Introduction to the CASTEP code

Stewart Clark
University of Durham

The CASTEP Developers Group (CDG)

- Stewart Clark (Durham)
- Keith Refson (STFC-RAL)
- Matt Probert (York)
- Chris Pickard (UCL)
- Phil Hasnip (York)
- Matt Segall (Industry/Cambridge)
- Jonathan Yates (Oxford)
- Mike Payne (Cambridge)

With contributions from a number of collaborators and PGs

What is Castep?

- First principles electronic structure code for predicting properties of materials
- Incomplete summary includes:
  - Total energies (forces, stresses, elastic constants)
  - Electronic structure (electronic charge, potential, band structure, DOS, atomic populations)
  - Geometry Optimisation (atomic positions, cell parameters, external pressure/stress)
  - Molecular dynamics (finite temperature properties)
  - Transition state searches (chemical reaction pathways, diffusion barriers)
  - Phonons (IR and Raman spectroscopy)
  - Electric field response (polarisability, dielectric constants, Born charges LO/TO splitting)
  - Magnetic Response (NMR, Chemical shifts)
  - Core level spectroscopy

A Brief History of (now not so-)New Castep

- July 1999: A meeting of interested parties.
- July 1999 to Jan. 2000: Informal specification group outlines the spec. of a new code.
- Feb. 2000: Implementation of Castep begins. CDG formed.
- Many meetings, train journeys, late nights, discussions, rants, ravings, beer/ curry, 5 new jobs…
- Dec. 2001: First release (with "standard functionality" and first Castep workshop
- New Functionality continuously added since then, eg:
  - 2002: Castep phonon linear response calculations
  - 2003: Castep on-the-fly pseudopotentials
  - 2003: Castep electric field responses
  - 2004: NMR
  - 2005: Wannier functions
  - 2005 Non-local XC functionals
  - 2006: Finite difference phonons
  - 2007: Gamma-point speed enhancements
  - 2008: Core level spectroscopy
  - 2009: Raman spectroscopy

Get Castep Source Code

If you do not already have the source code for Castep then:

- You work in UK institution
  - Email ukcp@dl.ac.uk, get a license form and we'll send a CD
- You work elsewhere:
  - Talk to one of us, sign a form, get a CD

Design Philosophy of Castep

- Written specification, prior to coding
  (recently: new science, write code, deeper understanding, write again!)
- Implemented in F95
- Modular approach
- Data abstraction using derived data types
- Overloading for simple, clear subroutine names
- Easy(!!) to add new functionality

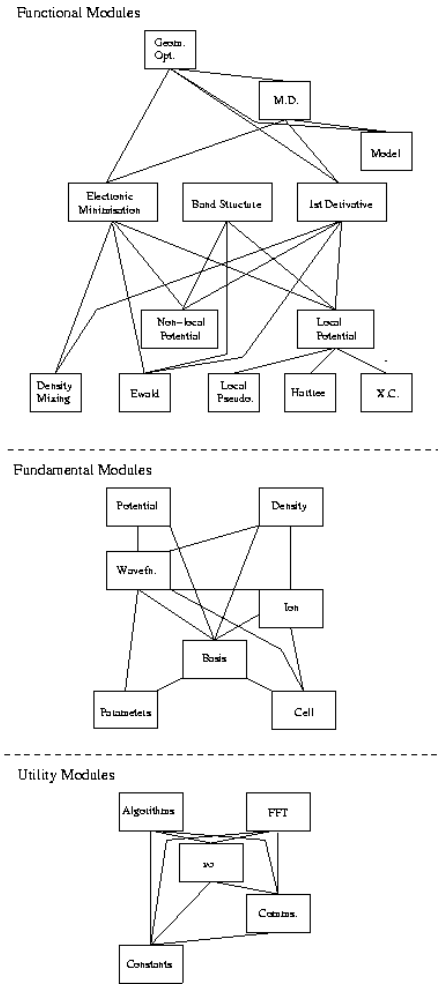
Understanding the code

The next few slides concern the design of Castep:

Important if you want to design new features yourself, or if you want to know how we do it!

It's taken 10 years to get to this point, so I hope you don't mind this slight deviation in topic.

Code Structure (Overview)



Functional Modules

- The 'Physics'

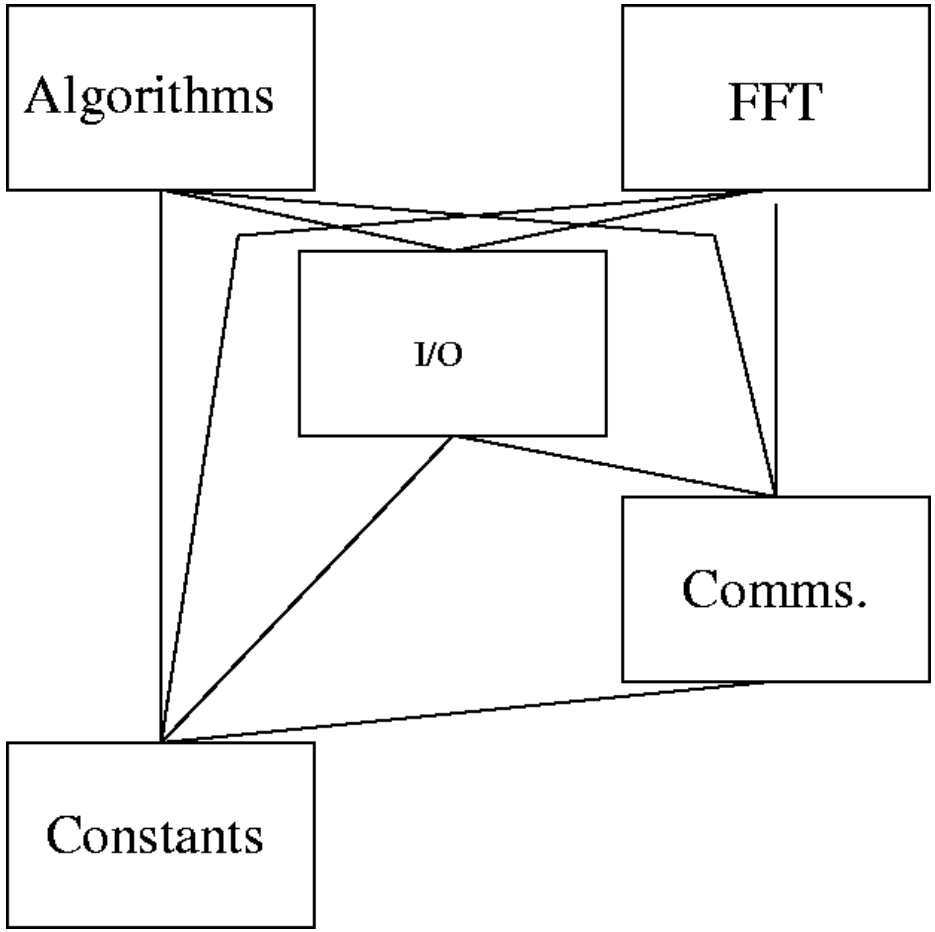
Fundamental Modules

- The building blocks
- Define types and operations

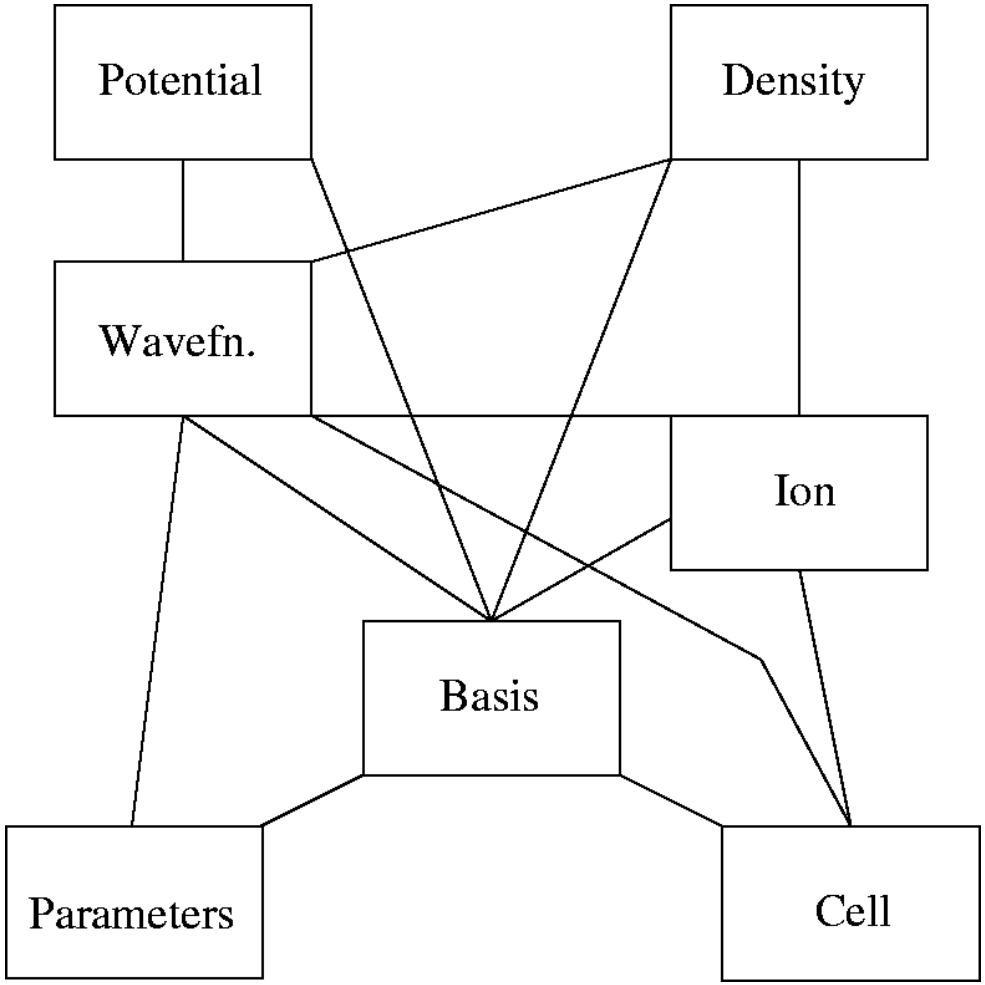
Utility Modules

- Machine-dependent
- Low-level algorithms

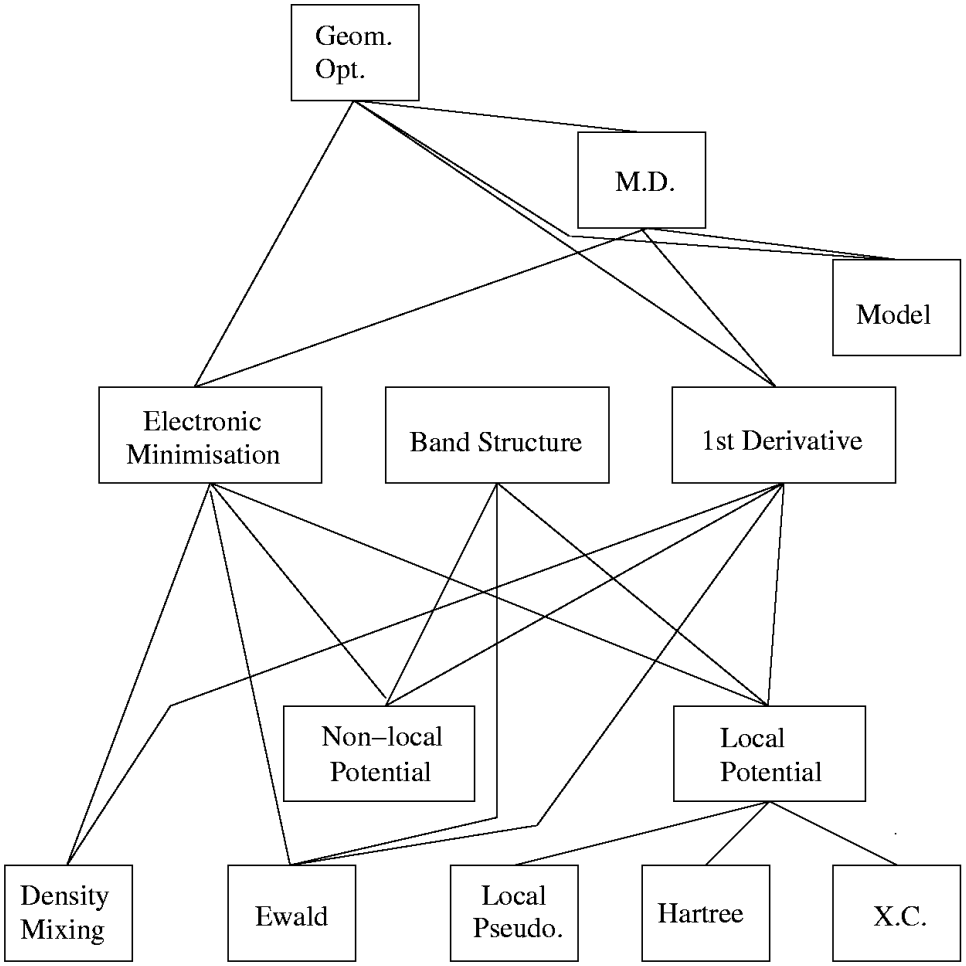
Utility Modules



Fundamental Modules



Functional Modules



Coding Style

Clear code

- •Meaningful variable and subroutine names
- •Lots of comments
- •Good structure

Care with performance issues

- •Some features of F95 sub-optimal
- •Use BLAS/LAPACK where applicable
  - -This gives optimal speed
  - -Can link in machine specific BLAS/LAPACK

FFT can dominate

- -Generic FFT
- -Vendor FFT
- -FFTW

Compiling Castep

Top level Makefile contains two main options:

- COMMS_ARCH
  - serial (for serial code)
  - mpi (for parallel code)
- FFT
  - default (will use GPFA FFT)
  - vendor (will use vendor supplied FFT)
  - fftw/fftw3 (will use FFTW - www.fftw.org)

Compiling Castep

- Type “make”. Wait for 30 minutes…
- If you are using a “standard” unix install and compiler this will compile the code
- Executable in machine specific directory

e.g. `New_Code/obj/linux_ia32_g95/castep`

(Non-standard compiler options can be added in `New_Code/obj/platforms/*.mk`)

Pre-built Castep

- For those not familiar with compiling or developing code
- We provide some “standard” pre-built binaries for several flavours of Unix, Mac OS and Windoze

Now for the IMPORTANT bit!

Running CASTEP

Example on UNIX/Mac machine using the command line

Command-line Castep

Castep can be run within Materials Studio or can be run as a stand-alone programme.

We will now discuss Castep as a stand-alone package

2 Input files

`<name>.cell`

`<name>.param`

Run with "`castep <name>`"

Input Files

The `cell` file

- Contains details of what you want to do a calculation on
  - Atomic positions
  - Unit cell

The `param` file

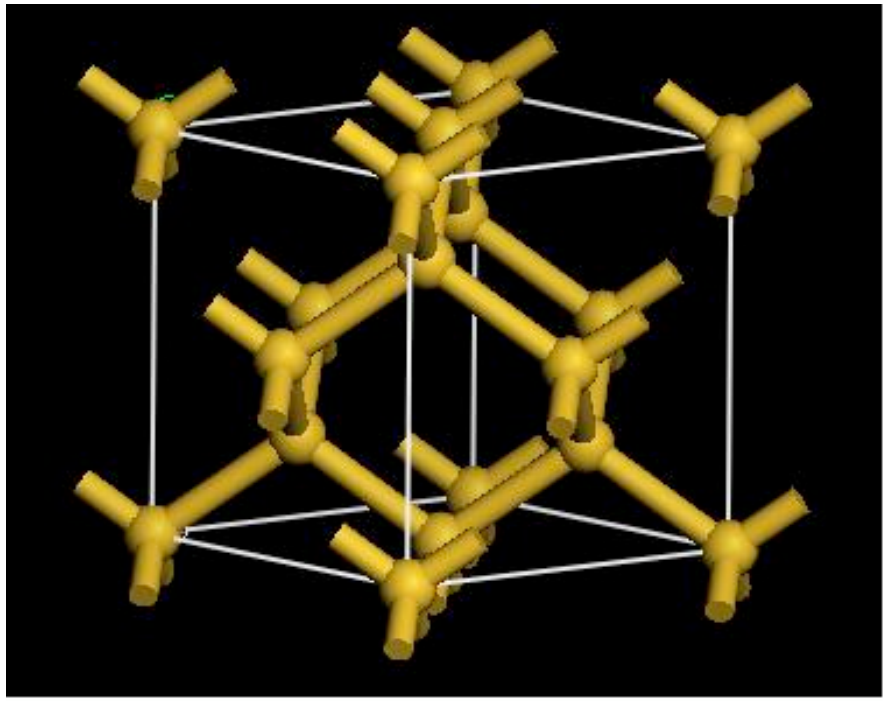
- Contains details of what you want to do
  - Single Point energy (Just solve $H\Psi=E\Psi$)
  - Geometry
  - Band Structure
  - Phonons
  - NMR
  - etc…..

Input Files

- All input parameters have a sensible default (except crystal structure details)
- You just need to specify anything that is different from the default
- Input files contain `keywords`

Example `cell` file

```
%block lattice_abc
5.4 5.4 5.4
90.0 90.0 90.0
%endblock lattice_abc

%block positions_frac
Si 0.00 0.00 0.00
Si 0.50 0.50 0.00
Si 0.50 0.00 0.50
Si 0.00 0.50 0.50
Si 0.25 0.25 0.25
Si 0.75 0.75 0.25
Si 0.75 0.25 0.75
Si 0.25 0.75 0.75
%endblock positions_frac
```



The `.castep` output file

- Contains
  - The header (version number, references)
  - A summary of the parameters used
  - A summary of the cell used
  - A summary of the results of the calculation

Other output files

- `.err` - error messages
- `.check` - restart data

Files depending on the task

- `.geom` - geometry relaxation data
- `.bands` - band structure data
- `.phonon` - phonon data
- .magres – magnetic response (NMR)
- etc…

More Cell File Keywords

```
kpoints_mp_grid 4 4 4

symmetry_generate

%block species_pot
     Si   /usr/local/pseudos/Si_00.recpot
%endblock species_pot
```

Controlling the Calculation: The `param` File

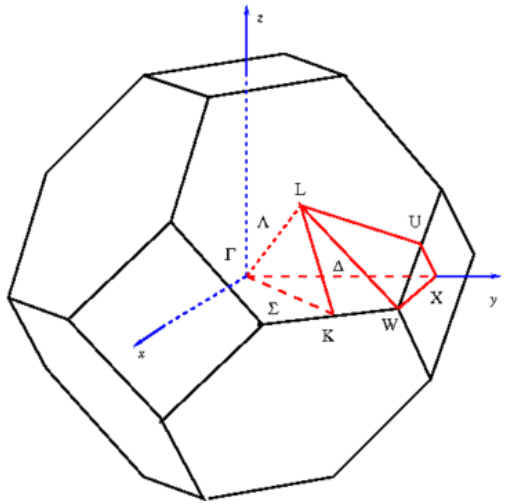
```
Task BandStructure

XC_functional PW91

Basis_precision Precise

Electronic_minimiser CG

Elec_energy_tol = 0.000001 eV
```

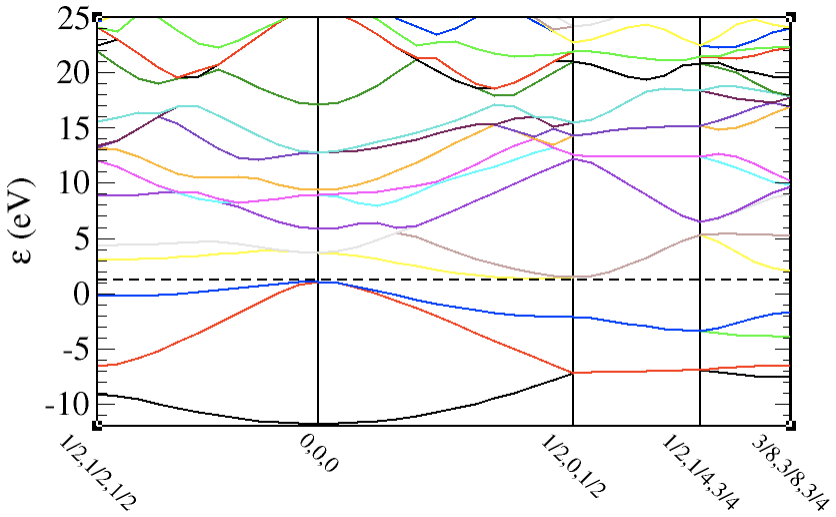
Example: Band Structure

```
%block lattice_cart
2.6 2.6 0.0
2.6 0.0 2.6
0.0 2.6 2.6
%endblock lattice_cart

%block positions_frac
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
%endblock positions_frac

kpoints_mp_grid 4 4 4

symmetry_generate

%block bs_kpoint_path
0.500 0.500 0.500
0.000 0.000 0.000
0.500 0.000 0.500
0.500 0.250 0.750
0.375 0.375 0.750
%endblock bs_kpoint_path
```

task : bandstructure

- castep Si2
- dispersion.pl –xg Si2.bands



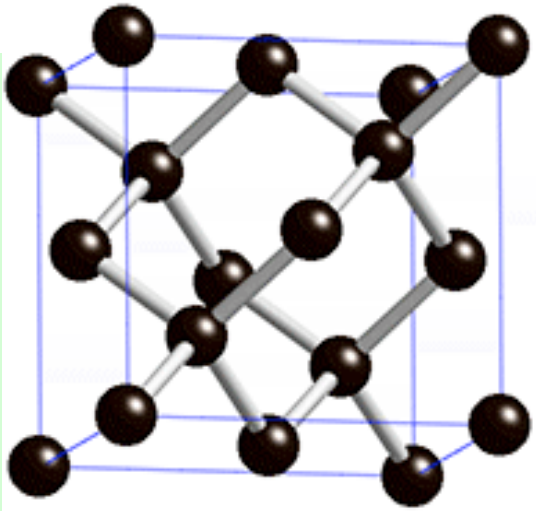


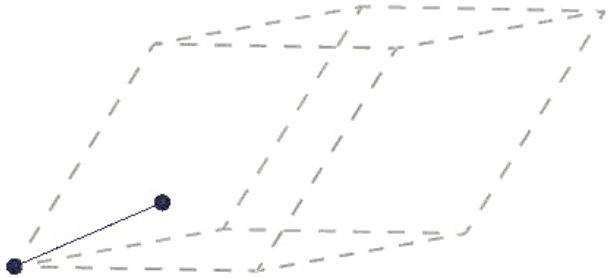
More Parameters

```
task Phonon+Efield

Geom_force_tol : 0.01 hartree/bohr

Fix_occupancies = TRUE

Continuation = my_last_run.check

Energy_unit = kcal/mol
```

Example: Phonon Calculation

```
%block lattice_cart
2.6 2.6 0.0
2.6 0.0 2.6
0.0 2.6 2.6
%endblock lattice_cart

%block positions_frac
Si 0.00 0.00 0.00
Si 0.25 0.25 0.25
%endblock positions_frac

kpoints_mp_grid 5 5 5

symmetry_generate

%block species_pot
Si /usr/local/pseudos/Si_00.recpot
%endblock species_pot
```





```
task : phonon+efield
fix_occupancy : true
opt_strategy : speed
```

Run Castep:
Generate output files –
.castep and .phonon

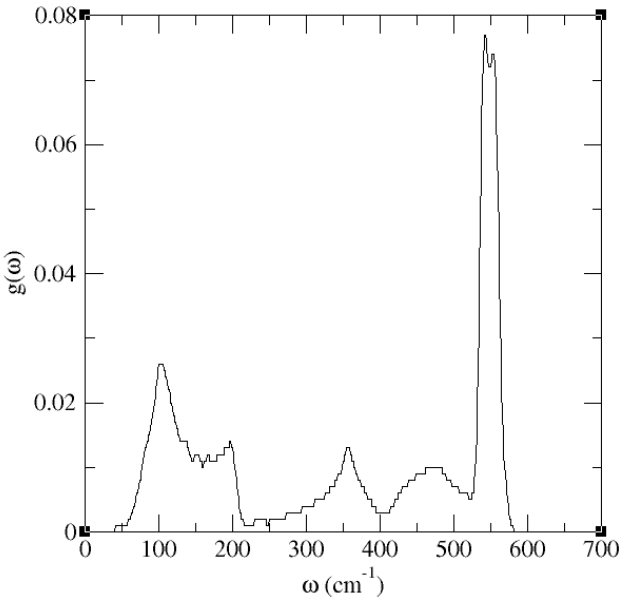
Some phonon analysis

Last run produced phonons on a 5x5x5 grid. Require finer grid?

D.O.S.

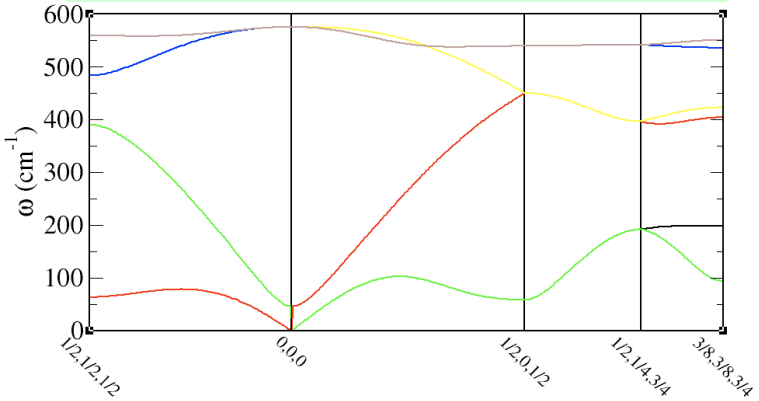
```
phonon_fine_kpoint_mp_grid 20 20 20
```

```
phonon_fine_method : interpolate
continuation : default
```



Dispersion

```
phonon_fine_kpoint_path_spacing 0.01
%block phonon_fine_kpoint_path
   0.500000   0.500000   0.500000
   0.000000   0.000000   0.000000
   0.500000   0.000000   0.500000
   0.500000   0.250000   0.750000
   0.375000   0.375000   0.750000
%endblock phonon_fine_kpoint_path
```



Castep help function

- To search for all keywords containing the phrase <string> use

```
castep -help search <string>
```

- This will print a summary of each of the keywords containing that phrase

- To get help with a particular keyword use

```
castep -help keyword
```

- This will print a detailed help message about that keyword

Running in parallel

Two important things to note:

Number of k-points

Number of cores/cpus

Running on parallel machine

Golden rule: Try to use the number of cores that gives you the highest common factor with the number of k-points

Example of 5-point calculation

10 cores: 48 seconds

13 cores: 172 seconds

Many other considerations, eg. interconnect, etc…

Running on a parallel machine

Can over-do number of cores

- Parallel communication is bottleneck, so there comes a point when adding more cores slows calculation

In DFPT calculations, k-point numbers change due to symmetry breaking

- Investigate phonon_kpoints.f90 code which analyses run with respect to k-point and core number

Update: bands parallel version coming soon, which will help a lot with this

Castep Tools

Castep has a number of tools to help with analisys located in the Source/Tools and Source/Tools/cteprouts directories of the source code distribution.

dispersion.pl
dos.pl
phonon_kpoints
cell2… (eg. cell to cell2pdb, cell2xyz, etc)
…2cell

Investigate the range of tools during the practical sessions