e-Galform 1 year research masters

Mark Seaden

e-Galform
1 Year Research Masters

Mark Seaden
Academic Year 2004-2005
Student No. 000450096

February 2007

Contents

Figures

Abstract

Advancing internet technologies and increasing computer processing and data transfer rates have allowed computers separated by large distances to communicate with each other and transfer large amounts of data that were previously impractical. This has opened new opportunities allowing university departments to share research and information via web servers and web browsers.

In this thesis, I describe the development of e-Galform, an internet based database application that seeks to allow scientists both within the University of Durham and from other universities across the globe to take advantage of Galform, a galaxy formation model developed by theoretical galaxy formation research staff at Durham.

e-Galform features a web based interface allowing users to understand the capabilities of Galform without the necessity to understand the finer underlying technical and scientific complexities, whilst offering documentation that would support further understanding. A user can extract tailored predictions from a library of pre-existing Galform runs using the e-Galform web interface. A further primary feature is the production of Galform data in a new and more verbose data format, VOTable, which may be used in other database applications and is expected to become a standardised data format for use in astronomical software globally. The VOTable format is under development primarily by the United States Virtual Observatory (US-VO). Rather than run the Galform simulation directly, e-Galform extracts requested galaxy properties by running an intermediate binary program (samplegals.exe) on a pre-generated Galform dataset.

e-Galform is also configurable and extendible via the use of in-built administrative facilities. The aim of the administrative facilities is to allow users to extend the facility to extract newly added galaxy properties as the underlying Galform model is extended, without the necessity of requiring new code.

1. Introduction

1.1 A Background To Database Applications In Research

The past few centuries of academia have often been characterised by universities working independently from one another in the same academic areas. Groups of researchers have investigated certain scientific problems and, where computing has been a useful tool in pursuing research, it has often been the case that any standards developed within research groups have remained unique to those groups and cooperation between different universities has often produced differing standards of code and data formats.

For example, in the field of computational cosmology, the FLASH adaptive mesh simulation code (developed at the University of Chicago) produces data in its own native data format only, providing dedicated software tools to reading the output dataset to allow for the use of standard plotting tools such as *gnuplot* {1}. However, semi-analytical galaxy simulation projects such as Galform (based in the University of Durham) produce data in a similarly delimited yet structurally different ASCII data format. {2}

Other similar ambiguities exist between research groups in the field; for example, the GalICS (Galaxies In Cosmological Simulations) group's semi-analytical galaxy formation code produces a dataset capable of being interrogated by a web form via MySQL queries.{3}

With such large datasets involved (the Millennium Simulation, for example, produced over 20TB of output){4} lack of cooperation may be partly due to slow and inefficient communications technologies between universities, which was certainly an issue prior to the introduction of cheap computing equipment during the late 20th century (see Appendix B, graph showing Moore's Law of computer processor transistor density) {5}. The beginnings of the web today are generally traced back to the ENQUIRE system, developed by Tim Berners-Lee in 1980, followed by a more formal proposal for the World Wide Web by Berners-Lee in 1989 {6}.

Ultra-fast internet communication generally began at the behest of the scientific academic community. For example in the UK, the JANET system was introduced in the 1970s, giving rise to the first network capable of delivering 8Mbit/s connections between computers across the internet in the early 1990s, far before such connection speeds were available at the domestic level. {7}

With the advent of the internet and the World Wide Web (WWW) in the early 1990s {6}, science involving computers as research tools is entering a new phase and opens up the possibility for the consolidation of standards that allow academic departments in different universities to understand each other's work more quickly by only learning a set of standard format criteria; saving time for all research groups involved and also enabling more people to help refine and define given standards.

Data can be transmitted electronically across the internet in a wide variety of formats, opening up the possibility for different research groups in different universities to combine together in an interdependent approach to developing new common data processing facilities. This is highly important, as it allows groups of researchers from across the globe – even in different continents – to develop uniform standards that will allow research groups to work on the same project goals and share their results with other universities easily, reliably and efficiently. It also gives other universities access to datasets – and the ability to influence the properties produced and input parameters – that they may not have the facilities to produce themselves.

In particular, the WWW in the last decade has seen a revolution in data transmission methods, moving from basic, static HTML (HyperText Markup Language) {8} that allows information to be displayed in the form of web pages to new, complex database driven websites that generate web pages using programming languages called server-side programming languages.

Server-side programming languages generate web pages, in addition to performing cookie and file management and handling connections and access to a database server if required. When a server-side programming language is used to generate a website using a database, such a site is called a database driven application. Examples of server-side programming languages include PHP {9} (an open source project developed by a large

number of software engineers worldwide) and Microsoft C#.NET through the Microsoft ASP (Active Server Pages) server-side scripting environment, which also supports Microsoft Visual Basic (VB.NET) scripting. {10}. In this dissertation, Chapter 3 discusses the use of web technologies in the implementation of e-Galform.

Database driven applications are particularly advantageous. Firstly, web based database applications are not platform specific; in other words, a database application may be executed on any operating system, provided certain minimal standards of the client's web browser are met (in WWW terminology, the 'client' refers to the computer on which the browser showing the site's content is being viewed). Clients are served by the web server, which generates web pages and sends the data to the client using HTTP (Hyper Text Transfer Protocol).

A major advantage of this software architecture is that it allows the development of applications that may be executed on multiple platforms without the need to develop specific code for each platform. A well-developed web application may be executed via a web browser on a variety of operating systems – for example Windows, Linux or SunOS. In academic circumstances in particular, this is useful as academic departments in scientific contexts may use a variety of different operating systems, in contrast to common business or domestic environments, in which Microsoft Windows is used most often {11}.

Furthermore, as the client is served only web pages by the web server, the client does not require specific technologies unique to the web application installed on their local machine, other than a suitable web browser (see Figure 1.1). This has the advantage of convenience, and also allows the client to use the speed and technologies installed on the central web server, which may far surpass the client's computer in performance. The server and client can communicate with each other, and within reason or specified limits, any number of clients can communicate with the central server. The server responds to client requests for information. The server-side programming language then generates HTML from the installed database, and sends the page to the client as requested. The server-side programming language may also perform other requests on the server, for example file manipulation, before sending HTML to the client (see Figure 1.2).

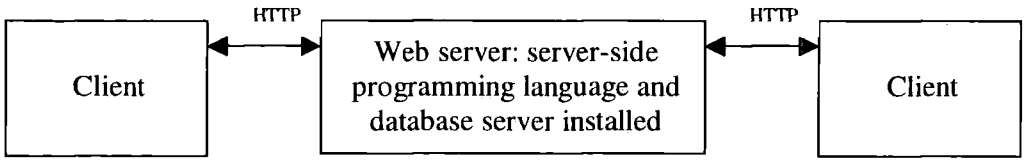

Figure 1.1 – The client/server relationship between a Web browser and Web server {12}

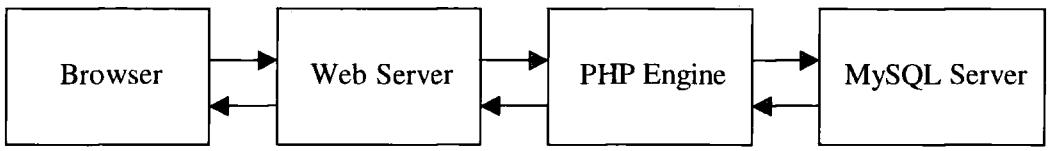

Figure 1.2 – The basic PHP & MySQL Web database architecture consists of the Web browser, Web server, scripting engine and database server using PHP and MySQL. {12}

Because the client's browser simply receives HTML from the server, which is capable of being read by all web browsers (independent of platform or operating system) the client does not need PHP or MySQL installed on their computer in order to use the web application. An excellent example of this today are internet search engines; Google for example is able to serve millions of users per minute, with each client taking advantage of Google's search technology and supercomputers, requiring only a basic client computer to use such technology {13}.

Such an advantage is particularly appropriate to academic circles, where sometimes scientific supercomputers may be used by departments across the globe without the need for staff interference or labour in order to serve data to other research departments according to their specifications. Finally, given appropriate administrative facilities, a web application's database may be configured and expanded upon without the need for administrative users of the application to know the underlying technical nature of the application's development. Changes to user interface or data processing methods, however, will still require a suitably qualified developer. {14}

An expansion in the number of users of a site may lead to increased costs, as server bandwidth must be increased to accommodate an increased number of requests whilst

maintaining transfer rates and an increasing number of users may need to be authenticated and given user quotas at any given time. Chapter 4 of this dissertation will focus on principles of developing the infrastructure of e-Galform using the chosen web development framework.

1.2 Standardising Data Formats

1.2.1 Introducing XML

Other data standards of particular use for transferring data from one computer to another via the WWW include XML and RSS. XML (eXtensible Meta Language) {15} was originally developed in 1996 by a voluntary standards group, and allows a group or individual to define a given data format in verbose terms. Being able to define data verbally makes XML of particular use in sharing scientific data. By storing verbal definitions of column quantities within the dataset, a scientist may lookup a definition of a parameter they do not understand without necessarily having to refer to formal documentation on the source data generation program; or alternatively stored descriptions may point to further resources on the web which allow for greater understanding of the nature of the quantities being described. Given related documentation may be found on the internet, up to date documentation can be found if the web resource is updated and the same links maintained.

For example, other programs capable of reading the XML data format may read verbal descriptions of parameters and match them to their respective programmatic parameter labels, allowing matching of verbal descriptions with labels automatically, and within a presentable and useful context within an application (see Chapter 7 for example of VOTable format data being imported with parameter descriptions in the plotting program TopCat).

A Document Type Definition (DTD) can be created describing the XML format {16}, such that other groups or individuals may develop XML output adhering to the same standard. This is particularly important for research, allowing different groups to cooperate by choosing to adhere to the same strictly defined specification, as well as

perhaps producing their own proprietary standards with consensual agreement. Such changes to a standard may be reflected by updating the DTD.

As an example, a team producing genetic information investigating cancer cells may choose to cooperate with other research teams globally to define a uniform data format. Different research groups working on the same area may wish to produce different data processing applications to derive workable results from the same data – for example, one group may choose to build a database application that will investigate the frequencies of certain genetic patterns within the data; whilst another may choose to build a database application to compare two DNA sequences (for example, the European Bioinformatics Institute website contains data from a variety of sources, downloadable in XML format, however only contains the DNA researched by the institute) {17}. In both cases, with a standardised database format, both research groups may use each other's data seamlessly in their investigations, provided that they have adhered to the given standard. In addition, both research groups may use each other's database applications, because both can be made available on the WWW without need to download and install platform-specific additional software.

The opportunities provided by internet database applications in scientific research are therefore highly notable, and may speculatively lead to a "digital revolution" in scientific research in the coming years with increasing numbers of departments forming groups to develop common standards, developed over time by one or more groups.

The principle of the sharing of data and the sharing of data processing tools between academic research groups on a global scale, and to do so in a way that is professionally presented and easy to use (similar to the practices produced by commercial competition) is the main philosophy of the e-Science movement. Further examples of current major e-Science projects, such as Astrogrid {55} and associated tools, and also grid computing and the main challenges faced in utilising database technologies will be discussed in Chapter 4.

1.2.2 Potential Use Of Open Source Technology In Academic Research

In particular, the past few years have seen the rise of the Open Source development model, with teams of developers pooling together from across the world via the internet to work on particular projects, voluntarily in their own time. Because of the numbers of programmers that may be attracted to a given project, those that attract sufficient numbers are characterised by frequent new incremental releases of both stable and test release software.

Open Source technologies are already very popular for developing web based database applications; for example PHP (Pre-Hypertext Protocol) and MySQL (a database server) are free of charge for all educational institutions to use. MySQL currently shares approximately one third of the market for relational database server software {18}. From a financial perspective this makes Open Source technologies particularly useful in academic environments that are often of limited resources, whilst maintaining the quality and advancement of the available technologies. However, one drawback of Open Source software is that less support is available than is the case with commercially supported products. However, MySQL database software, for example, offers support contracts at standard market rates {19}. In addition, other companies such as Red Hat (who produce Linux operating system packages) also offer support contracts for individuals or organisations.

1.2.3 Introduction to Galform and e-Galform

Galform is a galaxy formation simulation code developed by members of the Institute of Computation Cosmology (ICC) of the University of Durham {20}{21}. It is a Fortran based program that simulates the physics behind galaxy formation in dark matter haloes. The output produced by Galform can then used in N-body simulations to produce mock galaxy catalogues.

A second program, samplegals, is used to read the raw output data of Galform (stored in a binary format) and produce a text-based ASCII table of data according to input parameters specified on the command line, which is simpler to plot and analyse. Chapter 2 of this dissertation focuses on the scientific background of Galform, and

contrasts the Galform approach with that of GalICS (an alternative semi-analytical galaxy formation simulation) and Hydra (an N-body simulation code).

e-Galform seeks to bridge the gap between research groups outside of the ICC and the important work of the Galform research group by building a set of facilities enabling scientists in other universities to use Galform data in a standardised VOTable format, via a web interface, for both theory and observation work without users of the data requiring a fine technical understanding of Galform or samplegals. The VOTable format, and its specific implementation in e-Galform, will be discussed in Chapter 4. e-Galform will also be able to act as a central communication point for the development of Galform activities and be a highly effective communication tool within the academic community. A discussion of e-Science principles with reference to e-Galform will be discussed in Chapter 6.

By using business-like marketing and communications methods for disseminating developments to those members of research groups likely to be interested in Galform, the focus on and prominence of Galform as a research project may increase, resulting in greater citations from external groups. Such a method for producing a 'virtuous circle' in the development of academic research projects highlights a further major advantage of the use of database applications on the internet in making research data and processing systems accessible to external groups.

Thus, in summary, the advantages of e-Galform as a database application are as follows:

- It allows other research groups to use Galform data for their own comparisons, research and data processing efforts without requiring finer technical knowledge of Galform or Samplegals. This opens the internal academic market for the use of Galform data and conclusions, in effect promoting the use of Galform work and perhaps concentrating focus on Galform in the academic community as the source of simulated data of hierarchical galaxy formation, in conjunction with the efforts of other research groups. It may thus also increase the citations by other academics of Galform work as it becomes more easily accessible to a wider audience.

- It acts as a 'hub' for information regarding the progression of both Galform and e-Galform, and a communications tool allowing a large number of potentially interested scientific departments to be informed of the existence of Galform.
- The database driving e-Galform is capable of being extended, edited and reconfigured without the need for a technical understanding of the underlying application code by means of web-based background administrative facilities. This means it becomes independent of its original author but still maintains the ability to adapt to advances in Galform or Samplegals over time, provided that such changes only merit additional rows being added to e-Galform's current database tables (processing and user interface changes will have to be made by a person proficient in PHP and MySQL).

This thesis will discuss the scientific background of Galform, and compare and contrast Galform with other galaxy formation simulations. I will then discuss the advantages and philosophy of the e-Galform system and compare e-Galform with other similar online database systems. Finally, I will discuss the various components of the e-Galform system and describe their role in fulfilling the technical and philosophical goals of the project, and suggest further ideas for advancement in both the general e-Science field (from the principles implemented in this project) and also for e-Galform specifically in the future.

2. An Overview of Galform And Other Galaxy Formation Models

2.1 Modelling galaxy formation

Theoretical astronomy has experienced great developments over the past 20 years due in part to the ever increasing speed of processors and the ability to network separate PCs together to produce effective supercomputers ('grid' computing) {56}, and also with advances in new numerical algorithms. At the same time, programming languages have advanced to allow more efficient and advanced memory allocation techniques; the languages have themselves been made more efficient through more sophisticated compilation facilities.

The detailed physics of galaxy formation is still poorly understood. Unfortunately, real observations of the process of galaxy formation, despite adequate optical technology are still difficult due to the fact that observers can only see snapshots in time. Galaxies take such long lengths of time to form from a primordial gas cloud that it is not possible to view the formation of a single galaxy by observation over human timescales. The best approach to determining the physics of galaxy formation is therefore to attempt to simulate the process in a computer, making educated judgements as to the physical mechanisms that govern various components, such as initial density fluctuations in primordial gas, gas dynamics and star formation.

This chapter will focus on the model of galaxy formation used at Durham, Galform. I will then briefly look at alternative ways in which other research groups are simulating galaxy formation, and contrast the two basic techniques that groups are using to study the underlying formation mechanisms, semi-analytic modelling and particle-mesh based gas dynamic simulations.

Both approaches have the same underlying basis: an initial mass of gas is perturbed, and the consequent evolution of the gas is followed. A description of the astrophysical processes that govern the cooling of gas in haloes, the formation of stars, supernovae (production of heavy elements and their distribution within the inter-stellar medium,

ISM), dust effects and galaxy mergers then lead to an overall description of galaxy formation.

2.2 Two Approaches: Numerical Simulation And Semi-Analytical Modelling

Attempts to model the formation of galaxies have so far been separated into two classes; numerical simulation and semi-analytical modelling, with efforts in some cases to combine elements of both approaches into a single model. In the first case, numerical simulation, equations governing gravitation and hydrodynamics are solved explicitly using a variety of techniques developed for this purpose over the past 20 years. Such an approach is outlined in {22}, {23}, {24}, {25}, {26}.

In the second case, semi-analytic modelling, the evolution of baryons is calculated using a simple analytic treatment, and the evolution of dark matter is determined either via Monte Carlo merger trees or via direct N-body simulation. Such an approach is outlined in {20} and {21} in the case of Galform (see Section 2.3), and {27} in the case of GalICS (a similar galaxy formation model to Galform – see Section 2.4).

Numerical simulation has an advantage in that all calculations are carried out in full generality; there is no need to make assumptions for simplification purposes. Such an approach can also readily yield graphical representations of the galaxy formation process. However, such a treatment requires computational performance that, even by today's standards, places impractical restrictions on resolution and timesteps. Such methods are still incapable of resolving the internal structure of an individual galaxy within a cosmological volume, and it is for this reason that other research groups have devised alternative methods for simulating galaxy formation. Also, the physics of star formation and feedback are currently not sufficiently well understood to allow them to be modelled without resorting to parametric recipes.

2.3 An Overview Of Galform: A Semi-Analytical Galaxy Formation Model

Galform has been developed for a number of years at University of Durham's Institute of Computational Cosmology, in collaboration between Shaun Cole, Cedric G. Lacey, Carlton M. Baugh, Carlos S. Frenk and Andrew Benson. A list of the key physical processes is first listed, followed by a brief description of each in turn. For full details of Galform, the reader is referred to {20}.

2.3.1 Brief Summary of Galform

• Galform assumes that galaxies form in dark matter haloes. The dark matter haloes themselves are then assumed to form via a merging history determined by halo merger trees, which can be generated via a Monte Carlo method or extracted from an N-body simulation.

• Within these haloes, a recipe for the formation of both discs (disc galaxies) and spheroids (elliptical galaxies and bulge components of spiral galaxies) is then applied. Discs are assumed to form by the cooling of gas initially in the halo, with spheroids assumed to form by galaxy mergers.

• Star formation within discs is assumed to occur at a rate directly proportional to the mass of cold gas. The size of a disc galaxy is determined via centrifugal equilibrium and the conservation of angular momentum. The size of a spheroid galaxy formed by mergers is determined via the virial theorem and energy conservation.

• The resulting galaxy luminosities and spectra are then determined from the predicted star formation history using stellar population synthesis techniques and an assumed IMF (initial mass function).

• These spectra may then be compared with observed spectra and the resultant analysis will determine whether the model is accurate, and identify particular areas in which the model may need improvement.

2.3.2 Forming Dark Matter Haloes Through Merger Trees

1. Dark matter haloes grow through the accretion of smaller haloes and mergers with similar sized objects. The process of halo growth is encapsulated in a "merger tree".

2. The tree records the number and mass of progenitors over a series of timesteps.
3. In modern Monte-Carlo algorithms, the progenitors may take any fraction of the descendent halo's mass {28}.

The resulting merger tree may then be analysed and the various model processes applied in order to produce a set of data and results from which may be extracted various galaxy properties at selected redshifts.

The Monte Carlo method has some advantages over merger trees extracted from *N*-body simulations, which due to practical resolution constraints caused by hardware are limited in dynamic range. The combination of N-body simulations with semi-analytical models techniques however provide a highly effective way to study galaxy clustering.

The Monte-Carlo algorithm uses an equation derived from extended Press-Schechter theory {28} to give the fraction of mass f_{12} in haloes of mass M_2 at time t_2 which at an earlier time t_1 was in haloes of mass range M_1 to $M_1 + dM_1$. Taking the limit of this equation as t_1 tends to t_2 gives an expression for the mass of M_2 which was in haloes of mass M_1 at time t_1 (with $t_1 < t_2$). This expression may then be used to obtain the mean number of objects of mass M_1 that a halo of mass M_2 fragments into when taking a step dt_1 back in time, yielding the average number of progenitors as a function of fragment mass M_1. This expression is then finally used to build a binary merger tree. In practical utilizations of the merger tree, due to the way in which Galform tracks galaxy formation, the merger tree is placed onto a regular grid of time-steps. The original tree is then used to find which haloes exist at each time-step, and to find information on which branches merge between time-steps. The justification for such an approximation (loss of information) is that mergers in practical terms do not occur in an instant, but rather over a period of time, and that the time-steps are much smaller in size than the dynamical time-scale on which haloes merge.

Although the merger tree is generated from the trunk of the tree (e.g. the present day, $Z = 0$) to the top (i.e. high redshifts) the galaxy formation process is modelled from the top down, corresponding to starting with the lowest mass haloes in the hierarchy. Haloes are assumed to have a lifetime defined by the time required to find a halo of mass 2 times the original halo mass whilst moving sequentially through time-steps. In investigating the statistical behavior of galaxies, Galform generates a number of merger trees from haloes specified at some redshift z_{halo}. Luminosity functions can be estimated from a weighted sum over the

model galaxies which takes into account the expected abundance of haloes of different mass.

2.3.3 Macroscopic Modelling Of Halo Properties

Galform models certain properties of dark matter haloes:

- Spin distribution of haloes - Dark matter haloes rotate at rates determined by the tidal torques operating during their formation. Galform uses the results of {29} to model spin distribution of haloes, based upon the results of *N*-body simulations.
- Halo density profile - Halo densities are modelled using the NFW model {30}.
- Halo rotation velocity - Galaxy formation occurs in those regions in which gas cools. In order to determine the angular momentum of such regions of the halo, the halo is split into concentric shells of constant rotation velocity. The average rotation velocity of the concentric shells in the halo is determined by the spin distribution.

2.3.4 Formation Of Disc And Spheroid Galaxies Within Haloes

The halo merger tree and macroscopic descriptions of the properties of the haloes then allows the description of galaxy formation within haloes. Galform can follow the formation of the disc and bulge component of galaxies:

- Disc formation - Galform assumes that discs form when gas cools within a halo. When the gas cools, it loses pressure. Tidal torques distribute angular momentum throughout the halo material. The gas conserves angular momentum as it cools, collapsing to form a rotationally supported disc. The mass of the forming disc is calculated by determining the radiative cooling rate of gas within the halo.
- Star formation in discs - The star formation process determines the conversion of cold gas into luminous stars, and also the physical state of surrounding gas as supernovae and young stars inject energy and metals into the interstellar medium (ISM). The star formation rate within the disc is assumed to be directly proportional to the mass of cold gas. Likewise, the rate of mass ejection from the disc due to energy input from young stars and supernovae is assumed to be a function of the instantaneous star formation rate.

- Spheroid formation – Spheroids (bulge components of spiral galaxies and elliptical galaxies) in Galform are assumed to form via galaxy mergers. Upon the merger of haloes, the most massive galaxy within a halo is assumed to form the halo's central galaxy. Other galaxies are then assumed to be satellite galaxies, orbiting the central galaxy, with their orbits decaying gradually due to dynamical friction. Satellite galaxies within the model therefore eventually merge with the central galaxy, if there is sufficient time before a new halo is deemed to have formed.

2.3.5 Determination Of Galaxy Sizes

- Disc galaxies - sizes are determined by centrifugal equilibrium and conservation of angular momentum.
- Spheroid galaxies - sizes are determined via the condition of virial equilibrium and the energy conservation to the merger progenitors and remnant.

2.3.6 Galaxy luminosities and spectra

Galform predicts the star formation history of galaxies. This can be used to build the spectral energy distribution of a galaxy using stellar population synthesis models. Deriving spectra from the model predictions is very important in assessing the overall validity of the model, and identifying particular areas in which the model is lacking in comparison with observed spectra. This is of great help in improving the model and providing systematic feedback on how the model responds to changes in the underlying basic physics.

- Stellar population synthesis - The model uses the work of {31} to compute the observable properties of the galaxies, combining the star formation history predicted by Galform with a choice for the form of the initial mass function (IMF). The models provide the spectral energy distribution (SED) for a single stellar population. In Galform, a composite stellar population is built up by a weighted sum of single age stellar populations of different ages. The IMF is then

assumed to be universal in time and space, and the SED is then convolved with observational filters to produce magnitudes in various pass bands.

- Yield and recycled fraction - The recycled fraction of gas, and the yield, of metals have an impact on the ISM. For massive stars, material is released back into the ISM through supernovae and through stellar winds. This gas then in turn is involved in further star formation, giving rise to stars of increasing metallicity over time.
- Extinction by dust - When star light passes through dust, a significant part of the spectrum is absorbed by the dust. Galform is able to model the effects of dust. Dust effects optical luminosities, the colour of galaxies and far-UV luminosities. Galform computes the dust extinction self consistently, calculating an optical depth from the size and metal content of the disk and bulge.
- Modelling of emission lines - Galform uses the models of {31} to compute the flux of ionising Lyman continuum photons, combining this with HII region models to calculate line luminosities and equivalent widths of emission lines in galaxies.

2.4 GalICS: A Hybrid Approach To Galaxy Formation

The GalICS semi-analytical model {32} of galaxy formation was developed at the Institute for Astrophysics in Paris and Oxford University by Steve Hatton, Julien E. G. Devriendt, Stephane Ninin, Francois R. Bouchet, Bruno Guiderdoni and Didier Vibert. GalICS is a hybrid approach to galaxy formation, combining the results of large *N*-body simulations with semi-analytical recipes to produce its output data, consisting of detailed dark matter merger trees and complete knowledge of a large range of galaxy properties. In doing so, it attempts to combine the benefits of both approaches. GalICS combines the accurately computed merger trees of *N*-body simulations with semi-analytical recipes to model the dissipative physics.

2.4.1 Forming Dark Matter Haloes Through Merger Trees

The GalICS N-body simulation uses a parallel tree code on the Cray T3E supercomputer. The simulation consists of a spatially flat, low density cold dark matter

model with a cosmological constant in a cube of side $100h^{-1}$ Mpc, with the power spectrum amplitude set according to the approximate abundance of present day rich clusters. The simulation uses $N_{part} = 256^3$ particles to trace the dark matter (additional simulations are run at lower resolutions of 128^3 and 64^3 which were used for resolution testing). GalICS then uses a percolation algorithm, the 'friends of friends' algorithm, to detect haloes.

The halo merger tree is constructed in the following way. For a halo h_1 at one time-step, a halo h_2 at a previous time-step is regarded as a progenitor if at least one particle is common to both haloes. Once the list of progenitors is compiled, the mass fraction of the halo is recorded for each contribution by progenitors. This information then gives the basic structure of the merger tree. This approach has the unique advantage in that, although it is resolution limited whereas Press-Schechter approaches can in principle attain infinite resolution (because the whole computer memory is dedicated to one tree at a time), full spatial information is recorded for the haloes and their progenitors rather than a statistical realisation of the history.

2.4.2 Baryon tree

GalICS only considers branches of the merger tree with 20 particles or more, on the grounds that such haloes are generally gravitationally bound. The main disadvantage of this method is that multiple mergers may be counted as a single merger between two output time-steps. For each halo, all progenitors are listed in decreasing order of contribution to the halo mass. The halo is then analysed to determine whether it is the most massive, and if so, it is assumed that all baryonic components of the 'father' halo are transferred to this 'son' halo. When a halo is first identified using the 'friends of friends' algorithm in the N-body simulation, it is assigned a mass of hot gas that is consistent with the primordial gas fraction of the Universe.

At each time-step, the hot intrahalo medium cools on to a disc assumed to be at the centre of each halo. The rate of cooling is determined by computing a characteristic cooling time, given as a function of radius. The cooling time depends upon factors such as the temperature as function of radius and density of gas as function of radius and its metallicity. The approximation is made that the halo is in hydrostatic equilibrium. It is

common for hierarchical models to overproduce the abundance of bright galaxies. This implies that larger haloes contain central galaxies that are generally too bright. This is could be due to too many mergers of galaxies, or too much gas cooling providing 'fuel' for star formation.

2.4.3 Galaxy Properties

GalICS models four processes to determine the content of the galaxy in terms of gas, stars and metals, namely cooling, star formation feedback, and metallicity.

- Modelling star formation - GalICS uses a dynamical time-step to determine what fraction of the cold gas within a halo is allowed to form stars. The dynamical timescale is determined from the circular velocity, and the radius of the disk. Starbursts are regarded to have the same geometry as bulges, and so their star formation rules are assumed to be the same.
- Feedback - For all stars formed, a certain fraction (determined by the IMF) will be stored within large, massive stars that expire quickly via supernovae. Some of the energy released is imparted to the ISM, 'blowing off' some gas. In exceptional circumstances and with enough kinetic energy this can lead to the gas being ejected from the halo. When this happens, less gas is available for star formation and hence this inhibits star formation rate, termed 'feedback'.
- Metallicity - Baryonic gas in haloes and galaxies is initially composed soley of hydrogen and helium, however the stellar population gradually processes these light elements into heavier elements. They are then ejected into the ISM via supernova.

2.4.4 Galaxy luminosities

GalICS uses stellar synthesis models to determine galaxy luminosities after the physical properties of the baryons have been used to determine the amount of light produced.

2.5 Comparison Of Theoretical Approach And Physical Outputs Of Galform And GalICS

2.5.1 Comparison Of Theoretical Approach

The two approaches are very similar, however differ most notably in terms of

- Generation of merger trees, and
- Feedback modelling.

GalICS uses the output of an *N*-body simulation in order to generate a halo merger tree. The Monte Carlo method also has the added advantage that it can be generated to an effective infinite resolution, whereas the *N*-body simulation output is by nature discrete. The Monte Carlo method considers one halo at a time, and so can attain a much higher resolution than N-body simulation by using all the available memory in this single halo. By contrast, a N-body simulation follows a huge number of haloes simultaneously, using the same total memory. Despite their differences, both have advantages and disadvantages, and both produce reasonably similar results {33}. It is therefore, arguably, not a significant factor in determining differences in output of the actual simulation.

Feedback modelling also differs. In the Galform approach, hot gas that has been ejected from the dark matter halo due to supernovae, and may be allowed to cool back into the halo. The same process is modelled within the GalICS framework, however the GalICS model allows hot gas to remain outside a halo, depending upon how the velocity imparted to the ejected gas compares with the escape velocity of the halo.

2.5.2 Comparison Of Physical Output – Samplegals ASCII Table (Galform) And GalICS MySQL Table

Both simulations offer a range of predicated properties of galaxies (and their host haloes). A full synopsis of the properties given by GalICS may be found at http://galics.cosmologie.fr/main_frames.php?dir=database. GalICS currently offers an

online facility, allowing a range of pre-generated tables to be queried using SQL queries. The data tables are described in the online documentation, and are driven by an underlying MySQL database (see Section 4.2). In the case of Galform, the samplegals program is used to generate an ASCII table containing galaxy properties from the binary format produced by the Galform simulation (see Section 1.2.3). Both have their own unique choices of output properties generated, field names and units attributed to these properties, with some similarities. Some output properties of the Galform and GalICS models are contextually similar; for example GalICS gives a range of properties regarding the bulge, disc and summed components of the galaxies generated, as does Galform. Examples include:

	Galform (samplegals ASCII table field and unit)	**GalICS (MySQL galaxy table and unit)**
Circular velocity of galaxy (bulge component)	vbulge (km/s)	bulge_speed (km/s)
Inclination of galaxy	inc (degrees)	inclination (cosine)
Stellar mass (disc component)	mstars_disk ($h^{-1}M_{sol}$)	disc_mstar (10^{11} M_{sol})
Star formation rate	mstardot ($h^{-1}M_{sol}$/Gyr)	tot_inst_sfr (M_{sol}/yr)

Figure 2.1 Potential property conversion table for GalICS to Galform formats.

Notice that whilst field names chosen for scientifically equivalent properties can differ between the models, chosen units can also differ (for example degrees are expressed for galaxy inclination in Galform, whilst the cosine of the angle is given in GalICS). Also present in the physical output of both models, among other properties are various luminosities predicted for a range of filter bands.

Results of e-Galform runs could potentially be used in the future to compare the results of the two models statistically in order to compare relative strengths and weaknesses of the models with respect to observation (see Section 2.5.4). Problems may arise however where definitions are different between models (for example the definition of what is meant by "disc" and "bulge" components).

The GalICS MySQL tables also contain spatial information (spatial coordinates of the haloes), however Galform does not produce spatial information unless combined with the results of N-body simulations (for example, the Millenium Simulation {4}). Spatial information is thus not directly comparable between the two models. An investigation of how e-Galform may be particularly useful in making comparisons of different cosmological models, and with observations, is given in Chapter 10 when describing how to improve e-Galform's XML output file.

2.6 An Alternative To Semi-Analytical Galaxy Formation: Smooth Particle Hydrodynamics

{34}describes the approach by Springel & Hernquist for the smooth particle hydrodynamics (SPH) approach to simulating galaxy formation. This work is the basis for the outline given below.

2.6.1 Key differences between SPH and semi-analytical approaches

The approach differs significantly from the semi-analytical approach to galaxy formation. Both gas and dark matter particles interact through gravity, however gas particles also interact through pressure forces and hydrodynamic shocks. The bulk properties of the gas are computed by averaging over a fixed number of neighbouring particles. In the Springel & Hernquist SPH model {34}, simulations of radiative cooling and heating processes lead to reionisation of the Universe at redshift z of approximately 6. A multi-phase model of star formation and supernovae feedback in the ISM is employed based upon an earlier model.

The resolution available with current hardware does not permit detailed enough results to resolve individual stars and therefore, similar to semi-analytical approaches, numerical simulations also use prescriptions in order to describe small scale or sub-grid processes.

The model assumes that thermal instabilities in a hot ambient medium lead to the formation of cold gas clouds, which due to resolution contraints must be described by a

coarse-grain approach, looking at the averages of fluid quantities within the medium. At this level, a particle normally consists of a mixture of cold gas and hot ambient gas, and a prescribed set of equations govern the behaviour of the mass and energy exchange between the two phases.

2.6.2 Star formation

Star particles are created from cold gas particles. The newly formed star particles are allocated half of the mass of the original gas particles, in a stochastic fashion. Massive stars (greater than 8 solar masses) explode on a short timescale via a type II supernova.

Galactic outflows or winds are implemented by hand in this model, following the observational evidence for winds obtained by {35}. Galactic winds are believed to explain the chemical enrichment of the low-density intergalactic medium, and also regulating the rate of star formation in order to reconcile the simulation predictions with the observed abundance of bright galaxies in the Universe. Simulations that do not take into account kinetic feedback can often lead to overcooling of gas, which is a particular problem for current SPH simulations.

2.7 Results From SPH Simulations: The Cosmic Data ArXiv

Attempts to compare semi-analytical simulation results (with spatial information) with SPH adaptive-mesh approaches might be made using the data available at the Cosmic Data ArXiv, which is available in a delimited ASCII format. The data is based upon the SPH adaptive-mesh approach include spatial information (see http://t8web.lanl.gov/people/heitmann/test3.html), based upon the results of the FLASH SPH simulation codes {1}. The Cosmic Data ArXiv ASCII table includes spatial and velocity information regarding the particles of the simulation within the simulation cube (simulations are often carried out with 256^3, 128^3 and 64^3 particles for integrity testing purposes, to test the sensitivity of the results to the mass resolution). This includes the x, y and z coordinates of particles, and particle velocity components. It also contains halo catalogs with spatial and velocity information, which could be directly compared with results of Galform and GalICS for equivalent cosmological conditions.

3. Use of MySQL, PHP, HTML and CSS in developing e-Galform

3.1 Choice Of Development Languages – Why PHP And MySQL?

e-Galform uses the server side programming language PHP {9} (Pre Hyper Text Protocol) in order to generate web pages and the MySQL database server {36} for relational database functionality. The basic web server on the University of Durham computers is Apache, which is run via the Linux operating system.

The role of the web server is to allow other computers with internet connections to connect via network sockets to the computer on which the web server is installed, and to deliver requested files to connected computers or clients of the server (including HTML files, which may be displayed on the client computer via web browser software).

3.1.1 Alternative Approaches To Developing e-Galform

e-Galform is a client-server application; that is, each client connects to the server, which delivers data to the client upon request of the client. Using a web browser and web server however is not the only framework by which to build a client-server application. Alternatives include:

- Developing native binary executables for a standalone server and client, for use on specific operating systems (e.g. using Microsoft Visual C++ 2005 for Windows {37}, or GCC (GNU Compiler Collection) for Linux {38}).
- Using an intermediate language such as Java bytecode (compiled from the Java programming language), and a Java Virtual Machine (JVM) application to execute the Java bytecode, using the same sourcecode for different operating systems but with a JVM specific to the platform on which execution is taking place.
- Using a web development language framework – for example Java (which may also be used to serve pages on a web server) or Visual C#.NET with Microsoft SQL Server, or PHP and MySQL. In this case, there is no need to develop

standalone server and client programs, as the web server and web browser become the server and client respectively.

This chapter will explain the reasoning behind the decision to use a web application (rather than standalone server and client application) developed using PHP and MySQL, as opposed to alternative architectures or web development frameworks. It will then describe the implementation of e-Galform using PHP, MySQL, CSS and HTML. Furthermore, Chapter 5 will elaborate on more general development issues not specific to PHP or MySQL, and describe e-Galform's system architecture.

3.1.2 Using Standalone Server And Client Through Native Compiled Code

Microsoft Visual Studio on Microsoft Windows is one of the most well-known development tools for creating native binary code for the Windows operating system. For Linux, development tools such as GCC {38} may be used to develop native binary codes for the Linux family of operating systems.

One of the advantages producing native binaries is efficiency, as the compiler is optimised for the particular operating system; there is no need for middleware interpretation of code as is the case for example with Java bytecode, or for a programming language executed via interpretation (for example JavaScript run via a web browser is interpreted line-by-line, rather than first compiled into binary executable form). This leads to highly efficient execution optimised to the particular operating system on which it is run.

One of the major disadvantages to this approach however, particularly with respect to e-Galform, is the fact that binaries compiled for a single operating system may only be executed on that platform. There is thus no direct cross-platform compatibility (although emulation may be possible if running on another operating system, emulation is also a form of middleware). In addition, some development frameworks (e.g. Win32 and C++ on Microsoft Windows) specifically tailored to individual operating systems can often not be ported or compiled on other operating systems, as their C functions are obtained from base linked libraries unique to particular operating systems (e.g. dynamic link libraries for Windows such as kernel32.dll, and shared system objects in Linux).

A further major disadvantage is that specific, standalone server and client applications, using a suitable network sockets application programming interface (API) would have to be developed specifically for e-Galform's requirements (such as the Windows Sockets API for Microsoft Win32 {40}, based upon the Berkely University of California implementation of Winsock). The server would have to handle network communications between itself and concurrently connected clients, and execute samplegals by request of each client before serving the ASCII and VOTable data to each client computer. The client computer itself would have to run a custom user interface client program that would allow the user to make settings to determine the data they wished to extract.

Such an approach would incur significantly greater development time, which may not offer significant advantage over a web application model, other than optimisation for a specific platform. However, being a native binary executable, it would only operate if both the server and client were executed on the same platform. In the case of a web application, the application is executed directly on the server, and the client served web pages, which removes the requirement to download binaries (or any other form of code other than web pages) in order to run the program.

Cross-platform incompatibility could be solved by developing concurrent source code versions for a range of operating systems, such as Linux and Windows, however this would significantly increase development time and would not present such advantage so as to justify the time required.

Because it is desirable for e-Galform to be executed on as many platforms as possible to reflect the diversity of operating systems present in academic departments (for example, the Durham University ICC uses SunOS, Microsoft Windows and Linux/UNIX systems), and because the user interface required would not need the level of complexity offering by directly calling an operating system's API (over and above the user interface that may be developed using HTML forms in a web browser) native compiled code is not necessary or well suited to e-Galform's requirements. It should be noted, however, that native binary executables offer the most highly optimised approach to its development in terms of speed of execution.

3.1.3 Using Java And Java Virtual Machine (JVM)

Applications developed in Java (such as the Astrogrid Workbench, see Chapter 4), and compiled into Java bytecode form using a suitable Java bytecode compiler, may be executed on a range of operating systems, provided that a suitable JVM may be found to execute the Java bytecode on a particular platform. In this way, the Java bytecode may be common to all platforms, but executed by a JVM unique to each platform.{41}

Java may also be run on a web server, whereby it may be used to serve web pages. "Applets" within these pages may then be interpreted and executed as client-side code, also through the use of a JVM. Java may also be used to create a standalone client-server based application through its network sockets application programming interface (API). In this case, a separate server and client application would be developed to handle client-server communications, in contrast to the use of a web server and web browser for web applications.

In the case of using Java to create a standalone client and server application, Java was judged less favourable for e-Galform as it would have required more development time, and would offer no particular advantage over a web browser/web server model to justify the extra development time.

3.1.4 Using C#.NET And Active Server Pages (ASP) As An Alternative Web Development Framework

Although C#.NET and ASP (Active Server Pages) is a framework capable of implementing e-Galform, it was decided that PHP and MySQL would be more suitable due to the fact that they are already readily available facilities on the University of Durham network (for example, any student may apply for a PHP and MySQL account through the University IT services). Because of this, the use of PHP and MySQL in itself does not require any further expense in order to build the e-Galform facility.

One advantage of ASP.NET over PHP is the implementation of "server objects" {42}, which are able to automate common user interface (UI) functionality in respect of database tables. For example, the ASP.NET "GridView" object is able to display

contents of a database table and allow data validation (checking that new values are within the range requirements set by the database) and editing using only a few lines of ASP.NET code, which would be particularly useful for example in e-Galform's administrative facilities (see Section 8.3). It is, however, possible to develop PHP class objects that would implement the same functionality.

Both ASP and Microsoft SQL Server would require additional financial commitment on behalf of the University. Because PHP and MySQL do not lack any particular functionality essential to the development of e-Galform in comparison to ASP.NET, it was decided to use PHP and MySQL over ASP.NET and MS SQL.

3.1.5 Use Of PHP And MySQL

As described in the introduction, PHP and MySQL are server-side components. In other words, they need only be present on the university's web server in order for e-Galform to be executed on the client machine. This allows both computers within the university boundaries and in other university departments globally to use the system without needing to install any additional software other than a web browser. The origin and number of users of the facility may be restricted by allowing only applicants confirmed by the administrators of e-Galform.

Linux, Apache, MySQL and PHP (known collectively as LAMP in Open Source terminology) are already available at Durham University and therefore if any further development of e-Galform is required (for example the creation of completely new modules), it is likely that multiple research groups in different universities will already have LAMP facilities in order to develop such modules. As LAMP technologies are Open Source, they may be upgraded or downloaded free of charge (however staff will need to be trained and adequately knowledgeable of these technologies to exploit them scientifically). 'Enterprise' level support contracts for PHP or MySQL are available at a cost to the institution using such facilities. The remainder of this chapter will describe the implementation of PHP and MySQL in developing e-Galform, particularly focussing on the use of MySQL and the accessing of a MySQL database using PHP.

3.2 Relational Database Structure

3.2.1 MySQL Database Schema Design

e-Galform uses multiple database tables to drive the underlying system, with interrelations between the tables. An SQL database table consists of a number of rows and columns of data. Each column is assigned a unique data type (for example "integer" or "char") and column label (a short identifier string). Each table also has an assigned *primary key*, a column of data that uniquely identifies the rows of the table (in some cases, a key may be formed by two or more columns). No data tables require more than one column for use as a record key in e-Galform. In the development of an application, the database tables are usually one of the first elements of the application to be designed (see Introduction), as they define both the content and structure of the data upon which the application processes and manipulates {43}. They may, however, change over the development lifetime of the application if superior methods of development or structuring are found to better suit the application's process. The following is a list of the tables that the e-Galform system uses to drive its database application (the primary key column is shown in grey; columns used to reference primary keys in other tables are shaded in light grey).

Parameters

ParameterID	ParameterIDText	ParameterDescription
<int>	<text>	<text>

Please note that parameters are called *properties* under Galform terminology. Includes a list of all properties predicted by Galform and derivable from Galform output via Samplegals. As Galform expands and develops to produce more data properties, this list may be expanded by the administration facility (see Section 8.3).

Filters

FilterID	FilterTitle
<int>	<text>

Includes a list of all filters built into the underlying Samplegals program. As Samplegals and Galform expand and develop to produce more filter outputs, this list may be expanded via the administration facility (see Section 8.3).

EmissionLines

EmissionLineID	EmissionTitle
<int>	<text>

Includes emission from dust at given bands as built into the underlying Samplegals program. As Samplegals expands and develops to produce more dust emission lines, this list may be expanded via the administration facility (Section 8.3).

Models

ModelID	ModelTitle	ModelDir
<int>	<text>	<text>

A list of all physical models from which the output data is generated. At time of writing this includes the models given by {20}{21}. This is expandable via the administration facility (Section 8.3).

News

NewsID	NewsHeadline	NewsStory	NewsDate
<int>	<text>	<text>	<date>

This contains news stories for the front page of e-Galform. This table includes the news headline, news story date and the article text. This is called by the front page PHP script each time the front page is loaded, and the news story headline and date displayed accordingly. A clickable web link is also generated for each story, and when the link is clicked on, a story page is generated using PHP.

ParameterStore

ParameterStoreID	ParameterStoreSessID	ParameterID
<int>	<text>	<int>

The affixed 'store' tables record all information currently saved by users of e-Galform, with their unique session ID (see Section 3.2.6) that allows e-Galform to differentiate

their saved data from other concurrent users. This includes only an ID reference number to the corresponding property stored in the Parameters table.

FilterStore

FilterStoreID	FilterStoreSessID	FilterID
<int>	<text>	<int>

Includes a list of all filters currently in use by e-Galform users, including a unique session ID to differentiate saved filters for simultaneous users. This includes only an ID reference number to the corresponding filter stored in the Filters table.

EmissionLineStore

EmissionLineStoreID	EmissionLineSessID	EmissionLineID
<int>	<text>	<int>

Includes a list of all dust emission lines currently in use by e-Galform users, including a unique session ID to differentiate saved filters for simultaneous users. This includes only an ID reference number to the corresponding dust emission line stored in the EmissionLines table.

MailingList

MailingListID	MailingListName	MailingListEmail	MailingListInstitution
<int>	<text>	<text>	<text>

This stores a list of all target recipients of e-Galform news updates. This includes the target name or mailing list, e-mail address and destination institution. The mailing list includes those not necessarily registered as a user with e-Galform.

Applications

ApplicationID	Application Name	Application Email	Application Username	Application Password	Application Institution
<int>	<text>	<text>	<text>	<text>	<text>

Includes a list of all those applying for a username and password for e-Galform. Once confirmed, they are passed to the RegUsers table. Includes name, e-mail address, username and password.

AdminUsers

AdminUserID	AdminUserName	AdminUserPassword	AdminUserEmail
<int>	<text>	<text>	<text>

This stores a list of staff members who are administrative users of e-Galform. These users may edit the list of administrative users, the mailing list and list of properties used by e-Galform, as well as the list of filters, adding models as Galform papers are updated.

RegUsers

RegUserID	RegUserName	RegUserEmail	RegUserUsername	RegUser Password	RegUser Institution
<int>	<text>	<text>	<text>	<text>	<text>

This stores a list of users registered on the e-Galform system. It includes the user's name, e-mail address, username and password in encrypted form. This table is referred to when a user attempts to start using e-Galform from the front page.

Logs

LogID	**RegUserID**	**LogTime**	**LogSessionID**
<int>	<int>	<timestamp>	<text>

The Logs table stores usage data regarding e-Galform runs by registered users (indicated by the user's RegUserID from the RegUsers table). When a registered user makes a run, the time started is logged on in the Logs table along with the user's unique session ID (LogSessionID) for the particular user session.

3.2.2 Operating On A MySQL Database Using PHP

e-Galform uses SQL (Structured Query Language) to operate upon the MySQL database via PHP. A basic general SELECT query is of the form {43}:

```
SELECT <ColumnName> FROM <TableName>
```

For example, the following simple SQL query retrieves the usernames of all e-Galform registered users from the RegUsers table:

```
SELECT RegUserName FROM RegUsers
```

This query would return all rows from column "RegUserName" from table of name "RegUsers". More complex queries may be formed from this basic general syntax, for example to order the returned records in ascending or descending order with respect to some column, or to return multiple columns, or to only return particular rows with particular properties. Because a SELECT query returns records, the information it returns must be stored in memory, or on disc, so that it may be used as a basis for further processing. In PHP, there is native support for MySQL databases.

3.2.3 Authenticating And Connecting To The MySQL Database Server

e-Galform first connects to the MySQL database using the e-Galform function *do_dbconnect()*:

```
function do_dbconnect($dbcnx)
{
      $dbcnx = @mysql_connect("ganymede.phyast.dur.ac.uk", "username",
      "password");

      if(!$dbcnx)
      {
            do_html_header('Server Unavailable', FALSE, 0);
            echo 'Unable to locate the website database at this time -
'.mysql_error();
            do_html_footer();
            exit();
      }

      // Locate the galform database (change database name according
to changes in this name)
      if(!@mysql_select_db("galform") )
      {
            do_html_header('Database Unavailable', FALSE, 0);
            echo 'Unable to locate the database at this time. Please
try again later - '.mysql_error();
            do_html_footer();
            exit();
      }

    return true;
}
```

Figure 3.1 – e-Galform PHP source code for connecting to MySQL database.

The native PHP function *mysql_connect* is first called with the database network address, username and password. These are set when first creating the database, which was initially designed using the Open Source web software *phpMyAdmin*. In all cases in the e-Galform source code, any errors generated by the database are handled so that processing is halted to ensure that a malfunction does not affect further processing causing unpredictable results. This is particularly important in the case of e-Galform, where large data volumes may be generated and processed. The resultant error message includes the MySQL error text returned by the MySQL server so that it is clear to the developer whilst debugging the error that has occurred. In the case where an error occurs and the user is not a developer, the error text serves as a useful verbal indication to network administrators of the possible reasons why a problem may have occurred.

3.2.4 Selecting The Database On The Database Server

A database server may often host more than one database. Once connected to the MySQL server, the native PHP function *mysql_select_db()* is then used to select the MySQL database itself, hence allowing PHP to perform SQL queries on the database (the database name is "galform"). The *do_dbconnect()* function is called from all e-Galform PHP scripts that require access to the e-Galform MySQL database, and is hence is included via the PHP *include* directive.

3.2.5 Performing MySQL Queries On The Database Using PHP

Once the database is selected, rows from specific tables may then be retrieved using SELECT queries. The SELECT query returns records, and this information must be stored in some way by PHP so that it may then be processed.

PHP performs SQL queries on the selected database using the native function *mysql_query()*, whose single argument is a string containing the correct SQL syntax. When the query returns records, PHP returns a handle to the query result, which may then be used to 'fetch' rows from the returned record set. The following source code from e-Galform's parameter selection page (Figure 4.3) retrieves the list of supported galaxy properties from the Parameters table and populates the drop-down list box with the property description:

```
$result = mysql_query( 'SELECT * FROM Parameters' );

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo '<OPTION
NAME="'.$row['ParameterID'].'">'.$row['ParameterDescription'].' [<DIV
STYLE="font-color:#ff0000;font-
style:bold">'.$row['ParameterIDText'].'</DIV>]</OPTION>';
}
```

Figure 3.2 – PHP source code populating galaxy properties drop-down list box.

PHP first requests all columns from the Parameters table via the SQL query

```
SELECT * FROM Parameters
```

Where * indicates that all columns of the table are to be returned. PHP then returns a MySQL query result handle, *$result*, which uniquely identifies the SQL query. If an error occurs whilst processing the query, *$result* returns null and is hence detected by the error handling code. Again, error handling is implemented for every SQL query performed by e-Galform's PHP scripts, as erroneous access to the MySQL database may cause unpredictable behaviour.

Once a valid handle is received indicating a successful query, it is passed as a single argument to the native PHP function *mysql_fetch_array()*. Each time the function is called, it returns the current row of the query record set and places the cursor on the next row. The PHP array *$row* includes all the data in the retrieved row as a set of elements, with one element for each column in the row. Once the end of the record set is reached, *$row* returns NULL and hence the condition of the *while()* loop is falsified.

PHP is particularly advantageous for retrieving rows from an SQL database as it is 'weak' typed; that is, a single array may contain multiple data types (e.g. integer, string, double) with no requirement to create separate arrays for columns of the same data type. An element in a PHP array may also be accessed via a label rather than a

numerical index. Therefore, in the case above, the column data in the 'ParameterDescription' column for the particular row returned may be accessed via *$row['ParameterDescription']*. This returns a string. At the same time, the data element in the *$row* array containing the 'ParameterID' column data element for this particular row may be accessed via *$row['ParameterID']*, returning an integer. This greatly simplifies management of data types in comparison to 'strong' typed languages such as C++, for which variable data types must be explicitly defined and converted to other data types if required (in Visual C++ and ODBC, Open Database Connectivity, class objects are created to represent individual data tables; these classes contain member variables specific to each column, according to its data type {44}).

3.2.6 Session Management Using PHP: Handling Multiple Concurrent Users Of e-Galform

As the web server itself cannot identify clients uniquely (a computer's IP address for example may be dynamic or may reflect more than one computer on the same network for a shared internet connection), the user's session ID is stored in the form of a small text file directly on the client's hard disc, usually within the system's registered temporary directory (on Windows for example, this temporary directory is found in the Windows registry). This text file is called a *cookie* {45}, and is accessed by the web server via the client's web browser. Hard disc access by the browser on the client's computer is only permitted only to the cookie for security reasons, so that a web site cannot make any other changes to the client's computer without his or her voluntary permission.

Changing a computer's configuration or hard disc contents is possible via websites, through client-side scripting (for example *ActiveX* controls {46}), however many modern web browsers or firewall software packages now contain a number of security measures that either prevent such scripts from executing, or require voluntary permission from the client user before such scripts are executed. e-Galform does not use client-side scripting.

The inclusion of a session ID in a number of data tables (specifically tables that record parameters requested by e-Galform users to state the information they wish to retrieve

from the site's base Galform data), including the ParameterStore, FilterStore and EmissionLineStore tables allows e-Galform to distinguish between multiple users of the site whom may be using the facility at the same time. Hence, a single data table may be used to record the parameters requested by concurrent users.

When the user first loads the e-Galform homepage on their web browser, e-Galform calls the PHP *session_start()* function. This function is found at the top of all scripts that use cookies, so that even if the user does not begin their visit to the site on the front page, PHP will still write a cookie to the client computer when required. The PHP *session_start()* function also needs to be called on all scripts that use cookies even if the session is not new.

Upon starting a new PHP session, PHP generates a pseudo-random, 32 character length session ID which (in all probability) should be unique to all currently active sessions. A PHP session remains until it expires, the time lapse of which is determined by the PHP configuration {47}. In e-Galform, it is recommended that the cookie expiry time should be set equal to the length of time determined to allow for a session's files and database entries to remain on the server before being automatically removed (see Section 4.3.5). Preserving the cookie over the full term of use, from first visiting the site to downloading generated data files, is necessary so that files generated uniquely for a particular user may be identified by the server once e-Galform has finished generating a user's requested data (in order to provide links to the correct files upon completion of processing).

The cookie text file then contains *session variables*, which may be written or removed from the cookie via the PHP *$_SESSION* array. An example from the e-Galform source code below shows the identification of the user when reading the *ParameterStore* table for any galaxy properties:

```
// Check tables in order and output, matching the current session id
$result = mysql_query('SELECT * FROM ParameterStore WHERE
ParameterStoreSessID=\''.session_id().'\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
```

```
}

while( $row = mysql_fetch_array( $result ) )
{
  $result2 = mysql_query('SELECT * FROM Parameters WHERE
ParameterID='.$row['ParameterID']);

  if(!$result2)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

  $row2 = mysql_fetch_array( $result2 );
  echo 'Output Property: <b>'.$row2['ParameterIDText'].'</b>
('.$row2['ParameterDescription'].')<BR>';
}
```

Figure 3.3 – PHP source code cross-referencing the user's selected output properties with the Parameters table in order to obtain the output property's label and description

The MySQL query called via the PHP function *mysql_query()* returns all rows in the ParameterStore table that match the session ID contained in the cookie stored on the client's computer (the function *session_id()* returns the session ID for the client). It then iterates over the Parameters table and finds the parameter label and description for the parameter chosen by the user. These are then displayed at the top of the advanced settings interface screen (see Figure 4.2).

3.2.7 Use Of Cascading Style Sheets To Maintain Consistency Of Appearance

CSS (Cascading Style Sheets) were first introduced into web development in 1994 in response to a need to increase the control of the web developer over the appearance of HTML web pages. However CSS only came into full support in mainstream web browsers such as Microsoft Internet Explorer in 1999{48}. HTML pages initially contained very little layout property control apart from table cells, and so the finer layout was decided by the web browser dependent upon its interpretation of HTML tables. This occasionally led to HTML pages appearing recognisably different between browsers.

e-Galform uses CSS classes {49} to maintain consistency between pages, but also between different browsers (for example Mozilla FireFox, Internet Explorer) running on different operating systems. CSS allows the developer to 'finetune' the layout of an

HTML table and remove default settings a browser applies to table cells (for example, a grey border several pixels in width appears by default around the cells of an HTML table in Internet Explorer; this may not always be desirable). By using CSS, pixel widths between cells of a table, the background colour of a cell, and the font used in a cell, and text alignment within a cell can be enclosed within a CSS class and applied to an HTML element by including the *class* property within its opening tag. For example, the following HTML applies the CSS style class "myclass" to an HTML <td> tag:

```
<td class="myclass">Contents of HTML element.</td>
```

3.2.8 Maintaining Consistency Of Typefaces

e-Galform's default typeface is Tahoma, however Tahoma may not be available on all platforms and so the secondary choice of font is Arial, which is then used in its place if available. Some web browsers attempt to automatically choose the closest equivalent according to available system fonts, however deliberately stating a priority order gives the greatest guarantee that consistency of appearance is maintained between browsers and platforms, because not all browsers may support automatic font substituting. Because the same CSS script is included at the beginning of each page, consistency of appearance is maintained throughout the e-Galform site and, to the maximum extent possible, between different web browsers or operating systems (which may be supplied with different system fonts) that adequately support CSS.

4. Principles Of Development In e-Galform

4.1 Authenticating The User

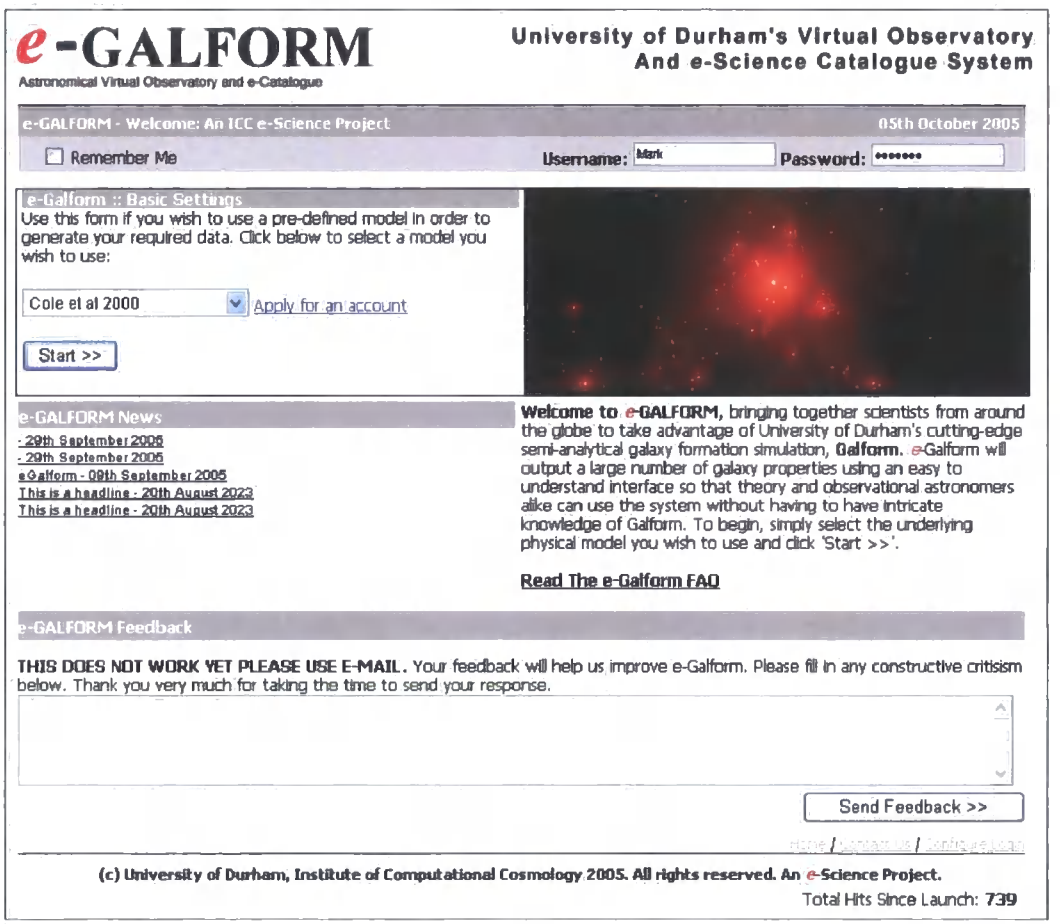

Figure 4.1 – Front page welcome and authentication interface

The front page of the interface (Figure 4.1) presents the user with a selection of models based upon published Galform papers (read from the Models data table, see Section 3.2.1). Each paper has an associated pre-generated Galform binary dataset, stored in a directory accessible by the web server. The original Galform simulation is not run from first principles via e-Galform, as this is very time consuming, on occasion taking a number of days (dependent upon hardware). It is therefore not suitable for a web server to handle, particularly for concurrent requests. Galform is more complicated to run,

than, for example an N-body simulation code, as it models many more physical processes as discussed in Section 2.

Instead of running the Galform simulation directly for each request, e-Galform uses a pre-generated Galform binary data set as a basis for providing galaxy data, and then uses a binary executable called 'samplegals' to extract specific properties from the pre-generated data set in a proprietary ASCII format. Samplegals is a Fortran based program that extracts galaxy properties from the binary results of the Galform simulation according to set of commandline parameters. The size of the underlying Galform binary data set is not fixed by e-Galform, and may typically account for several megabytes, to several gigabytes in size. By using samplegals to extract galaxies from the underlying dataset, retrieving data requested by the user may be carried out in practical timescales whilst limiting the data output only to those properties the user has requested. A discussion of timescales involved in using e-Galform will be discussed in section 4.3.5).

Also on the front page is a link with the text "Apply for an account", which invites the user to leave their name, e-mail address, and institution, in order to gain a place on the site's user application queue. Asking users to register – and for manual confirmation of their user account by a site administrator – provides enhanced security for the application (disallowing free use of the facility for anyone who does not have an account, helping to prevent "denial of service attacks" via repeated requests to the server). It also allows a limit on the number of users who may use the system over time. At the bottom right of the front page is a link to the administrative area of the site, which will be explained later in this chapter, allowing site managers to confirm or deny an applicant the right to use the facility. All passwords are stored in encrypted format using PHP's *crypt()* function for added security, which encrypts a string according to a "seed" word, and are not permitted to be changed via administrative facilities.

4.1.1 Authentication Of The User

Upon entering a correct username and password in the top right-hand corner of the front page (see Figure 4.1), clicking on the "Start >>" button beneath the model selector then sends the username and password strings to the following page via POST variables (a

method of transmitting information from one page to another using a web server). These variables, sent via stdin/stdout pipes on the server locally, cannot be read by the client directly, and so the user's password entered via the web browser are not visible on, for example, the URL pointing to the next page as in the case of using GET variables (whereby variables are transmitted from one page to another by including them as parameters in the destination page web address). {50}

The user's original choice of password is never entered directly into the site database, and so the password entered on the front page is immediately sent through the PHP *crypt()* function and cross-referenced with the password on the database. For the same "seed" word and same string to be encrypted, the same string results from the call to the *crypt()* function. Upon valid authentication, the user is presented with the scientific properties selection page (see Figure 4.3). The interface is based upon the properties available from the basic Galform model data, which are stated in the source code of samplegals. However, the author had to obtain from Galform research staff correctly worded definitions for each of the properties (stored in the Parameters table; see Section 3.2.1) listed on the scientific properties selection page. The properties described in the Parameters data table may be extended or modified by use of the administrative facilities (see Section 8.3), as well as new properties added or removed.

4.2 Executing Samplegals In e-Galform

```
for( $x=0; $x<count($redshifts);$x++)
{
  if($_POST[$redshifts[$x]] != '')
  {
      exec($_POST[$redshifts[$x]],$output);
  }
}
```

Figure 4.2 – PHP source code executing samplegals for each selected redshift. The commandline string is stored in POST variables, passed from the previous web page.

The property selection user interface (Figure 4.3) is derived from the commandline parameters of the binary executable samplegals. The interface both creates a more intuitive, visual method of extracting data from Galform simulation data and also gives a more verbal description of the various properties and processing that may be applied

to the data to be extracted. This removes the need for the user to understand samplegals, and also informs the user of the verbal meanings of the properties they are selecting for extraction (which is, at time of writing, only available via the samplegals source code). Note also that the web interface removes the need for either the Galform data, or samplegals, to be present locally on the user's computer.

The verbal descriptions on the interface are written in such a way as to target a theoretical or observational astronomy audience, rather than a member of the general public with little or no astronomical background (the facility is unlikely to be of practical use to any audience other than those from a qualified astronomical background). The facility can then process the ASCII table and produce a schema compliant XML document based upon the VOTable schema, which may then be used in other applications (see Chapter 7). Figure 4.2 shows how PHP has been used to execute samplegals on the server computer by storing the commandline within an array, and circulating through the array for all redshifts which have been selected by the user on the property selection interface (Figure 4.3; redshift selections are requested in section 6 of the interface).

4.2.1 Interpreting The User Interface State And Forming Commandline Parameters For Samplegals

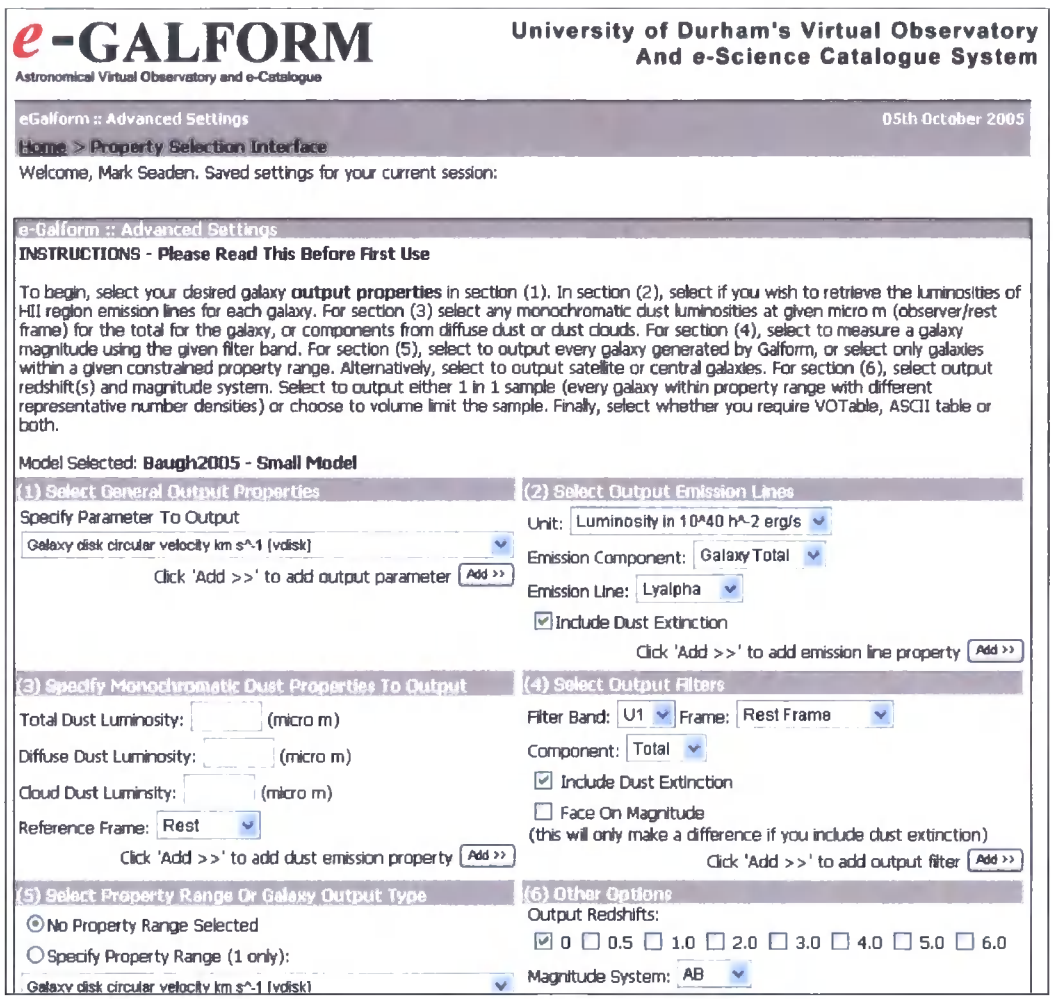

Figure 4.3 – e-Galform Property Selection Interface

The e-Galform property selection interface is divided into six sections: output galaxy properties, emission lines, dust properties, filter bands, an option to limit output to a certain property range, and redshifts at which to output the Galform simulation data.

Some of these properties are mutually exclusive: for example, a user may select a property range *or* galaxy type to output, or neither. In other cases, the user may select more than one property per section. For example, the user may select multiple galaxy properties (interface section 1), and multiple emission lines (interface section 2).

So that multiple properties may be selected, the user adds individual properties to their output dataset by clicking on the drop-down list box and selecting the "Add >>" button. For each property, the property type, and the user's unique PHP session ID is stored in the "ParameterStore" data table (see Section 3.2.1). The same principles are followed for section 2 (emission lines), 3 (monochromatic dust properties), and 4 (output filters). This allows e-Galform to uniquely identify the particular user's settings when running samplegals for the particular user's run.

4.2.2 Initial Problems Encountered Involving Recording User's Data Selection

A problem was initially encountered whilst developing the interface, involving both recording data regarding the user's selection and the transfer of this record to the next page when they are ready to generate their Galform dataset.

Firstly, it was desirable to keep client side scripting – for example the use of JavaScript – to a minimum, as JavaScript is dependent upon a browser fully supporting its functions on the client. As e-Galform may be exposed to a number of different platforms and browsers (and browser versions), by keeping as much code as possible server-side, compatibility is guaranteed as only HTML pages are required for support by a web browser to support purely server-side coding.

However, execution at the server side requires the reloading of the web page in order to execute a script. When the user clicks the 'Add >>' button in sections 1-4 of the web form, a new page must be loaded which executes a new PHP script adding their selection to the correct storage table, with their settings and a unique session ID to identify their selection from any other concurrent users. This was initially implemented by loading a new page, and asking the client to click a link on the new page to return to the property selection interface (Figure 4.3). This inconvenience, particularly for situations in which the user wishes to request a wide variety of parameter selections, was overcome by using the HTML server-side directive <meta http-equiv="...."/>:

```
<meta http-equiv="refresh"
content="0;url=/galform/advanced.php">
```

When executed at the end of each PHP script that records the user's selections, this directive instantly returns the user to the (refreshed) property selection interface page (Figure 4.3). The new property selection is then shown in a list, just beneath the page's header. If the user wishes to reset their selection, a "reset" button is available at the bottom of the page. This resets the user interface, and also resets any session data associated with the user's property selections.

Once the user has selected the redshifts at which they wish to extract galaxy properties from the data, they may then press the "Generate >>" button at the bottom of the screen, which will then execute Samplegals with the correct commandline parameters dependent on the selections stored in the relevant MySQL tables by the property selection interface. The code to produce the correct execution commandline is found in the *start.php* script found in Appendix B. The sequence of scripts executed to carry out this functionality is also described in Figure 4.6 and associated script descriptions in Section 4.6.

4.2.3 Security Concerns Regarding Use of PHP *Shell()*

In order to use PHP's *shell()* command, PHP Safe Mode must be turned 'off' (by default, 'safe mode' is turned 'on'). This is because executing binary code on a server via a web application may leave a server vulnerable to outside attack, if authentication procedures are not in place to prevent unsolicited use (for example, a malicious user may choose to repeatedly run the script that executes a binary application on the server to overload its processing or storage capacity).

e-Galform has authentication processes in place to prevent unsolicited use, however this does present security risks from experienced hackers. In terms of security present currently on the e-Galform system, even if a potential hacker knows the name of the script to execute via their web browser in order to run PHP's *shell()* command, they will be denied from doing so due to a cookie check carried out by the web browser when loading any page available only to users that have logged in. A valid session cookie must be present on the client computer to run any scripts that require authentication to access on the server, and this cookie may be obtained only by successfully logging on via the front page authentication process. It is possible for a person to copy a valid

cookie from a computer that has successfully logged into e-Galform from another computer, however this would require physical access to the computer storing the cookie.

Despite this safeguard, it is always possible for hackers with advanced skills to penetrate security and this must be left as a risk factor, given that the site is unlikely to attract a high, diverse range of traffic. Passwords (required to gain access to e-Galform's data generating facilities) are stored in the e-Galform MySQL database. Even if a hacker manages to obtain access to the data tables containing registered usernames and passwords, for which they would need the database password, the hacker will not be able to read the password column as they have been encrypted without means of decryption (see Section 4.1.1). This prevents a user that has hacked into the e-Galform database from reading any accepted username and passwords.

The 'seed' word initially used to generate the password encryption is also readable on the PHP scripts; however because the script itself only checks to see if the password entered by the user is the same when placed through the same encryption process (running the *crypt()* function on the password string with the same seed word), the password itself is never decrypted, and so is only ever known to the original registered user. PHP does not offer any functionality to decrypt the password, and therefore there is no means for a hacker to decrypt an encrypted password string, even if they are aware of the 'seed' word used for encryption.

The author has used methods of implementing authentication given in an established PHP and MySQL text book {51}. However, this is unlikely to be sufficient to fend off the most advanced hackers; all websites are ultimately vulnerable to outside intrusion, and ultimately standard network administration practices should be used to monitor potential hackers (such as regularly reviewing server log files).

4.3 Forming An XML VOTable From Samplegals ASCII Format

4.3.1 Galform ASCII Format

The samplegals ASCII table is formatted with a header, followed by rows of numerical data. A samplegals ASCII table is produced for each execution by e-Galform on the server, for each redshift requested by the user. The ASCII file header contains a number of properties of the data extraction, for example the type of magnitude system in which numerical values are expressed, and the redshift at which the data is taken. Each line in the header is prefixed with a '#' character. For maximum extensibility, e-Galform determines the length of the header from the number of lines prefixed with this character indicator, rather than assuming a fixed number of lines forming the header indefinitely into the future. Provided that a '#' character remains the distinguishing indicator between lines in the header and the numerical columns of data, e-Galform will be capable of finding where the columns of numerical data begins and the header ends. Also of note is that the labels indicating the data contained in the columns form the last line of the header, with the column label matching the corresponding numerical column.

```
# Catalogue Volume=  0.000E+00 (Mpc/h)^3
# Magnitudes are in the AB system
# dustfile= dust/dust_MW_hz1.0.dat    emdustfile= 0
rfacburst=  1.0000E+00 fcloud=  2.5000E-01
# redshift z= 0.000
#  weight              vdisk              metals_bulge
 1.57729E-03   8.06469E+01   0.00000E+00
 1.57729E-03   1.26099E+02   0.00000E+00
 1.57729E-03   8.09951E+01   0.00000E+00
 1.57729E-03   6.96513E+01   0.00000E+00
 1.57729E-03   6.98694E+01   0.00000E+00
 1.57729E-03   6.91017E+01   0.00000E+00
```

Figure 4.4 – Example samplegals ASCII table, including the properties *weight*, *vdisk* and *metals_bulge* for 6 galaxies (rows of data).

Provided that the format described as produced by samplegals is not changed, the VOTable will generate for varying numbers of columns and properties via e-Galform.

4.3.2 e-Galform VOTable Format

VOTable is an XML (eXtensible Meta Language) data format. XML consists of data *tags* enclosed in directional arrows < >, which themselves enclose may enclose data attributes {52}. An example of an XML data tag and internal data might be:

```
<MYDATATAG>12.5</MYDATATAG>
```

In the above example, the tag MYDATATAG is begun by the opening part <MYDATATAG>, with enclosed data 12.5, and the tag is terminated by </MYDATATAG>. Because both tags, and their internal data, have no restrictions on choice of definition, XML has become an increasingly popular choice in defining cross platform-compatible data formats, as it is one of the most verbose methods of defining data structures available on the internet.

Because it is highly verbose in nature and sometimes uses lengthy tags, it also can produce very large data files. A typical VOTable format table might be double the size of the equivalent ASCII galform table for the same output data. This is because VOTable also stores verbal definitions of all parameters included in the data file in addition to property labels and numerical rows, and the data values themselves must be stored in individually opened and closed tags. This contrasts with the ASCII format table, in which the data is simply listed in rows of numbers, with a fixed column width for columns in the table.

The advantage of storing verbal definitions with the data is that it allows the receiver of the data to cross reference data labels with their strict verbal definition. In the standard Galform ASCII data table format, only the data labels are stored. Because of this, those unfamiliar with Galform would be unable to determine concise verbal definitions of the parameters.

In addition, VOTable, although larger in size, can also be used in other programs such as TopCat {53}, which is an academic standard graph generation package. It is expected that as the format becomes more popular over time, more and more programs will begin to support the format and thus e-Galform's data output will be able to be used in a wider range of contexts. The VOTable format has been developed by a number of contributors, particularly by Francois Ochsenbein of the Astronomical Observatory of Strasbourg, France, and Roy Williams of the California Institute of Technology {54}.

The general structure of the e-Galform VOTable compatible XML format comprises an XML header structure that indicates the file is a VOTable compatible data file, followed by a list of <FIELD... > tags describing verbally the meanings of each of the properties included in the table. The file then includes a full list of all values for each of the described properties enclosed in table row <TR... > and table data <TD... > tags. An example of a simple file may be as follows:

```
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
      <VOTABLE version="1.0">
            <RESOURCE ID="GalformOutput">
                  <TABLE NAME="GalaxyProperties">
                        <LINK content-role="doc" title="e-Galform
                  documentation"
                  href="http://www.dur.ac.uk/egalform"></LINK>
                        <FIELD ID="vdisk" datatype="float">
                              <DESCRIPTION>Galaxy disc circular
                        velocity km s^-1</DESCRIPTION>
                        </FIELD>
                        <DATA>
                              <TABLEDATA>
                              <TR><TD>12.5</TD></TR>
                        </TABLEDATA>
                  </DATA>
            </TABLE>
      </RESOURCE>
</VOTABLE>
```

Figure 4.5 – Example VOTable XML file

Given the highly generic nature of VOTable, it is possible for the above format to perhaps be expressed in different ways. The header contains information regarding both the VOTable document definition, the VOTable version number, a description of the nature of the output, and an example link to relevant e-Galform documentation. The field description contains the property label, its data type, and a verbal description of the property. e-Galform may be improved however by including further information with the <FIELD...></FIELD> tags (see Chapter 11).

Finally, the table data contains rows and cells that contain all of the numerical data related to the described fields. In this way, the program reading the VOTable file can acquire a large amount of specific information regarding its contents, to the extent that a person initially unfamiliar with Galform and samplegals labels may be able to understand the contents without prior knowledge of the samplegals ASCII table or Galform (the user would, however, require adequate knowledge of astronomy).

In addition, programs such as TopCat may use the verbal definitions of parameter labels to develop features such as data legends and other labels to help the viewer understand the nature of the information. This gives a distinct advantage over the samplegals ASCII data format. The XML VOTable format also contains a header, in the form of XML tags, including a web address to the XML schema document that concisely defines the scope and design. It also includes labels for the output table, and links to relevant documentation available on the e-Galform website front page, which introduces the facility and offers the ability to register (Figure 4.1).

The header is followed by a sequential list of field types, which are derived directly from the ParameterStore data table (see Section 3.2.1) for the session ID of the current user. The fields included in the data include the field's parameter ID (e.g. *vdisk* below), followed by its data type (e.g. int, float), followed by a verbal description of the parameter as given in the Parameter table (Section 3.2.1). The fields are listed in column order for each row of numerical data in the table. The footer of each XML document is formed as terminating tags for those given in the header, with the <VOTABLE.../> tags forming the enveloping opening and closing tags of the file.

4.3.3 Conversion Of Galform ASCII Format To VOTable XML Format

For each row of numerical data in the output samplegals file, e-Galform splits the columns of fixed width into an array by using the PHP *explode()* function, using a column delimiter of a double space. Provided that the delimiter remains the same, the fixed width of the numerical data columns may change in the future without resulting in a loss of functionality of e-Galform.

The process is repeated for each requested redshift, as indicated by the user on the property selection interface (Figure 4.3). Once processing is complete, the user is presented with a list of links to the files they have requested. If they have requested to download the file in Samplegals ASCII format, they will also be given links to the ASCII format files used to generate VOTables. VOTables will not be generated if they are not requested by the user.

4.3.4 Overcoming Initial Problems With Data Processing

Initial problems were experienced when producing a VOTable XML file from the output samplegals ASCII table. The first approach involved creating a MySQL data table and occupying the table with values read from the samplegals ASCII table by reading on a line-by-line basis and inserting a row into the MySQL table for each line read. This intermediate MySQL table was then used to produce a VOTable by reading the MySQL table using SQL statements. However, after testing with a relatively small samplegals ASCII table of approximately 60MB in size, it was discovered that this method involved significant lengths of time because of the need to create the intermediate data table.

The introduction of the intermediate MySQL table was initially performed so that should the data be used for purposes other than producing a VOTable (for example, to make the table able to be queried using SQL statements from e-Galform), a MySQL table would be present in order to enable such further functionality. However, given the significant delays produced by even relatively small tables, it was decided that this intermediate table should be removed.

Rather than read the ASCII table and accordingly create rows in a MySQL table, it was decided that the VOTable should be created by reading the ASCII data table on a line-by-line basis, and for each line read, immediately write directly to a VOTable file rather than first writing to an intermediate MySQL table. This reduced processing times to a small fraction of the former method.

4.3.5 Problems Of Showing Real-time Progress Of e-Galform Processing To User

One of the major disadvantages of the server-side web application taken, in contrast to the advantages outlined in the introduction, is that all major algorithmic processing takes place on the server, making incremental indication of the progress of processing and writing VOTables or ASCII tables on the client, in 'real time', particularly difficult.

The server-side call to *exec()* to execute samplegals is a blocking call; that is, PHP does not execute any further lines of code until *exec()* returns, when samplegals has completed generation of the Galform ASCII table. During the execution of samplegals, therefore, it is not possible for PHP to carry out any further instructions, including informing the client computer in some fashion of the progress of samplegals. Because the ASCII table is located on the server, it is not possible for client-side code to access size information concerning the ASCII table, which could be used to indicate the progress of samplegal's execution to the user.

In addition, it is not possible to display text on the browser's window through server-side scripting by placing HTML output in the PHP script before the *exec()* call, as no page is sent to the web browser (the client) until the complete PHP script has completed execution, including the *exec()* function.

It may have been possible to show a "Processing, please wait..." indicator of some form, with no information on the size of the ASCII table during generation on the server, by using client-side code placed in the start.php script (see Figure 4.6), which is executed before the generate.php script responsible for executing samplegals. This would prevent a blank screen from displaying during generation of the ASCII table and VOTable (if requested) for potentially long periods, which may be a source of confusion of the user as to the current run's progress.

4.3.6 Automated Management Of Files And Data In e-Galform Using CRON

With multiple user sessions and runs within any given period, e-Galform may quickly generate large amounts of data, particularly Galform ASCII files and VOTable XML files. For example, a typical e-Galform run for a single redshift may produce hundreds or thousands of megabytes of data, including both ASCII and VOTable files, depending on the number of properties requested. This will quickly increase for a single session if multiple redshifts are requested by the user (requiring ASCII and VOTable files for each redshift).

Requesting runs also makes data entries into the various "Store" database tables (for example ParameterStore and FilterStore table, see Section 3.2.1) with reference to the unique session ID for the user. To prevent this data building up over time, e-Galform periodically removes files and session-related data from the MySQL database automatically, by expiring data in after a fixed period to prevent the server's storage space from reaching its limit. To do this, e-Galform's cron.php script is executed via a CRON job (see Chapter 11 for configuration issues) every hour.

When a registered user executes a run, e-Galform logs their unique session ID and RegUserID (Section 3.2.1), and also a UNIX timestamp, indicating the date and time of execution via the Logs (Section 3,2,1) table. Data written to the server's storage space (ASCII and VOTable format files) are then written to the *output/* directory, found in e-Galform's root web directory, which contains a directory of name equal to the user's unique session ID. The session ID is used as the directory name to ensure that the output directory for the user's run is unique, and another concurrent session's files are not overwritten.

If an entry is found in the Logs table that is older than the expiry period when the cron.php script is executed, the data in the *output/* directory for the particular session ID and entries in the "Store" tables (all rows matching the session id) are removed. The choice of the fixed length of time for this window depends upon the storage capacity of the server, the length of time taken per run, and the length of time it may typically take to download data to a client computer after the run has finished.

For example, if a typical user executed two runs of two redshifts at 1 GB/redshift, and e-Galform typically at peak usage attracted 5 users per 24 hour period, the server would need a storage capacity of approximately (5 * 2) = 10GB storage capacity to accommodate such users. If a user were able to transfer data from the e-Galform server to their own computer at a rate of 50Kb/sec (a rate typical of, for example, a domestic broadband connection with University networks likely accommodating greater bandwidth) then a 2GB download for each user would take approximately 12 hours.

Generating data for such runs may take up to several tens of hours to complete (dependent upon the number of properties requested for output). Therefore, it is suggested that the 24 hour period, chosen initially for testing of a smaller dataset, be increased significantly to, for example, two weeks (or 350 hours) according to the number of users observed and the storage capacity of the server.

It is suggested that the storage requirements are monitored as e-Galform becomes more popular, and that this figure is adjusted. If storage capacity limits are reached too quickly, this time window should be reduced in size to save server storage space. If the storage capacity of the server is not taken full advantage of, the time window should be increased in order to allow e-Galform users more time to carry out runs and download data from the server. The storage requirements during any given time window may be monitored via the size of the e-Galform *output/* directory.

4.4 e-Galform System Architecture

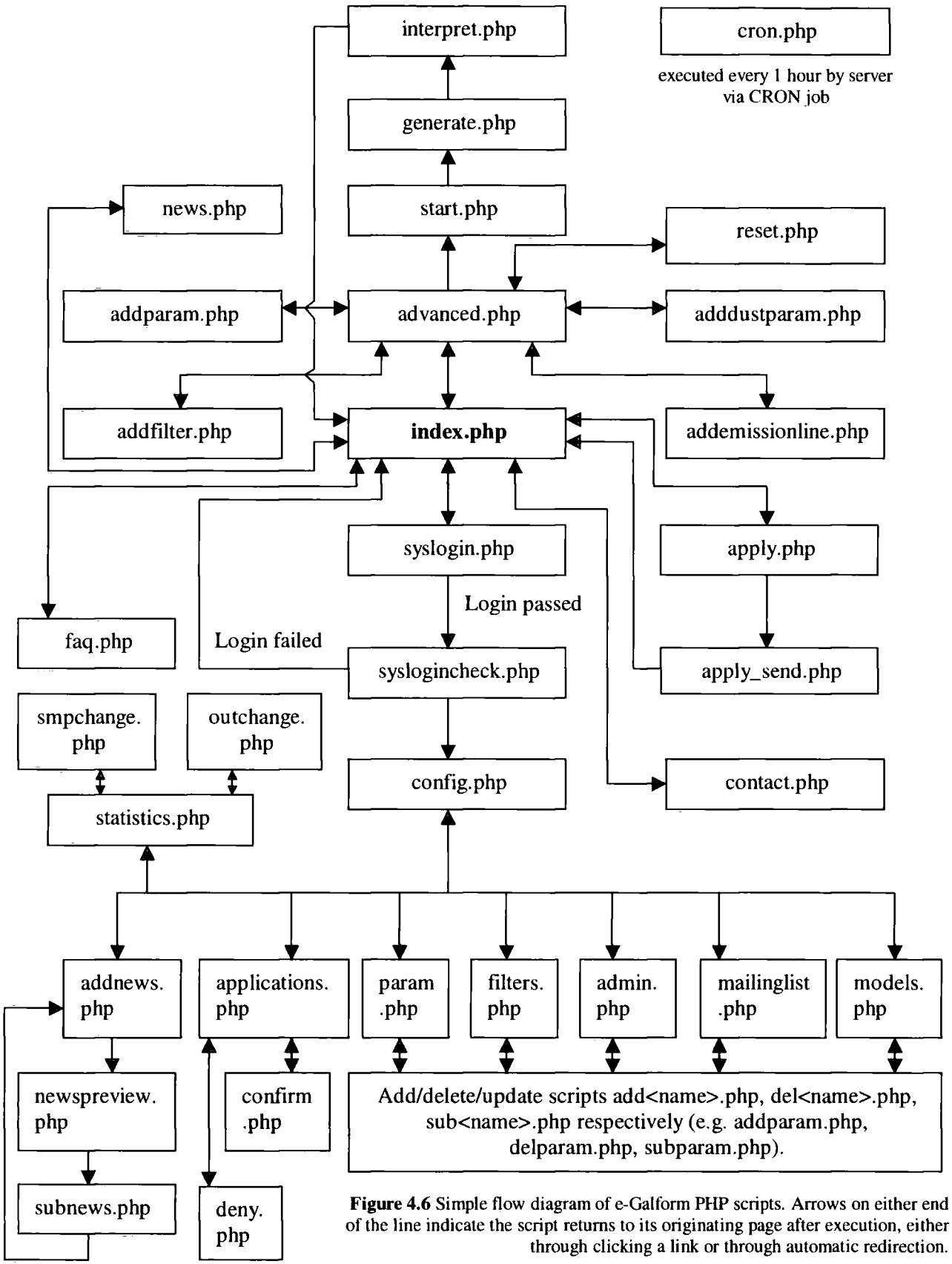

Figure 4.6 Simple flow diagram of e-Galform PHP scripts. Arrows on either end of the line indicate the script returns to its originating page after execution, either through clicking a link or through automatic redirection.

4.5 Verbal Account Of e-Galform Scripts

With reference to the flow diagram (Figure 4.6), the scripts listed yield the following functionality in e-Galform:

4.5.1 Front Page And Information

index.php – This is the front "gateway" page of e-Galform (Figure 4.2), which includes the news stream, the login facilities, the model selector and access to the frequently asked questions list. The news headlines are read from the News database table (Section 3.2.1), whilst the models available for selection are read from the Models database table (see Section 3.2.1).

faq.php – The e-Galform frequently asked questions (FAQ) list. The FAQ is 'hardcoded' into the script and is not editable via the e-Galform administration facilities.

contact.php – Displays contact details for the Galform research group, including the Institute of Computational Cosmology.

news.php – Displays e-Galform news stories, stored in the News MySQL data table. Data includes the news headline, story and timestamp indicating when the story was published. The script receives URL parameters via the front page (index.php) to indicate which news story to display according to the NewsID of the story in the News data table.

4.5.2 Running Samplegals, Generating VOTable Data

advanced.php – The simulation property selection tool (Figure 4.3), which also rejects users who do not authenticate. User interface settings are recorded in session variables so that, unless deliberately reset, the user interface will remain with the same settings whilst the current user session has not expired, even if the user views another page before returning to the selection interface (see Section 4.2.1).

addparam.php/adddustparam.php/addfilter.php/addemissionline.php – These scripts record the user's selections for output properties, output emission lines, dust properties, and output filters (see Section 4.2.1). Property selections are stored with session ID in the ParameterStore table, emission line selections are stored with session ID in the EmissionLineStore table, and filter selections are stored in the FilterStore table (see Section 3.2.1).

reset.php – Resets the current user's session data, so that when the similation property tool (Figure 4.3) is reloaded it will reside in its default positions. It also removes all references to the current user's session ID within the ParameterStore, EmissionLineStore and FilterStore tables (see Section 3.2.1).

start.php – This page summarises the user's settings and requested data, and indicates that the user should leave their browser open for a potentially long period of time whilst processing is carried out. This helps the user to understand the way their computer might behave whilst running a job through e-Galform.

generate.php – This script is responsible for generating the required Galform ASCII tables via samplegals. It translates the user interface settings for the user's current session (determined via cookies) first entered via the interface Figure 4.3 into commandline parameters. samplegals.exe is then executed, on the server, for each redshift requested (see Section 4.2).

interpret.php – Now that the Galform ASCII tables have been generated by samplegals according to the user's settings, they are now translated if required into VOTable schema compatible XML files (see Section 4.3.2), and finally links to the data files (both ASCII and VOTable) are given, along with a notice that they will expire.

4.5.3 Handling Of User Registration Applications

apply.php – This script presents the user with a form asking them to leave their name, institution and e-mail address in application for becoming a registered user of e-Galform. Registration allows the user to authenticate and thus use e-Galform facilities.

apply_send.php – Once the user selects to submit their application, PHP's *mail()* function is then used to e-mail administrative users via their e-mail addresses stored in the AdminUsers table (see Section 3.2.1). Details are then stored in the Applications table awaiting confirmation by an administrative user.

4.5.4 Administration Facilities And Authentication

syslogin.php – Requests the user to enter their username and password in order to access e-Galform administrative facilities via a simple web form.

sysloginchеck.php – Verifies the username and password entered on the web form against the username and password stored in the AdminUsers table (see Section 3.2.1). If not authenticated, the user is not allowed to pass.

config.php – This page displays the e-Galform administration menu (see Figure 8.5). Menu options are "hard coded" into the PHP script and may not be amended using e-Galform administration facilities.

applications.php – Allows administrative users to accept or decline applications issued by applicants via the web form (see apply.php, Section 4.5.3). The script reads the Applications data table (see Section 3.2.1) and displays the applicants via a web form, with option to accept or decline the application.

confirm.php - Accepted applicants are then e-mailed automatically to confirm registration using PHP's *mail()*function (with appropriate content and e-mail headers), and are then moved from the Applications data table (see Section 3.2.1) to the RegUsers data table (see Section 3.2.1).

deny.php – Applications who are denied registration are removed from the Applications table (see Section 3.2.1) via this script. Declined users are not e-mailed to indicate that their application has been declined.

addnews.php – Displays a web form allowing an administrative user to compose a news story to add to the News data table (see Section 3.2.1). The web form includes a headline and news story body. The news story headlines are displayed on the front page via the index.php script (Section 4.5.1).

newspreview.php – Displays a preview of the news story (in terms of its layout on the screen when viewed via a headline link from the front page) and allows the administrative user to submit or reject the story.

subnews.php – Upon requesting submission of the news story via the news preview page (newspreview.php), the news story is stored in the News table, with timestamp indicating the date and time of submission.

param.php/filters.php/admin.php/mailinglist.php/models.php – These scripts display web forms showing the galaxy properties, filters, administrative users, mailing list members and Galform model papers stored in the Parameters, Filters, AdminUsers, MailingList and Models data tables respectively (see Section 3.2.1). Entries may be removed by checking a checkbox next to each desired entry (multiple entries may be checked). Checked entries may then be removed by clicking the 'delete' button. New entries may added by clicking the "add >>" link on each page. For each table, a similar approach is adopted. Please see Section 8.3 for an example of adding a data entry to the mailing list table. Figure 8.6 shows the mailing list table data display, generated by mailinglist.php.

addparam.php/addfilter.php/addadmin.php/addmailinglist.php/addmodel.php – These scripts are responsible for adding a new entry to their respective data tables. Pages display forms requesting the administrative user to enter new data parameters according to the fields required for each table. Please see Figure 8.7 for an example of the web form allowing the addition of a new entry to the MailingList table.

subparam.php/subfilter.php/subadmin.php/submailinglist.php/submodel.php – These scripts are responsible for validating and writing the contents of the data entered in the respective "add entry" web form (generated for example by addmailinglist.php for the MailingList table). They then use an HTML redirect to return to their respective

data management page (generated for example by mailinglist.php for the MailingList table).

delparam.php/delfilter.php/subadmin.php/delmailinglist.php/delmodel.php – These scripts are responsible for deleting entries selected by the user in the respective data management page (generated for example by filters.php for the Filters table). POST variables are used to determine the data entries selected in the respective data management page, before using an HTML redirect to return to the respective data management page.

statistics.php – Shows statistics regarding e-Galform's use for the benefit of those managing e-Galform and monitoring its growth in traffic; including but not limited to:

- The date and time of the last run.
- The current total number of registered users.
- The total number of e-mail addresses on the e-Galform mailing list.

It also allows the configuration of core paths; for example changing the output directory of the session data, and changing the path of execution for the samplegals.exe program.

smpchange.php – Writes the new path for execution of samplegals.exe binary executable according to changes made on the statistics page (generated by the statistics.php script), and returns directly back to the statistics page.

outchange.php – Writes the new path for the output directory in which to store session data (for example ASCII and VOTable files), and returns directly back to the statistics page.

cron.php – This script is executed on a periodic basis (currently suggested on an hourly basis - see Chapter 11, Server Configuration And Installation) via the use of a CRON job. It periodically removes files, data and directories created to store session specific data for particular runs within a given time window (see Section 4.3.5) thus preserving

server storage space and preventing, over time, the server from reaching its storage limits thus preventing e-Galform from functioning.

globals.php (not included in flow diagram) – Included via the #include directive in most e-Galform scripts, this script contains commonly used functions, such as the do_html_header() and do_html_footer() functions which produce the consistent header and footer HTML shown on each page of e-Galform. It also includes the database connection function do_dbconnect() and a range of other functions which are used throughout e-Galform scripts.

5. Other Astronomical Database Applications And Comparison With e-Galform

Previous chapters have explored the use of web technologies in the development of e-Galform. This chapter will explore other astronomical database applications and systems, including grid computing and Astrogrid, and compare e-Galform with similar efforts to make galaxy formation data available publicly via the internet.

5.1 Astrogrid And Grid Computing

5.1.1 Grid Computing: General Principles

Grid computing, in general, involves the creation of a virtual infrastructure in order to federally unite a number of computer networks and data systems into a single point of use or system. This includes making databases across different networks interoperable, and making efforts to share data more efficiently between different groups; including but not limited to the development of common standards so that data formats need not be converted from one format to another. It also involves integration of different applications, perhaps executed across different computer networks, available in a single system. Grid computing may also involve the use of multiple computers to carry out processing tasks, breaking down the data to be processed into smaller "pieces", generally called "workunits", and synchronising the processing efforts of individual computers so that they may act collectively in a way similar to a much more powerful, single computer operating on the same data {55}.

SETI@home (Search For Extraterrestrial Intelligence)

For example, in an astronomical context, the SETI@home project run by University of California at Berkely (http://setiathome.berkeley.edu/) takes advantage of the computers of many different individuals (over 5.2 million) across the globe, through the use of a

screen saver which activates when a computer is idle, thus taking advantage of its otherwise used processing time. Users voluntarily download the screen saver, in return for which they are shown graphics depicting a visual representation of the analysis whilst the screen saver is active on their computer (see Figure 5.1). The goal of SETI@home is to analyse patterns in radio signals collected by SETI's radio telescope at the Arecebo Observatory in Puerto Rico for possible signs of extraterrestrial intelligent life.

Grid Republic

SETI@home has recently become a member of the Grid Republic (http://www.gridrepublic.org) group, part of the BOINC (Berkeley Open Infrastructure For Network Computing) project also run by Berkeley, which is seeking to allow users to share their idle processing time between multiple different grid systems (for example as well as SETI@home, the user can choose other projects such as BBC Climate Prediction, and Einstein@Home, which is seeking to find pulsars from data provided by gravitational wave detectors). Users can also decide how their processing resources are shared between projects (see Figure 4.2 for an example of GridRepublic Desktop user interface for Windows).

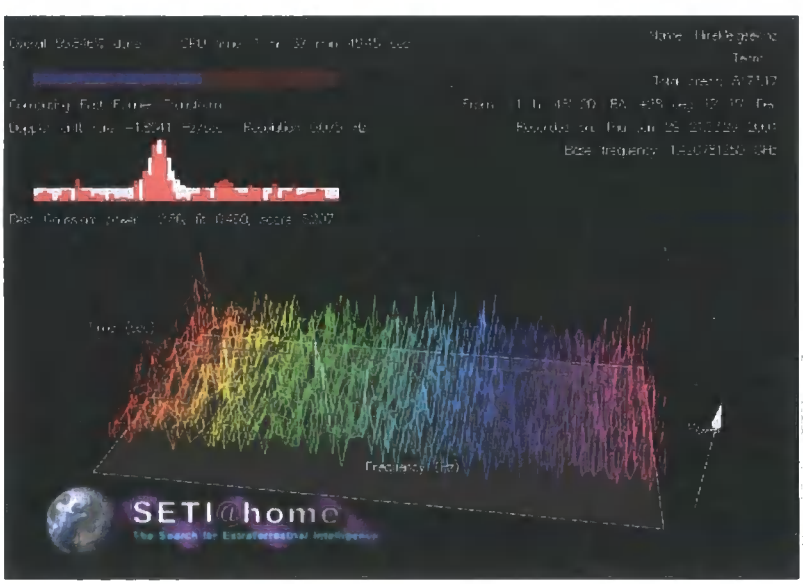

Figure 5.1 The SETI@home screensaver, which is active during otherwise idle processing time.

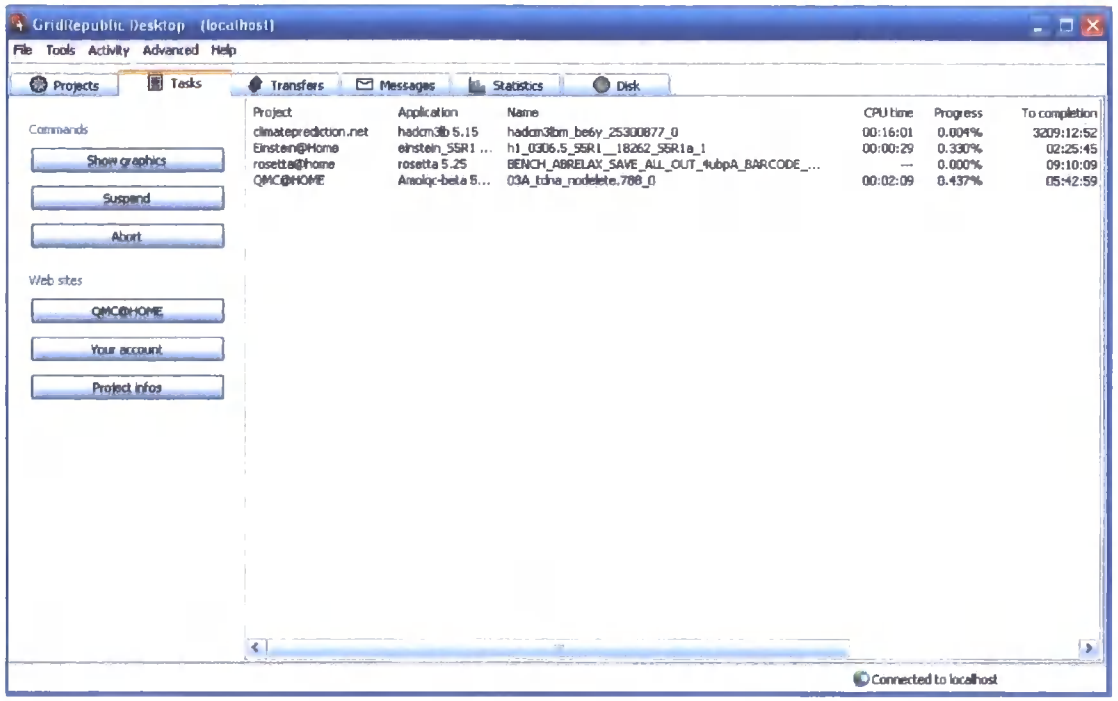

Figure 5.2 The GridRepublic Desktop application allows multiple grid networks (mainly scientific and medical research) to process "workunits" on a single computer during idle processor time.

5.1.2 Astrogrid's Key Aims

Astrogrid is currently being developed by a number of consortium members, principally the Queens University of Belfast, University of Cambridge, University of Edinburgh, University of Leicester, University of London, University of Manchester and Rutherford Appleton Laboratory. The University of Edinburgh is currently playing a leading role {56}.

The project was started during 1999, when it was recognised that the sizes of data files produced by astronomical observations were accelerating in size at a rate greater than database technologies were advancing. There was also a need to "federate" datasets taken of the same patch of sky with different telescopes. In response to OST "e-science" initiatives, the AstroGrid consortium formed during summer 2000. According to the AstroGrid information website (see http://www.astrogrid.org), Astrogrid's key aims are the following:

- To create a working datagrid for key UK databases
- High throughput datamining facilities for interrogating those databases
- A uniform archive query and data-mining software interface
- The ability to browse simultaneously multiple datasets
- A set of tools for integrated on-line analysis of extracted data
- A facility for users to upload code to run their own algorithms on the datamining machines
- An exploration of techniques for open-ended resource discovery

5.1.3 Key Principles Of Astrogrid In A Grid Computing Context

The overriding principle of Astrogrid is to make key databases – from various research groups – interoperable; that is, to become seamlessly browsable and able to be queried in a simple way without the need to know the location of the database or to be familiar with any peculiarities about its composition, and also to offer a single "virtual" infrastructure through which to access a number of integrated applications from different source networks. Astrogrid is planned to lead on to communal development of database analysis and exploration tools, to enable easy data mining; that is, the extraction of information easily from within the database tables themselves using an easy set of tools. The final step is to develop information discovery tools; that is, the combination of various data sources and tables to produce new information and data that may be analysed, sourcing multiple data tables and sources.

5.1.3 Key Motivations

Downloads of data from the HST archive {57} are now growing faster than the rate of data accumulation, growing at 1-2 TB/year with daily download rates of 20GB/day. The SOHO archive {58} similarly is growing at 1TB/year. Such growth is mostly due to growth in detector size, which in line with computer processing power has been exponential for three decades. In some cases, for example in the storage of pixel images, data sizes may be reduced by the use of compression algorithms without much loss of original data. This does, however, place larger requirements on processing power as the data must be decompressed once delivered.

'Collectivisation And Empowerment Of The Individual'

Similar to e-Galform, the aim of Astrogrid is to bring together the work of different departments in a way that will not require those from each department to fully understand the technical background of the work. In this way, departments may use each other's data to further their work without needing to invest the time to learn the data formats involved. However, Astrogrid does not intend to fulfil the functions of e-Galform, in that Astrogrid's goal is to provide an infrastructure to bring data and applications together into a unified framework, rather than generate scientific data itself. The Astrogrid project also seeks to make possible, through easy to use datamining tools, the manipulation of the increasingly large datasets, which might not otherwise be possible using current datamining tools. By making tools customised to astronomical use, such datamining can be made more intuitive and focussed on certain key tasks. For example, an optical astronomer can use sub-mm or radio data without worrying about the way in which basic operations such as astrometry differ.

Some Example Key Tasks

- The federation of different datasets for a common patch of sky obtained from multi-wavelength observing campaigns.
- Rare object searches – involving trawling through large databases, looking for (quite literally) one in a million objects. For example, cool white dwarfs have been found in the (extremely large scale) SuperCOSMOS database, constraining the nature of Galactic dark matter. Quasars have been found at z=4 in the APM data.
- Deducing the history and structure of the Galaxy from stellar motions requires huge statistical manipulations, as does the determination of cosmological parameters from the anisotropy of the microwave background. The use of techniques from computational geometry can make such manipulations much easier, and quicker.
- The star formation histories of galaxies in clusters could be obtained via population modelling from ugrizJHK diagrams, with the same descriptive powers for galaxy evolution.

- Massive photometry – the monitoring of thousands or millions of objects simultaneously can yield exciting advances in micro-lensing, dark matter, high redshift supernovae and the cosmological constant.
- Finally, previously unknown objects may be found in the combination of previously unfederated large and disparate datasets; it may lead to the discovery of new types of objects in the cosmos.

It may be possible to integrate e-Galform into the Astrogrid framework; see Section 7.2 for suggestions as to a possible approach.

5.2 GalICS User Interface

GalICS has a basic implementation of a query facility form on its website (see Figure 5.3). Unlike Astrogrid, it is directly comparable in aims to e-Galform. The GalICS web interface allows the direct execution of an SQL query on the underlying GalICS database tables, stored on a MySQL server. Due to overuse, the facility is currently deactivated {59}.

Figure 5.3: The GalICS Project SQL query page

In comparison to e-Galform, the query facility suffers from a number of disadvantages.

(1) It assumes knowledge of the structure of GalICS tables before use. This would normally require the user to have read thoroughly all technical documentation on the site, which takes time.
(2) The data output itself lacks portability due to the lack of storage of verbal definitions of parameters in the output file.
(3) The output is given in its own proprietary format, rather than seeking to conform to an international standard such as VOTable.
(4) There are no limitations on use to the public, and hence the GalICS team have experienced problems with overloading, leading to the facility being deactivated temporarily at time of writing. In e-Galform, the form is password protected and therefore only confirmed registered applicants for e-Galform may use the service.

The user may, however, if already familiar with SQL queries and having read the documentation available on the GalICS data tables, find the SQL interface more flexible. However, the e-Galform interface generates data from the underlying Galform binary data according to the user's requirements (including for example, a number of redshifts and output galaxy types), and educates the user as to the meanings of the astronomical properties they are choosing to extract without the need to reference external material.

5.3 US National Virtual Observatory – Theory in Virtual Observatory (TVO) Demonstration

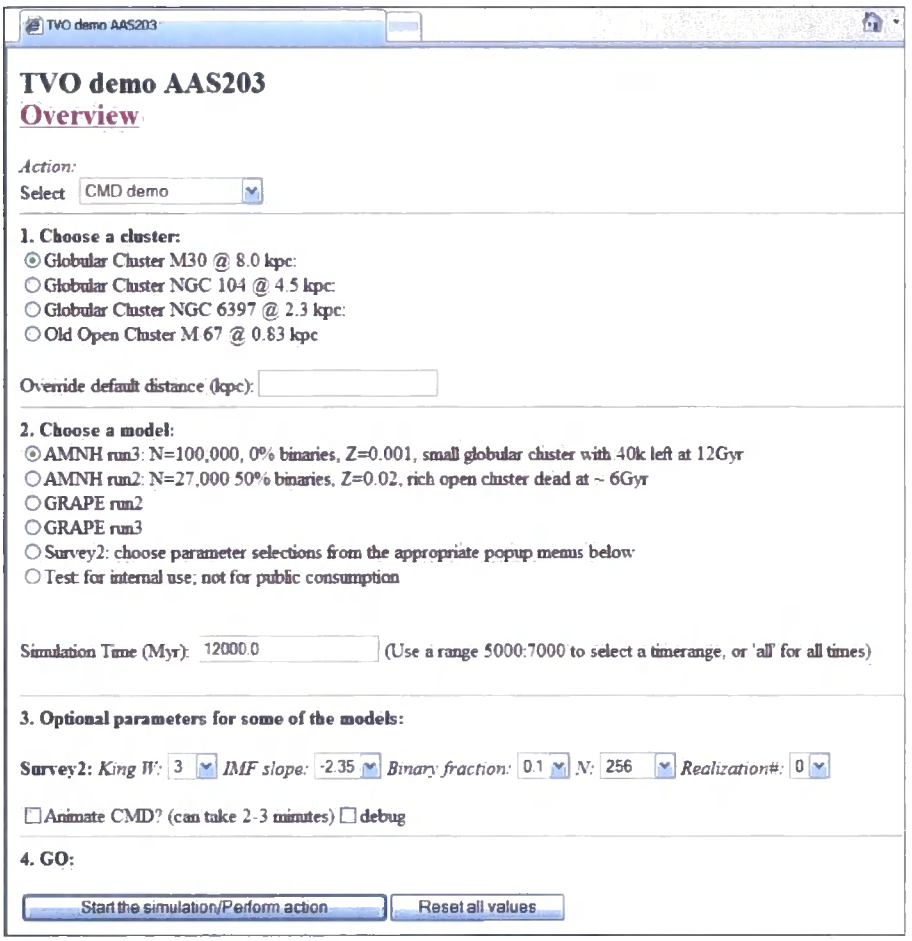

Figure 5.4: The TVO demonstration web interface

The Theory in Virtual Observatory (TVO) demonstration user interface, shown at the 203rd meeting of the American Astronomy Association (AAS) is shown in Figure 5.5. It was developed by Peter Teuben (see http://bima.astro.umd.edu/cgi-bin/nemo/tvo3.pl). The interface uses Perl scripting (another form of server-side development language), combined with VOPlot, which is an applet that may be used to plot VOTable data (Figure 5.5).

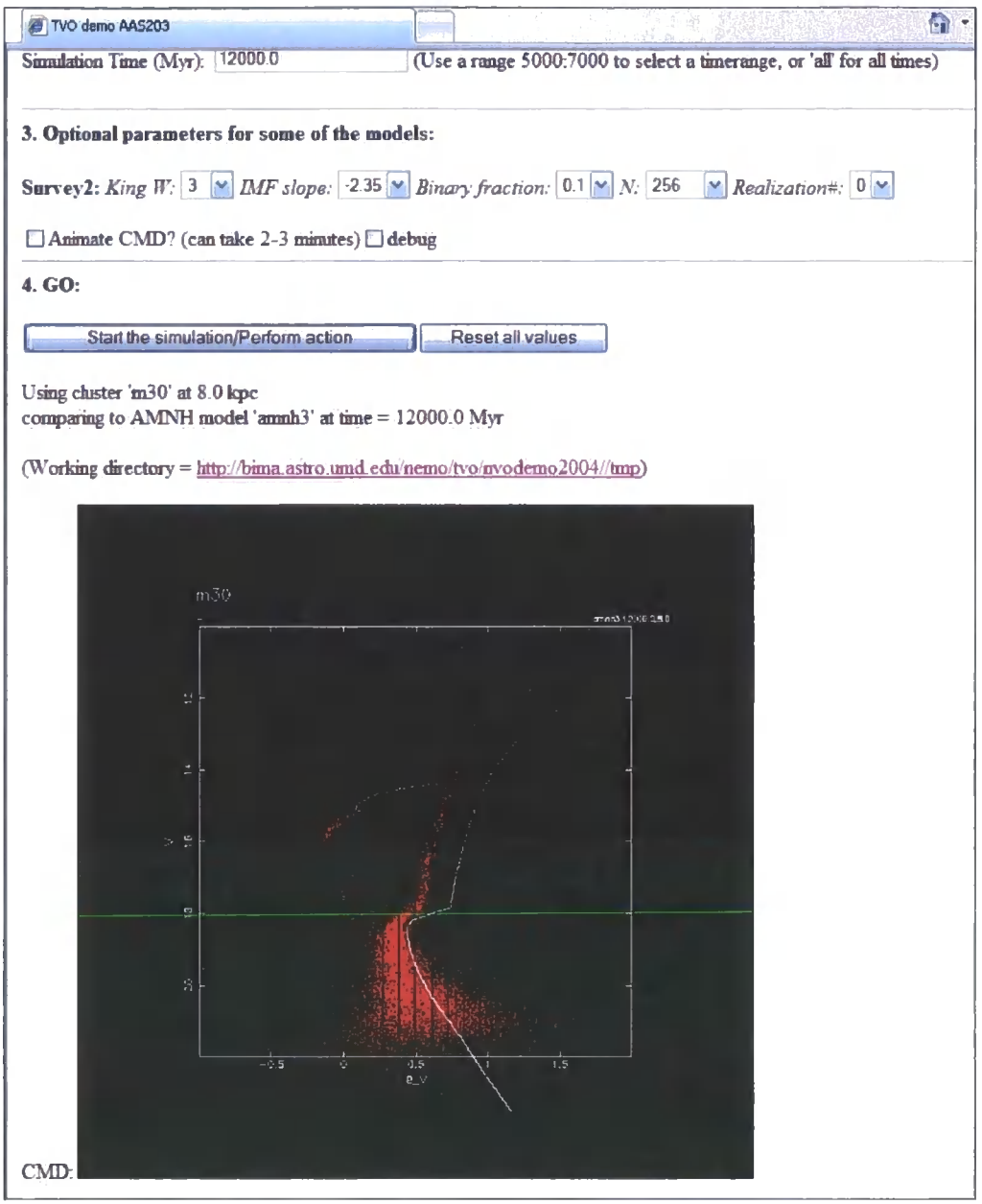

Figure 5.5 – A VOPlot graphic of a colour-magnitude diagram for a globular cluster displayed after use of the TVO interface. The interface accesses a series of pre-run simulations to plot data according to the user's requirements.

It is similar to e-Galform in that it accesses pre-generated data to produce a result. However, TVO plots data using VOPlot (e-Galform does not graphically display data) and has a simpler web interface due to the lack of options presented to the user relating to the run settings.

There is no need, for example, for cookies or for temporary data storage of a series of parameters selected by the user for their specific data session in a database, whereas e-Galform stores a collection of parameters and settings selected by the user for each run in a database according to the user's unique session identifier (see Section 4.2.1).

There is also no form of public authentication or registration of users (the demonstration can be run by any member of the public) and no news facilities, or an ability or apparent need to dynamically extend the facility (for example, by using configuration facilities to add new properties or selections to those available on the web form). The appearance of the interface is also more basic and based upon default web browser styles. At time of writing, the demonstration also appears to produce only one form of result (a plot of globular cluster M30), despite the requesting of several different types of plot.

Overall, as the options offered to the user of the demonstration are rather more limited than in the case of e-Galform, the interface is simpler. However, the use of VOPlot might prove useful in the case of e-Galform. A future improvement to e-Galform might be to add VOPlot support so that the user need not download the XML file and load separately into a program such as TopCat (see Section 7.3).

6. The Use Of e-Science Principles In e-Galform

6.1 Primary Objectives Of e-Galform And e-Science

The primary objectives and philosophy of e-Galform, as stated in the introduction, are as follows:

- It allows other research groups to use Galform data for their own comparisons and in their own research and data processing efforts without requiring finer technical knowledge of Galform or samplegals. This allows for more accessible use of Galform data and conclusions by other research groups, in effect promoting the use of Galform work. It may thus also increase the academic citations of Galform work.
- It provides an easy to use, verbose interface for scientists throughout the globe to access without the need for any further software other than a standard web browser.
- It acts as a 'hub' for information regarding the progression of both Galform and e-Galform, and a communications tool allowing a large number of prospectively interested scientific departments to be informed as to the existence of e-Galform, and advances in the Galform model.
- Its underlying database tables, where appropriate, are capable of being extended and edited without the need for technical understanding of the underlying application code. This means it becomes independent of its original author but still maintains the ability to adapt to advances in Galform or samplegals over time, provided that any advances require changes to the underlying data only rather than processing or user interface (or database schema).

This chapter will expand on these points and also make further suggestions for the e-Science movement as related to the development of e-Galform. e-Science by definition essentially involves the use of the internet (mainly through web browsers) or visualisation to allow research scientists to share their knowledge and data through the internet ('e' stands for *electronic*).

The National e-Science Centre (NESC) {60} for example, defines e-Science as the following:

> *"What is meant by e-Science? In the future, e-Science will refer to the large scale science that will increasingly be carried out through distributed global collaborations enabled by the Internet. Typically, a feature of such collaborative scientific enterprises is that they will require access to very large data collections, very large scale computing resources and high performance visualisation back to the individual user scientists.*
>
> *The World Wide Web gave us access to information on Web pages written in html anywhere on the Internet. A much more powerful infrastructure is needed to support e-Science. Besides information stored in Web pages, scientists will need easy access to expensive remote facilities, to computing resources - either as dedicated Teraflop computers or cheap collections of PCs - and to information stored in dedicated databases."* – National e-Science Centre website (http://www.nesc.ac.uk/nesc/define.html).

6.1.1 Professional Presentation And Communication

One of the main features that distinguishes e-Galform other systems has not only been to make using the facility straightforward to those not technically literate in the basic workings of *Galform* or *samplegals*, but also to make it as professionally presented as possible, similar to a commercial website, despite the fact that e-Galform does not sell products. Many alternative systems currently online often arguably neglect or deprioritise the presentational aspects of their facilities, and although this is not relevant to the underlying science, it contributes to the enjoyment of using the facility and likely to the impression of the underlying work. By providing a presentable, professional and easy to use portal through which to coordinate Galform research news and data sharing, e-Galform seeks to build a large number of interested users.

6.1.2 The e-Galform Communications System

e-Galform contains both components of a successful communications campaign, similar to that used in profit and non-profit institutions: a *launch* campaign (to recruit future users of the system) and a *maintainance* campaign (to retain interest in e-Galform into the future and to provide alerts to new developments and facilities) {61}.

e-Galform uses an e-mail based communications system, as opposed to more traditional forms of communications such as letters or paid-for advertisements. As e-Galform is not selling a product or service, advertising in its traditional form is not appropriate to the aims of the communications system, which is to develop and maintain an interest in the Galform project and to encourage use of Galform data via e-Galform for academic research. One of the primary disadvantages of e-mail communications however is that there is a danger that they will be ignored, or regarded as 'spam' by the receiver. However, if the mailing list is developed appropriately, any e-Galform news should be more likely to generate interest.

The communications plan is comprised of two components. The launch campaign will comprise of a brief e-mail synopsis of e-Galform and an invitation to register a username and password on the e-Galform system. A pre-prepared list of e-mail addresses of those potentially interested in the project is used to target individual or group e-mail addresses, and this will also be the basis target list for the maintenance campaign. I acquired a list of beta-testers after giving a Friday journal club talk (please see Chapter 9 for a discussion of e-Galform beta testing).

All e-mails submitted by e-Galform are plain text-based, so as to make sure that all potential receivers of the e-mail are able to read messages created via the e-Galform administrative facilities. This is due to the fact that many scientists use UNIX based text e-mail systems that are unable to handle HTML e-mail messages (Microsoft Outlook for example was one of the first e-mail handlers to support HTML e-mails, which may include HTML tables and pictures as well as text).

The 'maintenance' e-mail campaign will be facilitated via the news announcement system. When a new news article is written using the e-Galform administrative

facilities, it is automatically e-mailed to registered users (see MailingList table, Chapter 4). These news articles may contain developments to e-Galform, or perhaps more often the Galform model itself.

6.1.3 e-Galform Target Demographics – Who Will Use e-Galform?

e-Galform produces data that is only scientifically useful to those involved in astronomical research; either observational or theoretical. It fulfils the need for a straightforward, easily accessible facility for researchers to use in exploiting the results of the Galform galaxy formation simulation according to their needs.

The target e-mail list must target this demographic profile. The list may be extended by administative users of e-Galform in the future via the e-mail list editor interface (see Section 8.3.2), which forms part of the administrative facilities for the site. Sources of e-mail contact information on research staff who fall within the boundaries of the target demographic profile include, but is not limited to:

- NASA ADS website: http://adswww.harvard.edu. This website contains comprehensive indices of papers published by astronomy research staff (either theoretical or observational astronomy). Interested parties could be found via this source by looking for scientific papers of published work in similar research fields, and also for observational astronomers whom might benefit from the ability to compare their observational results with those produced by theoretical models such as Galform. If e-mail addresses of individual research staff are not immediately available from research papers, a Google search should suffice as most research staff publish their e-mail contact addresses online.
- Similar e-mail contact addresses could be obtained via the preprint server for astro-ph (http://xxx.lanl.gov/archive/astro-ph).
- The websites of similar research projects – for example GalICS or Hydra, are very likely to be interested in using e-Galform. Links from these websites may lead to websites belonging to other research groups with a similar likely interest in using e-Galform or keeping informed of Galform progress.

- The e-mail addresses of those already in contact with the Galform research group regarding use of Galform data, either individuals or mailing lists for University departments.
- Individual University websites for Universities with Astronomy research departments.

It is estimated that the number of staff targeted may reach hundreds of e-mail addresses globally, however this estimate is particularly difficult given the uncertainty of the numbers of staff involved in sufficiently parallel scientific projects without completing research into appropriate University mailing lists, or e-mail addresses of individuals. In terms of technical limitations, the e-Galform database is capable of holding many thousands of e-mail addresses, as a MySQL currently can store up to 2 TB of data per table on Windows NTFS systems, or 2 GB for Linux based systems {62}.

6.1.4 What Need Does e-Galform Fulfil In Respect Of Its Users?

e-Galform has been developed in response to the growing e-Science movement and the trend for the development of new infrastructure and standardisation of data formats at an international level. The benefits of this are easier access to the research work of groups globally, the introduction of a unified infrastructure of exploration tools which seamlessly access and harness multiple data sources, and the saving of time in the development of new tools which adhere to new global standards, rather than requiring the understanding of and conversion between differing data standards. This will enhance the accessibility of scientific data, and help accelerate the development of tools that adhere to a smaller number of data standards globally. e-Galform helps in this process by producing a VOTable standard output of the Galform galaxy formation model (see Section 4.3.2), and also potentially the ability for registration with the Virtual Observatory (VO; see Section 7.4) for use by those requiring access to theoretical data on the formation of galaxies.

6.1.5 e-Galform Interface And Style

Figure 6.1 The e-Galform logo

The author has designed a distinctive logo for the system that forms the e-Galform 'brand' (see Figure 6.1). The interface also uses standardised type faces, colours and interface structures to maintain an ease of use and professional looking image. Consistency has partly been achieved by the use of HTML header and footer functions standard to all pages on the system (developed via PHP functions) and also the use of CSS (Cascading Style Sheets, see Section 3.2.7) to standardise typefaces and properties of the various HTML elements on each page (for example borders around HTML table cells, cell colours and widths).

In Summary: e-Galform Principles

- To make the e-Galform system adaptable to new Galform models without the requirement for administrative users of e-Galform to have working knowledge of PHP or MySQL, provided that changes to the Galform model require only additions to the current e-Galform background database (e.g. adding new properties to the Parameters table, see Section 3.2.1).
- To create a standardized, easy to use and recognisable interface for the system.
- To create a communications system to launch and grow the userbase within academic departments globally, and to retain this userbase using regular e-mail communications via news announcements.
- To create a commercial quality look and logo for the system in order to make use more enjoyable. The e-Science movement currently does not focus on presentation, rather interoperability and the construction of global infrastructure; in terms of web interfaces to a group's research, an improvement to the principles of the e-Science movement may also be to adhere to high quality standards of presentation in terms of website layout, and ease of use of help systems to aid other users to understand the work.

- To produce standardised VOTable file that is portable, self-contained and able to be used in external database applications for processing in other scientific contexts.

7. Scientific Exploitation Of e-Galform

7.1 Use Of VOTable Format

7.1.1 The International Virtual Observatory Alliance (IVOA)

One of the primary aims of e-Galform was to produce a facility capable of converting the samplegals ASCII table output into VOTable format, the primary goal of which has been to enable the use of Galform data in a variety of circumstances, for example in other database applications that adhere to the growing VO standard.

The VOTable format was developed as part of the International Virtual Observatory Alliance (IVOA - http://ivoa.net/) group, which includes projects such as Astrogrid (http://www.astrogrid.org). The International Virtual Observatory Alliance was formed in June 2002 "with a mission to facilitate the international coordination and collaboration necessary for the development and deployment of the tools, systems and organizational structures necessary to enable the international utilization of astronomical archives as an integrated and interoperating virtual observatory" {63}, otherwise known as a "grid". Currently the IVOA coordinates eight Working Groups as well as four Interest Groups in areas such as applications and theory. The efforts of the Working Groups have been coordinated and focused through five international Interoperability Workshops held in the US, the UK, France, India, and Japan between October 2002 and May 2005.

7.1.2 Example Of Use And Benefits Of VOTable Data Format

VOTable is a data format capable of being imported by the TopCat graphing and tabulating package (part of the Astrogrid suite of tools) to generate graphs from e-Galform data. Because the VOTable format contains verbal descriptions of all fields in <FIELD...></FIELD> tags as generated by e-Galform code (see Section), it is possible through TopCat for users of this package to understand the meanings of all properties from the descriptions contained within the VOTable format. This offers a significant advantage over the ASCII format table, which does not contain verbal descriptions of the data. This is particularly of value for those that are not immediately familiar with

Galform, and helps fulfil one of the primary aims of e-Galform, which is to help those in observational fields or from other research groups use Galform data in their own research in an efficient and easy to understand manner.

7.2 Application Integration Of e-Galform Into Astrogrid

A fundamentally important use in the future may be to interface e-Galform with Astrogrid, so that e-Galform is more visible and more likely to attract regular use. Integration of the application into Astrogrid may offer a more superior method of promoting e-Galform to relevant parts of the scientific community than using a mailing list and news service or blog.

e-Galform may be potentially integrated into the Astrogrid Workbench (see Figure 7.1), as part of the suite of tools offered by the Astrogrid system. This would require the Astrogrid team to place an icon linking to the e-Galform website on the Astrogrid Workbench. A suitable Astrogrid Workbench folder for e-Galform might be the "Data Discovery" folder, or an additional folder dedicated to data providers via a web interface. Adding e-Galform to Astrogrid would require the cooperation of the developers of Astrogrid Workbench, which has been developed using the Java programming language.

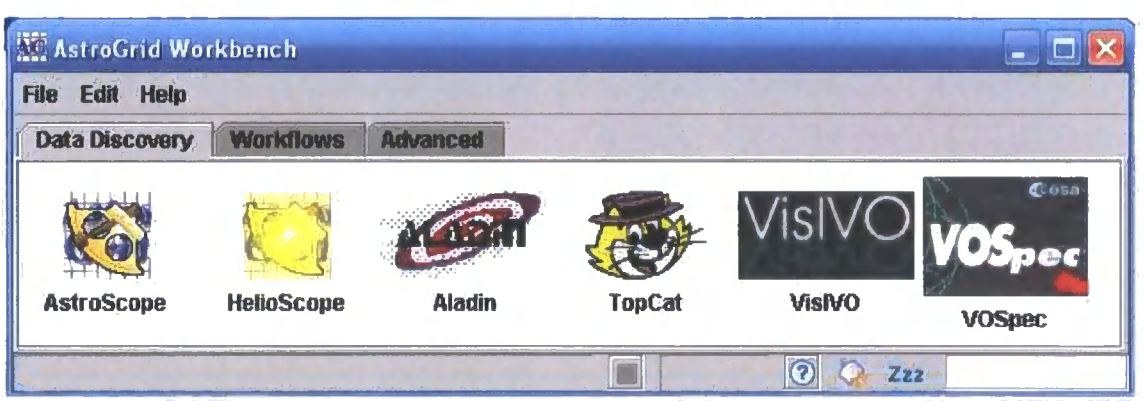

Figure 7.1 The Astrogrid Workbench utility, running in a Microsoft Windows environment.

7.2.1 Using Astronomy Client Runtime (ACR)

However, as e-Galform is a data provider rather than a data discovery or exploration tool in itself, it may be more appropriate in the context of use within AstroGrid to register e-Galform as a service at the ESA-VO, thus making the service accessible via the Astronomy Client Runtime (ACR) API provided by Astrogrid Workbench {64}. This will enable other developers of VO compatible applications to access e-Galform services using other development languages such as C++, Perl or Python, with some reservations (see Section 7.4 regarding possible issues in allowing external access to e-Galform's functionality). Making e-Galform accessible externally would enable other research groups to seamlessly use Galform data through their own scripts without having to use the e-Galform site directly, making e-Galform a potentially valuable data provider for theoretical galaxy formation research (for example, a program that sought to compare theoretical models may wish to use such a service). External applications would still access the e-Galform server and make use of its processing and storage capabilities when requesting data through their own scripts (in addition to web users of e-Galform). However, such access would also be restricted using Astrogrid's own authentication facilities.

An important advantage to users accessing the facility through the ACR would be the seamless upgrading of data provision with underlying upgrades to the e-Galform service. As the Galform model developed and e-Galform was upgraded with new Galform base data and a new samplegals binary, any external access to e-Galform would then automatically benefit from such changes, without the external facility having to make any changes to their own code. Such a characteristic highlights the importance of developing standards of data specifications when developing an interoperable grid.

7.3 Use In TopCat Astronomical Plotting Tool

The VOTable format, as explained previously, may be used in the graph and tabulation package TopCat, developed by the Starlink team. Because of the verbose storage of the e-Galform data in VOTable format, in data grid view, verbose descriptions of the data

parameters are available. Figure 7.2 illustrates the use of TopCat in creating graphs of VOTable data generated by e-Galform. TopCat is available either as a standalone service via the Starlink team's website, or as a "data discovery" tool via the Astrogrid Workbench.

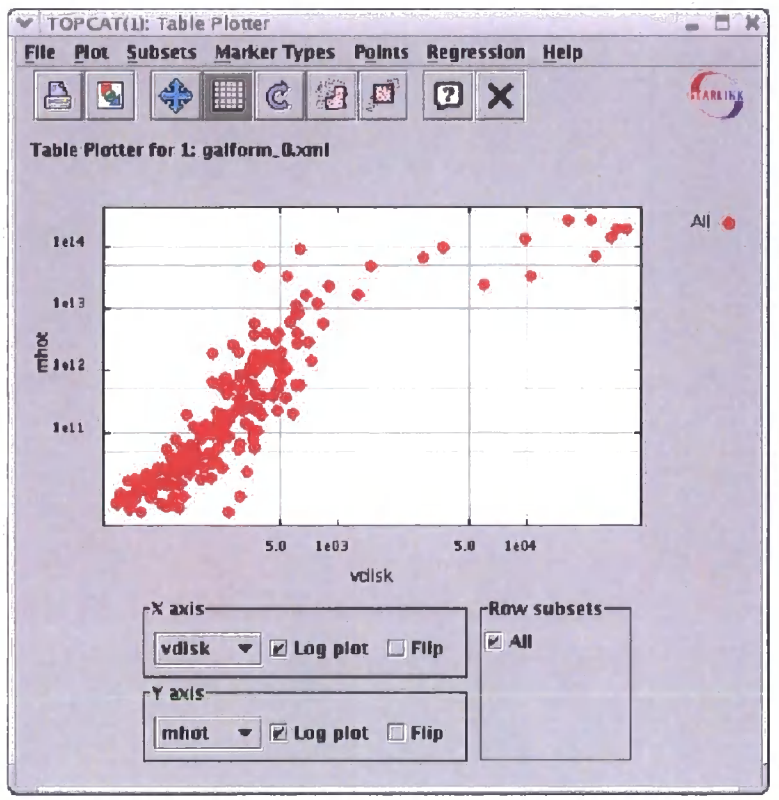

Figure 7.2: A TopCat graph generated from an e-Galform generated VOTable. The *y*-axis shows the mass of hot gas in units of $h^{-1}M_{sol}$ and the *x*-axis shows the rotation speed of the disk in km s^{-1}. Note that some disks seem to have excessively high rotation speeds; this is linked to the implementation of adiabatic contraction.

7.4 Registration Of e-Galform With European Space Agency Virtual Observatory (ESA VO)

7.4.1 Registration As A Service

The discoverability of the service would be further enhanced by registration as a "service resource" at the ESA VO, which may be found at http://esavo.esa.int/registry/. The following information would be relevant for registration as a service resource,

however refinement of the choice of names and titles may be required by the Galform research group before submission:

Resource Type: Service

- **Title:** e-Galform : A data provider for the Galform galaxy formation research group.
- **Identifier:** (Dependent upon choice of IVO identifier address by Durham University)
- **Short name:** e-Galform
- **Status:** Active
- **Password:** (Choice by Durham University)

Resource Content

- **Subject:** Hierarchical Galaxy Formation
- **Description:** e-Galform is a VO enabled web interface to University of Durham's Galform semi-analytical hierarchical galaxy formation simulation.
- **Publisher:** University of Durham
- **Publisher ivo-id:** (Dependent upon choice of IVO identifier address by Durham University)
- **Reference URL**: (Choice by Durham University)

Misc. Informations - Interfaces

- **Interfaces**: Web Browser
- **Access URL**: (choice by Durham University)

7.4.2 Integration With Other VO Applications

Section 7.2.1 discusses the possibility of making e-Galform a scriptable service via registration with ESA-VO and the Astronomy Client Runtime (ACR) provided by Astrogrid Workbench. Registration allows the specification of POST variables and a

URL address to a script accepting such variables. Appropriate POST variables could potentially be passed to the generate.php (see Section 4.5.2) script from a remote location (for example, perhaps through another web or non-web based application through the ACR) without requiring the use of the e-Galform property selection interface (see Figure 4.3). However, given that the property interface also stores the user's requested properties in the "Store" database tables (see Section 3.2.1) and reads these tables when running samplegals, passing POST variables to the generate.php script may not be adequate, and true integration may require the creation of a new script to handle direct execution via passing POST variables only.

7.5 Use Of VOTable For Computational Comparison With Other Theoretical Galaxy Formation Simulations And Observational Data

7.5.1 Computational Comparison With Other Theoretical Models

As discussed in Section 2.5, one of the major facets of exploiting e-Galform scientifically might be for statistical comparison with the results of other, contextually equivalent, galaxy formation simulations. This may first present a few barriers. At time of writing, major competing models such as GalICS (semi-analytical approach to simulation) and Hydra (particle mesh based approach to simulation) do not offer support for VOTable XML format, which is the only data format that e-Galform supports that may offer the opportunity for competing models to compare directly without some form of intermediary conversion software.

In exploiting e-Galform scientifically in comparing results to other theoretical galaxy formation models, there would also have to be a manual, contextual match between the properties present in the VOTable output of one galactic formation model when compared with another, as property labels or descriptions for the same properties may differ between models simply because of the choices adopted by each group.

VOTable's UCD (Unified Content Descriptor) and unit attributes do however provide a potential means of standardising property descriptions between research groups (besides forming a committee between multiple galaxy formation groups to agree upon

common labels or descriptions for scientifically equivalent quantities; see Section 10.8 for an example of adding possible UCD and unit attributes for e-Galform). UCDs have not presently been implemented in e-Galform due to the lack of development of the standard, but given that the standard is actively under development, it may be useful to include the ability to record UCD attributes with e-Galform output in preparation for future extensions to the standard.

7.5.2 Comparison With Observational Sky Surveys: Sloan Digital Sky Survey And VOTable Format

Each row of data in the samplegals ASCII table (and hence e-Galform VOTable) represents a single model galaxy in the data. These properties are directly comparable, for example, with data given by sky surveys such as the Sloan Digitial Sky Survey (SDSS) {65}.

The SDSS data is available through the "SkyServer" (http://cas.sdss.org/dr5/en/), which is browsable via standard SQL queries. The SDSS website contains a comprehensive guide to the data schema through which the SDSS data is available. It is also a registered service (type "SkyNode") with the ESA-VO and hence is able to return data in a VOTable format.

e-Galform's VOTable could be compared with the SDSS survey data (and other surveys at appropriate redshifts) for example by using the Galform "weight" property (given in units of $Mpc^{-3}h^3$) for each galaxy to calculate a luminosity function, and this could then be compared, statistically, with an estimate of the luminosity function obtained from a sky survey.

This could be achieved, for example, by the use of the AstroGrid Astronomy Client Runtime (ACR; see Section 7.2.1). With both data sources accessible via the ACR, an application could be developed to calculate luminosity functions from Galform data via e-Galform against luminosity functions for galaxies from the SDSS via querying its SkyNode VO service. Because such an application would request data from both sources at time of execution, as both facilities were updated, such an application would immediately reflect such changes without needing any changes to its own code.

8. A Demonstration Of e-Galform

8.1 Front Page

The front page contains a username and password box (top right, figure 8.1), into which the user may enter their details if they wish to start using the facility. A correct username and password is required before the user may enter any pages enabling the execution of e-Galform runs; this is designed to prevent unsolicited use, which may potentially overload the server in demands for both processing time and server space. The selection box allows the user to choose their Galform model data (at time of writing, papers {20}, {21}).

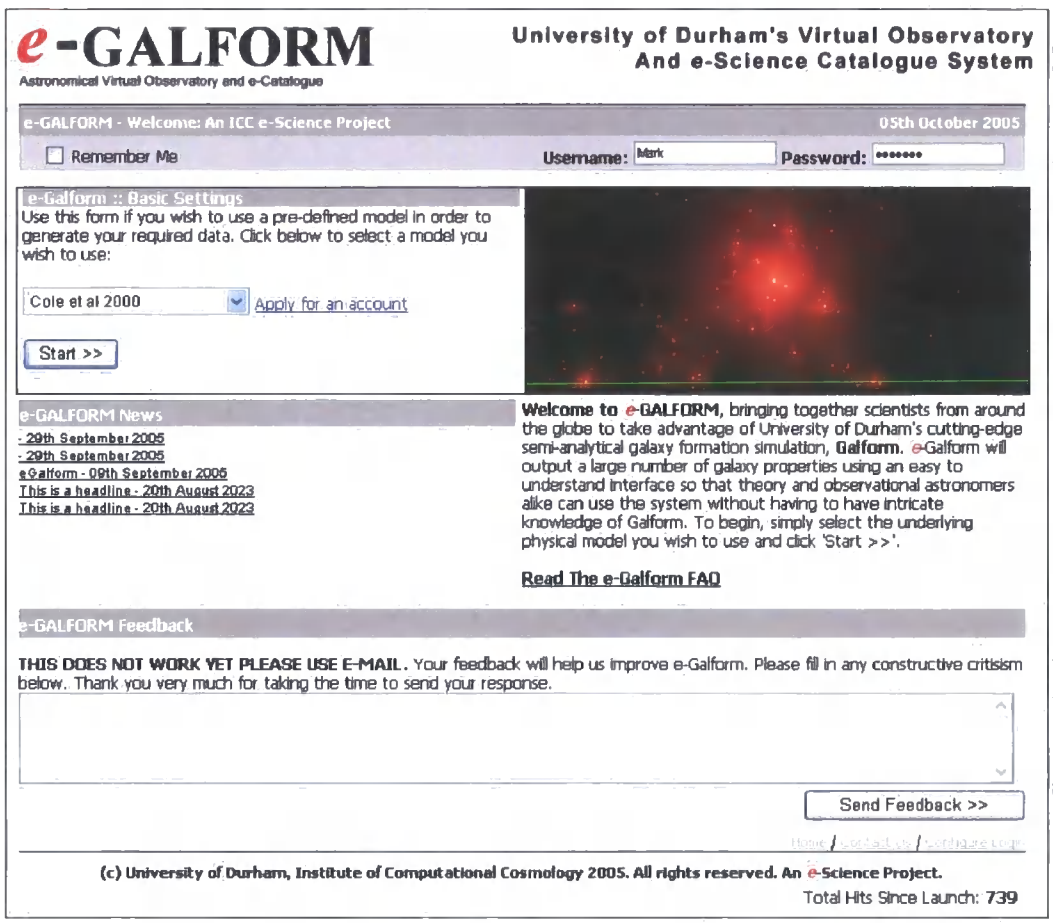

Figure 8.1: e-Galform front page

The front screen also contains an 'apply for an account' link which enables those interested in using the system to apply for an account. This automatically sends an e-mail to administrative users of the system, who may confirm or decline the application.

All passwords are encrypted in the database so that they remain private. There is also a FAQ (Frequently Asked Questions) link which is primarily directed to new visitors to the site, so they can learn basic facts about Galform, e-Galform and the functions the e-Galform performs.

8.2 Property Selection Interface

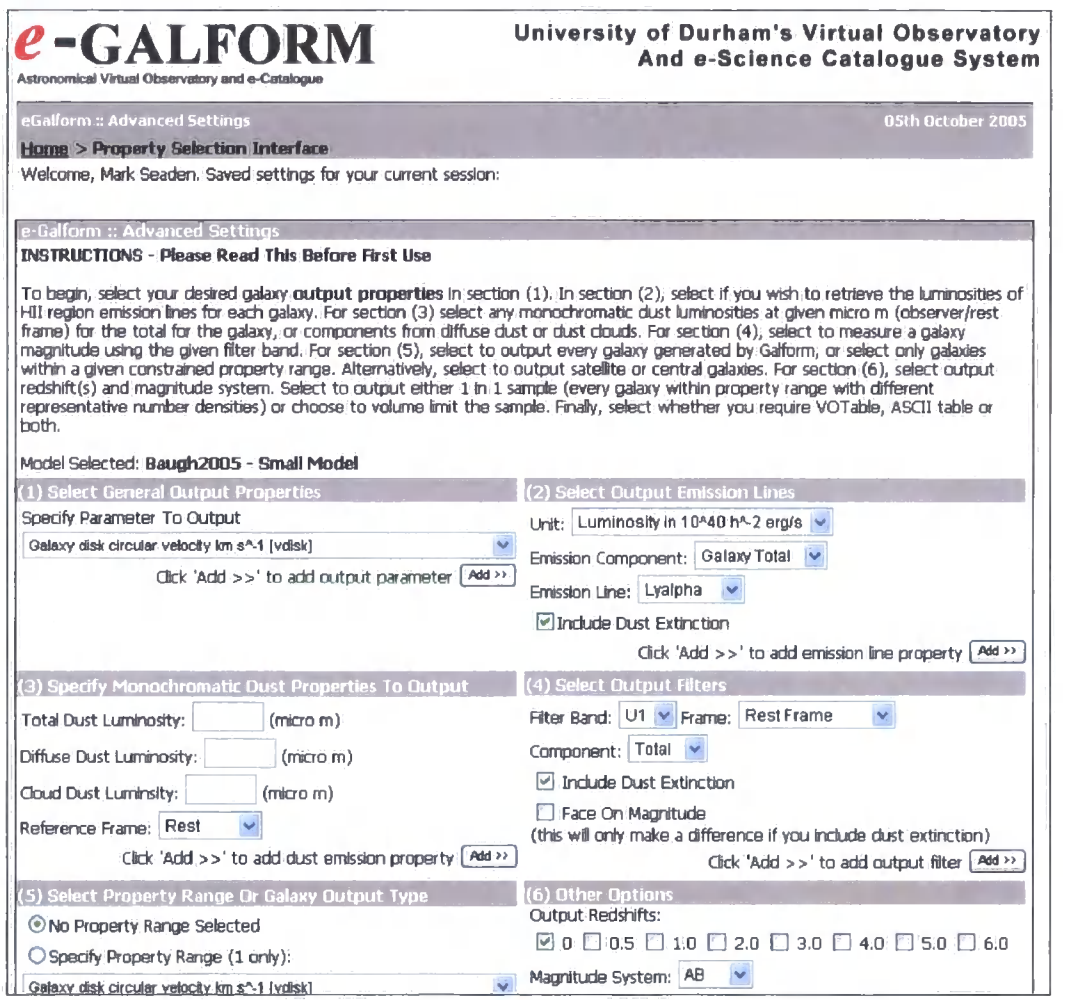

Figure 8.2: e-Galform property selection interface

Upon pressing the 'start' button, the user (after password and username are successfully confirmed) is taken to the property and settings selection page, where instructions are given in setting up the run. All properties, emission lines, monochromatic dust properties, and output filters are added through the use of individual 'Add >>' buttons and a drop-down selection box. When a user hits the 'Add >>' button, the setting is

recorded in the corresponding 'Store' data table (see Section 3.2.1), recording their unique session ID (generated by PHP) for their particular use. After other basic settings are recorded, the user is then instructed to press the 'Generate >>' button.

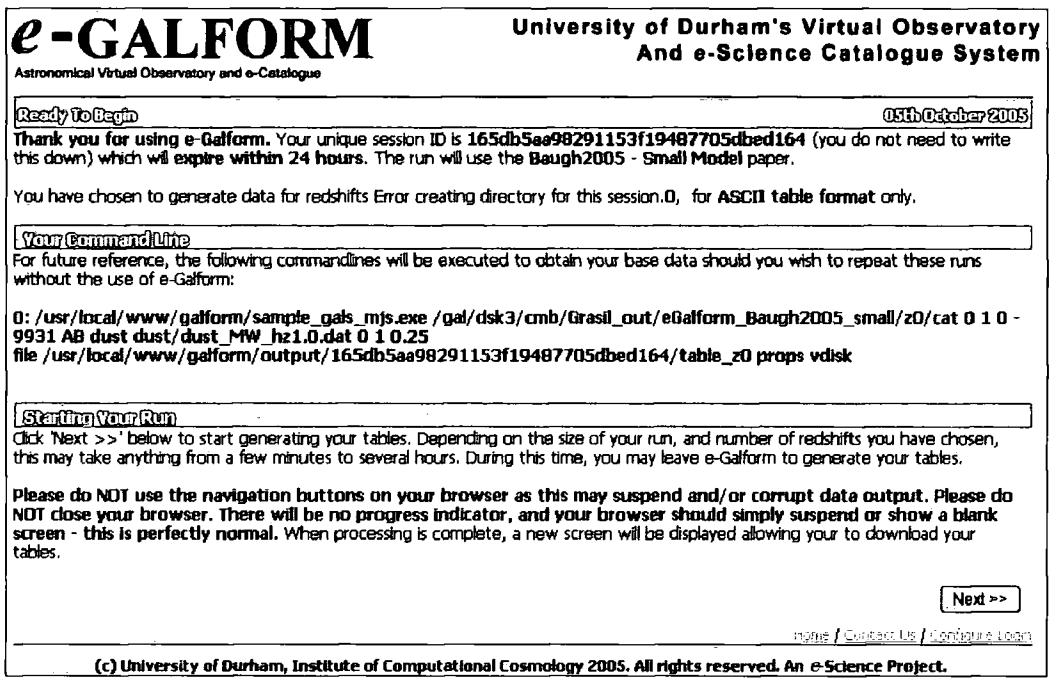

Figure 8.3: Pre-run summary page.

The above screen explains the settings that the user has requested, and outputs command line strings that may be copied and stored for future use. Finally, it explains what to expect whilst e-Galform is running and some precautions that should be taken in its use. Upon clicking the 'Next >>' button, e-Galform runs samplegals, and consequently creates ASCII and VOTable files that are downloadable by right clicking on a given link (see Figure 4.6).

Note that whilst the command line reference is given for each run, such a command line may only be executed locally and is useful only if the user makes specific contact with the Galform research team in order to repeat the run.

8.3 Administration Facilities

8.3.1 Administration Menu

In addition to generation of ASCII and VOTable format output, one of the primary key features of e-Galform is the background configuration system, which may be accessed by administrative users from any page on the e-Galform site.

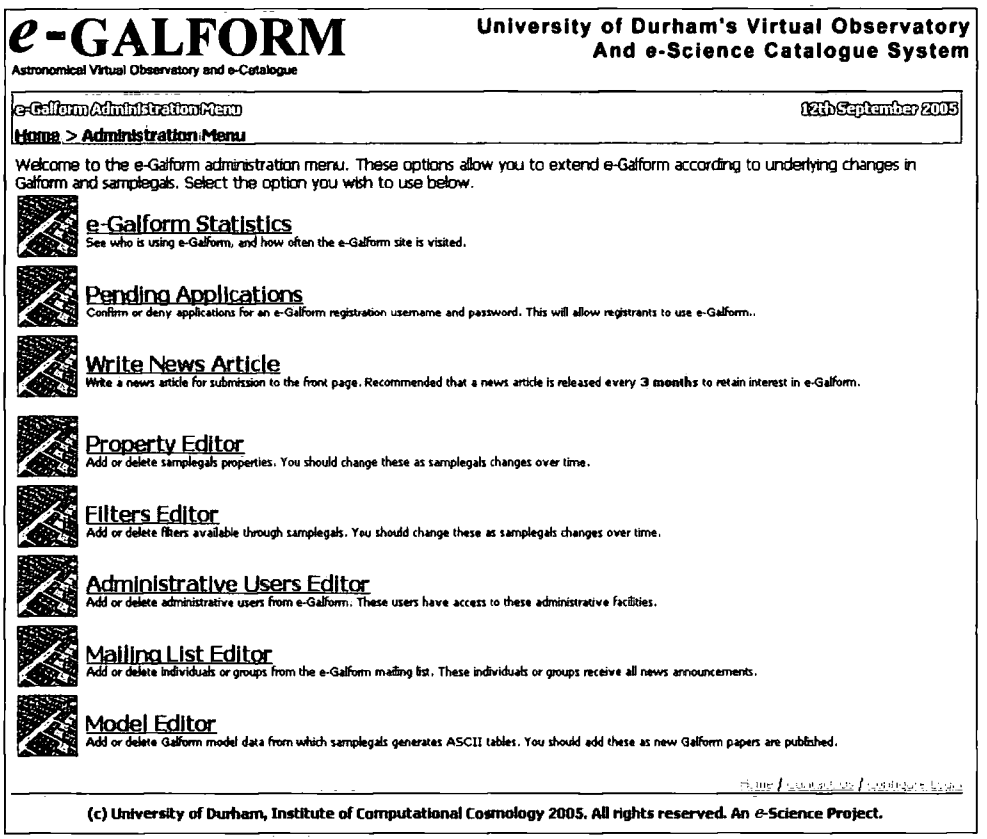

Figure 8.4: e-Galform administration menu

The system contains the following features:

- Addition and deletion of output properties, and output filters.
  - For output properties, this includes property label, description and unit.
  - For output filters, filter band may be indicated.
- Addition and deletion of background data according to Galform papers (e.g. {20}, {21}). This includes the base directory of the Galform binary data from which samplegals reads.

- Addition and deletion of mailing list entries to target individuals or group mailing lists in Durham and other institutions globally.
- Addition and deletion of administrative users with privileges to use the background configuration facilities.
- A background statistics analysis tool, which logs the use of e-Galform according to individual users and also tracks the number of visits to the front page of the site. It also tracks the ratio of numbers on the mailing list to those who sign up to use e-Galform.
- Access to the news facility, which e-mails those on the mailing list with news or announcements regarding e-Galform.
- A facility to allow confirmation of pending applications for those wishing to sign up to use e-Galform.
- A facility to list all registered e-Galform users, whom may be deleted by administrative users.

8.3.2 Example Of Adding An Entry To The Mailing List

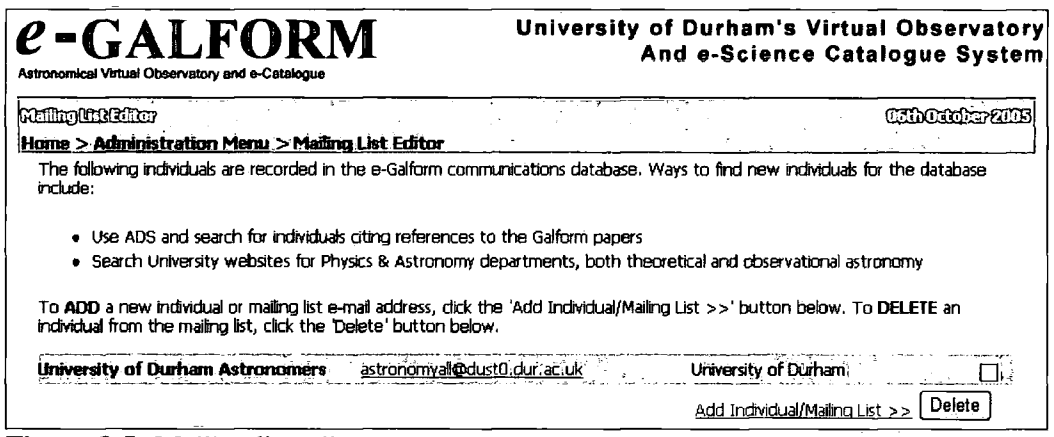

Figure 8.5: Mailing list editor

When 'Mailing List Editor' is selected, the user is presented with the above screen, which gives the option to either add a new mailing list entry (which may include those who have not yet registered with e-Galform) or to delete those already on the mailing list. The mailing list contains e-mail addresses who are automatically sent new e-Galform news stories via e-mail.

If the user selects 'Add Individual/Mailing List >>', the following screen is found:

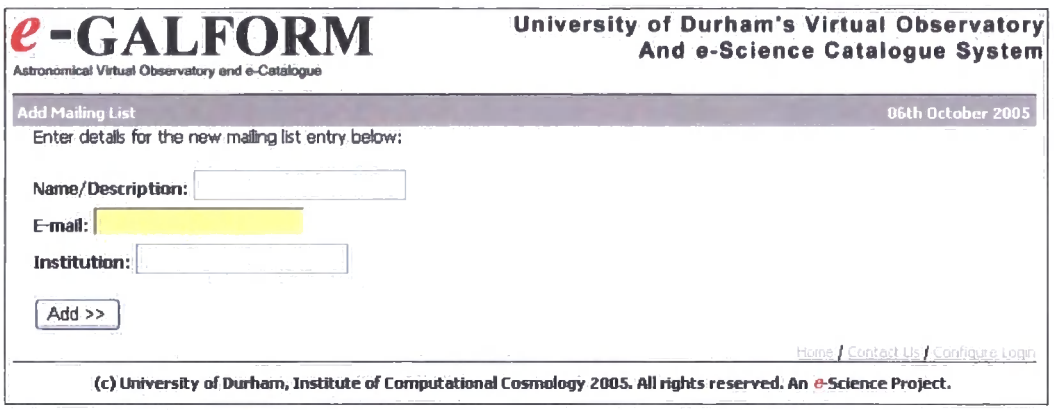

Figure 8.6: Mailing list data entry page

The user is then asked to type in the name or description of the individual or mailing list, e-mail address, and institution. The individual or mailing list is then added to the MailingList table via internal SQL queries upon clicking the 'Add >>' button.

9. e-Galform Beta Testing

9.1 Responses To Beta Testing e-Galform

9.1.1 Beta Testing The Interface

e-Galform was beta tested through consultation with 15 members of staff of the ICC. The aim of the test was to determine the ease of use of the interface, with a particular emphasis on the intuitiveness of the interface for those not immediately familiar with Galform.

This chapter will explore some of the responses and changes made to e-Galform in reaction to these responses. In general, initial responses were positive but with an emphasis on understanding the meanings of the property labels presented to the user via the interface. Other respondents requested that some of the controls be altered to be more intuitive, for example the sample volume setting. Initially, the interface contained a text box control with a default setting of zero. A zero setting requests samplegals to output data for an unspecified sample volume, however some users requested that this be changed to reflect the different meaning of a specified and unspecified volume.

The following is a response from ICC staff member, John Helly:

John Helly

```
Hi Mark,

I have a few comments on e-Galform. The interface is very impressive
and I found it very easy to use. I do have a few suggestions though:

I use Galform all the time so I can see that the parameters on the
interface correspond to the command line arguments for sample_gals,
but I think maybe someone less familiar with Galform would be left
with very little idea what e-Galform actually does.

It could do with some explanation of what its for. Obviously anyone
who wants the details of the model would have to read the papers, but
some basic information would help a lot. I think the points to make
would be:

- A quick explanation of what Galform is with some references
- Explain that we've done several simulations of galaxy formation
using different sets of assumptions about the physics involved
- The results of these simulations are available for download in the
form of galaxy catalogues for various different redshifts
- State the purpose of e-Galform - ie that it lets you extract the
  galaxy properties you want from these catalogues in an easy to use
  format. This isn't entirely clear from the interface at
```

the moment. Parts of it give the impression that you're running a simulation when you click the generate button, rather than extracting data from one of several ready made Galform simulations.

So if someone comes along wanting some theoretical results to compare to their data they know what they're getting even if they aren't involved in semi-analytics or simulations themselves.

The descriptions of the galaxy properties aren't bad but it might be helpful if each one had a longer description available - maybe you could put a 'Help' button next to the "Add" button to bring up a detailed description for the parameter. And you definitely need to put the units for each quantity in there somewhere! Are they in the output VOTable? (it wasn't working when I tried it)

The volume box needs some explanation. It really controls how the simulated galaxies are sampled, but thats not at all clear from the web page. For example, 'Leave at 0 for no restriction' - I know that means output all the galaxies without resampling them, but thats because I've used sample_gals before.

I hope some of thats useful. I'm very interested in how this project goes because I may need to do something similar with the galaxies from the Millennium simulation.

Cheers,

John

Figure 9.1 E-mail from beta tester John Helly regarding e-Galform

In response to these suggestions, the following changes were made to e-Galform:

- The front page was changed to include an introduction and explanation of both Galform and e-Galform, and in addition further information regarding the use of VOTable with other scientific database applications.
- A feedback form was also added to the front page so that any future users can suggest improvements 'on the fly'.
- The volume text box was removed and replaced with two options, either a one-in-one sample with galaxy weights (equivalent to zero volume) or a volume limited catalogue (nonzero volume specified by user).
- All units were added to the property descriptions, which are then subsequently included in the VOTable output data in the FIELD description tags. Prior to this change, the VOTable only included descriptions of output properties, rather than also including FIELD tags for emission lines, filters, and other properties generated by samplegals.

9.1.2 Other Suggestions

Observers were of particular focus in beta testing e-Galform, as it was this group that would likely have the most difficulty using samplegals directly. e-Galform hence was designed to bridge the gap between observers and theoreticians and allow them to understand compare research findings more effectively.

George Hau, observer at the ICC, requested that the sections of the interface be made more clear in their function. The introductory paragraph located above the main e-Galform interface was therefore changed to describe in clear English the meaning of the various interface sections, and some of the section titles were changed so that they were more concise. Descriptions of the output properties produced by samplegals from base Galform simulation data were confirmed by George Hau to be generally adequate from an observational perspective.

Ideally, however, more observers should have been involved in testing; a major cause of this was a significant delay (several weeks) between asking for beta testers at a Friday journal talk, and finally asking beta testers to participate in the testing session. Thus, when those who had signed up for testing were contacted via e-mail, only a relatively small fraction of those e-mailed responded. Also, a technical fault prevented actual running of data extraction during the first beta testing session, and this also served to reduce enthusiasm for involvement (this was the case for example in John Helly's comments in Figure 9.1 regarding the VOTable not being produced).

9.1.3 Testing The Validity of e-Galform VOTable Output

The e-Galform VOTable was tested by parsing to the graphics and analysis package TopCat, and was initially found to lack a <TABLE...> tag nested within the <RESOURCE...> tag. This was corrected, and subsequently found to function in all respects. The VOTable however should have been tested for validity using the VOTable DTD (Document Type Definition); see Chapter 12.

9.1.4 E-mail To External Domains

E-mails to external web addresses via the PHP *mail()* function were initially found not to be delivered by the web server. This was subsequently found to be due to the mail server not being configured to allow e-mails to external domains, rather only those on the local intranet.

10. Potential Improvements To e-Galform

10.1 Moving VOTable Conversion Into Samplegals Binary Executable

One of the assumptions made before developing e-Galform was that the PHP file handling routines would be efficient enough not to produce significant delay in producing a VOTable format file from the samplegals ASCII table format. However, given that PHP is executed by an interpreter based on the server, it cannot compare in speed and efficiency to that of compiled code executed on the server.

e-Galform could be made more efficient in terms of processing times by creating equivalent Fortran code based upon the e-Galform ASCII to VOTable interpreter, and merging this directly into the samplegals executable. A new commandline parameter could be created to ask samplegals to produce both the ASCII table and VOTable format. e-Galform could then post a link to the VOTable produced by samplegals, rather than a VOTable produced by its own server-side code. Currently, a run for several redshifts can take up to tens of hours to complete depending upon the number of properties requested by the user. This is likely to be unacceptable, particularly as it would require the user to have a web browser running uninterrupted during this period of time for successful delivery of the data. In seeking to improve e-Galform, attempts should be made to test how processing speeds were improved if the VOTable were generated by the samplegals binary executable.

10.2 Disappointing Response From Beta Testers Due To Post-Announcement Delay

The beta testing period was carried out later than planned following the Friday journal talk given to approximately 30 members of academic staff (approximately one month delay). Because of this, fewer responses to beta testing requests were acquired than should have been obtained for a reliable and productive beta testing period. However, at the end of the practical stage of the project, the system is in proper working condition, and it is hoped that once the site is setup on a dedicated server new users of the system

will post feedback on the site via the front page feedback system. Amendments will, however, require a person with a working knowledge of PHP and MySQL.

If the project were carried out again, it would be more prudent to have the project code at a more complete stage of development before asking for beta testers, therefore having a shorter period of time between the announcement of beta testing and the testing period itself. This would likely have resulted in greater interest in the testing of the project immediately after the Friday journal talk. Unfortunately, a series of technical setbacks resulted in the project being behind in schedule at the time at which the Friday journal talk was given.

10.3 Validation Of VOTable Output; Allowing Computational Comparison With Data From Other Research Groups

Whilst the VOTable produced by e-Galform during beta testing was found to be correctly opened by the TopCat facility, the VOTable should have been validated against the Document Type Definition (DTD) for e-Galform, found at http://us-vo.org/xml/VOTable.dtd.

In addition, whilst the VOTable <FIELD...> tags in their current form (Figure 4.5) are according to the specification, improvements could be made to implementation. For e-Galform to be completely interoperable, that is, both readable by other applications and to be useful as a tool to compare the results of Galform with other research groups conducting similar studies (either theory or observation), the following amendments and additions should be made to the VOTable output as follows:

- The **unit** attribute should be included in the <FIELD..></FIELD> tags for each data column expressed in the VOTable. The attribute is a string describing the unit used to describe the quantity in the data column.
- The **width** attribute should also be added to the <FIELD...></FIELD> tags to indicate the fixed width of the numerical data (currently 11 characters, as produced by the samplegals ASCII table).

- The **unified content descriptor** (ucd) attribute should also be included in the <FIELD...></FIELD> tags, if possible, for each of the properties (see below).
- To comply fully with VOTable document definition, <TABLE...></TABLE> tags should encapsulate the <FIELD...></FIELD> and <DATA...></DATA> tags. This was not included in the original VOTable generator code, however this is easily amended.
- The VOTable should then be validated against the DTD document. A simple VOTable may be validated for example through the use of the online XML validator at W3 Schools (http://www.w3schools.com/dom/dom_validate.asp). This tool uses Microsoft's XML parser to create an XML validator. The VOTable already includes a web reference to the DTD document at http://us-vo.org/xml/VOTable.dtd as an attribute of the <!DOCTYPE> tag at the header of the file.

10.4 Practically Implementing The Unit Attribute

Including the **unit** attribute within the <FIELD...></FIELD> tags will allow other applications to convert between units automatically if other research group's data is used to compare with that generated in e-Galform is expressed in different units. Implementing the **unit** attribute would be possible by creating a new table column in the Parameters table (see Section 3.2.1) containing the set of standard units specified by the Centre de Données Astronomiques de Strasbourg, as required by the VOTable format specification. This would then allow other applications to understand the units in which the Galform data is expressed in a common format, thereby achieving interoperability.

10.5 Practically Implementing The Unified Content Descriptor (UCD) Attribute

Including the **ucd** attribute within the <FIELD...></FIELD> tags will allow other applications to better understand the nature of the quantities in each of the columns of data within the VOTable. Currently, without the **ucd** attribute specified, the VOTable is valid but the nature of the quantities may be understood computationally from the

column label or description. The **ucd** attribute may potentially allow a computer to identify the nature of the property described by the data for a particular column. Implementing the **ucd** attribute would be possible by creating a new MySQL table containing the set of ucd attributes appropriate to the given physical quantity, determined from the published UCD semantics given at the IVOA (http://www.ivoa.net/internal/IVOA/IvoaUCD/VO-standard-vocabulary_8.pdf). Note that the process of developing unified content descriptors for various astronomical processes, objects, properties and other vocabulary is ongoing and not all Galform properties may have corresponding UCDs at present. However, given that the standard is developing, it would be sensible to assume that UCDs will become important in computational recognition of astronomical properties in the future. In terms of properties currently supported by samplegals, the following are sensible examples for CDS compliant unit descriptors and IVOA UCDs with respect to e-Galform:

Samplegals Property	**Unified Content Descriptor**	**Unit (CDS Format)**
Galaxy disk circular velocity	"phys.veloc;galaxies"	"km/s"
Velocity of halo in which galaxy is stored	"phys.veloc"	"km/s"
Temperature of diffuse dust	"phys.temperature;ISM.cloud"	"K"

Figure 10.1 Possible implementations of the UCD attribute for a selection of galaxy properties.

Note that current inadequacies of the UCD semantics include the inability to state a disk property (for example "galaxies.disk" is not available), and the inability to describe a halo object (for example "halo" or "haloes" is not available). There is also currently no object to describe a dust cloud (for example "ISM.dust" or "ISM.cloud.dust") either in the interstellar medium or within a particular radius of a galaxy. The unified content descriptors given most closely resemble the property being described at time of writing, in the opinion of the author.

10.6 Allowing Computational Comparison With Other Theoretical And Observational Research Groups

After the UCD standard has developed further, implementing both the **unit** and **ucd** attributes for computational reading may allow algorithmic comparison of datasets from different research groups modelling galactic halo formation, and also allow direct comparison with properties produced by sky surveys such as the SDSS with less work required when deciding which properties should be directly compared. However, direct comparison will still require other research groups to produce valid VOTable data.

In addition, given that the choice of UCDs for a particular astronomical property is not absolute but at some level subjective, agreement between research groups is still likely to be a necessity prior to computational comparison. It does, however, improve the ability to describe astronomical properties within a VOTable according to a defined, machine readable format, contrasting with the e-Galform VOTable in its current form.

Comparing data computationally would be very important in understanding statistical differences between the predictions of different models quickly and easily. By adjusting methods or assumed properties within a particular model, improvements to underlying prescriptions could be tested against other theoretical models and observational data (where available) without any reformatting or prior preparation being required and would be a very useful way of exploiting Galform's predictions scientifically.

10.7 Adding Newsfeed Generation To e-Galform And Feed Reader To Astrogrid

Both the RSS (definition of format found at http://blogs.law.harvard.edu/tech/rss) and the ATOM data format (definition of format found at http://tools.ietf.org/html/rfc4287) are compatible with many software applications that accept newsfeeds, for example FeedReader {66} and the new Internet Explorer 7 {67}.

It is foreseeable in the future that academic research groups may wish to publish updates on their models via RSS or ATOM XML formats, so that other research groups

may read them on a suitable newsfeed parser on a daily basis. Different research groups would also be able to display news from other research groups on their own websites, by forming HTML from RSS or ATOM feeds.

RSS output could be added to e-Galform by reading the RSS specification (reference given) and re-generating an RSS file at a specified location on the e-Galform server whenever an administrative user added a news story to the e-Galform site. This would, in addition, help continue to promote Galform research.

A potential improvement to the Astrogrid system may be to develop an integrated, Java-based newsfeed reader that could act as a pool of information regarding the development of astronomical research projects globally, including newsfeed published by the Galform team. Different users may wish to subscribe to different RSS feeds via AstroGrid depending on their current interests.

10.8 Adding An e-Galform Blog

Instead of, or in addition to, an RSS/ATOM newsfeed, an e-Galform blog could be added to the e-Galform site allowing research staff to add new developments, ideas or updates about Galform or e-Galform quickly and easily for reading by interested research staff in both Durham and other Universities. This could be achieved through a number of free PHP weblog libraries, for example Serendipity (http://www.s9y.org), which does not require the administrator to have any knowledge of PHP or MySQL to install and administrate.

Many blog systems, for example "blogspot" run by Google (see http://www.blogspot.com) now feature RSS/ATOM feed so that a person's blog can be syndicated onto another site or read by a specialised feed reader without the blog's author needing to do any development work themselves. In this case, the blog would fulfil much the same function as the news facility, which is already built in to e-Galform.

10.9 Compressing VOTable Data At The Server Side To Reduce Download Times And Storage Requirements

The size of and length of time taken to generate the data depends upon several factors, including the number of concurrent users between which the server processor is divided, processor speed and memory of the server, number of properties requested by the user (determines number of columns in resultant table) and number of redshifts (determines number of tables produced). Currently, depending upon number of properties requested, a particular run may take up to tens of hours to produce hundreds to thousands of megabytes of data per run.

By reducing the amount of data required to be transferred to the client from the server by using lossless compression (i.e. the complete data may be recovered after compression in its original form without loss or alteration of the data), the amount of storage space required on both the client and the server could be reduced.

It would, however, increase the processing time required per run in order for compression algorithms to be executed on the data, the burden for which would be shared between concurrent users. Therefore it should be determined whether the saving in download time for the user outweighs the extra processing time required for compression; this could only be sensibly determined by first implementing, and then testing, compression algorithms in e-Galform. In addition, the storage space for the full uncompressed file would still be required, at least for a brief period of time after the run, as the compressed data would have to be formed from the stored, uncompressed data before being removed from the server. A sensible set of classes to enable the gzip compression of e-Galform data is available at the PHP classes website (http://www.phpclasses.org).

Separate files (ASCII tables and VOTables) could also be archived into a single compressed file, allowing the user to simply download one file instead of downloading single files in succession.

Other Improvements To e-Galform

Other potential improvements might include:

- Allowing registered users to save a snapshot of selected parameters for use as presets, to allow frequently used property settings to be saved for future use. The parameter settings could be saved in a set of tables with reference to the user's registered username. Presets could then be selectable in the property selection interface (see Figure 4.3).
- Create summary documentation to complement e-Galform, with a web link stated within all VOTable output.
- Include N-body information from the Millenium simulation, e.g. similar to the VirtU project – the Virtual Universe, a new development initiated by the ICC {68}. This would enable e-Galform data to be used with Astrogrid facilities such as VisIVO, as spatial information would then be available in e-Galform data.
- The VOTable XML output file should be checked validated against the VOTable schema specification, available from http://us-vo.org/xml/VOTable.dtd.
- Develop PHP scripts to install e-Galform on a server. Currently, the Galform database will need to be transferred using MySQL administration software to a new server. An installation script, creating the MySQL tables and basic data in those tables (such as properties and filters) should be created, which might also configure the web server through batch scripts (executable within a PHP script). The PHP script might also setup the PHP configuration relevant to e-Galform's execution. PHP and web server settings will be discussed in Chapter 11.

11. Server Configuration & Installation

11.1 e-Galform Web Server And PHP Configuration

This chapter outlines aspects of server configuration and e-Galform configuration details that need to be administered before e-Galform will function correctly on new server space. Some of these configuration aspects will require administration by a skilled network administrator.

11.1.1 Apache Server Configuration

e-Galform is designed to run only on a Linux/Apache web server, and not a Windows compatible IIS server. The following server configurations need to be administered before e-Galform will function as required:

- A CRON job must be configured to run *cron.php* in the e-Galform root directory every hour. This may be setup in the following manner:
    - On the server on which e-Galform is located, run 'crontab –e' in a UNIX shell window.
    - Setup the PHP binary/cgi to execute *cron.php*, found in e-Galform's root web directory, every hour.

11.1.2 PHP Configuration

The PHP configuration must also be setup in the following way from default values in order for e-Galform to function correctly:

- PHP safe mode must be turned off to allow e-Galform to execute samplegals.exe on the web server. To do this, set the PHP server variable *safe_mode* to OFF.
- PHP session expiry must also be set to over 300 hours to allow users to save their settings over this period. PHP session expiry lengths are stated in minutes,

with 1440 minutes in 24 hours. To do this, set the PHP server variable *session.gc_maxlifetime* in the php.ini file.

11.1.3 Site Configuration

The site will need to know the locations of various executables and data, which are set via the site's own administrative configuration utilities. Once the server has been setup, login to the administrative menu and click on the 'e-Galform Statistics' option in the administration menu.

Change the **output directory path** to your desired target, which must already be a created directory. The default is **/galform/output**.

Change the **samplegals executable path** to your desired target. The default is **/galform/samplegals.exe**.

11.1.4 Adding New Models

In order to add new model papers to the system, which are then consequently used by samplegals in order to generate ASCII tables, you must use the **Model Editor** options via the administration facility. New binary model data from a pre-generated Galform run must be manually copied to the **dust** directory, found in e-Galform's root web directory.

12. References

[1] *http://www.astro.uiuc.edu/~pmricker/research/codes/flashcosmo/software.html.* The HTML page describes the different software produced for various actions upon FLASH's native data format from the Virgo Consortium's FLASH web site.

[2] *http://icc.dur.ac.uk.* Galform currently does not have a dedicated website, and hence data is not available for download in either an adopted standard for cosmological simulations in the native format produced by *samplegals*, a program that reads Galform's binary data output and converts into a native ASCII fixed width format.

[3] http://vo.iucaa.ernet.in/galics/main_frames.php?dir=database. The GalICS website indicates that the database is available in MySQL database tables, which may be queried from the GalICS website. The user, however, must acquaint himself with the structure of the database tables and the meaning of the data parameters they contain before they can make meaningful use of the MySQL query facility on the GalICS website.

[4] *http://science.slashdot.org/article.pl?sid=05/07/06/0311220&from=rss. Slashdot* computing magazine article, July 6th 2005 *Scientists Complete Universe Millennium Simulation* describes a brief summary of the project's progression and content of scientific value.

[5] *http://en.wikipedia.org/wiki/Moore's_law.* The article describes Moore's Law of processor transistor density (see Appendix B for graph). As an interesting point, processor manufacturing has almost been reached the level of component scaling such that transistors are becoming so small in size, that quantum tunnelling between transistors on the silicon wafer is beginning to produce sufficient uncertainty in the state of the transistor so as to make them unusable. Interference between the wavefunctions associated with the potential is sufficient to destroy the integrity of the on or off state.

[6] *http://en.wikipedia.org/wiki/World_wide_web. Wikipedia* article 'World Wide Web', section *How The Web Works* describes the advent of the internet as pioneered by Tim Berners Lee, whom coined the original term 'World Wide Web'.

[7] *http://en.wikipedia.org/wiki/JANET.* The article describes the history of the JANET communications network, in particular highlighting its high speed in comparison to domestically available internet connections during the 1990s.

[8] *http://www.w3.org/TR/REC-html32.* Official W3 documentation defining the HTML 3.2 standard.

[9] *http://www.php.net.* This is the official PHP site, where the user can download the latest source code for PHP and the latest binaries for a variety of platforms.
[10] *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/activeservpages.asp.* Microsoft Developer Network (MSDN) is an online resource to help Microsoft developers create programs using Microsoft development products.
[11] *http://en.wikipedia.org/wiki/Microsoft_Windows#Widespread_Usage.* A *Network Computing* magazine poll indicated that some 90% of organisations used Microsoft's operating systems.
[12] *PHP and MySQL Web Development, Second Edition*, Welling-Thomson, Indianapolis, 2003. Page 178. Both figures adapted from figures 7-8 and 7-9 respectively.
[13] *http://www.google.com/corporate/history.html.* The page describes Google's rapid growth from 1999 onwards, including the rapid increase in the number of searches requested through its online web search.
[14] *PHP and MySQL Web Development, Second Edition*, Welling-Thomson, Indianapolis, 2003. Chapter 22 describes implementing scalable, maintainable PHP and MySQL code for large-scale projects, including issues of administration and user authentication.
[15] *http://www.w3.org/XML/.* The W3 (WWW) website gives the official definition of XML format.
[16] *http://en.wikipedia.org/wiki/Document_Type_Definition.* A Document Type Definition should be written for all XML formats where the intention is for the format to be adopted by others.
[17] *http://www.ebi.ac.uk/cgi-bin/emblfetch.* The page (called 'dbfetch') allows the user to download specified amino acid sequences via a simple online form.
[18] *http://www.mysql.com/why-mysql/marketshare/.* Above MySQL are the two most popular commercial relational database systems, Microsoft SQL Server (first) and Oracle (second). It should also be noted that MySQL is commercially licensed for profit seeking utility.
[19] *http://www.mysql.com/support/.* MySQL AB (the owner of MySQL database technologies) at time of writing offers MySQL in the form of *Community* and *Enterprise* edition. *Enterprise* edition offers personal technical support and is recommended for government agencies.

[20] Cole, Shaun; Lacey, Cedric G.; Baugh, Carlton M.; Frenk, Carlos S. 2000,MNRAS,319,168

[21] Le Delliou, M.; Lacey, C.; Baugh, C. M.; Guiderdoni, B.; Bacon, R.; Courtois, H.; Sousbie, T.; Morris, S. L. 2005, MNRAS, 357, 11

[22] Katz, N.; Hernquist, L.; Weinberg, D. H., 1992; ApJ, 399, 109

[23] Evrard, Summers & Davis; 1994; ApJ, 422, 11

[24] Katz, N; Weinberg, D. H.; Hernquist, L., 1996; ApJS, 105, 19

[25] Pearce, F. R.; Jenkins, A.; Frenk, C. S.; Colberg, J. M.; White, S. D. M.; Thomas, P. A.; Couchman, H. M. P.; Peacock, J. A.; Efstathiou, G., 1999; ApJ, 521, 99

[26] Springel, V.; Hernquist, L, 2003; IAUS, 208, 273

[27] Hatton, Steve; Devriendt, Julien E. G.; Ninin, Stéphane; Bouchet, François R.; Guiderdoni, Bruno; Vibert, Didier, 2003; MNRAS, 343, 75

[28] Lacey, Cedric; Cole, Shaun, 1993; MNRAS, 262, 627

[29] Cole, Shaun; Lacey, Cedric, 1996; MNRAS, 281, 716

[30] Navarro, Julio F.; Frenk, Carlos S.; White, Simon D. M., 1997; ApJ, 490, 493

[31] Bruzual A., Gustavo; Charlot, Stephane, 1993; ApJ, 405, 538

[32] The GalICS website may be found at http://galics.iap.fr/.

[33] Helly, J. C.; Cole, S.; Frenk, C. S.; Baugh, C. M.; Benson, A.; Lacey, C.; Pearce, F. R., 2003; MNRAS, 338, 913

[34] Springel, V.; Hernquist, L., 2003; IAUS, 208, 273

[35] Leitherer, C.; Heckman, T. M., 1995; ApJS, 96, 9

[36] Official site *http://www.mysql.com/*, where latest binaries are available for download, with free administration software.

[37] See *http://msdn.microsoft.com/visualc/*. Microsoft Visual C++ is now available in a free, limited edition called Visual C++ Express Edition.

[38] See *http://gcc.gnu.org/*. GNU Compiler Collection (GCC) is available under the GNU public license.

[39] See *Programming Windows, 5th Edition* by Charles Petzold, Page 8. ISBN 1-57231-995-X

[40] See *Inside Visual C++, 4th Edition* by David J. Kruglinski, Page 861. ISBN 1-57231-565-2.

[41] *http://en.wikipedia.org/wiki/Java_programming_language* contains a referenced article on the basics of Java bytecode and JVMs.

[42] See *http://quickstarts.asp.net/QuickStartv20/aspnet/doc/ctrlref/data/gridview.aspx* for an example of the GridView object, which can greatly simplify administration facilities.

[43] See *PHP And MySQL Web Development* by Luke Welling And Laura Thompson, Page 207, ISBN 0-672-32525-X.

[44] See *Inside Visual C++, 4th Editing* by David J. Kruglinski, Page 787. ISBN 1-57231-565-2.

[45] See *PHP And MySQL Web Development* by Luke Welling And Laura Thompson, Page 414, ISBN 0-672-32525-X.

[46] See *http://activex.microsoft.com/* for examples of Microsoft ActiveX components.

[47] See *PHP And MySQL Web Development* by Luke Welling And Laura Thompson, Page 416, ISBN 0-672-32525-X.

[48] See *http://en.wikipedia.org/wiki/Cascading_Style_Sheets*, section on history of CSS.

[49] See *http://www.w3schools.com/css/css_reference.asp*

[50] See *http://www.w3schools.com/php/php_post.asp*; this article describes the reading of POST variables in PHP.

[51] See *PHP And MySQL Web Development* by Luke Welling And Laura Thompson, Page 421 (Implementing Authentication With Session Control), ISBN 0-672-32525-X.

[52] Ibid. See Chapter 31 (Connecting to Web Services With XML And SOAP).

[53] *http://www.star.bris.ac.uk/~mbt/topcat/*. TopCat is a project managed by the StarLink team, and can produce graphical plots from VOTable format data.

[54] *http://www.us-vo.org/VOTable/*. The latest documentation describing VOTable format is available at the National Virtual Observatory website.

[55] *http://www.astrogrid.org/* is Astrogrid's website.

[56] *http://en.wikipedia.org/wiki/Grid_computing*. The article cites various books and academic journals about the concept of grid computing.

[57] *http://archive.eso.org/archive/hst/* Home page of the Hubble Space Telescope Archive.

[58] *http://sohowww.nascom.nasa.gov/data/*. Home page of the Solar and Heliospheric Observatory (SOHO).

[59] *http://galics.cosmologie.fr/main_frames.php?dir=main*. GalICS has its own dedicated website.

[60] *http://www.nesc.ac.uk/* The National e-Science Centre website.

[61] *http://www.cambsenterprise.co.uk/index_peter.htm.* Based upon the "Marketing Your Small Business" course available from Peterborough Enterprise Programme, a pseudo-government agency setup to help entrepreneurs to setup businesses.
[62] *http://dev.mysql.com/doc/refman/5.0/en/table-size.html.* Note that Linux operating systems can store up to 2GB per file, and hence is limited to this size per table (the data for which is stored in a file).
[63] *http://www.ivoa.net/pub/info/.* Page contains information about the International Virtual Observatory Alliance (IVOA).
[64] *http://software.astrogrid.org/developerdocs/index.html.* The AstroGrid website contains an online reference for developers.
[65] *http://www.sdss.org/* - Sloan Digital Sky Survey website.
[66] *http://www.feedreader.com/.* FeedReader is a free to download RSS feed subscription tool.
[67] *http://www.microsoft.com/windows/ie/default.mspx.*
[68] An executive summary proposing the VirtU project may be found at *star-www.dur.ac.uk/~csf/virtU/virtU-final.pdf.*
[69] *http://hydra.mcmaster.ca/hydra/index.html* is the main website for the Hydra N-body cosmological simulation.

Appendix A: A Selection Of Scripts From e-Galform

All scripts are programmed using PHP, MySQL, HTML, XML, and CSS.

generate.php

This script is responsible for executing samplegals, and according to requirements generating ASCII tables, VOTables or both and presenting them for download using an easy to use form.

```
<?php
session_start();
include "globals.php";
do_html_header('Generating Your Tables', 'e-Galform Table
Generation');

function rmdirr($dir) {
   if($objs = glob($dir."/*")){
       foreach($objs as $obj) {
           is_dir($obj)? rmdirr($obj) : unlink($obj);
       }
   }
   rmdir($dir);
}


function splitString( $str, $numChars ) {
    $strArray = array();

    while ( strlen( $str ) > 0 ) {
        array_push( $strArray, substr( $str, 0, $numChars ) );
        $str = substr( $str, $numChars );
    }
    return $strArray;
}

function listArray( $array )
{
    for($x=0; $x<count($array); $x++)
    {
        echo '|'.$array[$x].'|<BR>';
    }
}

function trimArray( $array )
{
    for($x=0; $x<count($array); $x++)
    {
        trim($array[$x]);
    }
}

$redshifts = array('R0','R05','R10','R20','R30','R40','R50','R60');
```

```
do_dbconnect($dbcnx);

$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'Output Directory\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row = mysql_fetch_array( $result );
$outdir = $row['StatisticValue'];
?>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
   <TR>
     <TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px"
BGCOLOR="#FFFFFF">
<?php
// Kill everything in output directory
rmdirr( $_SERVER['DOCUMENT_ROOT'].'/'.$outdir.'/'.session_id() );
mkdir( $_SERVER['DOCUMENT_ROOT'].'/'.$outdir.'/'.session_id() );

// For each redshift generate ASCII tables - this is required even if
only VOTables are required, as
// VOTables are generated from ASCII tables
for( $x=0; $x<count($redshifts);$x++)
{
  if($_POST[$redshifts[$x]] != '')
  {
      exec($_POST[$redshifts[$x]],$output);
  }
}

// ** This script generates VOTables **
if( $_POST['votable'] != '' )
{
   // Open output data table (this would be normally obtained after
running SampleGals)
   // Find the tables used by the current run
   for( $x=0; $x<count($redshifts);$x++)
   {
      if( $_POST[$redshifts[$x]] != '' )
      {
          $redshift = return_redshift($redshifts[$x]);
          $filename =
$_SERVER['DOCUMENT_ROOT'].'/'.$outdir.'/'.session_id().'/table_z'.$red
shift;
          $linecount = 0;

          $fp = fopen( $filename, 'r');

          if(!$fp)
          {
              echo 'fopen() could not open '.$filename;
              exit();
          }

          $check = FALSE;

          while(!feof($fp))
```

```
        {
            // Look at the beginning of the line - we want to find
out the last line in the file with '#'
            // This is the column header line which gives the
parameter labels
            // The next line is the beginning of the numeric data
            $linecount = $linecount + 1;
            $line = fgets( $fp );

            if((substr( $line, 0, 1 )!='#') && $check == FALSE)
            {
                $colstart = $linecount -1;
                $check = TRUE;
            }
        }

        fclose($fp);

        /* VOTable header */
        // This will be written to a file 'galform.xml' for now
        $fp2 = fopen(
$_SERVER['DOCUMENT_ROOT'].'/'.$outdir.'/'.session_id().'/galform_'.$re
dshift.'.xml', 'w' );
        $fp = fopen($filename, 'r');

        $lineno = 0;

        fwrite( $fp2, '<?xml version="1.0"?>'."\n");
        fwrite( $fp2, '<!DOCTYPE VOTABLE SYSTEM "http://us-
vo.org/xml/VOTable.dtd">'."\n".'<VOTABLE
version="1.0">'."\n".'<RESOURCE ID="e-Galform VOTable
Output">'."\n".'<TABLE name="Galform Galaxy
Properties">'."\n".'<DESCRIPTION>sample_gals.exe output VOTable
automatically generated by e-Galform on '.date('dS F
Y').'</DESCRIPTION>'."\n".'<LINK content-role="doc" title="e-Galform
documentation" href="http://www.dur.ac.uk/egalform"/>'."\n");

        while( $lineno != $linecount )
        {
            $lineno = $lineno + 1;
            $line = fgets($fp );

            // Output parameters -
properties/filters/emissionlines/dustemission
            // $colstart contains line on which header begins, so
only start processing once this line has been reached

            if($lineno == $colstart)
            {
                // Search for all Storage entries according to
current session ID
                // All values are single point precision, which is
VOTable language is called a 'float'
                // Order by props - emission lines - dust - filters

                // Properties (called parameters in database)
                $result = mysql_query('SELECT * FROM ParameterStore
WHERE ParameterStoreSessID=\''.session_id().'\'');

                if(!$result)
                {
```

```
                    echo 'Error: Could not perform MySQL query -
'.mysql_error();
                    exit();
               }

               while( $row = mysql_fetch_array( $result ) )
               {
                    $result2 = mysql_query('SELECT * FROM
Parameters WHERE ParameterID='.$row['ParameterID']);

                    if(!$result2)
                    {
                         echo 'Error: Could not perform MySQL query
- '.mysql_error();
                         exit();
                    }

                    $row2 = mysql_fetch_array( $result2 );

                    // Write property FIELD
                    fwrite( $fp2, '<FIELD
ID="'.$row2['ParameterIDText'].'" datatype="float">'."\n" );
                    fwrite( $fp2,
'<DESCRIPTION>'.$row2['ParameterDescription'].'</DESCRIPTION>' );
                    fwrite( $fp2, "\n</FIELD>\n" );
               }

               // Emission Lines
               $result = mysql_query('SELECT * FROM
EmissionLineStore WHERE
EmissionLineStoreSessionID=\''.session_id().'\'');

               if(!$result)
               {
                    echo 'Error: Could not perform MySQL query -
'.mysql_error();
                    exit();
               }

               while( $row = mysql_fetch_array( $result ) )
               {
                     // Write emission line FIELD
                     fwrite( $fp2, '<FIELD
ID="'.$row['EmissionLineStoreVariableText'].'" datatype="float">\n' );
                     fwrite( $fp2,
'<DESCRIPTION>'.$row['EmissionLineStoreDescription'].'</DESCRIPTION>'
);
                     fwrite( $fp2, "\n</FIELD>\n" );
               }

               // Dust properties (called parameters in database)
               $result = mysql_query('SELECT * FROM
DustParameterStore WHERE
DustParameterStoreSessionID=\''.session_id().'\'');

               if(!$result)
               {
                     echo 'Error: Could not perform MySQL query -
'.mysql_error();
                     exit();
               }
```

```
                while( $row = mysql_fetch_array( $result ) )
                {
                        fwrite( $fp2, '<FIELD
ID="'.$row['DustParameterStoreString'].'" datatype="float">'."\n" );
                        fwrite( $fp2,
'<DESCRIPTION>'.$row['DustParameterStoreDescription'].'</DESCRIPTION>'
);
                        fwrite( $fp2, "\n</FIELD>\n" );
                }

                // Filters
                $result = mysql_query('SELECT * FROM
FilterParameterStore WHERE
FilterParameterStoreSessionID=\''.session_id().'\'');

                if(!$result)
                {
                        echo 'Error: Could not perform MySQL query -
'.mysql_error();
                        exit();
                }

                while( $row = mysql_fetch_array( $result ) )
                {
                     fwrite( $fp2, '<FIELD
ID="'.$row['FilterParameterStoreString'].'" datatype="float">'."\n" );
                        fwrite( $fp2,
'<DESCRIPTION>'.$row['FilterParameterStoreDescription'].'</DESCRIPTION
>' );
                        fwrite( $fp2, "\n</FIELD>\n" );
                }

                // Parameter writing COMPLETE
            }
            // if($colstart) ends

            // Line 8 onwards
            // Columns of width determined by number of decimal
places - 2 spaces between.
            // 1 space at the beginning for negative symbols
            if(($lineno >= ($colstart+1)) && ($lineno !=
$linecount))
            {
                if( $lineno == ($colstart+1)) fwrite($fp2,
"<DATA>\n<TABLEDATA>\n");

                $numbers = explode( '  ', $line );
                trimArray($numbers);

                fwrite( $fp2, '<TR>' );

                for($x=0;$x<count($numbers);$x++)
                {
                    fwrite( $fp2, '<TD>' );
                    fwrite( $fp2, ltrim($numbers[$x], ' ') );
                    fwrite( $fp2, '</TD>');
                }

                fwrite( $fp2, "</TR>\n" );
            }
```

```
            }
            // while(!feof) comes to an end

            fwrite( $fp2,
"</TABLEDATA>\n</DATA>\n</TABLE>\n</RESOURCE>\n</VOTABLE>\n" );
            fclose($fp2);
            fclose($fp);
        }
        // if($redshift) comes to an end
    }
    // for statement comes to an end
}
// if(votable) comes to an end
echo 'end here';
// Post links to download
$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'Output Directory\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row = mysql_fetch_array( $result );
$outdir = $row['StatisticValue'];

// Generate Output According To Requirements
if( $_POST['votable'] == '' )
{
  // ** ASCII Tables Only **
?>
<FONT>
   Your samplegals ASCII tables are now ready for download. To
download, write click on the links below and click 'Save As...' (or
similar, depending on your browser). <b>Please note you should
download as soon as possible, as your files will automatically be
removed within 24-48 hours</b>.<BR><BR>
</FONT>
<?php
for($x = 0;$x<count($redshifts);$x++)
{
  if($_POST[$redshifts[$x]] != '')
  {
    echo '<A
HREF="'.$outdir.'/'.session_id().'/table_z'.return_redshift(
$redshifts[$x] ).'">ASCII Table Redshift '.return_redshift(
$redshifts[$x] ).'</A><BR>';
  }
}

}
elseif( ( $_POST['asciitable'] == '' ) && ( $_POST['votable'] != '' )
)
{
?>
<FONT>
   Your VOTables are now ready for download. To download, write click
on the links below and click 'Save As...' (or similar, depending on
your browser). <b>Please note you should download as soon as possible,
```

```
as your files will automatically be removed within 24-48
hours</b>.<BR><BR>
</FONT>
<?php
  // ** VOTables Only **
 for($x=0;$x<count($redshifts);$x++)
 {
  if($_POST[$redshifts[$x]] != '')
  {
     echo '<A
HREF="'.$outdir.'/'.session_id().'/galform_'.return_redshift(
$redshifts[$x] ).'.xml">VOTable Redshift '.return_redshift(
$redshifts[$x] ).'</A><BR>';
  }
  }
}
else
{
?>
<FONT>
   Your ASCII and VOTables are now ready for download. To download,
write click on the links below and click 'Save As...' (or similar,
depending on your browser). <b>Please note you should download as soon
as possible, as your files will automatically be removed within 24-48
hours</b>.<BR><BR>
<?php
  // ** VOTables AND ASCII Tables **
  for($x=0;$x<count($redshifts);$x++)
  {
  if($_POST[$redshifts[$x]] != '')
  {
     echo '<A
HREF="'.$outdir.'/'.session_id().'/table_z'.return_redshift(
$redshifts[$x] ).'">ASCII Table Redshift '.return_redshift(
$redshifts[$x] ).'</A> - <A
HREF="'.$outdir.'/'.session_id().'/galform_'.return_redshift(
$redshifts[$x] ).'.xml">VOTable Redshift '.return_redshift(
$redshifts[$x] ).'</A><BR>';
  }
  }
?>
</FONT>
<?php
}
?>
<FONT><BR>Click <A HREF="index.php">here</A> when you have finished
downloading to return to the front page.</FONT>
<?php
     do_html_footer();
?>
```

statistics.php

This script is responsible for reporting in the administrative tools the statistics related to e-Galform usage, for example number of simulation runs, and number of front page hits.

```
<?php
include 'globals.php';
do_dbconnect($dbcnx);
```

```
do_html_header('e-Galform User Statistics');
breadline('<A HREF="system.php" CLASS="breadlink">Administrative
Menu</A> > Statistical Analysis');
?>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px"
BGCOLOR="#FFFFFF">
<FONT>
The following are summary statistics of e-Galform use:<BR><BR>
<?php
$result = mysql_query('SELECT LogTime,RegUserID FROM Logs ORDER BY
LogID DESC LIMIT 1');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row = mysql_fetch_array($result);
echo 'Last Use: <b>'.date('dS F Y H:m:s');

$result2 = mysql_query('SELECT RegUserName FROM RegUsers WHERE
RegUserID='.$row['RegUserID']);

if(!$result2)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row2 = mysql_fetch_array( $result2 );
echo '</b> by <b>'.$row2['RegUserName'].'</b><BR><BR>';

$result = mysql_query('SELECT LogTime FROM Logs');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$count = mysql_num_rows($result);

echo 'Total Logged Runs: <b>'.$count.'</b><BR>';

$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'Total Front Page Hits\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row = mysql_fetch_array($result);
$hits = $row['StatisticValue'];

echo 'Total Front Page Hits: <b>'.$hits.'</b><BR>';
```

```
$result = mysql_query('SELECT * FROM MailingList');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$mailingcount = mysql_num_rows($result);

$result = mysql_query('SELECT * FROM RegUsers');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$regcount = mysql_num_rows($result);

// Read paths from Statistics database
$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'Output Directory\'');

if(!$result)
{
    echo 'Error: Could not perform MySQL query - '.mysql_error();
    exit();
}

$row = mysql_fetch_array( $result );
$outdir = $row['StatisticValue'];

$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'SamplegalsPath\'');

if(!$result)
{
    echo 'Error: Could not perform MySQL query - '.mysql_error();
    exit();
}

$row = mysql_fetch_array( $result );
$samplegals = $row['StatisticValue'];

echo 'Total Mailing List Entries: <b>'.$mailingcount.'</b><BR>';
echo 'Total Registered Users: <b>'.$regcount.'</b><BR><BR>';
echo '<b>Ratio Analysis</b><BR><BR>';
echo '<i>Logged Uses:Front Page Hits</i> <b>'.number_format(
($count/$hits)*100, 0, '.', ',').'%</b><BR>This will give a rough
indication of the number of registered users who visit the site to
non-registered users. The lower the number, the more visitors are
visiting without registering and/or using the e-Galform
facitilies.<BR><BR>';
echo '<i>Registered Users:Mailing List Entries</i> <b>'.number_format(
($regcount/$mailingcount)*100,0,'.',',').'%</b><BR>This will give an
indication of the success of the e-Galform marketing campaign via e-
mailed news articles. The lower the number, the lower the number of
the people on the mailing list consequently registering on the
system.<BR><BR>';
echo '<b>e-Galform Paths Setup</b><BR><BR>';
```

```
echo '<FORM METHOD="POST" ACTION="outchange.php"
STYLE="margin:0px">Output Directory Path:
<b>'.$_SERVER['DOCUMENT_ROOT'].'/</b><INPUT TYPE="text"
NAME="txt_Outdir" VALUE="'.$outdir.'"/> <INPUT TYPE="submit"
STYLE="font-size:8pt" VALUE="Update >>"/> e.g.
<b>galform/output</b></FORM><BR>';
echo '<FORM METHOD="POST" ACTION="smpchange.php"
STYLE="margin:0px">Samplegals Path:
<b>'.$_SERVER['DOCUMENT_ROOT'].'/</b><INPUT TYPE="text"
NAME="txt_Sampath" VALUE="'.$samplegals.'"/> <INPUT TYPE="submit"
STYLE="font-size:8pt" VALUE="Update >>"/> e.g.
<b>galform/sample_gals.exe</b></FORM>';
?>
</FONT>
</TD>
</TR>
</TABLE>
<?php
do_html_footer();
?>
```

advanced.php

This script is responsible for generating and displaying the user interface stating the input and output properties of the e-Galform simulation run.

```
<?php
session_start();
include "globals.php";
do_dbconnect($dbcnx);

do_html_header('eGalform :: Advanced Settings', 'eGalform :: Advanced
Settings');
?>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#000000">
Saved settings for your current session:<BR><BR>
<FONT STYLE="font-family:courier">
<?php
// Check tables in order and output, matching the current session id
$result = mysql_query('SELECT * FROM ParameterStore WHERE
ParameterStoreSessID=\''.session_id().'\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  $result2 = mysql_query('SELECT * FROM Parameters WHERE
ParameterID='.$row['ParameterID']);

  if(!$result2)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
```

```
      exit();
    }

  $row2 = mysql_fetch_array( $result2 );
  echo 'Output Parameter Selected: '.$row2['ParameterIDText'].'<BR>';
}

$result = mysql_query('SELECT * FROM EmissionLineStore WHERE
EmissionLineStoreSessionID=\''.session_id().'\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo 'Emission Line Output Selected:
'.$row['EmissionLineStoreVariableText'].'<BR>';
}

$result = mysql_query('SELECT * FROM DustParameterStore WHERE
DustParameterStoreSessionID=\''.session_id().'\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo 'Dust Parameter Output Selected:
'.$row['DustParameterStoreString'].'<BR>';
}

$result = mysql_query('SELECT * FROM FilterParameterStore WHERE
FilterParameterStoreSessionID=\''.session_id().'\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ))
{
  echo 'Output Filter Selected:
'.$row['FilterParameterStoreString'].'<BR>';
}
?>
<BR>
</FONT>
</FONT>
</TD>
</TR>
</TABLE>
```

```
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" STYLE="border-
left:1px solid black;border-right:1px solid black;border-top:1px solid
black">
   <TR>
      <TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px"
BGCOLOR="#BBBBBB">
         <FONT SIZE="2" COLOR="#FFFFFF">
            <b>e-Galform :: Advanced Settings</b>
         </FONT>
      </TD>
   </TR>
   <TR>
      <TD WIDTH="742" STYLE="padding-left:4px;padding-
right:4px;padding-top:4px;padding-bottom:4px">
         <FONT>
            Use this form if you wish to specify the parameters you
wish to run through e-Galform in finer detail. When you are finished
selecting appropriate parameters, click the 'Generate >>' button below
to commence the run. If settings are already present from a previous
session that you wish to clear (settings are generally saved for a few
hours before automatic reset), click 'Clear Session Settings' at the
bottom of this page.
         </FONT>
      </TD>
   </TR>
</TABLE>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" STYLE="border-
left:1px solid black;border-right:1px solid black">
   <TR>
      <TD WIDTH="375" VALIGN="TOP">

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
   <TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(1) Select General
Output Parameters</b></FONT>
</TD>
</TR>
</TABLE>

<FORM NAME="parameter" STYLE="margin:0px" METHOD="POST"
ACTION="addparam.php">
<DIV STYLE="padding-left:5px;padding-top:4px"><FONT>
Specify Parameter To Output<BR>
<SELECT STYLE="font-size:8pt" NAME="chk_PropertyOut">
<?php
$result = mysql_query( 'SELECT * FROM Parameters' );

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo '<OPTION
NAME="'.$row['ParameterID'].'">'.$row['ParameterDescription'].' [<DIV
```

```
STYLE="font-color:#ff0000;font-
style:bold">'.$row['ParameterIDText'].'</DIV>]</OPTION>';
}
?>
</SELECT><BR>
<DIV ALIGN="RIGHT" STYLE="padding-right:5px">
Click 'Add >>' to add output parameter <INPUT TYPE="submit"
NAME="submit" STYLE="font-size:9px" VALUE="Add >>"/>
</DIV>
<BR>
</DIV>
</FORM>
</FORM>

       </TD>
       <TD WIDTH="375" VALIGN="TOP">


<FORM ACTION="addemissionline.php" METHOD="POST" STYLE="margin:0px">
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(2) Select Output
Emission Lines</b></FONT>
</TD>
</TR>
</TABLE>

<FONT>
<DIV STYLE="padding-left:5px;padding-bottom:5px">
Unit:
<SELECT NAME="chk_Unit">
<OPTION NAME="L">Luminosity in 10^40 h^-2 erg/s</OPTION>
<OPTION NAME="EW">Equivalent Width in A</OPTION>
</SELECT><BR>
Emission Component:
<SELECT NAME="chk_EmissionType">
<OPTION NAME="tot">Galaxy Total</OPTION>
<OPTION NAME="disk">Galaxy Disk</OPTION>
<OPTION NAME="bulge">Galaxy Bulge</OPTION>
</SELECT><BR>
Emission Line:
<SELECT NAME="chk_Emission">
<?php
$result = mysql_query('SELECT * FROM EmissionLines');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo '<OPTION
NAME="'.$row['EmissionLineID'].'">'.$row['EmissionLineTitle'].'</OPTIO
N>';
}
?>
</SELECT><BR>
```

```
<INPUT TYPE="checkbox" NAME="chk_DustExtEmission" CHECKED/>Include
Dust Extinction
<DIV ALIGN="RIGHT" STYLE="padding-right:4px">Click 'Add >>' to add
emission line property <INPUT TYPE="submit" NAME="submit" STYLE="font-
size:9px" VALUE="Add >>"/></DIV>
</DIV>
</FONT>
</FORM>

      </TD>
   </TR>
</TABLE>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" STYLE="border-
left:1px solid black;border-right:1px solid black">
   <TR>
      <TD WIDTH="375" VALIGN="TOP">

<FORM ACTION="adddustparam.php" METHOD="POST">
<FONT>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(3) Specify Dust
Parameters</b></FONT>
</TD>
</TR>
</TABLE>

<DIV STYLE="padding-left:4px;padding-bottom:4px">
Total Dust Luminosity: <INPUT TYPE="text" SIZE="3"
NAME="txt_DustLumTot"/> (micro m)<BR>
Diffuse Dust Luminosity: <INPUT TYPE="text" SIZE="3"
NAME="txt_DustLumDif"/> (micro m)<BR>
Cloud Dust Luminsity: <INPUT TYPE="text" SIZE="3"
NAME="txt_DustLumCloud"/> (micro m)<BR>
Reference Frame: <SELECT NAME="chk_DustFrame">
<OPTION NAME="Rest">Rest</OPTION>
<OPTION NAME="Observer">Observer</OPTION>
</SELECT>
<DIV ALIGN="RIGHT" STYLE="padding-right:4px">Click 'Add >>' to add
dust emission property <INPUT TYPE="submit" NAME="submit" STYLE="font-
size:9px" VALUE="Add >>"/></DIV>
</DIV>
</FONT>
</FORM>

      </TD>
      <TD WIDTH="375" VALIGN="TOP">
<FORM ACTION="addfilter.php" METHOD="POST">
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(4) Select Output
Filters</b></FONT>
</TD>
</TR>
</TABLE>
```

```
<DIV STYLE="padding-left:5px;padding-bottom:5px">
<FONT>
Filter Band:
<SELECT NAME="chk_FilterBand">
<?php
$result = mysql_query('SELECT * FROM Filters');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

while( $row = mysql_fetch_array( $result ) )
{
  echo '<OPTION
NAME="'.$row['FilterID'].'">'.$row['FilterTitle'].'</OPTION>';
}
?>
</SELECT>
Frame:
<SELECT NAME="chk_FilterFrame">
<OPTION NAME="r" SELECTED>Rest Frame</OPTION>
<OPTION NAME="o">Observer Frame</OPTION>
</SELECT><BR>
Component:
<SELECT NAME="chk_FilterComp">
<OPTION NAME="total">Total</OPTION>
<OPTION NAME="bulge">Bulge</OPTION>
<OPTION NAME="disk">Disk</OPTION>
</SELECT><BR>
<INPUT TYPE="checkbox" NAME="chk_FilterDustExtinction" CHECKED/>
Include Dust Extinction<BR>
<INPUT TYPE="checkbox" NAME="chk_FilterFaceOn"/> Face On Extinction-
Corrected Magnitude
<DIV ALIGN="RIGHT" STYLE="padding-right:4px">Click 'Add >>' to add
output filter <INPUT TYPE="submit" NAME="submit" STYLE="font-size:9px"
VALUE="Add >>"/></DIV>
</DIV>
</FONT>
</FORM>


      </TD>
   </TR>
</TABLE>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" STYLE="border-
left:1px solid black;border-right:1px solid black">
   <TR>
      <TD WIDTH="375" VALIGN="TOP">

<FORM NAME="galform" STYLE="margin:0px" METHOD="POST"
ACTION="start.php">
<FONT>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(5) Select Parameter
Range Or Galaxy Output Type</b></FONT>
```

```
</TD>
</TR>
</TABLE>

<DIV STYLE="padding-left:5px;padding-bottom:4px">
<INPUT TYPE="radio" NAME="chk_Parameters" VALUE="NoParam" CHECKED/>No
Parameter Range Selected<BR>
<INPUT TYPE="radio" NAME="chk_Parameters" VALUE="RangeParam"/>Specify
Parameter Range (1 only):
<SELECT STYLE="font-size:8pt" NAME="chk_Property">
<?php
$result = mysql_query('SELECT * FROM Parameters');

if(!$result)
{
    echo 'Error: Could not perform MySQL query - '.mysql_error();
    exit();
}

while( $row = mysql_fetch_array( $result ) )
{
    echo '<OPTION STYLE="font-size:8pt"
NAME="'.$row['ParameterID'].'">'.$row['ParameterDescription'].'
['.$row['ParameterIDText'].']</OPTION>';
}
?>
</SELECT><BR><INPUT TYPE="text" NAME="txt_ParaFrom" SIZE="3"/> to
<INPUT TYPE="text" NAME="txt_ParaTo" SIZE="3"/> <BR>
<INPUT TYPE="radio" NAME="chk_Parameters" VALUE="isat"/>
Select Galaxy Type
<SELECT NAME="isat">
<OPTION NAME="0">Satellite Galaxies</OPTION>
<OPTION NAME="1">Central Galaxies</OPTION>
</SELECT><BR>
</DIV>

      </TD>
     <TD WIDTH="375" VALIGN="TOP">

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="366" BGCOLOR="#BBBBBB" STYLE="padding-left:2px;padding-
right:2px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(6) Other
Options</b></FONT>
</TD>
</TR>
</TABLE>

<DIV STYLE="padding-left:5px;padding-bottom:4px">
<FONT>

Output Redshifts:<BR>
<INPUT TYPE="checkbox" NAME="0" CHECKED/> 0
<INPUT TYPE="checkbox" NAME="0.5"/> 0.5
<INPUT TYPE="checkbox" NAME="1.0"/> 1.0
<INPUT TYPE="checkbox" NAME="2.0"/> 2.0
<INPUT TYPE="checkbox" NAME="3.0"/> 3.0
<INPUT TYPE="checkbox" NAME="4.0"/> 4.0
<INPUT TYPE="checkbox" NAME="5.0"/> 5.0
<INPUT TYPE="checkbox" NAME="6.0"/> 6.0
```

```
<BR>

Magnitude System: <SELECT NAME="chk_Mag">
<OPTION NAME="AB" SELECTED>AB</OPTION>
<OPTION NAME="Vega">Vega</OPTION>
</SELECT><BR>
Volume: <INPUT TYPE="text" SIZE="3" NAME="txt_Vol" VALUE="0"/> (Leave
at 0 for no restriction)<BR>
Output table <INPUT NAME="outtable" STYLE="margin-top:10px"
TYPE="checkbox" VALUE="V" CHECKED> VOTable <INPUT NAME="outtable"
TYPE="checkbox" VALUE="G"> GALFORM
</FONT>
</DIV>

      </TD>
   </TR>
</TABLE>

<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" STYLE="border-
left:1px solid black;border-right:1px solid black;border-bottom:1px
solid black">
<TR>
<TD WIDTH="742" BGCOLOR="#BBBBBB" STYLE="padding-left:4px;padding-
right:4px">
<FONT SIZE="2" FACE="Tahoma" COLOR="#FFFFFF"><b>(7) Generate Output
Table(s)</b></FONT>
</TD>
</TR>
</TABLE>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="371" STYLE="padding-left:4px">
<FONT><A
HREF="http://ganymede.phyast.dur.ac.uk/galform/reset.php"><b>Clear
Session Settings</b></A></FONT>
</TD>
<TD WIDTH="371" STYLE="padding-right:4px">
<DIV ALIGN="RIGHT" STYLE="padding-top:3px;padding-bottom:3px;padding-
right:3px"><INPUT STYLE="font-size:8pt" TYPE="submit" VALUE="Generate
>>" NAME="submit"/></DIV>
</DIV>
</TD>
</TR>
</TABLE>
</FONT>
</FORM>

</TD>

   </TR>
</TABLE>
<?php
do_html_footer();
?>
```

Start.php

This script is responsible for forming the execution strings required to generate samplegals format ASCII tables for each redshift selected by the user on the property selection interface (Figure 5.3).

```
<?php
session_start();
include "globals.php";
do_dbconnect($dbcnx);

// return_paramid :: returns the parameterid of the property selected
on the front form
// we do this by deriving the paramid from the string returned in the
dropdown menu
function return_paramid( $propertystring )
{
  $result = mysql_query('SELECT * FROM Parameters');

  if(!$result)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

  while($row = mysql_fetch_array( $result ) )
    {
      $check_string = $row['ParameterDescription'].'
['.$row['ParameterIDText'].']';

      if($check_string == $propertystring)
      {
          $paramid = $row['ParameterID'];
          break;
      }
    }

  return $paramid;
}

do_html_header('Ready To Begin', 'Start Using e-Galform');
?>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
    <TR>
       <TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px">
<FORM STYLE="border:none" METHOD="POST" ACTION="generate.php">
<FONT>
<?php
$redshifts = array( 'R0', 'R05', 'R10', 'R20', 'R30', 'R40', 'R50',
'R60' );
$exec = array( '', '', '', '', '', '' );

if($_POST['chk_Model'] == '')
{
  // default to first model if empty model indicator
  $result = mysql_query('SELECT ModelName FROM Models WHERE
ModelID=1');
```

```
  $row = mysql_fetch_array($result);

  $_POST['chk_Model']   = $row['ModelName'];
}

  echo '<b>Thank you for using e-Galform.</b> Your unique session ID
is <b>'.session_id().'</b> (you do not need to write this down) which
will <b>expire within 24 hours</b>. The run will use the
<b>'.$_POST['chk_Model'].'</b> paper.';
?>
<BR><BR>You have chosen to generate data for redshifts
<?php
// Generate the correct execution command line from the various
parameter databases and the INPUT controls on the front page (the
complicated bit)
// sample_gals_mjs.exe (or similar) must be run in a separate instance
for each output redshift. Therefore, we create an $exec string for
each redshift

// Create a directory in the /output directory for this session_id()
$result = mysql_query('SELECT StatisticValue FROM Statistics WHERE
StatisticTitle=\'Output Directory\'');

if(!$result)
{
  echo 'Error: Could not perform MySQL query - '.mysql_error();
  exit();
}

$row = mysql_fetch_array($result);
$outdir = $row['StatisticValue'];

if(!file_exists($_SERVER['DOCUMENT_ROOT'].$outdir.'/'.session_id() ))
{
if(!mkdir( $_SERVER['DOCUMENT_ROOT'].$outdir.'/'.session_id() , 0770
))
{
  echo 'Error creating directory for this session.';
}
}

for( $i=0; $i < count( $redshifts ); $i++ )
{
  if( isset( $_POST[ $redshifts[$i] ] ) )
  {
  $fcloud  = 0.25;
  echo '<b>'.return_redshift( $redshifts[$i] ).'</b>, ';

  // Read in model from model specification
  $result = mysql_query('SELECT ModelDir FROM Models WHERE
ModelTitle=\''.$_POST['chk_Model'].'\'');

  if(!$result)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

  $row = mysql_fetch_array($result);

  // Work out redshift from name of check input
```

```
  switch( $redshifts[$i]  )
  {
  case 'R0':
    $redshift = 0;
    break;
  case 'R05':
    $redshift = 0.5;
    break;
  case 'R10':
    $redshift = 1.0;
    break;
  case 'R20':
    $redshift = 2.0;
    break;
  case 'R30':
    $redshift = 3.0;
    break;
  case 'R40':
    $redshift = 4.0;
    break;
  case 'R50':
    $redshift = 5.0;
    break;
    case 'R60';
    $redshift = 6.0;
    break;
  default:
    $redshift = 0;
  }

  $gfile = $row['ModelDir'].'/z'.$redshift.'/cat';

  // Generate the table name so that it can be identified for this
session at the end of the run
  // All output tables go into the /output directory relative to the
home directory in which this script should be stored
  $table =
$_SERVER['DOCUMENT_ROOT'].$outdir.'/'.session_id().'/table_z'.$redshif
t;
  $upsilon = 1.0;
  $rfacburst = 1.0;
  $isat = '';

  if( isset($_POST[$redshifts[$i]]) )
  {
    // Check the range/isat requirements
    switch ( $_POST['chk_Parameters'] )
    {
    case 'NoParam':
      $range = '';
      break;
    case 'RangeParam':
      $result = mysql_query('SELECT ParameterIDText FROM Parameters
WHERE ParameterID='.return_paramid( $_POST['chk_Property'] ));

      if(!$result)
      {
          echo 'Error: Could not perform MySQL query -
'.mysql_error();
          exit();
      }
```

```
      $row = mysql_fetch_array( $result );
      $range = 'range '.$row['ParameterIDText'].'
'.$_POST['txt_ParaFrom'].' '.$_POST['txt_ParaTo'];
      break;
      case 'isat':
      if( $_POST['isat']=='Satellite Galaxies' ) $isat = 'isat 0';
      else $isat = 'isat 1';
      $range = '';
      break;
      }
   }

  // Determine the volume parameter from selection of one-on-one or
constrained volume
  if( $_POST['chk_Volume'] == '1in1' )
    {
      $vol = 0;
    }
  else
    {
      $vol = $_POST['txt_Vol'];
    }

    $exec[$i] =
$_SERVER['DOCUMENT_ROOT'].'/galform/sample_gals_mjs.exe '.$gfile.'
'.$redshift.' '.$upsilon.' '.$vol.' -9931 '.$_POST['chk_Mag'].' dust
dust/dust_MW_hz1.0.dat 0 '.$rfacburst.' '.$fcloud.' file '.$table.'
'.$range.' props '.$isat;

    // Now read the various database tables for this session and add
the properties to the $exec[] string
    //
    // We have
    //
    // (1) Output properties in ParameterStore table
    // (2) Output emission lines in EmissionLineStore table
    // (3) Output dust parameters in DustParameterStore table
    // (4) Output filters in FilterParameterStore table

    $props = '';
    $result = mysql_query('SELECT * FROM ParameterStore WHERE
ParameterStoreSessID=\''.session_id().'\'');

    if(!$result)
      {
         echo 'Error: Could not perform MySQL query - '.mysql_error();
         exit();
      }

    while( $row = mysql_fetch_array( $result ) )
    {
      $result2 = mysql_query('SELECT * FROM Parameters WHERE
ParameterID='.$row['ParameterID']);

      if(!$result2)
      {
          echo 'Error: Could not perform MySQL query -
'.mysql_error();
          exit();
      }
```

```
      $row2 = mysql_fetch_array( $result2 );
      $props .= $row2['ParameterIDText'].' ';
    }

    $result = mysql_query('SELECT * FROM EmissionLineStore WHERE
EmissionLineStoreSessionID=\''.session_id().'\'');

    if(!$result)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

    while( $row = mysql_fetch_array( $result ) )
    {
      $props .= $row['EmissionLineStoreVariableText'].' ';
    }

    $result = mysql_query('SELECT * FROM DustParameterStore WHERE
DustParameterStoreSessionID=\''.session_id().'\'');

    if(!$result)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

    while( $row = mysql_fetch_array( $result ) )
    {
      $props .= $row['DustParameterStoreString'].' ';
    }

    $result = mysql_query('SELECT * FROM FilterParameterStore WHERE
FilterParameterStoreSessionID=\''.session_id().'\'');

    if(!$result)
    {
      echo 'Error: Could not perform MySQL query - '.mysql_error();
      exit();
    }

    while( $row = mysql_fetch_array( $result ) )
    {
      $props .= $row['FilterParameterStoreString'].' ';
    }

    $exec[$i] .= $props;

    // And that completes the execution string
  }
  }


// Now store in hidden INPUT objects for generate.php
for( $i=0; $i < count( $redshifts ); $i++ )
{
  echo '<INPUT TYPE="hidden" NAME="'.$redshifts[$i].'"
VALUE="'.$exec[$i].'"/>';
}
```

```
// Check options for VOTable or ASCII table
if( isset( $_POST['votable'] ) && isset( $_POST['asciitable'] ) )
{
  echo ' for both <b>VOTable and ASCII table</b> formats.';
}
elseif( isset( $_POST['votable'] ) )
{
  echo ' for <b>VOTable format</b> only.';
}
else
{
  echo ' for <b>ASCII table format</b> only.';
}
?>
<BR><BR>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="742" STYLE="padding-left:8px;padding-right:8px;margin-
bottom:5px" BGCOLOR="#BBBBBB">
<FONT COLOR="#FFFFFF"><b>Your Command Line</b></FONT>
</TD>
</TR>
</TABLE>
<FONT>
For future reference, the following commandlines will be executed to
obtain your base data should you wish to repeat these runs without the
use of e-Galform:<BR>
<?php
for( $i = 0; $i < count( $redshifts ); $i++ )
{
  if( isset( $_POST[$redshifts[$i]] ) ) echo '<BR><b>'.$i.':
'.$exec[$i].'</b><BR>';
}
?>
<BR><BR>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0">
<TR>
<TD WIDTH="742" STYLE="padding-left:8px;padding-right:8px;margin-
bottom:5px" BGCOLOR="#BBBBBB">
<FONT COLOR="#FFFFFF"><b>Starting Your Run</b></FONT>
</TD>
</TR>
</TABLE>
<?php
echo '<INPUT TYPE="hidden" NAME="votable"
VALUE="'.$_POST['votable'].'"/>';
echo '<INPUT TYPE="hidden" NAME="asciitable"
VALUE="'.$_POST['asciitable'].'"/>';

if( isset( $_POST['votable'] ) && isset( $_POST['asciitable'] ) )
{
  echo 'Click \'Next >>\' below to start generating your tables. When
the first stage is complete, e-Galform will inform you of further
instructions. Depending on the size of your run, and number of
redshifts you have chosen, this first stage may take anything from a
few minutes to several hours. During this time, you may leave e-
Galform to generate your tables. ';
}
elseif( isset( $_POST['votable'] ) )
{
```

```
  echo 'Click \'Next >>\' below to start generating your tables. When
the first stage is complete, e-Galform will inform you of further
instructions. Depending on the size of your run, and number of
redshifts you have chosen, this first stage may take anything from a
few minutes to several hours. During this time, you may leave e-
Galform to generate your tables. ';
}
else
{
 echo 'Click \'Next >>\' below to start generating your tables.
Depending on the size of your run, and number of redshifts you have
chosen, this first stage may take anything from a few minutes to
several hours. During this time, you may leave e-Galform to generate
your tables. ';
}
?>
<b>During this time, please do NOT use the navigation buttons on your
browser, and this may suspend and/or corrupt data output.</b>
</FONT>
<BR><BR>
       </TD>
    </TR>
    <TR>
       <TD WIDTH="742" STYLE="padding-left:4px;padding-right:4px"
ALIGN="RIGHT">
            <INPUT TYPE="submit" NAME="submit" VALUE="Next >>"
STYLE="font-size:9pt"/>
       </TD>
    </TR>
</FORM>
</TABLE>
<?php
do_html_footer();
?>
```

Appendix B: Moore's Law Of Processor Transistor Density