An Introduction to NAG Numerical Components

Overview
- An introduction to NAG
- Problems in numerical computation
- The NAG numerical libraries
- Some example routines
- What’s coming next?

NAG: The Numerical Algorithms Group
- 1970 - Nottingham Algorithms Group
  - formed to create numerical software for ICL computer
- 1971 - Mark 1 NAG Library released
- 1973 - NAG moved to Oxford
  - renamed Numerical Algorithms Group
- 1976 - Formation of NAG Ltd
  - a non-profit company
  - offices in US and Japan
  - distributors worldwide

NAG's products and users
- Products
  - Mathematical, statistical, data analysis components
    - NAG Numerical libraries
    - Fortran compiler and Windows IDE
    - HPC software engineering services
    - Consultancy work for bespoke application development
  - Users
    - Academic researchers
    - Professional developers
    - Analysts / modelers

Science, engineering, finance customers
- Major research organisations:
  - HPCVL, COSMOS (Cambridge), UK National Physical Laboratory, CERN, UK MetOffice, ECMWF, NERSC, Universities of Aachen, Bochum, Oxford, Stanford...
- Life sciences:
  - GlaxoSmithKline, Pfizer, National Institutes of Health...
- Finance:
  - BNP Paribas, Deutsche Bank, Fidelity, Goldman Sachs, IMF, JP Morgan, Lloyd's of London, UBS, Oxford Centre for Computational Finance...

High-End Computing Terascale Resource
- Latest high-end computing service for UK
  - funded by EPSRC, NERSC & BIS/RC
  - will run from 2007-2013
- Partners:
  - Hardware: Cray Inc
  - Service Provider: University of Edinburgh HPC Ltd
  - hardware hosting, user services, help desk
  - CSE Support: NAG Ltd
  - technical assessment of project application
    - porting / testing / optimisation of user codes
    - training courses, best practice guides, documentation, FAQs

The NAG Numerical libraries
- Contain mathematical & statistical components
  - ~ 1600 of them
  - ~ 65 of them
  - Available on variety of different platforms
  - Full documentation
    - printed and on-line
    - example programs
    - Used as building blocks by package builders
  - since 1971

NAG Fortran compiler & IDE
- NAG Fortran compiler
  - world's first Fortran 90 compiler
    - now includes Fortran 95 and most of Fortran 2003
    - regularly updated, fully supported
    - excellent (world's best) checking compiler
- IDE for NAG compiler on Windows PC
  - fully integrated with NAG Library
  - includes Fortran 90 library
  - includes program run-time
  - integrated debugger

http://www.nag.co.uk/
Access to NAG for UK academia

- New CHEST (Eduserv) Contract
  - started 1st August 2010
  - can join current agreement any time (on pro-rata basis)
- Order of licences is via Eduserv
- NAG happy to help with Eduserv paperwork!
- Most UK universities have major NAG licences
  - Signed up for a minimum of 3 or 5 years
  - Most popular platforms are Windows & Linux
  - Most popular option is all NAG Libraries + Fortran compiler
  - smaller option (e.g. NAG Library, one platform) also available

---

Access to NAG for UK academia, contd

- What do these licences get you?
  - Unlimited usage of NAG software, on staff and students’ desktops, home machines, and personal laptops
  - Direct access to the NAG support desk: support@nag.co.uk
- Our software
  - Includes online documentation – also www.nag.co.uk
  - Supplied with extensive example programs
  - data and results (see later)

---

NAG at Middlesex

- One licensed C Library Mark 7 User, Dr Enver Ever
  - kindly helped organising seminar today – many thanks
  - provided user story for NAG website: http://www.nag.co.uk/market/articles/nag_communications_systems.pdf
- Middlesex users who wish to use NAG components...
  - NAG will offer extended trials till 31st December 2011
  - for all students / staff
  - to encourage proper evaluations of our software
  - before considering a department or site licence

---

Overview

- An introduction to NAG
- Problems in numerical computation
  - The NAG numerical libraries
  - Some example routines
  - What’s coming next?

---

Why bother?

- Numerical computing: difficult to do accurately
- Problems of
  - Overflow / underflow
  - How does computation behave for large / small numbers?
  - Condition
  - How is it affected by small changes in the input?
  - Stability
  - How sensitive is the computation to rounding errors?
- Importance of
  - error analysis
  - Information about error bounds on solution

---

The sine function

\[
\sin(x) = \begin{cases} 
0 & \text{for } x = 2\pi \text{ or } 0 \\
1 & \text{for } x = \pi/2 \\
-1 & \text{for } x = -\pi/2 \\
\end{cases}
\]

---

\[
\pi = 3.1415926535897932
\]

---

\[
n = \frac{x}{2\pi} \\
x' = x - n \cdot 2\pi
\]

\[
x' \in [-\pi, \pi]
\]
An Introduction to NAG Numerical Components

Overview

- An Introduction to NAG
- Problems in numerical computation
- The NAG numerical libraries
- Some example routines
- What's coming next?

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

Origins of NAG algorithms

- Collaboration with academic community
  - experts in fields donate code to NAG
- Participation in funded research projects
  - UK (e.g. dept of Trade & Industry)
  - EU (e.g. Framework programme)
  - US (e.g. NSF, ARPA)
- In-house development and implementation

The NAG Numerical libraries

- NAG Fortran Library
- NAG C Library
- NAG Fortran 90 Library
- NAG Library for .NET
- NAG Library for SMP & multicore
  - for symmetric multi-processor machines (OpenMP)
- NAG Parallel Library
  - for distributed memory parallel machines (MPI)
- NAG Toolbox for MATLAB
- NAG Data Mining Components

Other NAG library interfaces

- C
- C++
- C# / .NET
- Fortran
- Java
- Python
- Visual Basic
- CUDA
- OpenCL
- R
- Burland Delphi
- ...
- Excel
- MATLAB
- Octave, SciLab, FreeMat ...
- Maple
- Mathematica
- SciLab
- PowerBuilder
- LabVIEW
- R and S-Plus
- SAS
- Simulink
- ...

NAG library documentation

- Each library has complete documentation
  - distributed as PDF and HTML
- Each chapter has an introduction
  - technical background to the area
  - assistance in choosing the appropriate routine
- Each routine has a self-contained document
  - description of method
  - specification of each parameter
  - explanation of error exit
  - remarks on accuracy
- Document contains example program
  - to illustrate use of routine
  - source, input data, output
- Examples are also part of installation

NAG library documentation (2)

- Each implementation has a user's note
  - implementation = hardware + O/S + compiler
- User's note contains implementation-specific info
  - how to access the library
  - linker options
  - routine-specific information
- Each implementation also has an installer's note
  - how to install it
  - details of how it was built & tested
NAG Toolbox for MATLAB

- Set of interfaces to NAG Fortran Library
- Fully integrated into MATLAB
  - many routine arguments become optional
  - easier to read code
  - can be used with NVML compiler
  - complete documentation for each routine
  - including examples

NAG Toolbox help chapters
MATLAB formatting
NAG formatting (in PDF)

Overview

- An Introduction to NAG
- Problems in numerical computation
- The NAG numerical libraries
- Some example routines
- The NAG SMP library
- What’s coming next?

NAG Library Contents

- Basic Linear Algebra
  - Systems of linear equations
  - Eigenvalue and singular value problems
  - Matrix inversion and determinant calculation
- Several routines to choose from
  - depending on characteristics of the matrix
  - 'Black Box' or 'General Purpose' interfaces

Linear Algebra chapters

- f01 - matrix operations, inc. inversion
- f02 - eigenvalues and eigenvectors
- f03 - determinants
- f04 - simultaneous linear equations
- f05 - orthogonalization
- f07 - linear equations (LAPACK)
- f08 - least squares & eigenvalues (LAPACK)
- f11 - large-scale linear systems
- f12 - large-scale eigenproblems

f – Linear Algebra

- Solutions to a wide variety of linear problems:
Linear equations
- f04 & f07 have Black Box routines for solving $Ax = b$ and $AX = B$
  - With $A$ a $m \times n$ real or complex nonsingular matrix
  - Or, use a matrix factorization routine from f03 or f03, plus a solve routine from f04
  - Or, use a matrix factorization plus a solve from f07
  - If $A$ is large and sparse, use routines from f11

Which routine?
- Depends on characteristics of $A$
  - General
  - Symmetric or Hermitian
  - Symmetric or Hermitian positive definite
  - Banded
  - Skew-Hermitian
  - Sparse
  - Waste of resource to use most general routine
  - Or, use a matrix factorization routine from f02 or f08, or e04 routine
  - Or, use a matrix factorization plus a solve from f04 or f08
  - Use decision tree in f04 introduction
  - Points to f07 and f11

Example
$$ AX = B $$
$$ A = \begin{pmatrix}
3.0 & 2.1 & 0.0 & 0.0 & 0.0 \\
3.4 & 2.3 & -1.0 & 0.0 & 0.0 \\
0.0 & 3.6 & -3.0 & -0.0 & 0.0 \\
0.0 & 0.0 & 7.0 & -0.9 & 8.0 \\
0.0 & 0.0 & 0.0 & -6.0 & 7.1 
\end{pmatrix} $$
$$ B = \begin{pmatrix}
2.7 & 6.6 \\
-0.5 & 10.8 \\
2.6 & -3.2 \\
0.6 & -11.2 \\
2.7 & 19.1 
\end{pmatrix} $$

Linear least squares problems
- f04 & f08 have Black Box routines for solving
  $$ \min r^T r \quad r = b - Ax $$
  - With $A$ a $m \times n$ (possibly rank deficient) matrix
  - Or, use a matrix factorization routine from f02 or f08, plus a solve routine from f04 or f08
  - Use decision tree in f04 introduction
  - Or, use decision tree in f02 introduction

Which routine?
- Depends on characteristics of $A$
  - General
  - Symmetric
  - Hermitian
  - Banded
  - Skew-Hermitian
  - Use decision tree in f04 introduction, and advice in f08 introduction

Eigenvalue & singular value problems
- f02, f08 & f12 have Black Box routines for solving $Ax = \lambda x$
  - With $A$ a $m \times n$ real or complex matrix, and $Ax = \lambda Bx, \ ABx = \lambda x$
  - Black Box routines for finding singular values and/or singular vectors of $A$, $m \times n$ real or complex matrix are in the same chapters

Example
Is $A$ real?
- Yes

Is it symmetric?
- Yes

Is it banded?
- Yes

Is it tridiagonal?
- Yes

Use f07/ea
- Or (f04/e) (condition no. + error), or f07/e (closed error)

Linear least squares problems
- f04 & f08 have Black Box routines for solving
  $$ \min r^T r \quad r = b - AX $$
  - With $A$ a $m \times n$ (possibly rank deficient) matrix
  - Or, use a matrix factorization routine from f02 or f08, plus a solve routine from f04 or f08
  - Use decision tree in f04 introduction
  - Or, use decision tree in f02 introduction

Which routine?
- Use decision tree in f04 introduction
  - Or, use g02 routine (inc. statistical information)
  - Or, use f08 routine
    - Linear equality constrained least-squares problem
    - General Gauss-Markov linear model problem
  - Or, use e04 routine
    - General linearly constrained linear least-squares problem

Matrix inversion, determinant calculation
- f01 and f08 contain routines for inverting matrices
  - Use decision tree in f02 introduction (points to f07)
- Don't use inversion to solve linear equations or least-squares problems
  - Other routines are faster, more stable and accurate
  - F03 contains routines for calculating determinants
    - Use decision tree in f02 introduction
An Introduction to NAG Numerical Components

- f01 contains routines for
  - matrix transposition
  - matrix addition
  - matrix multiplication
  - conversion between matrix storage formats.

- f06 also contains routines for matrix manipulation
  - computing matrix norm
  - performing a dot product
  - setting up a plane rotation
  - ...

Other matrix operations

- d03 – Partial differential equations
  - General linear 2nd order PDE
    \[ a_{11} u_{xx} + 2a_{12} u_{xy} + a_{22} u_{yy} + a_{10} u_x + a_{20} u_y + f(x,y) = 0 \]
  - Is \( w^{-1} \) positive? \( \Rightarrow \) equation is elliptic
  - negative? \( \Rightarrow \) equation is hyperbolic
  - zero? \( \Rightarrow \) equation is parabolic
  - Geometry & dimensionality of domain of interest
  - Subsidiary conditions
    - Values on domain boundary, and/or at initial time

Elliptic equations
- E.g., Poisson equation
  - \[ -\nabla^2 u + \phi = 0 \]
  - with Dirichlet boundary condition \( u = f \)
  - or Neumann boundary condition \( \nabla u \cdot n = g \)
  - or generalized boundary value problem \( \lambda \nabla^2 u + \phi = g \)
- d03e solves on 2D and 3D domains

Hyperbolic equations
- E.g., wave equation
  - \[ u_{tt} - c^2 u_{xx} = f (x,t) \]
  - with initial conditions
    - \( u(x,0) = f \)
    - \( u_t(x,0) = g \)
  - initial value (or Cauchy) problem
- d03hp, d03hp, d03hp solve convection-diffusion eqns
  - with optional source terms

Parabolic equations
- E.g., diffusion equation
  - \[ u_t - \nabla^2 u = f \]
  - with subsidiary conditions
    - \( u(x,0) = f \)
    - \( u_t(x,0) = g \)
  - mixed initial/boundary value problem
- d03j, d03j, d03j solve general parabolic eqns in 1D

Other d03 solvers
- Black-Scholes equation
  - d03ca, d03ca
  - Also see s30 for closed-form option-pricing solutions
- 1st order systems in 1D (linear time derivative)
  - d03pa, d03pa, d03pa
- Nonlinear 2nd order time-dependent systems in 2D
  - d03da, d03da

Mesh generation
- d03ma generates a triangular mesh over 2D domain
  - Also see d06 for mesh generation routines
    - d03aa meshes domain using incremental method
    - d03ac meshes domain using Delunay-Voronoi method
    - d03ad meshes domain using Advancing Front method

Alternative / additional approaches
- d03 routines
  - use finite differencing
  - applicable to specific types of PDEs
  - on domains having regular geometries
- Use finite-element method
  - subdivide domain into mesh
    - using d06 routine
    - transform PDE into set of simultaneous equations
      - solve linear systems using f11 (large sparse systems)
      - solve non-linear systems using Newton's method
Solves heat diffusion equation over 2D domain
User specifies boundary
- Can contain holes, but boundary can’t self-intersect
- Use f06a to check this
d06de generates boundary mesh
- Domain meshed by d06aa, d06ab or d06ac
User specifies temperature on boundary
- I.e. Dirichlet boundary conditions

Use f11 to solve linear system
f12zb reorders non-zero matrix elements
- for symmetric matrices
- more efficient
f13ja performs incomplete Cholesky factorization
- preconditioning step increases speed & efficiency of solver
f13jc solves real sparse symmetric linear system
- uses either conjugate gradient or Lanczos method
- black box routine
- Calls five other f11 routines

Coordinate transform allows for sample point reordering
Far field boundary gives inflow / outflow conditions
Routines from other chapters may be relevant
User specifies temperature on boundary
Domain meshed by d06ac

Reads Stokes equation over 2D domain
solves real sparse symmetric linear system
f11za reorders non-zero matrix elements
- more efficient
Operations research

Transportation
Domain meshed by d06aa

Shortest path calculation
h – Operations research
Solves problems in
- Integer programming
- Transportation
- Shortest path calculation
- Routines from other chapters may be relevant

General linear programming
Solve
\[
\max F(x) = \sum c_i x_i \\
x = (x_1, x_2, \ldots, x_n) \geq 0
\]
(possibly) subject to constraints:
\[
\sum x_j = 0, j = 1, 2, \ldots, n_i \\
\sum x_j = 0, j = 1, 2, \ldots, n_i \\
\sum x_j = 0, j = 1, 2, \ldots, n_i \\
\]
An Introduction to NAG Numerical Components

Integer programming
- Some or all of the elements of \( \mathbf{x} \) must be integers
- Chapter e04 handles general (non-integer) problems
- IP problems may or may not have a solution
- Solution may or may not be unique

Chapter e04 handles general (non-integer) problems
- IP problems may or may not have a solution
- Solution may or may not be unique

Random number generation
- Used to model real-life processes
- Sequences have to be very long
- Humans are bad at choosing them
  - eg faking a random sequence of coin tosses is difficult
  - Most people’s sequences would be easily proved non-random
- Hence, good algorithms are required

Random number generation
- Used to model real-life processes
- Sequences have to be very long
- Humans are bad at choosing them
  - eg faking a random sequence of coin tosses is difficult
  - Most people’s sequences would be easily proved non-random
- Hence, good algorithms are required

Two types of random sequence
- **Pseudo-random**
  - Negligible correlation between successive values
  - Statistical properties as close as possible to true randomness
  - Eg. time elapsed between clicks of Geiger counter
- **Quasi-random**
  - Designed to give a more even distribution in space
  - Maximally avoiding
  - “Looks more random”
  - Better suited for Monte Carlo methods
  - More accurate results than a pseudo-random sequence

Pseudo-random generator
- Multiplicative congruential algorithm
  \[ n_i = (a \times n_{i-1}) \mod m \]
  \[ n_i = n_i / m \]
- NAG generator uses
  \[ a = 13 \times 2^{31}, m = 2^{59} \]
  giving a cycle length of \( 2^{59} \)
  - sequence starts repeating itself after this
  - don’t use more than \( 2^{28} \) numbers in one experiment

Example: g05fa
- Uses algorithm selected by call to g05za
  - Multiplicative congruential, or Wichmann-Hill
  - Or uses default (implementation dependent)
- Generates a vector of numbers
  - Equal to successive calls of g05kaf (but more efficient)
- In Fortran:
  ```fortran
  SUBROUTINE G05FAF(A, B, N, X)
  INTEGER N
  DOUBLE PRECISION A, B, X(N)
  ```
- In MATLAB:
  ```matlab
  [x] = g05fa(a, b, n)
  ```
  \( x \) is the array of numbers
  \( n \) is its length

Example: g05yd
- Quasi-random number generator
  - initialized by calling g05yc
- Generates a vector of numbers
  - In Fortran:
    ```fortran
    SUBROUTINE G05YDF(N, X, IREF, IFAIL)
    ```
  - In MATLAB:
    ```matlab
    [x, irefOut, ifail] = g05yd(n, iref)
    ```
    \( x \) is the array of numbers
    \( n \) is its length

Other routines in g05
- Random numbers from other distributions
  - Make these from a uniform random sequence by
    - Transformation methods
    - Rejection methods
    - Accept random numbers from an envelope distribution (similar to required distribution) with some probability
    - Table search methods
  - For discrete distributions
  - Uses table of cumulative probabilities
  - Searches for given random number in table

Other routines in Chapter g05 (2)
- g05da: pseudo-random from uniform distribution
- g05db: exponential distribution
- g05dd: normal distribution
- g05de: log-normal distribution
- g05df: Cauchy distribution
- g05hf: chi^2 distribution
- g05j: Student’s t distribution
- g05rf: Poisson distribution
- many other distributions
  - plus GARCH, random matrices, copula, etc.
Copulas...

- ...are used to simulate multivariate distributions
  - by linking them to univariate distributions
- An m-D distribution function can be represented
  \[ f(x_1, x_2, \ldots, x_m) = f_1(x_1)f(x_2, \ldots, f_m(x_m)) \]
  with marginal distributions
  \[ f(x_1), f(x_2), \ldots, f_m(x_m) \]
  and the copula distribution

Copulas...

- ...used to simulate multivariate distributions
  - by linking them to univariate distributions
- An m-D dist’n function can be represented
  \[ f(x_1, x_2, \ldots, x_m) = f_1(x_1)f_2(x_2, \ldots, f_m(x_m)) \]
  with marginal distributions
  \[ f_1(x_1), f_2(x_2), \ldots, f_m(x_m) \]
  and the copula distribution
  \[ c(u_1, u_2, \ldots, u_m) = f(c_1(u_1), c_2(u_2), \ldots, c_m(u_m)) \]

Using a copula

- Copula describes correlation between variables
  - Gaussian, Student’s t, etc
- Gaussian copula often used in financial modelling
- Marginals define distribution within each variable
  - Uniform, Beta, Gaussian, etc
- Each variable can have a different type of distribution

Copula example

- Construct a bivariate distribution
  - using Gaussian copula + 2 Beta marginal distributions
  - g03ad generates copula
  - g03db computes Beta distribution
- Distribution characterised by values for
  - correlation coefficient
  - Beta distribution parameters

A multivariate dataset has

- several variables measured for
  - E.g., height, weight, age for a class of schoolchildren
- Two classes of method
  - Variable-directed
  - Individual-directed

Variable & individual-directed methods

- Variable-directed
  - What are the relationships between the variables?
  - Can a smaller set of variables still represent the data?
  - E.g., are height and weight strongly correlated?
- Individual-directed
  - What are the relationships between the objects?
  - How “far apart” are they from each other?
  - Use distance to
    - Group: i.e., collect together according to closeness
    - Classify: i.e., assign objects to nearest group
  - Map: i.e., show distances between objects
  - E.g., are boys different from girls?

Principal component analysis - g03aa

- Find new variables which
  - are linear combinations of the observed variables
  - have maximum variation, and are uncorrelated
- Canonical variate analysis - g03ac
  - Examines relationship between groups of variables
- Canonical correlation analysis - g03ad
  - Examines relationship between two sets of variables
- All use SVD, or eigenvalue decomposition

Individual-directed methods

- Compute distance matrix for objects - g03ea
- Then use distances to
  - Group objects (cluster analysis)
    - Hierarchical (g03eb), or non-hierarchical (g03ed) methods
  - Classify objects (discriminant analysis)
    - Given a training set of objects which are already grouped, allocate a new object to one of these groups
    - g03ad, g03eb, g03fc
  - Map objects (scaling methods)
    - Represent the distances in D-dimensional Euclidean space
    - g03da, g03fc

Variable & individual-directed methods

- Principal component analysis - g03aa
  - Find new variables which
    - are linear combinations of the observed variables
    - have maximum variation, and are uncorrelated
- Canonical variate analysis - g03ac
  - Examines relationship between groups of variables
- Canonical correlation analysis - g03ad
  - Examines relationship between two sets of variables
- All use SVD, or eigenvalue decomposition
Multivariate methods example

- How many species of water vole (Arvicola) in UK?
- Measurement data
  - Presence / absence of 13 skull characteristics
  - 300 observations, each in one of 14 regions
  - 3 groups:
    - A. terrestris / A. sapidus / unclassified UK cases
- Treatment
  - Average data within each region
  - Gives 14 data points in 13 dimensions
  - 13 variables measured for 14 regions
  - How to display dataset?

2D scatterplots

- 2D scatterplots?
  - Structure is unclear
  - (13 x 12) / 2 = 78 plots needed
- Principal component analysis?
  - 2 PCs explain 49% of the variance
  - 3 PCs explain 65% of the variance
  - Should be > 85% for confident representation
- Fisher’s iris dataset (4 variables) is 95%

Analysis

- 14 data points - one for each region
  - Each point has values for 13 variables
- Construct 14 by 14 distance matrix, \( \Delta \) (g03ea)
  - \( \Delta_{ij} \) = distance between points \( i \) & \( j \) in 13D space
  - \( \Delta \) is symmetric, with zero diagonal elements
- Want to find a new matrix, \( \Delta^* \)
  - set of 14 new data points in 3D space that preserve \( \Delta \)
- Project \( \Delta \) to \( \Delta^* \) using metric scaling (g03fa)
- Display data points in 3D

Exploratory data analysis conclusions

- 2D scatterplots don’t indicate group structure
- cf. iris dataset
- 3D PCA unreliable here
- Metric scaling of \( \Delta \) reduces D from 13 to 3
- 3D visualization reveals group structure
  - Distinct A. sapidus group
  - UK sample represents only A. terrestris
e04 – Optimization

Optimization is the minimization of a function

\[ \min_F(x) \]

**F** is called the objective function

Want to determine \( x \) that minimizes it

Constrained problem

**Might be subject to bound constraints, e.g.**

\[ 0 \leq x \leq 1 \]

\[ -2 \leq x \leq 3 \]

**or linear constraints, e.g.**

\[ 3x_1 + 2x_2 \leq 4 \]

**or nonlinear constraints, e.g.**

\[ x_1^2 + x_2^2 + x_3 \geq 2 \]

Categorizing problems

**Using a single, all-purpose method is inefficient**

**Important to classify optimization problems**

\( \bullet \) according to the properties of

- objective function
- constraints

**Then need to choose a appropriate method**

\( \bullet \) for your problem

\( \bullet \) to ensure efficiency, & best chance of success

Categorizing problems (2)

**Properties of objective function**

\( \bullet \) linear

\( \bullet \) sum of squares of linear functions

**Properties of the constraints**

\( \bullet \) none

\( \bullet \) bounds

\( \bullet \) linear

\( \bullet \) nonlinear

Special cases

**Linear** \( F \) + linear constraints

**Quadratic** \( F \) + linear constraints

**Nonlinear** \( F \) + nonlinear constraints

Non-linear problem example

**Minimize**

\[ F(x_1, x_2) = (1-x_2)^2 + 100(x_2-x_1^2)^2 \]

This \( F \) is called Rosenbrock\'s function

\( \bullet \) good test for optimizers

\( \bullet \) minimum is

\[ F(1,1) = 0 \]

Rosenbrock\'s function

Other information

**Some methods require derivatives of** \( F \)

**Gradient vector**

\[ g(x) = \left( \frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_n} \right)^T \]

**Hessian matrix**

\[ G(x) = \frac{\partial^2 F}{\partial x_i \partial x_j} \]
Example: e04wd
- Minimises an arbitrary smooth function
  - subject to bound, linear or nonlinear constraints
  - or no constraints at all
- user should supply first derivatives
- otherwise, they get approximated by finite differences
- uses forward communication for evaluation
- user supplies routines to calculate
  - objective function
  - constraints
- library contains auxiliary functions for e04wd
- initializing it, getting & setting its parameters

Problems in numerical computation

In MATLAB:
- uses objective function

In C:
- user supplies routines to calculate
  - objective function
  - derivative

Some example routines
- e04ab: function of one variable, no derivatives
- e04cc: ditto, using 1st derivative
- e04dg: minimization using conjugate gradients
- e04j: linear programming
- e04qc: linear least squares
- e04qc: quadratic programming
- e04mc: LP or QP for sparse problems
- e04uc: constraints, forward communication
- e04uf: constraints, reverse communication
- e04uc: nonlinear least squares

Routines for non-linear least squares
- e04fc: unconstrained, no derivatives
- e04fy: unconstrained, no derivatives, easy to use
- e04gb: unconstrained, 1st derivatives
- e04gy: unconstrained, 1st derivatives, easy to use
- e04he: unconstrained, 2nd derivs.
- e04hd: unconstrained, 2nd derivs., easy to use
- e04js: nonlinear constraints, optional 1st derivs.

Overview
- E04AB: unconstrained, no derivatives
- E04CC: ditto, using 1st derivative
- E04DG: minimization using conjugate gradients
- E04J: linear programming
- E04QC: linear least squares
- E04QC: quadratic programming
- E04MC: LP or QP for sparse problems
- E04UC: constraints, forward communication
- E04UF: constraints, reverse communication
- E04UC: nonlinear least squares

Special case
- Non-linear least squares problem
  \[ F(x) = \sum_{i} f_i^2(x) \]
  some routines require the Jacobian matrix
  \[ J_i(x) = \frac{\partial f_i(x)}{\partial x_j} \]
Future (1)

- Continued algorithmic development
  - 118 new routines added at Mark 23
  - 122 were added at Mark 22
- Mark 23 includes:
  - Bound Optimization by Quadratic Approximation
  - Quantile Linear Regression
  - Sampling with unequal weights
  - Nearest correlation matrix
  - Copulas
  - Partial least squares / Ridge regression
  - ...

Future (2)

- Now key criteria for selecting future algorithms
  - potential for parallelism
  - Example: numerical optimisation
    - Current algorithms written for serial performance
    - not suitable for parallelism
    - Currently working on a swarm optimisation algorithm
    - Stochastic method
    - Poor performance on one thread
    - but scales extremely well
    - PSO will complement, not replace existing routines

NAG research and development

- Routines for SIMD architectures (GPUs etc)
  - Early successes with RNGs on NVIDIA hardware
  - NAG CUDA code available to academic institutions
    - at no charge
    - under a collaborative licence
- Investigating new languages

Conclusions

- NAG offers software components for developers
  - no wheel-reinvention, stone canoes, chocolate teapots
- Portable
  - constantly being implemented on new architectures
  - made accessible from different software environments
- Scalable
  - same user code will work with 1, 2, 4, ... processors
  - unobtrusive parallelism with the SMP Library
- Reliable
  - extensive experience at implementing numerical code

NAG key contacts

- Technical support and help
  - support@nag.co.uk
- UK academic account manager
  - louise.mitchell@nag.co.uk
  - Tel: 01865 518047
- Today's speaker:
  - jeremy.walton@nag.co.uk