Physics and the HECToR
HPC Service / 
NAG product update

Craig Lucas
The Numerical Algorithms Group Ltd
craig.lucas@nag.co.uk
Contents

• HECToR
• The CSE Service
• Physics on HECToR
• NAG Product Update
So what is HECToR?

- Latest high-end academic computing service for UK
  - After CSAR (1996–2006) and HPCx (2002-2010)
  - High End Computing Terascale Resource
  - Managed by EPSRC on behalf of RC-UK
  - Funded by EPSRC, NERC & BBSRC
  - Will run from 2007-2013
HECToR

• Objective:
  – To provide a service to the academic community enabling it to do true capability science

• Partners:
  – Service Provision: UoE HPCx Ltd (EPCC)
    • hardware hosting and maintenance
    • user services, helpdesk, etc
  – Hardware: Cray Inc
  – Computational Science & Engineering Support: NAG Ltd
Hardware Solution

• Phase 1 (16 October 2007 – 2009)
  – Cray XT4 (63 Tflops) – 5664 dual core Opterons
  – Cray X2 Vector Processor (2.9 Tflops)

• Phase 2a (June 2009)
  – Cray XT4 (210 Tflops) – 5664 quad core Opterons

• Phase 2b (June – December 2010)
  – New “Gemini” interconnect for XT allow true async comms
  – 12 core processors
  – 2 socket SMP node

• Phase 3 (2011? – late 2013)
  – Contract yet to be awarded
Phase 2a: Cray XT4

- 5,664 compute nodes, i.e. 22,656 cores
- One 2.3GHz quad-core Opteron per node
- 8 GB memory per node, 2 GB per core
- Peak performance of 210 TFlops
  - 22,656 cores * 9.2 GFlops per core, double precision
  - SSE (Streaming SIMD Extensions) instructions
  - 128 bit registers (4 single precision or 2 dp numbers)
  - 2 floating point units, operating on whole registers
  - 2.3 GHz * 2 dp words * 2 units = 9.2 GFlops
Phase 2a: Quad-core nodes

- Single core (incl. 64KB L1)
- L2 Cache 512KB
- L3 Cache 4MB
- Quad-core die
- AMD Barcelona processor
- Hyper Transport port
- 16-bit Hyper Transport 1 link, 6.4GB/s
- 8GB DDR Memory @ 800 MHz, 12.3GB/s across node
- Memory channel
Phase 2b: Cray XT6

- 44,544 cores from 1856 nodes, 2 x 12, 2.1GHz cores.
- 32 GB memory per node, 1.33 GB per core
- Peak performance of around 340 Tflops
- Initially interconnect remains at SeaStar, then upgraded to Gemini by the end of 2010.
- Half of XT4 will remain until this time.
- Gemini will allow one-sided communication in MPI, UPC and Co-Array Fortran.
Single core (incl. 64KB L1)
L2 Cache 512KB
L3 Cache 6MB (~1MB used by HT Assist)
Hex-core die
G34 socket – Magny-Cours Opteron

Hyper Transport port
16-bit Hyper Transport 1 link, 6.4GB/s
16-bit Hyper Transport 3.1 link, 25.6GB/s
8-bit Hyper Transport 3.1 link, 12.8GB/s
8GB DDR3 Memory @ 1333 MHz, 85.3GB/s across node
Memory channel
Performance at Phase 2

• Code must vectorize to benefit from the increased (2 dp numbers) SSE registers.
• There is increased contention for the communication network. So users looking at mixed mode or System V for MPI all-to-alls etc.
• Memory per core decreasing, maybe no longer enough for an MPI process on each core.
• Performance likely to be best as 4 6-way SMP, or under populated nodes.
Storage

• Shared by both machines
• 70 TB NAS storage (/home)
  – Backed up
  – Not accessible from the compute nodes
• 864 TB Lustre (/work)
  – High performance parallel filesystem
  – Not backed up
• Moving to eLustre.
• Archive facility (finally!)
XT Software Environment

• OS:
  – UNICOS/lc
    • SuSE Linux on login nodes
    • Compute Node Linux (CNL) on compute nodes

• Compilers:
  – PGI, Pathscale, GNU (Fortran 95, C, C++)
  – NAGware f95

• Tools:
  – CrayPat + Apprentice2 for performance profiling
  – Totalview debugger
XT Software Environment

• Libraries
  – Cray MPT (based on MPICH-2. Includes MPI2 and Cray SHMEM)
  – xt-libsci (BLAS, LAPACK, ScaLAPACK, SuperLU, IRT)
  – FFTW
  – Older Cray FFT interfaces
  – AMD ACML (BLAS, LAPACK, FFTs, RNGs)
  – NAG Fortran, SMP and Parallel Libraries (PGI only)

• Batch scheduler - PBS Pro
Third Party Application Codes

• Chemistry and Life Sciences
  – AMBER, CASINO, CASTEP, CHARMm, CPMD, CRYSTAL, DALTON, DL_POLY, GAMESS_UK, LAMMPS, NAMD, NWChem, SIESTA, VASP

• Engineering
  – ParaFEM, Pnewt, ROTORMBMGP

• Other
  – NEMO, HDF5, NetCDF, PetSc, AIMPRO, GS, H2MOL, HELIUM, PCHAN, POLCOMS, PRMAT, Globus, R
Getting Time On HECToR

• Precise mechanism varies between research councils
  – EPSRC and BBSRC: Apply direct to RC
  – NERC via 4 consortia or direct
  – See individual website for more information
  – Resources awarded in terms of Allocation Units. 1 Gflop for 1 hour (which have a notional value)
Getting Time On HECToR

• Full Peer Reviewed Access (Class 1a)
  – Typically peer-reviewed and part of larger research proposal
  – Application form requires you to provide info on previous HPC experience, type of jobs to be run, software needed, support requirements, etc
  – Requires Technical Assessment from NAG as part of full proposal

• See www.hector.ac.uk for more details
Getting Time On HECToR

• Direct Access (Class 1b)
  – New pilot scheme designed to give quicker access to a large (>125,000 core hours) resource
  – Valid for 6 months, could be used for:
    • bridging access between grant applications
    • trialing application developments at scale
    • providing preliminary results to aid grant applications
  – Independent panel assesses application every 4 months, next deadline 21st September.
  – Requires Technical Assessment
Getting Time On HECToR

• Pump Priming/New Research (Class 2a)
  – For pump-priming projects or new users of national service
  – Requires Technical Assessment from NAG
  – Requires short (1 page) outline of project
  – Limited resources (up tp 25,000 core hours))

• Distributed CSE (Class 2b)
  – Awarded by NAG, via independent panel.
  – Can include up to 50,000 core hours for new users
Technical Assessments

• Confirm whether project is suitable for HECToR
  – capability science
  – not practical on local resources
  – all software is available/budgeted for
  – expectations from service are reasonable
  – HPC aspects of proposal are plausible

• Gather information about use of service
  – job profile
  – software requirements
  – CSE requirements
CSE SERVICES
Overview of the CSE Service

• Partnership with HECToR user community to assist in deriving maximum benefit from the hardware

• Central Team
  – ~8 FTEs based in Oxford and Manchester

• Distributed Team
  – ~12 FTEs seconded to particular users, research groups or consortia
The Central Team

• Technical Assessments of applications
• Helpdesk
  – Part of single HECToR helpdesk
  – Available to deal with problems that may take several days to resolve
• Documentation
• Help with porting, parallelizing and code optimisation
  – Could be several weeks of effort
• Training
• Manage Distributed CSE Support
Training

• Some HECToR-specific
  – Introduction to HECToR
  – Debugging, Profiling and Optimising (2 days)
  – X2 Programming (2 days)
  – Multicore (2 days)

• Application Specific Courses
  – DL_Poly
  – CASTEP
Training

• More General Courses
  – Parallel Programming with MPI (3 days)
  – OpenMP (2 days)
  – Fortran 95 (3 days)
  – Others covering IO, Core HPC Algorithms, Visualization, Portability, Testing etc

• Open to user requests
  – Training for specific projects
  – Training on specific application codes
Training

• The current schedule of courses is available at http://www.hector.ac.uk/cse/training/

• A full list of the courses we offer can be found at http://www.hector.ac.uk/cse/training/courselist/

• All of these courses are free to HECToR users and to anyone whose work comes under the remit of EPSRC, NERC or BBSRC.

• So you don’t need to be currently funded.
Distributed CSE Support

- Allocated to specific individuals or groups
- Awards for software development to improve the capability of codes on HECToR
- Usually at least 6 months of effort for
  - Porting
  - Tuning, optimisation, scaling
  - Functional enhancements, etc.
Distributed CSE Support

• Regular calls for proposals
  – Next deadline is 21\textsuperscript{st} June
  – Open to any HECToR user funded by a sponsoring Research Council
  – Also available for potential HECToR users

• “Panel-style” evaluation process
  – Independent experts rank proposals
  – NAG negotiates agreements according to available resources
  – Whole process is open and transparent
Distributed CSE Support

• Staff managed by, and part of, CSE team
  – Programme of work/targets agreed in advance with PI
  – Managed/co-ordinated by member of central team
  – Could be based in PI’s institution, at NAG, or elsewhere
  – Could be employed directly by NAG or by host institution via contract

• More information
  http://www.hector.ac.uk/cse/distributedcse
Where to find more information

- Main HECToR website:
  - www.hector.ac.uk

- CSE pages at:
  - www.hector.ac.uk/cse

- EPSRC
  - www.epsrc.ac.uk/ResearchFunding/FacilitiesAndServices/HighPerformanceComputing/

- NERC
  - www.nerc.ac.uk/research/sites/facilities/hpc/

- BBSRC
  - www.bbsrc.ac.uk/funding/hpc_access.pdf
Where to find more information

• More information on hardware, software and operation.
  – docs.cray.com In depth reference materials.

• Stay in touch, send subject of:
  SUBSCRIBE HECTOR-INTERESTED Forename Surname
  To: LISTSERV@jiscmail.ac.uk
HECToR Summary

• Cray XT4 and X2 vector machines in service
• Cray XT6 coming very soon. c/w programming challenges!
• Several routes for getting HECToR time.
• CSE Service available to help users get the most out of HECToR.
• Distributed CSE awards available for software development.
• Training available to all.
PHYSICS ON HECTOR
Physics Codes on HECToR

• **Electronic Structure**
• These codes provide a periodic description of the electronic structure of the system and are often used for studying condensed-phase systems.
  
  – cp2k
  – **CASTEP**
  – CPMD
  – Siesta
  – CRYSTAL
  – VASP
  – ONETEP
  – Wein-2k
Physics Codes on HECToR

- Classical Molecular Simulation
- These codes use an empirically derived 'force-field' to describe the interaction between particles and can often treat much larger systems than the electronic structure codes.
  - Amber
  - DL_POLY
  - Gromacs
  - CHARMM
  - LAMMPS
  - NAMD
  - ChemShell
Physics Codes on HECToR

• Plasma Physics
• Codes used for studying the properties of high-energy plasmas.
  – CENTORI
  – GS2
  – H2MOL
  – HELIUM

• Top three codes for usage on HECToR are all Physics codes, and VASP accounts for 22%!
Case Study

• We look at two codes in a little more detail:

• Helium
• CASTEP
HELIUM

- HELIUM models the interaction between a single Helium atom and an intense, short laser pulse.
- The code is used to study the interaction between the two electrons as they ionize.
- To solve this problem HELIUM directly solves the full Time-Dependent Schroedinger Equation, with a linearly-polarised laser field, this is a time-dependent PDE with 5 spatial dimensions.
- No simplifying assumptions made in the physics.
HELium

- One of the HECToR benchmarks.
- Fortran chosen as there are a lot of 3D and 4D arrays, and the need for array operations.

- Implements the 5 spatial dimensions as a 2-D finite-difference grid for the two radial co-ordinates, and a basis set of coupled spherical harmonics for the three angular co-ordinates.
HELIUM

• Has a 3-D array of ~2000-3000 2-D radial grids (or "partial waves"), each of size ~5000x5000 grid points, so ~10-100 billion grid points in total.

• At timestep t we update each 2-D radial grid based upon the values at timestep t-1 of that 2-D radial grid and some number of other 2-D radial grids.
HELUM

- Parallelised with MPI over the 2-D radial grids.
- So each MPI task will have a square block of every 2-D radial grid.
- Communications are mostly nearest-neighbour halo exchange, plus some global sums.
- The code has been tested on over 70,000 cores on Jaguar (Oak Ridge ~2PFlop system)
- Regularly run on 8,000 - 16,000 cores on HECToR.
HELUM

• Currently a dCSE is to look at hybrid parallelism to OpenMP parallelise over the basis set of partial waves within each MPI task.
• There is a limit to how small we can make each square block on each MPI task. Communication worsens for smaller blocks per MPI task.
• Proposed crossed-fields (two different laser field at 90 degrees to each other) will make this a full 6-D calculation increasing basis set in memory.
HELUM

• Thus a hybrid MPI-OpenMP approach could allow scaling to larger numbers of cores in total by paralleling the work currently done by an MPI process.
• Help to accommodate increases in the size of the basis set in memory.
• And perhaps improve efficiency by reducing the MPI communication overhead.
CASTEP

• CASTEP is a software package which uses density functional theory to provide atomic-level description of materials and molecules.
• It can provide information about total energies, forces and stresses on an atomic system, as well as calculating optimum geometries, band structures, optical spectra, phonon spectra etc.
• It can also perform molecular dynamics simulations.
CASTEP

• CASTEP began in 1999, with the first release in 2001.
• There was a written specification, prior to coding.
• Implemented in Fortran 90 with TR 15581 (allocatable dummy arguments and derived type components.)
• Modular approach
• Data abstraction using derived data types
• Overloading for simple, clear subroutine names
CASTEP

- CASTEP consists of three coding levels.
- “Utility” routines – these provide the core algorithms to CASTEP along with FFTs, communication routines (built on MPI) and the IO.
- “Fundamental” routines – the building blocks with definitions of types, density, potential, wave function etc,
- “Functional” routines – the physics.
CASTEP

• This approach allows for functionality to be express in physics notation at the functional level.
• Easier to add new functionality.
• All machine dependent code at the lowest utility level.
• This allows all the work required for a move to HECToR Phase 2b, to be done at the utility level.
CASTEP

- CASTEP links to appropriate vendor libraries for BLAS, LAPACK and FTTs.
- Most of the computational time is spent in these routines.
- Also 3D FFT responsible for much of the communication.
- For each dimension x, y, z, each process performs a subset of 1D transforms in that direction.
- Before moving from one dimension to the next data must be “transposed” so that each process has the data it needs. This requires a call to MPI_Alltoallv.
CASTEP

• Two optimization techniques are being used that result in only one MPI message being sent per node.
• Option 1 - Use MPI_Gather to marshal the outgoing data and MPI_Scatter to distribute the incoming data.
  – Synchronous
• Option 2 - System V shared memory segments - allow processes on a node to collate data prior to an MPI message.
  – One segment that all processes in a node can write their outgoing data directly into.
  – One that all processes in a node can read incoming data from.
  – Asynchronous
Options on XT4

<table>
<thead>
<tr>
<th>Nprocs</th>
<th>Default</th>
<th>Option 1</th>
<th>Option 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>2261</td>
<td>3103</td>
<td>2665</td>
</tr>
<tr>
<td>128</td>
<td>1788</td>
<td>1784</td>
<td>1572</td>
</tr>
<tr>
<td>256</td>
<td>1564</td>
<td>1191</td>
<td>1087</td>
</tr>
</tbody>
</table>

- Time in seconds
- Both options show a speed up, but only on larger jobs, where there is a lot of communication.
- Option 2 is best.
- This should be really important on Phase 2b...
## System V on XT6

### Table: System V on XT6 Performance Comparison

<table>
<thead>
<tr>
<th>#cores per node</th>
<th>Default CASTEP (Initial)</th>
<th>Option 2 - System V (Initial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4139 (78.7)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4820 (85.6)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9775 (134.8)</td>
<td>2646 (81.8)</td>
</tr>
<tr>
<td>8</td>
<td>21957 (258.1)</td>
<td>2515 (62.4)</td>
</tr>
<tr>
<td>12</td>
<td>36605 (407.7)</td>
<td>6177 (82.8)</td>
</tr>
<tr>
<td>24</td>
<td>? (869)</td>
<td>? (700.3)</td>
</tr>
</tbody>
</table>

- Using sparsely populated jobs (i.e. using fewer MPI processes per node than number of cores per node)
- Shows improved performance. (Less contention on resources, memory and interconnect)
- Using the executable with System V gives up to 8X speed up.
CASTEP

- A third approach is the use of shared memory libraries – hybrid MPI/OpenMP code via use of threaded (OpenMP) BLAS and LAPACK.

- First we compare just using all and one core per node on the XT4:

<table>
<thead>
<tr>
<th>al3x3 Benchmark</th>
<th>MPI Processes</th>
<th>Packed</th>
<th>1 core / node</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
<td>2483</td>
<td>1933</td>
<td>1.3X</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>1573</td>
<td>1103</td>
<td>1.4X</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1258</td>
<td>761</td>
<td>1.7X</td>
</tr>
</tbody>
</table>

But 4X AU cost!
CASTEP

• When using only one core per node we have three idle cores.
• So with threaded libraries:

<table>
<thead>
<tr>
<th>al3x3 Benchmark</th>
<th>MPI Processes</th>
<th>Packed</th>
<th>1 core / node</th>
<th>Speedup</th>
<th>Threaded</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
<td>2483</td>
<td>1933</td>
<td>1.3X</td>
<td>1688</td>
<td>1.5X</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>1573</td>
<td>1103</td>
<td>1.4X</td>
<td>974</td>
<td>1.6X</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1258</td>
<td>761</td>
<td>1.7X</td>
<td>676</td>
<td>1.9X</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>1895</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Good, but still not 4X
CASTEP

• We are still using 4 times the resources!
• We therefore need a reason to do it.
  • There are limits to scalability. E.g. al3x3 case using 1024 processes takes 1895s. Using the same number of cores the 256 process, 4-thread case takes 676s
  • Large problem that exceeds the memory-per-core limit (currently 2GB, soon 1.3GB). Idle cores, so make them do something. (This is considered critical and being investigated now.)
• You want the result as soon as possible.
References

• CSE/dCSE reports:
  – www.hector.ac.uk/cse/reports

• Shared memory report:
  – www.hector.ac.uk/cse/reports/castep_m.pdf

• (Draft) Phase 2b guide:
  – www.hector.ac.uk/cse/documentation/Phase2b
Contents

- The NAG Compiler
- The NAG Libraries
The NAG Fortran Compiler

- 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)
- 2010 – Release 5.3 later this year
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 – not yet implemented

- Ad hoc type comparison (EXTENDS_TYPE_OF & SAME_TYPE_AS)
- Parameterised derived types
- Finalisation (priority for 5.3)
- 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
- C_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
- ...
Summary of F2003 Features

- All of the object-oriented features (except finalisers which will be in 5.3)
- 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, later this year
- **OpenMP**
  - We’d like to introduce some OpenMP support, in a future release.
- **Improved checking**
- **Improved efficiency**
- **Better debugger**
Strengths

- Great pedigree – team headed by NAG principal consultant, Malcolm Cohen, secretary to the international working group on Fortran, ISO/IEC JTC1/SC22/WG5.
- Co-author of "Fortran 95/2003 Explained" with John Reid and Michael Metcalf.
Strengths

- World’s first Fortran 90 compiler
- Developed and enhanced to include Fortran 95 and most features of Fortran 2003.
- Regularly updated, fully supported.
- EXCELLENT (the world’s best) checking compiler
Fortran Builder

- Integrated Development Environment for NAG compiler on Windows PC
- Perfectly integrated with NAG Library
  - NAG example program templates
- Extra facilities: tools e.g. Fortran Polisher, Fortran converter, LAPACK examples
- Integrated debugger
program main
    integer, parameter :: dp = selected_real_kind(15)
    real(dp) :: x, y
    read *, x
    y = SIN(x)
    print *, y
end program main
Release 5.2 Availability

- x86 Linux
- x64 Linux
- SPARC
- Mac
  - Intel
  - PowerPC
- Windows (Fortran Builder)
The NAG Numerical Libraries

- NAG Fortran Library
- NAG C Library
- NAG SMP Library
  - for symmetric multi-processor machines (OpenMP)
- NAG Parallel Library
  - for distributed memory parallel machines (MPI)
- NAG Toolbox for MATLAB

- Documented with error & accuracy information and example programs.
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 and Regression Analysis
- Multivariate Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research
New at Mark 22

- Global Optimization
- Nearest Correlation Matrix
- Wavelets
- Roots of equations
- Ordinary Differential Equations Solvers
- Various linear algebra
- Various statistics, including random number generators, multivariate methods and time series analysis
- Option pricing
- Sorting and searching
NAG Library Interfaces

- Fortran
- C
- C++
- C# / .NET
- Java
- Borland Delphi
- Python
- Excel
- MATLAB
- Maple
- LabVIEW
- R and S-Plus
- SAS
- Simfit
- ...
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 finite point molecule form on a two-dimensional topologically-rectangular mesh. (Topological means that a polar grid, for example (r,θ), can be used, being equivalent to a rectangular box.)

2 Syntax

[d, itcoun, itmade, residu, chnum, ifail] = d03eb(n1, n2, a, b, q, t, apar, itcoun, itmade, residu, itnum, rnum, conres, conch, r2, t2)

3 Description

Given a set of simultaneous equations

\[ M \mathbf{u} = \mathbf{g} \]

(where \( M \) 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 \( \mathbf{u} \) which is a known vector of length \( n_1 \times n_2 \) and \( M \) is a square \( (n_1 \times n_2) \) by \( (n_1 \times n_2) \) matrix.

The equations must be of five-diagonal form:

\[ a_{i1}u_{i-1} + a_{i2}u_i + a_{i3}u_{i+1} + a_{i4}u_{i-1} + a_{i5}u_{i+1} = g_i \]

for \( i = 1, 2, \ldots, n_1 \times n_2 \), provided \( c_i \neq 0 \). Indeed, if \( c_i = 0 \), then the equation is assumed to be

\[ u_i = 0. \]

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

\[
\begin{bmatrix}
   \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} \\
   \mathbf{a}_{12} & \mathbf{a}_{22} & \mathbf{a}_{23} \\
   \mathbf{a}_{23} & \mathbf{a}_{24} & \mathbf{a}_{25}
\end{bmatrix}
\begin{bmatrix}
   u_1 \\
   u_2 \\
   u_3
\end{bmatrix}
= 
\begin{bmatrix}
   \mathbf{g}_{11} \\
   \mathbf{g}_{22} \\
   \mathbf{g}_{33}
\end{bmatrix}
\]
NAG Toolbox: c05ax

1 Purpose
c05ax attempts to locate a zero of a continuous function using a continuation method based on a scalar iteration. It uses reverse communication for evaluating the function.

2 Syntax
[x, c, ind, ifail] = c05ax(x, fx, tol, ir, c, ind, 'scal', 'scal')

3 Description
c05ax uses a modified version of an algorithm to locate a zero of a continuous function. The algorithm is a sequence of problems.

4 References

5 Parameters

How to call the NAG routine in MATLAB

Calling the routine in MATLAB

MATLAB plot
Mark 23

- Code freeze imminent for Mark 23 of NAG Fortran Library and NAG Toolbox for MATLAB will be out soon after.
  - Global Optimization
  - Image processing
  - Dense Linear Algebra
  - Sparse Linear Algebra
  - Correlation and Regression Analysis
  - Random Number Generators
  - Nonparametric Statistics
  - FFTs
  - ODE
  - Integration
  - Roots of Equations
  - Option Pricing
  - Wavelets
  - Special functions
NAG Library for SMP and Multicore

- NAG Library for SMP and Multicore will follow much quicker from Mark 23.
- The same interface to serial library, just re-link

- NAG-specific routines parallelised with OpenMP
  - Focus of future NAG SMP library development work
  - Seeking to broaden scope of parallelism to different parts of the library
NAG Library for SMP and Multicore

- Root Finding
- Summation of Series (e.g. FFT)
- 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 and Regression Analysis
- Multivariate Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research
Thanks to ...

- Keith Refson, Rutherford Appleton
- Stuart Clarke, Durham

- NAG colleagues
  - Lucian Anton
  - Chris Armstrong
  - Ian Bush
  - Ian Hounam
  - Phil Ridley
  - Ed Smyth