NAG Quant Seminar / Finance Focus Event

Latest releases and news

John Holden

New York, July 2012

nag®

Experts in numerical algorithms and HPC services

WILMOTT
SERVING THE QUANTITATIVE FINANCE COMMUNITY

CERTIFICATE IN QUANTITATIVE FINANCE
CQF

The NAG Library is now at Mark 23

Now available as:

- The NAG Fortran Library
- The NAG Library for SMP & Multicore
- The NAG C Library
- The NAG Toolbox for MATLAB

The NAG Library is now at Mark 23

Now available as:

- The NAG Fortran Library
- The NAG Library for SMP & Multicore
- The NAG C Library *1st implementations available*
- The NAG Toolbox for MATLAB

1st implementations became available this week!

The NAG Library Contents

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

The NAG Toolbox and NAG C Library

Senior quant from Tier 1 Investment Bank

"We deploy production code in C++ embedding NAG C Library functions wherever we can, but often prototype new models in MATLAB before writing our C++ code. Having the same NAG algorithms in MATLAB via the NAG Toolbox for MATLAB is a real win for us"

NAG Toolbox: Ease of use improvements

- Function Handles
  - In previous versions of the NAG Toolbox for MATLAB, users had to provide some parameters as m-files. While this functionality is still supported, users may also provide parameters as function handles.
- Better Exception handling
- Integer Utility introduced
  - Making it easier to write portable code between 32 & 64 bit platforms
- Improved example programs and long names

NAG Library : *new* Mark 23

Mark 23 has new functions in many areas including...

- Wavelet Transforms
  - One dimensional continuous transforms
  - Two dimensional discrete single level and multi-level transforms
- ODE's
  - BVP solution through Chebyshev pseudo-spectral method
- Matrix Operations
  - Matrix exponentials
  - Functions of real symmetric and Hermitian matricies
  - Sparse matrtix functions
  - LAPACK 3.2 Cholesky solvers and factorizations, and many other LAPACK driver functions
- Interpolation
  - Modified Shepard's method in 4D/5D
- New vector functions* (in G01 and S)
- optimisation
  - Multi-start optimisation*
  - Minimization by quadratic approximation (BOBYQA)
  - Stochastic global optimisation using PSO
- RNG's
  - Generators of multivariate copulas
  - Skip-ahead for Mersenne Twister
  - L'Ecuyer MRG32K3a generator
- Statistics
  - Quantiles of streamed data, bivariate Student's t, and two probability density functions
  - Nearest correlation matrices
  - Quantile regression
  - Peirce Outlier detection
  - Anderson–Darling goodness-of-fit

NAG Library : *new* Mark 23

Mark 23 has new functions in many areas including...

- Wavelet Transforms
  - One dimensional continuous transforms
  - Two dimensional discrete single level and multi-level transforms
- ODE's
  - BVP solution through Chebyshev pseudo-spectral method
- Matrix Operations
  - Matrix exponentials
  - Functions of real symmetric and Hermitian matricies
  - Sparse matrtix functions
  - LAPACK 3.2 Cholesky solvers and factorizations, and many other LAPACK driver functions
- Interpolation
  - Modified Shepard's method in 4D/5D
- New vector functions* (in G01 and S)
- optimisation
  - Multi-start optimisation*
  - Minimization by quadratic approximation (BOBYQA)
  - Stochastic global optimisation using PSO
- RNG's
  - Generators of multivariate copulas
  - Skip-ahead for Mersenne Twister
  - L'Ecuyer MRG32K3a generator
- Statistics
  - Quantiles of streamed data, bivariate Student's t, and two probability density functions
  - Nearest correlation matrices
  - Quantile regression
  - Peirce Outlier detection
  - Anderson–Darling goodness-of-fit

nag®

* only available in NAG C Library & NAG Toolbox (at MK23)

Focus on new routines in Mark 23

- Slides derived from http://www.nag.com/numeric/cl/newarticles23

Focus on new routines at Mark 23 (1)

- Matrix functions
  - exponential, principal logarithm, sine, cosine, sinh, cosh
    - further marks will include square roots, non-integer powers...
  - used in the solution of differential equations

$$\frac{d\boldsymbol{y}}{dt} = A\boldsymbol{y} \Rightarrow \boldsymbol{y} = \boldsymbol{y_0} e^{At}$$

  - used in Markov chains in finance
    - given the transition probability matrix, often want its p-th root
  - used in graph theory
    - given the network matrix, its exponential measures connectedness

Focus on new routines at Mark 23 (2)

Global optimisation

- Multilevel Coordinate Search method

  added at Mark 22

  - Handles simple bound constraints only
  - Derivatives are not required
- Multiple-start algorithms for optimisation
  - run local optimizer from several different points
    - users have asked for a list of all local minima – not just the global
    - useful when other criteria exist for an acceptable solution
  - alternative to more complicated global optimisation methods

Focus on new routines at Mark 23 (3)

- Particle Swarm optimisation - some might say *“Method of last resort!”*
  - The routines search for a global minimum using a variant of the particle swarm heuristic.
    - This involves the initialization of a ‘swarm’ of particles in hyperspace, which are advected though the domain using a velocity dependent upon ‘inertia’, ‘cognitive memory’ and ‘global memory’.
  - These routines are most effectively used when multiple cores are available for computation.

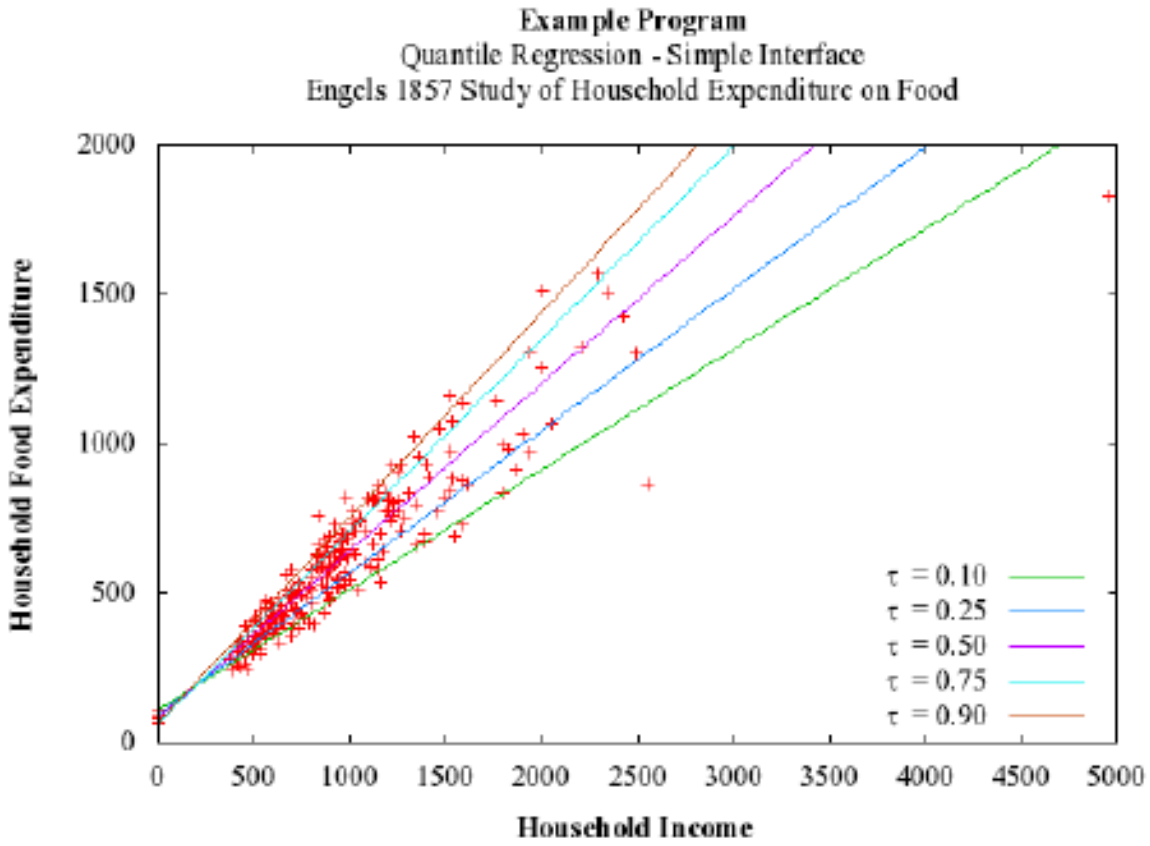
Focus on new routines at Mark 23 (4)

- Bound optimisation BY Quadratic Approximation
  - robust, easy-to-use minimization routine
    - minimizes objective function subject to bound constraints
  - uses quadratic approximation and trust regions
  - doesn't require derivatives
    - useful in inverse problems – e.g. trying to infer material properties by matching simulation output to experimental measurement
  - Example: distribute 50 points equally on a sphere
    - BOBYQA : 4633 evaluations
    - Nelder-Mead simplex solver (also in NAG Library) : 16757 evaluations

Focus on new routines at Mark 23 (5)

- Quantile regression
  - models quantile of response variable
    - *cf.* least-squares regression, which models the mean
  - allows for more comprehensive data analysis
    - potentially more insight into the data and underlying relationships
    - less sensitive to large outlying distributions
  - used in, e.g., ecology
    - to discover more useful predictive relationships between variables
      - when there is only a weak relationship between means

Quantile regression example



Koenker R (2005) *Quantile Regression* Econometric Society Monographs, Cambridge University Press, New York

Focus on new routines at Mark 23 (6)

- Improvements to nearest correlation matrix
  - correlation matrix is used to construct sensible portfolios
    - gives correlation between stocks in portfolio
      - must be *positive semidefinite*
      - sometimes isn't – e.g. due to missing values
  - routine for nearest correlation matrix added to Mark 22
    - "closest approximation" to input (non-semidefinite) matrix
  - Mark 23 enhancements allow for use of
    - weighted norm
    - factor structure
  - incorporates more information about problem to be solved

Focus on new routines at Mark 23 (7)

Sampling = randomly selecting one or more observations or records from a particular dataset.

- Sampling with unequal weights
  - selecting records from a dataset
  - with / without replacement, with equal / unequal weights
    - without replacement, with unequal weights new in CL23
- Sampling used widely
  - selecting subjects for study in controlled experiment
  - converting from a continuous signal to a discrete one
  - re-using parts of sound recordings

Focus on new routines at Mark 23 (8)

- Better Mersenne Twister Random Number Generator
  - Mersenne Twister has very long period
    - also fast implementation and good statistical properties
  - Want to use it to create multiple streams
    - for parallel applications
  - Recommended method is skip-ahead
    - sequence is partitioned into non-overlapping streams
  - Now possible in Mark 23
    - generate multiple streams of huge numbers of values with no overlap
- New routine for L'Ecuyer MRG32k3a RNG, as well

COMING SOON…..

NAG and Java

Currently we steer the user to write their own wrappers and provide a few examples.

- Use JNI library to wrap calls to NAG routines
  - in either C Library or Fortran Library
  - Technical report with detailed examples available
    - online at www.nag.com
  - Wrappers for specific routines available from NAG
    - on request

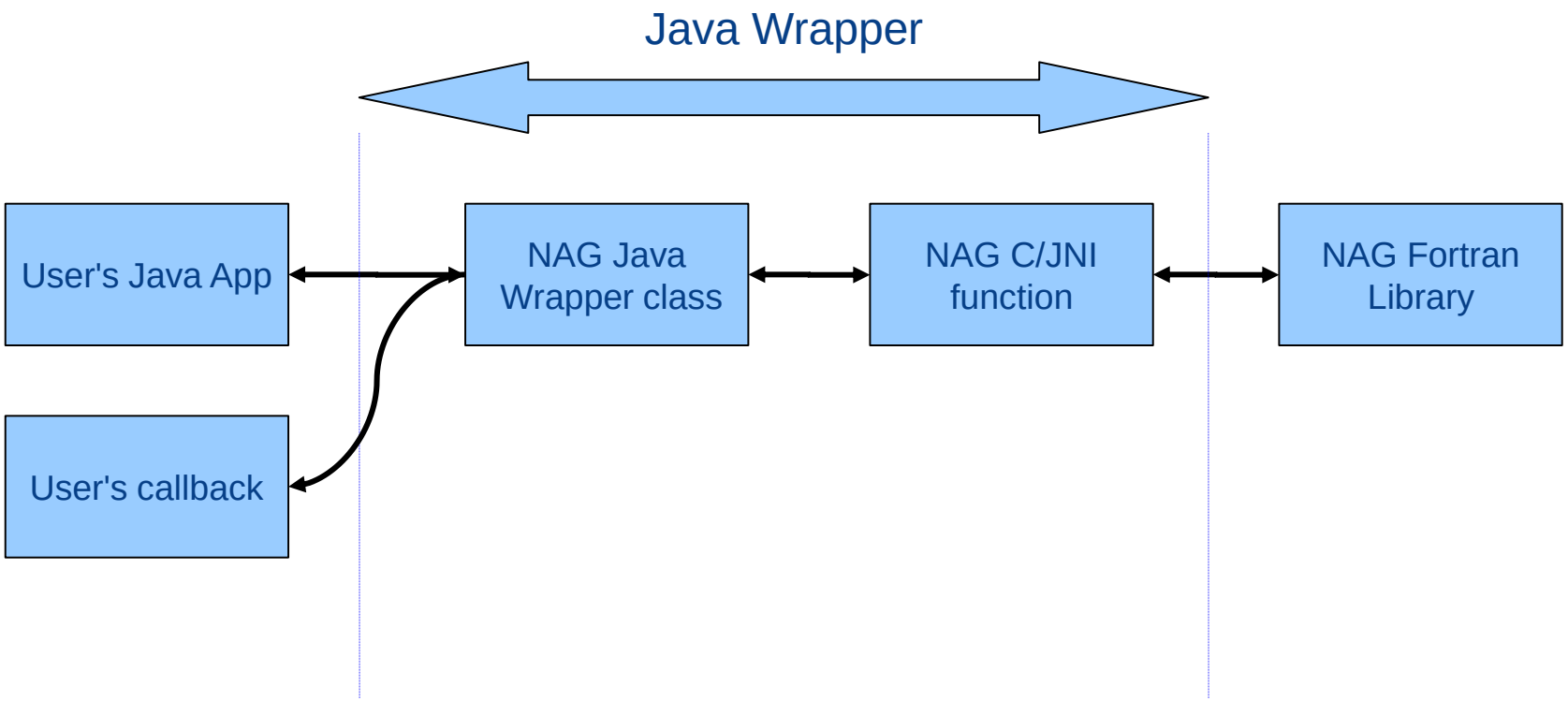
NAG and Java

Currently we steer the user to write their own wrappers and provide a few examples.

- Use JNI library to wrap calls to NAG routines
  - in either C Library or Fortran Library
  - Technical report with detailed examples available
    - online at www.nag.com
  - Wrappers for specific routines available from NAG
    - on request

For some of you this is NOT enough!

So coming soon…. The NAG Library *for Java*



Inside the wrapper

Java wrapper class

```java
package com.nag.routines;

import ...

public class S17ACJ extends Routine{


private double X;
private int IFAIL;


public double getX(){
        return this.X;
}
public void setX(double X){
        this.X=X;
}
public int getIFAIL(){
        return this.IFAIL;
}
public void setIFAIL(int IFAIL){
        this.IFAIL=IFAIL;
}

public S17ACJ( double X, int IFAIL){
/*Initialising arguments*/
        this.X = X;
        this.IFAIL = IFAIL;
}

public double eval(){
        Routine.ExtendedBuffer buff = new Routine.ExtendedBuffer(8 +4 ,2);

        buff.put(X);
        buff.put(IFAIL);
        Routine.ExtendedBuffer resultBuff = new Routine.ExtendedBuffer(8,1);
        byte[] result = resultBuff.getValues();
        this.S17ACF(buff.getIndexes(),buff.getValues(),result);

        //POST TREATMENT
        buff.rewind();
        this.X = buff.get(X);
        this.IFAIL = buff.get(IFAIL);
        double res  = 0.0;
 res = resultBuff.get(res);
        return res;
}
public double eval( double X, int IFAIL){

        this.X = X;
        this.IFAIL = IFAIL;
        return this.eval();
}

private native void S17ACF(int[] indexes, byte[] values, byte[] result);
```

C/JNI function

```c
//########## INCLUDES SECTION ##########
#include <naglibutilities.h>
#include <com_nag_routines_S17ACJ.h>
#include <string.h>
//######### END INCLUDES SECTION #########

//########## EXTERN SECTION ##########
extern double s17acf_(jbyte *arg_0 ,jbyte *arg_1 );
//######### END EXTERN SECTION #########

//########## GLOBAL VARIABLES SECTION ##########
//NO GLOBAL VARS REQUIRED
//######### END GLOBAL VARIABLES SECTION #########

//########## CALLBACKS SECTION ##########
//NO CALLBACK FOR THIS ROUTINE
//######### END CALLBACKS SECTION #########

//########## JNI FUNCTION SECTION ##########
JNIEXPORT void JNICALL Java_com_nag_routines_S17ACJ_S17ACF(JNIEnv *env, jobject obj,
                                                           jintArray indexes, jbyteArray values,jbyteArray result){
    jint *C_indexes = getIntArrayFromJava(env,indexes);
    jbyte *C_values = getByteArrayFromJava(env,values);
    jbyte *C_result = getByteArrayFromJava(env,result);


    double res = s17acf_(
    &C_values[C_indexes[0]],
    &C_values[C_indexes[1]]);

    memcpy((void*)&C_result[0],(void*)&res,8);
    setByteArrayToJava(env,values,C_values);
    setIntArrayToJava(env,indexes,C_indexes);
    setByteArrayToJava(env,result,C_result);
}
//######### END JNI FUNCTION SECTION #########
```

Calling the NAG Library *for Java*

```
import com.nag.routines.e04.E04GB;

[...]
public static main(String[] args){
   [...]
   LSQFUN lsqfun = new LSQFUN();
   [...]
   E04GB e04gb = new E04GB(m, n, lsqlin, lsqfun, lsqmon, iprint, maxcal, eta, xtol,
                           stepmx, x, fsumq, fvec, fjac, ldfjac, s, v, ldv, niter,
                           nf, iw, liw, w, lw, ifail);
   [...]
   e04gb.eval();
   ifail = e04gb.getIFAIL();
   [...]
}
private static class LSQLIN implements E04GBJ.E04GBJ_LSQLIN {
   [...]
       public void eval(int SELCT) {
           this.SELCT = SELCT;
           E04HEV e04hev = new E04HEV(SELCT);
           e04hev.eval();
           this.SELCT = e04hev.getLSQLIN_SELECT();
      }
   }
```

Many-core & GPU

- Developed CUDA software for Monte Carlo
  - new work for a PDE Solver for Stochastic Local Volatility progressing well
- Collaboration with Intel on MIC architecture
- Considerable expertise for training and consulting in
  - OpenCL
  - CUDA
  - use of libraries such as MAGMA and PLASMA
    - with plans to integrate into the NAG Library

NAG Library Development Continues….

- NAG Library for .NET
  - Following the success of the first release a second release is scheduled for 2013
  - new functions will include optimisation, NCM,…
  - Updated documentation for usage with VS2012
- Work already underway for Mark 24 of
  - NAG Fortran, NAG C, NAG Toolbox for MATLAB,..
- Algorithmic Differentiation (AD)
  - As well as providing training and consultancy we will be implementing AD versions of existing NAG Library functions

Finally….

- Thank you and……

Finally….

- Thank you and……

KEEP TELLING US WHAT YOU WANT