nag®

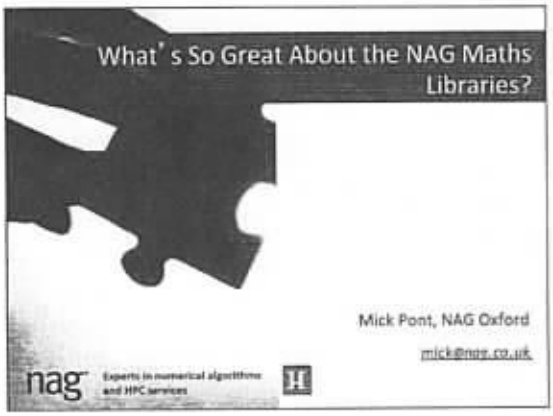
CSC / NAG Autumn School on

Core Algorithms in High-Performance Scientific Computing

NAG I

Mick Pont

What's so great about the NAG Maths Library



Agenda

- Intro to NAG
- NAG Libraries overview
- Some demos
- What's New
- Calling NAG from other environments
- Technical Support

Introduction to NAG

- Numerical Algorithms Group - Founded 1970
  - Co-operative software project: Birmingham, Leeds, Manchester, Nottingham, Oxford, and Atlas Laboratory
- Incorporated as NAG Ltd. in 1976
  - Not-for-profit
  - Based in Oxford, with offices in Manchester, Chicago, Tokyo, Taiwan
- Sometimes called "New Miracle Algorithms Group"
- Main product still the NAG Libraries
  - Also visualization, compiler, software tools
  - HECToR support

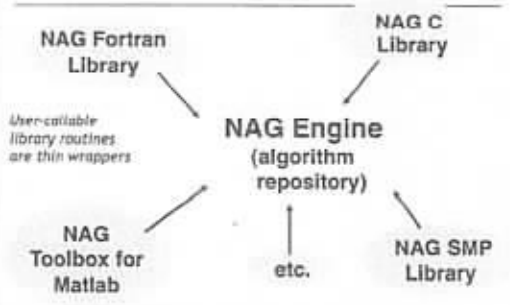
What do NAG do?

- Algorithmic development
  - Collaboration
- Software engineering
- Consultancy
- Implementation / porting
- Surely everything's been done now?
  - (everybody always thinks that)

NAG Library Contents Overview

- Root Finding
- Summation of Series
- Quadrature
- Ordinary Differential Equations
- Partial Differential Equations
- Numerical Differentiation
- Integral Equations
- Mesh Generation
- Interpolation
- Curve and Surface Fitting
- Optimization
- Dense Linear Algebra
  - BLAS, LAPACK
- Sparse Linear Algebra
- Correlation and Regression Analysis
- Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research
- Special Functions

The NAG Engine



Documentation

- Detailed manuals
- Every routine has an example program

<Show Mark 21 doc>

How do we decide what goes into libraries?

- Typically we get functionality requests from customers
  - via technical support calls
  - via salespeople
- We maintain a database of requests
- Probability of responding is weighted by number of requests, importance of customers, and difficulty of job

What's so hard about *that* then?

- Computer arithmetic makes things tricky.
  Numbers have finite precision and are subject to rounding error.
  e.g. on a decimal machine with 3 digits:

  a = 5.55 b = 6.56 c = 12.1

  a+b-c = 0.00 (due to rounding) instead of 0.01 in exact arithmetic
- We must also worry about overflow and underflow
  - numbers that become too large or too small to store

Example: PDF of Gamma Distribution

Probability density function (PDF)

$$f(x) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-x/b} \quad x \geq 0, \quad a, b > 0$$

looks easy to evaluate. But, depending on a, b and x, any of the quantities

$$b^a \quad \Gamma(a) \quad x^{a-1} \quad e^{-x/b}$$

might overflow or underflow, even if the end result is in a reasonable range.

Program code to check all contingencies can become monstrous!

We also have to worry about performance

- Tune up code
  - Use benchmarking programs
  - Use high-performance math libraries like ACML or MKL where possible
- Investigate possibilities for parallelism
  - OpenMP (for NAG SMP Library)
  - MPI (for NAG Cluster Parallel Library)
- But correctness is most important
  - How fast do you want the wrong answer?

What are the most important things in the NAG Library?

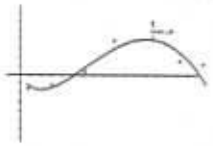
- Optimization (E04 / E05 chapters)
- Statistics (G chapters)
- Transforms (FFTs, Wavelets) (C06, C09 chapters)
- Linear Algebra (F chapters)
- Data fitting and approximation (E01 / E02 chapters)
- Differential Equations (D02 / D03 chapters)

Demos

- Curve and Surface Fitting
- Optimization
- Others if time

Chapter e02 – Curve and Surface Fitting

Problem to solve: given a set of data points, find the value of a function at points other than the data



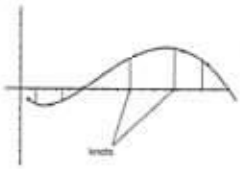
Unlike with interpolation, fitted function need not pass through data points

Typically, data contain random errors (e.g. from experimental measurement) – so interpolation is not appropriate

Smoothness of fitting function is likely to be desirable

e02 – Curve and Surface Fitting

Piecewise polynomial *splines* are useful.



Segments are joined with first and second derivative continuity at the joins (*knots*).

Chapter e02

- Data fitting usually involves minimizing the norm of the residuals
  - e.g. minimize largest residual
  - or minimize sum of squares of residuals
- Data points may be weighted according to their importance - bigger weight = more confidence
- Splines play an important role
  - Choice of knots may be crucial
  - Some routines are automatic – no need to choose knots
- Spline representation:

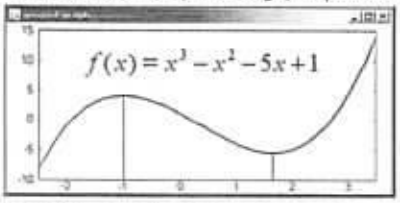
$f(x) = c_1N_1(x) + c_2N_2(x) + \ldots + c_pN_p(x)$

where $N_i(x)$ is a normalized cubic B-spline

<run spline1d_demo and Curve_and_surface_fitting/e02dc_demo here>

What do we mean by "Optimization"?

- For our purposes we mean "given a scalar real-valued mathematical function of n variables x_i, find values of the variables x that make the function as small (or as large) as possible"



How do we do it?

- We systematically choose values of the variables x_i from within the set of values that are allowed.
- Sometimes *all* values are allowed (unconstrained optimization)
- Sometimes some values are forbidden (constrained optimization)
  - e.g. $-3 <= x_2 <= 3$ or $x_2 > 0$
- Typically the user must supply a *starting point* $\underline{x}$

Are derivatives necessary?

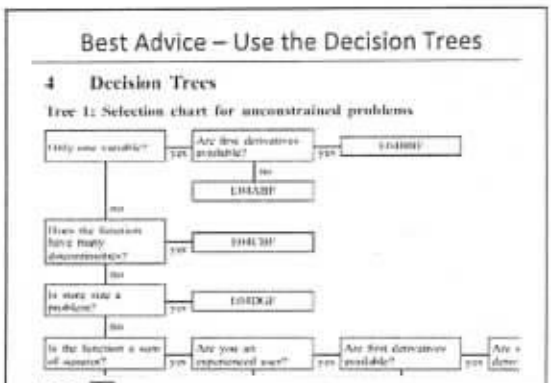
- Some algorithms require them
- Some don't
- Some prefer them but can manage without

<run Minimization/steepest_descent_demo and e04uc_demo here>

E04 Routine Classification

- Number of variables:
  - Single variable *f(x)*
  - Multiple variable *f(x)*
- Type of objective:
  - Linear
  - Quadratic or sum of squares
  - Nonlinear
- Constraints:
  - None
  - Simple bounds
  - Linear
  - Nonlinear

Best Advice – Use the Decision Trees



NAG Library – new at Mark 22 / 09

- Global optimization
- ANOVA - Analysis of Variance
- Nearest Correlation Matrix
- Partial Least Squares Regression Analysis
- Prediction intervals for fitted models
- Option Pricing
- Fast quantile selection routine
- Wavelets
- Adoption of LAPACK 3.1
- New RNGs
  - Scrambled Seq for QMC
  - Mersenne Twister
  - Sobol Sequence generator (50,000 dimensions)

NAG Software – work in progress...

- Bayesian Statistics
- Copulas (CL09)
- Extreme Value Theory Statistics (CL09)
- Generalised Mixed Effects Regression (CL09)
- Monte Carlo (Brownian Bridge)
- Optimisation (particle swarm, ...)
- Special Functions
- Time series
- ... and much more

NAG Libraries availability

Supporting Wide Range of O/S

- Windows
- Linux
- Mac
- Solaris
- AIX
- HPUX
- ..

Support leading compilers (Fortran, C & C++)

- NAG
- Intel, Pathscale, PGI
- GCC, GFortran
- Absoft, Lahey, Visual C++
- ..

NAG Libraries Ease of Integration

- C++ (various)
- C# / .NET
- Visual Basic
- Java
- Borland Delphi
- Python
- F#
- ... and more
- Excel
- MATLAB
  - and Octave, SciLab
- Mathematica
- Maple
- LabVIEW
- R and S-Plus
- SAS
- ... and more

NAG Technical Reports and Training

- New Technical Reports and additional support for:
  - NAG and R
  - NAG and Python
  - NAG and Bayesian Statistics
  - NAG and C# / .NET
  - NAG and Excel
  - NAG and Octave/SciLab
- Training courses for NAG clients
  - NAG Toolbox for MATLAB
  - NAG and Excel

NAG Technical Support

- support@nag.co.uk
  - *"NAG's support team is superb"* – Mike Croucher, University of Manchester

How do I get hold of NAG software?

- Two routes:
  - Via HECToR
  - Via existing licence *

* Most of you work at institutions which already have NAG site licences

- Contact operations@nag.co.uk for more information, or leave details today

Site licences allow:

- Installation on any university, staff or student machine
- Direct access to NAG support support@nag.co.uk
- Free training courses at NAG offices (or at your institution)