

Design and Implementation of Web-Based Software Demonstrations

Francis J. Wright
School of Mathematical Sciences
Queen Mary and Westfield College, University of London
E-mail: F.J.Wright@QMW.ac.uk
<http://www.maths.qmw.ac.uk/~fjw/>

21 April 1999

CATHODE Workshop, Marseilles, May 1999

The public demonstrations that I have set up can be accessed via the URL <http://cathode.maths.qmw.ac.uk/>. (Development versions may also be accessible via <http://centaur.maths.qmw.ac.uk/>.)

The following are examples of the basic resources that are required to set up a web-based software demonstration:

Web documentation *WebMaster in a Nutshell*, *Programming Perl*, (also *CGI Programming on the World Wide Web*, *JavaScript: The Definitive Guide*), all published by O'Reilly. There is also online documentation on the web.

Web server O'Reilly WebSite, Apache, Microsoft Personal Web Server (all available as free versions).

HTML editor Microsoft FrontPage Express, Netscape Composer, Emacs (all free).

CGI processor perl (free). (A debugger is also useful!)

Software to demonstrate REDUCE / ODESolve, Maple / diffgrob2.

Package information The user manual and/or online help page as a basis for the demonstration.

Some understanding of the package!

When designing public demonstrations one should consider the following criteria.

The user model:

- Assume by default a fairly naive user, who may not be a professional mathematician, may know little about differential equations, and may have no experience of computer algebra. Such users need pre-defined examples to run, which produce useful (and ideally impressive) output without undue delay.

- For more expert users, who are nevertheless not familiar with the package, there should be more flexible facilities that allow users to try some of their own problems.
- The web page should provide brief self-contained documentation, with links to more complete information sources.
- The input format should be flexible and not tied too closely to any particular computer algebra system; the output format should (at least optionally) be as close to mathematical notation as possible.
- Users should be warned that a demonstration is not a network-based research facility and it may have very limited computational resources.

The transaction model:

- Each transaction is non-interactive, consisting of a single batch run with no stored state.
- An html form controls the input.
- Input options: procedure arguments via form fields; options via menus, check boxes or “radio buttons”. Normally one-dimensional input of mathematics.
- Output options: tty prettyprinted, T_EX (via techexplorer), MathML.
- The CGI processor can pre-process input and post-process output.

A web-based form interaction works as follows. A page containing the form is sent by a web server to a client browser. The completed form is sent back to the server and interpreted by a CGI (Common Gateway Interface) processor. This runs the problem and returns the output to the server, which sends it on to the client as a new document.

Some of the more important technical issues that need to be addressed are:

- the need for both a live and a development web site;
- I/O via pipes or temporary files;
- security against hacking;
- resource control;
- choice of CGI processor language;
- controlling simultaneous accesses;
- portability, e.g. Windows – UNIX, Netscape – Microsoft Internet Explorer;
- input validation – browser (CGI) and/or client (e.g. JavaScript).

I propose to describe in more detail the most recent demonstration that I have set up, which is for Liz Mansfield’s Maple package `diffgrob2`. I will show the html page containing the form that provides the user input, the perl CGI script that mediates between the web server and Maple, and the temporary files that I currently use for Maple input and output. I will also mention aspects of our REDUCE demonstrations, in particular the `ODESolve` demonstration, which accepts input using standard REDUCE, restricted Maple and “prime” representations of derivatives.