ADMB DEVELOPERS' WORKSHOP REPORT

September, Wednesday 18th - Sunday 22nd, 2013

Marine Research Institute (HAFRO) and University of Iceland

Reykjavik, Iceland

Compiled and edited by **Athol Whitten**[1] and **Arni Magnusson**[2]

[1] University of Washington School of Aquatic and Fishery Sciences, Seattle, USA (whitten@uw.edu)

[2] Institute for Marine Research (Hafro), Reykjavik, Iceland (arnima@hafro.is)

Contents

Introduction

This report summarises the events of the 5th ADMB Developer's Workshop, a meeting held at the Marine Research Institute (Hafro) and University of Iceland in Reykjavik, Iceland. The meeting was attended by many ADMB core developers who were unable to attend the Seattle workshop, and a few who were lucky enough to attend both.

Developers were joined by invited European and American experts, who presented elements of their research relevant to the ADMB-project and added to discussions and working group activities by sharing their experiences with ADMB and other similar tools. The meeting was held in the typical informal style, allowing open group discussions and demonstrations of new features, possible improvements, and current issues. Meeting participants spent considerable time working in sub-groups, each of which tackled a related set of priority issues for the ADMB project.

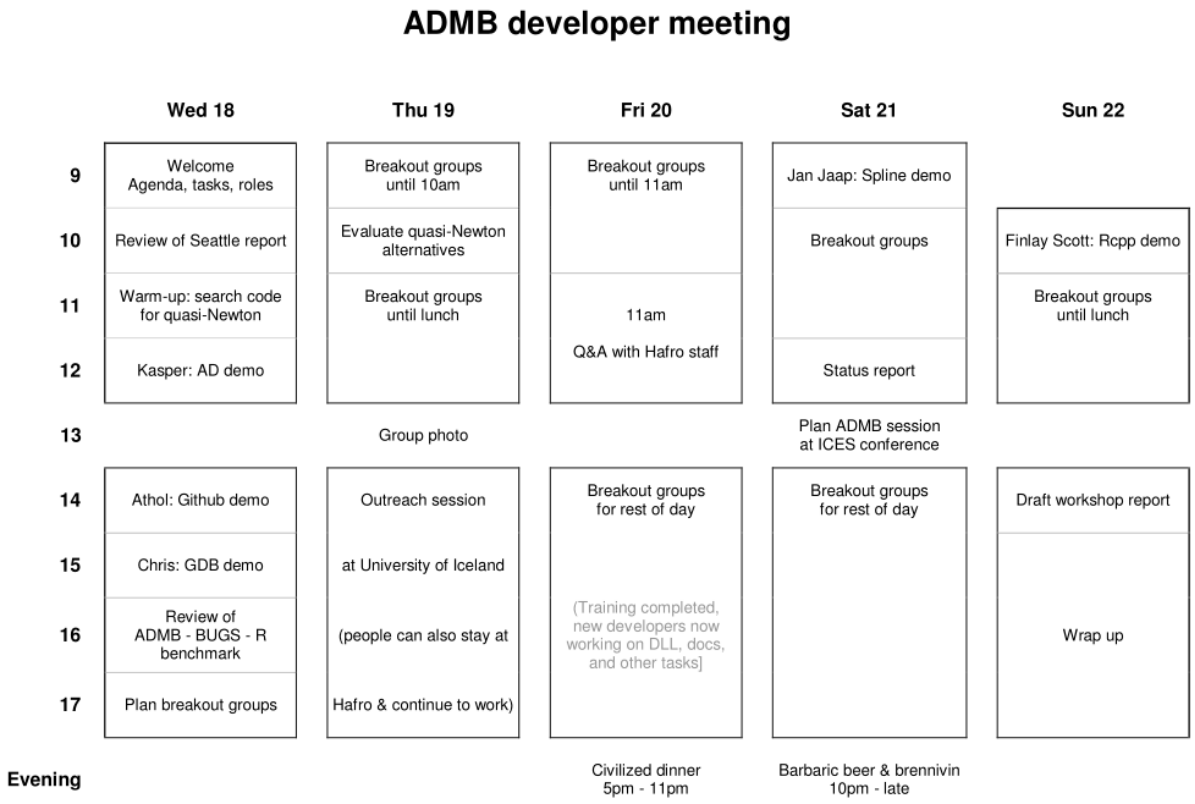
The meeting convened in Reykjavik on Wednesday September 18th, with participants from many institutions around the world taking part; a list of participants is included as an appendix to this document. Arni Magnusson chaired the majority of the workshop, and together with Athol Whitten, acted as rapporteur and compiled and edited this report. An ADMB executive meeting was held on Saturday 21st September with some executive members attending remotely. The meeting was used to discuss key topics arising from the workshop and helped to set priorities for future meetings.

Summary of Workshop Topics

Workshop participants began the week by collectively reviewing the Seattle Developers' Workshop Report. The report included a review of the Seattle Workshop proceedings and the results of a survey of participant satisfaction. Together with the major findings of the previous workshop, survey results were used to plan the Reykjavik meeting and establish an agreed agenda.

A wide variety of topics were discussed but effort was focussed on a set of agreed priority areas: Autodifferentiation (AD) and parallelisation, documentation, developer training, and inter-application connectivity. Other significant topics included debugging options and the use of Github services for ADMB. A key highlight of the workshop was a presentation regarding some new algorithms that in some cases outperformed existing ADMB versions. The final day of the workshop was given to discussing priorities for further development, and opportunities for funding. An overview can be found at: http://www.admb-project.org/developers/workshop/reykjavik-2013

Participants agreed the 2014 workshop should be held in North America to aid involvement of key members of the ADMB Foundation and Developers Team, and that a single workshop should be held in preference to two smaller ones.



1) AD & Parallelisation

Anders Nielsen coordinated a group to work on at this important area of development for ADMB. Enabling AD Model Builder to benefit from multiple cores in modern computers has been identified as one of the highest priorities for the ADMB Foundation.

During and after the Seattle meeting a great effort was made to improve and extend the parallelisation (multithreading) capabilities of ADMB. Some examples were produced using the `pthread` technology in a separate branch of the ADMB source code. During the Reykjavik workshop, the `pthread` approach was tested and compared to the approach used in `TMB`, an R package for Auto Differentiation and model building, which uses `openMP`, an API that supports multi-platform shared memory multiprocessing programming in C++ and other common languages. See the next section for more details on `TMB`.

Both techniques for performing parallel computations scaled equally, but overhead was apparently higher using `pthread`. The method requires the user to handle the detail of passing data and parameters between cores. Altering the *flex* part of ADMB so that it can specify communication between cores automatically was suggested as one way to make this setup useful for ADMB users. The setup for `openMP` in `TMB` is simple to use, but is restricted to use in models where the objective function (e.g., the negative log likelihood) can be constructed strictly as a sum of terms. This work has been summarised in a separate document on the website: http://www.admb-project.org/developers/workshop/reykjavik-2013/parallel.pdf

John Sibert and Dave Fournier maintain a shared document at http://goo.gl/JgVbXY describing the `pthread_manager` class as it is updated and as the API evolves.

Next steps

The workshop noted the need to have prepare a shell script to run many parallel MCMC chains with the current ADMB (e.g., catage example), merge and diagnose chains, and explore how this could be generalized within ADMB.

2) TMB

Kasper Kristensen, an invited expert from the Danish Technical University, reported on his development of a package bundled in R named `TMB` which uses a separate Autodif system and is designed to estimate random effects models (c.f. ADMB-RE). The coding requires a similar level of user input to implement a model using `RcppAD` compared to ADMB. The template style format, inspired by ADMB, is used to write a model, which is fit to data using C++ algorithms dynamically linked to R. Input data, and outputs of the estimation process, are also handled by R.

A key difference with `TMB` is that it links existing powerful C++ libraries: `CppAD`, `Eigen`, `CHOLMOD`, and `BLAS`, and supports parallel processing using `openMP` technology. Parallelisation can take place on the three levels: `BLAS` at the lowest level, `OpenMP` at an intermediate level, and via the R package [`Multicore`] at the top level. When implementing random effects models, `TMB` employs an automatic sparseness detection algorithm, meaning the modeller does not have to describe the Hessian structure (bands, blocks).

Kasper and Anders, together with Casper Berg, and Hans Skaug, developed example models to compare the performance of `TMB` with ADMB. They have demonstrated that in some cases, `TMB` can estimate the parameters of a model much faster than ADMB (up to ~10 times faster). These initial tests indicate roughly 2 to 3 fold speedups in runtime for standard models and 10 fold speedups for random effects models. The summarised results of these tests, as well as a general introduction to the software, are provided in a separate document on the ADMB website:

http://www.admb-project.org/developers/workshop/reykjavik-2013/RcppAD.pdf/view

The `TMB` package is currently undergoing further development on Github and can be downloaded from: https://github.com/kaskr/adcomp. Command line functionality can be invoked to install the R package using the following commands:

```
> git clone https://github.com/kaskr/adcomp

> make install
```

Workshop participants were impressed with the simplicity and performance of the `TMB` package and agreed that understanding such features could greatly benefit the ADMB project. The ADMB project has previously identified attracting R users as a priority, and the `TMB` package provides an effective way of doing so. Workshop participants sought to identify the strengths and weaknesses of both projects and to how best to proceed with providing benefits to both efforts.

Participants agreed both the ADMB and `TMB` projects could benefit from working together. The ADMB project could benefit from embracing new technologies and improving the useability and

speed of its software. For example, experiments have indicated the `sdreport` calculations of ADMB might be improved. The `TMB` project could benefit from access to the ADMB user base, development expertise, and infrastructure. A merger of the two projects might serve to bolster the ADMB project and could certainly help to continue its goals of providing free, open source, and well supported software for non-linear model building.

3) Developer Training

Arni Magnusson coordinated a day of 'developer training' for newer members of the ADMB Core Team and other participants of the workshop. The need to train more people to write and contribute code fixes and new features to Autodif and ADMB was noted as an ongoing priority at the Seattle workshop. The following topics were covered intermittently over the course of two days:

Topic Covered	**Things Learned**
Install ADMB, compile .tpl files, and run models	`tpl2cpp, adcomp, adlink, admb, dll switch`
Build ADMB from source code	`gcc, flex, sed, make`
Repository checkout & commit	`svn, viewvc, log entries`
Code manipulation	`diff, grep`
Scripts	`bash, dos`
Test and debug	`buildbot, gdb`
Dismantling ADMB-IDE	`emacs, lisp, inno`
Documentation	`latex, doxygen`
Redmine Issue Tracker	Look for tasks to work on, update progress

Participants of the developer training, Ben Stevenson, Mollie Brooks, Aaron Greenberg, and Athol Whitten, committed to working on some of the more 'introductory problems'.

4) DLL and ADMB Connectivity

Dynamic link library (DLL) building in ADMB has previously been highlighted as a priority area for the project. DLLs provide a mechanism to share code and data among different applications (between an ADMB model and R for example), allowing the user to upgrade functionality of one application without requiring it to be re-linked or re-compiled. DLLs execute in the memory space of the calling process and with the same access permissions meaning there is little overhead in their use.

The option to create linked libraries has existed in ADMB for some time, but the option was not fully functional prior to the Reykjavik workshop: Only the Windows Visual Studio version was known to be working effectively. During the workshop the Linux version was fixed by adding the `-fPIC` command line option to the source makefile (`src/GNUmakefile`) and `adcomp` script, and DLLs were successfully compiled and accessed using a simple program in R.

There is a known problem in which the Windows MinGW version of ADMB will create a DLL that crashes when accessed through other code (e.g. through R): Since the feature is available in release version 11.1 and there are projects that depend on its functionality, the issue must be addressed as soon as possible. As such the problem has been listed as issues #124 and #125 on the Redmine issue tracking site. The webpage documents some example applications of dlls: http://www.admb-project.org/developers/dll.

Finlay Scott presented his work using the `Rcpp` package to communicate with AUTODIF. In his demonstration he passed R objects to AUTODIF and called `get_value_and_grad` to calculate the gradients. This interface requires good knowledge of both R (S4 classes and Rcpp specifics) and C++, but the advantage is that the special '`dll_`' objects in ADMB and the DLL compilation pathway are not necessary. Finlay's code is available at https://github.com/drfinlayscott/RcppAutodif

5) GDB support

Chris Grandin reviewed the status of debugging abilities for ADMB, including use of Gnu Debugger (GDB) functionality. Since the Seattle workshop, when some new functionality was first discussed, there has been significant progress on the use of GDB, combining the work and ideas of both Chris and Dave Fournier.

The GDB debugger can now be used most efficiently from within the ADMB integrated development environment (ADMB-IDE), which uses the Emacs editor. The contributed library, `'gdbprintlib'` implements functions which expose the ADMB types' print functions to the GDB debugger. This library was created and linked to the ADMB source in response to the Seattle developer's meeting. The functions which are used to view the ADMB objects during a debug session are called `pad()`, with several different prototypes which can display different ADMB variable types. The variants of this can be found in the header file `admb-trunk/contrib/gdbprintlib/gdbprintlib.h`. They have also been fully documented using Doxygen and should appear on the ADMB website, under the "Comprehensive Documentation of ADMB Functions and Classes".

Currently, the GDB scheme works in both native Linux and Windows with a MinGW build. Visual Studio does not support GDB but it should be possible to use the Visual Studio IDE GUI to debug although this has not been tested by developers.

Next steps

Build ADMB libraries with full debug symbols for all platforms; these would be included optionally in future releases of ADMB, along with the optimized and safe libraries.

Continue to improve debugging functionality by supporting DDD (Data Display Debugger). This will be a large job as all of the structures and classes (e.g. `dvector, dvar_matrix`) will have to

have user-viewable wrappers written specifically using something like *python-gdb*. There are many custom structures and classes in ADMB for which this will have to be done.

6) Exploring Github

For Distributing Binaries

Since November 2008, the ADMB project has used Google Code to make ADMB (source and binary releases, ide, manuals) available for download. Google Code has been the equivalent of CRAN for R. In many ways the ADMB Google Code site has acted as the main ADMB webpage, with other webpages representing underlying layers. From January 2014 onwards, Google Code will no longer provide this service, see http://google-opensource.blogspot.com/2013/05/a-change-to-google-code-download-service.html for more details.

It is thus necessary to consider some other way of making ADMB available for download. An ideal alternative service would offer the same benefits as Google Code, i.e.:

- Global mirrors, so 100-200 MB can be downloaded fast
- Complete history of all items ever uploaded
- Very clean and simple webpage, with no instructions. Shows only the current version of each item by default, but deprecated items can be shown as well
- Download counter for every item
- Excellent filter and sort facilities. For example, searching "all downloads" for "ide" gives a very clear version history of ADMB-IDE.

Github was suggested by many ADMB users as a viable alternative to Google Code. In particular, it was thought that the Github 'Releases' feature might work well for ADMB. Investigations by Ben Stevenson concluded that Github however is not well suited as a platform for distributing binaries. Although "releases" is a nice feature, and allows free uploading and hosting of binary files, individual files are encouraged to be below 50MB, with a hard limit of 100MB. Some ADMB release binaries already exceed that hard limit, so unless the project can ensure all future binary releases are less than 100MB, this Github feature will be untenable.

Viable alternatives for hosting binaries include:

- Amazon S3. Pricing is approx. US$0.095/GB for storage, US$0.12/GB outward data transfer.
- Sourceforge, who welcome the distribution of software for source code hosted in a Github repository. A free service.
- Via an FTP server on the ADMB website.

For Code Hosting, Version Control, and More

Github has the potential to replace, and improve upon, several tools currently employed by the ADMB project. Currently, the ADMB project uses the following services to meet various demands:

- **Redmine** is used to log, track, and discuss issues
- **SVN** is used for code version control
- **Viewvc** is used to view and browse SVN control files and features in a browser

Github can be used for source code hosting, version control, issue tracking, and even has an ability to host and display Doxygen created documentation, such as that used for the ADMB API. A pilot Github page has been established for ADMB at https://github.com/admb-project and the ADMB Project has been registered as an 'organisation' with Github, allowing registered Github users to participate in code development as members of the organisation, without the need to create a separate ADMB Github user account. A single repository has been created [`master`] to trial hosting of the ADMB source and to allow users to test and comment on the utility of Github features. A branch of the master repository has been created with 'Github Pages' to host the API and associated Doxygen documentation, and can now be viewed at https://admb-project.github.io/master.

It was agreed by workshop participants, and during the executive meeting, that Github should be tested over an extended trial period, to determine its suitability for providing the ADMB project with a single code hosting and development service to replace the current suite of tools. The trial period will operate as follows:

Users interested to try Git and Github should install and setup Git on their local machines (see

http://git-scm.com and documentation at http://git-scm.com/book) and clone the ADMB master repository with the following command at the Bash Terminal or Windows Command line:

```
> git clone https://github.com/admb-project
```

Users are also encouraged to try the issue tracker and associated discussion tools, and to view code changes and development stats using Github features such as 'Pulse', 'Graphs', and 'Network'.

Github Permissions

Neither Git nor Github forces developers into rigid working structures and permission levels like SVN does. If the ADMB Project chooses to use Github, a clear working policy and permission structure should be prepared, especially in relation to organisation members who can and cannot commit to the main source repositories. Workshop participants suggested Github workflow might work best with a 'master controller' or group of controllers, in charge of submitting push changes, or accepting pull requests. This could be made part of a suggested ADMB teams' structural change, discussed further in **Part 8**.

7) Documentation and Website

Ongoing improvement of ADMB documentation has been a long-standing priority area of the ADMB project. Workshop discussions focussed on the need to improve documentation for new users, especially to produce new, and improve existing, non-technical documentation, start guides, and install procedures.

Build and Install Instructions

Online build instructions for Windows were tested and updated where needed: The instructions are somewhat mixed as they stand, and need to be simplified, or split into two sections, i.e. one for general new users, and another for those interested in developing or modifying the source for their own benefit. The former would outline how to download and install from binary distributions (including with the IDE) and the latter would describe how to download and build ADMB from source. Currently, the 'Quick Install Guide' for Windows contains instructions for building ADMB

from source, which requires additional tools (e.g. compilers) to be installed and may be confusing for new users.

A problem was discovered where Windows won't allow the user to delete certain parts of the downloaded source distribution (see **Issues** section in **Part 8** of this report).

Single-page Introduction to ADMB for New Users

An up-to-date and compact introduction to ADMB was written for the benefit of new users. It briefly explains that 1) ADMB is fast and efficient, 2) how to install, 3) how to build and run models, and 4) outlines the next steps for a new user. See http://admb-project.org/documentation/intro/brief.

Introducing ADMB to R Users

Mollie Brooks and Ben Stevenson created a document aimed at introducing ADMB to R users who aren't familiar with C++ programming. The document explains the concept behind, and necessary components of, an ADMB `.tpl` file and describes examples using `R2admb` to run some simple demonstration models.

A working draft of this document can be found in the ADMB Github repository, see https://github.com/admb-project/master/blob/master/docs/manuals/IntroFromR/ADMB-intro-from-R.pdf.

ADMB Manuals in HTML Format

Making documentation available in HTML format on the ADMB website would allow users to explore components of the documentation without the need to download large files. It would also allow for indexing and web-based searching (e.g. via Google) of the documentation, meaning users could more easily navigate to specific parts of the manuals. Aaron Greenberg made HTML versions of the ADMB Manual using *htlatex*. He wrote a Latex style file to make the resulting HTML output resemble the format and style of the ADMB website: The resulting text is a great start to what is a non-trivial task.

Next steps: The style file needs a bit more work to improve the formatting of the end product, and work should be done to create html versions of the Audodif and ADMB-RE manuals too.

ADMB Training Videos

The group reviewed ADMB training videos created by researchers at the Quantitative Fisheries Center of Michigan State University. The videos are available for download from the University website, and a link is provided on the ADMB website at: http://www.admb-project.org/news/admb-training-videos-on-line

The first video viewed was a very basic introduction to building models in ADMB: It included an introduction to the expected structure and naming conventions of input files and the code they contain. The videos are interactive, and contain stop points where the viewer is required to answer questions about what they have just learned in order to progress through the session. As such, these videos provide an excellent resource for the newest users. The videos could also be used by instructors giving ADMB courses, especially by encouraging participants to access and view the videos prior to the start of courses.

The second video viewed was a basic introduction to the Random Effects version of ADMB. The group, including Anders Nielsen, reviewed and approved the content of the video, as an effective

and accurate introduction to random effects and their implementation in ADMB-RE. The group concluded that the videos should be made more visible and promoted on the ADMB website.

Source directory tree

Workshop participants reviewed and documented the configuration of ADMB source files. The following elements and their use were noted:

Source subdirectory	Code component details
`df1b2-separable`	ADMB-RE
`linad99`	AUTODIF
`nh99`	ADMB
`tools99`	Utilities (when **not** passing user option `-shess`)
`Sparse`	Utilities when passing user option `-shess`

Information relating to the structure and content of the source tree was uploaded to the website and is available at http://admb-project.org/developers/source-tree

ADMB Minimizer

A visit from academics from the University of Iceland, interested in function optimization, prompted workshop participants to investigate the current ADMB quasi-Newton minimizer (function optimizer) and to improve general understanding of its functionality. That relevant code is located in the file `newfmin.cpp` in the `linad99` folder of the source tree.

The minimizer function (`void fmin`) contains a quasi-Newton function minimizer with an inexact line search using strong Wolfe conditions and a BFGS correction formula for the Hessian update.

The function requires:

`f` - objective function value to be minimized

`x` - vector of independent variables

`g` – the gradient vector returned by function `gradcalc`, containing partial derivatives of `f` with respect to each independent variable `x`.

Modifies:

`h` - The Hessian matrix

Returns (via parameter vector x):

`X` – a vector, completed after a series of linear searches in the direction of the gradient descent

The code for the minimizer function is very old and not elegantly written. Dave Fournier reported to the group that the original code was written in Fortran II and obtained from a colleague at UBC in 1977. He translated it into ANSI C around 1988 and discovered many problems with the code over the years, making adjustments and fixes as required. The code contains a lot of further checks and workarounds, after decades of use by the ADMB user community. The group considered re-writing the minimizer with cleaner code and associated documentation but agreed it should be tested against other minimizers to determine whether it would be best to replace the code entirely.

The `fmin` code was thoroughly studied and commented by Inna, who investigated the main steps of the quasi-Newton method such as line search and backtracking, calculations of step length satisfying Wolfe conditions, Hessian updating formulas, and the calculation of Newton steps. A few obsolete although not significant bits of code were detected.

Investigations determined that the ADMB quasi-Newton method is not a conventional implementation of its kind. It does not update the inverse Hessian as the classic quasi-Newton methods do, but rather builds an approximation of the Hessian itself. This approach involves the matrix inversion problem while computing the search direction. The use of Cholesky decomposition of the Hessian as suggested by Gill and Leonard (2001) can be envisaged. This approach is known to reduce the number of function evaluations during the initial phase of optimization and to help solve the problem of round-off errors, which can cause a nearly singular or non-positive-definite Hessian.

Next Steps: Compare the efficiency of ADMB quasi-Newton with other implementations (for example, those embedded in R `optim` function) to see if it allows a significant reduction of the number of iterations and function evaluations needed to reach the solution of minimization problems.

Reference: *P. E. Gill and M. W. Leonard, Reduced-Hessian quasi-Newton methods for unconstrained optimization, SIAM J. Optim., 12 (2001), pp. 209–237.*

NCEAS Benchmarking Tests

The group reviewed code and documentation related to the benchmarking tests performed by the NCEAS non-linear modelling working group. The results of the tests were tabulated and made available as a .pdf file on the website at: http://admb-project.org/developers/benchmarks/optimization/nceas.pdf

In general, ADMB performs several times faster than R and BUGS, except for trivial models that take less than 0.5 seconds to run. This important result should be promoted to attract new users to ADMB. The benchmarking results should thus be made more visible than they are currently, and the group suggested creating a 'Benefits of Using ADMB' section of the website and placing a link to it on the front page.

Priorities for Documentation

The following tasks were identified as needing continuing effort:

- Update manuals with new features introduced since version 8
- Identify parts of the website that are outdated and should be archived
- When providing new users with examples and presentations, show them relevant contributed functions (e.g. `dnorm` and `Ricker`). Edit existing examples on the website to use these also
- Make all .pdf files viewable on the website without downloading
- Edit the `Makefile` of the `docs` directory so that it creates an HTML version of the manual and possibly other documents. Link to these from the website.
- Add new Doxygen comments
- Assess the quality of introduction videos and if satisfactory, post them to YouTube.

It might also be possible to collect existing online ADMB documentation from a range of authors and to merge those into useful summaries on the ADMB website.

8) Secondary Topics

Several other topics were discussed during the course of the week, but not necessarily given a lot of time. The following details may help to set priorities for the coming year and future meetings.

ADMB Spline Functionality

Jan Jaap, from the Netherlands Institute for Marine Resources & Ecosystem Studies (IMARES) gave a presentation to the group about implementing Splines in ADMB. His work highlighted the need to test and document existing spline capabilities in ADMB, including recent spline functions in contributed libraries (e.g. the *statslib* which contains functions for spline fitting). Basic documentation of the ADMB cubic spline classes can be found on the API page:

http://www.admb-project.org/documentation/api/group__cub__spline.html

Next Steps: Create spline basis matrix in preliminary calcs section, preferably to be similar to matrix functionality in R.

ADMB Teams

Currently the ADMB User Community is divided into 'Developers' and 'Users'. Many users may wish to contribute code or participate intermittently in development, without wanting to commit to becoming a recognised 'Developer'. As such, it was suggested the ADMB Project could more formally define and recognise the roles and responsibilities of its various user community members, including a group for people who might fall somewhere between User, and Core Developer. A formal separation of roles could also be linked to a permissions structure and workflow policy should the ADMB Project adopt the use of Github for code development and version control. For example, the ADMB user community could be stratified as follows:

Name	Type	Role	Permissions	Notes
Masters	Formal	Grant pull requests, push changes to master repository	Push and Pull	Small group (4-5) with full control of source repository, responsible for overseeing changes
Developers	Formal	Core development and source maintenance, make pull requests to Master	Pull Only	Larger group (10-15) with key responsibilities to test, maintain, and contribute to code development. Expected to attend Developer Workshops when able.
Contributors	Informal	Intermittent development, issue tracking, bug fixes	Pull Only	Soft entry to the Development Team: Includes recognition on the website without other responsibilities. Invited to Developer Workshops.
Users	Informal	End use of software, identify and highlight bugs	Public	Participate in online forums and help discussions.

Code Naming Conventions

There is currently no formally defined standard for function naming conventions for ADMB. This can be quite a barrier for new developers, and leads to conflicting conventions and standards among code contributions from different contributors. Workshop participants also identified that existing code does not necessarily follow naming conventions and that a naming convention

protocol should be developed and voted upon by the Development Team. The following issues were identified in relation to this topic:

- Naming conventions should be straight-forward and follow closely with existing code and/or existing naming conventions from another project (C++ naming conventions for example).
- Existing code and functions may not follow naming conventions but changing these could be an inconvenience to existing users: There are existing models and legacy code implementation that should be supported or at least considered when making such changes.
- Using R naming conventions (e.g. `dnorm`) might lead R Users to reasonably expect functions to follow R input and output conventions too. This is not the case for some existing ADMB functions with naming conventions borrowed from R. The ADMB Project should determine if using R naming conventions is appropriate, and if so, how best to follow other expectations.

There is currently a place-holder for information relating to naming conventions on at the ADMB website at the bottom of the Coding-Standards page http://www.admb-project.org/developers/coding-standards and the need for ADMB Naming Conventions is recognised as a new priority area for ADMB.

Installation Process

There is an existing problem for current and potential future developers, or for anyone interested in downloading the ADMB source distribution to a Windows Machine. Once the source distribution has been saved to a local directory, there are folders which cannot be fully deleted from the local machine. The problem appears to be related to some *GNUMakefiles* in the build folder, but those workshop participants who investigated the issue were unable to determine the root cause of the problem.

9) Priorities and Next Steps

Priorities developed at this meeting (and carried over from the Seattle meeting) include

1. Matrix library improvement Linking external (large) matrix library to improve random effects library
   This topic was discussed extensively and examples were developed at the Icleandic workshop. This includes leveraging the facility of TMB with ADMB to benefit both efforts
2. Parallelization Internal ADMB part, and user control part, with a "howto"
   Work progressed prior to and during the Seattle 2013 workshop. A simple model was refined and was committed to the branch threaded2. Goal to merge current branch to the trunk so that those interested could test and aid in further development.
3. Prepare for alternative site for downloading binaries
   Google code site is closing and an alternative site (and mirrors) is needed. Download count is important to include.
4. Evaluate option of using github for version control system
   Appears to be compatible with present SVN system and has a number of benefits and improvement over svn repositories.
5. Hybrid MCMC Fully implement and document this capability
   Cole Monnahan along with Ian Taylor and Jim Thorson presented their work done on this in the past year. The idea of using multiple "heated" chains together with the multithreading was discussed. The group noted that some work coordinating documentation is needed.
6. MCMC streamlining Model conversion to constant objects
   Dave F. noted that this work would be relatively easy to implement but first should be tested by simply converting an existing model to use constant objects to test if the improvements are real (i.e., overhead maybe improved (speed-wise) with newer compilers)
7. Improve the ability to test/debug code
   A valgrind tutorial was made and posted on the website. Reviews of using GDB (plain and with DDD, in addition to others such as the netbeans GUI) was demonstrated.
8. Design matrices Add capabilities for creating design matrices
   Unsure if this has been added, Anders Nielsen had done some of this.
9. Shared library object
   To add flexibility ADMB was compiled as a shared object library in the standard distribution and also by Matthew Supernaw in CLang.
   Johnoel notes that this is complete (revision 1064) hence such libraries are available.
10. Code tuning, debugger Profiling tool to improve code, e.g., where adjoint code may best be applied. Gcc profiler, memory usage. Develop example howto
    See valgrind example.
11. Flag/switch class Simplify the use of control and MCMC files, perhaps MFCL-like object capability
    This work is underway.
12. Developers workshop To train more people on how to write and contribute code to Autodif/ADMB
    Remains a priority, limited progress at June 2013 workshop

13. Data alternatives XML may be a useful approach to adding robustness between datafiles and DATA_SECTION, Netcdf capabilities?
    John Sibert reported on his draft XML library that could fit as an alternative run-time parameter dimensions, bound, phases etc. Progress has waned on building this to the extent needed to become part of the contributed library.

It was noted that the ADMB should encourage student support (e.g., improving ADMB with whatever fisheries problems might arise). Trevor Branch (UW professor) might be keen to take on this type of student with one idea to make use of Dave's multithreading work.

10) Long Term Goals and Funding

The ADMB Project is approaching a transitional phase, with major funding available and allocated through until September 2014. The ADMB Foundation has recently undertaken a review of long-term progress and begun the process of establishing a new set of long-term goals and funding strategies. The Foundation will produce a report toward the end of 2013 outlining those goals and strategies. Workshop participants contributed to the discussion by identifying the following opportunities for funding.

Opportunities for Funding

The group considered and discussed the need to acquire additional funding for the coming years, and to broaden the scope of the sources of funding. It was suggested the ADMB Foundation may be able to relate ADMB funding to research grant applications, but this posed some key questions: e.g. How restricted are research grants, in terms of how money is spent?

- **NOAA** has been a main supporter, and may support new ADMB-related research projects (Rick Methot, Jim Ianelli). NOAA currently supports ADMB development through the **JIMAR** cooperative agreement at the University of Hawaii at Manoa
- **Nordic Community Grant** has funded workshops (Hans)
- **UK** animal population density estimation (Ben, Jeff Laake)
- **LL** compilation R-ADMB (Laurie Kell)
- **TMB**, documentation, publication (Kasper)
- **CAPAM** research group in La Jolla, selectivity (Mark, Paul Crone)
- **ICES** [Recently contacted, awaiting response] (Anders, Arni)
- **a4a** is a fisheries-related software project that has received funding (Ernesto, Colin Millar, Finlay), research project within JRC
- **FLR** received initial grant from European Union (Iago)
- **European Union** support large research projects, often for 5 years, well-defined deliverables (Ken Patterson)
- Research related to **large pelagic species** may support ADMB-related research projects (Laurie Kell, Victor Restrepo, Iago, John Sibert, John Hampton)

Appendix A: List of Participants

Anders Nielsen	(*Denmark, core developer*)
Arni Magnusson	(*Iceland, core developer*)
Athol Whitten	(*Australia, core developer*)
Casper Berg	(*Denmark, core developer*)
Chris Grandin	(*Canada, core developer*)
Hans Skaug	(*Norway, core developer*)
Mollie Brooks	(*USA, core developer*)
Aaron Greenberg	(*USA, invited expert*)
Ben Stevenson	(*New Zealand, invited expert*)
Finlay Scott	(*UK, invited expert*)
Inna Senina	(*France, invited expert*)
Jan Jaap Poos	(*Netherlands, invited expert*)
Kasper Kristensen	(*Denmark, invited expert*)

And professors and scientists at the University of Iceland and Hafro: Birgir Hrafnkelsson, Bjarki Elvarsson, Einar Arnason, Gudmundur Gudmundsson, Helgi Tomasson, Hoskuldur Bjornsson, Kristjan Jonasson, Stefan Valdimarsson, Sven Sigurdsson, and Thorvaldur Gunnlaugsson.

Appendix B: Evening Workshop Report (Barbaric Beer and Brennivin Night)

On Saturday night, an evening session of fun and frivolity was held in beautiful downtown Reykjavik: Arni introduced the team to a typical Icelandic way to start an evening, with rotten shark and brennivin! That was followed by a more civilised session of beer (and more brennivin) at a local hipster bar. Hipsters apparently go to bed early in Reykjavik so the team moved on to a local whiskey bar and enjoyed some local (and some not so local) spirits, before finishing the night at a dance club. Few knew that some of the ADMB developers could dance so well. The entire group rose early with a fresh and enthusiastic disposition to continue developing on Sunday.