Science & Technology Facilities Council

HSL and Challenges in Sparse Direct Linear Solvers

Jonathan Hogg

STFC Rutherford Appleton Laboratory

Sparse Linear Algebra Solvers for High Performance Computing Workshop
8th July 2013

Numerical Analysis Group at RAL

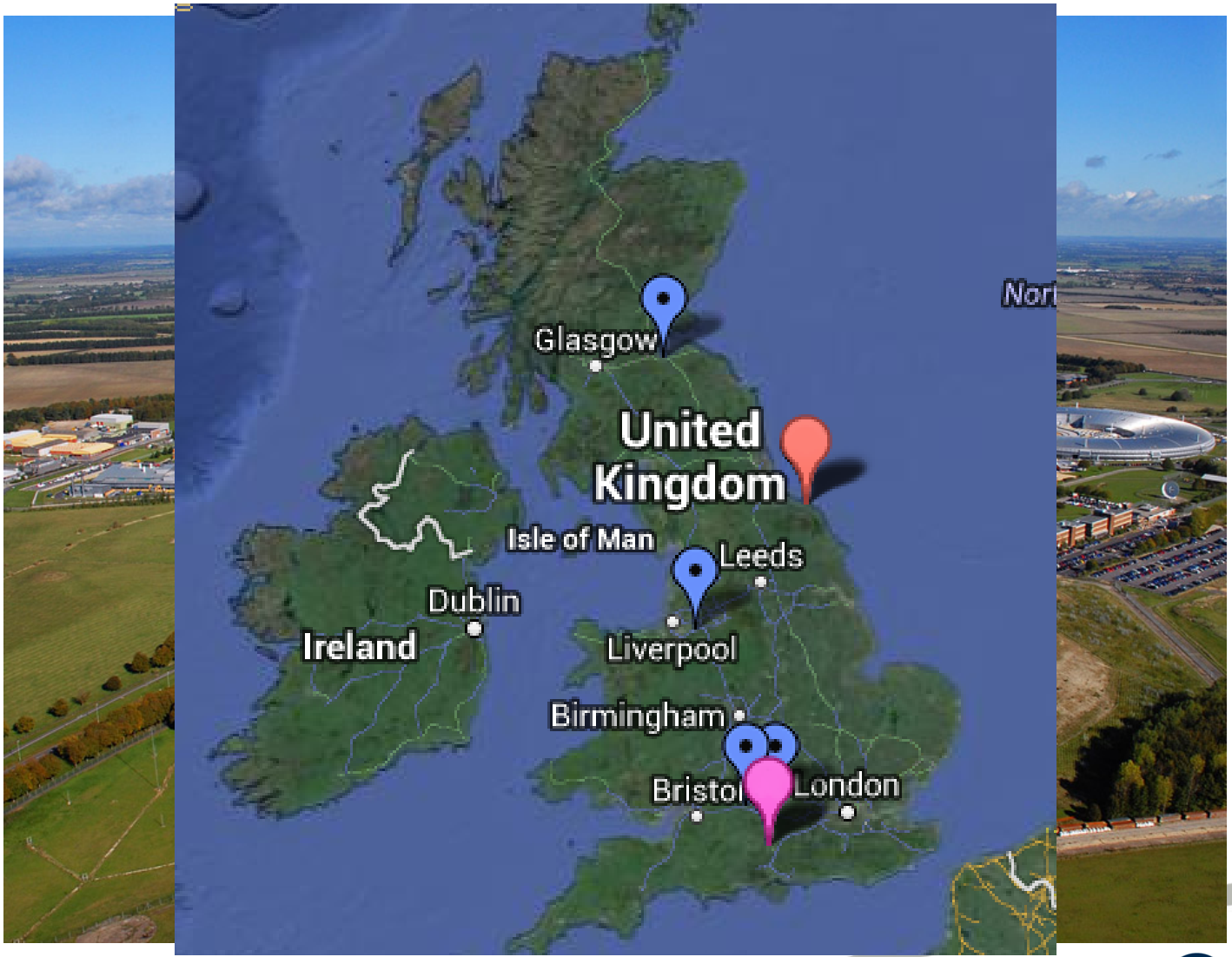
- Part of Scientific Computing Department (SCD) of the Science and Technology Facilities Council (STFC).
- STFC is a research council whose overall mission is
  - "To maximise the impact of our knowledge, skills, facilities and resources for the benefit of the United Kingdom and its people."
  - World-class research
  - World-class innovation
  - World-class skills

Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

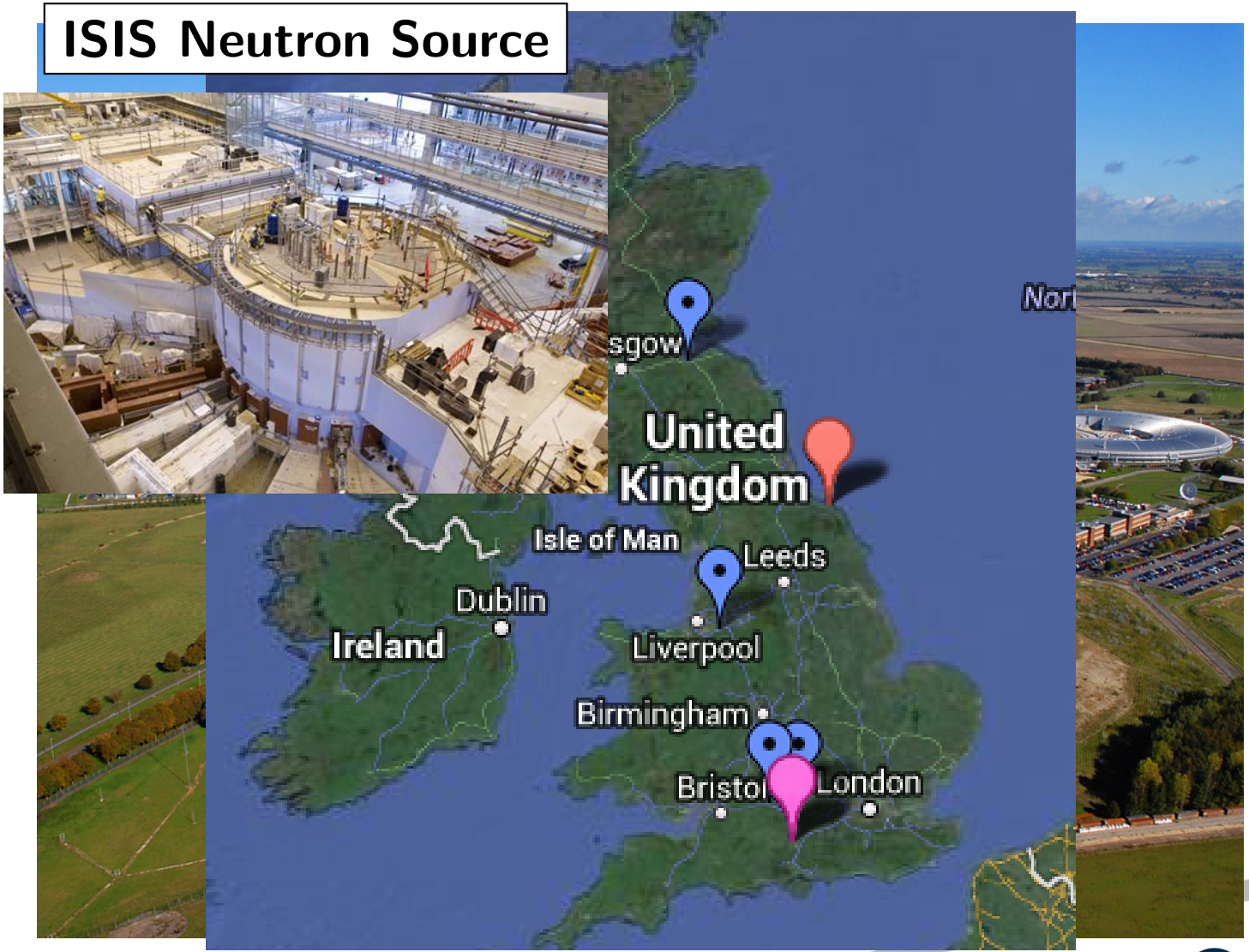
Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory



Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

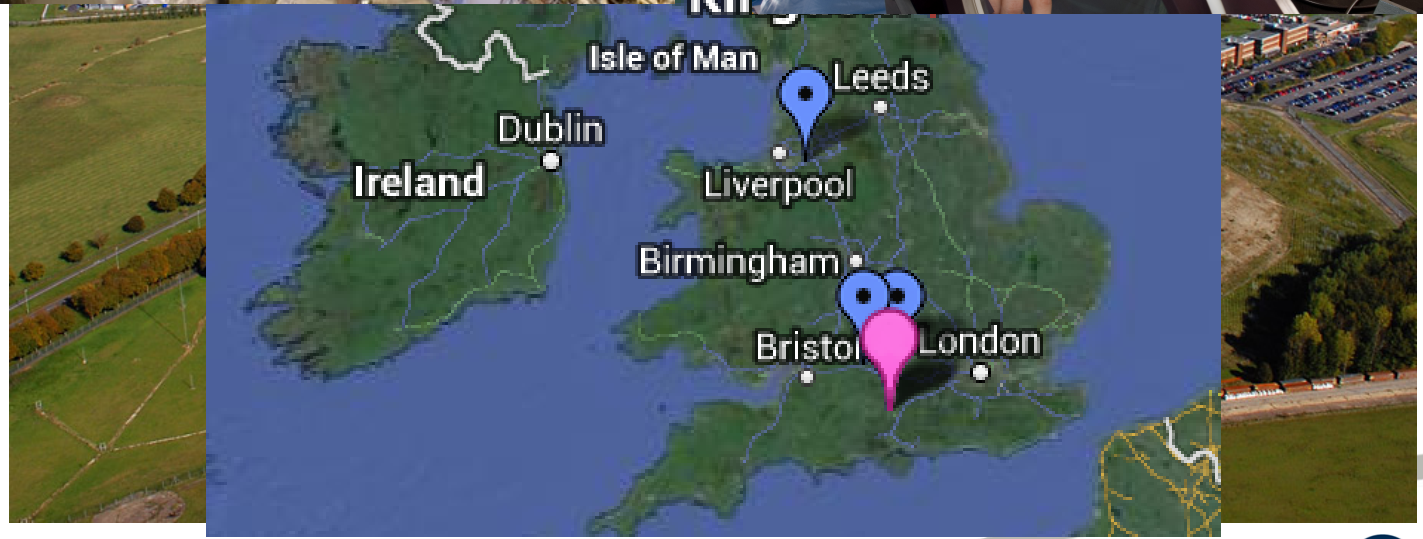


Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

ISIS Neutron Source

Central Laser Facility



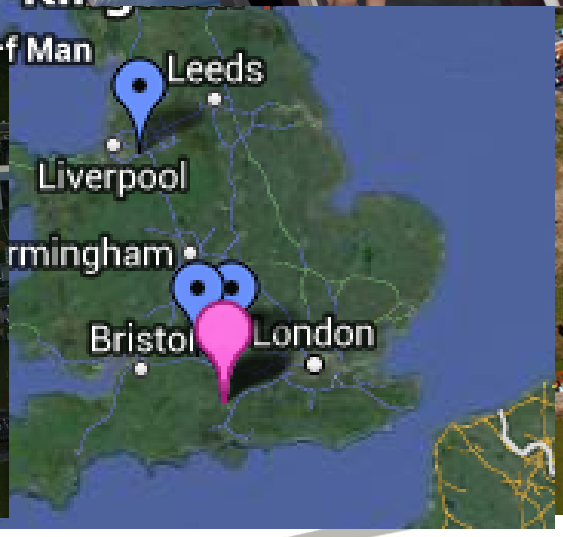
Science & Technology Facilities Council

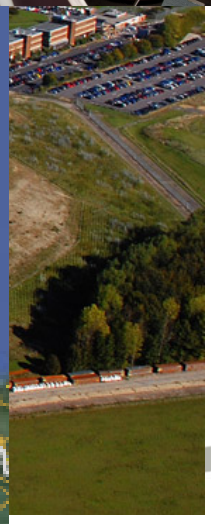
STFC Rutherford Appleton Laboratory

ISIS Neutron Source

Central Laser Facility

Diamond Light Source





Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

ISIS Neutron Source

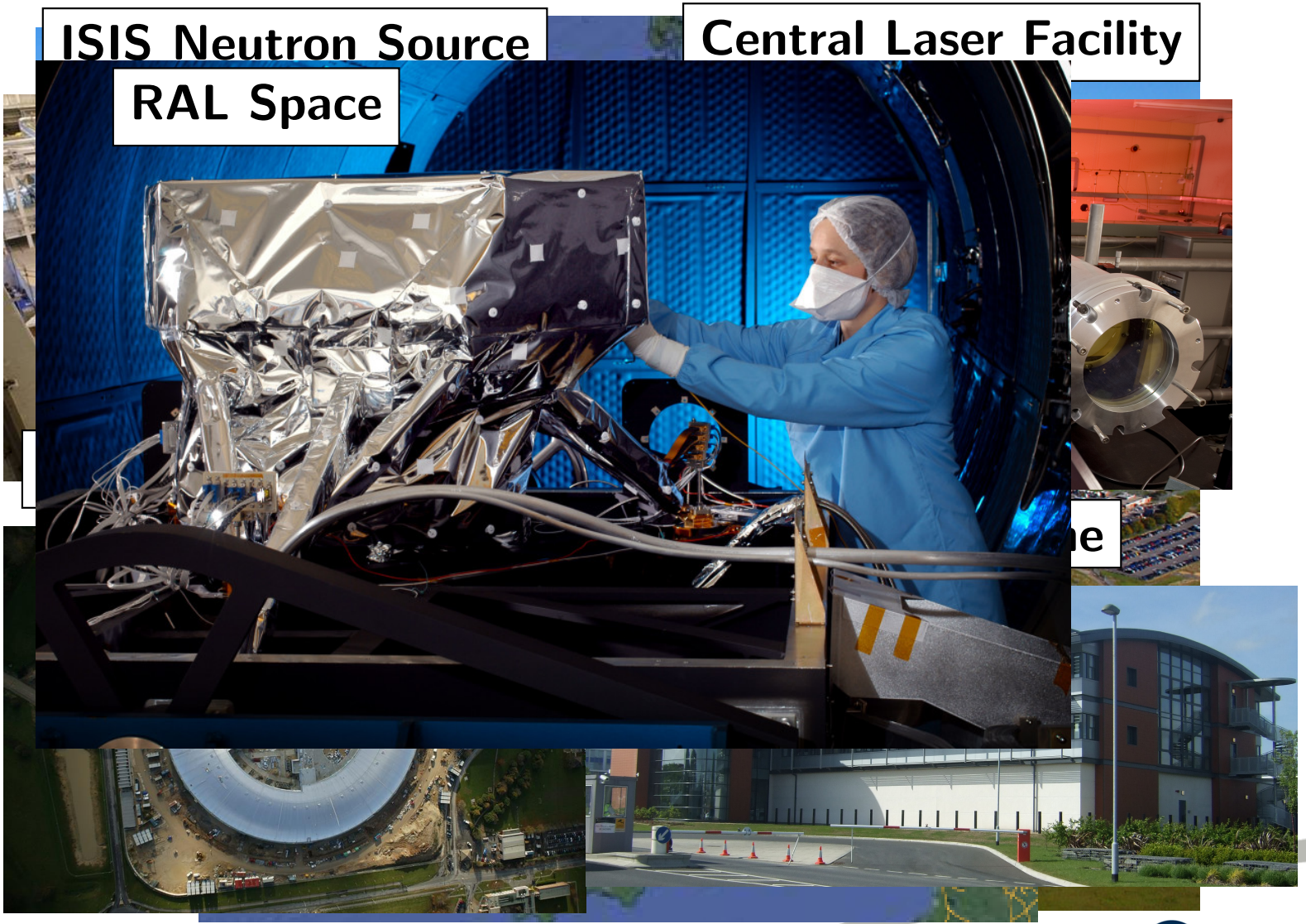
Central Laser Facility

Diamond Light Source

LHC Tier One

Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

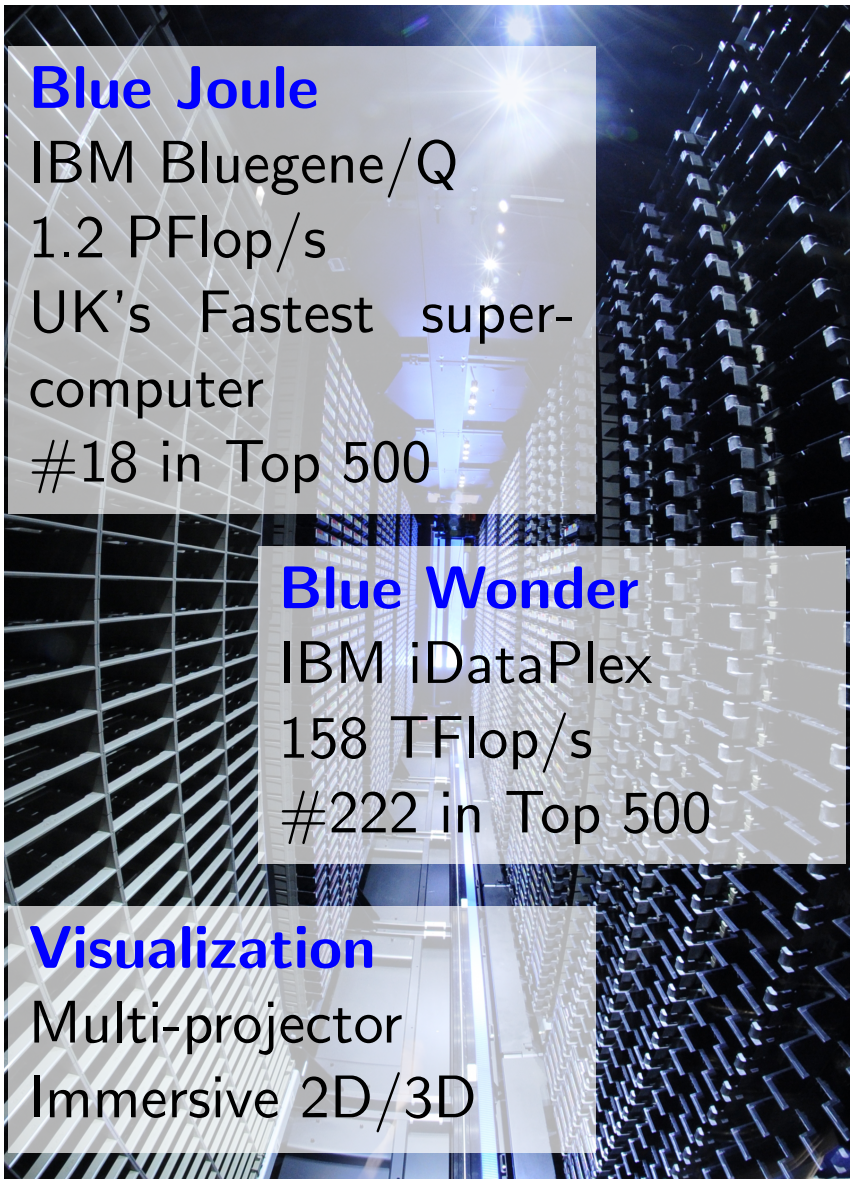


Science & Technology Facilities Council

STFC Rutherford Appleton Laboratory

Science & Technology Facilities Council

The Hartree Centre



- £37.5m investment
- STFC/IBM collaboration
- Aim: HPC Development
- Industry and academia
- Actively looking for partners

http://www.stfc.ac.uk/hartree

Science & Technology Facilities Council

Numerical Analysis Group at RAL

- SCD provides world-class expertise and support for UK theoretical and computational science communities, in both academia and industry.
- Group is funded by an ESPRC grant and since late 1950s has been one of the leading UK numerical analysis groups.

Science & Technology Facilities Council

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.
Evgueni Ovtchinnikov	Eigensolvers, linear solvers, GPUs.
Sue Thorne	Preconditioners, saddle-point problems and sparse linear systems.

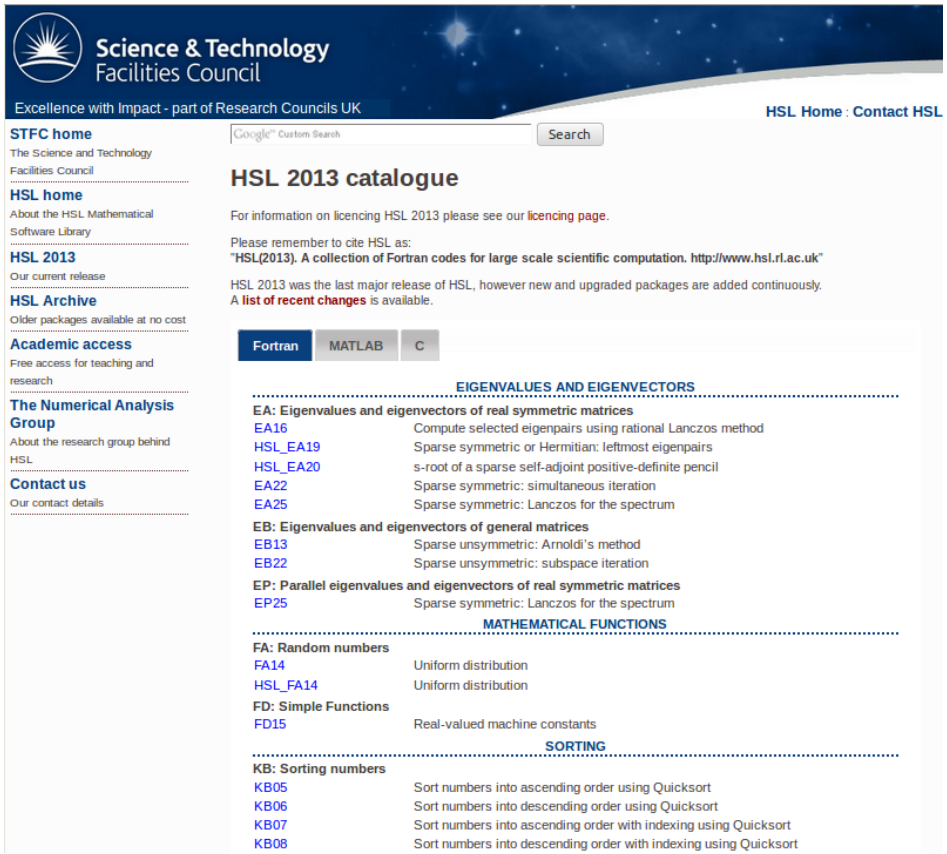
Science & Technology Facilities Council

Numerical Analysis Group at RAL

Main areas of expertise

- Sparse linear algebra
  - $Ax = b$
  - $Ax = \lambda Bx$
- Direct *and* Iterative
- Large-scale optimization
- High-quality software

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



Science & Technology Facilities Council

Sparse Direct Solvers

Solve

$$Ax = b$$

where A is Sparse.

Direct Methods Factorize $A = LU$, solve $Ly = b, Ux = y$.

Black-box, robust, **compute-bound**.

Memory-hungry⇒ slow for large matrices?.

Iterative Methods CG, GMRES, BiCGStab, etc.

Matrix-free. Fast? Efficient? **memory-bound**.

Non-robust, performance depends on preconditioner.

Science & Technology Facilities Council

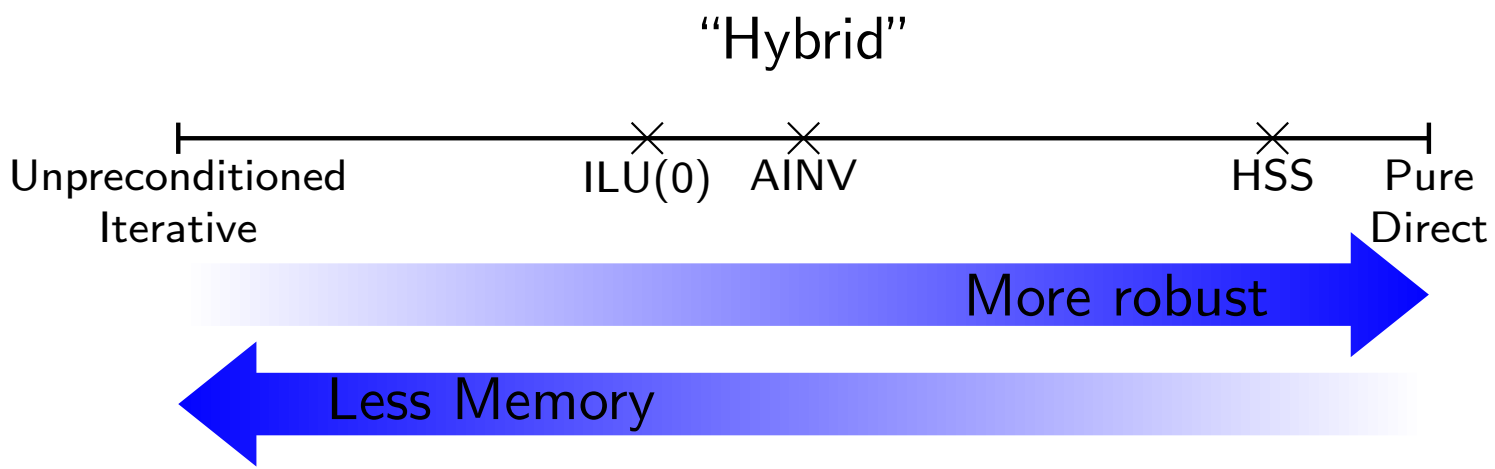
Sparse Direct Solvers

Solve

$$Ax = b$$

where A is Sparse.

New view: Spectrum



Science & Technology Facilities Council

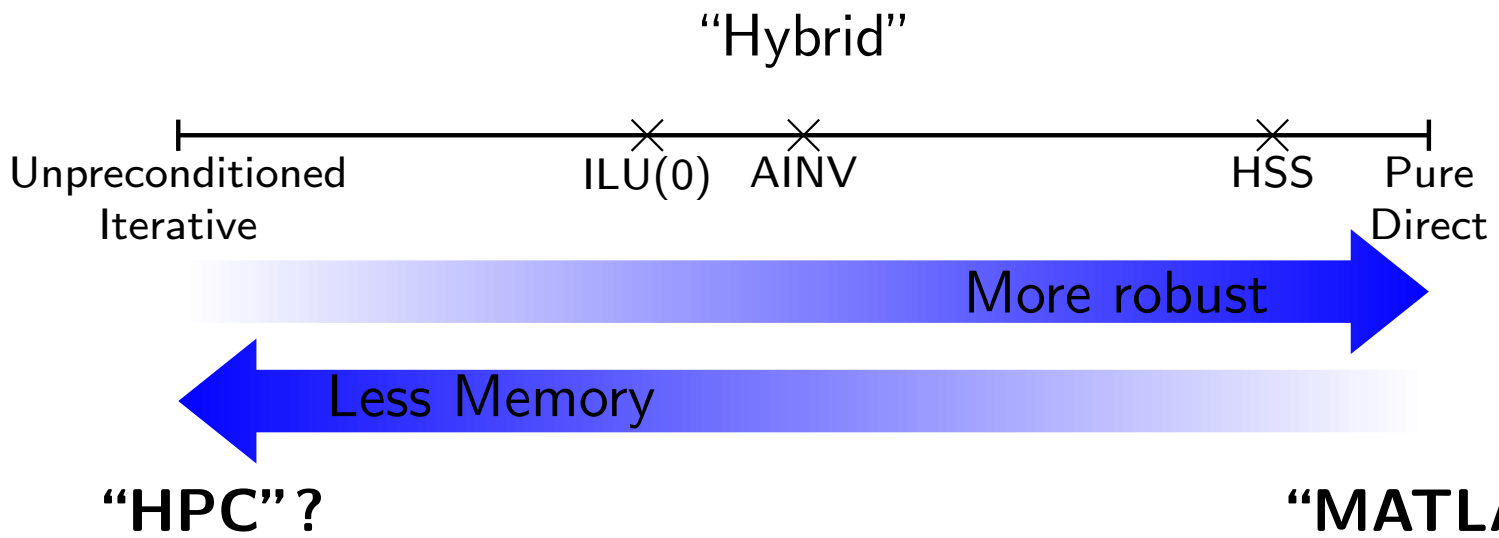
Sparse Direct Solvers

Solve

$$Ax = b$$

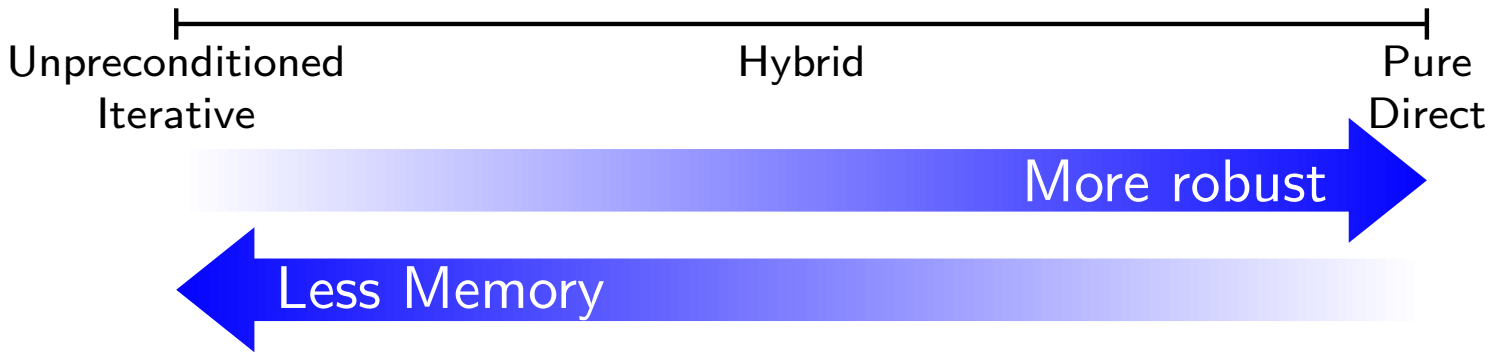
where A is Sparse.

New view: Spectrum

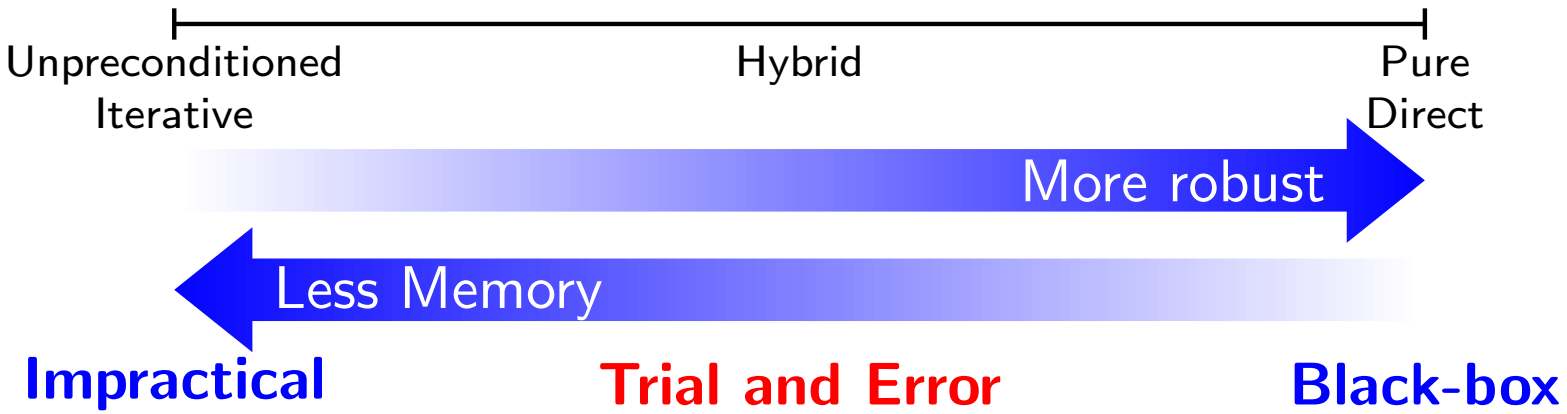


Science & Technology Facilities Council

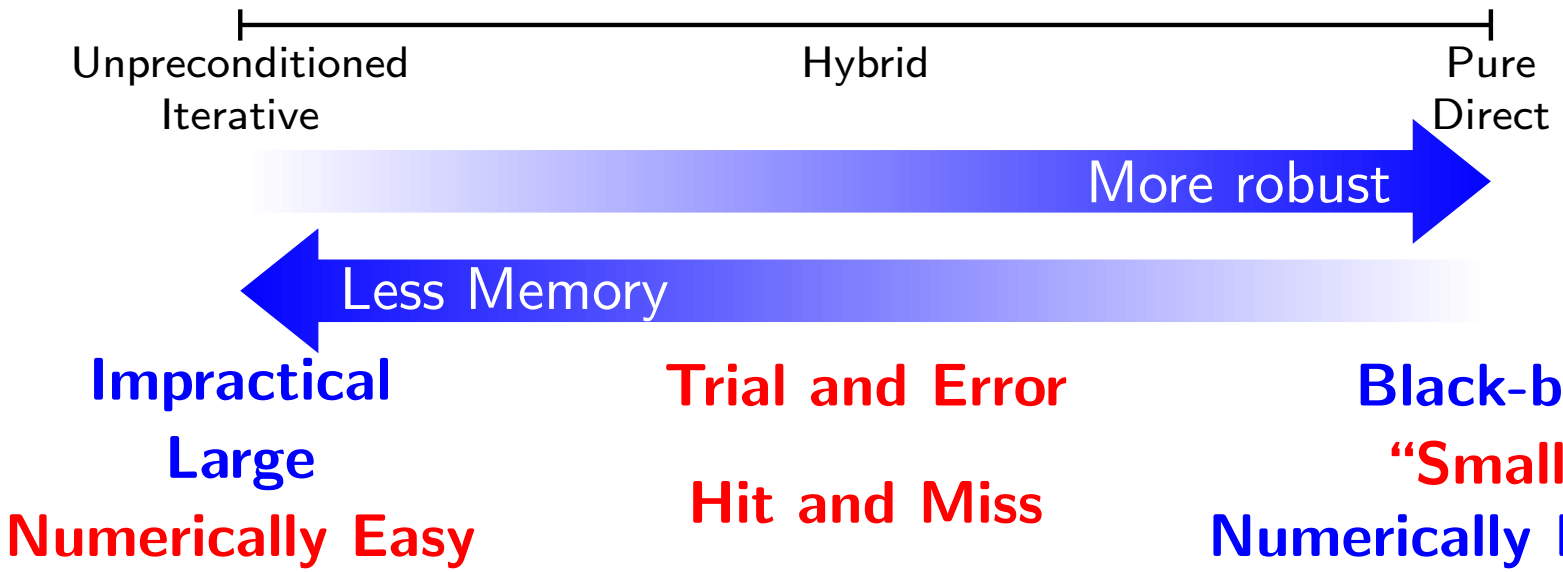
Horses for Courses



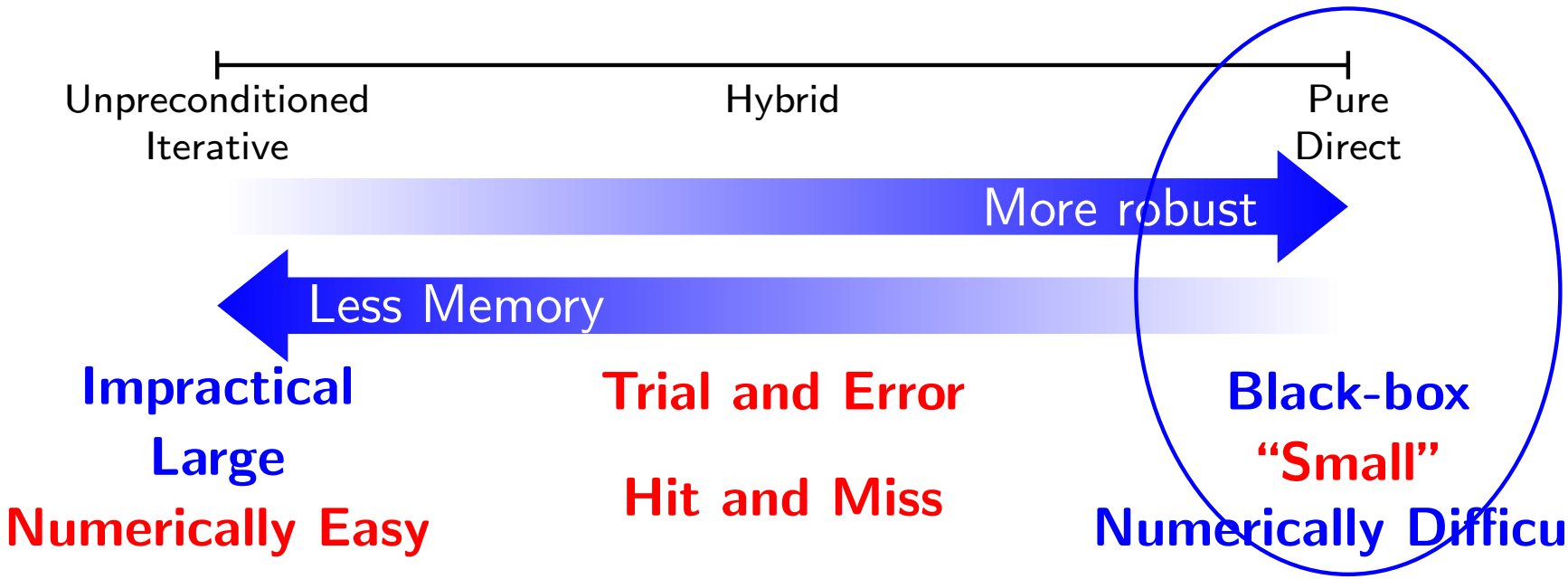
Horses for Courses



Horses for Courses



Horses for Courses



Challenge: + **Parallel**

Science & Technology Facilities Council

"If it fits in memory I can factorize it in under 10 minutes"

Problem	Memory	Flops	Factorize time
audikw_1	9.3 GB	5.8×10^{12}	25 seconds
bone010	8.0 GB	3.9×10^{12}	16 seconds
nd24k	2.4 GB	2.1×10^{12}	18 seconds

- audikw_1 is 943,695 $\times$ 943,695 with 77,651,847 non-zeroes.
- bone010 is 986,703 $\times$ 986,703 with 47,851,783 non-zeroes.
- nd24k is 72,000 $\times$ 72,000 with 28,715,634 non-zeroes.

Science & Technology Facilities Council

Problem classification

Problem classification

Symmetric

Positive-definite

Cholesky factorization ($A = LL^T$)

Conjugate Gradient

Algebraic Multigrid

Combinatorial preconditioners

Unsymmetric

Problem classification

Symmetric

Positive-definite

Cholesky factorization ($A = LL^T$)

Conjugate Gradient

Algebraic Multigrid

Combinatorial preconditioners

Indefinite

Sym. Indef. factorization

$A = LDL^T$

Unsymmetric

Problem classification

Symmetric

Positive-definite

Cholesky factorization ($A = LL^T$)

Conjugate Gradient

Algebraic Multigrid

Combinatorial preconditioners

Indefinite

Sym. Indef. factorization

$A = LDL^T$

Unsymmetric

Almost-symmetric pattern

Order as $A + A^T$ or AA^T

Then apply symmetric LU

Problem classification

Symmetric

Positive-definite

Cholesky factorization ($A = LL^T$)

Conjugate Gradient

Algebraic Multigrid

Combinatorial preconditioners

Indefinite

Sym. Indef. factorization

$A = LDL^T$

Unsymmetric

Almost-symmetric pattern

Order as $A + A^T$ or AA^T

Then apply symmetric LU

Highly unsymmetric

Unsymmetric LU:

Find block triangular form

Markowtiz ordering $(r-1)(c-1)$

Switch to dense at end

Codes

Symmetric	Unsymmetric
Positive-definite	Almost-symmetric pattern
Indefinite	Highly unsymmetric

Codes

Symmetric	Unsymmetric
Positive-definite	Almost-symmetric pattern
HSL_MA77, HSL_MA87, HSL_MA97 PARDISO MUMPS WSMP	
Indefinite	Highly unsymmetric

Codes

Symmetric

Positive-definite

HSL_MA77, HSL_MA87,

HSL_MA97

PARDISO

MUMPS

WSMP

Indefinite

MA57, HSL_MA86, HSL_MA97

PARDISO

MUMPS

WSMP

Unsymmetric

Almost-symmetric pattern

Highly unsymmetric

Codes

Symmetric

Positive-definite

HSL_MA77, HSL_MA87,
HSL_MA97
PARDISO
MUMPS
WSMP

Indefinite

MA57, HSL_MA86, HSL_MA97
PARDISO
MUMPS
WSMP

Unsymmetric

Almost-symmetric pattern

UMFPACK, MA41
PARDISO
MUMPS
WSMP
SuperLU

Highly unsymmetric

Codes

Symmetric

Positive-definite

HSL_MA77, HSL_MA87,

HSL_MA97

PARDISO

MUMPS

WSMP

Indefinite

MA57, HSL_MA86, HSL_MA97

PARDISO

MUMPS

WSMP

Unsymmetric

Almost-symmetric pattern

UMFPACK, MA41

PARDISO

MUMPS

WSMP

SuperLU

Highly unsymmetric

HSL_MA48

KLU

Iterative solver notice:
The following comments may also apply to factorization-based preconditioners!

"Small" + Parallel = Strong scaling

Example

Non-linear optimization solver, unknown problem origin

$\Rightarrow$ Preconditioning difficult (at best)!

Direct solver: solves 100 systems ($n = 35000$) to reach solution in 5 seconds. 95% of time in linear solver.

$\Rightarrow$ 0.05s per serial factorization

Maybe 2 million flops with 250,000 non-zeroes (8 flops/non-zero)

2015 desktop: 16 CPU cores + 1024 GPU cores?

$\Rightarrow$ Fewer than 250 non-zeroes per core!

Science & Technology Facilities Council

"Small" + Parallel = Strong scaling

Context: Many sequential small factorizations.
Enemy is: **LATENCY!**
Aim: Avoid fine-grained communication.

8 flops/non-zero $\Rightarrow$ Communication is King!

"Small" + Parallel = Strong scaling

Context: Many sequential small factorizations.
Enemy is: **LATENCY!**
Aim: Avoid fine-grained communication.

8 flops/non-zero $\Rightarrow$ Communication is King!

Solution: Communication avoiding algorithms

- See Grigori + Demmel for work on dense algorithms.
- Our work on sparse LDL^T.

Science & Technology Facilities Council

"Small" + Parallel = Strong scaling

Context: Many sequential small factorizations.
Enemy is: **LATENCY!**
Aim: Avoid fine-grained communication.

8 flops/non-zero $\Rightarrow$ Communication is King!

Solution: Communication avoiding algorithms

- See Grigori + Demmel for work on dense algorithms.
- Our work on sparse LDL^T.

Research concepts: In-cache solver

- Speeds up subsequent solves, saves memory bandwidth
- Xeon Phi has (total) 32MB L2 cache
- 2020: Plan for 16,000 cores each with 1MB cache?

Science & Technology Facilities Council

Communication avoiding sparse LDL^T

Sparse LDL^T:

- Quickly identify dense submatrices
- Use BLAS
- Most *FLOPS* in matrix-matrix multiply
- **But** *critical path* is half pivoting.

Communication avoiding sparse LDL^T

Sparse LDL^T:

- Quickly identify dense submatrices
- Use BLAS
- Most *FLOPS* in matrix-matrix multiply
- **But** *critical path* is half pivoting.

Traditional pivoting:

- Compare largest entry in column to diagonal
- Accept/reject

Communication avoiding sparse LDL^T

Sparse LDL^T:

- Quickly identify dense submatrices
- Use BLAS
- Most *FLOPS* in matrix-matrix multiply
- **But** *critical path* is half pivoting.

Traditional pivoting:

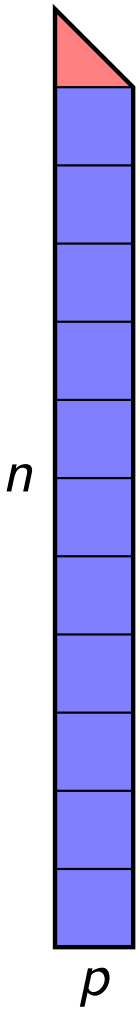
- Compare largest entry in column to diagonal
- Accept/reject

Try-it-and-see pivoting:

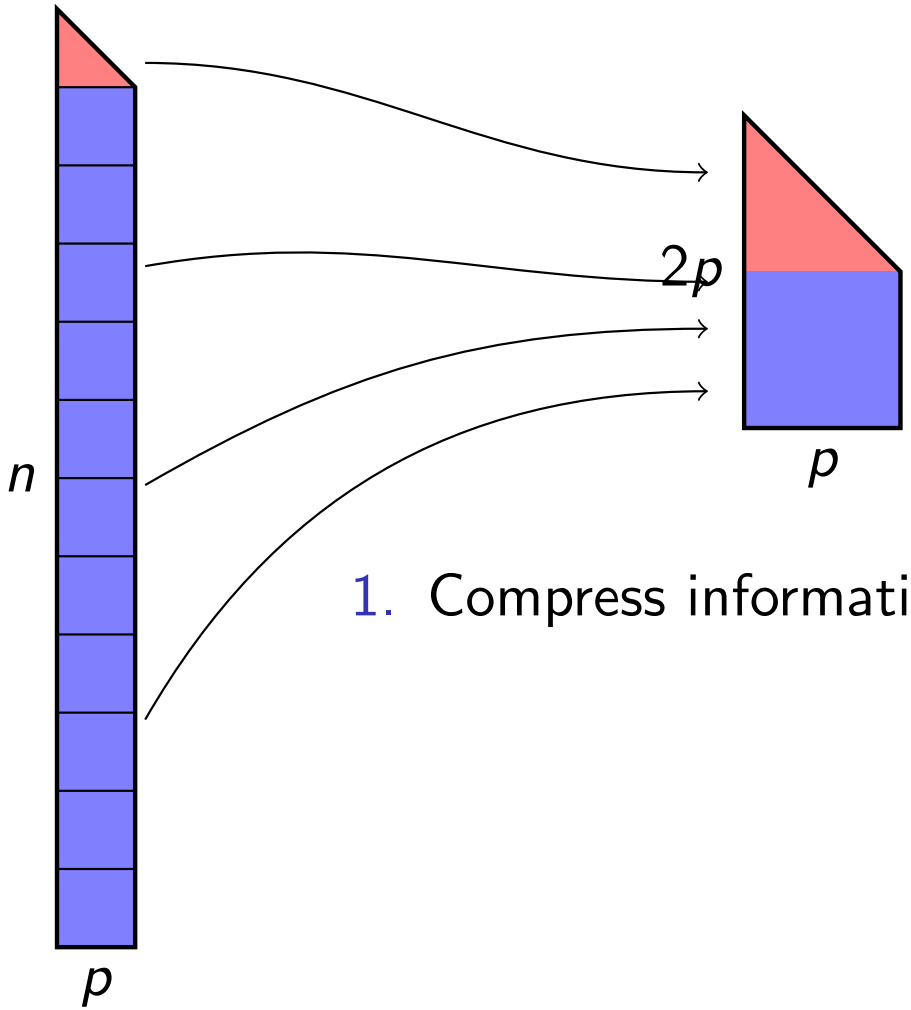
- Store a copy before pivoting
- Do numerical test a posteroi
- Back-track if it breaks
- **Still need a fall back plan**

Science & Technology Facilities Council

Compressed pivoting

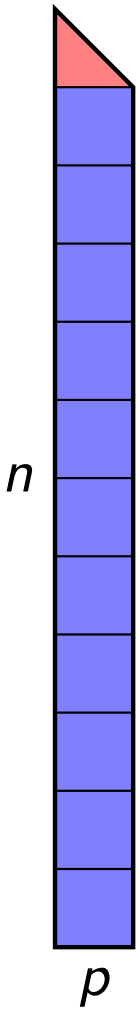


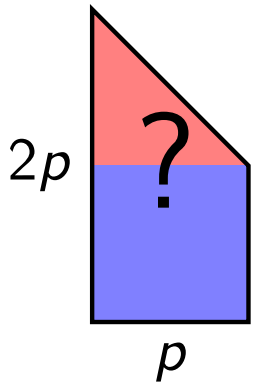
Compressed pivoting



1. Compress information into small matrix

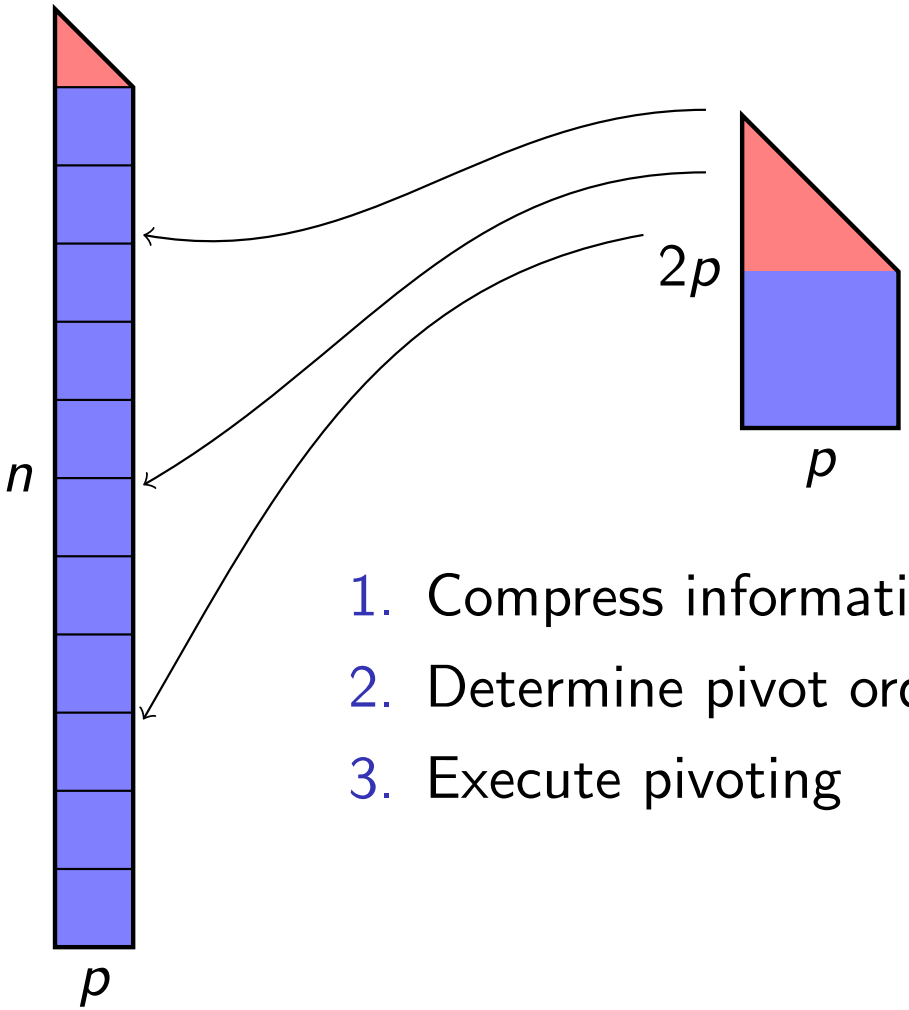
Compressed pivoting





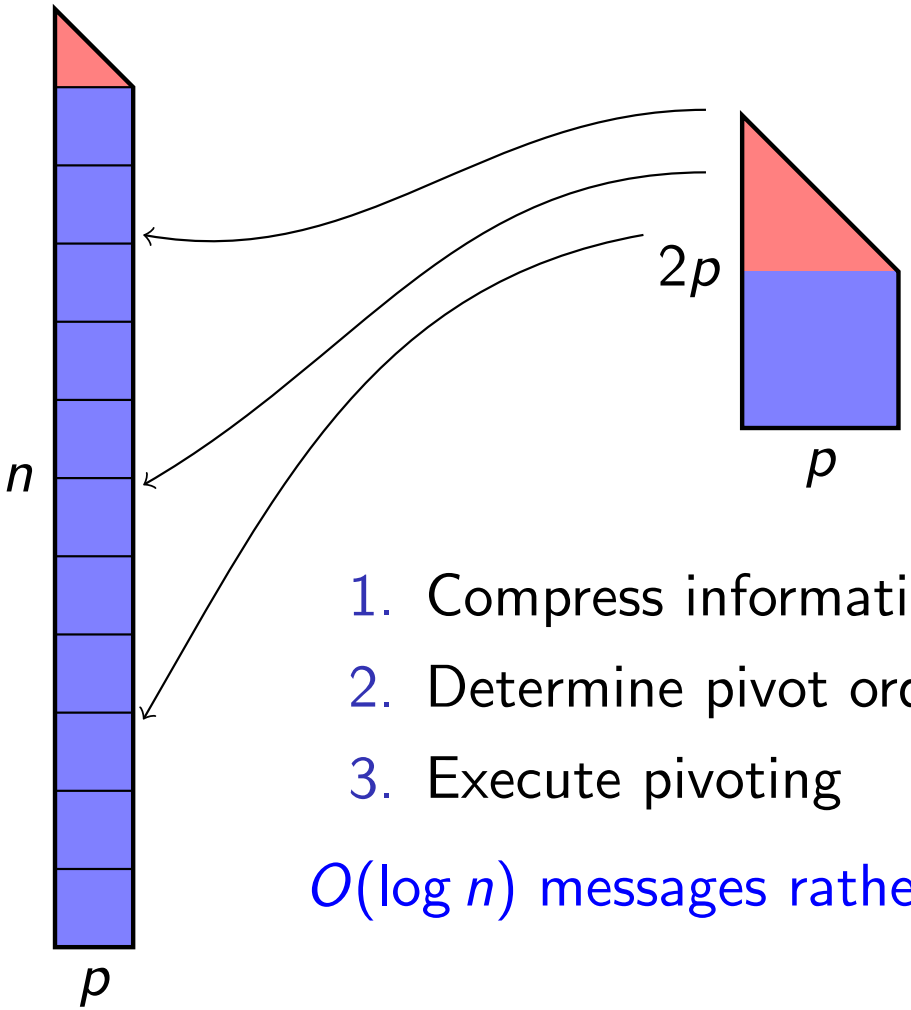
1. Compress information into small matrix
2. Determine pivot order

Compressed pivoting



1. Compress information into small matrix
2. Determine pivot order
3. Execute pivoting

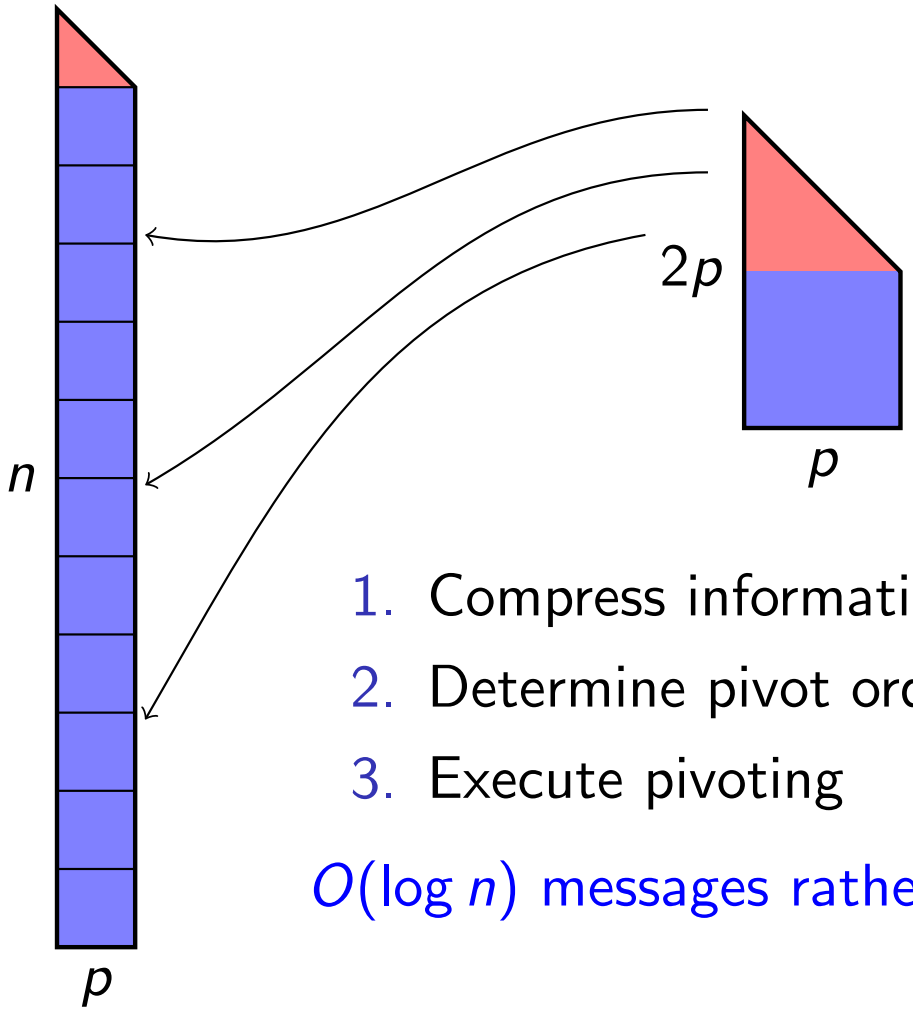
Compressed pivoting



1. Compress information into small matrix
2. Determine pivot order
3. Execute pivoting

$O(\log n)$ messages rather than $O(p \log n)$

Compressed pivoting



- **2× faster on CPU**
- Working on GPU version

1. Compress information into small matrix
2. Determine pivot order
3. Execute pivoting

$O(\log n)$ messages rather than $O(p \log n)$

Science & Technology Facilities Council

Memory bound solve phase

Context: Reading all those factors from memory (twice!)
Enemy is: **BANDWIDTH!**
Aim: Compress the data somehow.

Parallel factorization $\Rightarrow$ need faster solve.

Memory bound solve phase

Context: Reading all those factors from memory (twice!)
Enemy is: **BANDWIDTH!**
Aim: Compress the data somehow.

Parallel factorization $\Rightarrow$ need faster solve.

Solutions:

- Generic compression
- Low-rank approximation for elliptic problems

Science & Technology Facilities Council

Generic Compression

J.D. Hogg and J.A. Scott
A note on the solve phase of a multicore solver
RAL-TR-2010-007

Idea:
Compress data blocks before storing factors, decompress into cache before use. Otherwise 1 flops/non-zero in solve phase.

LZO Compression Library Higher compression than GZIP, *much* faster.

Science & Technology Facilities Council

Generic Compression

J.D. Hogg and J.A. Scott
A note on the solve phase of a multicore solver
RAL-TR-2010-007

Idea:
Compress data blocks before storing factors, decompress into cache before use. Otherwise 1 flops/non-zero in solve phase.

LZO Compression Library Higher compression than GZIP, *much* faster.

Outcome:
Performance matched that of original algorithm:
Wait for more flops/unit bandwidth.

Science & Technology Facilities Council

Low-rank approximation

Multiple works by *J. Xia, S. Chandrasekaran, M. Gu, X.S. Li* et al.

Background:

Fast multipole methods

$\mathcal{H}$-matrices

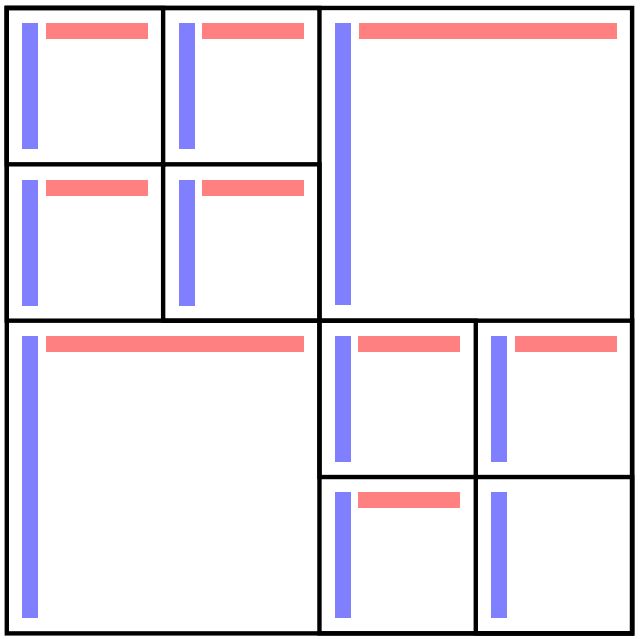
Hierarchical Semi-Separable (HSS) matrices.

Elliptic PDEs:

Little interaction at long-range

$\Rightarrow$ low rank away from diagonal.

$\mathcal{H} =$



Science & Technology Facilities Council

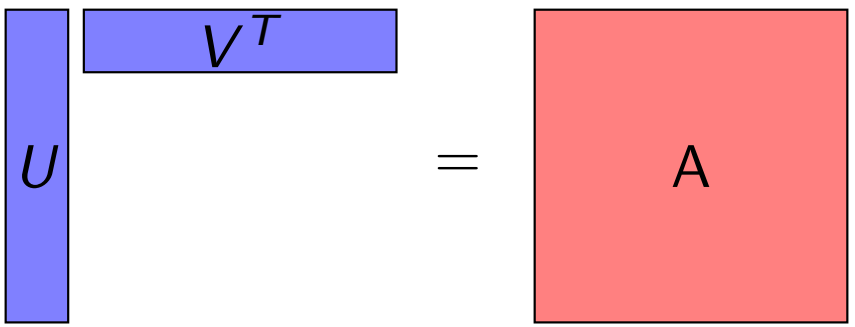
Low-rank approximation cont.

Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, Weisbecker
Improving multifrontal methods by means of block low-rank representations
INRIA Report No 8199

Idea:

Sparse $\mathcal{H}$ and HSS forms tricky
$\Rightarrow$ Block Low-Rank + RRQR to recover

Rank-revealing QR:



Flops are cheap!

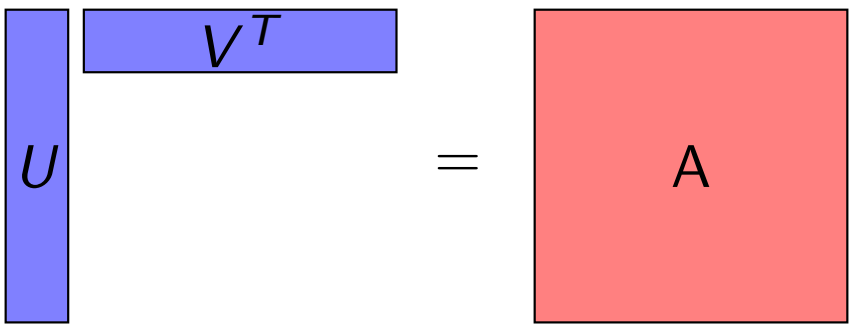
Low-rank approximation cont.

Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, Weisbecker
Improving multifrontal methods by means of block low-rank representations
INRIA Report No 8199

Idea:

Sparse $\mathcal{H}$ and HSS forms tricky
$\Rightarrow$ Block Low-Rank + RRQR to recover

Rank-revealing QR:



Flops are cheap!

Outcome:

Good *preconditioner* for some classes of matrix.
More work needed!

Science & Technology Facilities Council

Bit-compatibility

Context: Subsequent runs on same data give different answers
Enemy is: **USER EDUCATION** or **FLOATING POINT**?
Aim: Get the same answer every time.

$$1 + (\epsilon/2 + \epsilon/2) \neq (1 + \epsilon/2) + \epsilon/2$$

Bit-compatibility

Context: Subsequent runs on same data give different answers
Enemy is: **USER EDUCATION** or **FLOATING POINT**?
Aim: Get the same answer every time.

$$1 + (\epsilon/2 + \epsilon/2) \neq (1 + \epsilon/2) + \epsilon/2$$

Why would we not do this?

- If we don't, answers are still equally valid
- Less efficient: restrict parallelism, optimization
- More difficult to achieve
- Must be achieved by all libraries used

Science & Technology Facilities Council

Bit-compatibility

Context: Subsequent runs on same data give different answers
Enemy is: **USER EDUCATION** or **FLOATING POINT**?
Aim: Get the same answer every time.

$$1 + (\epsilon/2 + \epsilon/2) \neq (1 + \epsilon/2) + \epsilon/2$$

But it's very attractive

- Hard to debug without it: make it an option?
- Confuses non-expert users no end
- Methods built on top may behave unexpectedly:

  e.g. Different local maxima found for non-linear optimization
  Different iteration counts

Science & Technology Facilities Council

Achieving bit-compatibility

Option #1: Add up in the same order

J.D. Hogg and J.A. Scott, **HSL_MA97**

Enforce ordering on additions:

$$((1+2)+3)+4 \qquad \text{or} \qquad (1+2)+(3+4).$$

Option #2: Add up in high precision

J.D. Hogg and J.A. Scott

Achieving bit-compatibility in sparse direct solvers

Use double-double precision for accumulations: compensated addition with a salt τ: $\epsilon_{dd} < \tau < \epsilon_d$.

Only communicates in double, and extra flops are cheap.

Science & Technology Facilities Council

Compensated addition for error detection

Let double precision $= d$, extended precision $= e$.
Compensated addition:

$$\begin{aligned} sum &= (a+b)_e \\ corr &= b-(sum-a) \end{aligned}$$

Total error for large sum $\delta = \sum corr$ for each addition.

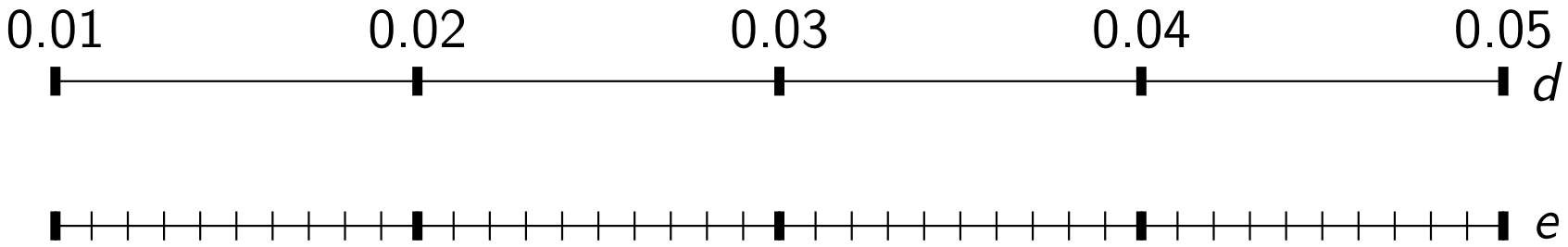
Ideally we want:

$$(sum_e + \delta_e)_d = (sum_e - \delta_e)_d$$

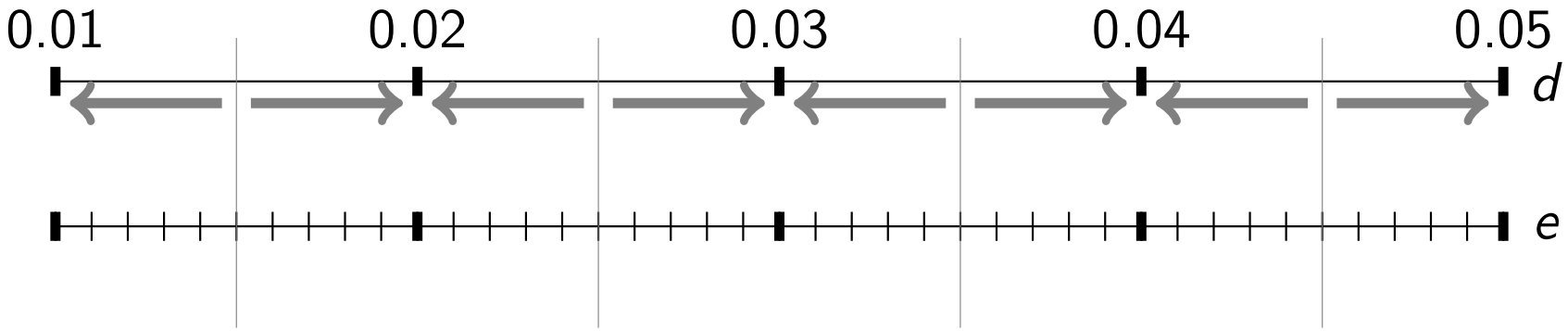
i. e. exactly the true value rounded to double precision.

Science & Technology Facilities Council

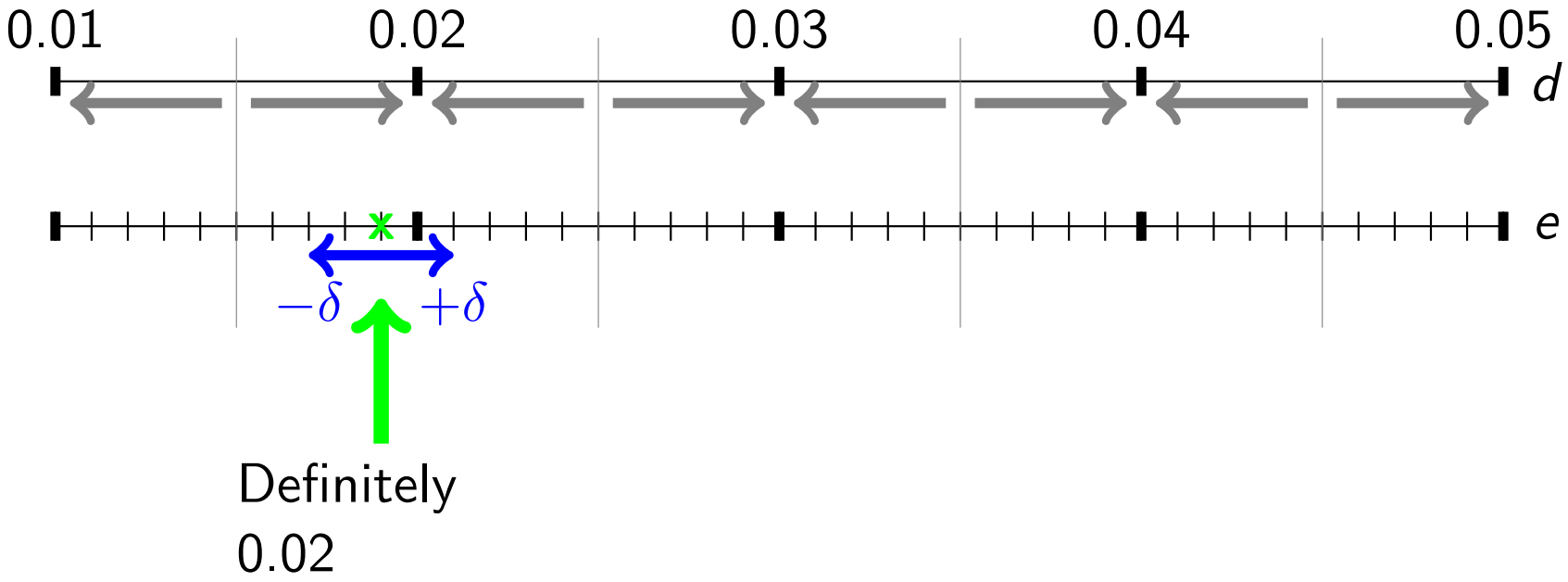
On a number line



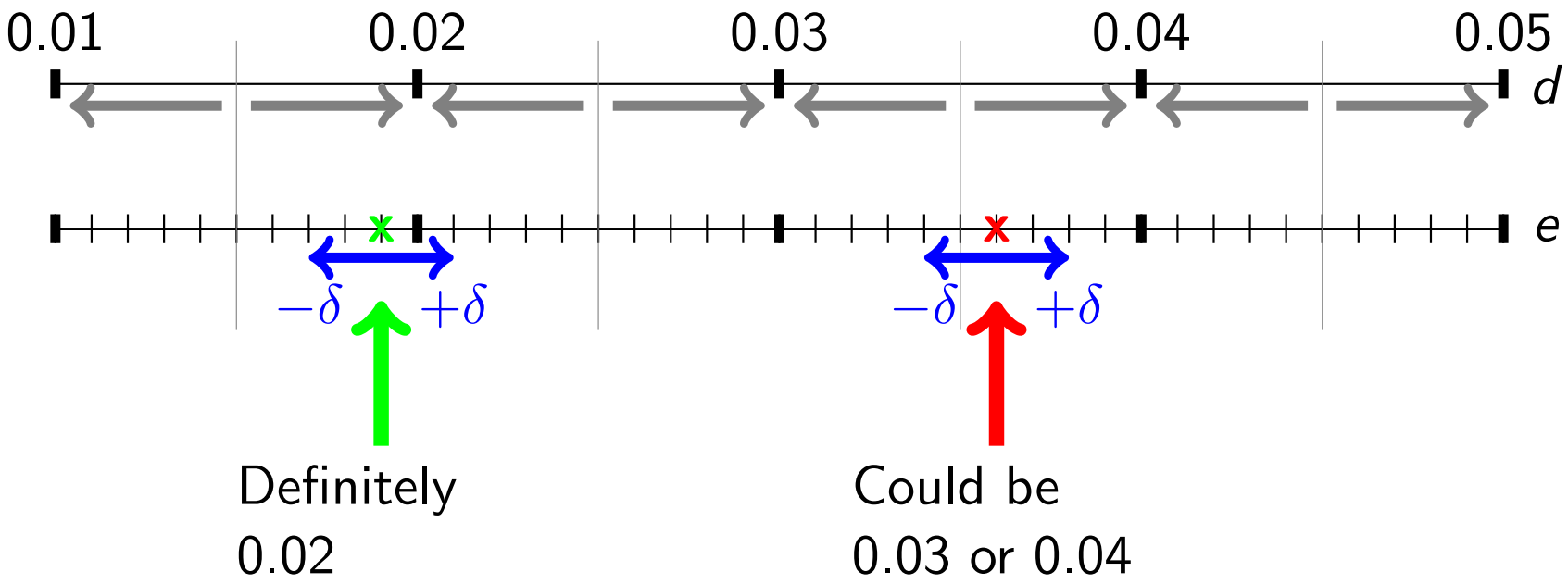
On a number line



On a number line

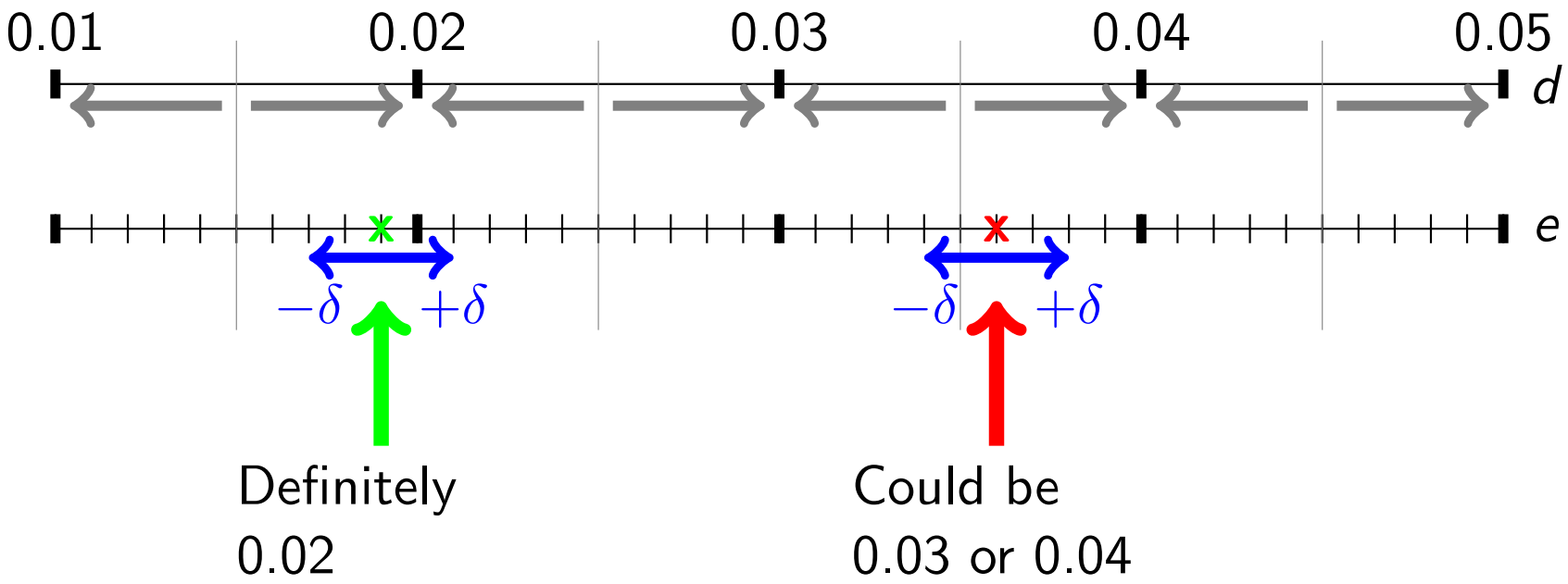


On a number line



Science & Technology Facilities Council

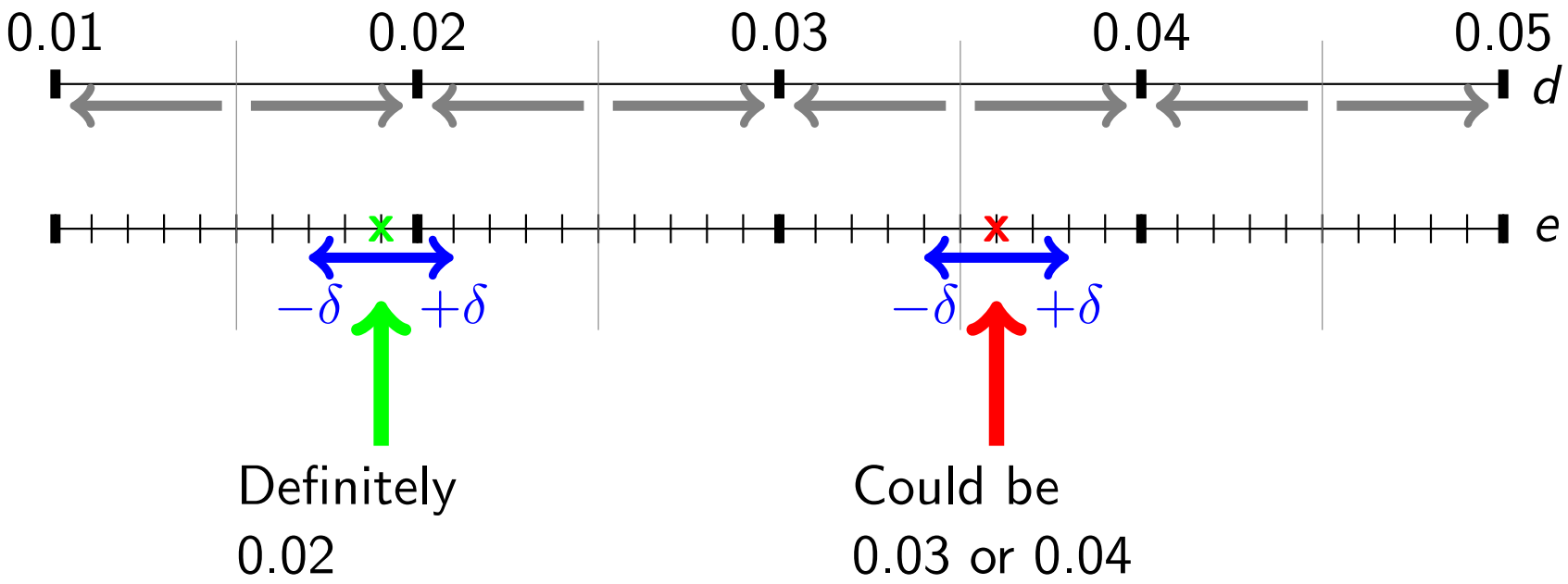
On a number line



For d =double, e =quad, there are 2^{60} more e than d values.
If we're accurate to a couple of binary places $P(red) = 10^{-15}$.
$P(10^9$ values accurate$) = 10^{-8}$.

Science & Technology Facilities Council

On a number line



For d =double, e =quad, there are 2^{60} more e than d values.
If we're accurate to a couple of binary places $P(red) = 10^{-15}$.
$P(10^9$ values accurate$) = 10^{-8}$.

Observed: $P(red) = 0.05$. !!!

Science & Technology Facilities Council

Salting

Why?

- Adding operands of different magnitudes

$$(1.000 \times 10^{0} + 1.005 \times 10^{-1})_d$$

- Answer lies on a rounding discontinuity!

Salting

Why?

- Adding operands of different magnitudes

$$(1.000 \times 10^{0} + 1.005 \times 10^{-1})_d$$

- Answer lies on a rounding discontinuity!

Solution

Perturb problem away from discontinuities:

Add *salt* τ to the sum.

τ must be bigger than common δ, but smaller than ϵ_d

$\tau = 0.2345 \times 2^{exponent(\epsilon_d)}$ fixed everything.

Overhead: 60% slower to run (unoptimized code)

Science & Technology Facilities Council

Scaling

Context: Running on BIG machines
Enemy is: **Insufficient traditional parallelism**
Aim: Find as much parallelism as possible.

Traditional sparse direct parallelism:

- Tree-level
- Node-level
- Assembly

Problems:

- Pivoting disrupts a priori scheduling
- Stop-start in nature
- Insufficiently fine grained

Science & Technology Facilities Council

Task DAGs

Take a leaf out of dense direct book:

- Work with blocked algorithms
- Identify data dependencies
- Schedule via a task graph

Task graph:

- Each task is a node
- Data dependencies are edges

Task DAGs

Take a leaf out of dense direct book:

- Work with blocked algorithms
- Identify data dependencies
- Schedule via a task graph

Task graph:

- Each task is a node
- Data dependencies are edges

Benefits:

- Scheduling separated from mathematics
- Interleave work on different operations
- Avoid stop/start
- Identifies all possible parallelism?

Science & Technology Facilities Council

Sparse task-based

Sparse task-based implementation *exist*: `HSL_MA86`, `HSL_MA87`, `PaStiX`.

Problems:

- Block alignments — need dynamic reblocking for best efficiency.
- Building on top of LAPACK/PLASMA — dynamic reblocking on same data desirable.
- Building on top of LAPACK/PLASMA — can we use the same task scheduler?
- Dynamic task sizing — splitting/merging across levels.
- Bit-compatibility?

Science & Technology Facilities Council

Tasking

- Each task may have its own way of blocking.
- Run in parallel — different optimal block sizes.
- Want to compose libraries.

Combinatorial Preconditioners

Papers by combinations of:
Klarreich, Koutis, Miller, Peng, Vaidya, Spielman, Teng, Kolla

General-purpose preconditioners?

- Symmetric diagonally-dominant **only** (so far...)
- Based on *support theory* for network laplacians
- Heavily reliant on efficient graph algorithms
- $O(m \log n (\log \log n)^2)$

Watch this space???

Science & Technology Facilities Council

Summary

"Direct" Methods Still required:

- Black-box solution
- Small problems
- Numerically difficult problems

Challenges:

- Strong scaling
- Parallel pivoting
- Bit-compatibility (software/user education)
- Interface to rest of software stack (up *and* down)
- Hybrid methods

Science & Technology Facilities Council

But iterative methods aren't perfect either...

Iterative methods challenges

If Matrix-vector product is main cost:

- Already Memory-bound
- Look for ways to use spare cycles $\Rightarrow$ More expensive preconditioning?
- 2 or 4 M-v product not much more expensive than 1 M-v. Can you exploit this?

Iterative methods challenges

If Matrix-vector product is main cost:

- Already Memory-bound
- Look for ways to use spare cycles $\Rightarrow$ More expensive preconditioning?
- 2 or 4 M-v product not much more expensive than 1 M-v. Can you exploit this?

Existing Efforts:

- *Mark Hoemmen* (Berkeley), Communication Avoiding Krylov Methods
- Computes $[v, Av, A^2v, ..., A^sv]$ simultaneously
- Uses QR for orthogonalize
- Need to use Chebyshev basis for stability

Science & Technology Facilities Council

Science & Technology
Facilities Council

Thank you!

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