APPLICATIONS FOR THE FREEBSD OS IN MECHANICAL ENGINEERING II

Pedro F. Giffuni

M. Sc. Industrial Engineering, University of Pittsburgh, Pittsburgh, PA, USA
Mechanical Engineer, Universidad Nacional de Colombia, Bogotá, Colombia

FreeBSD, a freely available UNIX-like Operating System, has evolved in the last 7 years to become an excellent platform for Mechanical Engineering related applications. The article presents a review of several applications, in particular CAD and FEA tools, that have been packaged for use in this OS and presents some guidelines for their application and use.

Keywords: FreeBSD, UNIX, Computer Assisted Design, Computer Assisted Engineering, Finite Element Analysis.

I. INTRODUCTION

During the first Colombian National Mechanical Engineering Congress an article [1] was presented showing some of the features available in the FreeBSD Operating System, along with some basic engineering applications that were available at the time. Due to the highly dynamical nature of engineering software development such article was meant to become obsolete very soon: in year 1999 the "opensource" movement, that provided numerous free software applications along with the source code used to build them, was in it's infancy and a myriad of new applications have been made available.

For the second part of this article we follow the general guidelines of the first article: while providing a starting point for the neophyte mechanical engineer this article doesn't pretend to be an exhaustive review of all existing applications nor is it a tutorial on FreeBSD or any specific application being presented. More than a mere presentation of a series of interesting technologies, we do try to focus on how to make these individual utilities interact between them and gain some coherence.

II. FREEBSD: AN ADVANCED FREE OPERATING SYSTEM

FreeBSD is a UNIX-like Operating System (OS), based on the original source code releases from the University of Berkeley, which is freely available on the Internet. FreeBSD is relatively well known due to its networking support and it's very efficient Virtual Memory system; late versions of FreeBSD have focused on security, and much improved performance with multiple processors. FreeBSD is not a mainstream desktop OS, most of its users are Internet Service Providers and websites that want a cheap and efficient way to get their job done. FreeBSD is not the only freely available OS either, and the growing popularity of Linux and OpenSolaris make them viable options to try and test new applications being developed. Our initial involvement with FreeBSD was motivated by the excellent documentation [2] and the great performance exhibited in networked environments but still nowadays there are important reasons to keep using it.

As the name suggests, FreeBSD is free; both in price, definitely an important factor for a young engineer trying to start his/her own business, and in openness as the C source code is available with basically no license restrictions. Another important factor to consider in FreeBSD is the ease with which applications can be adapted to new technological advances: most of the applications covered in this article were built natively in a machine platform with a 64-bit processor and multiple cores, a configuration that is not expected to be properly supported by the most common commercial OS this year. A final, but surely not the last, argument towards adopting FreeBSD is the community behind it: the FreeBSD ports team does extensive validation of the applications and the pre-packaged software included in FreeBSD's distribution and the OS is usually well known and supported by software developers. Up to the day this article was written FreeBSD is known to build more than 15000 applications for its distribution.

According to a recent study sponsored by the US Department of Home Security [3], open source applications have better quality than their commercial counterparts. While all the applications discussed in this article have been tested on FreeBSD and have sufficient quality to be included in the "ports tree", the general Mechanical Engineering community is welcome to try them on other Operating Systems and will probably not be disappointed.

III. GENERAL ENGINEERING APPLICATIONS

There is a huge number of useful applications in FreeBSD that can be used by engineers: of course two well known complete environments, **KDE** and **GNOME**, are available along with web browsers, text editors, and general utilities. A quick search shows that applications like **OpenOffice** – a complete office suite -, **Scilab** - a math environment -, and **SPICE** - an electronic circuit analyzer – work fine in FreeBSD. Most interesting for engineers are the Math, Science and CAD sections of the ports tree that total 498 packages. Of course an extensive review of 498 packages is not possible in a small article so we will only mention a, perhaps unfair, subset of applications in three different categories: basic libraries, drawing/CAD tools, and general FEA solvers and tools, Most of the selected utilities have been "ported" to FreeBSD over an interrupted period of 7 or 8 years by the author of this article

Before getting into the main material that will be considered, it is worthy to mention two other applications that can be extremely useful. The National Institute of Standards and Technology designed **OOF** – The Object Oriented Finite Element Analysis of Microstructures – a utility that can take a micro photography in PPM format and calculate macro-properties like material strength and temperature resistance. OOF was named one of the "25 Technologies of the Year"[4] by Industry Week Magazine. **MBDyn** is a multi-body dynamics analysis system [5] – written at the *Politecnico di Milano,* in Italy.

IV. BASIC LIBRARIES

Software libraries are computer code archived in such a way that they can be used by other libraries or programs. Libraries are frequently not as visible to the common user but are important building blocks and very commonly the base for all calculations. Libraries are either static (usually with a ".a" extension) if they are included exclusively during the linking phase or dynamic (usually with a ".so" extension) if they need to be loaded during execution time. An additional consideration that may be important is the programming language used to build the library: in FreeBSD as in many other platforms the Application Binary Interface is different if the program is being built with C or with Fortran77, and recent implementations of Fortran99 require special options to be interoperate with older Fortran77 libraries. Java™ is also certified on FreeBSD.

Probably the most important math library is the standard Basic Linear Algebra Subprograms, **BLAS**.[5]. BLAS is used by many programs, commercial and free, to provide basic matrix and vector operations. The library is so popular that many UNIX resellers include their own heavily optimized version. In addition to the standard version, FreeBSD gives the option to use the Automatically Tuned Linear Algebra Software, **ATLAS** [7], which implements the same interface as BLAS but attempts to do more aggressive hardware dependent optimizations. An informal performance test, provided by **LAPACK**[8] showed little advantage when using ATLAS over the standard BLAS for small problems. A much bigger performance differences can be expected for complex problems. ATLAS is used by many commercial packages including MAPLE™, MATLAB™ and MATHEMATICA™.

Linear Algebra operations like solving complex systems of linear equations are an important requirement for most engineering packages. Several new algorithms and methods have been implemented in universities and organizations that have made the results freely available. Among them some notable implementations available in the math section of the ports tree are: **UMFPACK**[9], **SuperLU**, **SPOOLES**, **TAUCS** and **MUMPS**. Most of these applications have support for multithreading and can optionally be compiled with support for the Message Passing Interface (MPI) for clustering.

Each of these libraries has some particular feature that is required by some of the solvers available in the VI section of this article. It's difficult to know offhand which solver to use unless extensive benchmarking with the particular problem type to solve is made. Perhaps the best general purpose solver is the "MUltifrontal Massively Parallel sparse direct Solver" MUMPS [10], which is the latest state-of-the-art parallel solver, developed with partial funding from the European Union ESPRIT IV Research Project..

Partitioning software is another important operation for engineering applications: it consists of dividing huge problems into smaller ones in such a way that they can be solved more effectively in a cluster of processors. Some well known packages used to do this are **Chaco**, written and distributed by Sandia US National Laboratory, **METIS** by George Karypis, **Scotch** from the *Laboratoire Bordelais de Recherche en Informatique* (LaBRI), and the Padenborn Ordering package PORD, currently included in MUMPS..

V. CAD AND GENERAL MODELING APPLICATIONS

Computer Aided Design CAD is a term frequently misused as referring to software that typically only draws but doesn't assist in other ways in engineering projects. FreeBSD has several utilities that support key graphic formats but there are currently no tools that match the capacities and ease of use offered by commercial packages like ProEngineer™ or SolidWorks™. Furthermore, a growing number of companies only works with proprietary formats and require their providers to use that same application. For this situation, the only option on FreeBSD is using a software emulation layer, like the one available for Linux, which is not the ideal solution but often, just works very well. An example of this situation is **GiD**™, a commercial application developed by the *Universidad Politécnica de Cataluña´*s CIMNE, in Spain.

The CAD section has several tools to work with 2D technical drawings like **QCad** and **Varkon**; the later is a parametric tool with its own programming language that can be very useful as a general purpose tool. An option which has been recently been made available from the US Army Research Laboratory is BRL-CAD.

BRL-CAD is a powerful solid modeler. While **BRL-CAD** was referenced in the (older) first part of this article [1], only recently the US Army made available the source code for this application, which has made it possible to find and fix some security vulnerabilities and further enhance the code. BRLCAD's development started in the late 1970's with the desire from the US Army to provide a set of tools that would assist in the design and study of combat systems and environments. The package has modules that aim to support ballistic and electromagnetic analyses (some of these not yet publicly available). The toolkit basically consists of many small tools to do format conversions, an interactive geometry editor "mged", a parallel rendering engine, and image processing tools. Figure 1 shows a photorealistic image generated by ARL's ADRT/RISE tool available in BRL-CAD.

Figure 1. Stryker ICV w/ Slat Armor rendered with ADRT/RISE (generated by US ARL)

A drawing in BRL-CAD is a database of elements built following a general representation called Constructive Solid Geometry CSG, by defining solids primitives like spheres, cubes, and plates and making Boolean operations like union, intersection and subtraction on them. CSG permits many interesting features like providing perfect representations of spherical objects. Unlike BRL-CAD, most of the commercial tools available today use Boundary Representation (BREP) which defines the volumes by their boundaries, something that makes it difficult to do accurate conversions from those formats to BRLCAD. Nevertheless, BRL-CAD can import files from standard formats like STEP and STL which makes it possible to interact with many commercial packages.

Modern modeling software, and particularly software that will be exchanging information with Finite Element Analysis (FEA) tools are expected to be able to generate meshes from CAD files. Two important tools that can do exactly this are GMSH and NETGEN.

NETGEN is an excellent 3D tetrahedral mesh generator designed and implemented by Joachim Schöberl as part of the larger "Numerical and Symbolic Scientific Computing" Project from the Johannes Kepler University Linz. NETGEN can read files in STEP and IGES exchange formats and can generate meshes to export to well established proprietary formats like Nastran, Abaqus, Elmer, and to more academic formats like **FEAP**[11].

NETGEN is relatively straightforward to use: just define the refinement level and mesh. Figure 2 shows a tetrahedral mesh.

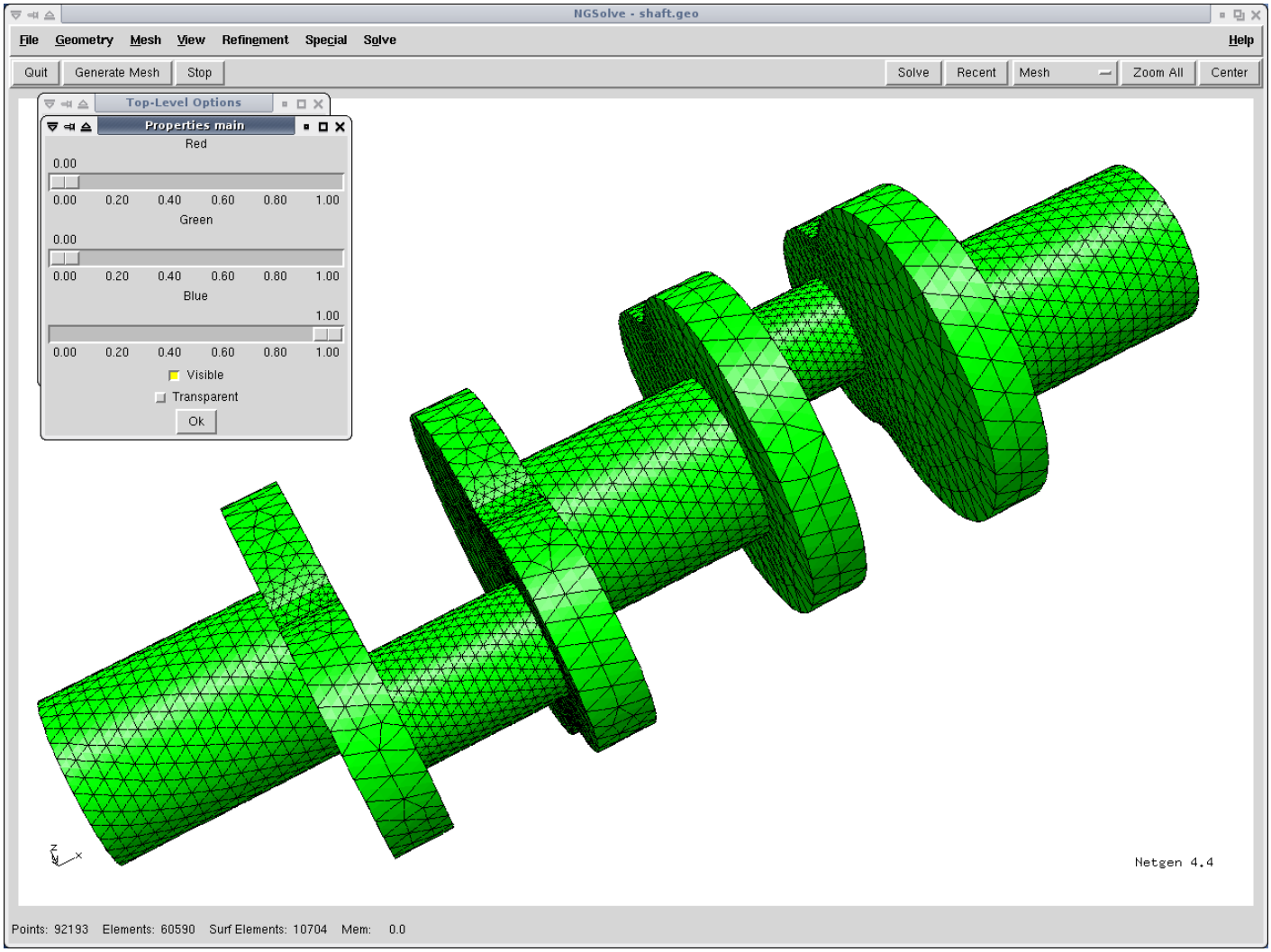

Figure 2. Example shaft mesh generated with NETGEN

A second important general purpose meshing utility we must mention is GMSH, developed by Christophe Geuzaine and Jean-François Remacle This package has the disadvantage of not offering many conversion utilities to its geometry format, although an experimental converter from DXF format is available. The meshing options for this package are very flexible: while it has it's own Delaunay implementation the 3D mesh generator used by NETGEN is also included and additional meshing libraries are available. On FreeBSD support for 2D meshes using Triangle [12], and Tetgen, another tetrahedral 3D meshing tool, are included but not enabled in the package due to licensing restrictions.

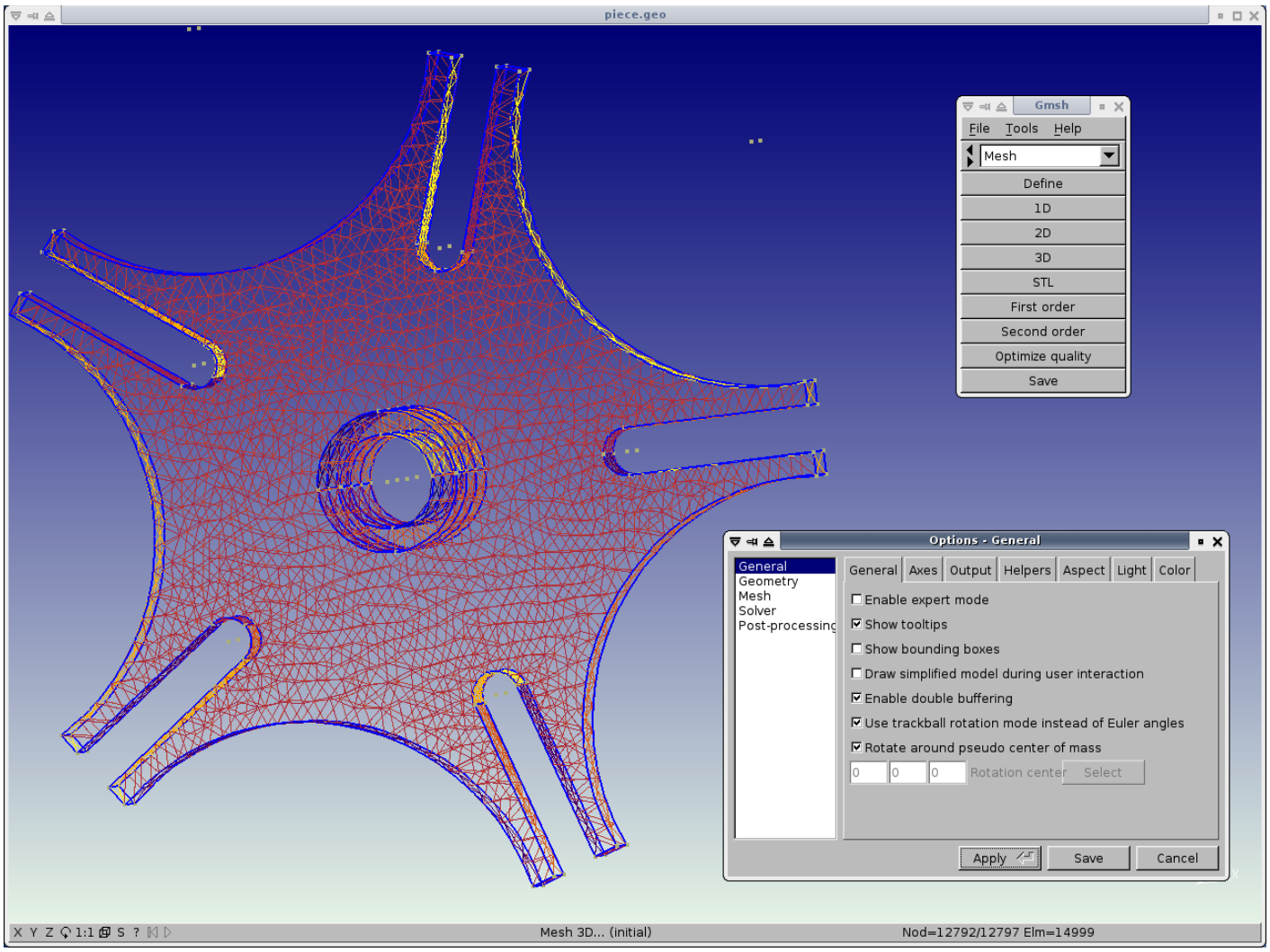

Figure 3: An example of a part meshed using GMSH

Similarly to NETGEN, GMSH is also useful as a postprocessor by loading the results.

VI. FEA AND SOLVER TOOLS

While powerful, easy to use, CAD packages and general proprietary file manipulation tools are still not commonplace in free software repositories, the situation for Finite Element tools is different: There are several extremely high quality packages that permit Finite Element Analysis with performance levels equal or superior to many commercial packages. One critical factor to be able to use these tools is, precisely, the possibility of transferring CAD files from commercial packages to the powerful solvers available.

The main point of entry for solving any FEA problem is through the use of a meshing utility and in cases where the solver includes its own pre-processing and post-processing utility, the best results are usually obtained using the native tool. Of course, learning to use the generic meshing utility like NETGEN and GMSH can save a lot of time.

With a mesh generated from a CAD file or from a geometric description file, the next step is to define constraints, loads and general material conditions, the format of which varies according to the specific FEA or solver tool that will be used. The final step is post-processing the files, something that basically implies loading the results and displaying them graphically.

Among the many interesting solvers, the three most notable discussed here, are CalculiX, Code Aster and Elmer. In the order in which they were ported:

CalculiX[13] started as a three-dimensional Structural Finite Element program, developed by Dr. Guido Dhondt and Klaus Wittig, employees of MTU Munich, an Aero Engine manufacturer in Germany. Recent versions of CalculiX have been extended to support thermo-mechanical and dynamic problems, buckling, heat transfer, and LaPlace and Helmholtz problems by analogy. Future versions will include contact with and without friction.
.
CalculiX consists of two utilities: GraphiX (*cgx*), a pre- and postprocessor that can work with several other freely available solvers (**OpenFOAM** and **ISAAC-CFD** to name just two), and CrunchiX (*ccx*), a solver that supports the same style formats as ABAQUS™. The default solver used in *ccx* is SPOOLES, but experimental support for TAUCS, which has an out-of-core solver, is also available. The FreeBSD port by default attempts to use ATLAS instead of BLAS, and enables TAUCS and multithreading, which also means it will try to use all the processors available in the machine. CalculiX is known to provide the same results as leading commercial packages but is faster

Code Aster is a powerful FEA package released to the public in 2001, after twelve years of internal development by *Electricité de France* EDF R&D. It is a fully multiphysics capable package, and includes its own solvers for linear, nonlinear and modals problems. Currently Code Aster supports the following analysis types: standard, decomposition into Fourier modes, substructuring, modal superposing, adaptive meshes, sensitivity calculation, fitting and optimization and mechanical reliability analysis. It also has a material data catalogue and a library of Finite Elements with 360 elements.

Most of the documentation for Code Aster is in French but work is being done to localize it; and the package interacts very well with other applications that can be used without learning French. GMSH is one of the preprocessors that can be used by this package by exporting meshes in MED format. Code Aster uses its own modified version of .METIS, and can optionally use SCOTCH as an ordering algorithm. It also implements its own command language, it has a specialized editor called EFICAS and a Job Control utility called ASTK (see Figure 4).

Code Aster is being actively developed by EDF, which has been very effective in integrating third party software packages and in near future major packages with full Graphical User Interfaces for preprocessing will be available. One of the particular features that has been worked in FreeBSD is the optional use the powerful MUMPS package to do the calculations. This gives excellent, consistent, performance and provides more sensitivity information[14]. The default solver is known to compare very well with the parallel versions of the popular NASTRAN™ code.

Study cases for more that 70 industrial applications of Code Aster are available through *Aster-Echos*, a three-monthly journal, including: behavior in a rotor turbine component, wear in tanks under mixed loads and corrosion, seismic vibrations in civil engineering components and much more. The 26th of June 2006, Code Aster won the "*Lutèce d'Or*" award for the best free software project developed by a group.

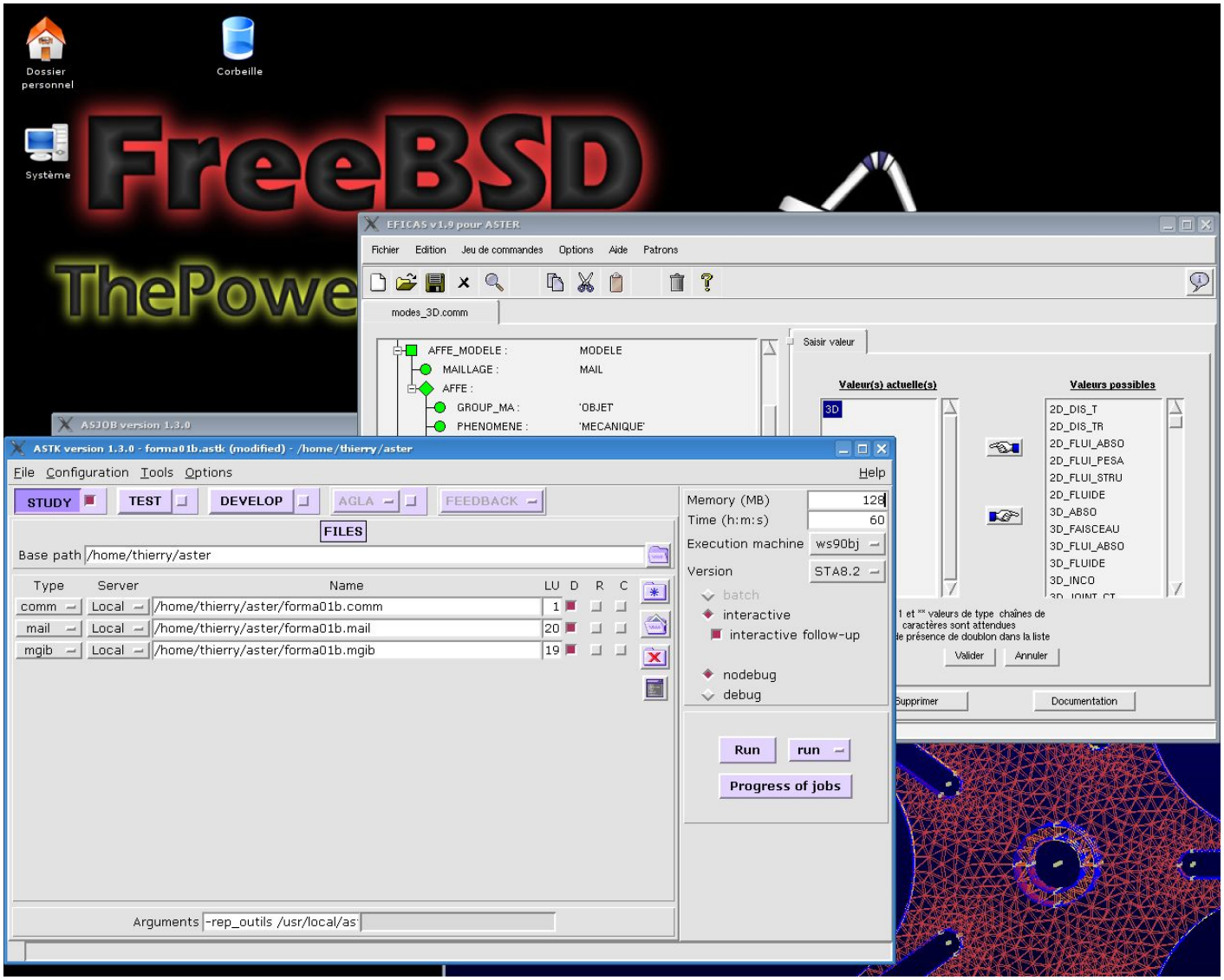

Figure 4.ASTK and other utilities from Code Aster

The latest FEA package ported to FreeBSD is **Elmer**[15], a multi-physics computational tool developed by the Finnish IT center of science in collaboration with several Finnish universities. Elmer was initially only a CFD tool but the methodology used proved useful also for fluid dynamics, heat transfer, structural mechanics and electromagnetics.

Elmer is written in Fortran 90, a practice very common in the scientific community, which presented some problems in FreeBSD that ended up being solved satisfactorily after providing Fortran90 versions of BLAS, LAPACK, along with a customized version of UMFPACK. Using a Fortran90 version of the libraries had as a side effect the temporary limitation of not being able to use BLAS, so in an attempt to alleviate the situation it is possible to activate additional optimizations in the Fortran 90 version of the math utilities. Some preparations were also made to support MPI in future builds.

Elmer consists of several tools called *ElmerFront*, a preprocessor package, *Elmersolver*, the FEM solver, *ElmerPost*, a postprocessor, and *ElmerGrid*, a program for mesh manipulation. Even though 3D problems can be analyzed in Elmer, the problems solved are more frequently 2-dimensional in nature, something that is confirmed by the lack 3D support in *ElmerGrid* Elmer's documentation encourages the use of NETGEN, but personal experience suggests the use of *ElmerFront*, particularly for setting constraints, loads, and building the model.
The algorithms used by Elmer are very well documented and an important feature in Elmer is the possibility of writing customized solvers based on Partial Differential Equations. Some typical examples of problems solved with Elmer include: temperature

distribution in a sculpture, deflection of an elastic shell, compressible flow – Von Karman Vortices, Transient gas flow due to moving boundary, Heat Transfer in MCVD of synthetic glass tubes and Nozzle flow.

VII. CONCLUDING REMARKS

In the last decades technology has become a critical asset for industrial research and development. An important question nowadays would be to wonder exactly what constitutes "acquiring technology". Is it valid to say someone has acquired technology after he paid for its use without knowing how it operates? Applications like the ones presented here completely change the paradigm by providing tools that have no initial cost but that openly provide all the information about how they work. Admittedly few mechanical engineers need to understand the inner workings of a CAD or CAE tool, however having access to that information may very well make the difference between a country that buys technology and a country that develops it.

Few commercial applications offer the source code at all; one of the few applications that makes it available is NASA's Structural Analysis program NASTRAN™, and even in this case the application is distributed with important limitations: the source code used to be available only for US citizens, and the source code is, of course, not available for free. Yearly licenses of NASTRAN can be obtained through the Open Channel Foundation for a price of 10,000 US$ (for US citizen a different license that costs 3000 US$ less is available). This doesn't mean at all that commercial utilities are undesirable; commercial tools like solid modelers and integrated CAE environments offer features that are very welcome and usually are accompanied by excellent technical support that is worth every cent.

Most of these applications required funding for their development: the process by which all the packages discussed in this article were developed and were ported to FreeBSD involved the effort of many developers and programmers. Perhaps one of the most interesting things to realize is that while all these applications are available for free, and most of the development ends up being not-for-profit, there is always an added economical value behind them. Making available these tools is good for developers as they get feedback and in many occasions added development for their tools and, on the other hand, for a recently graduated (read "usually not wealthy") engineer it really doesn't make sense to spend a huge amount of money in commercial utilities when a set of freely available tools can provide the same result.

Some specific tools are easier to find and use than others but progress will continue, of course, as long as Universities, research centers and companies perceive an added value in making their tools available for a bigger audience.

SPECIAL THANKS

The author wishes to thank Thierry Thomas for porting Code Aster to FreeBSD and for his sustained effort and patience in so many other ports, Maho Nakata for his guidance while porting CalculiX and for the long discussions we had about what to do with certain problematic math library, and the FreeBSD developers for making such an awesome development platform a reality.

REFERENCES

[1] Pedro Giffuni, *Aplicaciones del Sistema Operativo FreeBSD en Ingeniería Mecánica, Memorias del Primer Congreso Nacional de Ingeniería Mecánica, Tomo II*, Bogotá, Nov. 1999.
[2] Greg Lehey, The Complete FreeBSD, Third Edition, Walnut Creek CDROM, 1999
[3] Coverity Inc., Measuring Software Quality: a Study of Open Source software, 2006
[4] Industry Week, National Institute of Standards & Technology Gaithersburg, Md. OOF (Object-Oriented Finite Element) Computer Program, December 1999.
[5] P. Masarati, M. Morandini, G. Quaranta and P. Mantegazza, An Open-source multibody analysis software, paper and poster presented at MultiBody Dynamics 2003, July 2003, Lisboa Portugal
[6] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, *Basic Linear Algebra Subprograms for FORTRAN usage*, ACM Trans. Math. Soft., 5 (1979), pp. 308--323.

[7] R. Clint and Whaley and Antoine Petitet, Minimizing development and maintenance costs in supporting persistently optimized BLAS, Software: Practice and Experience, Volume 35 Number 2, February 2005.
[8] E. Anderson, Z. Bai, C Bischof, S Blackford, J Demmel J Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK User's Guide, Society of Industrial and Applied Mathematics, 3rd Edtition, Philadelphia, USA 1999.
[9] T. A. Davis, Algorithm 832: UMFPACK - an unsymmetric-pattern multifrontal method with a column pre-ordering strategy, ACM Trans. Math. Software, vol 30, no. 2, pp. 196-199, 2004.
[10] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent and S. Pralet. Hybrid scheduling for the parallel solution of linear systems, Technical report INRIA RR-5404 also ENSEEIHT-IRIT RT/APO/04/4 and LIP report RR2004-53. Improved version appeared in *Parallel Computing* 32 (2): 136-156, 2006.
[11] O.C. Zienkiewicz and R.L. Taylor, The Finite Element Method, 6th ed., Vols. 1 and 2, Elsevier, Oxford, 2005
[12] Jonathan Richard Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, in ``Applied Computational Geometry: Towards Geometric Engineering" (Ming C. Lin and Dinesh Manocha, editors), volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996.
[13] Dhondt, G. *The Finite Element Method for Three-Dimensional Thermomechanical Applications*, Wiley, 2004
[14] O.~Boiteau and F.~Huelsemann and X.~Vasseur, Comparison of the linear algebraic solvers {M}umps and the multifrontal solver of Code ASTER, Contract Report CR/PA/06/11, CERFACS, Toulouse, France, 2006,
[15] Mikko Lyly, Juha Ruokolainen and Esko Järvinen. ELMER - A finite element solver for multiphysics. CSC-report on scientific computing 1999-2000, pp. 156-159