Visualisation of SPH data using SUPERSPHPLOT - v1.0

Daniel Price

March 10, 2005

Contents

1 Introduction

Whilst many wonderful commercial software packages exist for visualising scientific data (such as the widely used Interactive Data Language), I found that such packages could be somewhat cumbersome for the manipulation and visualisation of my SPH data. The main problem was that much of what I wanted to do was fairly specific to SPH (such as interpolation to an array of pixels using the kernel) and whilst generic routines exist for such tasks, I could not explain how they worked, nor were they particularly fast and whilst interactive gizmos are handy, it can prove more difficult to perform the same tasks non-interactively, as required for the production of animations. In fact I have found that the major work in the visualisation of SPH data is not the image production itself but the manipulation of data prior to plotting. Much of this manipulation makes sense within an SPH framework (for example the interpolation provided by the kernel and rotation of the particles for perspective).

SUPERSPHPLOT is designed for this specific task - to use SPH tools to analyse SPH data and to make this a straightforward task such that publishable images and animations can be obtained as efficiently as possible from the raw data with a minimum amount of effort from the user. I have found in the process that the development of powerful visualisation tools has enabled me to pick up on effects present in my simulation results that I would not otherwise have noticed (in particular the difference between a raw particle plot and a rendered image can be substantial). Part of the goal of SUPERSPHPLOT is to eliminate the over use of particle plots as a means of representing SPH data!

1.1 What it does

SUPERSPHPLOT is a utility for visualisation of output from (astrophysical) simulations using the Smoothed Particle Hydrodynamics (SPH) method in one, two and three dimensions. It is written in FORTRAN 90/95 and utilises PGPLOT subroutines to do the actual plotting. In particular the following features are included:

- Rendering of particle data to an array of pixels using the SPH kernel
- Cross-sections through 2D and 3D data (as both particle plots and rendered images).
- Fast projections through 3D data (ie. column density plots, or integration of other quantities along the line of sight)

- Vector plots of the velocity (and other vector quantities), including vector plots in a cross section slice in 3D.
- Rotation and fly-throughs (multiple cross-section slices) of 3D data.
- Automatic stepping through timesteps, making animations simple to produce.
- Interactive mode for detailed examination of timestep data (e.g. zooming, rotating, plotting particle labels, working out the gradient of a line, stepping forwards/backwards through timesteps)
- Multiple plots on page, including option to automatically tile plots if $y-$ and $x-$ limits are the same.
- Plot limits can be fixed, adaptive or particle tracking. Also simple to change axes to log, invert, square root or absolute of a quantity.
- Exact solutions for common SPH test problems (e.g. hydrodynamic shock tubes, polytropes).
- Calculation of quantities not dumped (e.g. pressure, entropy)
- Transformation to different co-ordinate systems (for both co-ordinates and vector components).
- Straightforward production of GIF and Postscript images which can then be converted into animations or inserted into LaTeXdocuments.

1.2 What it doesn't do

At the moment SUPERSPHPLOT is designed specifically for gas dynamics simulations with SPH and has basically grown out of my visualisation needs. However as other applications arise other features may be needed which have not yet been included. In particular the following have not been addressed specifically:

- Full 3D plotting. Visualisation in 3D is achieved by taking cross-sections and projections through the SPH data. Similarly 2D data is plotted by means of rendered images. At present there are no capabilities for surface plots or cubes with faces showing cross sections and the like. However particle plots can be plotted in a three dimensional manner using the rotation feature (in this case 3D rotated axes/boxes are plotted appropriately).
- It also doesn't make cappucinos.

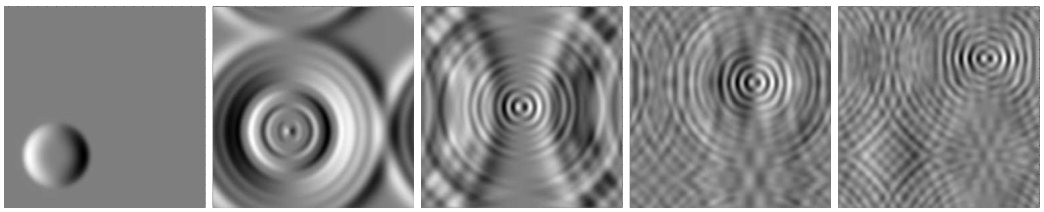

Figure 1: A wave equation solved on SPH particles. This simulation actually resulted from some experiments for cleaning the divergence of the magnetic field in my MHD simulations. The figures show the divergence of the magnetic field in a periodic, two dimensional SPMHD simulation using a hyperbolic divergence cleaning.

1.3 Version History

Version 1.0 This is the pre-public-release (Guinea-pig) version.

1.4 Licence

2 Gallery

3 Getting started

3.1 Compiling the code

To compile SUPERSPHPLOT on your system you will need the following on your system:

- The PGPLOT graphics subroutine library
- A Fortran 95 compiler

3.1.1 PGPLOT

The PGPLOT graphics subroutine library is freely downloadable from

`http://www.astro.caltech.edu/~tjp/pgplot/`

or by ftp from

```
ftp://ftp.astro.caltech.edu/pub/pgplot/pgplot5.2.tar.gz
```

however check to see if it is already installed on your system (if so, the libraries are usually located in /usr/local/pgplot). For details of the actual plotting subroutines used by the SUPERSPHPLOT source code, you may want to refer to the PGPLOT userguide:

```
http://www.astro.caltech.edu/~tjp/pgplot/contents.html
```

3.1.2 Fortran 90/95 compilers

By now, many Fortran 90/95 compilers exist. Amongst the most popular commercially available ones are perhaps the NAG Fortran compiler and the Intel Fortran Compiler (ifc). However, the g95 project has produced a free compiler, which is downloadable from:

```
http://g95.sourceforge.net/
```

and which successfully compiles SUPERSPHPLOT. Another, apparently unrelated project is located at

```
www.gfortran.org
```

however this compiler is not fully implemented and therefore does not as yet compile SUPERSPHPLOT.

3.1.3 Makefile options

In the Makefile, you will need to set the FORTRAN compiler and flags to your local version, e.g..

```
F90C = g95
F90FLAGS = -O
```

Secondly the compiler must be able to link to the PGPLOT and X11 libraries on your system. As a first attempt try using:

```
LDFLAGS = -lpgplot -lX11
```

If that works at a first attempt, take a moment to think several happy thoughts about your system administrator. If these libraries are not found, you will need to enter the library paths by hand. On most systems this is something like:

```
LDFLAGS = -L/usr/local/pgplot -lpgplot -L/usr/X11R6/lib -lX11
```

(assuming the PGPLOT libraries are in the /usr/local/pgplot directory and the X11 libraries are in /usr/X11R6/lib). If this does not work, try using the `locate` command to find the libraries on your system:

```
locate libpgplot
locate libX11
```

If, having found the PGPLOT and X11 libraries, the program still won't compile, it is usually because the PGPLOT on your system has been compiled with a different compiler to the one you are using. A first attempt is to try using the g2c libraries

```
LDFLAGS = -L/usr/local/pgplot -lpgplot -L/usr/X11R6/lib -lX11 -lg2c
```

If the PNG drivers are incorporated into the PGPLOT installation, the `-lpng` libraries must also be added. Failing that, ask your system administrator(!) or simply download your own copy of the PGPLOT libraries and make sure it is compiled with the same compiler as you are using to compile the SUPERSPHPLOT source code.

3.2 Environment variables

Several useful environment variables can be set for PGPLOT and several of them are very useful for SUPERSPHPLOT. In a tcsh shell type:

```
setenv PGPLOT_DEV /xwin
setenv PGPLOT_BACKGROUND white
setenv PGPLOT_FOREGROUND black
```

The first command sets the default device to the X-window, rather than the /null device. The latter two commands set the background and foreground colours of the plotting page. Note that these environment variables should be set *before* invoking supersphplot (it is simplest to set them upon starting the shell by placing them in your .tcshrc or bash/sh equivalent file). For other environment variables which can be set, refer to the PGPLOT user guide.

3.3 System dependent routines

System dependent subroutines are interfaced in a separate `system` module in the file `system_unix.f90`. At present the only calls to these routines are made from `supersphplot.f90` which reads the run name(s) from the

command line (see § **??** on page ??). A standardised format for performing this task is included in the standards for the next release of Fortran (Fortran 2003), however in the meantime the calls in the `system` module may require some adjustment depending on the particular system you are compiling the code on. The program is still fully functional without this call working, but it does make things convenient (in particular it means that SUPERSPHPLOT can be invoked using wildcards ($*$, ?) in filenames).

4 Reading data

The most important part is getting SUPERSPHPLOT to read your data format. I have supplied subroutines for reading output from the publically available GADGET code (`read_data_gadget.f90`) and also for Matthew Bate's SPH code (`read_data_mbate.f90`) which is widely used in the UK. Another example of a data read which I use is given in `read_data_dansph.f90`. Since the specifics of the data read can vary between codes, I recommend examining these subroutines before writing your own version. Further details on writing your own subroutine are given in appendix D

Using one of the subroutines provided, invoke SUPERSPHPLOT with the name of the data file on the command line, e.g.

```
supersphplot myrun
```

With multiple filenames on the command line, ie.

```
supersphplot myrun1 myrun2 myrun3
```

or simply

```
supersphplot myrun*
```

files will be read consecutively in the order that they are given.

5 A brief tour...

After a successful data read, the menu should appear as something like the following:

```
 You may choose from a delectable sample of plots
 ------------------------------------------------------
  1) x                      7) particle mass
  2) y                      8) u
```

Figure 2: example plot

```
 3) z                      9) \gr
 4) v\dx                  10) pressure
 5) v\dy                  11) entropy
 6) v\dz                  12) h
-------------------------------------------------------
13) multiplot [  4 ]      m) set multiplot
-------------------------------------------------------
 d(ata) i(nteractive) p(age) o(pts) l(imits) h(elp)
 r(ender) v(ector) x(sec/rotate) s(ave) q(uit)
-------------------------------------------------------
Please enter your selection now (y axis or option):
```

The simplest plot is of two quantities where only one is a coordinate, for example

```
Please enter your selection now (y axis or option): 9
(x axis) (<cr>=1): 1
 Graphics device/type (? to see list, default /xwin): /xw
```

A full list of available graphics devices is given in the PGPLOT user guide. Some of the most useful devices are given in table 1. In the above we have selected the x-window which means that the output is sent to the screen, producing the graph shown in Figure **??**.

`/xw`, `/xwin`	X-Window (interactive)
`/ps`	Postscript (landscape)
`/vps`	Postscript (portrait)
`/cps`	Colour Postscript
`/gif`	GIF
`/png`	PNG (if installed)
`/null`	null device (no output)

Table 1: Commonly used graphics devices available in PGPLOT

Plot settings are controlled via the menu options and are described below.

6 Menu options

The program options may be changed in a series of submenus. The defaults for all of these options are initially set in the subroutine defaults_set. The

options set using the submenus can be saved using the (s)ave option from the menu. This saves all of the current options to a file called 'defaults' in the current directory, which is automatically read upon starting supersphplot the next time. This file is written using the namelist formatting option provided by Fortran 90. An alternative way of setting options is to edit this file directly prior to invoking the program, referring to the variable list given in §A.3 of this user guide.

6.1 set (m)ultiplot

The multiplot option enables plotting of several different plots on the same physical page. Plots with the same $x-$ and $y-$ axes are tiled if the tiling option from the (p)age options menu (§ 6.4 on the next page) is set. Each plot can have independent $x-$ and $y-$ axes as well as a different rendering (cross section or projection) and/or vector plot. For rendered plots plotting of contours can be turned on/off between plots and for cross sections the position of the cross section can also be changed between plots. An alternative method for specifying a multiplot is to edit the `defaults` file directly prior to starting SUPERSPHPLOT.

6.2 (d)ata options

These options relate to the data read and are as follows:

1. **read new data**. Read new data, using a different runname.
2. **change number of timesteps read**. Set timestep number to start reading from, timestep number to stop reading from and the frequency of timesteps to read (e.g. every two steps). This can also be changed interactively in interactive mode.

6.3 (i)nteractive mode

The menu option turns on/off interactive mode. With this option turned on and an appropriate device selected (ie. the X-window, not /gif or /ps), after each plot the program waits for specific commands from the user. With the cursor positioned anywhere in the plot window (but not outside it!), commands can be invoked using the following keystrokes:

h	:	display/hide help text
left click	:	select area to zoom in on
+	:	zoom in by 10% with plot centred on current cursor position
-	:	zoom out by 10% with plot centred on current cursor position
s	:	save the current plot limits for all timesteps
p	:	label the particle closest to the cursor position
l	:	draw a connecting line between the closest particle and a second particle (to be selected) and calculate the slope of this line.
r	:	replot the current plot.
space	:	advance to next timestep
right click	:	go back to previous timestep
1,2,3..9	:	on space/right click, jump forward/backwards by n timesteps
q	:	quit plotting

NB: If the multiplot option has been used, the timestep changing commands only take effect on the last plot per timestep. Many more commands could be added to the interactive mode, limited only by your imagination. Please send me your suggestions!

6.4 (p)age options

This submenu contains options relating to the PGPLOT page setup. The options are as follows:

1. **toggle page change**. (default=on) When turned off, no new page is created between plots. This means that consecutive timesteps are plotted directly on top of one another. For particle plots this can be used to trace the position of the particles as they evolve in time.

2. **toggle axes**. Changes the appearance of the axes. The variable is the same as the AXIS variable used in the call to PGENV in the standard PGPLOT routines, except that I have also added the -3 option. The options are as follows:

-3 : same as AXIS=-1, but also draw tick marks;
-2 : draw no box, axes or labels;
-1 : draw box only;
0 : draw box and label it with coordinates;
1 : same as AXIS=0, but also draw the coordinate axes (X=0, Y=0);
2 : same as AXIS=1, but also draw grid lines at major increments of the coordinates;

3. **change paper size**. Sets the size of the plotting page.

4. **change plots per page**. Changes the number of plots on each physical page.

5. **toggle plot tiling**. When set, plots with more than one plot on the page and the same $y-$ and $x-$ axes are automatically tiled together.

6. **adjust title position**. Adjusts the position of the title on each plot. To turn off plot titling, set the position outside the viewport (ie. enter some large numbers). For details of plot titling see § 7.1 on page 17

7. **adjust legend position**. Adjusts the position of the legend on each plot. Again, to turn off the legend, set the position outside of the viewport (ie. enter some large numbers). To customise the legend see § 7.2 on page 18

8. **toggle animate**. When set, this turns off the prompting given before the page is changed. Does not apply in interactive mode, where the page changing can be controlled using the mouse.

6.5 particle plot (o)ptions

These options relate to pure 'particle plots'. The options are as follows:

1. **toggle plot line**. When set, this option plots a line connecting the particles in the order that they appear in the data array. Useful mainly in one dimension, although can give an indication of the relative closeness of the particles in memory and in physical space in higher dimensions.

2. **toggle plot average line**. Bins the particles into *nbin* bins along the $x-$axis, calculates the average value of the $y-$axis quantity in each bin and plots a line through these points.

3. **toggle label particles**. This option prints the number of each particle next to its position on the plot (for all particles). Primarily useful for debugging neighbour finding routines. An alternative is to use the 'p' option in interactive mode.

4. **circles of interaction**. On coordinate plots this option plots a circle of radius $2h$ around either all particles or on up to 10 selected individual particles. This is primarily useful in debugging neighbour finding routines. On non-coordinate plots an error bar of length $2h$ in either direction is plotted in the direction of the coordinate axis.

5. **plot ghosts/sinks**. Enables particles of certain types only to be plotted (e.g. dark matter particles only, gas particles only or both).

6. **change graph markers**. This option sets the PGPLOT marker used for each type of particle in the particle plots. The list of markers is given in the PGPLOT user guide and is also listed in Appendix **??**.

7. **toggle cross section/projection**. In three dimensions this determines whether to all the particles projected onto the plane or to plot particles only in a given co-ordinate range. Also applies to rendered and vector plots.

8. **change co-ordinate systems**. This feature transforms the particle co-ordinates and vector components into non-cartesian co-ordinate systems. This is useful for simulation data which has a particular symmetry associated with it (e.g. for accretion disc simulations, the components of velocity in the radial and azimuthal directions can be plotted). Note that this option does *not* change the co-ordinate system used on rendered plots, as this would mean that the interpolation using the SPH kernel would not be correct.

9. **toggle exact solution**. The following exact solutions are provided

   - Hydrodynamic shock tubes (Riemann problem)
   - Spherically-symmetric sedov blast wave problem. This
   - Polytropes (with arbitrary γ)
   - One dimensional toy stars. This is a particularly simple test problem for SPH codes described in Monaghan and Price (2004).
   - Linear wave. This simply plots a sine wave of a specified amplitude, period and wavelength on the plot specified.

- MHD shock tubes (tabulated). These are tabulated solutions for 7 specific MHD shock tube problems.
- Exact solution from a file. This option reads in an exact solution from the filename input by the user, assuming the file contains two columns containing the $x-$ and $y-$ co-ordinates of an exact solution to be plotted as a line on the plot specified.

Details of the calculation of the exact solutions and examples of their output are given in Appendix B. Note that the PGPLOT call to plot the line is included in the exact solution subroutine so as to provide flexibility should an exact solution require multiple lines on the page / different line styles etc. (although none of those provided do).

6.6 plot (l)imits

The options for plot limits are as follows:

1. **toggle fixed/adaptive limits**. With limits set to adaptive, plot limits are minimum and maximum of quantities at current timestep. However, the co-ordinate limits are not adapted in the case of rendered plots. With fixed limits, the plot limits retain their default values for all timesteps.
2. **set manual limits**. Manually set the limits for each column of data.
3. **x-y limits track particle**. Co-ordinate limits are centred on the selected particle for all timesteps, with offsets as input by the user. This effectively gives the 'Lagrangian' perspective.
4. **zoom in/out**.
5. **apply transformations (log10,1/x)**.
6. **save current limits to file**. Saves the current values of the limits calculated from the data read or set manually using option (2) to the file 'rootname.limits' where rootname is the name of the current run. This file is automatically read upon the next invocation of supersphplot. The limits apply only when fixed limits are set.
7. **re-read limits file**. Re-reads the plot limits from the 'rootname.limits' file. This means that the limits contained in this file can be manually changed by the user whilst the program is still running.

6.7 (r)endering options

1. **change number of pixels**. Set the number of pixels along the $x-$axis. Pixels are assumed to be square such that the number of pixels along the $y-$axis is determined by the aspect ratio of the current plot.

2. **change colour scheme**. Changes the colour scheme used on rendered images. A demonstration of all the colour schemes can be also be invoked from this menu option. Setting the colour scheme to zero plots only the contours of the rendered quantity (assuming that plot contours is set to true). The colour schemes given are as follows:

   0 : contours only
   1 : greyscale
   2 : red
   3 : ice blue
   4 : rainbow
   5 : frog monster

   User contributed colour schemes are eagerly invited (see the subroutine 'colour_set' for details on how to do this).

3. **toggle cross section/projection**. For 3D data, toggles whether to plot the rendered quantity integrated along the line of sight (projection) or in a particular cross section along the line of sight. For 2D data setting the cross section option gives arbitrary 1D cross sections through 2D data. Also applies to vector and plots.

4. **toggle plot contours**. Determines whether or not to plot contours in addition to the rendered quantity.

5. **change number of contours**. Sets the number of contours to be plotted.

6. **toggle colour bar**. Determines whether or not to plot the colour bar on rendered images.

6.8 (v)ector plot options

1. **change number of pixels**. Set the number of pixels along the $x-$axis to be used in the vector plots. As in the rendered images, pixels are assumed to be square such that the number of pixels along the $y-$axis is determined by the aspect ratio of the current plot.

2. **toggle particle plotting if no rendering**. If no rendered quantity is set, toggles whether or not to plot the particles in addition to the vector plot.

3. **toggle background/foreground colour**. Determines whether or not to plot the vector arrows in the current background or foreground colour (by default these are white and black respectively). This must be user determined to give the best contrast between the vector arrows and the rendered image.

6.9 (s)ave, (h)elp, (q)uit

The (s)ave options saves the default options to a file called 'defaults' in the current directory which is read automatically upon the next invocation of supersphplot. The (h)elp option simply expands the text for the options menus (I may add to this soon). (q)uit, unsurprisingly, quits. Typing a number greater than the number of data columns also exits the program (e.g. I often simply type 99 to exit).

7 Other features

7.1 Plot titles

Plots may be titled individually by creating a file called `titlelist` in the current directory, with the title on each line corresponding to the position of the plot on the page. Thus the title is the same between timesteps unless the steps are plotted together on the same physical page. Leave blank lines for plots without titles. For example, creating a file called `titlelist` in the current directory, containing the text:

```
plot one
plot two
plot three
lucky me

oops
```

and positioning the title using the default options, produces the titles shown on the graph in Figure 3 (where there are 6 plots on the physical page).

Figure 3: Example use of plot titles

Figure 4: Example of plot rotation in 2D

7.2 Plot legends

Legends are a tricky business.

7.3 Rotation

Rotation is achieved by tranforming to spherical co-ordinates, incrementing the azimuthal and tilt angles appropriately and transforming back to cartesians. An example of rotation of two dimensional data is shown in Figure 4

An example of the rotation of 3D data to give a 3D particle plot is shown in Figure 5. Note the axes which have been rotated and plotted accordingly.

7.4 Flythru

7.5 Co-ordinate transformations

7.6 Power spectrums (1D only)

In one dimension an extra plot item appears in the data menu which takes a power spectrum (in space) of a particular variable defined on the particles. Upon selection the user is prompted for various settings before plotting the power spectrum. For data defined on irregularly distributed particles, there are two methods for taking the power spectrum: Either to interpolate to an even grid and use a Fourier transform or to use a method for calculating a periodogram of irregularly sampled data which can have significant advantages over interpolation. Algorithms for both of these methods have been implemented. For the first, the SPH data is interpolated to a one dimensional grid using the kernel via the `interpolate1D` subroutine before calculating the (slow!) fourier transform in `powerspectrum_fourier`. For the second,

Figure 5: Example of particle plot rotation in 3D

Figure 6: Example of one dimensional power spectrum using the Lomb periodogram

an algorithm due to Lomb and **?** described in Press et al. (1992) is used[1], located in the subroutine `powerspectrum_lomb`. The actual plotting is done in the subroutine `plot_powerspectrum`. An example of this feature is shown in Figure 6, where a power spectrum of a given spectral index has been defined on the particles as an initial condition. The plot shows the velocity variable given the initial power spectrum and the power spectrum calculated via the Lomb periodogram.

It should be stressed, however, that *neither* of the subroutines for calculating the power spectrum is particularly fast and have *only* been included as a preliminary feature since I have used them once or twice in one dimensional simulations where speed is not an issue. The algorithms are fairly simple to extend to multidimensional data, although faster implementations would be needed (such as a Fast Fourier Transform routine).

8 Interpolation routines

8.1 Two dimensional interpolation

8.1.1 Rendering of 2D data

For a contour or rendered plot of a scalar quantity ϕ we interpolate from the particles to an array of pixels using the SPH summation interpolant. In two dimensions the interpolant is simply

$$\phi(x,y) = \sum_b m_b \frac{\phi_b}{\rho_b} W(x-x_b, y-y_b, h_b) \tag{1}$$

where the summation is over contributing particles and W is the standard cubic spline kernel, given by

$$W(q) = \frac{\sigma}{h^\nu} \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3, & 0 \le q < 1; \\ \frac{1}{4}(2-q)^3, & 1 \le q < 2; \\ 0 & q \ge 2 \end{cases} \tag{2}$$

where $q = |\mathbf{r}_a - \mathbf{r}_b|/h$, ν is the number of spatial dimensions and the normalisation constant σ is given by $2/3$, $10/(7\pi)$ and $1/\pi$ in 1, 2 and 3 dimensions respectively.

[1]Note that the subroutines given in Press et al. (1992) have *not* been used as they are not free software.

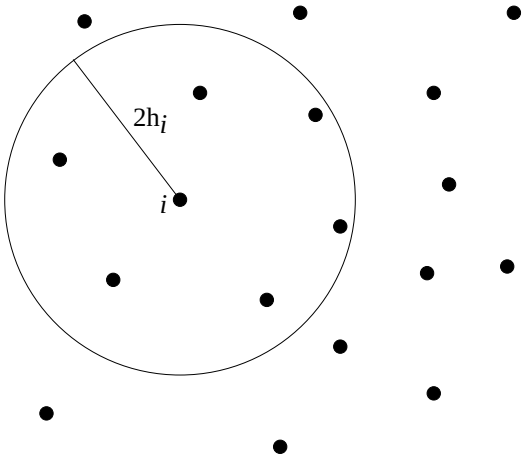

Figure 7: Computation of a two dimensional cross section through 3D data

In three dimensions, we must either take a cross section or a projection through the data.

8.2 Projections through 3D data

8.3 Cross sections

8.3.1 Cross sections of 3D data

A cross section can be taken of SPH data by summing the contributions to each pixel in the cross section plane from all particles within $2h$ of the plane. This is done in the subroutine interpolate_3D_fastxsec. In this case the cross section is always at a fixed value of the third co-ordinate (ie. for xy plots the cross section is in the z direction). Oblique cross sections can be taken by rotating the particles first.

A routine is also provided to do cross sections of two dimensional data (ie. take one dimensional cross sections) (interpolate2D_xsec). In this case the cross-section line can be arbitrarily specified. I will hopefully write an interactive version of this one day.

8.3.2 Cross sections of 2D data

The cross-sectioning algorithm for 2D data (giving a 1D line) is slightly different, in that it also works for oblique cross sections. The cross-sectioning is done in the subroutine interpolate2D_xsec. The cross section is defined by two points (x1,y1) and (x2,y2) through which the line should pass. These

points are converted to give the equation of the line in the form

$$y = mx + c \tag{3}$$

This line is then divided evenly into pixels to which the particles may contribute. The contributions along this line from the particles is computed as follows:

For each particle, the points at which the cross section line intersects the smoothing circle are calculated (illustrated in Figure 8). The smoothing circle of particle i is defined by the equation

$$(x - x_i)^2 + (y - y_i)^2 = (2h)^2 \tag{4}$$

The x-coordinates of the points of intersection are the solutions to the quadratic equation

$$(1 + m^2)x^2 + 2(m(c - y_i) - x_i)x + (x_i^2 + y_i^2 - 2cy_i + c^2 - (2h)^2) = 0 \tag{5}$$

For particles which do not contribute to the cross section line, the determinant is negative. For the particles that do, it is then a simple matter of looping over the pixels which lie between the two points of intersection, calculating the contribution using the SPH summation interpolant

$$\phi = \sum_b m_b \frac{\phi_b}{\rho_b} W(x - x_b, h_b) \tag{6}$$

An example of a 1D cross section through 2D data is shown in Figure 8.3.2.

In principle a similar method could be used for oblique cross sections through 3D data. In this case we would need to find the intersection between the smoothing sphere and the cross section plane. However in 3D it is simpler just to rotate the particles first and then take a straight cross section as described above.

8.4 Projections

8.5 Vector plots

9 FAQS: How do I...

9.1 Read/process my data into images without having to answer prompts?

Firstly edit the settings in the `defaults` file in the current directory (created by doing a 'save defaults' from the main menu) before invoking SU-

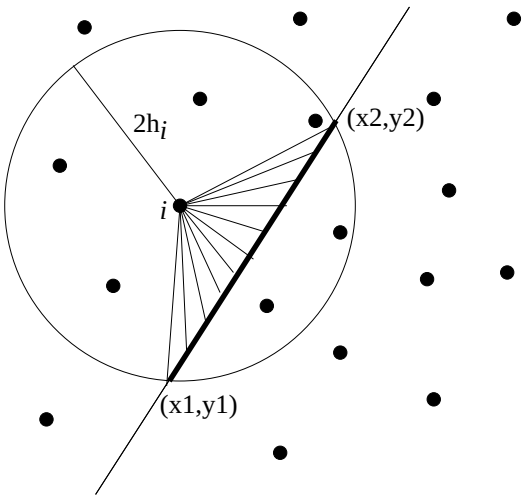

Figure 8: Computation of a one dimensional cross section through 2D data

PERSPHPLOT. An explanation of the variables used in this file is given in Appendix A.3.

Having edited the defaults file, the simplest way of running SUPERSPHPLOT non-interactively is to write a small shell script which runs SUPERSPHPLOT and answers the prompts appropriately. Something like the following should work:

```
#!/usr/bin/tcsh
cd plot
supersphplot myrun* << ENDINPUT
2
1
8
0
mypostcript.ps/ps
q
ENDINPUT
```

which would plot the data in columns 2 and 1 and render the data in column 8 with output to file `mypostscript.ps`.

9.2 Calculate additional quantities?

Additional quantities are calculated in the subroutine `calc_quantities`, in which it should be a simple matter to add your own. If the calculated quantity is to be used elsewhere (for example in an exact solution), an indicator should be created for its position in the data array (e.g. the integer variable `ih` refers to the position of the smoothing length in the data array). It is also preferable to indicate those quantities from which the new quantity is calculated, so that no error will occur if they are not present.

9.3 What about boundaries? How does the rendering work near a boundary?

Usual practise in SPH simulations near boundaries is to introduce ghost particles which mirror the real particles. SUPERSPHPLOT does not explicitly setup any ghost particles but will use any that are present in the data (specified using `labeltype = 'ghost'` in the `read_data` subroutine and then specifying the number of particles of this type). Ghost particles contribute to the rendering calculations but not to the determination of the plot limits. Note, however, that SUPERSPHPLOT does *not* set up ghost particles itself, as this may depend on the type and location of the boundary. Thus if your simulation uses ghost particle boundaries, the ghost particles should be dumped alongside the gas particles in the output file so that their positions, masses, densities and smoothing lengths can be read into SUPERSPHPLOT and used to render the image appropriately.

9.4 Use special characters in the plot labels?

Several of the examples shown in this manual use special characters (such as the $\int$ character) in the plot labels. The PGPLOT user guide explains how to do this, but the basic idea is that PGPLOT uses escape sequences to plot special characters. For example to plot the greek letter ρ we would use

```
label = 'this would print the greek letter \gr'
```

where `\gr` is the PGPLOT escape sequence for ρ. For other characters the escape sequence is given by a number. For example for the integral

$$\int v_x \mathrm{dx} \tag{7}$$

we would use

```
label = '\(2268) v\d x \u dx'
```

where `\(2268)` is the escape sequence for the integral sign. The `\d` indicates that what follows should be printed as subscript and `\u` correspondingly indicates a return to normal script (or from normal script to superscript). All of the escape sequences for special characters are listed in the appendix to the PGPLOT user guide.

> WARNING: Note that the use of escape characters can be compiler dependent and may not therefore work on all compilers.

9.5 Customise the legend?

9.6 Make movies?

At the PGPLOT device prompt

```
Graphics device/type (? to see list, default /xwin):
```

choose `/gif` or `/vgif` (some installations of PGPLOT may also have `/png` installed). The images will then be written as `.gif` files which can then be easily compiled together to form an animation. Under unix the `gifmerge` command (if installed) is a simple way of making a single animated gif file out of a series of gif images. Animated gifs are robust but require large amounts of memory to run at any reasonable speed. However many software packages exist for converting animated gifs into other, more compressed formats (such as the windows .avi format or under unix the .fli format[2]). One such package is the `convert` command included as part of the GIMP (Gnu Image Manipulation Package) toolkit. For presentations the windows .avi format is a good choice, although codecs for conversion are a little harder to come by. Under windows the commercially available `videomach` program is one such tool as well as several Microsoft products.

Another tool under unix is the `fbm2fli` package, which will take a series of .gif or .png files and convert them into a .fli animation (which can be played, for example by the `xanim` tool). This format is great for animations of simulations but as yet does not import into powerpoint and the like.

[2]Note that MPEG is a particularly poor choice for simulation data

10 User contributions / Wishlist for future improvements

Please contribute!! Any user contributions and/or suggestions would be greatly appreciated. The following in particular would be very useful:

- Exact solutions for your favourite test problem(s). It would be great to build a library of user-contributed exact solutions.
- Data analysis tools (e.g. fourier transforms / statistical analysis / algorithms for finding binary stars etc) which could be incorporated.
- New visualisation techniques (e.g. an isosurfacing routine for SPH, better vortex line tracing).
- More colour schemes (see `colour_set.f90` for how to do this)
- Pretty pictures! If you happen to plot some of your data and spend the next several minutes marvelling at how astoundingly beautiful it all looks, please send me a copy (either ps, gif or a movie) to add to the gallery and a few lines describing the simulation.

If you wish to send me subroutines or snippets of (Fortran only!) code for doing any of the above or more, please also send me a LaTeXfile documenting the subroutine similar to the documentation given in this user guide. Also, please, please comment your code clearly so that others can figure out what it does and try to catch as many errors as possible so that the whole program is robust. One thing I have avoided doing is to use any SPH routines which explicitly require finding neighbours, to avoid introducing treecodes and the like and to keep the program independent of any particular implementation of SPH. An example of a routine would be to find the div/curl of a vector quantity using the SPH summation.

Contributions, comments and inevitable bug fixes should be sent to:

`dprice@ast.cam.ac.uk`

although check that this email address is current because I am still a postdoc!

10.1 Known limitations / issues

Here is a partial to do list or things which currently conflict:

- rotation settings override limits transformations on co-ordinate plots

- exact solutions cannot be transformed yet
- Does not yet handle non-cartesian input data

Acknowledgements

Several of the routines were developed from ideas used by Matthew Bate. The polytrope exact solution is from a routine by Joe Monaghan. I am indebted to one Thomas S. Ullrich at the University of Heidelberg who wrote the prompting module which is used throughout the program.

A Source code overview

insert flowchart here

A.1 Modules

A.2 Subroutines

The program consists of the following subroutines (in alphabetical order):

Subroutine		Description
allocate	:	allocates memory for main arrays
calc_quantities	:	calculates additional quantities from particle data
colour_demo	:	demonstration of colour schemes for rendering
colour_set	:	sets up pgplot colour table for rendering
coord_transform	:	transforms between various coord systems
danpgsch	:	sets character height independent of page size
danpgtile	:	my utility for tiling plots on the pgplot page
danpgwedg	:	my very minor modification of pgwedg
defaults_read	:	read default plot options from file
defaults_set	:	sets default plot options if not read from file
defaults_write	:	write default plot options to file
exact_fromfile	:	reads an exact solution tabulated in a file
exact_mhdshock	:	some tabulated solutions for mhd shocks
exact_polytrope	:	exact solution for a polytrope
exact_rhoh	:	plots exact relation between density and smoothing length
exact_sedov	:	exact solution for sedov blast wave
exact_shock	:	exact solution for hydrodynamic shocks
exact_swave	:	exact solution for a linear sound wave
exact_toystar	:	exact solution for the toy star problem
get_data	:	wrapper for main data read
interactive_part	:	interactive utilities for particle plots
interpolate1D	:	interpolation of 1D sph data to 1D grid using sph kernel
interpolate2D	:	interpolation of 2D sph data to 2D grid using sph kernel
interpolate2D_xsec	:	oblique 1D cross section through 2D sph data using kernel
interpolate3D	:	interpolation of 3D sph data to 3D grid using sph kernel

continued on next page

Subroutine		Description
interpolate3D_fastxsec	:	fast cross section through 3D data using sph kernel
interpolate3D_projection	:	fast projection of 3D data to 2D grid using integrated sph kernel
interpolate3D_xsec_vec	:	fast cross section of vector quantity in 3D data using kernel
legend	:	plots legend on plot (time)
limits_read	:	reads plot limits from file
limits_save	:	saves plot limits to file
limits_set	:	calculates plot limits
main	:	main plotting loop
menu	:	main menu
modules	:	contains all shared (global) variables
options_data	:	sets options relating to current data
options_exact	:	sets options and params for exact solution calculation/plotting
options_limits	:	sets options relating to plot limits
options_page	:	sets options relating to page setup
options_particleplots	:	sets options relating to particle plots
options_powerspec	:	sets options for power spectrum plotting
options_render	:	sets options for render plots
options_vector	:	sets options for vector plots
plot_average	:	bins particles along x-axis and plots average line
plot_kernel_gr	:	plots the kernel shape in non-cartesian co-ordinates
plot_powerspectrum	:	calls powerspectrum and plots it
powerspectrum_fourier	:	calculates power spectrum of 1D data on ordered pts
powerspectrum_lomb	:	calculates power spectrum of 1D data on disordered pts
read_data_dansph	:	reads data from my format of data files
read_data_mrbsph	:	reads data from matthew bate's format of data files
render	:	takes array of pixels and plots render map/contours etc
riemannsolver	:	Riemann solver (called by exact_shock)
setpage	:	sets up the PGPLOT page (replaces call to PGENV/PGLAB)
supersphplot	:	main program, drives menu loop
titles_read	:	reads a list of titles to be used to label each timestep
transform	:	applies various transformations to data (log10, 1/x, etc)
vectorplot	:	produces a vector plot from particle data

A.3 Variables

The variables output to the `defaults` file are formatted using Fortran 90's NAMELIST formatting. A typical 'save defaults' command from the main menu will output something like the following (where the corresponding feature is given):

```
&PLOTOPTS
 IADAPT = T,
 XSEC_NOMULTI = F,
 FLYTHRU = F,
 PLOTCIRC = F,
 IPLOTLINE = F,
 IPLOTLINEIN = F,
 LINESTYLEIN = 4,
 IMARKTYPE = 1 4 17 1 1 1,
 IPLOTPARTOFTYPE = T F F F F F,
 IEXACT = 0,
 IPLOTAV = F,
 NBINS = 24,
 ICOLOURS = 1,
 IPOWERSPECY = 4,
 IDISORDERED = F,
 WAVELENGTHMAX = 1.0000000,
 NFREQSPEC = 32,
 ICOORDSNEW = 1,
 NCIRCPART = 1,
 ICIRCPART = 1 1 1 1 1 1 1 1 1 1,
 BUFFER_DATA = T/
```

B Exact solutions

B.1 Shock tubes (Riemann problem)

The subroutine `exact_shock` plots the exact solution for a one-dimensional shock tube (Riemann problem). The difficult bit of the problem is to determine the jump in pressure and velocity across the shock front given the initial left and right states. This is performed in a separate subroutine (riemann-solver) as there are many different methods by which this can be done (see e.g. Toro 1992). The actual subroutine exact_shock reconstructs the shock

profile (consisting of a rarefaction fan, contact discontinuity and shock, summarised in Figure **??**), given the post-shock values of pressure and velocity.

The speed at which the shock travels into the 'right' fluid can be computed from the post shock velocity using the relation

$$v_{shock} = v_{post}\frac{(\rho_{post}/\rho_R)}{(\rho_{post}/\rho_R) - 1}, \tag{8}$$

where the jump conditions imply

$$\frac{\rho_{post}}{\rho_R} = \frac{(P_{post}/P_R) + \beta}{1 + \beta(P_{post}/P_R)} \tag{9}$$

with

$$\beta = \frac{\gamma - 1}{\gamma + 1}. \tag{10}$$

B.1.1 Riemann solver

The algorithm for determining the post-shock velocity and pressure is taken from Toro (1992).

B.2 Polytrope

The subroutine `exact_polytrope` computes the exact solution for a static polytrope with arbitrary γ. From Poisson's equation

$$\nabla^2\phi = 4\pi G\rho, \tag{11}$$

assuming only radial dependence this is given by

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\phi}{dr}\right) = 4\pi G\rho(r). \tag{12}$$

The momentum equation assuming an equilibrium state ($\mathbf{v} = 0$) and a polytropic equation of state $P = K\rho^\gamma$ gives

$$\frac{d\phi}{dr} = -\frac{\gamma K}{\gamma - 1}\frac{d}{dr}\left[\rho^{(\gamma-1)}\right] \tag{13}$$

Combining (12) and (13) we obtain an equation for the density profile

$$\frac{\gamma K}{4\pi G(\gamma - 1)}\frac{1}{r^2}\frac{d}{dr}\left[r^2\frac{d}{dr}\left(\rho^{\gamma-1}\right)\right] + \rho(r) = 0. \tag{14}$$

λ	wavelength
P	period

Table 3: Input parameters for the linear wave exact solution

This equation can be rearranged to give

$$\frac{\gamma K}{4\pi G(\gamma-1)}\frac{d^2}{dr^2}\left[r\rho^{\gamma-1}\right]+r\rho=0. \tag{15}$$

The program solves this equation numerically by defining a variable

$$\mathcal{E}=r\rho^{\gamma-1} \tag{16}$$

and finite differencing the equation according to

$$\frac{\mathcal{E}^{i+1}-\mathcal{E}^{i}+\mathcal{E}^{i-1}}{(\Delta r)^2}=\frac{4\pi G(\gamma-1)}{\gamma K}r\left(\frac{\mathcal{E}}{r}\right)^{1/(\gamma-1)}. \tag{17}$$

B.3 Linear wave

The subroutine `exact_wave` simply plots a sine function on a given graph. The function is of the form

$$y=\sin\left(kx-\omega t\right) \tag{18}$$

where k is the wavenumber and ω is the angular frequency. These parameters are set via the input values of wavelength $\lambda=2\pi/k$ and wave period $P=2\pi/\omega$.

B.4 Sedov blast wave

The subroutine `exact_sedov` computes the self-similar Sedov solution for a blast wave.

B.5 Toy stars

The subroutine `exact_toystar1D` computes the exact solutions for the 'Toy Stars' described in Monaghan and Price (2004). The system is one dimensional with velocity v, density ρ, and pressure P. The acceleration equation is

$$\frac{dv}{dt}=-\frac{1}{\rho}\frac{\partial P}{\partial x}-\Omega^2 x, \tag{19}$$

We assume the equation of state is

$$P = K\rho^{\gamma}, \tag{20}$$

The exact solutions provided assume the equations are scaled such that $\Omega^2 = 1$.

B.5.1 Static structure

The static structure is given by

$$\bar{\rho} = 1 - x^2, \tag{21}$$

B.5.2 Linear solutions

The linear solution for the velocity is given by

$$v = 0.05 C_s G_n(x) \cos \omega t) \tag{22}$$

density is

$$\rho = \bar{\rho} + \eta \tag{23}$$

where

$$\eta = 0.1 C_s \omega P_{n+1}(x) \sin(\omega t)) \tag{24}$$

B.5.3 Non-linear solution

In this case the velocity is given by

$$v = A(t)x, \tag{25}$$

whilst the density solution is

$$\rho^{\gamma-1} = H(t) - C(t)x^2. \tag{26}$$

where the parameters A, H and C are determined by solving the ordinary differential equations

$$\dot{H} = -AH(\gamma - 1), \tag{27}$$

$$\dot{A} = \frac{2K\gamma}{\gamma - 1} C - 1 - A^2 \tag{28}$$

$$\dot{C} = -AC(1 + \gamma), \tag{29}$$

The relation

$$A^2 = -1 - \frac{2\sigma C}{\gamma - 1} + kC^{\frac{2}{\gamma+1}}, \tag{30}$$

is used to check the quality of the solution of the differential equations by evaluating the constant k (which should remain close to its initial value).

B.6 MHD shock tubes

These are some tabulated solutions for specific MHD shock tube problems at a given time taken from the tables given in Dai and Woodward (1994) and Ryu and Jones (1995).

B.7 h vs ρ

The subroutine exact_hrho simply plots the relation between smoothing length and density, ie.

$$h = h_{fact} \left(\frac{m}{\rho} \right)^{1/\nu} \tag{31}$$

where ν is the number of spatial dimensions. The parameter h_{fact} is output by the code into the header of each timestep. For particles of different masses, a different curve is plotted for each different mass value.

C PGPLOT graph markers & line styles

D Writing your own read_data subroutine

The first `ndim` columns in the main data array *must* contain the particle co-ordinates. After these columns the ordering of data is not important, although vector quantities should always be listed with components in the correct order (e.g. $(\nabla \times \mathbf{v})_x$, followed by the $y-$ and $z-$ components) for both vector plotting and for the correct co-ordinate transformation of the vector quantities. Note that the co-ordinates and velocities can have different numbers of dimensions (specified by `ndim` and `ndimV`) since this can occur, for example, in MHD simulations.

Most important is that, for the rendering routines to work, the density, particle masses and smoothing lengths for *all* of the (gas) particles *must* be read in from the data file and their locations in the main data array labelled using the global parameters `irho`, `ipmass` and `ih`. Labelling of the location of other particle quantities (e.g. `iutherm` for the thermal energy) is used in order to plot the exact solutions on the appropriate graphs and also for calculating additional quantities (e.g. calculation of the pressure uses `iutherm` and `irho`).

D.1 Buffering of input

An important point in writing your own subroutine is the manner in which SUPERSPHPLOT buffers the data (ie. how many timesteps are read in one go).

SUPERSPHPLOT stores the data in a global array `dat(maxpart,maxcol,maxsteps)` (contained in the module `particle_data` in the `globaldata.f90` file), where `maxpart` is the maximum number of particles, `maxcol` is the maximum number of columns of data and `maxsteps` is the maximum number of steps which can be stored. Thus for low resolution calculations it is possible to simply read the entire data set into memory at program startup, with the advantages of faster plotting and that the plot limits can be calculated from the whole data set. With the variable `buffer_data` set to `.true.`, SUPERSPHPLOT will attempt to do this, allocating memory appropriately as the number of steps increases.

For calculations in 3D this quickly becomes impractical due to the memory constraints. Therefore, with `buffer_data` set to `.false.`, SUPERSPHPLOT will read the data one dump file at a time, buffering all of the steps in each file. Once the plotting loop reaches the end of this buffer (either by a straightforward loop or from interactively stepping forwards or backwards through the timesteps), the next (or previous) dump file is opened and all of the steps contained in it read.

In the GADGET data read (`read_data_gadget`) this means that only one step is buffered at a time, since each dumpfile only contains one timestep. In `read_data_mbate` each output file contains a number of dumps and this is the number of steps which is buffered. The plot limits are set according to the maximum and minimum of the various quantities in the initial data read (ie. the steps which are buffered at startup). These settings can be overridden using the menu (§ 6.6 on page 15) or by editing the `.limits` file (c.f. §6.6).

References

Dai, W. and P. R. Woodward: 1994, 'Extension of the Piecewise Parabolic Method to Multidimensional Ideal Magnetohydrodynamics'. *J. Comp. Phys.* **115**, 485–514.

Monaghan, J. J. and D. J. Price: 2004, 'Toy stars in one dimension'. *MNRAS* **350**, 1449–1456.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: 1992, *Numerical recipes in FORTRAN. The art of scientific computing*. Cambridge: University Press, —c1992, 2nd ed.

Ryu, D. and T. W. Jones: 1995, 'Numerical magetohydrodynamics in astrophysics: Algorithm and tests for one-dimensional flow''. *ApJ* **442**, 228–258.

Toro, E. F.: 1992, 'The Weighted Average Flux Method Applied to the Euler Equations'. *Philosophical Transactions: Physical Sciences and Engineering* **341**, 499–530.