Ubuntu 14.04 LTS x86-64 semtex Install

Contents

1 Installing semtex

Running semtex using openMPI is a much better choice than using openBLAS threading (may take less than 50% of the time) and using a single openBLAS thread is better than using multiple openBLAS threads. Using openMPI will also be less stress on your processor, running cooler than openBLAS threading. Hence it is suggested that you use openMPI. Most of steps in installing using openBLAS threading are required to use openMPI, so follow 1.1, then 1.2.

1.1 Installing semtex using openBLAS Threading

- Download semtex from `http://users.monash.edu.au/~bburn/semtex.html` (under downloads)
- Extra semtex to location of choice
- Open a terminal and install g++, gfortran, bison

```
sudo apt-get install g++ gfortran bison
```

- Open ../semtex/src/Makefile and find the section for 'Linux-x86_64'
- Comment out the g++44, gcc44 sections, and uncomment the g++, gcc and gfortran section under fastblas

```
#CXX = g++44
#LD = g++44
#CC = gcc44
#FC = gcc44
# --use these with fastblas:
CXX = g++
LD = g++
CC = gcc
FC = gfortran
```

- Navigate to the F77LIBS section under Linux-x86_64, comment out the first F77LIBS and uncomment the fastblas one. Change 'lfastblas' to 'lopenblas'

```
#F77LIBS = -lacml -lacml_mv -lgfortran
# --fastblas = GotoBLAS2. Does not work on AMD 6200 series CPUs
F77LIBS = -lopenblas -lgfortran
# --this works reliably with either compiler suite but is slow on AMD/Opteron
#F77LIBS = -llapack -lgfortran
```

- Download OpenBLAS (based on GotoBLAS2) `http://www.openblas.net/`. Extract the files from the archive
- Open a terminal and navigate to the location of the extracted OpenBLAS files
- run Make to compile OpenBLAS. This may take a while.

```
make
```

If you are running a Skylake CPU, then add the target to make under HASWELL.

```
make target=HASWELL
```

If everything is successful, something similar to the following should be shown.

```
OpenBLAS build complete. (BLAS CBLAS LAPACK LAPACKE)

  OS               ... Linux
  Architecture     ... x86_64
  BINARY           ... 64bit
  C compiler       ... GCC (command line : gcc)
  Fortran compiler ... GFORTRAN (command line : gfortran)
  Library Name     ... libopenblas_haswellp-r0.2.18.a (Multi threaded; Max num-threads is 8)
```

- OpenBlas needs to be installed to a directory, by default this is /opt/OpenBLAS, but this can be changed by specifying a PREFIX=install_directory argument after make install

```
sudo make PREFIX=/usr/ install
```

  This should return 'Install OK!'

- Generally for semtex, single threaded openBLAS is quicker than multi-threaded openBLAS, so to make openBLAS only use a single thread, the OPENBLAS_NUM_THREADS environment variable must be set. This can be set in your .bashrc file.

```
gedit ~/.bashrc
```

  Then add the following to the end of the file

```
# openBLAS Threads
export OPENBLAS_NUM_THREADS=1
```

- Navigate to semtex folder via command line, run make test

```
make test
```

  The process that is undertaken is described in the semtex user manual.

- Navigate to /semtex/utility
- Make the utilities

```
make all
```

- Navigate to /semtex/elliptic
- Make the elliptical solver

```
make
```

- Add the semtex utilities folder to the path variable by editing /.bashrc

```
gedit ~/.bashrc
```

  The path needs to be prepended, as there are other programs with the same name. Add the following line at the end of the file

```
# Adding semtex to path
export PATH="<Path to semtex folder>/utility":$PATH
```

- It is also worth adding the *dns* and *elliptic* folders to the path (prepend)

```
# Adding semtex to path
export PATH="<Path to semtex folder>/dns":$PATH
export PATH="<Path to semtex folder>/elliptic":$PATH
```

- The save utility will require the csh program. Install it using,

```
sudo apt-get install csh
```

1.2 Installing semtex with openMPI for Parallel Processing

This requires that the following steps above to install semtex using openBLAS threading to be completed.

- Install the openmpi packages

```
sudo apt-get install openmpi-bin openmpi-doc libopenmpi-dev
sudo apt-get update
```

- Navigate to the femlib directory of semtex, /semtex/femlib. Clean the directory,

```
make clean
```

- Make with MPI=1

```
make MPI=1
```

  If this cannot find <mpi.h>, then open the file message.c.

```
gedit message.c
```

  Replace the line "#include <mpi.h>" with the path to the mpi.h file. Generally this will be in "/usr/lib/openmpi/include". openMPI doesn't install to the directories that gcc looks in, when installed via apt-get. The other option is to compile openmpi manually, and put it in one of these directories, but this can take a while, so it's easier to just point it to the right path.

```
#include </usr/lib/openmpi/include/mpi.h>
```

- Then do make and make install, with MPI=1

```
make install MPI=1
```

- Move to the top level directory of semtex, /semtex. Make with MPI=1

```
make MPI=1
```

  This should have created dns_mp in the dns directory of semtex.

- Now if mpi were to be used now, the simulation would run slow because openBLAS would be creating multiple threads, and semtex would be created multiple instances of processes. So openBLAS needs to be told to only use one thread. This is achieved using export OPENBLAS_NUM_THREADS=1. This adds the variable to the environment variable. To avoid needing to set this each time, add this to your .bashrc file.

```
gedit ~/.bashrc
```

  Then add the following to the end of the file

```
# openBLAS Threads
export OPENBLAS_NUM_THREADS=1
```

  Now openBLAS will always only use 1 thread.

- Reload the bashrc file

```
source ~/.bashrc
```

2 Installing SuperMongo

- Obtain a copy of supermongo, extract to the desired location
- Open a terminal, navigate to this location (<install location>/sm2_X_XX)
- Run the setup, set_opts

```
./set_opts
```

- Use the defaults for every question, except for

```
"What devices do you want to compile in? Your options are:
TK X11
Choose one at a time, carriage return on an empty line to finish
```

  Answer with X11

```
X11
```

  This will have generated a few options files.

- Navigate to <install location>/sm2_XX_X/src and find "options.h". Open it.
- Remove the following lines from the top of the file

```
#error You must edit options.h before SM will compile.
SM will not compile until you have removed these seven lines
By removing them you acknowledge that you have read our
copyright notices, and you confirm that you have a legal
copy of SM. If you are not sure of your status please send
email to either Robert Lupton (rhl@astro.princeton.edu) or
Patricia Monger (monger@mcmaster.ca).
```

  It is also useful to enable double precision in supermongo.
  To do this, in "options.h", find the line

```
typedef float REAL;
```

  and replace it with,

```
typedef double REAL;
```

  Save the file.

- Move up out of the source directory, and run make

```
cd ..
make
```

- Run make install

```
sudo make install
```

  Enter your password.

- There are two way to setup the macros for supermongo. If you are not provided with an extra macros then do the following,
  Option A
  - ⋆ To enable the use of the supermongo macros provided with semtex, the macros folder of semtex needs to be given to sm.
    Navigate to <install location>/sm2_4_36/user
  - ⋆ Open sm.sm, copy it to your HOME directory and rename it *.sm*. Use *ctrl+H* to show the hidden files (. is for hidden files)

⋆ Change the macro2 entry from ./ to the location of the semtex supermongo directory, <install directory>/semtex/sm

⋆ Open <Install Directory>/semtex/sm/default

⋆ Replace the following lines

```
define sm_home "/home/hmb/lib/sm"
...
define prism_home "/home/hmb/lib/sm"
```

With

```
define sm_home "<install directory>/sm"
...
define prism_home "<install directory>/sm"
```

⋆ Add the line

```
device   x11 -focus
```

Option B

⋆ Create a .sm file in your home directory (use *ctrl+H* to show hidden files), and fill it with the following, where <extra_macros> is the directory of the extra macros, and <user> is the current user.

fonts	/usr/local/lib/sm/fonts.bin
history_file	.smhist
history	80
temp_dir	/tmp/
SHELL	bash
filecap	<extra_macros>/filecap
+graphcap	/home/bcub3d-ubuntu/lib/sm/graphcap.local
graphcap	/usr/local/lib/sm/graphcap
termcap	/usr/local/lib/sm/termcap
help	/usr/local/lib/sm/help/
macro	/usr/local/lib/sm/macro/
macro2	<extra_macros>/
printer	PostScript
file_type	C
#edit	<extra_macros>/maps.dat
#device	x11 -bg black -focus
device	x11 -focus
TeX_strings	1
history_char	∼
uppercase	0
case_fold_search	1
line_up_exponents	1
name	<user>
background	black

⋆ In the <extra_macros> directory, open the *default* file and change the *define sm_home* and *define prism_home* paths to the <extra_macros> directory

define sm_home	<extra_macros>
define prism_home	<extra_macros>

- Run supermongo to check that the macros have been read correctly

```
sm
```

This should give the following,

```
# $Id: default,v 8.1 2015/04/20 11:14:17 hmb Exp $
# $Id: ld.sm,v 8.1 2015/04/20 11:14:17 hmb Exp $
# $Id: meshnum.sm,v 8.1 2015/04/20 11:14:17 hmb Exp $
Hello <user>, please give me a command
```

- Delete the command history with

```
DELETE 0 10000
```

3 Installing SView

- Download from `http://users.monash.edu.au/~bburn/semtex.html`. Extract the files.
- Open a terminal and navigate to the extracted directory
- Install freeglut3 for Ubuntu.

```
sudo apt-get install freeglut3-dev
```

- Install a libtiff library

```
sudo apt-get install libtiff4-dev
```

- Install xmu and xi libraries.

```
sudo apt-get install libxmu-dev libxi-dev
```

- Open "sview.h" with a text editor, and replace the line 24

```
#include "/usr/local/include/GL/glut.h"
```

with

```
#include "/usr/include/GL/glut.h"
```

- Some includes may be missing when compiling on Ubuntu. In "sview.h", add the following include in between the other includes.

```
#include <cstring>
```

- Edit the makefile with a text editor, and locate the section on line 74,

```
ifeq ($(MACH),x86_64)  # --GNU/Linux/Opteron system.
```

Comment out "GLUT_LIBS" using a #.

- In the sview directory, run make.

```
make
```

Now add sview to your path.

1. Open a terminal
2. Enter the following to edit your .bashrc file

```
sudo gedit ~/.bashrc
```

3. Scroll down to the bottom of the file and add the following. If you compiled sview in a different directory, then change the directory.

```
export PATH=$PATH:~/sview
```

4. Re-source your .bashrc file

```
source ~/.bashrc
```

4 Using semtex - General Workflow

4.1 Session File - no extension

e.g. taylor2 session file

4.2 Generate Mesh File - (.msh)

```
meshpr sessionFile > sessionFile.msh
```

To visualise the mesh using supermongo

```
sm
meshplot sessionFile.msh 1
meshnum
meshbox
quit
```

4.3 Global Numbering Scheme - (.num)

If the session file is changed this needs to be rerun.

```
enumerate sessionFile > sessionFile.num
```

4.4 Initial Conditions - Restart File - (.rst)

Based on information in <USER>< /USER>.

```
compare sessionFile > sessionFile.rst
```

4.5 Solving - generate field file - (.fld)

4.5.1 Using openBLAS threading

```
dns sessionFile
```

4.5.2 Using openMPI with semtex

To use openmpi, the dns_mp exectuable must be called with

```
mpirun --np X dns_mp sessionFile
```

where, X is the number of processes to be used. The number of planes per processor must be even, set in the sessionFile, and the number of processes at maximum can be half of N_Z.

4.6 Post Process Output - (.fld),(.vtk)

For tecplot output,

```
sem2tec -m sessionFile.msh sessionFile.fld
```

If using a cylindrical coordinate system, then it is useful to extend the data by one additional plane in the z direction, using the data from the first plane. This can be used to complete a full revolution. This is achieved using the -w option.

```
sem2tec -w -m sessionFile.msh sessionFile.fld
```

For vtk output (can be used with python script)

```
sem2vtk -m sessionFile.msh -o sessionFile.vtk sessionFile.fld
```

It you want to generate a tecplot file which doesn't do an interpolation to the grid, but rather uses the actual mesh points, then add the -n0 option.

```
sem2tec -n0 -w -m sessionFile.msh sessionFile.fld
```

5 Other Useful semtex Information

5.1 save

Requires a field file (fld).
The save utility,

```
save sessionFile
```

- Creates a folder called Runs
- Updates the restart file to point to the latest field file
- Creates compressed mdl (Time series of kinetic energies in fourier modes) and flx (Time series of pressure and viscous forces integrated over the wall boundary group) file
- Updates the sequence file (hidden, .sequence. Contains a single digit to determine what the current sequence number is)

5.2 History Points

Are defined with

```
<HISTORY NUMBER=1>
# tag x y z
        1 0 0 0
</HISTORY>
```

The history points **must be inside the mesh** and can be on the last plane in the z direction.
If there are issues with the mesh file being empty, run dns with the -v option (for verbose) and see what the error is.
The output will be of the form

```
<ptNum> <time> <var1> <var2> <var3> <var4>
```

5.3 Calculate other field variables

addfield converts a field file to a selection of field variables, e.g. divergence, vorticity, strain rate magnitudes e.t.c. View the options with addfield -h.
e.g. To calculate the vorticity field, and save it to temp.fld,

```
addfield -v -s sessionFile sessionFile.fld > temp.fld
```

5.4 Adding Noise To a Solution and Creating a Restart File

Noise can be added to the solution, and the output used to start another simulation using a restart file.
p is the perturbation magnitude
m is the mode to add the pertubation to. Leaving this blank adds to all modes

```
noiz -p 0.001 -m 1 session.fld > session.rst
```

Don't use save after this, until you've run the simulation again once (otherwise it'll overwrite the rst file and you won't have any perturbation).

5.5 Projecting from 2D to 3D

To project a 2D solution onto n z-planes, use

```
project -z n sessionFile.fld > sessionFile.rst
```

This can be chained with the adding noiz command.

```
project -z 16 sessionFile.fld | noiz -p 0.001 -m 1 > sessionFile.rst
```

6 Using Supermongo

6.1 Visualise Mesh

- Start supermongo with

  ```
  sm
  ```

  A black window should appear.

- Plot the mesh

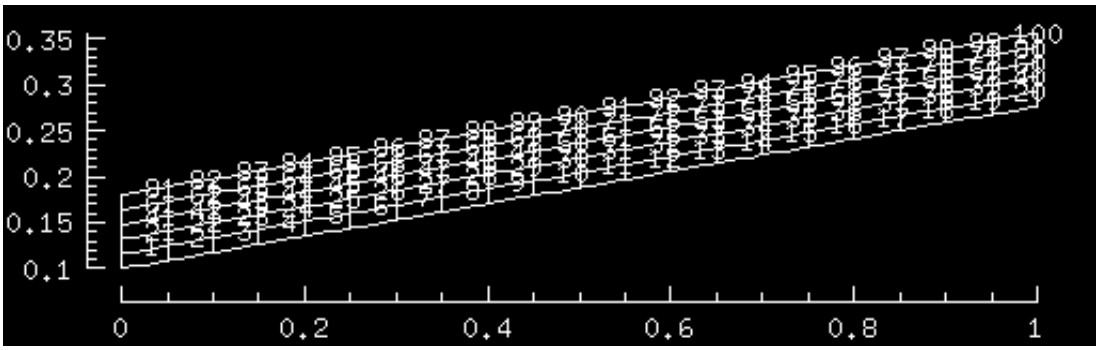
  ```
  meshplot meshfile.msh 1
  ```

  1 shows the grid within the elements, 0 just shows the element outline

- Add mesh numbering and axes

  ```
  meshnum
  meshbox
  ```

  The mesh should now be shown.



- Quit supermongo with

  ```
  quit
  ```

6.2 Plotting History Points

This requires a history file be present, outputFile.his.

6.2.1 2D History File

This will plot u, v, p on three separete plots in the same figure, and can also be used on 3D history files if w doesn't matter.

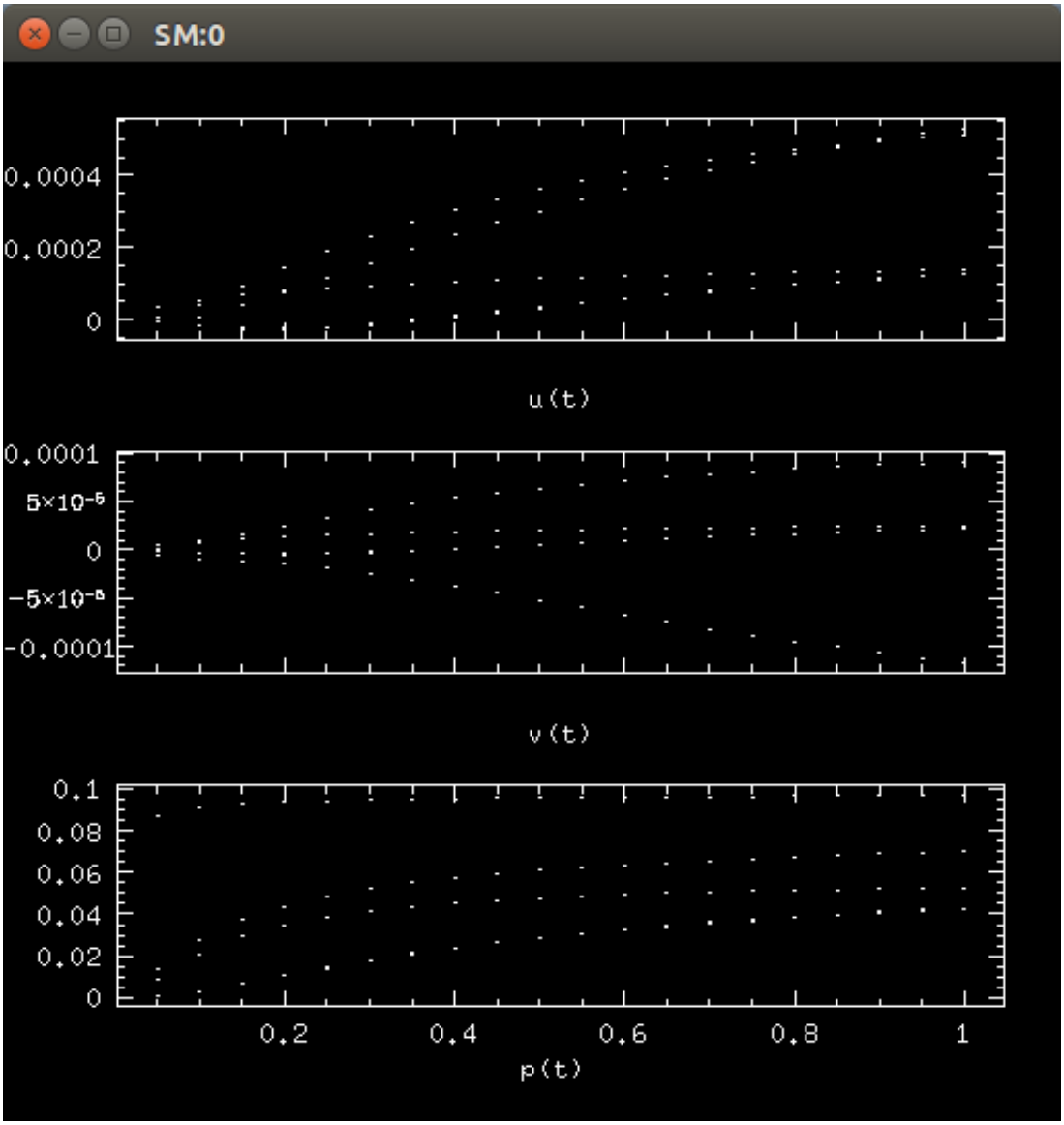
- Start supermongo

```
sm
```

- Plot the history data

```
his2d outputFile.his
```

The history plot for the u, v, p will be displayed.



- Quit supermongo

```
quit
```

6.2.2 3D History File

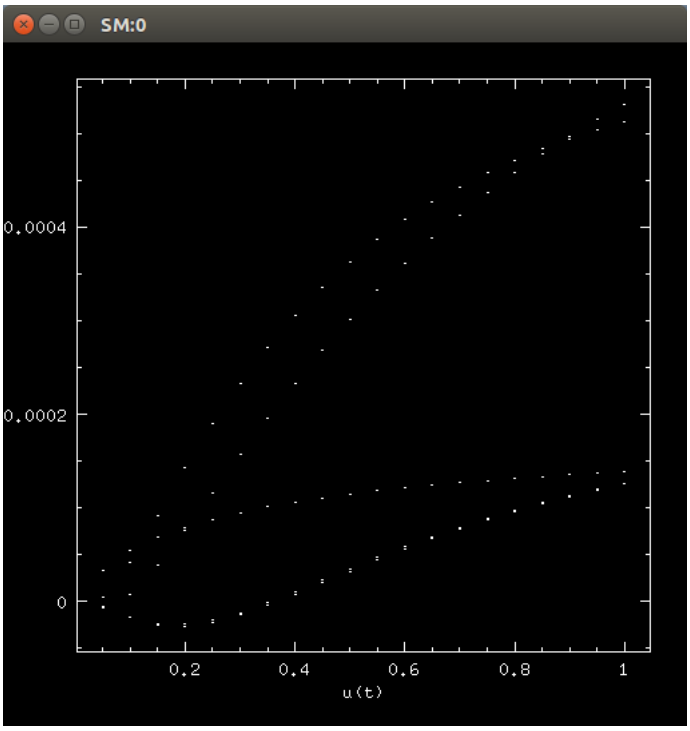
- Start supermongo

```
sm
```

- Plot the history data

```
his3d outputFile.his
```

The history plot for the first variable will be displayed.



- Cycle through the variable by hitting Enter
- Quit supermongo

```
quit
```

6.3 Plotting Modal Energies

This requires an .mdl file.

```
sm
mode0 outputFile.mdl
moden outputFile.mdl
```

6.4 Saving to file (postscript)

- Change the device to postscript and give it the filename to save as

```
device postfile test.ps
```

- Run the macro to create the plot (the plot will not be shown)

```
his3d outputSessionFile.his
```

- Switch back to the previous device. The file will now be created.

```
device x11
```

7 Tecplot Usage

7.1 Launch tecplot from the command line

It will be useful if the tecplot directory is added to your path.

1. Open a terminal
2. Enter the following to edit your .bashrc file

```
sudo gedit ~/.bashrc
```

3. Scroll down to the bottom of the file and add the following. If you installed tecplot to a different directory, then change the directory.

```
export PATH=$PATH:/usr/local/tecplot360ex/bin
```

4. Re-source your .bashrc file

```
source ~/.bashrc
```

5. Tecplot can now be run via command line using

```
tec360
```

7.2 Transform into cylindrical coordinates

1. Create an equation file, called cyl.eqn, enter the following

```
{YC} = {Y}*cos({Z})
{ZC} = {Y}*sin({Z})
{VC} = {V}*cos({Z}-{W}*sin({Z})
{WC} = {W}*cos({Z})+{V}*sin({Z})
```

2. Load in your tecplot file, .plt
3. Go to Data>Alter>Specify Equations
4. Click Load Equations and select the cyl.eqn file
5. Click Compute
6. Go to Plot>Assign XYZ
7. Change Y to YC and Z to ZC
8. To display, tick off Shade, tick on Contour and click Yes to surfaces for active zones
9. The variable being displayed can be changed by click Details next to contour, and changing the variable at the top left of the window

7.3 Macros

7.3.1 Recording Macros

- Run tecplot
- Go to Scripting>Record Macro
- Enter the name of the macro file
- Select Insert "Graphics Off" to not show the animation when the macro runs
- Do actions which you want to record for the macro
- Select Stop Recording

7.3.2 Running Macros via Command Line

Macros can be run from command line using

```
tec360 -p macroName.mcr
```

tempte

8 Using sview

sview requires the mesh and field files for the run. The view of the output and what is shown is controlled using a .sv file. As some commands will block in terminal, it is easier to write them in the .sv file and call it when sview is launched.

```
sview -s sessionFile.sv sessionFile.msh sessionFile.fld
```

sview has two modes, *Isosurface Display Mode*, which allows the alteration of the current viewpoint, and *Isosurface Manipulation Mode*, which allows commands to be entered in to change what is being displayed. To switch between the two, make sure the graphics window has focus, and press ESC.

8.1 Isosurface Display Mode

There are two submodes in this mode, rotation and linear displacement. Initially the mode is set to rotation, but to switch between them use the HOME key. The Up/Down, Left/Right arrows and the pg up/down keys can then be used to control the view.

8.2 Isosurface Manipulation Mode

Generally most commands for this mode should be defined in the .sv file that is loaded.
The current field to work with is set using

```
f <field variable>
e.g. f u
```

This will then print the maximum and minimum values of this field.
Next make an isosurface at value x of the current field with

```
m x
e.g. m 0.1
```

When plotted (see below), if the surfaces look inside out (dark colours on the outside, bright on the inside), then they need to be flipped using

```
n
```

The isosurface is then added to storage using

```
a
```

Finally to display the isosurface, enter the following, where y is the numbered isosurface (use l to list current isosurfaces),

```
d y
e.g. d 1 2 3
```

q can then be used to quit sview.

```
q
```

To launch with a window of 200 by 200 pixels, use the -geometry command line option.

```
sview -geometry 200x200+0+0 -s sessionFile.sv sessionFile.msh sessionFile.fld
```

To generate a TIFF image, add the -d command line option. The file will be called sview.tif.

```
sview -geometry 200x200+0+0 -d -s sessionFile.sv sessionFile.msh sessionFile.fld
```

9 Secure Shell (SSH)

9.1 Logging in

```
ssh <username>@<ipAddress>
e.g. bcub3d@192.168.1.4
```

If you want to forward an x11 window to your client computer (for example to show the output of a matplotlib plot), then add the -X argument.

```
ssh -X <username>@<ipAddress>
```

9.2 Using sftp to Transfer Files

Without being logged in, run

```
sftp <username>@<ipAddress>
```

You can exit stfp with 'bye' or 'exit'.

```
bye
exit
```

9.2.1 Navigating Directories

Remote directories use the normal navigation.

pwd	Show **remote** working directory
ls	List files in current **remote** directory
cd dir	Navigate to dir on **remote** computer

Local Directories use the same navigation with l appended to the start.

lpwd	Show **local** working directory
lls	List files in current **local** directory
lcd dir	Navigate to dir on **local** computer

9.2.2 Transferring Files to the LOCAL System

- Navigate to folder on remote system using normal commands

  ```
  cd remoteDir
  ```

- Nagivate to destination folder on local system using l-commands

  ```
  lcd localDir
  ```

- Enter the following to get the file, 'remoteFile'

  ```
  get remoteFile
  ```

- If a directory is desired, then use

  ```
  get -r remoteDirectory
  ```

To get multiple files at once, use mget.

```
mget file1.txt file2.txt
```

9.2.3 Transferring Files to the REMOTE System

- Navigate to destination folder on remote system using normal commands

```
cd remoteDir
```

- Nagivate to folder on local system using l-commands

```
lcd localDir
```

- Enter the following to send the file, 'localFile'

```
put localFile
```

- If a directory is desired, then use

```
put -r localDirectory
```

To put multiple files at once, use mput.

```
mput file1.txt file2.txt
```

10 Terminal Managers

10.1 Useful Things

Sometimes you'll be trying to print at a rate quicker than the terminal can print out. This can cause lag, and make it look like the process is still running for long after it has been killed. To avoid this, use grep on the output to only show say, every 1000 steps.

```
mpirun -np 4 dns_mp sessionFile | grep "000"
```

If you want to grep for multiple words, use egrep.

```
mpirun -np 4 dns_mp sessionFile | egrep "000|CFL|Div"
```

10.2 byobu

byobu is a window manager and multiplexer, similar to screen, but with more functionaility.
Install with

```
sudo apt-get install byobu
```

To start a byobu instance

```
byobu
```

The terminal can then be closed, or logged out of ssh, and apon return, the previous session can be regrabbed.

The help menu can be accessed using,

```
Shift + F1
q to quit
```

10.2.1 Dealing with Windows

The following are useful window commands,

F2	Create new window
F3,F4	Switch between windows
alt + left/right	Switch between windows
ctrl + shift + F3/F4	Move a window

10.2.2 Dealing with Splits

Splits can be quite useful to monitor multiple things at once,

Shift + F2	Make a horizontal split
Ctrl + F2	Make a vertical split
Ctrl + F6	Delete a split
Ctrl + F3/F4	Move a split
Shift + Alt + left/right/up/down	Resize a split
Ctrl + F6	Kill split in focus

10.2.3 Other Useful Commands

To scrollback through the split history, use

```
F7
```

This will allow the up/down arrow keys to go through the history. To return to the terminal, hit enter.

To disable the byobu command keys, so that you can use the corresponding program command keys, enter

```
Shift + F12
```

To toggle them back on, hit Shift + F12 again.

10.3 screen

screen is a window manager, that comes with most linux distributions (and is on the cluster). It will be able to keep a simulation alive, without having to be logged in.
To launch screen, type

```
screen
```

Screen can split windows horizontall, but unless patched, can't split vertically.
To split horizontally hit

```
ctrl + a
Shift + s
```

To switch between splits, enter

```
ctrl + a
tab
```

A new window won't have a terminal running, to start a new terminal inside a split, enter

```
ctrl + a
c
```

To detach a screen, enter

```
ctrl + a
d
```

To reattached a screen, enter

```
screen -r
```

To see list of current screens, enter

```
screen -ls
```

To attach to a specific screen, enter (where number is what is outputted by screen -ls)

```
screen -r number
```

To create a nested screen, enter

```
ctrl + a
n
```

To move between these screens, enter

```
ctrl + a
p
```

To exit a screen, type

```
exit
```

To kill a screen, enter

```
ctrl + a
shift + k
```

If you get stuck with a screen attached but can't access it, then enter

```
screen -r -d number
```

To detach the screen form the terminal that it's in.

10.4 Random Useful UNIX Things

10.4.1 Read a gz file without needing to extract it

```
zcat file.gz
```

11 Python

11.1 Adding Folder to the Python Path

Instead of having to specify the full path to a script, the folder containing the scripts can be added to the Python Path, allowing them to be just called with -m and their name.

To add a folder to the Python Path,

1. In a terminal, open your .bashrc file

   ```
   nano ~/.bashrc
   ```

2. Navigate to the end of the file and add

   ```
   # Add Python Folder to Python Path
   export PYTHONPATH=$PYTHONPATH:<Full Path to folder>
   ```

3. Save and exit with ctrl+O and ctrl+X

4. Reload your .bashrc file

   ```
   source ~/.bashrc
   ```

11.2 Running a Python Script Contained in the Python Path

To run a script in the Python Path, use the -m option. Tabbing will auto-complete the script names.

```
python -m <scriptName> <args>
```