Science & Technology
Facilities Council

The HSL Mathematical Software Library: Past, present and future

Jonathan Hogg
jonathan.hogg@stfc.ac.uk

Numerical Analysis Group
Computational Science and Engineering Department
STFC Rutherford Appleton Laboratory

7 October 2011

Overview of talk

- Brief introduction to who we are
- What does HSL do?
- Software issues
- Our approach to multicore and manycore

Numerical Analysis Group at RAL

- Part of Computational Science and Engineering Department (CSED) of the Science and Technology Facilities Council.
- STFC is an independent, non-departmental public body of the Department for Business, Innovation and Skills
  - funds researchers in universities directly through grants
  - provides access to world-class UK facilities (eg ISIS, Central Laser Facility ...)
  - provides in the UK a broad range of scientific and technical expertise
  - provides access to world-class facilities overseas (eg CERN, ESA, ESRF ...)

Numerical Analysis Group at RAL

- CSED provides world-class expertise and support for UK theoretical and computational science communities, in both academia and industry.
- The Group is based at Rutherford Appleton Laboratory in Oxfordshire (about 15 miles south of Oxford).
- RAL has around 1200 employees (mainly scientists and engineers) plus large number of visitors.
- Group is funded by an ESPRC grant and since late 1950s has been one of the leading UK numerical analysis groups.

Numerical Analysis Group at RAL

Jennifer Scott	(Group leader): Sparse linear systems and sparse eigenvalue problems, high-performance computing.
Mario Arioli	Numerical linear algebra, numerical solution of PDEs, error analysis.
Iain Duff	Sparse matrices and high-performance computing.
Nick Gould	Large-scale optimization, nonlinear equations and inequalities.
Jonathan Hogg	Parallel linear algebra, optimization.
John Reid	Sparse matrices, automatic differentiation, and Fortran. (Honourary scientist)
Sue Thorne	Preconditioners, saddle-point problems and sparse linear systems.

Science & Technology Facilities Council

Numerical Analysis Group at RAL

Main areas of expertise

- Sparse linear algebra
- Large-scale optimization
- High-quality software

A short history of HSL

1963	Harwell Subroutine Library, Harwell Laboratory
1964	First distributed externally
1966	First library catalog released
1990	Group moved to Rutherford Appleton Laboratory
1990	Converted remaining codes to Fortran 77
2000s	Modern packages written in Fortran 90/95/2003
2010s	Fortran for manycore?

Science & Technology Facilities Council

Areas of expertise

HSL focused around large-scale matrix computation:

Linear system solvers $Ax = b$ for symmetric, unsymmetric, complex, real, direct, iterative...

Eigensolvers $Ax = \lambda Bx$ generalised eigenvalue problem, can find selected eigenvalues.

HSL has international reputation for reliability and efficiency. It is used worldwide by academics and commercial organisations and has been incorporated into large number of commercial products.

Science & Technology Facilities Council

What is a sparse matrix?

Any matrix where exploiting known zeros is beneficial.

- Better accuracy
- Better speed
- Bigger problems

Many large-scale real world problems have sparsity:

- Hard to specify otherwise
- Items in a large system rarely interact globally

Science & Technology Facilities Council

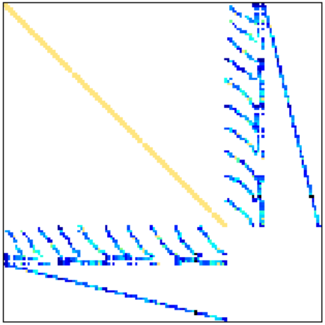
Applications

Oceanography	Oceans circulation,...
Meteorology	Climate change, weather forecast,...
Structural mechanics	Elasticity, Plasticity, ...
Chemical engineering	Large molecules simulation,
Hydrology	Underground water remediation, pollution,..
Fluid-dynamics	Stokes, Navier-Stokes, advection diffusion,...
Economics	Econometric, linear programming, ...
Physics	Hamiltonian problems, thermodynamics,...
Biosciences	Ecology, diseases control,...
Computer science	Data mining, Googling, networks, ...
...	

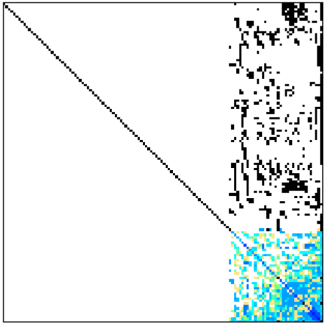
But all have different patterns and characteristics

Examples

Economics

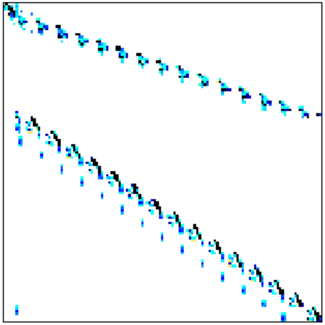


Electrical circuit simulation

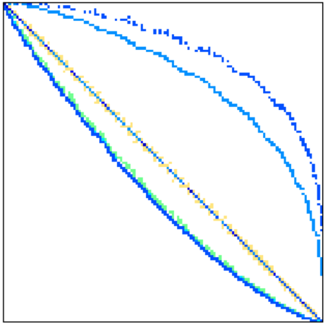


Examples

Chemical process simulation

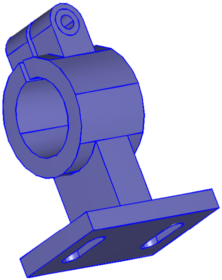


Network simulation



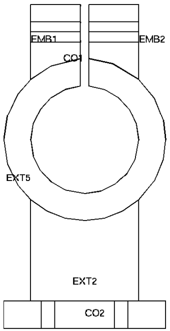
Science & Technology Facilities Council

Structural mechanics problem



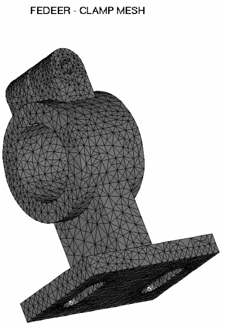
Elastic deformation of a clamp

Structural mechanics problem (cont.)



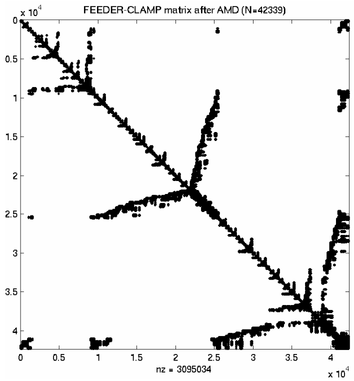
From the (CAD) design of the clamp

Structural mechanics problem (cont.)



To the mesh

Structural mechanics problem (cont.)



To the matrix A of size $n = 42339$, $nz(A) = 3095034$

Large sparse linear systems

When A is large and sparse there are two approaches to solving $Ax = b$:

Direct Factorize the matrix as $PAQ = LU$, P and Q permutation matrices, L lower triangular and U upper triangular then solve $Ly = Pb$ and $UQ^T x = y$. For example by Gaussian elimination.
Note: for dense systems, requires $O(n^2)$ storage and $O(n^3)$ flops

Iterative Use some iterative scheme to produce vectors x_k that converge to solution. Conjugate gradients is an example of this.

Why is it hard?

- We have to worry about the zero entries
- Need to use sparse data structures
- If we just go ahead and apply Gaussian elimination to sparse A, the zeros will, in general, rapidly fill-in.
- We have to order carefully eg

$$\begin{array}{ccccc} x & x & x & x & x \\ x & x & & & \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{array} \qquad \begin{array}{ccccc} x & & & & x \\ & x & & & x \\ & & x & & x \\ & & & x & x \\ x & x & x & x & x \end{array}$$

fills in totally

no fill-in

When to use a direct solver?

- Generally method of choice but memory is an issue
- Can be used a black-box solver
- Guaranteed accuracy
- Efficient if multiple solves required
- No good preconditioners available for iterative method

Note: (modified) direct methods are often used in combination with iterative methods eg incomplete factorization preconditioners.

Some HSL direct solvers

MA42	Unsymmetric finite problems; out-of-core options.
MA48	Unsymmetric problems.
HSL_MA57	Symmetric indefinite problems; positive definite problems with many solves.
HSL_MA77	Very large symmetric problems; out-of-core options.
HSL_MA87	Multicore solver for positive-definite systems.

Other direct solvers are available and some are very popular (notably MA27). All the above rely on the use of BLAS for performance.

Science & Technology Facilities Council

HSL iterative packages

- Main strength of HSL lies in its direct solvers but HSL includes efficient implementations of many of the standard iterative methods (CG, CGS, GMRES, BiCG, BiCGstab ...)
- These are all reverse communication packages (control is returned to the calling program for matrix-vector products and application of preconditioner).
- Some preconditioners are offered, the most recent being

  HSL_MI20 Algebraic multigrid preconditioner.

Generalised eigenvalue problem $Ax = \lambda Bx$

For large problems, only possible to compute selected eigenpairs.
Some HSL eigen routines:

EA16	Symmetric matrices, rational Lanczos, interior eigenvalues.
HSL_EA19	Symmetric matrices, subspace iteration, leftmost eigenvalues.

These are also reverse communication packages.

Other HSL routines

- Sparse matrix manipulation and file utilities
- Orderings (Reverse Cuthill McKee, Sloan algorithm for profile reduction, approximate minimum degree, block triangular form ...
- Scalings (`MC64` is widely used for preprocessing)
- Linear programming
- Optimization (also GALAHAD library developed by Group)
- ...

Organisation of HSL

- 2000: HSL divided into main HSL library and HSL Archive
- HSL Archive
  - older packages superseded either by improved HSL packages (eg `MA28` superseded by `MA48`) or by public domain libraries such as LAPACK
  - Free to all for non-commercial use but its use is not supported
- New release of HSL approx. every 3 years
- 2011: individual packages available without charge to individual academics worldwide for personal research and teaching.

`http://www.hsl.rl.ac.uk/`

Science & Technology Facilities Council

Organisation of HSL

- Packages classified into chapters. These were decided on in the early days (certainly by Release 1 of Catalogue 1971)
- Chapters led to HSL naming convention. eg `AD02` for automatic differentiation belongs to the 'A' chapter on computer algebra and `MA48` is part of 'MA' chapter of matrix linear algebra packages.
- Prefix `HSL_` used to indicate package written in Fortran 90/95/2003
- HSL catalogue provides complete list of packages in HSL 2011 and for each gives a brief outline of purpose, method, origin, language and other attributes.

Science & Technology Facilities Council

Programming Lanugages

We write in standard compliant Fortran:

- Older packages in Fortran 77
- More recent packages in Fortran 90/95
- Fortran 2003 features we use:
  - C interoperability
  - TR 15581 Allocatable Components

Science & Technology Facilities Council

Fortran 2003 features we'd like to use

Rule of thumb: Only use features supported $>$ 3 years.
Look for support by: gfortran, ifort, NAG, PGI, Oracle

Fortran 2003 features we'd like to use

Rule of thumb: Only use features supported $>$ 3 years.
Look for support by: gfortran, ifort, NAG, PGI, Oracle

- Optional `kind` argument to `size` and `int`.
  - Required for long integer support

Fortran 2003 features we'd like to use

Rule of thumb: Only use features supported > 3 years.
Look for support by: gfortran, ifort, NAG, PGI, Oracle

- Optional `kind` argument to `size` and `int`.
    - Required for long integer support
- The `ASSOCIATE` construct

```
do i = 1, n
   do j = 1, keep%nodes(i)%m
      keep%nodes(i)%val(j) = &
         keep%nodes(i)%val(j) + 1
   end do
end do
```

Science & Technology Facilities Council

Fortran 2003 features we'd like to use

Rule of thumb: Only use features supported > 3 years.

Look for support by: gfortran, ifort, NAG, PGI, Oracle

- Optional `kind` argument to `size` and `int`.
  - Required for long integer support
- The `ASSOCIATE` construct

```
do i = 1, n
   associate(this_val => keep%nodes(i)%val)
      do j = 1, keep%nodes(i)%m
         this_val(j) = this_val(j) + 1
      end do
   end associate
end do
```

Fortran 2003 features we'd like to use

Rule of thumb: Only use features supported > 3 years.
Look for support by: gfortran, ifort, NAG, PGI, Oracle

- Optional `kind` argument to `size` and `int`.
  - Required for long integer support
- The `ASSOCIATE` construct

```
do i = 1, n
   associate(this_val => keep%nodes(i)%val)
      do j = 1, keep%nodes(i)%m
         this_val(j) = this_val(j) + 1
      end do
   end associate
end do
```

- The `CONTIGUOUS` attribute (Fortran 2008) avoids need for packing pointers to arrays on subroutine calls.

Science & Technology Facilities Council

Testing

For each package:

- Have a comprehensive test deck that tries to ensure every line is executed.
- Run large number of real world problems (UFL Sparse matrix collection)
- Check against multiple compilers:
  NAG, gfortran, ifort, pgf95, Oracle, g95, lf95

Ease of use

Every package:

- Comprehensive documentation
- Includes a simple example users can build from
- Minimise arguments that must be set by user — have a control type with default values.
- Offer *optional* checks on user data
- Print meaningful error messages, not just obscure codes

Science & Technology Facilities Council

MATLAB interfaces: general

HSL and NAG have different philosophies here:

NAG	Try and match Fortran interface. Make prototyping easy.
HSL	Try and make a natural MATLAB interface. Make experimentation easy.

Example:

```
[iprmOut, nzlumxOut, il, lval, iu, uval, nnzl, nnzu, flop, ifail] =
  f11me(n, irowix, a, iprm, thresh, nzlmx, nzlumx, nzumx)
```

vs.

```
fact = ma86_factor(A)
```

MATLAB interfaces: technical

Fortran/MATLAB interfacing is difficult

- Fortran compiler version antique (gcc-4.3.x).
- Fortran allocate doesn't allow MATLAB control.
- Not type safe.
- Lots of segfaults to be had, suggests quite buggy.
- Complex data type handled strangely.
- `INTEGER*4` vs `INTEGER*8` headaches.
- Hard to debug.

Science & Technology Facilities Council

MATLAB interfacing solutions

Wrap Mathworks functions in typesafe fashion

e.g.

```
call matlab_to_fortran(mp, scalar, 'argname')
mp = fortran_to_matlab(array)
```

- Reduce type related segfaults
- Provide informative error messages
- Protect against API changes
- Accelerate interface development

Testing still a problem

Our approach to multicore and manycore

New codes currently using OpenMP

- Fortran/OpenMP 3.0: Compiler bug minefield

Our approach to multicore and manycore

New codes currently using OpenMP

- Fortran/OpenMP 3.0: Compiler bug minefield

What about bit-compatibility?

- Extra constraint $\Rightarrow$ reduces available parallelism.
- Dongarra: “Other sciences are allowed error bars!”
- Users: “Debugging is already hard!”, “How can I trust it?”

Our approach to multicore and manycore

New codes currently using OpenMP

- Fortran/OpenMP 3.0: Compiler bug minefield

What about bit-compatibility?

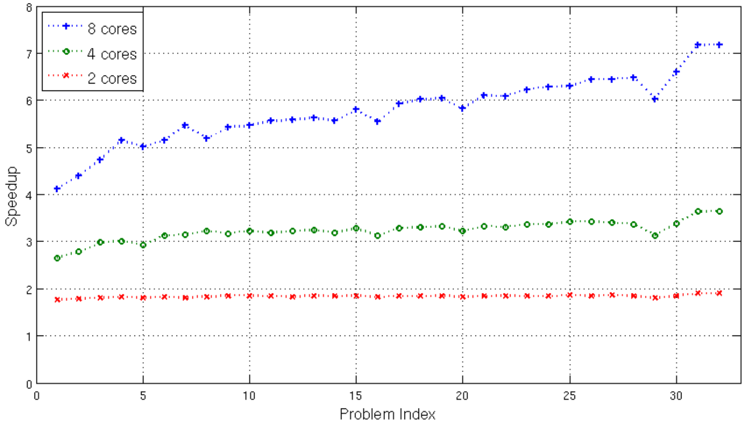
- Extra constraint ⇒ reduces available parallelism.
- Dongarra: “Other sciences are allowed error bars!”
- Users: “Debugging is already hard!”, “How can I trust it?”

Want to use some of the new frameworks:
QUARK, StarPU, Intel TBB

- Only available/reliable for C
- Make repeatability someone else’s problem!

Science & Technology Facilities Council

HSL_MA87 speedup



Mixed precision approach

Advantages of single precision include:

- Reduces amount of data moved within direct solver (memory bandwidth bottleneck).
- Uses less storage (potentially solve LARGER problems).
- On a number of modern architectures, currently more highly optimised.

BUT potential loss of accuracy. **Solution:** use **double precision**

iterative method preconditioned by **single precision factorization**. This is currently a hot topic, particularly with regard to multicore machines.

What do I want a task system to do?

I provide:

- Specify a series of operations...
- ...to get the result
- Perhaps *alternative* ways to get there
- Perhaps alternative implementations of tasks
    - Different parameters
    - Different architectures

Task system then:

- Builds dependency graph
- Builds on-the-fly performance models of tasks?
- Picks best method for its machine
- Schedules tasks efficiently

Science & Technology Facilities Council

Long term task facilities

- Runs it on a heterogeneous cloud/grid
- Can arbitrarily subdivide tasks at the “right” blocking level
- Mix tasks from different libraries

Summary: What we offer

- HSL is a fully supported and maintained library. We will:
  - Advise on which package(s) to use
  - Fix reported bugs

  But we are a research group and cannot hold users' hands.
- Collaboration. We have always worked with other researchers and a number of important recent packages were written in collaboration eg `HSL_EA19` (University of Westminster) and `HSL_MI20` (University of Manchester).
- HSL is freely available for academic research and teaching
- Some key packages now have C and MATLAB interfaces

Science & Technology
Facilities Council

The End

Our Questions for you

- Do you have any large test problems?
- What numerical software are you missing?
- Do you know anyone who might be interested in a GPU-related job?

http://www.hsl.rl.ac.uk