



CONTRACT NUMBER 508830

**DEISA**  
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR  
SUPERCOMPUTING APPLICATIONS**

**European Community Sixth Framework Programme**  
**RESEARCH INFRASTRUCTURES**  
**Integrated Infrastructure Initiative**

Documentation for D-JRA2-2.2,  
D-JRA2-2.3, D-JRA2-2.4

**Deliverable ID: D-JRA2-2.5**  
**Due date: October, 31, 2005**  
**Actual delivery date: November 28, 2005**  
**Lead contractor for this deliverable: EPCC, UK**

**Project start date: May 1<sup>st</sup>, 2004**  
**Duration: 4 years**

<b>Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)</b>		
<b>Dissemination Level</b>		
<b>PU</b>	Public	X
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	

## Document Keywords and Abstract

<b>Keywords:</b>	Grid-enabling, pre- and post-processing cosmology tools, group finders, Friends-of-Friends, FoF, SubFind, MergerTrees, Power, Correl, Virgo, GADGET, GADGET2
<b>Abstract:</b>	<p>This report functions as The User Guide for the tools ported to the DEISA infrastructure, namely FoF, SubFind, the MergerTrees suite, Power and Correl.</p> <p>Specifically, this report is D-JRA2-2.5, and provides the documentation for the reports D-JRA2-2.2, D-JRA2-2.3 and D-JRA2-2.4.</p>

---

**Table of Content**

<b>1 INTRODUCTION .....</b>	<b>2</b>
1.1 Executive Summary .....	2
1.2 References and Applicable Documents.....	2
1.3 List of Acronyms and Abbreviations .....	3
<b>2 VIRGO'S COSMOLOGICAL SIMULATION CODE GADGET. ....</b>	<b>4</b>
2.1 Typical Work Flow .....	4
<b>3 DOCUMENTATION FOR THE TOOLS .....</b>	<b>4</b>
3.1 FoF .....	5
3.1.1 Compilation and Execution.....	5
3.2 SubFind .....	6
3.3 Compilation and Execution .....	6
3.4 MergerTrees suite.....	7
3.4.1.1 TraceSubgroups .....	7
3.4.1.2 SplitHalos.....	7
3.4.1.3 BuildTrees.....	8
3.4.2 Compilation and Execution.....	8
3.5 Power.....	9
3.5.1 Compilation and Execution.....	9
3.6 Correl .....	9
3.6.1 Compilation and Execution.....	9
<b>4 CONCLUSIONS .....</b>	<b>10</b>

# 1 Introduction

## 1.1 Executive Summary

This project is DEISA's JRA2 – Cosmology Applications, and is a joint effort between EPCC and the UK's Virgo Consortium [1]. Funding for JRA2 comes from both DEISA and VirtU [8], where VirtU is Virgo's e-Science Virtual Universe project, funded by the UK Particle Physics and Astronomy Research Council (PPARC). VirtU started on the 1<sup>st</sup> October 2004 and will run for 36 months. VirtU will form the foundations of the Theoretical Virtual Observatory.

All the tasks of WP2 are jointly funded by both DEISA and VirtU and, as such, all deliverables are to be reported to both DEISA and VirtU.

This report is the final deliverable of WP2 "Documentation for D-JRA2-2.2, D-JRA2-2.3, D-JRA2-2.4". This report can be considered as the User Guide for all the tools ported to DEISA as part of JRA2's WP2, namely: Friends-of-Friends (FoF), SubFind, the MergerTrees suite (TraceSubgroups, SplitHalos and BuildTrees), Power and Correl. These tools were ported to the DEISA infrastructure, Grid-enabled and parallelised where necessary.

In this context, by Grid-enabling we mean to allow these tools to utilise distributed data sets within the DEISA infrastructure by introducing a portable binary data format into the tool's I/O routines. This format was chosen [3] to be HDF5 [13].

Demonstrator codes are available from the authors, however, it should be noted that these codes belong to Virgo and are not public. The purpose of the DEISA research infrastructure is to enable scientific discovery across a broad spectrum of science and technology. The Virgo Consortium are typical of new users of the infrastructure requiring support and modest development of their software applications to take advantage of the new infrastructure. This work has been done under the auspices of the DEISA and VirtU projects and as a result a new community of scientific researchers have been brought to the infrastructure. The software applications are available for other users under the existing terms of the Virgo Consortium who are the application owners.

## 1.2 References and Applicable Documents

- [1] D-JRA2-1.2, Grid-enabled implementation of GADGET for metacomputing environments.
- [2] D-JRA2-2.1, Specification Document for D-JRA2-2.2, D-JRA2-2.3, D-JRA2-2.4.
- [3] D-JRA2-2.2, Grid-enabled "Group finders".
- [4] D-JRA2-2.3, Grid-enabled tools to build halo merger trees.
- [5] D-JRA2-2.4, Grid-enabled tools for evaluating the basic properties of the dark matter.
- [6] D-JRA2-1.2, Grid-enabled implementation of GADGET for metacomputing environments.
- [7] Virgo: <http://www.virgo.dur.ac.uk>.
- [8] VirtU: <http://star-www.dur.ac.uk/~csf/virtU/virtU-final.pdf>.
- [9] GADGET: <http://www.mpa-garching.mpg.de/galform/gadget>.

[10] DEISA: <http://www.deisa.org>.

[11] Barnes Hut Treecode: Barnes, J., and P. Hut, *Nature*, 324, 1986.

[12] Monaghan, J.J., *ARA&A*, 30, 543, 1992.

[13] HDF5: <http://hdf.ncsa.uiuc.edu/HDF5>.

### **1.3 List of Acronyms and Abbreviations**

<b>FFT</b>	<b>F</b> ast <b>F</b> ourier <b>T</b> ransform
<b>GADGET</b>	<b>G</b> ALaxies with <b>D</b> ark matter and <b>G</b> as intErac <b>T</b>
<b>MPI</b>	<b>M</b> essage <b>P</b> assing <b>I</b> nterface
<b>HDF5</b>	<b>H</b> ierarchical <b>D</b> ata <b>F</b> ormat <b>5</b>
<b>VirtU</b>	<b>V</b> irtual <b>U</b> niverse

## 2 Virgo's Cosmological Simulation Code GADGET.

The codes described in this document are all used as pre- and/or post-processing tools by the Virgo [7] Consortium's current, main Cosmological Simulation Code, namely GADGET2 [9].

GADGET is a code used to simulate the evolution of the universe, modelling the motion of both dark matter and gas and, essentially, is a simulation code that can be used to model very large N-body systems. The code calculates the gravitational evolution of the dark matter and gas in two separate calculations. The force on both dark matter particles and the gas particles is computed using an FFT for long-range interactions, whilst a Treecode, similar to the Barnes-Hut Treecode [11], is employed for the short-range interactions. Further hydrodynamic forces for the gas component are calculated using Smooth Particle Hydrodynamics (SPH) - see [12] for a good overview of SPH methods.

As part of JRA2, WP1, GADGET2 was ported to the DEISA infrastructure [6]. GADGET2 is the latest version of GADGET available.

### 2.1 Typical Work Flow

This section gives a brief outline of how the tools are currently employed to post process data produced by a large cosmological simulation.

1. Firstly, one runs a large cosmological simulation using GADGET2. This code produces a number of snapshots (output files) of the simulation at different times in the system's evolution.
2. Two codes, namely *Power* and *Correl*, can be employed to compute the power spectrum and correlation function, respectively, once the snapshot files have been produced.
3. Use *FoF* and *SubFind* to identify dark matter halos and subhalos for each of the snapshots produced.
4. Use the *MergerTrees* suite to find the formation history (referred to as merger trees) of each halo in the final simulation output frame. MergerTrees takes snapshot files and the output from FoF and SubFind as input.

To date, the Virgo users have not used UNICORE to construct automated workflows. This is essentially a sociological issue - the Virgo users are accustomed to running the tools manually. Furthermore, it is often essential to review the output of one tool before using it as input to the next tool - an automated workflow is not well suited to this usage pattern. As the users become more familiar with the DEISA infrastructure we have no doubt they will experiment with UNICORE and automated workflows.

## 3 Documentation for the Tools

The section contains the necessary documentation required to use the tools considered under WP2, namely: FoF, SubFind, MergerTrees (TraceSubgroups, SplitHalos and BuildTrees), Power and Correl.

The purpose of the DEISA research infrastructure is to enable scientific discovery across a broad spectrum of science and technology. The Virgo Consortium are typical of new users of the infrastructure requiring support and modest development of their software

applications to take advantage of the new infrastructure. This work has been done under the auspices of the DEISA and VirtU projects and as a result a new community of scientific researchers have been brought to the infrastructure. The software applications are available for other users under the existing terms of the Virgo Consortium who are the application owners and, as such, are not currently available centrally nor publicly on the DEISA infrastructure. The demonstrator codes are available from the author.

The tools in question were developed by the Virgo Consortium to help them do their science. They were never intended to be used outside the consortium, where their use is well understood, therefore, as such, there is no other documentation available.

To use these tools, one must have GADGET2 check-pointed data files (*snapshots*). These snapshots contain the positions, velocities, ID numbers and other information (masses, density, star formation rate etc) for all of the particles in the simulation at a particular simulation time. One snapshot can consist of a single file, or the data may be split over several files corresponding to the particles from some subdomains of the simulation volume. The files are generally named `snapshot_xxx.yyy`, where `xxx` is the number of the snapshot (010 is the final snapshot) and `yyy` corresponds to the number of this file in the sequence within the snapshot when a multi-file snapshot is employed. For the Demonstrator, we have used 8 files per snapshot.

To run GADGET you need an initial conditions file (e.g. the binary `gadget.gdt`), a parameter file and a Makefile.

This initial conditions file has about 500,000 gas and dark matter particles more or less evenly distributed about the simulation volume. This corresponds to a snapshot of a smallish section of the universe just after the big bang. When you run GADGET2, it will evolve the mass distribution to the present day, by which time some clumps of particles should have formed.

### **3.1 FoF**

Once GADGET2 has been run, the first task is to locate groups within the *snapshots*. These snapshots are then employed to determine gravitationally bound dark matter groups.

The Friends-of-Friends (FoF) codes are actually a collection of *group finder* codes. Here, we describe how to employ a code called *P-GroupFinder\_Special*, which caters for both dark matter and gas. This is a parallel C/MPI code has around 3000 lines of code

#### *3.1.1 Compilation and Execution*

This latest version of FoF can be run using input and output data in either the original binary data format or using HDF5. The input/output formats are specified through command line arguments: 1 for the native binary format and 2 for HDF5 for the input read-in and also for the output produced.

Furthermore, the code reads particles in groups of 10000. This number is determined by `IO_BUFSIZE=10000`, which is defined near the top of `io_input_hdf5.c`. The adjacent parameter `BUFSIZE` is the number of particles to send to the other processors in each `MPI_SSEND` operation. This is also set to 10 000, however, it may be prudent to reduce this figure to increase performance.

To run FoF all you need is a suitable GADGET snapshot - there is no parameter file and the only option set in the Makefile is whether you have the HDF5 library or not. The *linking length* and the *minimum group mass* are set in the source code, but you should not need to change these. The input and output formats are command line arguments. The As already stated this demonstrator has 8 files per snapshot.

Compilation warnings may occur, as the types of a couple of parameters have changed recently in HDF5 v6.1.4 from v6.1.3. If this occurs, one must change any occurrences of `hssize_t` in the C source code to `hsize_t`. The Makefile must be edited to contain the location of the HDF5 library.

The parameter file for the FoF code is called `fof.param`.

The command line parameters for the FoF code are:

```
P-GroupFinder_Special <path> <basename> <num> <i> <o>
```

where `<path>` contains the snapshot files, `<basename>` is the part of the snapshot filename before the underscore ("snapshot" in this case), `<num>` is the snapshot number (0 to 10 for these simulations), and `<i>` and `<o>` are or the input and output formats, respectively, and are each set to 1 for the original native binary or 2 for the HDF5 format.

Thus, to run FoF, using HDF5 for input and output, use the following.

```
./P-GroupFinder_Special Gadget_Test snapshot 010 2 2
```

### 3.2 SubFind

Once a FoF group finder code has been run, which will have located the Friends-of-Friends dark matter halos, SubFind can then be used to locate the *subhalos* in the given snapshot.

After loading the particle data and the group catalogue of the snapshot, the dark matter density of all particles that are bound in groups is calculated. The search for neighbours is not restricted to particles in the FoF group. SubFind considers all dark matter particles. After this step, this process is applied to each FoF halo in turn, including an unbinding procedure, where the gravitational potential is computed with a tree algorithm. The net result of SubFind is that each FoF halo is decomposed into a set of substructures that are gravitationally bound. There can also be a number of particles left over ("fuzz") that are not bound to any of the subhalos

### 3.3 Compilation and Execution

Compilation warnings may occur, as the types of a couple of parameters have changed recently in HDF5 v6.1.4 from v6.1.3. If this occurs, one must change any occurrences of `hssize_t` in the C source code to `hsize_t`. The Makefile must be edited to specify the location of the HDF5 library.

The parameter file for the SubFind code is called `subfind.param`.

The command line parameters for the SubFind code are

```
FoF_Special_subfind <path> <basename> <num> <i> <o>
```

where <path> contains the snapshot files, <basename> is the part of the snapshot filename before the underscore ("snapshot" in this case), <num> is the snapshot number (0 to 10 for these simulations), and <i> and <o> are or the input and output formats, respectively, which can be set to 1 or 2 to specify whether the original native binary or HDF5 format is used, respectively.

Thus, to run FoF, using HDF5 for input and output, use the following:

```
./FoF_Special_subfind Gadget_Test snapshot 010 2 2
```

### 3.4 MergerTrees suite

Thus after the GADGET2 simulation has been run and after each of the FoF and SubFind codes have found the Friends-of-Friends halos and *subhalos* for each snapshot, we then need to determine the merger history of each dark matter halo. This is achieved by taking the output of the FoF and SubFind codes and running the MergerTrees code suite.

The MergerTrees suite consists of three codes, namely: *TraceSubgroups*, *SplitHalos* and *BuildTrees*, which are run in sequence.

These three codes are parallel Fortran90 codes, employing MPI, plus some serial ANSI C codes used for I/O. *TraceSubgroups* has around 3000 lines, *SplitHalos* has 1500 lines, and *BuildTrees* has around 200 lines.

#### 3.4.1.1 TraceSubgroups

For each subhalo, at each output time, *TraceSubgroups* attempts to find the 'same' subhalo at the next output time by following the particles that belong to the original subhalo. If the subhalo cannot be found at the next output time, the code may be able to find it at later output times.

For each subhalo, at each snapshot, this code outputs the ID number of the descendant subhalo and the associated snapshot (which is usually simply the next snapshot).

#### 3.4.1.2 SplitHalos

The FoF code often merges objects that probably should be treated as independent halos in the merger trees, so certain criteria are employed to decide if any of the subhalos in a particular halo should be split off and treated as halos in their own right. These criteria are:

- 1) Whether the subhalo is within twice the half mass radius of the halo and
- 2) Whether the subhalo has been stripped of some fraction of its mass since it was last identified as a separate halo.

The output from *SplitHalos* is a data structure that defines a 'cleaned' halo catalogue.

### 3.4.1.3 *BuildTrees*

This code uses the 'cleaned' halo catalogue, as produced by SplitHalos, and the descendant IDs, as calculated by TraceSubgroups, to build the merger trees. Each halo at the final time is the 'root' of a merger tree. The program starts at the final output time and works backwards in time, adding each halo to whichever merger tree this descendant belongs to.

### 3.4.2 *Compilation and Execution*

To build merger trees we need a large number (usually at least 50) of snapshots.

There are HDF5 C wrapper routines called from Fortran and the same C routine can be called with arguments of different types. Some Fortran compilers do not allow this, as this is not allowed for Fortran routines. However, most compilers can be forced to accept this.

NB the file `hdf5_wrapper.F90` has pre-processor directives and thus must be pre-processed before it is compiled.

The Makefile must be edited to specify the location of the HDF5 library.

To run the MergerTrees suite, you need a set of HDF5 GADGET snapshots and the corresponding HDF5 group files from FoF and SubFind, as described previously. Each code reads its parameters from a correspondingly named parameter file: `trace.param`, `split.param` and `build.param`. There are no command line arguments.

To run on another simulation you'd need to replace the location of the executables in the three parameter files and set the parameters `ifirst` and `ilast` to the indices of the earliest and last snapshots, respectively. The simulation directory should contain the HDF5 snapshot files and the subdirectories `groups_catalogue` and `groups_subhalos` which are produced by FoF and SubFind. If the snapshot names start with something other than `snapshot` you'll also need to change the `snapbase` parameter.

The output directory (`outdir` in the parameter files) should be the same for all three codes and must already exist. There is a `trees` subdirectory in the demonstrator directory for this purpose.

The output directory needs to contain one subdirectory per snapshot with names of the form `treedir_xxx`, where `xxx` is the snapshot number padded to three characters with leading zeros.

The final merger tree files appear in the `treedir` corresponding to the final snapshot. Some intermediate files are generated in the other directories.

Note that all three codes need to be run on the same number of processors.

### 3.5 Power

The Power code, call L-Power, is used to calculate the power spectrum of a dark matter distribution.

#### 3.5.1 Compilation and Execution

L-Power reads all of the particles of one type, e.g. gas particles, from a snapshot, calculates the density field on a mesh, and then does a Fast Fourier Transform to get the power spectrum. It needs some command line parameters to run; however, there is no parameter file.

The parameters are:

```

simdir   - directory containing the hdf5 snapshot files.
basename - the main part of the snapshot filename (before the underscore and
           the snapshot number).
isnap    - the snapshot to read.
ispecies - which type of particle to use, where this is set to 1 for the demon
           strator, as we wish to find the power spectrum of the dark matter.

```

The code writes the power spectrum as an ASCII table, namely `powerspec.txt`.

This program is the original L-Power code with a new routine, `read_snapshot.c`, which reads a HDF5 snapshot and distributes the particles.

### 3.6 Correl

The Correl code, called L-Correl, is used to calculate the correlation function of a dark matter distribution. L-Correl is quite sophisticated, in that the correlation function can be measured accurately, even for large scales, in a relatively fast time, when compared to more traditional methods.

L-Correl reads the dark matter particles from a snapshot and finds the two point correlation function by counting the number of pairs of particles as a function of separation. To do this it picks a random sample of particles. It places a sphere around each selected particle and calculates the distance to all of the other particles in the sphere. The radii of the spheres are chosen at random.

#### 3.6.1 Compilation and Execution

There are some pre-processor macros in `twopoint.c` that determine how the sampling is done:

- `FRACTION_TP` determines what fraction of the particles have spheres placed around them. The actual fraction used is  $\text{FRACTION\_TP} \times \text{np} / 1.0\text{e}7$  where `np` is the number of particles. We have set this to be very large so that all of the particles are used, but if the code runs too slowly, then the fraction could be reduced to, say,  $\text{FRACTION\_TP} \times \text{np} / 1.0\text{e}7 = 0.1$ .

- ALPHA determines the slope of the distribution from which the radii of the spheres are chosen. Originally, this was set to -1.0. We have considered a value of -0.5 (which gives larger spheres), which reduces the noise at large scales a little but this increased the execution time.

If the code is compiled with `-DDEBUG`, then the random numbers used to do the sampling are taken from a large table which is the same on all processors so that the output is independent of the number of processors. This does mean that each processor must store three numbers for every particle in the simulation, which puts a limit on the size of simulation one can consider.

L-Correl takes two command line arguments. The first is the name of a parameter file and the second is the name of the output file. There is an example parameter file in the demonstrator, called `parameterfile` in the source code directory. This contains comments that explain the parameters employed. One parameter that may need changing is the `SnapshotFile` parameter.

The output file is an ASCII table with the correlation function as a function of scale. The columns `Count` and `CountSpheres` give the number of particle pairs found and the number of spheres placed.

L-Correl is the same as the Correl code within L-GADGET2, which is a specialised version of GADGET2 intended for very large, dark matter only simulations, where the input routine of L-Correl is converted to HDF5.

## 4 Conclusions

This document has described the steps required to run the pre- and post-processing tools, namely FoF, SubFind, the MergerTrees suite (TraceSubgroups, SplitHalos and BuildTrees), Power and Correl on, but not limited to, the DEISA infrastructure.

The purpose of the DEISA research infrastructure is to enable scientific discovery across a broad spectrum of science and technology. The Virgo Consortium is typical of new users of the infrastructure requiring support and modest development of their software applications to take advantage of the new infrastructure. This work has been done under the auspices of the DEISA and VirtU projects and as a result a new community of scientific researchers have been brought to the infrastructure. The software applications are available for other users under the existing terms of the Virgo Consortium who are the application owners.

The benefit to Virgo Consortium users is that the output from GADGET cosmological simulations can now be post-processed from any of the DEISA sites, given that the tools described above have been ported to those sites. This is due to the global single file space offered by DEISA, via MC-GPFS, and the introduction of a portable, binary data format, namely HDF5, into the tools. Thus, no matter where the data is generated, the tools on the DEISA infrastructure can post-process the output without the usual data manipulation required when porting between heterogeneous platforms. Moreover, thanks to the parallelisation of the tools, the size of the simulation data under consideration can be far greater than ever before.