Why do you need Numerical Libraries?

Don’t try to reinvent the wheel

- Would you reimplement tools like git, valgrind, etc.?
- Are you paid for writing numerical components?
- Numerical Library is a tool
Why do you need Numerical Libraries?

- Model your code at a higher level
  - Use numerical algorithm instead of implementing them (e.g. optimizers)
- Concentrate on your task and core expertise
- Reduce development time
- Reduce maintenance time
- Use external expertise
  - choice of algorithms, support, speed
Desired Properties of Numerical Libraries

- Reliable components
  - Stable numerical algorithms

- Coverage and availability of alternative algorithms

- Portability
  - Different OS, languages etc.

- Maintenance
  - Bugfixes and regular updates

- Documentation and support

- Parallelization
The NAG Library

- Hundreds of routines devoted to numerical analysis and statistics, the NAG Library helps users build applications for many different industries and fields.

- For your current and future programming environments
  
  - NAG Library routines are available for Fortran, C, C++, Python, .NET, Java, MATLAB and others
  
  - NAG Library routines can be called many computer languages/environments such as Visual Basic, Octave, Scilab, R etc.

  - Assists migration of applications to different environments
NAG Library Full Contents

- Root Finding
- Summation of Series
- Quadrature
- Ordinary Differential Equations
- Partial Differential Equations
- Numerical Differentiation
- Integral Equations
- Mesh Generation
- Interpolation
- Curve and Surface Fitting
- Optimization
- Approximations of Special Functions

- Dense Linear Algebra
- Sparse Linear Algebra
- Correlation & Regression Analysis
- Multivariate Methods
- Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research
Why use NAG Libraries and Toolboxes?

- Global reputation for quality – accuracy, reliability and robustness...
- Extensively tested, supported and maintained code
- Reduces development time
- Allows concentration on your key areas
- Components
  - Fit into your environment
  - Simple interfaces to your favourite packages
- Regular performance improvements!
- Give “qualified error” messages e.g. tolerances of answers
NAG Library - Ease of integration

- Supporting Wide Range of Operating systems...
  - Windows, Linux, Mac, ...

- ...and a number of interfaces
  - C, C++,
  - Fortran,
  - VB, Excel & VBA,
  - C#, F#, VB.NET,
  - CUDA, OpenCL,
  - Java,
  - Python,
  - Julia,
  - ...
  - Excel,
  - MATLAB,
  - Hadoop / Apache Spark,
  - LabVIEW,
  - R, S-Plus,
  - Mathematica,
  - Scilab, Octave
  - ...

NAG development philosophy

- First priority: **accuracy**
- Second priority: **performance**
  - How fast do you want the wrong answer?
- Algorithms chosen for
  - usefulness
  - robustness
  - accuracy
  - stability
  - speed
NAG Technical Support Service

- Single point of contact: dedicated technical desk
- Highly knowledgeable team
  - Combined experience of software support > 50 years
- Advice on a wide range of areas including
  - functionality
  - diagnosis of user problems
  - work around to assist users ahead of standard updates
  - product availability for specific operating systems
  - advice on the best functionality for your application needs
  - wide range of documentation and technical reports
- Updates and access to new releases
NAG Technology Innovation

- Long history of collaboration with the world’s leading scientists and engineers across academia, government research and industry

- Examples of ongoing collaboration are:
  - work with mathematicians and statisticians across the globe to produce the best / most competitive algorithms for the NAG Library and bespoke solutions;
  - in accelerator computing / HPC (many core, GPU,...), working closely with:
    - the main hardware vendors (AMD, ARM, Intel and NVIDIA)
    - relevant leading academics (inc. Professors Mike Giles, William Shaw, Nick Higham, Jack Dongarra);
  - innovating by working with RWTH Aachen University to deliver Algorithmic Differentiation solutions (Prof. Uwe Naumann et al).
The NAG Library is divided into chapters, each devoted to a branch of maths or statistics. Each has a 3-character name and a title, e.g., F03 – Determinants.

Exceptionally, Chapters H and S have one-character names.

All routines in the Fortran Library have six-character names, beginning with the characters of the chapter name, e.g., d01ajf (last character stands for Fortran).

There are also “long names” that aim to be more descriptive.
The NAG Library has complete documentation

- Chapter introductions
  - technical background to the area
  - assistance in choosing the appropriate routine
- Routine Documentation
  - description of method
  - specification of each parameter
  - explanation of error exits
  - example programs
  - remarks on accuracy

All documentation is available online

- https://www.nag.co.uk/numeric/fl/nagdoc_latest/html/frontmatter/manconts.html
First Steps with the NAG Library

- A detailed implementation specific description on how to compile and run the examples is given in User’s Note

- The easiest way to start
  - On Windows
    - use `nag_example_* .bat` batch files located in `[INSTALL_DIR]/batch`
    - you might need to run the `envvars.bat` batch file first to set the environment variables
  - On Linux
    - use `nag_example_* scripts` located in `[INSTALL_DIR]/scripts`
Errors and Warnings

- The routine has detected a warning or an error if the value of argument IFAIL (or, in chapters F07 and F08 INFO) is non-zero on exit

- For details about how to interpret this value the user should consult the Error Indicators and Warnings section of the document for the particular routine
Errors and Warnings

- **IFAIL argument**
  - allow you to specify what action the Library routine should take if an error is detected
  - to inform you of the outcome of the call of the routine

- **On input if IFAIL=**
  - 0: Hard fail. The execution of the program will terminate if the routine detects an error
  - 1: Soft fail with silent exit. Returns control to the calling program without output of the error message
  - -1: Soft fail with noisy exit. Outputs an error message before the control is returned to the calling program

**Don’t forget to test the value of IFAIL in soft fail mode!**
Different Implementations of BLAS and LAPACK

- NAG Fortran Library provides static and shareable libraries that use different implementations of BLAS and LAPACK routines
  - Intel MKL (should be used for best performance)
    - Multithreaded
    - libnag_mkl.a (linux), nag_mkl_M*.lib (windows)
    - libnag_mkl.so (linux), FLW6I26DE_mkl.lib/FLW6I26DE_mkl.dll (windows)
  - NAG
    - libnag_nag.a (linux), nag_nag_M*.lib (windows)
    - libnag_nag.so (linux), FLW6I26DE_nag.lib/FLW6I26DE_nag.dll (windows)
NAG Library Interoperability with C/C++ and Python

- Interoperability with C/C++
  - C Headers
  - NAG C library

- Interoperability with Python
  - Full set of bindings available for NAG C library – for Windows, Linux and Mac
  - Access to NAG routines from Python for quick prototyping
  - Same high quality NAG routines used in production system (C, Fortran, .NET, Java, ...) as used under Python prototype
  - Supported by white papers for calling NAG Fortran or C Library from Python
    - http://www.nag.co.uk/python.asp
Conclusions – NAG Imperatives

- Help customers achieve better solutions to difficult problems in technical computing

- How is NAG different, as a not-for-profit organisation
  - No financial ownership: all profits are used to pursue our imperatives to the benefit of our customers and collaborators
  - NAG constitution means NAG (effectively) can’t be sold:
    - “perpetual company”
    - Longer term planning horizon
Conclusions – NAG solutions

The NAG Library provides:

- Standard and advanced routines
  - hundreds of numerical routines
- Reliability
  - all routines vigorously tested
  - extensive experience of implementing numerical code
- Portability
  - accessible from many software environments
  - constantly being implemented for new architectures
- Support
  - directly supported by the team that creates the code
Let’s Link Up
Ways to connect with us

Twitter:  www.twitter.com/NAGTalk

Blog:  http://www.nag.co.uk/blog

LinkedIn:  http://www.linkedin.com/e/vgh/2707514/

Experts in numerical algorithms and HPC services