Scientific Computing with Freeware

Russell Standish
High Performance Computing Support Unit
University of New South Wales

June 3, 1998

1 Introduction

Scientific computing has a distinctively different flavour to mainstream or business computing. The particular needs of scientists and engineers have largely been ignored by the big software houses such as Microsoft. As a result, scientists and engineers have contributed their own solutions to the growing body of freeware, allowing other scientists to make use these tools for getting on with the business of doing research.

I maintain that the open systems/unix environment is the richest environment for developing and using scientific tools. This is partly an inherent bias (I have invested a good deal of effort learning how to use unix systems effectively, because the Windows and Mac environments of the early '90s were woefully deficient), and partly from first hand experience managing a generalist computer support team. However, many of the software tools mentioned in this paper have been ported to the Windows and Mac environments, as there are certainly many scientists who prefer those environments. Unfortunately, these ported tools may not work quite as well as in the open systems environment. There are, of course, some very good freeware tools available for the PC and Mac environments, however, I will concentrate on the Unix ones in this paper, as that is what I'm familiar with.

The best resource for scientific software is *Scientific Applications for Linux*, available on `http://SAL.KachinaTech.com`. This lists over 1500 applications (not necessarily scientifically oriented), with an easy categorisation that allows one to "drill down" to the application of interest. Each software is marked with the license type, whether public domain, shareware, GPLed or commercial. In this paper, I have tried to pick those applications that scientists ought to know about as standard tools. I do not use all of them, as not all tools are applicable to what I'm studying, and there is some overlap in functionality, so it is sometimes easier to figure out how to use a tool that you are familiar with, than to learn a new tool that may be more suitable.

I don't promote free software to the exclusion of commercial software — clearly it is better to use the most appropriate tool for the job. However, there are some distinct advantages in using free software:

- You can easily set up a comfortable and familiar environment on each computer you use. I regularly use about a dozen different computers, spanning five different operating systems. If I had to buy my software, the licensing costs would be prohibitive.
- High Performance Computers typically require optimisation of the "computational kernel" in order to take advantage of the high performance architecture. Clearly this is only possible if the source code is available, which is generally either impossible, or prohibitively expensive with commercial codes.
- Access to source code is also extremely useful for correcting bugs or enhancing the code. Contact with the authors through the internet allows those enhancements to be incorporated into the main product code, for the benefits of fellow scientists. For some situations, it is vital to check the accuracy of the code, as scientific conclusions may be invalidated by buggy software.

2 Plotting and Visualisation

Plotting data is one of the most vital activities to understanding scientific results. It is considered as a basic form of *Scientific Visualisation*, however it has low performance requirements and can be performed easily on the average workstation. For this task, I cannot recommend *Gnuplot* more highly. It is able to perform most types of 2D and 3D plotting. It has a convenient and flexible scripting language. It also has a large selection of output drivers, including X11, Tektronix (for those unlucky enough not to have an X-window display), LaTeX and postscript. The encapsulated postscript driver is particularly convenient for including the plot directly into manuscripts.

For more sophisticated visualisation needs, a wide range of useful software is available. For example, *vis5d* is designed to plot up to 5 independent vector fields (gridded data) using animation and colour as the 4th and 5th dimensions. There is a gigantic visualisation workbench called *Khoros*, which is similar in concept to the commercial *AVS*.

3 Computation

Most computational scientists will need to do coding at some points during their research. The most commonly used computer language used is Fortran, although C and C++ have increased in popularity in recent years. C++ is

particularly interesting, because of its operator overloading feature means that codes can look closer to the original mathematical specification. Fortran90 does include operator overloading, however the mechanism doesn't seem to be quite as convenient or as flexible as that of C++.

The GNU suite of compilers is an excellent development environment, and works well with the gdb debugger, and with the GNU Emacs editor. The debugger can be run as a subprocess of the editor, allowing a seamless integration between editing, compiling and debugging. The debugger may also be run with a GUI interface, called xxgdb. Unfortunately, GNU doesn't include Fortran90, but does have a Fortran77 compiler. Fortran90 compilers are available commercially that work as front ends to the C compiler. GCC also comes with a substantial C++ library, including useful mathematical functionality such as complex arithmetic and infinite/arbitrary precision arithmetic.

There is a substantial body of public domain code for numerical libraries available through *Netlib*. The main libraries consist of *BLAS* (Basic Linear Algebra Subroutines), *LAPack* (Linear Algebra Package), *FFTPack* for Fast Fourier Transforms and *Slatec*, which includes the above libraries, as well as additional routines for things like integration. BLAS is a computational kernel for Linear Algebra routines. There is a public domain implementation in Fortran77, however manufacturers of high performance computers supply their own optimised versions written in assembler. There is also a public domain Pentium optimised BLAS project. By linking code that uses linear algebra routines — eg solving linear equations or finding eigenvalues — to LAPack, and the to optimised BLAS, the linear algebra will run close to the theoretical speed of the machine.

4 Vector/Matrix Toolboxes

Mathematicians have often used a toolbox of numerical techniques called *Matlab* to prototype mathematical algorithms. It consists of a command interpreter, with primitives to handle the construction of vectors and matrices, perform numerical algorithms such as LU decomposition or eigenvalue finding, and plotting. There are a number of such toolboxes available as freeware, the closest one to Matlab being *Octave*, a GNU product. Another similar freeware product is *RLab*. The advantage of these two products is that they use LAPack and FFTPack for performing the algorithm steps, so can be used quite efficiently on high performance computers by linking to optimised versions of these libraries.

Statistical analysis is another area where specific toolboxes are available. *xldas* is an X-windows versions of *ldas* (Lies, Damn Lies and Statistics). This has convenient means of manipulating data sets and running statistical tests such as ANOVA and Linear Correlation. Another such system is *Vista*, which appears to me more oriented to teaching than research. These programs have a spreadsheet-like character, which perhaps simplifies their use. However, as a scientist, if you

are already using a toolbox like Octave/RLab, or a Computer Algebra system, then you are probably better off using these systems for statistical analysis.

5 Computer Algebra

Computer Algebra involves manipulating mathematical strings that represent algebraic formulae. There are two main commercial Computer Algebra packages competing on the market, *Mathematica* and *Maple*. These largely differ by programming models offered — Mathematica offers an extremely flexible programming environment, with procedural, functional, declarative (à la Prolog) and object oriented programming styles supported. Maple, on the other hand is largely procedural with some functional support. However, in practice, the range of problems solved by the two systems are similar.

There are also some freeware computer algebra systems. One of the simplest is GNU Calc, which runs inside Emacs, using the elisp programming language. Sophisticated programming is not really supported, unless you wish to do detailed lisp programming, however one can specify rewrite rules (a sort of functional programming) to automate calculation. GNU Calc will drive Gnuplot for graphical output, so can easily be considered an extension of Gnuplot. GNU Calc is meant more to be of a calculator program with symbolic capabilities (it happens to use Reverse Polish Notation), than a full strength mathematical tool.

MuPaD, on the other hand, seems to be that full strength system, although being freeware, it is not as comprehensive as Mathematica or Maple. Over time, hopefully additional functionality will be added to it by users of the system.

Computer algebra systems can perform all the functionality of the Vector/Matrix toolkits mentioned in the previous section. However, because operations are performed symbolically, it is unlikely that they are as efficient for numerical problems.

6 Publication

There is no space for debate here on the relative merits of WYSIWYG word-processors versus document processors, but take a look at:

`http://parallel.acsu.unsw.edu.au/rks/docs/ps/texnolog.ps.gz`.

I just wish to say the best system for scientific (especially mathematical) publication is TEX/LATEX, without parallel. It is also extremely fortunate that LATEX is public domain. There are a number of ancillary pieces of software that go with LATEX that are worth knowing about:

Editor Macros If you are using GNU Emacs (as I do) then the best editing environment is provided by the AUCTEX macro package. An example fea-

ture of this package is two key codes for entering mathematical symbols eg 'a inserts α into the text. If you are MicroEmacs, there is an alternative package (originally written by myself, but now taken over by Zdenek Sedlacek) called LaTEmacs. There is also a GUI symbol palette called X-symbol.

ispell Free spelling checker from GNU — either use standalone, or from within GNU Emacs.

BibTEX Bibliographic database system integrated with LaTeX.

PSTricks A set of macros for performing special effects with Postscript. Examples include rainbow coloured letters, and text that follows an arbitrary path (eg a circle). A particularly useful part of the macro package relates to drawing networks - there are macros for drawing nodes and then specifying connections.

epsf style This macro is used for including *Encapsulated Postscript* files into the LaTeX file. These files may be produced by other software, eg Gnuplot or xfig.

xfig A fairly simple drawing editor, but useful for quickly drawing diagrams.

LaTeX2HTML This software (written in Perl) converts LaTeX documents to an HTML document tree. LaTeX constructs (eg mathematics) not directly representable in HTML are converted into GIF files, and turned into inline images.

ghostview An onscreen postscript viewer. Essential if you work in the TeX+Postscript model. Ghostview is based on the ghostscript interpreter, which may also be used for printing postscript files to non-postscript printers such as HP Deskjets and Canon Bubblejets.

CTAN Comprehensive TeX Archive Network. This is a network of ftp sites that distribute TeX related software and macros. Most of the above software is available from CTAN.

7 Internet Resources — Australian Mirrors where possible

Scientific Applications for Linux `http://SAL.KachinaTech.com`

GNU Archive `ftp://archie.au/gnu`

Sunsite Linux Archive `http://sunsite.anu.edu.au`

Netlib `ftp://netlib.uow.edu.au/netlib`

Comprehensive TEX Archive Network ftp://ftp.cs.rmit.edu.au/tex-archive

Linux Documentation Project http://www.wsc.monash.edu.au/LDP

X-symbol http://www.fmi.uni-passau.de/ wedler/x-symbol/index.html