NAG and Automatic Differentiation

Jeremy Walton, Ian Hounam

NAG Ltd, Oxford

Results Matter. Trust NAG

July 17, 2009 Automatic Differentiation Workshop, Oxford
Overview

- An introduction to NAG
- Numerical libraries
  - Library interfaces
    - The NAG Toolbox for MATLAB
  - New features
- Using AD with the NAG library
- The NAG Fortran compiler
  - Using AD with the NAG compiler
Overview

- An introduction to NAG
- Numerical libraries
  - Library interfaces
    - The NAG Toolbox for MATLAB
  - New features
- Using AD with the NAG library
- The NAG Fortran compiler
  - Using AD with the NAG compiler
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
- 1980 – NAG Ltd financially self-sufficient
  - offices in US and Japan
  - distributors worldwide
NAG’s products and users

- **Products**
  - Mathematical, statistical, data analysis components
    - NAG Numerical libraries
  - NAG Fortran Compiler
  - 3D visualization
  - HPC software engineering services
    - HECToR support
  - Consultancy work for bespoke application development

- **Users**
  - Academic researchers
  - Professional developers
  - Analysts / modelers
Overview

- An introduction to NAG
- Numerical libraries
  - Library interfaces
    - The NAG Toolbox for MATLAB
  - New features
- Using AD with the NAG library
- The NAG Fortran compiler
  - Using AD with the NAG compiler
The NAG Numerical libraries

- Contain mathematical and statistical components
  - ~ 1600 of them

- Available on variety of different platforms
  - ~ 65 of them
  - stringently tested on each platform

- Full documentation
  - On-line
  - example programs

- Used as building blocks by package builders
  - since 1971
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
- 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
The NAG Numerical libraries

- NAG Fortran Library
- NAG C Library
- NAG Fortran 90 Library
- NAG SMP Library
  - for symmetric multi-processor machines (OpenMP)
- NAG Parallel Library
  - for distributed memory parallel machines (MPI)
- Maple-NAG Connector
- NAG Toolbox for MATLAB
- NAG Data Mining Components
Other NAG library interfaces

- Fortran
- C
- C++
- C# / .NET
- Java
- Borland Delphi
- Python
- Excel
- MATLAB
- Octave
- Maple
- LabVIEW
- R and S-Plus
- SAS
- Simfit
- ...

Results Matter. Trust NAG
July 17, 2009
Automatic Differentiation Workshop, Oxford
NAG Toolbox for MATLAB

- Comprehensive interfaces to NAG Fortran Library
- Fully integrated into MATLAB
  - many routine arguments become optional
    - easier to read code
  - complete documentation for each routine
    - including examples
- Complementary functionality to MATLAB
- Alternative to several specialist toolboxes
NAG Toolbox: d03eb

1 Purpose

d03eb uses the Strongly Implicit Procedure to calculate the solution to a system of simultaneous algebraic equations of five-point molecule form on a two-dimensional topologically-rectangular mesh. (Topological means that a polar grid, for example, can be used, being equivalent to a rectangular box.)

2 Syntax

\[
[t, itcon, itcsov, residu, chges, ifail] = \text{d03eb}(m, n, x, y, f, i, j, k, l, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, ifail, \text{a}, \text{b}, \text{c}, \text{d}, \text{e}, \text{f}, \text{g}, \text{h}, \text{i}, \text{j}, \text{k}, \text{l}, \text{m}, \text{n}, \text{o}, \text{p}, \text{q}, \text{r}, \text{s}, \text{t}, \text{u}, \text{v}, \text{w}, \text{x}, \text{y}, \text{z}, \text{ifail})
\]

3 Description

Given a set of simultaneous equations:

\[
M \cdot \phi = q
\]

(which could be nonlinear) derived, for example, from a finite difference representation of a two-dimensional elliptic partial differential equation and its boundary conditions, the routine determines the values of the dependent variable \( \phi \) is a known vector of length \( n_1 \cdot n_2 \) and \( M \) is a square \( (n_1 \cdot n_2) \times (n_1 \cdot n_2) \) matrix.

The equations must be of five-diagonal form:

\[
\begin{align*}
&\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j} = q_{i,j} \quad \text{for } i = 1, 2, \ldots, n_1 - 2, j = 1, 2, \ldots, n_2 - 2, \\
&\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1} = q_{i,j} \quad \text{for } i = 1, 2, \ldots, n_1 - 1, j = 1, 2, \ldots, n_2 - 2.
\end{align*}
\]

For example, if \( n_1 = 3 \) and \( n_2 = 2 \), the equations take the form:

\[
\begin{bmatrix}
2 & 1 & 1 \\
1 & 2 & 1 \\
1 & 1 & 2
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix}
= 
\begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix}
\]

The system is solved iteratively, from a starting approximation \( \phi^0 \), by the formula:

\[
\phi^{(k+1)} = \frac{1}{\lambda} \left( q - M \phi^{(k)} \right)
\]
How to call the NAG routine

Calling the routine in MATLAB

MATLAB plot
New in Mark 22: global optimization

- Currently many optimization routines in NAG
  - but these have all been for local optimization
  - No guarantee about which minimum is returned
- Users have asked for *global optimization* methods
  - 'multilevel coordinate search' (Huyer & Neumaier)
- New chapter in library for global optimization
Local / global optimization
New in Mark 22: wavelet analysis

Analysis of foreign exchange market data:

GBP-EUR, June 2005 – June 2007, daily average

- Data shows activity on several time and frequency scales
- Wavelet Multi-Resolution Analysis (MRA) can extract information at selected scales
Other new routines in Mark 22 (1)

- **c05** – Roots of equations
  - Lambert’s W function
- **d02** – Ordinary Differential Equations
  - Index-2 differential algebraic equations
- **f01** – Matrix Operations
  - Matrix exponential
- **f02** – Eigenvalues and Eigenvectors
  - Obtaining leading terms in SVD of real general matrix
Other new routines in Mark 22 (2)

- **g01** – Simple calculations on statistical data
  - Fast quantile routine
- **g02** – Correlation and Regression Analysis
  - Partial least squares
  - Ridge regression
  - Nearest correlation matrix
- **g03** – Multivariate methods
  - ProMax rotations
Other new routines in Mark 22 (3)

- **g05** – Random Number Generators
  - Pseudo-random
    - Mersenne Twister
    - Wichmann-Hill
    - ACORN
    - Multiple independent sequences (skip-ahead, leap-frog)
  - Quasi-random
    - Sobol (up to 50,000 dimensions)
    - Scrambled sequences
Other new routines in Mark 22 (4)

- **g13** – Time series analysis
  - Exponential smoothing of univariate time series

- **m01** – Sorting and searching
  - Searching arrays of values
    - Double precision numbers
    - Integers
    - Characters
Overview

- An introduction to NAG
- Numerical libraries
  - Library interfaces
    - The NAG Toolbox for MATLAB
  - New features
- Using AD with the NAG library
- The NAG Fortran compiler
  - Using AD with the NAG compiler
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
- 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
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
- 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
Example NAG chapter: d02

- Integrates systems of ODEs

\[
\frac{dy_1}{dx} = g_1(x, y_1, \ldots, y_n) \\
\frac{dy_2}{dx} = g_2(x, y_1, \ldots, y_n) \\
\vdots \\
\frac{dy_n}{dx} = g_n(x, y_1, \ldots, y_n)
\]
Example NAG routine: d02nb

- Used for *stiff* ODE systems whose Jacobian
  \[
  \frac{\partial g_i}{\partial y_j}
  \]
  is a full matrix

- d02nb uses *forward communication*
  - user provides the function and Jacobian as external routines
  - d02nb calls them, and exits when the algorithm terminates

- cf reverse communication
  - NAG routine called in loop
    - along with user-supplied function
  - user checks for termination
Example problem (1)

- Integrate the *stiff Robertson problem*

\[
y_1' = -0.04y_1 + 10,000y_2y_3
\]
\[
y_2' = 0.04y_1 - 10,000y_2y_3 - 30,000,000y_2^2
\]
\[
y_3' = 30,000,000y_2^2
\]
Example problem (2)

- User must derive the Jacobian:

\[
\begin{bmatrix}
-0.04 & 10,000y_3 & 10,000y_2 \\
0.04 & -10,000y_3 - 60,000,000y_2 & -10,000y_2 \\
0 & 60,000,000y_2 & 0
\end{bmatrix}
\]
User provides routine for derivative vector

```fortran
SUBROUTINE FCN(NEQ, X, Y, YPRIME, IRES)
DOUBLE PRECISION X
INTEGER IRES, NEQ
DOUBLE PRECISION YPRIME(NEQ), Y(NEQ)
YPRIME(1) = -0.04D0*Y(1) + 1.0D4*Y(2)*Y(3)
YPRIME(2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) - 3.0D7*Y(2)*Y(2)
YPRIME(3) = 3.0D7*Y(2)*Y(2)
RETURN
END
```
User provides routine for Jacobian

```fortran
SUBROUTINE JAC(NEQ, X, Y, H, D, P)
DOUBLE PRECISION D, H, X
INTEGER NEQ
DOUBLE PRECISION P(NEQ,NEQ), Y(NEQ)
DOUBLE PRECISION HXD
HXD = H*D
P (1, 1) = 1.0D0 - HXD*(-0.04D0)
P (1, 2) = -HXD*(1.0D4*Y(3))
P (1, 3) = -HXD*(1.0D4*Y(2))
P (2, 1) = -HXD*(0.04D0)
P (2, 2) = 1.0D0 - HXD*(-1.0D4*Y(3)-6.0D7*Y(2))
P (2, 3) = -HXD*(-1.0D4*Y(2))
P (3, 1) = -HXD*(0.0E0)
P (3, 2) = -HXD*(6.0D7*Y(2))
P (3, 3) = 1.0D0 - HXD*(0.0D0)
RETURN
END
```
Using AD extensions

- Calculating Jacobian by hand is time-consuming
  - and error-prone (e.g. wavelet routine)
- Some NAG routines estimate derivatives
  - if user can’t supply them
  - finite differences used to estimate them
    - can be inaccurate
- Use AD to compute it from derivative vector
- Uses AD extensions to NAG Fortran compiler
New version of derivative vector routine

```fortran
SUBROUTINE FCN(NEQ, X, Y, YPRIME, IRES)
USE ACTIVE_MODULE
DOUBLE PRECISION X
INTEGER IRES, NEQ
DOUBLE PRECISION YPRIME(NEQ), Y(NEQ)
YPRIME(1) = -0.04D0*Y(1) + 1.0D4*Y(2)*Y(3)
YPRIME(2) = 0.04D0*Y(1) - 1.0D4*Y(2)*Y(3) - 3.0D7*Y(2)*Y(2)
YPRIME(3) = 3.0D7*Y(2)*Y(2)
RETURN
END
```
NEW VERSION OF JACOBIAN ROUTINE

SUBROUTINE JAC(NEQ, T, Y, H, D, P)
USE ACTIVE_MODULE
INTERFACE
  SUBROUTINE FCN(NEQ, X, Y, YPRIME, IRES)
    DOUBLE PRECISION X
    INTEGER IRES, NEQ
    DOUBLE PRECISION YPRIME(NEQ), Y(NEQ)
  END SUBROUTINE
END INTERFACE
DOUBLE PRECISION D, H, X
INTEGER NEQ, IRES, I
DOUBLE PRECISION P(NEQ,NEQ), Y(NEQ), YPRIME(NEQ)
TYPE(JACOB_TYPE) :: J
DOUBLE PRECISION HXD
...

Results Matter. Trust NAG
New version of Jacobian routine (contd)

...  
HXD = H*D
DIFFERENTIATE
  INDEPENDENT(Y)
  CALL FCN(NEQ,T,Y,YPRIME,IRES)
  J = JACOBIAN(YPRIME,Y)
END DIFFERENTIATE
CALL DERIVTOREAL(J,P)
P = -HXD*P
DO I=1,NEQ
  P(I,I) = P(I,I)+1
END DO
RETURN
END
Overview

- An introduction to NAG
- Numerical libraries
  - Library interfaces
    - The NAG Toolbox for MATLAB
  - New features
- Using AD with the NAG library
- The NAG Fortran compiler
  - Using AD with the NAG compiler
The NAG Fortran Compiler Release 5.2
Timeline

- 1991 – World’s first Fortran 90 Compiler (f90)
- 1997 – Fortran 95 (f95)
- 1999 – TR 15580 and 15581 added
  - IEEE modules
  - Allocatable attribute extensions
- 2003 – First new F2003 features
- 2008 – Release 5.2, most of F2003 (nagfor)
Key Features

- Standards Conformance
- Very few language extensions
- Extensive error checking
  - As required by the ISO standard
  - Checking for likely programming mistakes
  - Additional run time checking – `-C=undefined`, `-C=array`
- Portable
Portability

- Compiler converts internal representation to C
- Output C
- Use native C compiler as code generator
  - Available on major platforms
  - Allows “one-off” implementations, e.g. IBM z9/Linux
Fortran 2003

• How much is “most”?
Fortran 2003 – Features not yet implemented

- Ad hoc type comparison (EXTENDS_TYPE_OF & SAME_TYPE_AS)
- Parameterised derived types
- Finalisation
- Defined I/O
- Structure constructor syntax enhancements
Fortran 2003 Features implemented in 5.2

- Unlimited polymorphic
- Procedure pointers
- Object-bound procedures
- Allocatable scalars
- Deferred character length
- More intrinsic functions in initialisation expressions
- Reallocating assignment
- Recursive I/O
- ASSOCIATE
- MOVE_ALLOC
- New KIND= optional argument to some intrinsics
- CHARACTER argument to some intrinsics
- Type-spec for array constructor
- Asynchronous I/O
- Enhanced complex constants
- Pointer lower bound setting
- Renaming operators on USE
- _F_PROCPOINTER
- Changes to SYSTEM_CLOCK

- BOZ constants allowed in CMPLX, DBLE, INT and REAL
- C Interoperability
- Enum types
- Type bound procedures
- New I/O features
- I/O of NaNs
- Abstract derived types
- Deferred bindings
- PROCEDURE statement
- Public entities of PRIVATE Type
- ISO_FORTRAN_ENV module
- IMPORT statement
- INTENT for pointers
- Square bracket array constructors
- SOURCE in ALLOCATE
- GET_COMMAND etc
- GET_ENVIRONMENT_VARIABLE
- ...

Results Matter. Trust NAG
Mathematics and technology for optimized performance
Summary of F2003 Features

- All of the object-oriented features (except finalisers)
- All of C interoperability
- All the main new intrinsics
- Most of the new I/O features
Also in Release 5.2

- Double-double quadruple precision on all platforms that don’t have native quad precision.
  - Sun SPARC – native
  - Linux, Windows & Mac – double-double
- 31 decimal digits precision
- Slightly smaller exponent range than double
What’s next?

- **Fortran 2008?**
  - Big new addition to the language
  - First new features in Release 5.3, hopefully next year
- **OpenMP**
  - We’d like to introduce some OpenMP support…
- **Improved checking**
- **Improved efficiency**
- **Better debugger**
AD-enabled Fortran Compiler

- COMPAD – Collaboration between
  - RWTH Aachen
  - University of Hertfordshire
  - NAG
- Started collaboration in 2000
- COMPAD–I funded by ESPRC in 2002
- Now into the 3rd funded project COMPAD–III.
The NAG Fortran Compiler was designed from its inception as –

- Modular
- Extensible
- With libraries of “access” functions for operations on the internal representation of the program –
  - Parse Tree
  - Symbol Table

The compiler is an ideal base for building software tools.
NAG Fortran Compiler

AD software tool

Lex → Parse → Semantic Analysis → Code Generation → Object Code Generation

Lex → Parse → Semantic Analysis → Tool → Source Code Generation
AD-Enabled Fortran Compiler

- The main COMPAD team is
  - Uwe Naumann
  - Dmitri Gendler
  - Jan Riehme
  - Bruce Christianson

- COMPAD uses a hybrid approach
  - Overloading
  - Transformation of the IR

- The AD-enabled Compiler is being beta tested in a number of projects.

- The COMPAD team is advising these projects.
AD-enabled Compiler

- **Mode A**
  - Automatic type changes and overloading

- **Mode B**
  - Source transformation

- **Mode C**
  - new COMPAD-III development, hybrid
COMPAD-III Applications

- Project partners
  - QinetiQ Euler code
  - Structural dynamics solver (uses Complex datatype)

- Other projects
  - Max-Planck-Institute for Meteorology
  - German Federal Waterways Engineering and Research Institute.
  - NECTEC (Thailand) Oceanography

http://www.stce.rwth-aachen.de/
Summary

- **Products**
  - Mathematical, statistical, data analysis components
    - NAG Numerical libraries
      - Fortran
        - AD
      - C
      - Interfaces, e.g. MATLAB
  
- **NAG Fortran Compiler**
  - AD-enabled Compiler
  - COMPAD-III