Geodetic Data Modeling System Internet Interface: Honors Thesis

Michael Still
964076

Geodetic Data Modeling System Internet Interface: Honors Thesis

by Michael Still

Abstract

This document discusses the **GDMS Internet interface**. This interface allows remote access to the vast majority of the functionality available in the X Windows version of **GDMS**. The interface was developed in Perl, and uses the Common Gateway Interface (CGI) offered by most modern web servers to interact with the web browser.

Table of Contents

List of Figures

Glossary

The following acronyms are used throughout this document.

ASP

ASP stands for Active Server Pages. ASP is Microsoft's server side scripting language. Server side scripting languages embed program elements into the HTML pages, which are executed and then removed before the page is returned to the web browser.

CGI

CGI, or Common Gateway Interface, is a standard which defines methods by which web servers and server side executables may communicate. The main method of communication is the passing of the URL provided to the script as an environment variable (National Center for Supercomputing Applications 1996).

GDMS

GDMS, the Geodetic Data Modeling System, is the name given to the undergraduate portion of my final year project. This is an X Windows based modeling package for geodetic data, and includes least squares, interpolation, windowing and fast Fourier transformation functionality.

GIF

The Graphics Interchange Format (GIF) was co-developed by CompuServe Inc. and Unisys Corporation in the late 1980s and early 1990s. GIF is dependent on LZW compression (CompuServe 1990), which is now having it's patent enforced by Unisys (Anonymous 2002).

HTML

HyperText Markup Language is the *lingua franca* of the web. It is a content markup language based on SGML (World Wide Web Consortium 2002).

JFIF

The file format commonly called JPEG is actually JPEG JFIF (JPEG File Interchange Format) — created by the Independent JPEG Group (Lilly 1996).

JPEG

JPEG stands for Joint Picture Expert Group, which is a group of the International Standards Organization. This group devised the JPEG compression format, which is used in the JFIF file format.

LZW compression

GIF is dependent upon LZW compression (CompuServe 1990), a compression algorithm which is named after it's developers Lempel, Ziv, and Welch (Nelson 1998).

MOTD

A MOTD is the Message Of The Day, a greeting message presented to users as they login to the **GDMS Internet interface**.

PHP

PHP, which stands for "PHP: Hypertext Processor" (AIMS Group, 2002), is an Open Source server side scripting language. Server side scripting languages embed program elements into the HTML pages, which are executed and then removed before the page is returned to the web browser.

PNG

Portable Network Graphics (PNG) was initially developed because of the patent problems with the GIF format described earlier in this section. The PNG format has since grown well past the feature set provided by GIF, and now rivals TIFF for completeness (Randers-Pehrson 1999).

TIFF

TIFF (Tagged Image File Format) is a raster (bitmap) image format which was originally produced by Aldus and Microsoft. Aldus was later acquired by Adobe, who manage the TIFF specification to this day. At the time of writing, the current version of the TIFF specification is TIFF version 6.0 (Adobe 1992).

URL

A Uniform Resource Locator is the "address" of a web page.

Chapter 1. Introduction

1.1. Motivation

At the time that the **Geodetic Data Modeling System** (GDMS) implementation project was initiated, several of the potential users expressed a desire for the application to be available over the Internet. There are a variety of reasons that this type of functionality is enticing. The primary reason for **GDMS** is that it allows casual users to ability to analyze data, whilst not having to maintain their own copies of the datasets. A secondary reason is that it allows users who would normally use the X windows interface to the application to access data whilst "in the field", or otherwise physically separated from their normal research location.

Initially, it was thought that a Internet interface to **GDMS** was outside the achievable scope of the project for 2002. However, as the year progressed, it became clear to me that implementing a Internet interface was indeed achievable, and would add genuinely useful functionality to the application. Hence this honors extension was undertaken.

1.2. Outcome

This document describes the design process and implementation of the **GDMS Internet interface**, a Perl application intended to operate under the Common Gateway Interface (CGI) offered by most modern web servers, including Apache, which is the web server in use within the University of Canberra Survey Laboratory. This application allows the on line use of almost all of the functionality available under the X windows **GDMS** system.

Chapter 2. Implementation rationale

2.1. Selection of the implementation technique

A variety of implementation techniques were considered for the **GDMS Internet interface** before a final selection was made. This section documents the various alternatives that were considered, and then justifies the decision that was made.

2.1.1. PHP: Hypertext Preprocessor

There is some evidence that the name of PHP originally stood for *Personal Home Page tools* (Lerdorf 1995). PHP is an in-page scripting language which was originally developed by Rasmus Lerdorf (Lerdorf 1995). It is now maintained and extended by a on line team of developers lead by Rasmus Lerdorf (AIMS Group, 2002).

Because PHP is an in-page scripting language, the actual code to generate the HTML page viewed by the user of the web browser is actually stored within the HTML page on the servers secondary storage. A trivial example is:

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo "Hello World<p>"; ?>
</body>
</html>
```

In this example, the tag starting with ?php is the script element. The output of this script is a simple HTML page saying “Hello World”.

The main advantage of in-page scripting is that the source code of the application is very tightly tied to the HTML presentation of the application’s user interface. This means that when a user is editing a page, it is apparent what the given code does at the time that the page is created.

The main disadvantage of such as system is that it requires that the users who are editing the HTML appearance of an application must also understand the scripting language in which the application is implemented. This means that graphics designers and layout consultants require further training. The application is no longer a *black box* in which the HTML designer can simply be a user. A second drawback is that the inclusion of the scripting within the HTML pages is that it clutters the HTML representation of the user interface, making it much harder to read. This problem is compounded by poor in-page scripting support in many HTML editors, including Mozilla. These editors discard tags they don’t comprehend at the time of editing the page, and this unfortunately includes tags such as PHP.

Developing the **GDMS Internet interface** in PHP would necessitate the users of the application running web servers with PHP modules installed (PHP Documentation Group, 2002). This is a extra level of configuration for systems administrators for little additional benefit.

2.1.2. Active Server Pages

Active Server Pages (ASP) is Microsoft’s equivalent of PHP. Discussion of this alternative is deliberately brief, because ASP effectively has all the advantages and disadvantages of PHP, with the additional constraint that it only operates of Microsoft web server products running on Microsoft Operating Systems. This is not an acceptable constraint for the main target user of the application — the University of Canberra Survey Laboratory, who are not heavy users of Microsoft products.

2.1.3. C or C++ Common Gateway Interface

Another implementation alternative considered was to implement the system in either C or C++ using the Common Gateway Interface (CGI) subsystem offered by most web servers.

This implementation technique has the advantage of the code for the **GDMS Internet interface** being in the same language as the **GDMS** application itself. However it has the disadvantage that parsing the incoming URLs is less trivial, as there is limited language support without the inclusion of non-standard application libraries (Kahan 2002).

C and C++ are not idiomatic methods of implementing CGI applications, and this would therefore increase the difficulty of maintaining the system for future students and administrators.

2.1.4. Perl CGI

Perl was originally developed by Larry Wall in 1987 (Perl Mongers, 1999) and is now developed by a team of Open Source developers. Perl is a shell script like scripting language, although it's syntax is much more powerful than that of shell script. Perl also has world class regular expression support as a standard, well integrated, feature. Perl has well integrated CGI support built into the language itself, and is optimized for the parsing of complex strings (such as the requests which are returned by a web browser to the **GDMS Internet interface** via the CGI apparatus).

Perl's inbuilt handling of CGI web interfaces makes it much easier to develop web applications than it would be in C or C++. For example, the following is a simple hello world application in Perl:

```
# Import some modules
use strict;
use CGI;
# Declare variables because strict is enabled
my($result);
# Setup the CGI module, and output headers
$result = new CGI();
print $result->header;
print "Hello World";
```

It can be seen from this example that Perl makes CGI programming trivial. This same program in C or C++ would have taken many more lines, and would have needed to include the hard coding of the required headers.

Perl is also extremely stable, well documented, and is already installed at most sites.

2.1.5. Conclusion

After consideration of all of the factors outlined above, Perl CGI was identified as the most suitable implementation language for the **GDMS Internet interface**.

2.2. Presentation of the user interface

In accordance with Internet user interface design best practice, the **GDMS Internet interface** presents a fully configurable user interface. This is implemented by providing extension tags above those which are available in the standard HTML specification (Raggett, Le Hors, & Jacobs 1999).

The **GDMS Internet interface** follows the recommendations of the most recent HTML specification at the time of development for the insertion of scripting tags into HTML source files (Raggett, Le Hors, & Jacobs 1999). Further documentation on the **GDMS Internet interface** tags available can be found in Appendix A of this document.

2.3. Selection of an image format for graph display

Much thought was expended on which image format to use for the plots displayed in the **GDMS Internet interface**. There are advantages and disadvantages to all the formats available at the time of writing this document. The following formats were considered for the **GDMS Internet interface**:

2.3.1. Tagged Image File Format

TIFF (Tagged Image File Format) is a raster (bitmap) image format which was originally produced by Aldus and Microsoft. Aldus was later acquired by Adobe, who manage the TIFF specification to this day. At the time of writing, the current version of the TIFF specification is TIFF version 6.0 (Adobe 1992). It should be noted that no significant development of the TIFF standard has occurred since that date.

The main advantage of the TIFF image format is that it is extremely mature. It has a large selection of options to ease the development of imaging software, for example the developer can use which ever endian representation is convenient for them, as well as such fundamental parameters as photometric interpretation, whether a high value is lighter or darker than a low value for a given colour sample within a pixel (Adobe 1992). There is also a stable, well supported Open Source application programmers interface available to manage creation and interpretation of images within the TIFF format (Warmerdam & Welles 2002). The interface to this library does not frequently change, which is also an advantage for future portability of the **GDMS Internet interface**.

However, some of these advantages lead to the principal disadvantages that the TIFF format suffers from. Because the content of the file is so loosely specified, it is quite hard to write an application which can correctly decode all possible TIFF format images. This has resulted in there being only a few high quality TIFF viewing applications being available. The relative difficultly of implementing a TIFF decoder has also resulted in the web browser support for TIFF images being very poor (Bither 2002).

TIFF is therefore an inappropriate image format for use for the **GDMS Internet interface**, as good browser support in as many browsers as possible is vital.

2.3.2. Graphics Interchange Format

The Graphics Interchange Format (GIF) was co-developed by CompuServe Inc. and Unisys Corporation in the late 1980s and early 1990s. GIF is dependant on LZW compression (CompuServe 1990), which is now having it's patent enforced by Unisys (Anonymous 2002). The GIF format itself is quite limited, with a maximum palette size of 255 entries — in other words 255 distinct colours are available in each image.

Vendors are also dropping support for the GIF format because of the expense of licensing the Unisys LZW patent, for example, the Open Source *libgif* is no longer supported for the creation of GIF images (Raymond, 1998).

Despite all of these largely political issues, the GIF format is still well supported in all web browser versions, which would make it an ideal candidate for the **GDMS Internet interface** if it wasn't for the lack of a stable and reliable compression library.

2.3.3. Joint Picture Experts Group File Interchange Format

JPEG is an interesting image format because whilst the JPEG compression codec has been standardized by ISO, the actual on disc format commonly called JPEG is not part of the ISO standard. The file format commonly called JPEG is actually JPEG JFIF — created by the Independent JPEG Group (Lilly 1996).

JPEG support in web browsers is excellent, with support rapidly improving after 1996 (Lilly 1996). The biggest factor stopping the use of JPEG within the **GDMS Internet interface** is that the compression codec is lossy.

2.3.3.1. Loss

The JPEG compression algorithm is lossy — in other words, the act of compressing the image results in image data being discarded. This has the effect of noticeably reducing the clarity of the images — a characteristic which is especially noticeable with images which contain text at smaller point sizes. The use of such text is a feature of the **GDMS** graphing subsystem.

For example, JPEG compressing the following image results in the following output image:

Figure 2-1. The UC logo before JPEG compression

Figure 2-2. The UC logo after it has been JPEG compressed

This output is acceptible, because there is limited use of fine detail, and therefore limited loss of image clarity. The effects are much worse with an image which contains text at small point sizes:

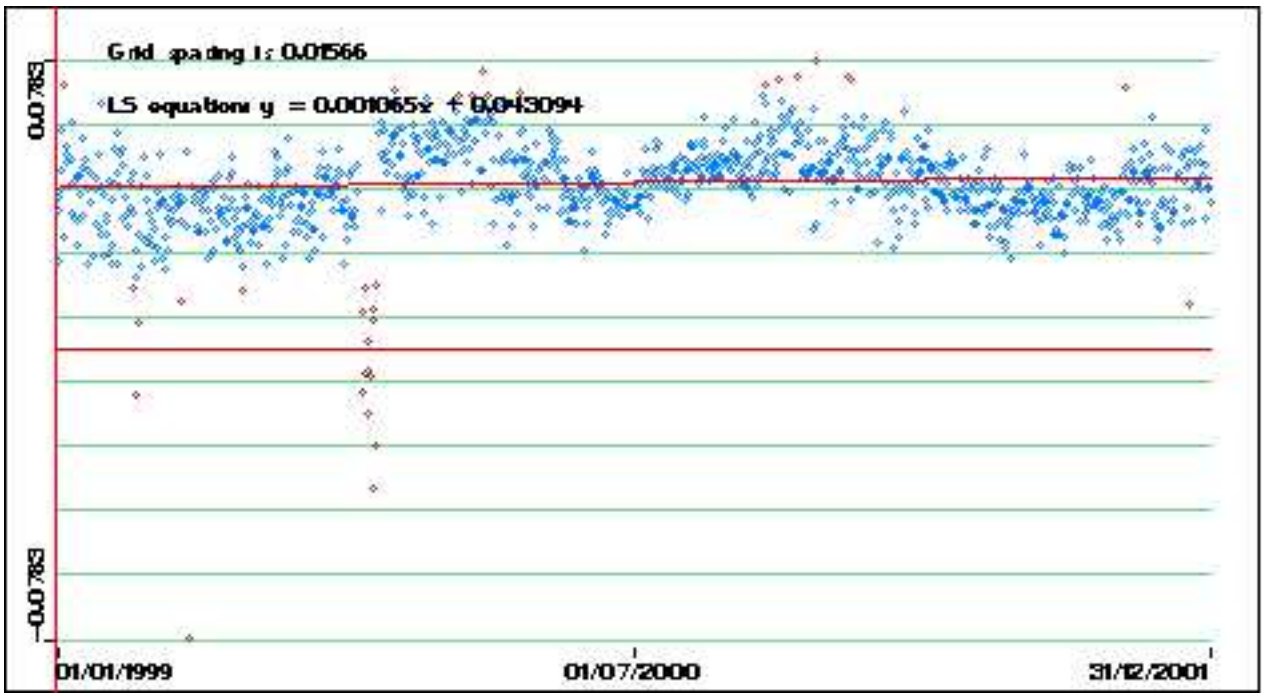

Figure 2-3. A sample GDMS plot before JPEG compression

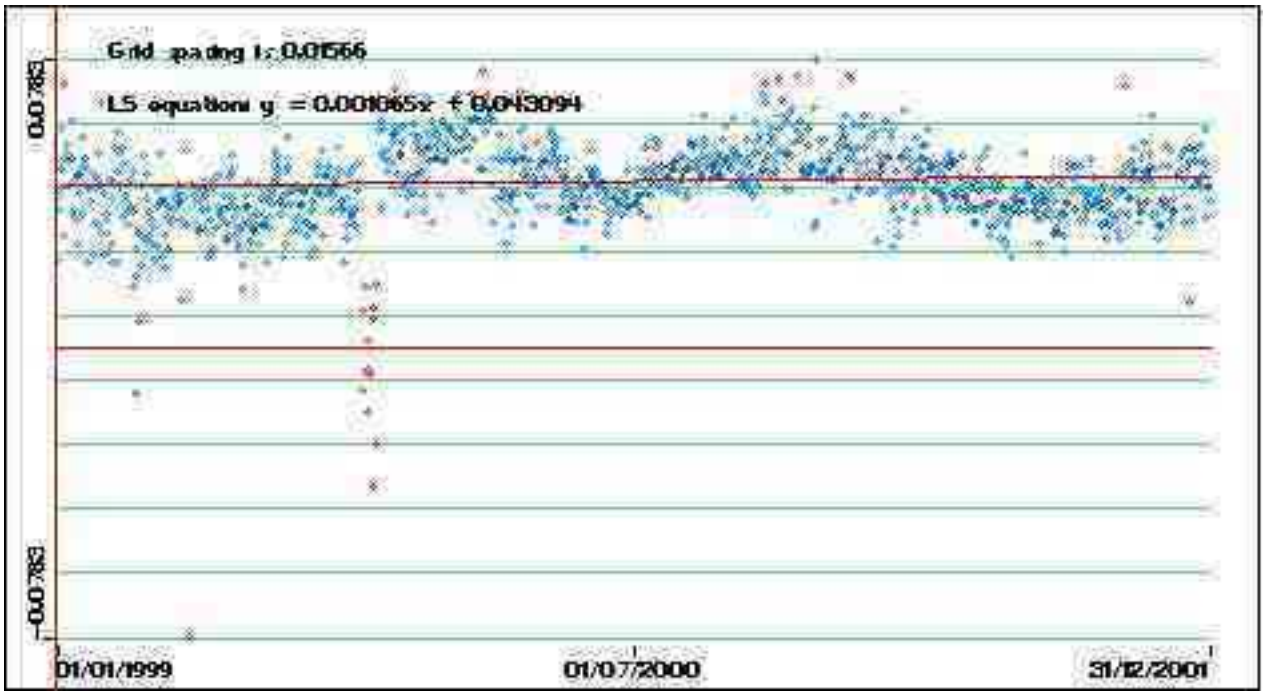

Figure 2-4. A sample GDMS plot after it has been JPEG compressed

2.3.4. Portable Network Graphics

Portable Network Graphics (PNG) was initially developed because of the patent problems with the GIF format described earlier in this section. The PNG format has since grown well past the feature set provided by GIF, and now rivals TIFF for completeness (Randers-Pehrson 1999).

The PNG support in many browsers is not perfect. However, browsers above version three for both Netscape and Microsoft Internet Explorer, and all versions of Mozilla have excellent support of PNG. There is an actively maintained and well supported Open Source programmers interface to the PNG format, in fact it is the reference implementation of the specification (Roelofs 2002), which reinforces it's stability.

2.3.5. Conclusion

Based on the factors outlined above, the PNG image format was selected for the **GDMS Internet interface**. This is because of it's excellent browser support in modern browsers, as well as it's mature and stable Open Source application programmers interface.

Chapter 3. Implementation

This chapter discusses the actual application which was produced. The application is quite complex, with a series of pages being used to prompt the user and display information. The interaction of these different pages, each of which has a "command" associated with it, is shown in figure 3.2 below.

When the application starts, the command is not defined, as the user is unlikely to have specified one as part of the URL. This results in the **GDMS Internet interface** using the *main* command. The user is then presented with a welcome screen, which displays the message of the day. The message of the day in intended to be used to warn the users of the **GDMS Internet interface** of scheduled unavailability of the system, or new datasets having been added. There are no limits on the text that can be included in the message of the day however, so it could inform users of an impending Christmas party, or other equally unrelated events.

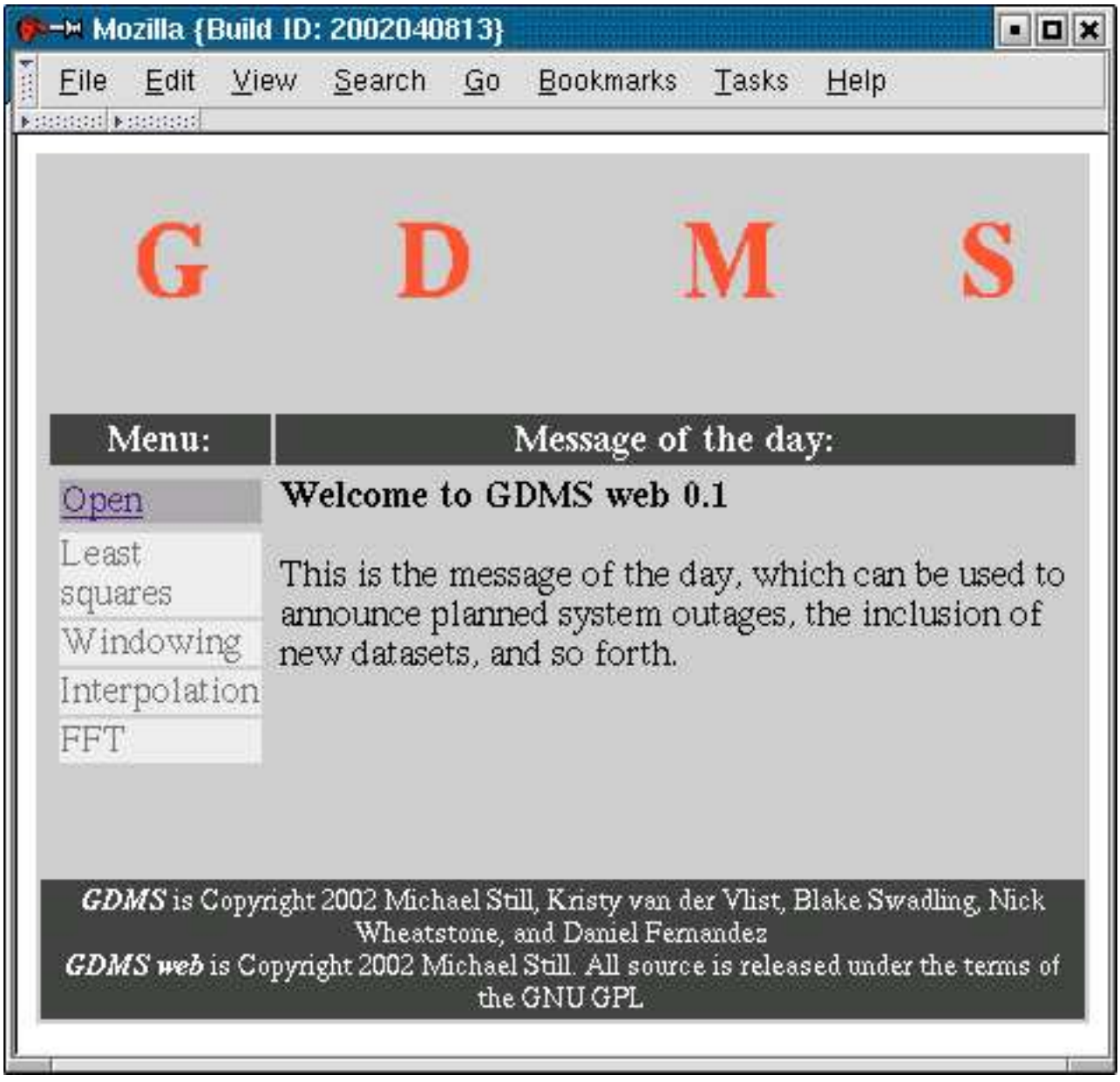

Figure 3-1. GDMS Internet interface welcome screen

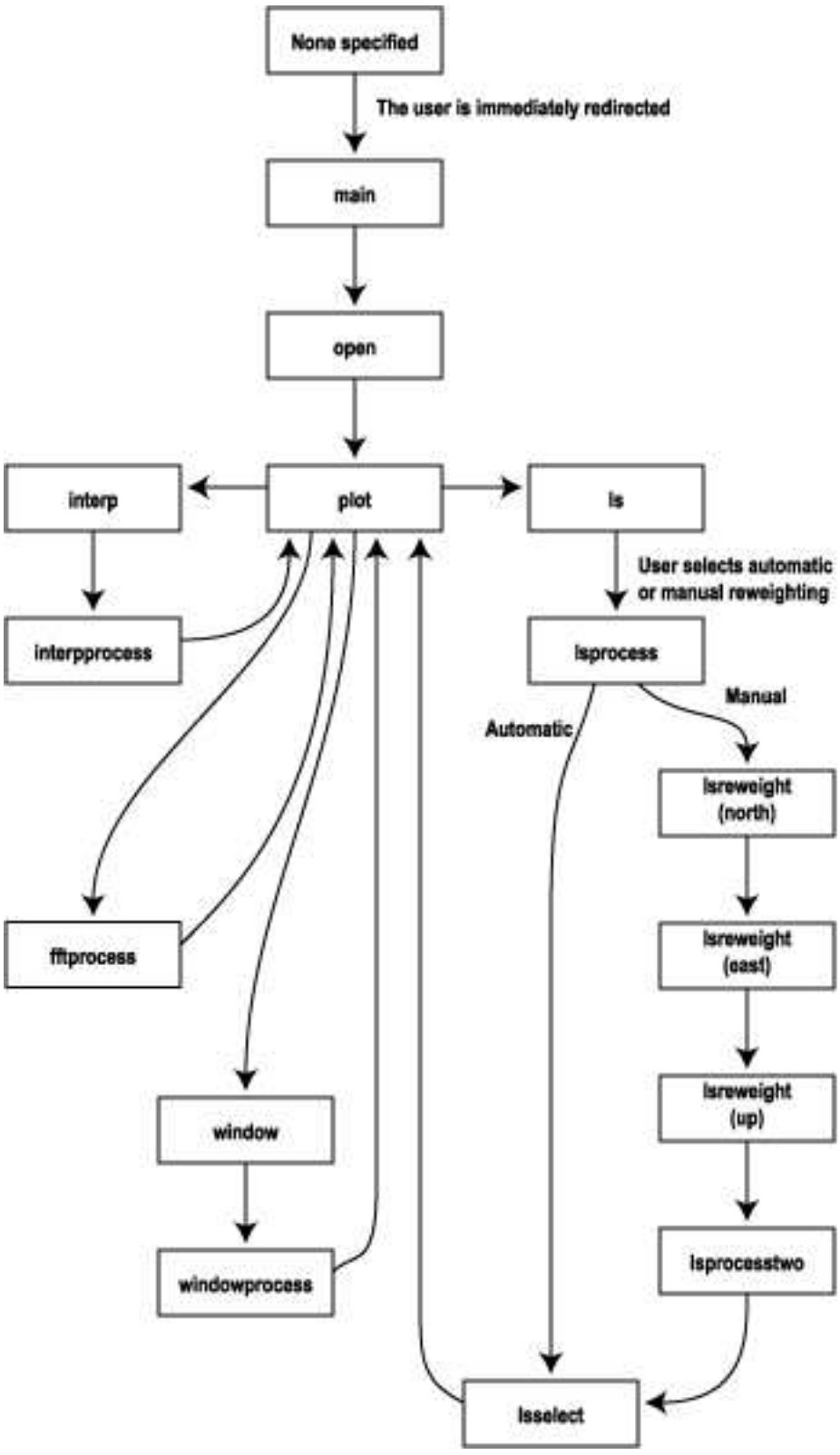

Figure 3-2. GDMS Internet interface command interaction

The message of the day is stored as a template named motd.html within the templates directory. See the configuration appendix of this document for more information about customizing this file.

The next operation that the user will perform is selecting a dataset to start processing. At

this stage all other commands are disabled, because of a reliance on a dataset being available for processing. The user selects the *open* command, and a new page displaying a list of the available datasets is displayed. Once the user has selected one of the datasets from this list, a new page showing a plot of the selected dataset is displayed.

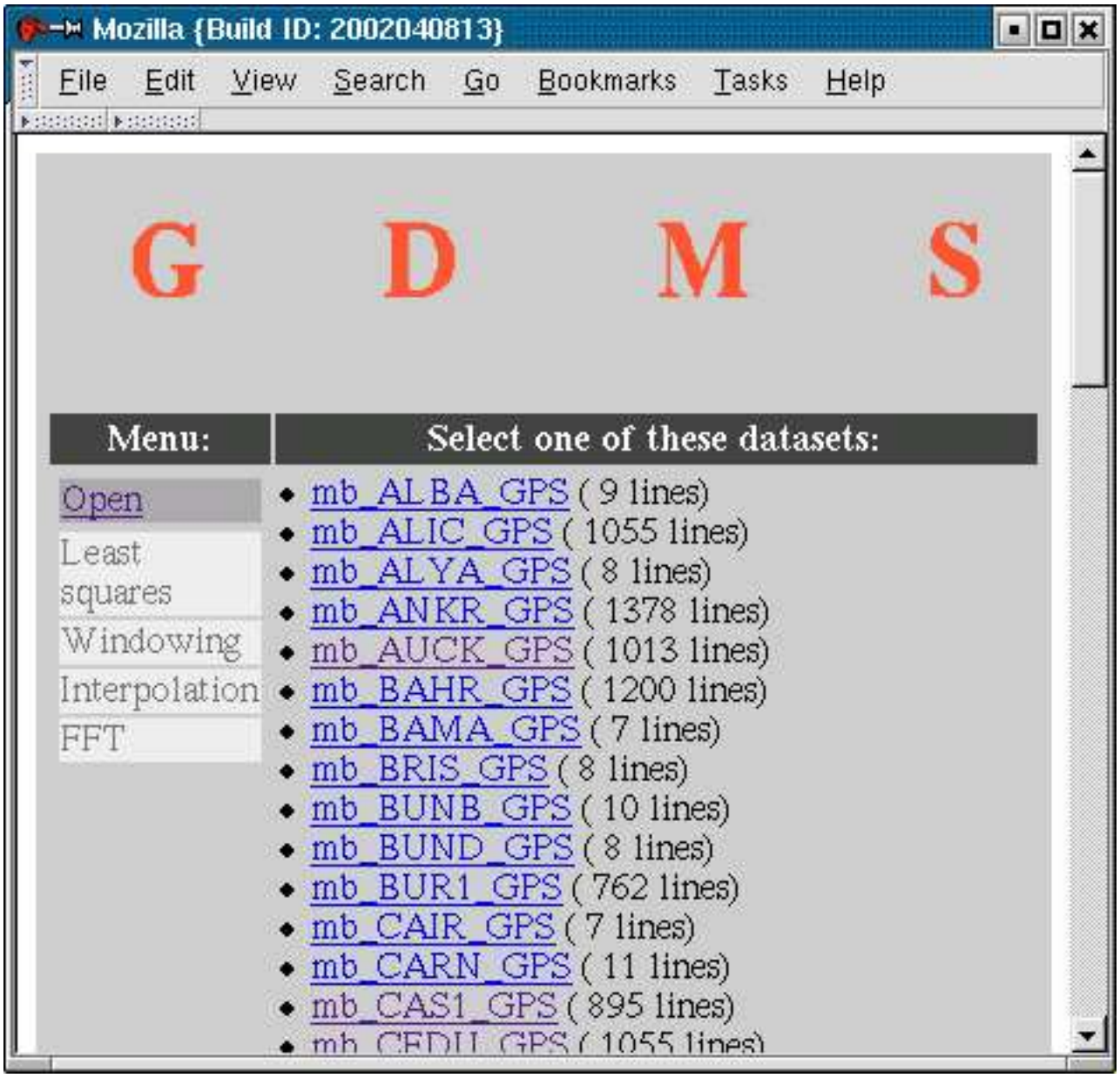

Figure 3-3. Selecting a dataset

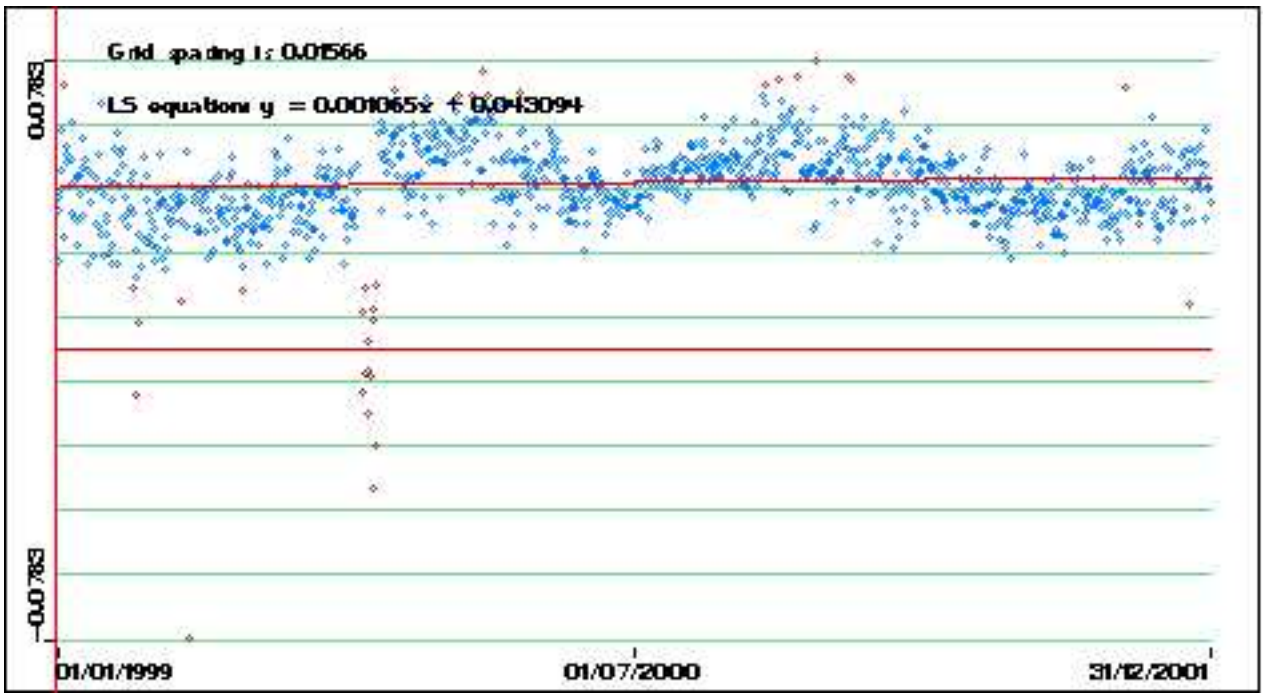

Figure 3-4. A dataset plot in the north direction (time domain)

It should be noted that at this time there is no way to configure the various graphing options offered by **GDMS** within the **GDMS Internet interface**. The recommended method for selecting the graphing style preferred by the users of the system is to run the **GDMS** X Windows application as the user id the web server runs as, and then copy the file ~/.cep to the same directory as the **GDMS Internet interface** executable. It should also be noted that at this time it is not possible to have individual graphing preferences for different users.

Now that the user has selected a dataset, the mathematical commands are now available. These may be selected from the same menu as the graphing commands, and include Least Squares regressions, windowing, interpolation, and Fast Fourier Transformations. All of the functionality available within the **GDMS** X Windows application is available for these mathematical operations.

The **GDMS Internet interface** graphing subsystem also handles graphs in the frequency domain, in the same manner as the **GDMS** X Windows interface.

Chapter 4. Conclusion

Based on the discussion in previous chapters, Perl CGI with PNG images were selected for the implementation of the **GDMS Internet interface**. This has resulted in a easy to understand code base, which will aid in future expansion of the application. The **GDMS Internet interface** should also correctly function with all modern web servers, including the Apache web server which is used by the University of Canberra Survey Laboratory.

There is some scope for future improvement of the **GDMS Internet interface**. This includes:

- A web aware method of configuring graphing preferences.
- Per user graphing preferences.
- Allowing zooming on the dataset graphs in the same manner that the **GDMS** X Windows application allows.
- Caching of processed datasets where it makes sense. Graphs are already cached.
- Implementation of features as they are added to the **GDMS** X Windows application.

Overall, the **GDMS Internet interface** is a stable and reliable light weight interface to the **GDMS** batch interface.

Chapter 5. References

5.1. References

Adobe, 3 June 1992, *TIFF Revision 6.0*, [Online] http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf

AIMS Group, 2002, *PHP CVS mailling list archive*, [Online] http://marc.theaimsgroup.com/?l=php-cvs

Anonymous, 2 February 2002, *The GIF Controversy: A Software Developer's Perspective*, [Online] http://www.cloanto.com/users/mcb/19950127giflzw.html

Bither, B., [Last accessed] November 2002, *Benefits of the PNG Image Format*, [Online] Available: http://www.atalasoft.com/png.asp

CompuServe, 1990, *Graphics Interchange Format: Version 89a* [Online] http://www.w3.org/Graphics/GIF/spec-gif89a.txt

Kahan. J., 12 June 2002, *Libwww - the W3C Protocol Library*, [Online] http://www.w3.org/Library/

Lerdorf, R., 8 June 1995, *Personal Home Page Tools (PHP Tools)*, [Online] http://groups.google.com/groups?selm=3r7pgp$aa1@ionews.io.org

Lilly, C., 13 February 1996 [Last updated: 18 July 2001], *JPEG JFIF*, [Online] http://www.w3.org/Graphics/JPEG/

National Center for Supercomputing Applications, 8 March 1996, *The Common Gateway Interface*, [Online] http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

Nelson, M., 1989, *Dr. Dobb's Journal October, 1989: LZW Data Compression*, [Online] http://dogma.net/markn/articles/lzw/lzw.htm

Perl Mongers, 1999, *Perl History*, [Online] http://www.perl.org/press/history.html

PHP Documentation Group, 2002, *PHP Manual*, [Online] http://www.php.net/manual/en/print/

Raggett, D., Le Hors, A., Jacobs, I., 24 December 1999 [last update], *HTML 4.01 Specification*, [Online] Available: http://www.w3.org/TR/html4/

Randers-Pehrson, G., July 1999, *PNG (Portable Network Graphics) Specification, Version 1.2*, [Online] http://www.libpng.org/pub/png/spec/png-1.2.pdf

Raymond, E. S., 8 July 1998, *The GIFLIB home page*, [Online] http://tuxedo.org/~esr/giflib/

Roelofs, G., 2002, *libpng*, [Online] http://www.libpng.org/pub/png/libpng.html

World Wide Web Consortium (w3c), 12 December 2002, *HyperText Markup Language (HTML) Home Page*, [Online] http://www.w3.org/MarkUp/

Warmerdam, F. and Welles, M., 24 January 2002 [last update], *The libtiff home page*, [Online] http://www.libtiff.org/

Appendix A. GDMS Internet interface scripting elements

A.1. Introduction

This appendix documents the various tags which are implemented by the **GDMS Internet interface** in addition to the standard HTML tags normally available. All of these tags are within the scripting name space provided by HTML 4.01 (Raggett, Le Hors, & Jacobs 1999).

A.1.1. commands

This tag will display a menu of the commands available in the current context — for example, when the Internet interface first starts, it will list the open command. Commands which are not available, but are normally available will appear in a disabled style. This results in a list of commands which is consistent, and therefore reduces the potential for user confusion.

An example use is as follows:

```
&{commands};
```

The display of the command menu is altered by the following configuration file entries:

- *commandentry*: this HTML snippet is used for enabled commands
- *discommandentry*: this HTML snippet is used for disabled commands

For example, these configuration entries ship by default with the **GDMS Internet interface**:

```
# This line is simple HTML used for format the command entries,
# %s is the name of the command (including link HTML)
$commandentry = "<tr><td bgcolor=\"AAAAAA\">%s</td><tr>";

# This line is used for commands which aren't available...
$discommandentry = "<tr><td bgcolor=\"EEEEEE\">
  <font color=\"777777\">%s</font></td></tr>";
```

A.1.2. dataset

This tag will display the name of the currently open dataset, if one is open. This might possibly be a descriptive string is processing has occurred on the dataset.

An example usage is:

```
&{dataset};
```

The dataset name is the portion of the filename before the extensions .dat1, .dat2, and .dat3 are applied. For example *mb_AUCK_GPS* is a dataset name.

A.1.3. datasets

This tag will display a list of the datasets available in the dataset directory.

An example usage is:

```
&{datasets};
```

The dataset directory is configured with the following configuration file entry:

```
# This line defines where the datasets are stored
$datasets = "/home/httpd/gdms-datasets/";
```

There are also several configuration options which alter the appearance of the list of datasets available. These are:

- *selectstart*: this is used for any output which should appear at the beginning of the list. This could include HTML tags for the creation of the required list markup (for example tables).
- *selectentry*: this configuration item is used for each entry in the list. The special text **%s** is replaced by the HTML for the entry itself.
- *selectend*: this is used for any HTML required to finalize the list.

```
# This is used for selection lists (for instance datasets)
$selectstart = "List start";
$selectentry = "<LI>%s";
$selectend = "List end";
```

A.1.4. eastplot

This tag will insert the HTML and required URL for the East direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{eastplot};
```

A.1.5. interpparams

This tag will output the HTML form required for the entry of interpolation processing parameters. The user will then select "OK" on this web form, and the interpolation operation will occur.

An example usage is:

```
&{winparams};
```

A.1.6. lsparams

This tag will output the HTML form required for the entry of Least Squares processing parameters. The user will then select "OK" on this web form, and the Least Squares regression will occur.

An example usage is:

```
&{lsparams};
```

A.1.7. lsreweightdir

This tag converts to the direction in which the user is being prompted for a Least Squares re weighting matrix.

An example usage is:

```
&{lsreweightdir};
```

A.1.8. lsselect

This tag will output the HTML form required for the selection of the Least Squares output. This is because each Least Squares operation produces two datasets, the data (with a Least Squares line and indication of the effect of re weighting), and the residuals dataset.

Note that if the user wants to open both datasets, then they can use the "Open in new window" or equivalent functionality in their web browser.

An example usage is:

```
&{lsselect};
```

A.1.9. matrices

This tag will output the HTML that represents the list of re weighting matrices which are available. Note that matrices must end in the extension .mat to be listed.

An example usage is:

```
&{matrices};
```

A.1.10. motd

This tag will display a welcome message to the user — the Message Of The Day (MOTD). This can be used to inform users of new datasets now being available, planned system outages, and other such informational messages.

An example usage is:

```
&{motd};
```

A.1.11. northplot

This tag will insert the HTML and required URL for the North direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{northplot};
```

A.1.12. upplot

This tag will insert the HTML and required URL for the Up direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{upplot};
```

A.1.13. winparams

This tag will output the HTML form required for the entry of windowing processing parameters. The user will then select “OK” on this web form, and the windowing operation will occur.

An example usage is:

```
&{winparams};
```

Appendix B. Source code

```
#!/usr/bin/perl

# GDMS web broker
# Copyright (C) Michael Still             2002
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

# This is the broker between the GDMS code and the CGI server. It's
# role is to parse templates, execute requests, and cache results for
# speedy processing next time that request is seen...
use strict;
use CGI;

my($result, $command, $TEMP);

# Variables set by the config file
my($templates, $datasets, $matrices, $commandentry, $discommandentry,
   $selectstart, $selectentry, $selectend, $plotcache, $tmpdir,
   $rooturl, $ploturl, $gdms, $temp);

# Setup the CGI module
$result = new CGI();

# Read in the config file
eval 'cat gdms.config' or die
    "GDMSweb $$: Could not read config file: $@";
$temp = $gdms;
$temp =~ s/\/.*$//;

# Determine what page we are accessing
$command = $result->param('command');
if($command eq ""){
    $command = "main";
}

# Some pages just do some processing and then immediately redirect to
# another page. I need to deal with these before I can display the
# page headers
print STDERR "GDMSweb $$: Processing $command\n";
print STDERR "GDMSweb $$: Dataset is permanent\n"
    if($result->param('intype') eq "");
print STDERR "GDMSweb $$: Dataset is temporary\n"
    if($result->param('intype') ne "");

if($command eq "lsprocess"){
    $temp = "$rooturl?dataset=".$result->param('dataset')."-ls".
    $result->param('reweight');

    # If we are using manual reweighting, then we need to get more
    # information
    if($result->param('reweight') ne "auto"){
    $temp = $temp."&command=lsreweight&dir=north";
    }
    else{
    $temp = $temp."&command=lsselect&data=".
        $result->param('dataset')."-ls".
        $result->param('reweight')."-data&residuals=".
```

```
        $result->param('dataset')."-ls".
        $result->param('reweight')."-residuals";

    # Do the processing here
    performLs("auto");
    }

    $temp = $temp."&intype=temp";

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after automatic LS to: $temp\n";
    print $result->redirect($temp);
    exit;
}
elsif($command eq "lsprocesstwo"){
    $temp = "$rooturl?dataset=".$result->param('dataset')."-ls".
    $result->param('reweight');

    $temp = $temp."&command=lsselect&data=".$result->param('dataset').
    "-ls".$result->param('reweight')."-data&residuals=".
    $result->param('dataset')."-ls".
    $result->param('reweight')."-residuals";

    # Do the processing here
    performLs($result->param('lsargs'));

    $temp = $temp."&intype=temp";

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after manual LS to: $temp\n";
    print $result->redirect($temp);
    exit;
}
elsif($command eq "windowprocess"){
    $temp = "$rooturl?dataset=".$result->param('dataset')."-win".
    $result->param('wtype')."&command=plot";
    $temp = $temp."&intype=temp";

    # Do the processing here
    performWindow();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after window to: $temp\n";
    print $result->redirect($temp);
    exit;
}
elsif($command eq "interpprocess"){
    $temp = "$rooturl?dataset=".$result->param('dataset')."-interp".
    $result->param('itype')."&command=plot";

    $temp = $temp."&intype=temp";

    # Do the processing here
    performInterpolation();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after interpolation to: $temp\n";
    print $result->redirect($temp);
    exit;
}
elsif($command eq "fftprocess"){
    $temp = "$rooturl?dataset=".$result->param('dataset').
    "-fft&command=plot";

    $temp = $temp."&intype=temp";

    # Do the processing here
    performFFT();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after FFT to: $temp\n";
```

```
    print $result->redirect($temp);
    exit;
}

# Determine if the GDMS script for this page exists already
print $result->header;
print processTemplate("$templates/$command.html");
exit;



################################################################################
# Find the template file, and then parse it
sub processTemplate(@args){
    my($file) = @_;
    my($pre, $post, $cmd, $output, $len, $line);

    # This local usage is used to make the TEMPLATE filehandle local
    # to this subroutine...
    local *TEMPLATE;
    print STDERR "GDMSweb $$: Processing template file: $file\n";

    $output = "";
    if(!open TEMPLATE, "< $file"){
    print STDERR "GDMSweb $$: $file not found\n";
    open TEMPLATE, "< $templates/commandnotfound.html";
    }
    while(<TEMPLATE>){
    # Repeatedly process a line until there are not more template
    # commands
    $line = $_;
    $len = -1;

    while($len != length($line)){
        $len = length($_);

        $_ = $line;
        if(/(.*)&{([^%]*)};(.*)/){
        $pre = $1;
        $cmd = $2;
        $post = $3;

        if($cmd eq "commands"){
            # List the available commands
            $line = $pre.getCommands().$post;
        }
        elsif($cmd eq "dataset"){
            my($tempdescription);

            # The name of the current dataset
            if($result->param('dataset') ne ""){
            if($result->param('intype') ne "temp"){
                open TEMP, ("head -1 $datasets/".
                    $result->param('dataset').
                    ".dat1 |");
            }
            else{
                open TEMP, ("head -1 $tmpdir/".
                    $result->param('dataset').
                    ".dat1 |");
            }

            $tempdescription = "";
            while(<TEMP>){
                chomp;
                $tempdescription = "Filename: <i>".
                $result->param('dataset')."</i><BR>".
                $_;
            }
            close TEMP;
            $line = $pre.$tempdescription.$post;
```

```
    }
    else{
    $line = $pre.$post;
    }
}
elsif($cmd eq "datasets"){
    # List the datasets in the dataset directory
    my($temptotal, $tempfile);
    $temptotal = "";

    open TEMP, "find $datasets -type f -name \"*.dat1\" |";
    while(<TEMP>){
    my($linecount);
    $tempfile = $_;
    $tempfile =~ s/$datasets\/*//;
    $tempfile =~ s/.dat1\n$//;

    $linecount = ‘cat $datasets/$tempfile.dat1 | wc -l‘;
    $temptotal = $temptotal.
        substHTML($selectentry,
              "<a href=\"$rooturl?".
              "command=plot&".
              "dataset=$tempfile\">".
              $tempfile."<\/a> ".
              "($linecount lines)");
    }
    close TEMP;

    $line = $pre.$temptotal.$post;
}
elsif($cmd eq "motd"){
    # Output a message of the day
    $line = $pre.processTemplate("$templates/motd.html").
    $post;
}
elsif($cmd eq "northplot"){
    # A plot in the X;
    $line = $pre.generateAndLink("x").$post;
}
elsif($cmd eq "eastplot"){
    # A plot in the Y direction
    $line = $pre.generateAndLink("y").$post;
}
elsif($cmd eq "upplot"){
    # A plot in the Z direction
    $line = $pre.generateAndLink("z").$post;
}
elsif($cmd eq "lsparams"){
    # Output a HTML form for the parameters of a LS
    # regression
    $line = $pre.generateForm("lsprocess",
                  $result->param(’dataset’),
                  "$templates/lsparams.html").
                  $post;
}
elsif($cmd eq "lsselect"){
    $line = $pre."<a href=\"$rooturl?dataset=".
    $result->param(’data’).
    "&command=plot&intype=temp\">Data</a><br>".
    "<a href=\"$rooturl?dataset=".
    $result->param(’residuals’).
    "&command=plot&intype=temp\">Residuals</a>".
    $post;
}
elsif($cmd eq "lsreweightdir"){
    $line = $pre.$result->param(’dir’).$post;
}
elsif($cmd eq "winparams"){
    # Output a HTML form for the parameters of a
    # windowing operation
    $line = $pre.generateForm("windowprocess",
```

```
                        $result->param('dataset'),
                        "$templates/winparams.html").
                        $post;
        }
        elsif($cmd eq "interpparams"){
            # Output a HTML form for the parameters of a
            # interpolation operation
            $line = $pre.generateForm("interpprocess",
                        $result->param('dataset'),
                        "$templates/interpparams.html").
                        $post;
        }
        elsif($cmd eq "matrices"){
            # List the datasets in the dataset directory
            my($temptotal, $next);
            $temptotal = "";

            if($result->param('dir') eq "north"){
            $next = "command=lsreweight&dir=east";
            }
            elsif($result->param('dir') eq "east"){
            $next = "command=lsreweight&dir=up";
            }
            else{
            $next = "command=lsprocesstwo&lsreweight=manual";
            }

            open TEMP, "find $matrices -type f -name \"*.mat\" |";
            while(<TEMP>){
            chomp;
            s/$matrices//;

            $temptotal = $temptotal.
                substHTML($selectentry,
                      "<a href=\"$rooturl?$next&lsargs=".
                      $result->param('lsargs').
                      "$matrices/$_,\">$_<\/a>");
            }
            close TEMP;

            $line = $pre.$temptotal.$post;
        }
        }
    }

    # And now we can print out the resultant line
    $output = $output.$line;
    }

    return $output;
}

# Determine what commands should be available at this time
sub getCommands(@args){
    my($output, $temp);

    $output = "";
    # If there is no current dataset
    if($result->param('dataset') eq ""){
    # Dataset open
    $output = $output.substHTML($commandentry,
                    "<a href=\"$rooturl?command=open\">Open<\/a>");

    # Maths
    $output = $output.substHTML($discommandentry,
                    "Least squares");
    $output = $output.substHTML($discommandentry, "Windowing");
    $output = $output.substHTML($discommandentry, "Interpolation");
    $output = $output.substHTML($discommandentry, "FFT");
    }
    else{
```

```
    # Datset open
    $output = $output.substHTML($commandentry,
                    "<a href=\"$rooturl?command=open\">Open<\/a>");

    # Maths
    $temp = "<a href=\"$rooturl?command=ls&dataset=".
        $result->param('dataset');
    $temp = $temp."&intype=temp"
        if($result->param('intype') ne "");
    $temp = $temp."\">Least squares</a>";
    $output = $output.substHTML($commandentry, $temp);

    $temp = "<a href=\"$rooturl?command=window&dataset=".
        $result->param('dataset');
    $temp = $temp."&intype=temp"
        if($result->param('intype') ne "");
    $temp = $temp."\">Windowing</a>";
    $output = $output.substHTML($commandentry, $temp);

    $temp = "<a href=\"$rooturl?command=interp&dataset=".
        $result->param('dataset');
    $temp = $temp."&intype=temp"
        if($result->param('intype') ne "");
    $temp = $temp."\">Interpolation</a>";
    $output = $output.substHTML($commandentry, $temp);

    $temp = "<a href=\"$rooturl?command=fftprocess&dataset=".
        $result->param('dataset');
    $temp = $temp."&intype=temp"
        if($result->param('intype') ne "");
    $temp = $temp."\">FFT</a>";
    $output = $output.substHTML($commandentry, $temp);
    }

    return $output;
}

# Substitute into the HTML stub from the config file
sub substHTML(@args){
    my($html, $insert) = @_;
    my($temp);

    $temp = $html;
    $temp =~ s/%s/$insert/;
    return $temp;
}

# This subroutine deals with generating plots as required and then
# outputs the HTML needed to link to that image
sub generateAndLink(@args){
    my($dir) = @_;
    my($file, $unique);
    local *COMMANDS;

    # Generate the filename
    $unique = "$$-".time()."-".rand();
    $file = "$plotcache/".$result->param('dataset')."-$dir.png";

    if(! -f $file){
    # We need to generate the image
    print STDERR "GDMSweb $$: Plot cache miss for ".
        $result->param('dataset')." ($dir)\n";
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
        die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
        print COMMANDS "open $datasets/".
        $result->param('dataset')."\n";
    }
    else{
```

```
        print COMMANDS "open $tmpdir/".
        $result->param('dataset')."\n";
    }

    print COMMANDS "plot $dir $file\n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
    print STDERR "GDMSweb $$: GDMS execute return code is $?\n";
    }

    # Now link to that image
    return "<img src=\"$ploturl/".$result->param('dataset').
    "-$dir.png\">";
}

# Generate a HTML form for the required paramers
sub generateForm(@args){
    my($command, $dataset, $passedform) = @_;
    my($resultstring, $DUMPFILE);

    # Start the form
    $resultstring = "<form method=\"post\" action=\"$rooturl\" ";
    $resultstring = $resultstring.
    "enctype=\"application/x-www-form-urlencoded\">";
    $resultstring = $resultstring.
    "<input type=\"hidden\" name=\"command\" value=\"$command\" />";
    $resultstring = $resultstring.
    "<input type=\"hidden\" name=\"dataset\" value=\"$dataset\" />";
    if($result->param('intype') ne ""){
    print STDERR "Continuing temporary tag\n";
    $resultstring = $resultstring.
        "<input type=\"hidden\" name=\"intype\" value=\"temp\" />";
    }

    $resultstring = $resultstring.processTemplate("$passedform");

    # End the form
    $resultstring = $resultstring."<BR><BR><div align=\"center\">";
    $resultstring = $resultstring.
    "<input type=\"submit\" name=\"commit\" value=\" OK \" /></div></form>";

    return $resultstring;
}

# Create a dataset in the temp location which is the output of a LS
# regression
sub performLs(){
    my($lsargs) = @_;
    my($unique);
    local *COMMANDS;

    print STDERR "GDMSweb $$: Performing LS regression\n";
    $unique = "$$-".time()."-".rand();
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
    print COMMANDS "open $datasets/".$result->param('dataset')."\n";
    }
    else{
    print COMMANDS "open $tmpdir/".$result->param('dataset')."\n";
    }

    print COMMANDS "ls $lsargs $tmpdir/".$result->param('dataset')."-ls".
    $result->param('reweight')."-data $tmpdir/".
    $result->param('dataset')."-ls".
    $result->param('reweight')."-residuals\n";
    close COMMANDS;
```

```
    # Execute the gdms main program with this command script
    ‘$gdms -b $tmpdir/gdms-$unique.cmd‘;
    print STDERR "GDMSweb $$: GDMS return code is $?\n";
}

# Create a dataset in the temp location which is the output of a
# windowing operation
sub performWindow(){
    my($unique);
    local *COMMANDS;

    print STDERR "GDMSweb $$: Performing window operation\n";
    $unique = "$$-".time()."-".rand();
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param(’intype’) ne "temp"){
    print COMMANDS "open $datasets/".$result->param(’dataset’)."\n";
    }
    else{
    print COMMANDS "open $tmpdir/".$result->param(’dataset’)."\n";
    }

    print COMMANDS "window ".$result->param(’wtype’)." ".
    $result->param(’wsize’)." ".$result->param(’woverlap’).
    " $tmpdir/".
    $result->param(’dataset’)."-win".$result->param(’wtype’)."\n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    ‘$gdms -b $tmpdir/gdms-$unique.cmd‘;
    print STDERR "GDMSweb $$: GDMS return code is $?\n";
}

# Create a dataset in the temp location which is the output of
# interpolation operation
sub performInterpolation(){
    my($unique);
    local *COMMANDS;

    print STDERR "GDMSweb $$: Performing interpolation\n";
    $unique = "$$-".time()."-".rand();
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param(’intype’) ne "temp"){
    print COMMANDS "open $datasets/".$result->param(’dataset’)."\n";
    }
    else{
    print COMMANDS "open $tmpdir/".$result->param(’dataset’)."\n";
    }

    print COMMANDS "interp ".$result->param(’itype’)." ".
    $result->param(’isamplerate’)." $tmpdir/".
    $result->param(’dataset’)."-interp".$result->param(’itype’)."\n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    ‘$gdms -b $tmpdir/gdms-$unique.cmd‘;
    print STDERR "GDMSweb $$: GDMS return code is $?\n";
}

# Create a dataset in the temp location which is the output of the FFT
sub performFFT(){
    my($unique);
    local *COMMANDS;

    print STDERR "GDMSweb $$: Performing FFT\n";
```

```
    $unique = "$$-".time()."-".rand();
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
    print COMMANDS "open $datasets/".$result->param('dataset')."\n";
    }
    else{
    print COMMANDS "open $tmpdir/".$result->param('dataset')."\n";
    }

    print COMMANDS "fft $tmpdir/".
    $result->param('dataset')."-fft\n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
    print STDERR "GDMSweb $$: GDMS return code is $?\n";
}
```

Code: gdms.pl

Appendix C. Installation

To install the **GDMS Internet interface**, simply place it within your web servers CGI executable directory. Then work through the various options listed in the configuration file, and ensure that they are correct for your system. The most commonly incorrect settings are the paths to locations such as the temporary directory. These may be set to anything you desire, with the limitation that the plots directory *must* be accessible to the web server, as the images are served directly from this folder.

Note that no compilation of the **GDMS Internet interface** is required, as Perl is an interpreted language.

It is also important to ensure that all permissions are set correctly on the new directories to ensure that the web server remains secure. Consult your local system administrator for assistance.

Appendix D. A sample configuration file

This appendix details a sample configuration file for the **GDMS Internet interface**. This file should be stored in the same directory as the **GDMS** CGI script. It should also be noted that the ".cep" configuration file that would normally reside in a user's home directory should also be in the same directory as the **GDMS** CGI script.

The easiest way to create a suitable ".cep" configuration file is to run **GDMS** as any user, configure it so that you are happy with the graph output, and then copy that .cep file to the right location. Don't forget to update the permissions so that the user the web server is running as has read and write permissions on the file.

The same gdms.config file is as follows:

```
##############################################################################
# Directories
##############################################################################

# This line defines where the templates are stored
$templates = "/home/httpd/gdms-templates/";

# This line defines where the datsets are stored
$datasets = "/home/httpd/gdms-datasets/";

# This line defined where the matrices for LS reweighting are stored
$matrices = "/home/httpd/gdms-matrices/";

##############################################################################
# HTML configuration
##############################################################################

# This line is simple HTML used for format the command entries, %s is the name
# of the command (including link HTML)
$commandentry = "<tr><td bgcolor=\"AAAAAA\">%s</td><tr>";

# This line is used for commands which aren't available...
$discommandentry = "<tr><td bgcolor=\"EEEEEE\"><font color=\"777777\">%s</font></td></tr>";

# This is used for selection lists (for instance datasets)
$selectstart = "List start";
$selectentry = "<LI>%s";
$selectend = "List end";

##############################################################################
# Caching configuration
##############################################################################

# This is the location of the plot cache -- it can get quite big
$plotcache = "/home/httpd/html/gdms-plots/";
$ploturl = "/gdms-plots/";

# Location for temporary files
$tmpdir = "/home/httpd/gdms-temp/";

##############################################################################
# Other
##############################################################################

# This is the root URL
$rooturl = "/cgi-bin/gdms.pl";

# This is the full path to the GDMS application
$gdms = "/home/httpd/gdmsbatch";
```

Code: gdms.config