

WORKING WITH ENGINEERING APPLICATIONS ON FREEBSD

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 greatly in the last 10 years to become an excellent platform for Mechanical and Electrical Engineering 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 1999 Colombian National Mechanical Engineering Congress an article¹ 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 since then a myriad of new applications have been made available.

This article is based on the continuation of the original article; it was updated to provide the new advances and tools ported for FreeBSD. We do follow the general guidelines of the first article: this article doesn’t pretend to be an exhaustive review of all existing applications nor is it a tutorial complete 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 coherence. Our experience will provide an insight into the decisions taken while porting the applications to FreeBSD, and how to take advantages of the specific features the environment provides.

II. FREEBSD: AN ADVANCED FREE OPERATING SYSTEM

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. Our initial involvement with FreeBSD was motivated by the excellent documentation² and the great performance exhibited in networked environments which are today very good reasons to choose 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 was not common to find a few years ago and that is still not fully supported by commercial software providers. It is considered that the war for high speeds in modern processors is practically over due to the heating problem overclocking involves, and that new advances will be in the are of multiple processors. Software is not really well prepared for this situation so the different approaches required the extra flexibility of having the source code available.

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 19000 applications for its distribution.

According to a recent study sponsored by the US Department of Home Security³, 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 engineering community is welcome to try them on other UNIX-like 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: but perhaps most interesting are the Math, Science and CAD sections of the ports tree that total more than 700 packages. Of course an extensive review of 700 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 Electronics/Mechanical 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.

IV. BASIC LIBRARIES

Software libraries are computer code archived in such a way that they can be used by other programs or libraries. Libraries are frequently not visible to the common user but are important building blocks and are frequently the very 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.

The programming language used to build the library can be an important consideration: most of the math related program are built with Fortran and starting with version 7.0, FreeBSD is not shipping with a Fortran compiler in the base. Not having a Fortran compiler by default imposed two inconveniences: code compiled with the Fortran compiler now requires the shared libraries used by the compiler, which means adding a runtime dependency on the complete compiler collection package. It is now also the case that the linker doesn’t really know how to link the proper Fortran programs, for this reason when linking Fortran libraries it is now recommendable to use the Fortran compiler to link them. The compiler F90 compiler used in recent versions of the GNU Compiler Collection uses a different Application Binary Interface so linking with older Fortran77 libraries requires a special option (-ff77).

The Internet brought many new technologies and computer languages. While Fortran has been consolidating, there are newcomers that are starting to consolidate. Java™, in particular, is also getting some common use in applications, like OpenFOAM and Impact, so it is an important asset to have it certified on FreeBSD.

Probably the most important math library is the standard Basic Linear Algebra Subprograms, **BLAS**.⁴ 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 for their hardware. In addition to the standard version, most of FreeBSD’s ports offer an option to use the Automatically Tuned Linear Algebra Software, **ATLAS**⁵, which implements the same interface as BLAS but attempts to do more aggressive hardware dependent optimizations. ATLAS is used by many commercial packages including MAPLE™, MATLAB™ and MATHEMATICA™. By default in FreeBSD we use the shared version of the slower BLAS library because ATLAS libraries can’t replace the standard *libblas* library during runtime; ATLAS is also very hardware dependant and the package produced by the cluster don’t work well with different processors.

An informal performance test, provided by **LAPACK**⁶ showed little advantage when using ATLAS over the standard BLAS for small problems; much bigger performance differences can be expected for complex problems. The recommended solution, however, does not involve rebuilding your packages to use the benefits of a faster BLAS version. FreeBSD has a port of the non-commercial GOTOBLAS which is known to give excellent performance results. For good performance it is recommended to build the math/gotoblas port and then use *libmap.conf(5)* file to redirect all *libblas* calls to *libgotoblas*.

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. Many commercial applications in fact depend on these utilities. Among the notable implementations available in the math section of the ports tree are: **UMFPACK**⁷, **SuperLU**, **SPOLES**, **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.

It’s difficult to know offhand which solver to use unless extensive benchmarking with the specific problem type is made. Each of these libraries has some particular feature that is required by some of the solvers available in section VI of this article.

TAUCS, for example, has an excellent out-of-core solver that will not exhaust your memory on big problems. Perhaps the best general purpose solver is the “Multifrontal Massively Parallel sparse direct Solver” MUMPS⁸, which is the latest state-of-the-art parallel solver, developed with partial funding from the European Union ESPRIT IV Research Project. As the name hints it is specially optimized to be used in parallel processing applications but we also support the simpler SMP that can be used with Scilab.

Partitioning software is another important part of 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. Of these, METIS is the most commonly used but you can expect some problems with version 4.0 on 64 bit platforms (we will start using version 5.0 when it’s released).

Recent versions of FreeBSD’s packaging system are supporting a REGRESSION-TEST target.: this runs and is logged in the package-building cluster as stage 5 (after building but before installing and packaging): it is extremely useful as a basic certification that the package works as expected. Regression testing is extremely important when you try to trace what went wrong in a calculation: you can always make a mistake while you are modeling a problem but if everything seems right, but regression-testing can save you from finding out that one of the tools that your software is using is completely broken on your particular OS/hardware combination. We made an special effort to do some basic testing on some of the tools that have most package dependencies: lapack, libgmp4, gsl, hdf5. This is an area where we still have to work more with free software authors but it’s definitely an advance.

V. CAD AND GENERAL MODELING APPLICATIONS

Using the math libraries and the excellent math packages available for FreeBSD is not a difficult decision to take: not only we have fine ports of well known applications like Scilab and Octave, we also can run some pretty useful applications running under the linux emulation. You can expect floating point differences between Linux (extended precision), and FreeBSD (double precision), but, in general, using general purpose math applications can be considered safe. Latest versions of FreeBSD have also benefited very much of using a thread aware malloc(3).

The first really interesting Computer Assisted Design port that made it into FreeBSD is SPICE⁹, a circuit simulator that was designed in Berkeley by a team headed by Prof. Donald Pederson. The original package was written in FORTRAN and was extremely successful in the industry, with many commercial forks and happy users. The version in FreeBSD is known as SPICE3, the official rewrite in C of SPICE2. Spice has not been well received in the Linux distributions due to some minor limitations in the original BSD-like license and the customary restriction the US imposes to “politically incorrect” countries. The FreeBSD port includes many cleanups and buffer overflows, and some effort has been done to collect the interesting patches available on the net. We also keep to forks of this code: the ngspice project and jspice, which offer some new functionality over the official version. Several packages like Xcircuit are able to generate netlists that can be processed by spice after defining the types of analysis desired (nonlinear dc, nonlinear transient, and linear AC). Even when there have been other attempts to write free applications similar to SPICE, SPICE remains a standard, a must-have for Electronic Engineers.

The decision to use a CAD package or an specific engineering application, however, is never easy. It depends pretty much on the size and policies of your organization. Furthermore, a growing number of companies only work with proprietary formats and require their providers to use that same application.

In general the tools available in UNIX are not made to be used on an intensive GUI, but frequently require learning a command language. The big CAD applications used in Mechanical Engineering have intelligent GUIs and specifically depend on their own proprietary formats. FreeBSD has several utilities that support key graphic formats (OpenCascade and BRLCAD support STEP) but there are currently no tools that match the capacities and ease of use offered by commercial packages like ProEngineer™ or SolidWorks™. I had the chance to visit PTC’s technical support center in the UK in 2003, and the interesting surprise is that while they do offer a Linux version of ProEngineer to their costumers, none of the costumers with technical support contracts used it. Big costumers are also very conservative when adopting new Operating Systems and software upgrades. It is my personal opinion that the free software world will have difficulties getting into the desktop engineering field, but that a mixed environment using virtualization can be very successful.

While the term CAD is supposed to apply to any engineering design program, Drafting tools have grown so much in complexity and capabilities that in Mechanical Engineering we refer to them as CAD tools and designate solvers and other analysis tools CAE.

The CAD section has several basic tools like **QCAD** and **Varkon** that do their job rather well; the later is a parametric tool with its own programming language, the former is a 2D draft plotting tool. Both QCad and Varkon are excellent options for low level shops that require the basic functionality provided by the early AutoCAD™-like tools. Nothing really matches the powerful Solidmodelers available for that other commercial OS, but perhaps the most promising CAD application that has been recently been made available openly is BRL-CAD.

BRL-CAD is a powerful solid modeler. 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. Our modest contribution while porting this software has been in the mutual sharing of patches from the Utah Raster Toolkit, and by reporting some bugs and portability issues that have been fixed. The maintainer is an active developer of this tool.

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. Many mechanical features are difficult to draw in CSG and it tends to be difficult to do accurate conversions from other formats to BRLCAD. There is a continued effort to add NURBS support which would greatly improve the situation, BRL-CAD can, nevertheless, import files from standard formats like STEP and STL which makes it possible to interact with many commercial packages.

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)

In many cases you don't really have to do extra calculations for your projects. Frequently mechanical components are over-designed in such a way that they won't reasonably fail any time soon, in other cases real-life prototypes are much more trustable than computer models: in the case of Formula 1 cars, Computer Fluid Dynamics have shown to be in it's infancy and only secondary when compared to a experimentation in a wind tunnel. On other situations competition is so fierce that companies want to move to Computer Assisted Analysis as soon as possible or at least take it into account as a validation procedure.

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. We got permission from the *Universidad Polit cnica de Catalu a*'s CIMNE, in Spain, to include **GiD**TM, a commercial pre- and postprocessor in the ports tree, but two free tools that can also do exactly this are GMSH and NETGEN.

NETGEN is an excellent 3D tetrahedral mesh generator designed and implemented by Joachim Sch berl 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**¹⁰. NETGEN is relatively straightforward to use: just open you CAD file in a portable format and define the refinement level and mesh. NETGEN is an excellent complement to CalculiX Crunchix (ccx) since both can work with the Abaqus format. See Figure 2 for a typical mechanical part meshed using tetrahedral elements.

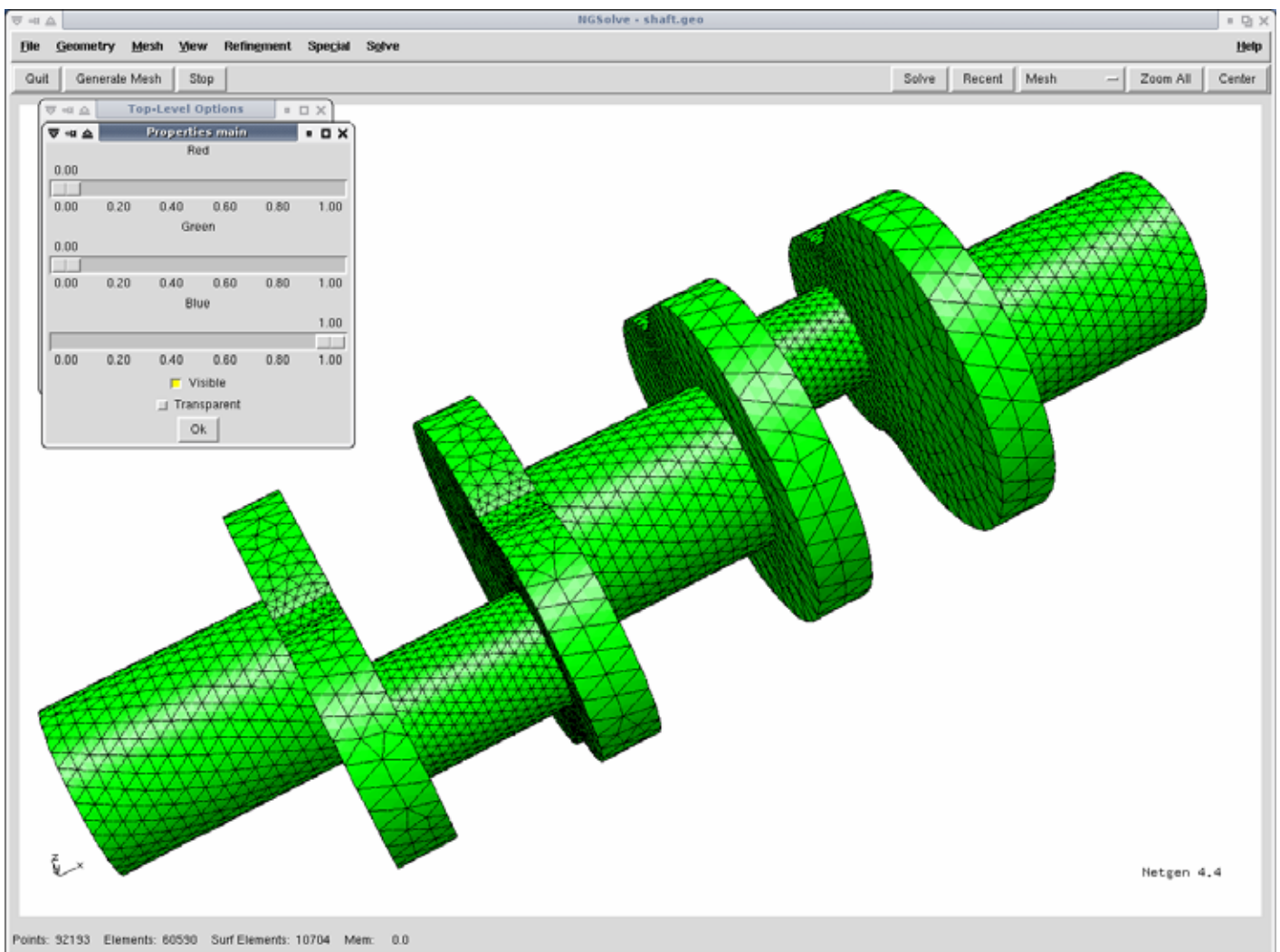


Figure 2. Example shaft mesh generated with NETGEN

A second important general purpose meshing utility we must mention is GMSH, developed by Christophe Geuzaine and Jean-François Remacle. This package has recently added support for STEP and other formats through OpenCascade. The meshing options for this package are very flexible: while it has its own Delaunay implementation, the 3D mesh generator used by NETGEN is also included and additional meshing libraries are available. On FreeBSD we added support for using Tetgen, another tetrahedral 3D meshing tool, and enabled it in the package after asking permission from the author due to licensing restrictions.

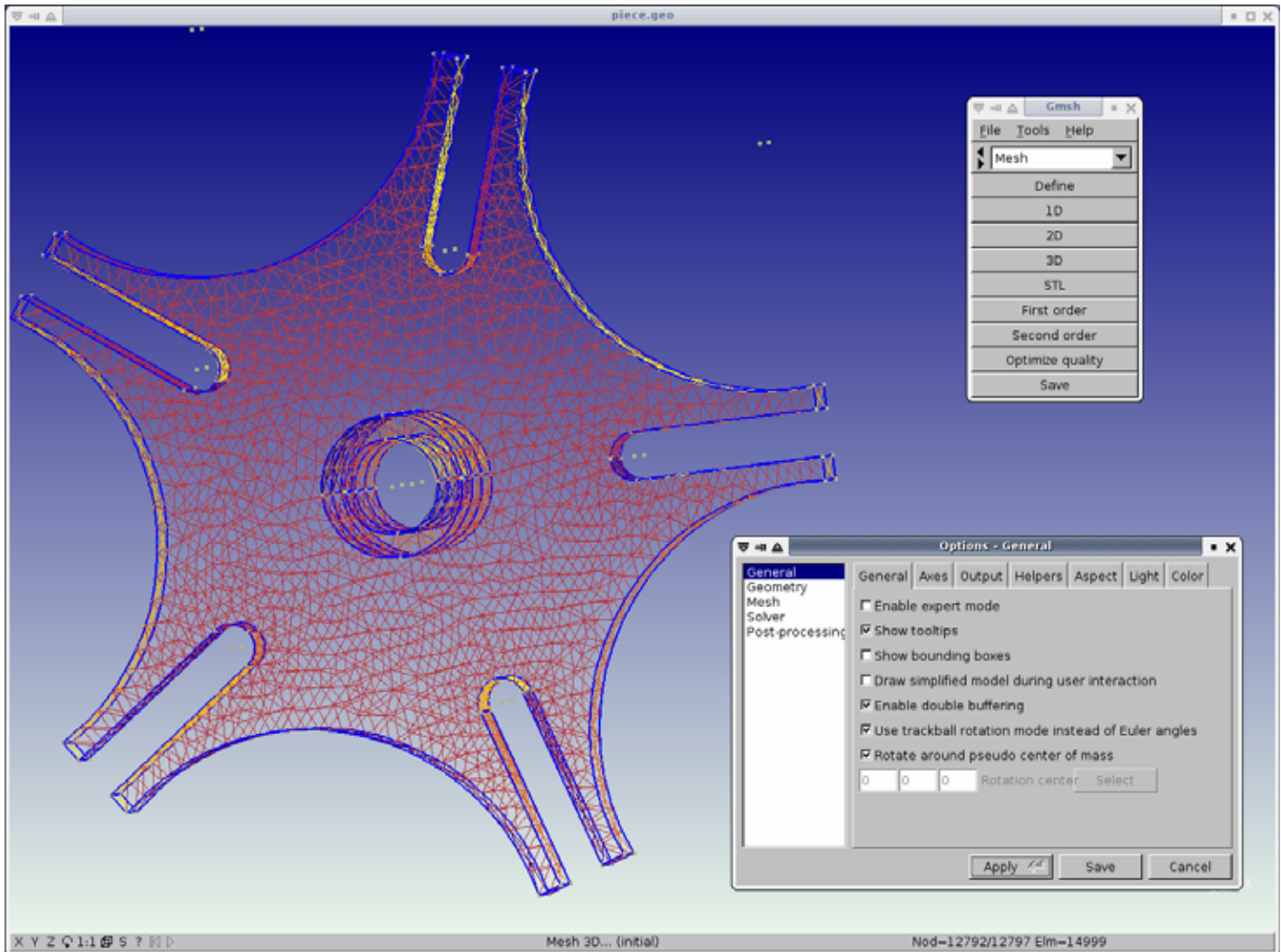


Figure 3: An example of a part meshed using GMSH

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 not bad at all: 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:

CalculiX¹¹ 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.

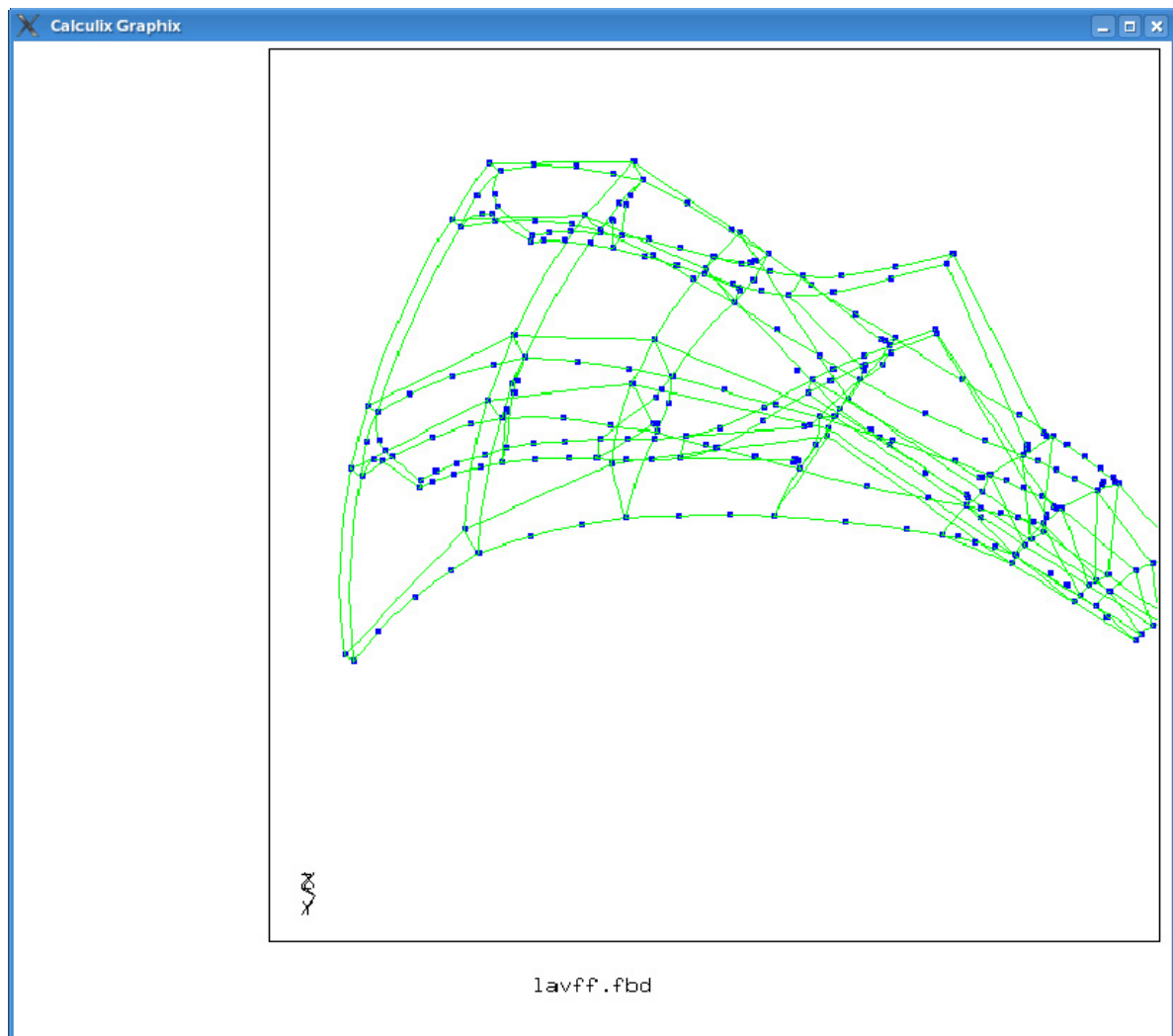


Figure 4. CalculiX Graphix preprocessing an example geometry.

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 ABAQUSTM. The default solver used in *ccx* is SPOOLES, but experimental support for TAUCS is also available. The FreeBSD port by default uses 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 has been reported to be 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 the effort to “localize” it has been going well; and the package interacts very well with other applications that can be used without learning French anyways. Code Aster uses its own modified version of .METIS, and can optionally use SCOTCH as an ordering algorithm. 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 information¹². The default solver is known to compare very well with the parallel versions of the popular NASTRAN™ code.

Code Aster is being actively developed by EDF, which has been very effective in integrating third party software packages: GMSH is one of the preprocessors that can be used by this package by exporting meshes in MED format. It also implements its own command language, it has a specialized editor called EFICAS and a Job Control utility called ASTK (see Figure 4). In near future major packages with full Graphical User Interfaces for preprocessing will be available.

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.

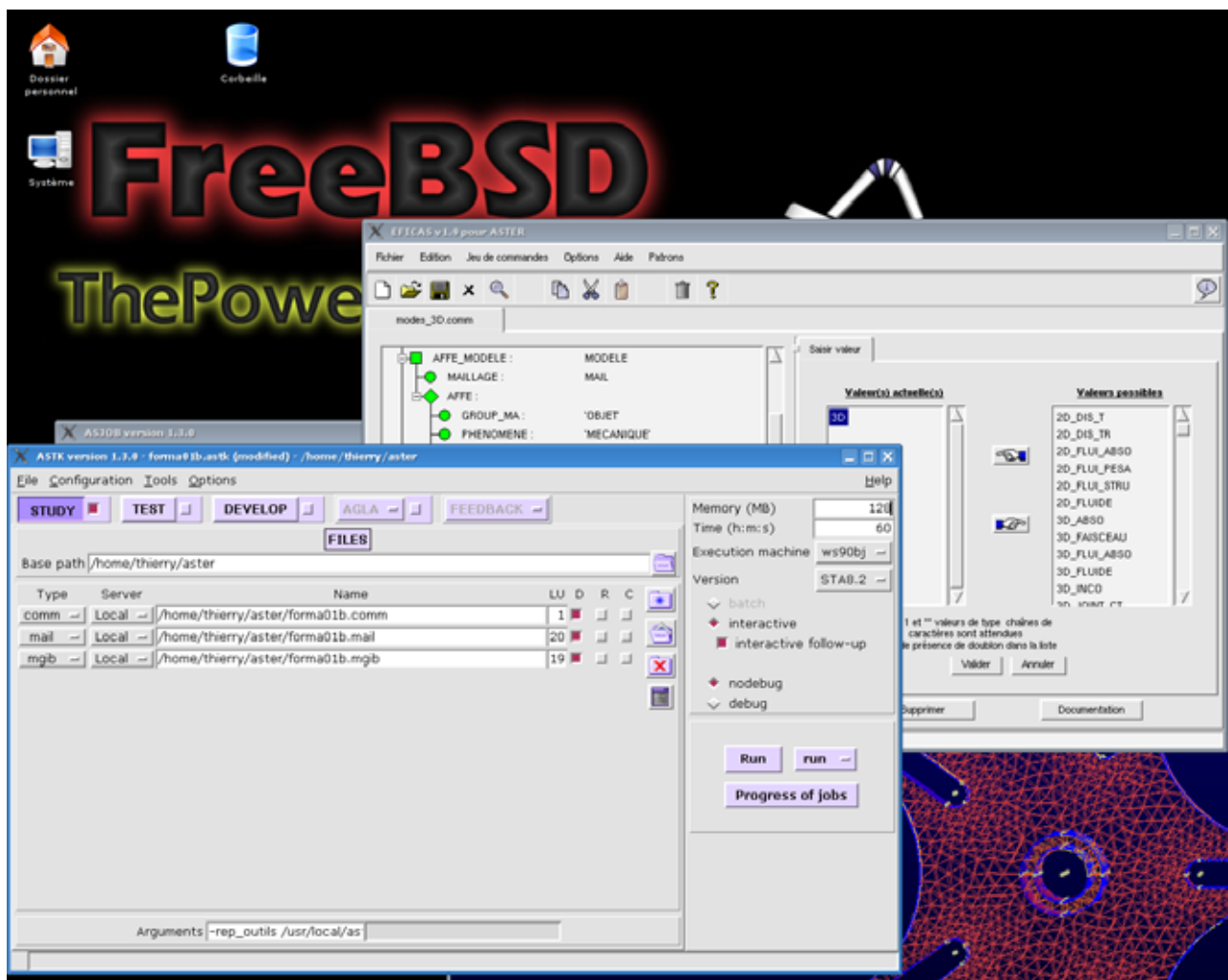


Figure 5. ASTK and other utilities from Code Aster

The latest FEA package ported to FreeBSD is **Elmer**¹³, 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 was the first port to require Fortran 90, a practice very common in the scientific community that imposed some trouble on early FreeBSD versions. We also did some preparations to support MPI in future FreeBSD builds but that job was never finished.

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.

Before closing this section, 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”¹⁴ by Industry Week Magazine. **MBDyn**¹⁵ is a multi-body dynamics analysis system [15] – written at the *Politecnico di Milano*, in Italy.

VII. FUTURE WORK

In general we are not expecting new tools to be developed soon or released freely in these particular engineering fields. We do know about some deficiencies that must be solved, but all the work is done as a volunteer effort so no time estimate for doing what is needed can be made.

For one thing McNeel corporation, the producers of the excellent Rhinoceros modeler has made available, OpenNURBS, the code to support their proprietary .3DM format. Some OpenNURBS support is being incorporated by the BRLCAD team but it is highly desirable in CalculiX, Code Aster and even in other tools like Blender.

Lawrence Livermore National Lab has released an excellent visualizer called VisIt, which has demanded some changes in important ports, but a port for FreeBSD will still take quite long to finish.

Sandia National Laboratories has opensourced the DAKOTA optimization package; in combination with our Finite Element tools this could let us do very sophisticated designs but still there are no hooks between the big tools and Sandia doesn't seem to be too receptive about FreeBSD bug reports.

There is indeed still a lot to be done, and work done on FreeBSD helps other platforms work reliably too, so new developers are welcome to built on the experience and keep free software being an option in many fields.

VIII. 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 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 developing technology or just buying it. In any case there's still the issue of the company knowing it's own business and not spending it's resources or products that are not critical to it's success.

For obvious reasons, commercial applications rarely offer the source code as an option; one of the few applications that makes available it's source code is NASA's Structural Analysis program NASTRAN™, but in this case the application is distributed with important limitations: the source code used to be available only for US citizens, at a high cost. 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 makes you think twice before buying software but it 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.

Perhaps one of the most interesting things to realize is that while all the applications we have considered in this article are available for free, and most of the development ends up being non-for-profit, there is always an added economical value behind them. 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. 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 “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

In addition to God, that always finds a way to put us in the right track, 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 libraries, and the FreeBSD developers for making such an awesome development platform a reality.

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 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.

⁵ R. Clint and Whaley and Antoine Petit, *Minimizing development and maintenance costs in supporting persistently optimized BLAS*, Software: Practice and Experience, Volume 35 Number 2, February 2005.

⁶ 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 Edition, Philadelphia, USA 1999.

⁷ 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.

⁸ 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 ENEEIH-IRIT RT/APO/04/4 and LIP report RR2004-53. Improved version appeared in *Parallel Computing* 32 (2): 136-156, 2006.

⁹ Perry, T. “Donald O. Pederson”. IEEE Spectrum 35: 22–27, June 1998.

¹⁰ O.C. Zienkiewicz and R.L. Taylor, *The Finite Element Method*, 6th ed., Vols. 1 and 2, Elsevier, Oxford, 2005

¹¹ Dhondt, G. *The Finite Element Method for Three-Dimensional Thermomechanical Applications*, Wiley, 2004

¹² 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

¹³ Mikko Lyly, Juha Ruokolainen and Esko Järvinen. ELMER - A finite element solver for multiphysics. CSC-report on scientific computing 1999-2000, pp. 156-159

¹⁴ Industry Week, National Institute of Standards & Technology Gaithersburg, Md. OOF (Object-Oriented Finite Element) Computer Program, December 1999.

¹⁵ 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.