V objects. Their executing method loops over all grids in a compiled sub-routine linked into AVS/express' executable. The software development is a lot more engaging, but we have already seen improvements of orders of magnitude in execution time[8].

3.2 Execution of a large time-dependent visualization

AVS/Express data-flow like executing environment simplifies frame-by-frame visualization for large time-dependent problems. Our simulation data are stored in multiple disk files, one file per grid, and all filenames for a given timestep are stored in a meta-file. Our reader module integrated into AVS/Express opens this metafile to read the time step and get links to all related files. Thus, running a large visualization production is done by updating the name of the meta-file, and all reading, processing, visualization and image saving is automatically scheduled for running, based on the dependency tree maintained by AVS/Express' Object Manager. The meta-files names for each time-step can have arbitrary names. We simply load an array of strings and loop over this array.

The only difficulty is then to keep the local disks of our SGI workstations loaded with the current timesteps. This is achieved with a special AVS/Express module allowing calls to the shell, and calling a file-transfer script.

Our data is stored at SCSC (http://www.cscs.ch/Official/Services/TechProfile.html) on two StorageTek silos with six Timberline drives and four helical scan Redwood (D3) drives. A MaxStrat Gen5 front-end disk array of 500 Gbytes serves as cache. To ensure a high-probability of cache hits, we requests groups of time-steps ahead of time. A perl script called by our AVS/Express module writes commands to the .netrc file required by ftp for automatic execution. Both caching commands and transfer commands are sent to the storage unit and ATM connections to the SGI workstations allow for rapid transfer. Once done with a time step, the same AVS/Express module cleans up the local disk, readying it to receive the next group of datasets. With this automatic scripting, we have visualized time-dependent solutions in the tera-scale range. Eventual interruption on the network can be re-started by synchronizing

the cache area and restarting AVS/Express at the appropriate time-step value of the module looping over filenames.

Once a set of image frames is saved on disk, video production is done with another set of tools developed at SCSC by our colleague Angelo Mangili. Allowing frame repetition, smooth blending of individual frames, and many other video tricks, we are able to create Betacam SP videos, as well as the MPEG videos treasured by our Web community[18].

4 Conclusion

AVS/Express' support for arrays of grids, its libraries of graphics objects, colormap objects, its rendering macros, and its open environment allowing customization of User Interface and Numerical Computations have made it possible to build a very productive environment for visualization. Its object-oriented architecture favors code re-use, and we have successfully shared the application-independent part of our multi-grid tools with other scientific applications supported by AVS/Express at our Center. Its scripting language offers a uniform interface to all data types and some powerful primitives which foster the concise and efficient programming of new data structures. Finally, its improved data-flow executing model relieves the programmer from worrying about execution control and it is well suited for the animation of time-dependent data.

References.

[1] AVS/Express, http://www.avs.com/products/expovr.htm

[2] M. J. Berger, Adaptive mesh refinement for hyperbolic equations, Lectures in Applied Mathematics, 22 (1985), 31--40.

[3] J. M. Blondin and B. S. Marks, Evolution of cold shock-bounded slabs, New Astronomy, 1, 235--244, 1996.

[4] R. A. Chevalier and J. N. Imamura, Linear analysis of an oscillatory instability of radiative shock waves, The Astrophysical Journal, 261, 543--549, 1982.

[5] R. A. Chevalier and J. M. Blondin, Hydrodynamical instabilities in supernova remnants: Early radiative cooling, The Astrophysical Journal, 444, 312--317, 1995.

[6] P. Colella, Multidimensional upwind methods for hyperbolic conservation laws, Journal of Computational Physics, 87 (1990), 171.

[7] T. A. Ensslin, P. L. Biermann, U. Klein, and S. Kohle, Cluster radio relics as a tracer of shock waves of the large-scale structure formation, http://xxx.lanl.gov/abs/astro-ph/9712293, submitted to Astronomy and Astrophysics (1997).

[8] J. M. Favre, Towards Efficient Visualization Support for Single-block and Multi-block Datasets, Proceedings of the 1997 IEEE Visualization Conference, 425--428, Eds. R. Yagel and H. Hagen.

[9] M. I. Forcada-Mir'o and S. D. M. White, Radiative shocks in galaxy formation. I: Cooling of a primordial plasma with no sources of heating, http://xxx.lanl.gov/abs/astro-ph/9712204, submitted to Monthly Notices of the Royal Astronomical Society (1997).

[10] B.-I. Jun and T. W. Jones and M. L. Norman, Interaction of Rayleight-Taylor Fingers and circumstellar cloudlets in young supernova remnants, The Astrophysical Journal, 468, L59--L63, 1996.

[11] Jr. J. H. Hunter, M. T. Sandford II, R. W. Whitaker, and R. I. Klein, Star formation in colliding gas flows, The Astrophysical Journal, 305 (1986), 309--332.

[12] S. T. Owocki, J. I. Castor, and G. B. Rybicki, Time-dependent models of radiatively driven stellar winds. I. Nonlinear evolution of instabilities for a pure absorption model, The Astrophysical Journal, 335 (1988), 914--930.

[13] D. Ryu and E. T. Vicinage, The Dynamic instability of adiabatic blast waves The Astrophysical Journal, 368, 411--425, 1991.

[14] W. Schmutz, Photon loss from the helium Ly α line -- the key to the acceleration of Wolf Rayet winds, Astronomy and Astrophysics, 321 (1997), 268--287.

[15] I. R. Stevens and J. M. Blondin and A. M. T. Pollock, Colliding winds from early-type stars in binary systems, The Astrophysical Journal, 386, 265--287, 1992

[16] E. T. Vishniac, Nonlinear instabilities in shock-bounded slabs, The Astrophysical Journal, 428 (1994), 186--208.

[17] R. Walder and D. Folini, Radiative cooling instability in 1D colliding flows, Astronomy and Astrophysics, 315 (1996), 265--284. http://www.astro.phys.ethz.ch/papers/walder/walderp.html

[18] R. Walder, D. Folini, and J. M. Favre, Stability of colliding radiative flows (1997), Video animations, http://www.astro.phys.ethz.ch/staff/walder/walder.html.

[19] R. Walder and D. Folini, Knots, filaments, and turbulence in radiative shocks, Astronomy and Astrophysics, 330 (1998), L21--L24. http://www.astro.phys.ethz.ch/papers/walder/walderp.html

[20] R. Walder and D. Folini, The formation of knots and filaments in shocks, Proceedings of the Workshop on Hypersonic Radiative Outflows out of Thermal Equilibrium, to appear in Astrophysics and Space Science (1998). http://www.astro.phys.ethz.ch/papers/walder/walderp.html