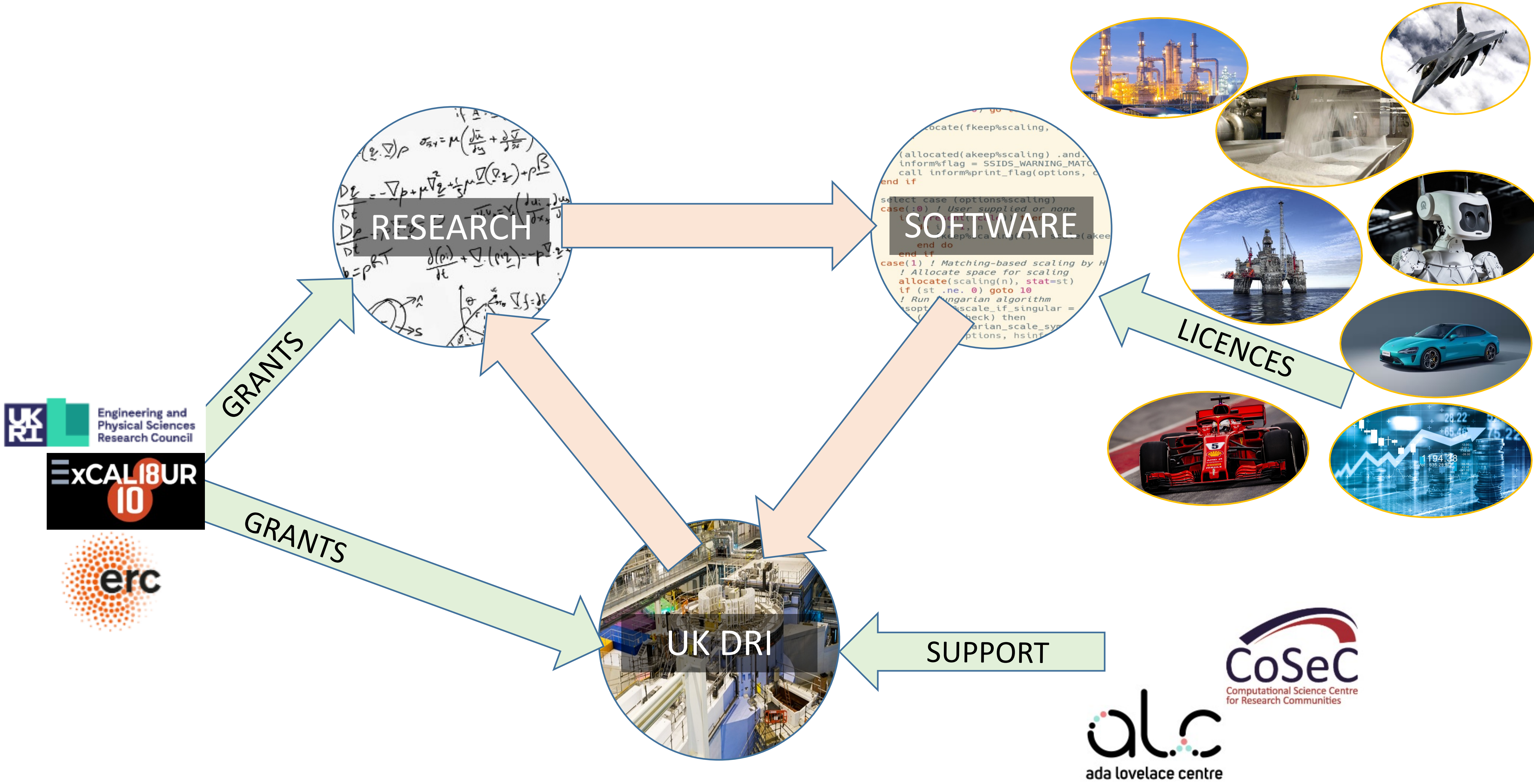
Science and Technology Facilities Council

Scientific Computing

Scientific Computing – Computational Maths

Computational Maths Theme vision

We are recognized as **world leaders** in the areas of computational mathematics used across STFC, positioning ourselves as **essential partners** within the UK Digital Research Infrastructure ecosystem and beyond, connecting researchers and innovators to the mathematical software they need to carry out the most ambitious and creative research.


RESEARCH
SOFTWARE
UK DRI
GRANTS
GRANTS
LICENCES
SUPPORT
Engineering and Physical Sciences Research Council
ExCALIBUR
erc
CoSeC
Computational Science Centre for Research Communities
ada lovelace centre

Computational Mathematics Theme

Theme Lead: Tyrone Rees

Numerical Linear Algebra

Hussam Al Daas
Niall Bootland
Kathryn Lund
Maike Meier
Jennifer Scott
(Vacancy – principal researcher)

Continuous Optimization

Jari Fowkes
Nick Gould
Boris Shustin
(Vacancy – (senior) researcher)

Mathematical RSE

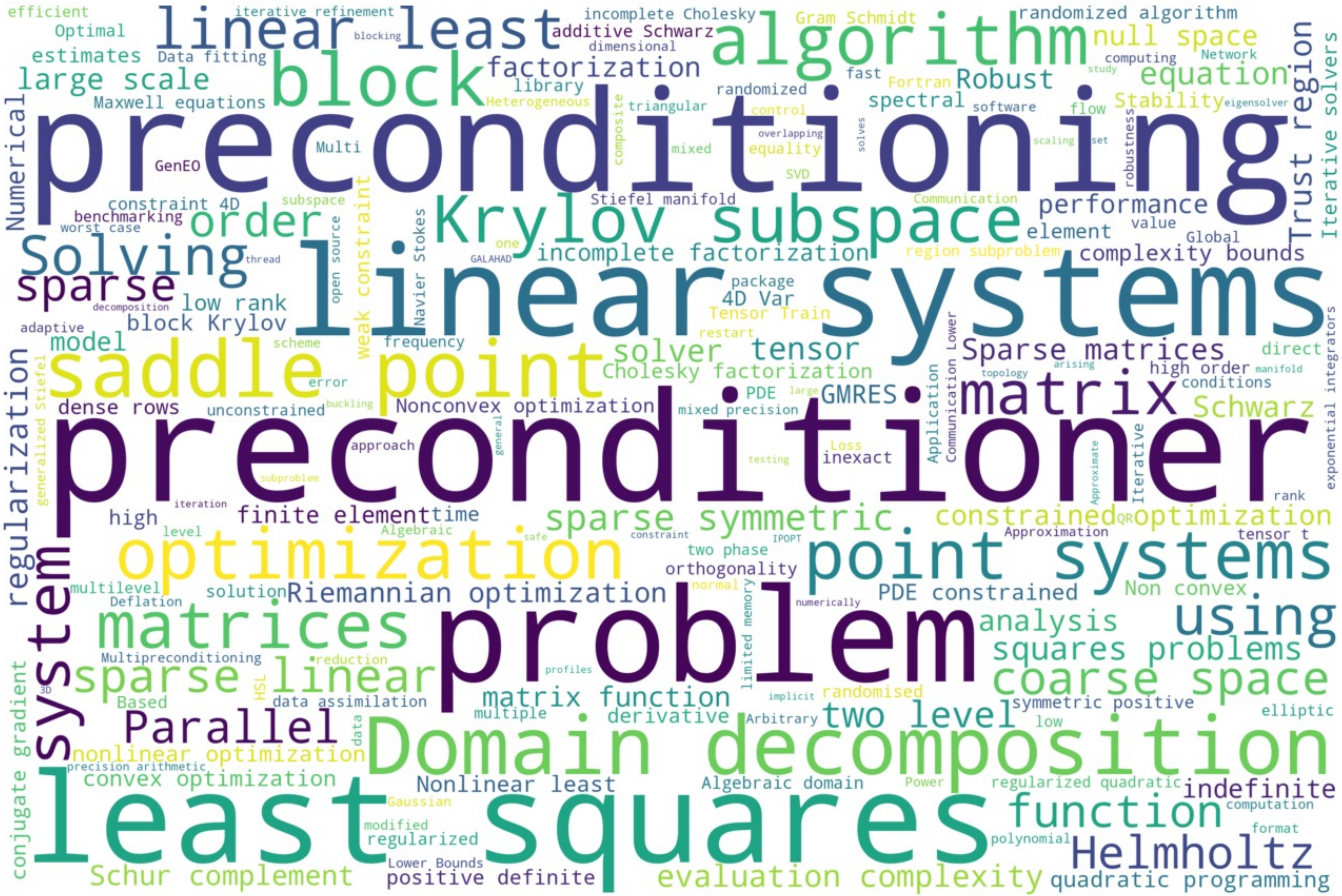
Andrew Lister
Jess Farmer
Letizia Protopapa
Bryce Shirley
Adam Greenbank
Reilly Hewitson

Emeritus Researchers

Iain Duff
John Reid

Visitors

Andreas Bock (DTU - postdoc)
Lorenzo Lazlo (Oxford – PhD)


preconditioning
linear systems
preconditioner
problem
least squares
Domain decomposition
saddle point
optimization
Krylov subspace
algorithm
linear least
block
point systems
matrix
matrices
sparse linear
system
Solving
sparse
order
Parallel
sparse symmetric
coarse space
squares problems
using
function
Helmholtz
two level
evaluation complexity
Schur complement
positive definite
quadratic programming
indefinite
regularization
conjugate gradient
Numerical
Trust region
Iterative solvers
null space
equation
factorization
Robust
large scale
estimates
Optimal
efficient
iterative refinement
blocking
incomplete Cholesky
additive Schwarz
dimensional
Gram Schmidt
randomized algorithm
computing
Network
Data fitting
library
randomized
fast
Fortran
Maxwell equations
Heterogeneous
triangular
control
spectral
software
flow
Stability
eigensolver
composite
overlapping
equality
mixed
Multi
GenEO
SVD
scaling
set
robustness
solves
constraint 4D
subspace
benchmarking
worst case
Stiefel manifold
Communication
performance
element
value
Global
incomplete factorization
complexity bounds
region subproblem
one
GALAHAD
thread
open source
weak constraint
Navier Stokes
package
4D Var
Tensor Train
low rank
block Krylov
decomposition
adaptive
restart
model
scheme
frequency
solver tensor
Sparse matrices
direct
high order
manifold
topology
arising
Cholesky factorization
conditions
PDE
large
GMRES
Schwarz
error
dense rows
unconstrained
buckling
Nonconvex optimization
mixed precision
generalized Stiefel
Application
Communication Lower
exponential integrators
approach
general
Loss
testing
inexact
Iterative
Approximate
subproblem
iteration
rank
finite element
time
high
level
Algebraic
constrained
QR
optimization
safe
constraint
IPOPT
Approximation
tensor t
two phase
orthogonality
multilevel
Deflation
solution
Riemannian optimization
normal
numerically
PDE constrained
Non convex
analysis
limited memory
Multipreconditioning
reduction
profiles
HSL
randomised
3D
Based
data assimilation
implicit
symmetric positive
elliptic
multiple
derivative
Arbitrary
low
data
nonlinear optimization
precision arithmetic
convex optimization
Nonlinear least
Algebraic domain
Power
regularized quadratic
Gaussian
computation
modified
format
regularized
polynomial
Lower Bounds

Alex Townsend

Marcus Webb

Davide Palitta

Frédéric Nataf

Haim Avron

Dominique Orban

Robert Scheichl

Edmond Chow

Martin Stoll

Miroslav Tůma

Grey Ballard

Coralia Cartis

Peter Benner

Kees Vuik

Stefan Güttel

Jack Dongarra

Laura Grigori

Stefan Vigerske

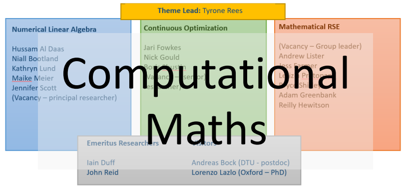
Ieva Daužickaitė

Amos S. Lawless

Pascal Hénon

Ivan Graham

Pierre-Henri Tournier



Computational Maths

Daniel P Robinson

Andreas Frommer

Lindon Roberts

Andy Wathen

Philippe L Toint

Miroslav Rozložník

Agnieszka Międlar

Peter Jan van Leeuwen

Pierre Jolivet

Hartwig Anzt

Victorita Dolean

Hans Mittelmann

Arvind K. Saibaba

Stefano Massei

Erin Carson

H. Alicia Kim

Daniel Kressner

Valeria Simoncini

Jen Pestana

Daniel Szyld

Yuji Nakatsukasa

Wil H.A. Schilders

Massimiliano Fasi

ACM Transactions on Mathematical Software

Most Frequent Affiliations

Name	Paper Counts
Argonne National Laboratory	
The University of Texas at Austin	
Sandia National Laboratories, New Mexico	47
Rutherford Appleton Laboratory	36
University of Florida	32
Nokia Bell Labs	32
Purdue University	
Lawrence Berkeley National Laboratory	
University of Toronto	
RWTH Aachen University	24
Umeå University	24

Association for Computing Machinery

As a scientific journal, *ACM Transactions on Mathematical Software (TOMS)* documents the theoretical underpinnings of numeric, symbolic, algebraic, and geometric computing applications. It focuses on analysis and construction of algorithms and programs, and the interaction of programs and architecture. Algorithms documented in *TOMS* are available as the Collected Algorithms of the ACM at calgo.acm.org.

Numerical Linear Algebra

Linear Systems

10  + 4  + 2  = £12.50

7  + 5  + 1  = £10.55

12  + 1  + 4  = £12.80

Linear Systems

$$\begin{bmatrix} 10 & 4 & 2 \\ 7 & 5 & 1 \\ 12 & 1 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 12.5 \\ 10.55 \\ 12.8 \end{bmatrix}$$

Linear Systems

$$\begin{bmatrix} 10 & 4 & 2 \\ 7 & 5 & 1 \\ 12 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12.5 \\ 10.55 \\ 12.8 \end{bmatrix}$$

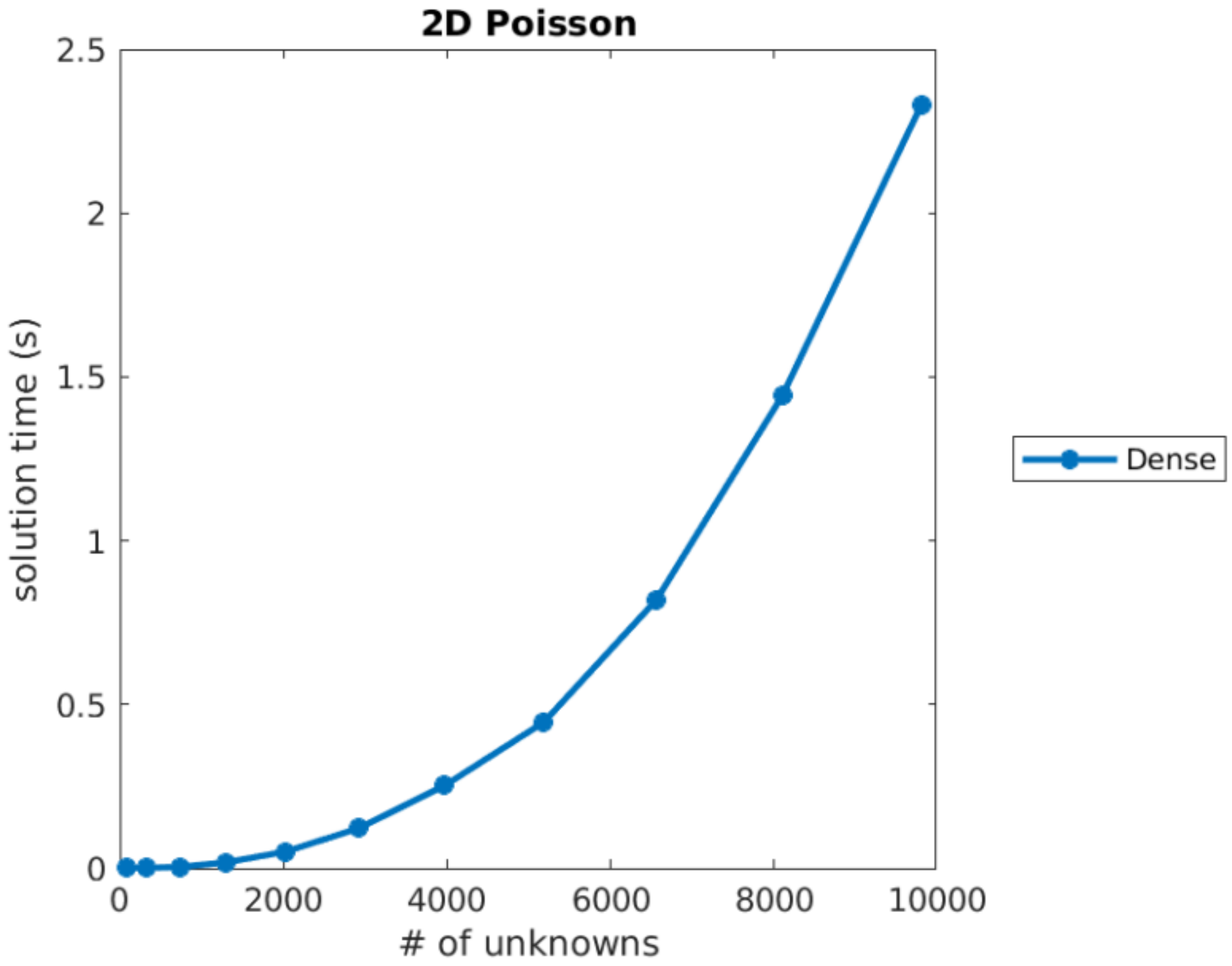
Linear Systems

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Linear Systems

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & \dots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & \dots & a_{3,n} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & \dots & a_{4,n} \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & \dots & a_{5,n} \\ & & \vdots & & & \vdots & \\ a_{n,1} & a_{n,2} & a_{n,3} & a_{n,4} & a_{n,5} & \dots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \vdots \\ b_n \end{bmatrix}$$

Linear System Solves



Sparse Linear Systems

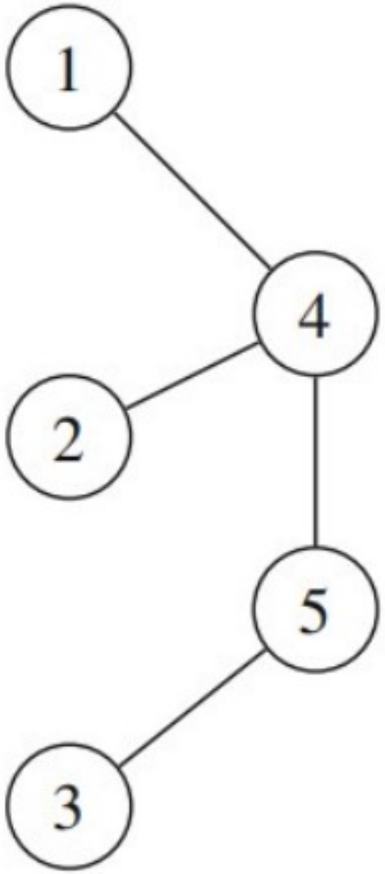
$$\begin{bmatrix} a_{1,1} & 0 & 0 & a_{1,4} & 0 & \dots & 0 \\ a_{2,1} & 0 & 0 & 0 & a_{2,5} & \dots & a_{2,n} \\ 0 & a_{3,2} & 0 & 0 & 0 & \dots & 0 \\ 0 & a_{4,2} & 0 & 0 & 0 & \dots & a_{4,n} \\ 0 & 0 & 0 & 0 & a_{5,5} & \dots & 0 \\ & & \vdots & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \vdots \\ b_n \end{bmatrix}$$

Sparse Linear Systems

$$
\begin{bmatrix}
a_{1,1} & & & a_{1,4} & & \cdots & \\
a_{2,1} & & & & a_{2,5} & \cdots & a_{2,n} \\
& a_{3,2} & & & & \cdots & \\
& a_{4,2} & & & & \cdots & a_{4,n} \\
& & & & a_{5,5} & \cdots & \\
& & \vdots & & & \vdots & \\
& & & & & \cdots & a_{n,n}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \vdots \\ b_n \end{bmatrix}
$$

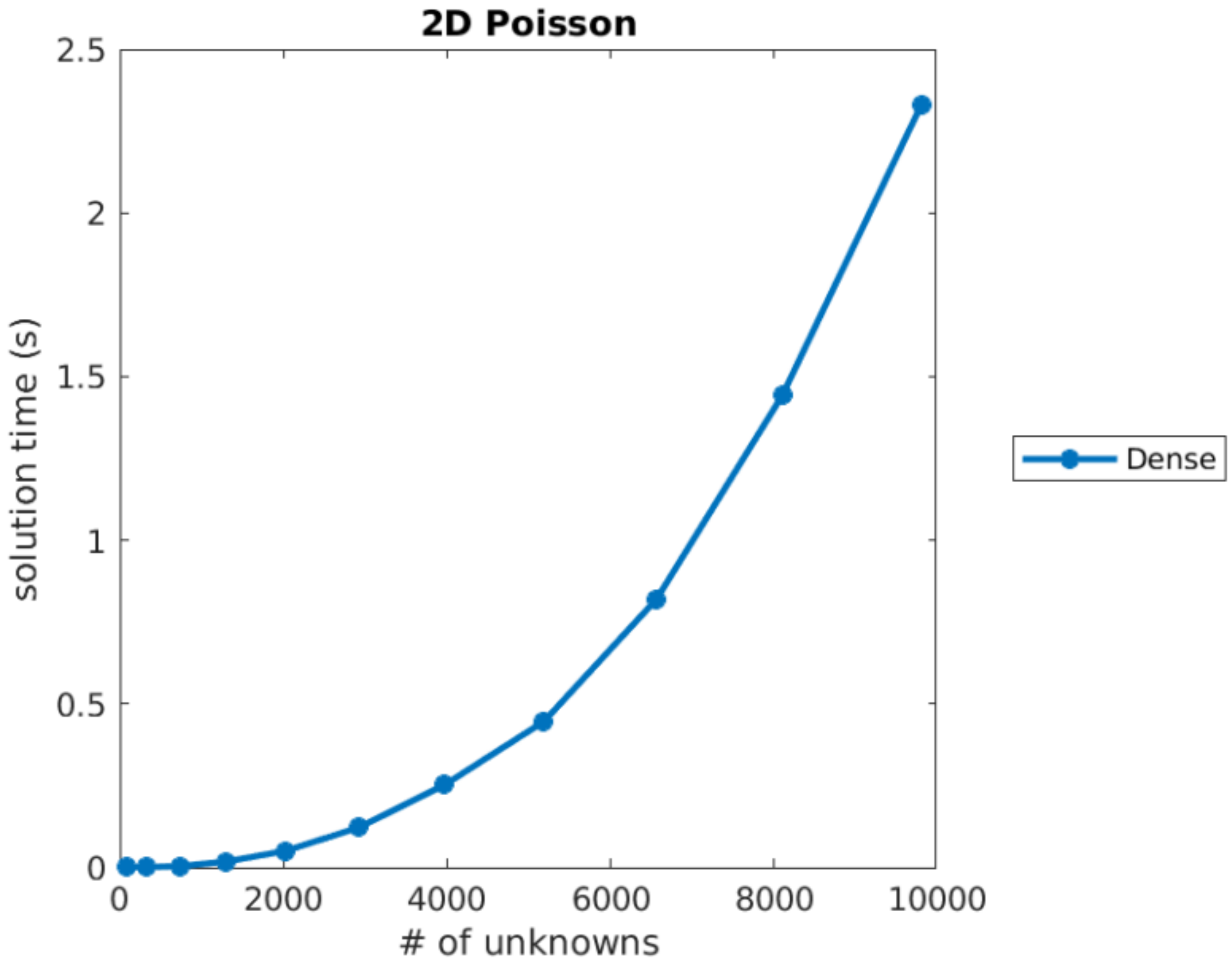
Sparse Matrices and Graphs

$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{pmatrix}
* & & & * & \\
 & * & & * & \\
 & & * & & * \\
* & * & & * & * \\
 & & * & * & *
\end{pmatrix}
\end{array}
$$

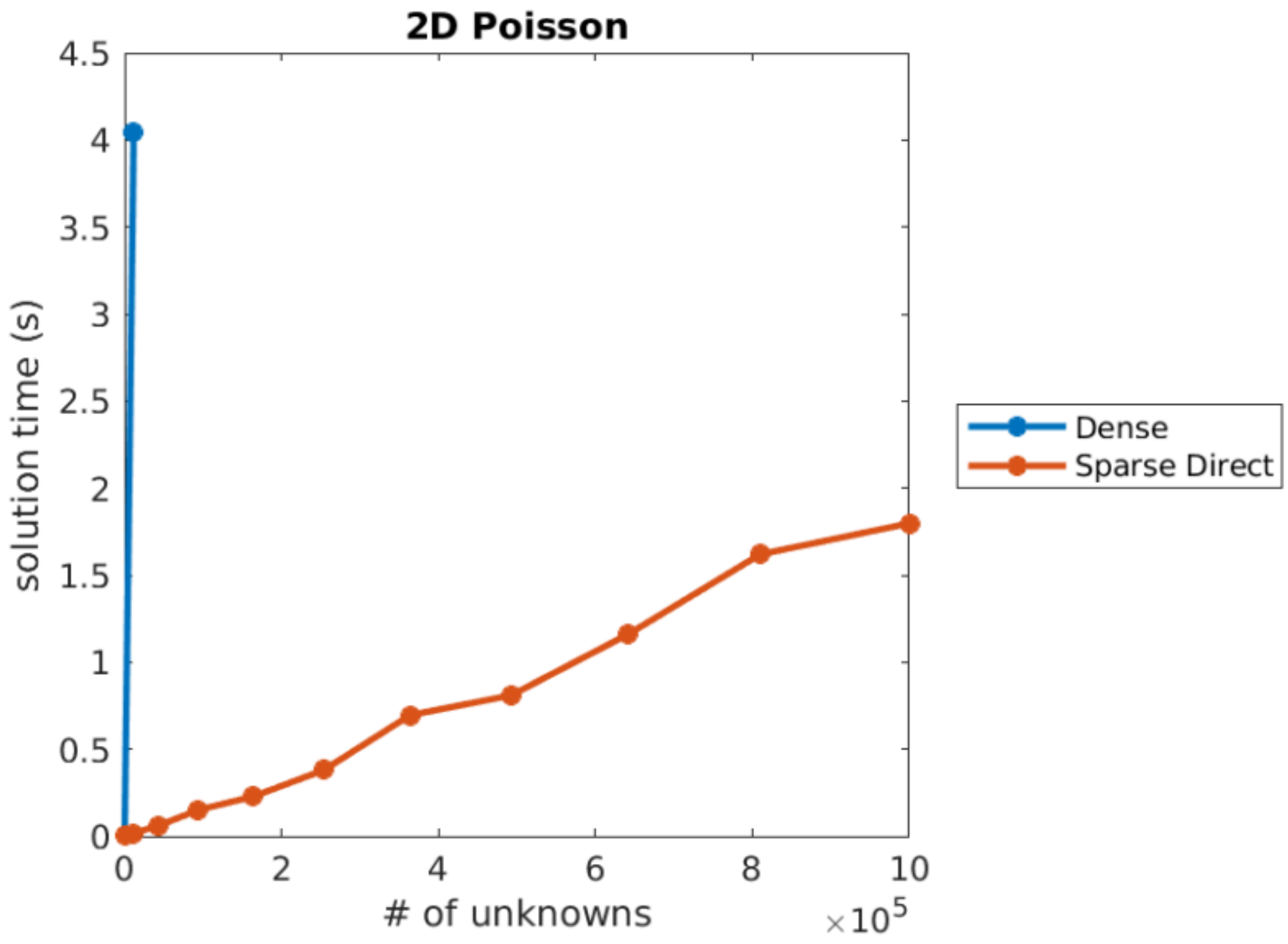


[Scott and Tůma, Algorithms for Sparse Linear Systems, 2023]

Linear System Solves



Linear System Solves



Software

LINEAR ALGERBA

MA: Solution of linear systems

MA38	Sparse unsymmetric system: unsymmetric multifrontal method
MA41	Sparse unsymmetric system: unsymmetric multifrontal method
MA42	Sparse unsymmetric system: out-of-core frontal method
HSL_MA42_ELEMENT	Unsymmetric finite-element system: out-of-core frontal method (real and complex)
HSL_MA42	Sparse unsymmetric system: out-of-core frontal method
MA43	Sparse unsymmetric system: row-by-row frontal method
MA44	Over-determined linear system: least-squares solution
MA46	Sparse unsymmetric finite-element system: multifrontal
MA48	Sparse unsymmetric system: driver for conventional direct method
HSL_MA48	Sparse unsymmetric system: driver for conventional direct method
MA49	Sparse over-determined system: least squares by QR
MA50	Sparse unsymmetric system: conventional direct method
MA51	Auxiliary for MA48 and MA50: identify ignored rows or columns in the rectangular or rank-deficient case, compute determinant
MA52	Sparse unsymmetric finite-element system: out-of-core multiple front method
HSL_MA54	Definite symmetric full matrix: partial or complete factorization and solution
HSL_MA55	Band symmetric positive-definite system
MA57	Sparse symmetric system: multifrontal method
HSL_MA57	Sparse symmetric system: multifrontal method
MA60	Iterative refinement and error estimation
MA61	Sparse symmetric positive-definite system: incomplete factorization
MA62	Sparse symmetric finite-element system: out-of-core frontal method
HSL_MA64	Indefinite symmetric full matrix: partial or complete factorization and solution
MA65	Unsymmetric banded system of linear equations
MA67	Sparse symmetric system, zeros on diagonal: blocked conventional
MA69	Unsymmetric system whose leading subsystem is easy to solve
HSL_MA69	Unsymmetric system whose leading subsystem is easy to solve
MA72	Sparse symmetric finite-element system: out-of-core multiple front method
HSL_MA74	Unsymmetric full matrix: partial or complete factorization and solution
MA75	Sparse over-determined system: weighted least squares
HSL_MA77	Sparse symmetric system: multifrontal out of core
HSL_MA78	Sparse unsymmetric finite-element system: multifrontal out of core
HSL_MA79	Sparse symmetric system: mixed precision
HSL_MA86	Sparse solver for real and complex indefinite matrices using OpenMP
HSL_MA87	Sparse Cholesky solver for real/complex matrices using OpenMP
HSL_MA97	Bit-compatible parallel sparse symmetric/Hermitian solver using OpenMP

Software

LINEAR ALGERBA

MA: Solution of linear systems

MA38	Sparse unsymmetric system: unsymmetric multifrontal method
MA41	Sparse unsymmetric system: unsymmetric multifrontal method
MA42	Sparse unsymmetric system: out-of-core frontal method
HSL_MA42_ELEMENT	Unsymmetric finite-element system: out-of-core frontal method (real and complex)
HSL_MA42	Sparse unsymmetric system: out-of-core frontal method
MA43	Sparse unsymmetric system: row-by-row frontal method
MA44	Over-determined linear system: least-squares solution
MA46	Sparse unsymmetric finite-element system: multifrontal
MA48	Sparse unsymmetric system: driver for conventional direct method
HSL_MA48	Sparse unsymmetric system: driver for conventional direct method
MA49	Sparse over-determined system: least squares by QR
MA50	Sparse unsymmetric system: conventional direct method
MA51	Auxiliary for MA48 and MA50: identify ignored rows or columns in the rectangular or rank-deficient case, compute determinant
MA52	Sparse unsymmetric finite-element system: out-of-core multiple front method
HSL_MA54	Definite symmetric full matrix: partial or complete factorization and solution
HSL_MA55	Band symmetric positive-definite system
MA57	Sparse symmetric system: multifrontal method
HSL_MA57	Sparse symmetric system: multifrontal method
MA60	Iterative refinement and error estimation
MA61	Sparse symmetric positive-definite system: incomplete factorization
MA62	Sparse symmetric finite-element system: out-of-core frontal method
HSL_MA64	Indefinite symmetric full matrix: partial or complete factorization and solution
MA65	Unsymmetric banded system of linear equations
MA67	Sparse symmetric system, zeros on diagonal: blocked conventional
MA69	Unsymmetric system whose leading subsystem is easy to solve
HSL_MA69	Unsymmetric system whose leading subsystem is easy to solve
MA72	Sparse symmetric finite-element system: out-of-core multiple front method
HSL_MA74	Unsymmetric full matrix: partial or complete factorization and solution
MA75	Sparse over-determined system: weighted least squares
HSL_MA77	Sparse symmetric system: multifrontal out of core
HSL_MA78	Sparse unsymmetric finite-element system: multifrontal out of core
HSL_MA79	Sparse symmetric system: mixed precision
HSL_MA86	Sparse solver for real and complex indefinite matrices using OpenMP
HSL_MA87	Sparse Cholesky solver for real/complex matrices using OpenMP
HSL_MA97	Bit-compatible parallel sparse symmetric/Hermitian solver using OpenMP

SSIDS - Sparse Symmetric Indefinite Direct Solver

```
#include <spral_ssids.h> /* or <spral.h> for all packages */
```

Purpose

This package solves one or more sets of $n \times n$ sparse **symmetric** equations $AX = B$ using a multifrontal method. The following cases are covered:

1. A is **indefinite**. SSIDS computes the sparse factorization

$$A = PLD(PL)^T$$

where P is a permutation matrix, L is unit lower triangular, and D is block diagonal with blocks of size 1×1 and 2×2.

2. A is **positive definite**. SSIDS computes the **sparse Cholesky factorization**

$$A = PL(PL)^T$$

where P is a permutation matrix and L is lower triangular. *However, as SSIDS is designed primarily for indefinite systems, this may be slower than a dedicated Cholesky solver.*

The code optionally supports hybrid computation using one or more NVIDIA GPUs.

SSIDS returns bit-compatible results.

An option exists to scale the matrix. In this case, the factorization of the scaled matrix $\overline{A} = SAS$ is computed, where S is a diagonal scaling matrix.

Usage overview

Solving $AX = B$ using SSIDS is a four stage process.

[HSL - Home Page (rl.ac.uk)](#)

Software

LINEAR ALGEBRA

MA: Solution of linear systems

MA38	Sparse unsymmetric system: unsymmetric multifrontal method
MA41	Sparse unsymmetric system: unsymmetric multifrontal method
MA42	Sparse unsymmetric system: out-of-core frontal method
HSL_MA42_ELEMENT	Unsymmetric finite-element system: out-of-core frontal method (real and complex)
HSL_MA42	Sparse unsymmetric system: out-of-core frontal method
MA43	Sparse unsymmetric system: row-by-row frontal method
MA44	Over-determined linear system: least-squares solution
MA46	Sparse unsymmetric finite-element system: multifrontal
MA48	Sparse unsymmetric system: driver for conventional direct method
HSL_MA48	Sparse unsymmetric system: driver for conventional direct method
MA49	Sparse over-determined system: least squares by QR
MA50	Sparse unsymmetric system: conventional direct method
MA51	Auxiliary for MA48 and MA50: identify ignored rows or columns in the rectangular or rank-deficient case, compute determinant
MA52	Sparse unsymmetric finite-element system: out-of-core multiple front method
HSL_MA54	Definite symmetric full matrix: partial or complete factorization and solution
HSL_MA55	Band symmetric positive-definite system
MA57	Sparse symmetric system: multifrontal method
HSL_MA57	Sparse symmetric system: multifrontal method
MA60	Iterative refinement and error estimation
MA61	Sparse symmetric positive-definite system: incomplete factorization
MA62	Sparse symmetric finite-element system: out-of-core frontal method
HSL_MA64	Indefinite symmetric full matrix: partial or complete factorization and solution
MA65	Unsymmetric banded system of linear equations
MA67	Sparse symmetric system, zeros on diagonal: blocked conventional
MA69	Unsymmetric system whose leading subsystem is easy to solve
HSL_MA69	Unsymmetric system whose leading subsystem is easy to solve
MA72	Sparse symmetric finite-element system: out-of-core multiple front method
HSL_MA74	Unsymmetric full matrix: partial or complete factorization and solution
MA75	Sparse over-determined system: weighted least squares
HSL_MA77	Sparse symmetric system: multifrontal out of core
HSL_MA78	Sparse unsymmetric finite-element system: multifrontal out of core
HSL_MA79	Sparse symmetric system: mixed precision
HSL_MA86	Sparse solver for real and complex indefinite matrices using OpenMP
HSL_MA87	Sparse Cholesky solver for real/complex matrices using OpenMP
HSL_MA97	Bit-compatible parallel sparse symmetric/Hermitian solver using OpenMP


SSIDS - Sparse Symmetric Indefinite Direct Solver

SSIDS - Sparse Symmetric Indefinite Direct Solver

```
#include <spral_ssids.h> /* or <spral.h> for all packages */
```

Purpose

This package solves one or more sets of $n \times n$ sparse **symmetric** equations $AX = B$ using a multifrontal method. The following cases are covered:

1. A is **indefinite**. SSIDS computes the sparse factorization

$$A = PLD(PL)^T$$

where P is a permutation matrix, L is unit lower triangular, and D is block diagonal with blocks of size 1×1 and 2×2.

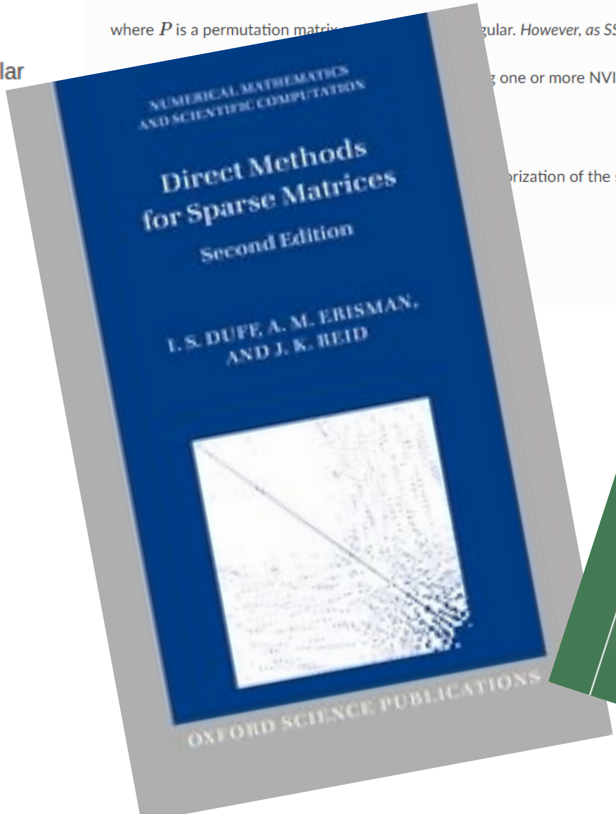
2. A is **positive definite**. SSIDS computes the **sparse Cholesky factorization**

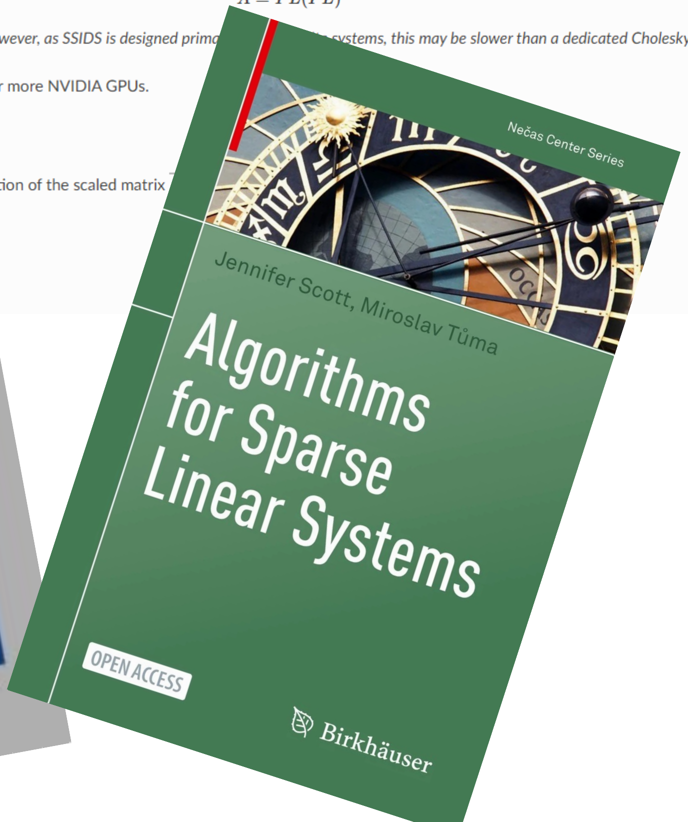
$$A = PL(PL)^T$$

where P is a permutation matri… …gular. *However, as SSIDS is designed prima… …systems, this may be slower than a dedicated Cholesky solver.*

… one or more NVIDIA GPUs.

…rization of the scaled matrix





Software

Exploring Benefits of Linear Solver Parallelism on Modern Nonlinear Optimization Applications

Byron Tasseff[1,2], Carleton Coffrin[1], Andreas Wächter[3], and Carl Laird[4]

[1] Los Alamos National Laboratory, Los Alamos, New Mexico, USA
[2] University of Michigan, Ann Arbor, Michigan, USA
[3] Northwestern University, Evanston, Illinois, USA
[4] Sandia National Laboratories, Albuquerque, New Mexico, USA

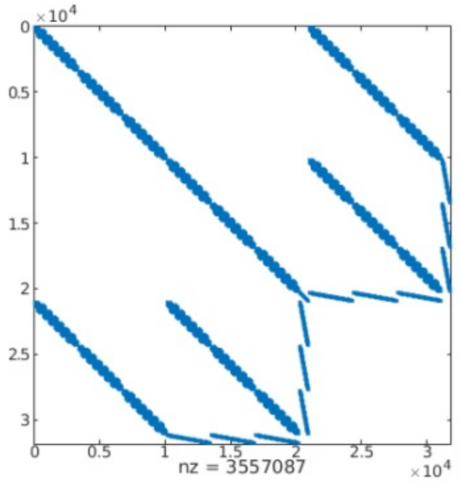
Solver	Method	Parallel CPU	Parallel GPU	License Restrictions
MA27	Multifrontal	No	No	Free to all Redistribution prohibited
MA57	Multifrontal	Threaded BLAS	No	Free to academics
HSL_MA77	Multifrontal	Limited	No	Free to academics
HSL_MA86	Supernodal	Multi-core Threaded BLAS	No	Free to academics
HSL_MA97	Multifrontal	Multi-core Threaded BLAS	No	Free to academics
MUMPS	Multifrontal	Multi-core Threaded BLAS	No	Free to all
PARDISO	Supernodal	Multi-core	No	Academic/corporate license
SPRAL	Multifrontal	Multi-core Threaded BLAS	Yes	Free to all
WSMP	Multifrontal	Multi-core	No	Trial/purchased license

Table 1: Summary of linear solvers available within IPOPT, including their factorization methodologies, parallel capabilities, and license restrictions.

Problem Type	Best Licensed Solver	Best Free Solver
Easy	MA57	MA27
Difficult	PARDISO	SPRAL

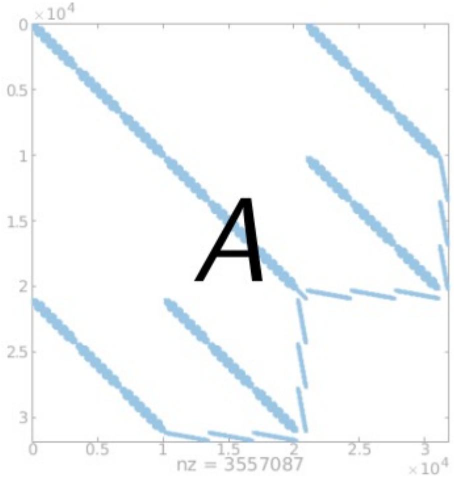
Table 3: Summary of suggested linear solvers within IPOPT based on problem type and solver license, inferred from computational experiments in Section 5.

Solving linear systems



$$\mathbf{x} = \mathbf{b}$$

Solving linear systems

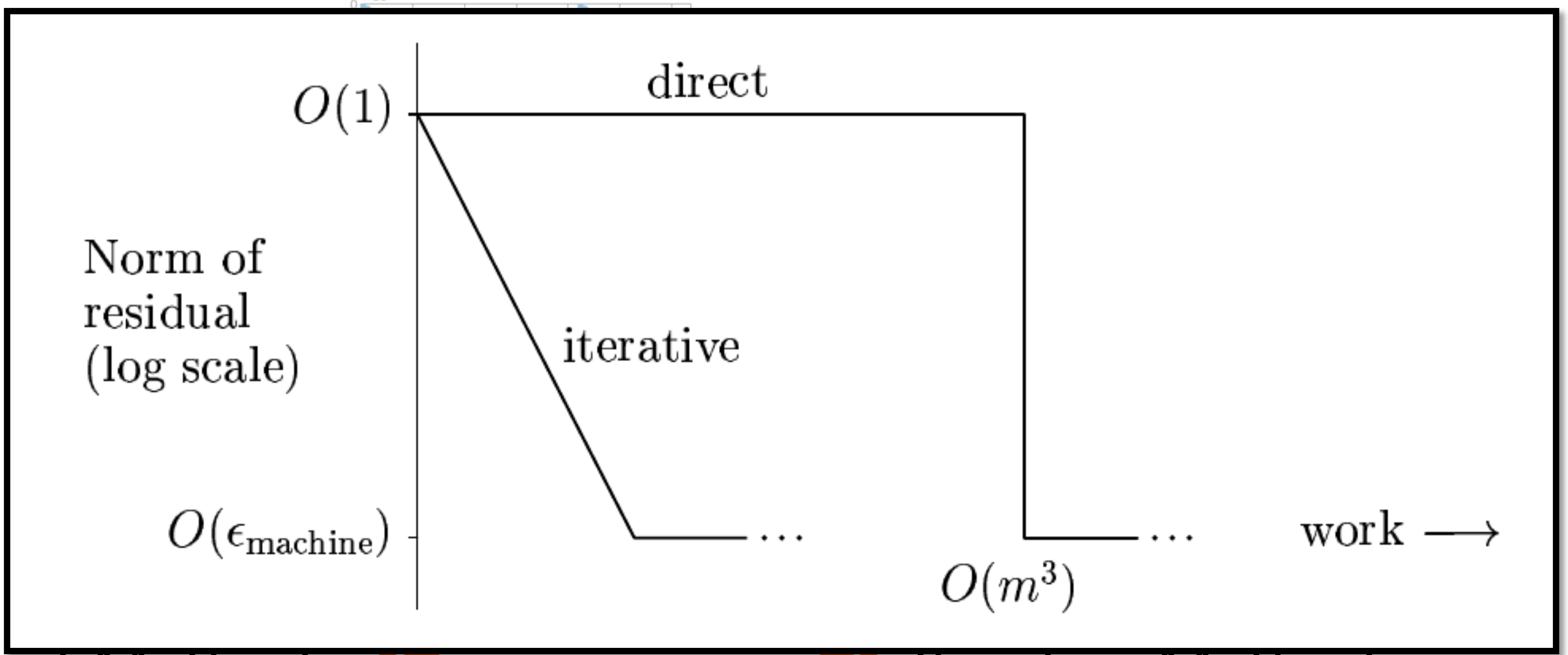


$$A \mathbf{x} = \mathbf{b}$$

Direct Methods

Iterative Methods

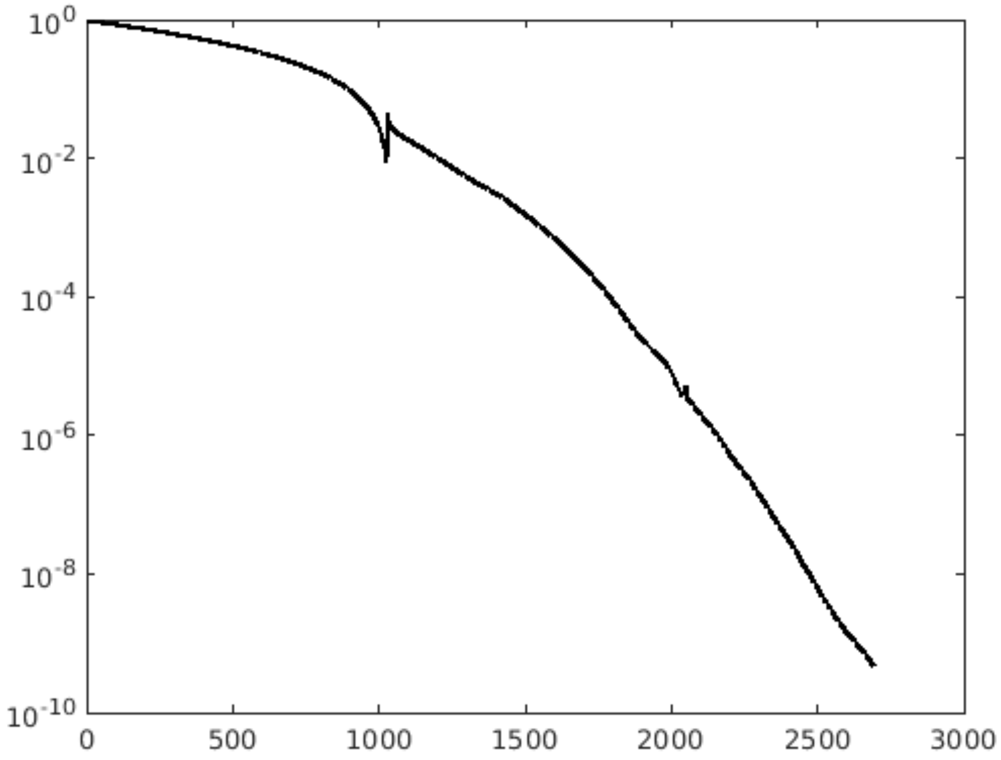
Solving linear systems



Direct Methods

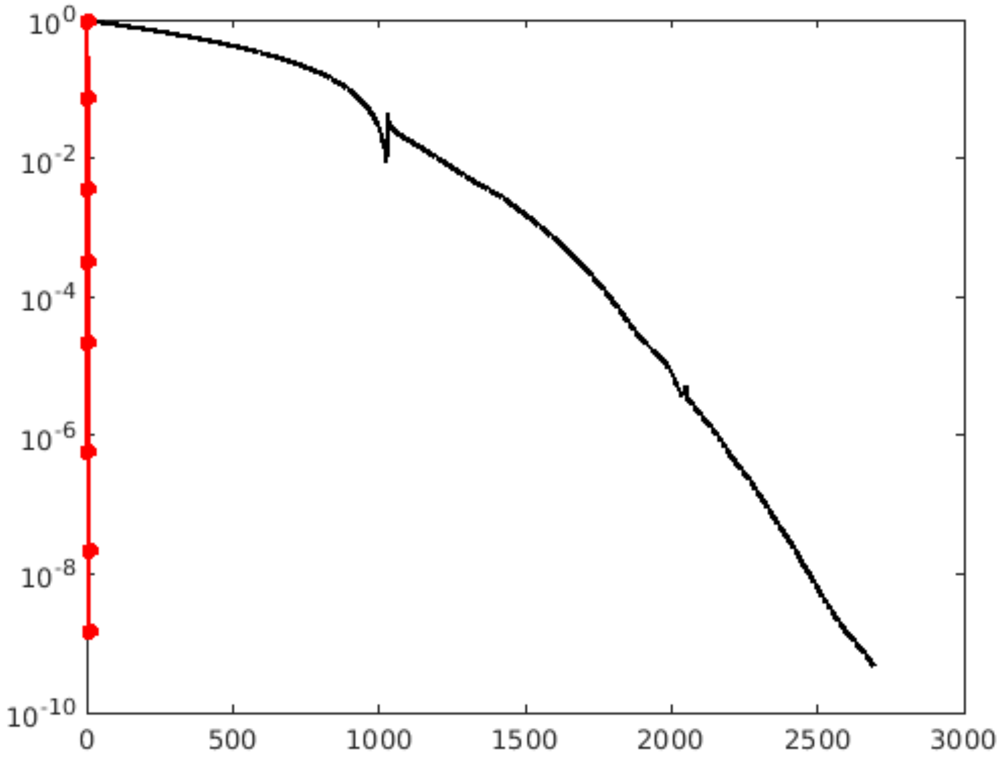
Iterative Methods

Iterative methods



$$A\mathbf{x} = \mathbf{b}$$

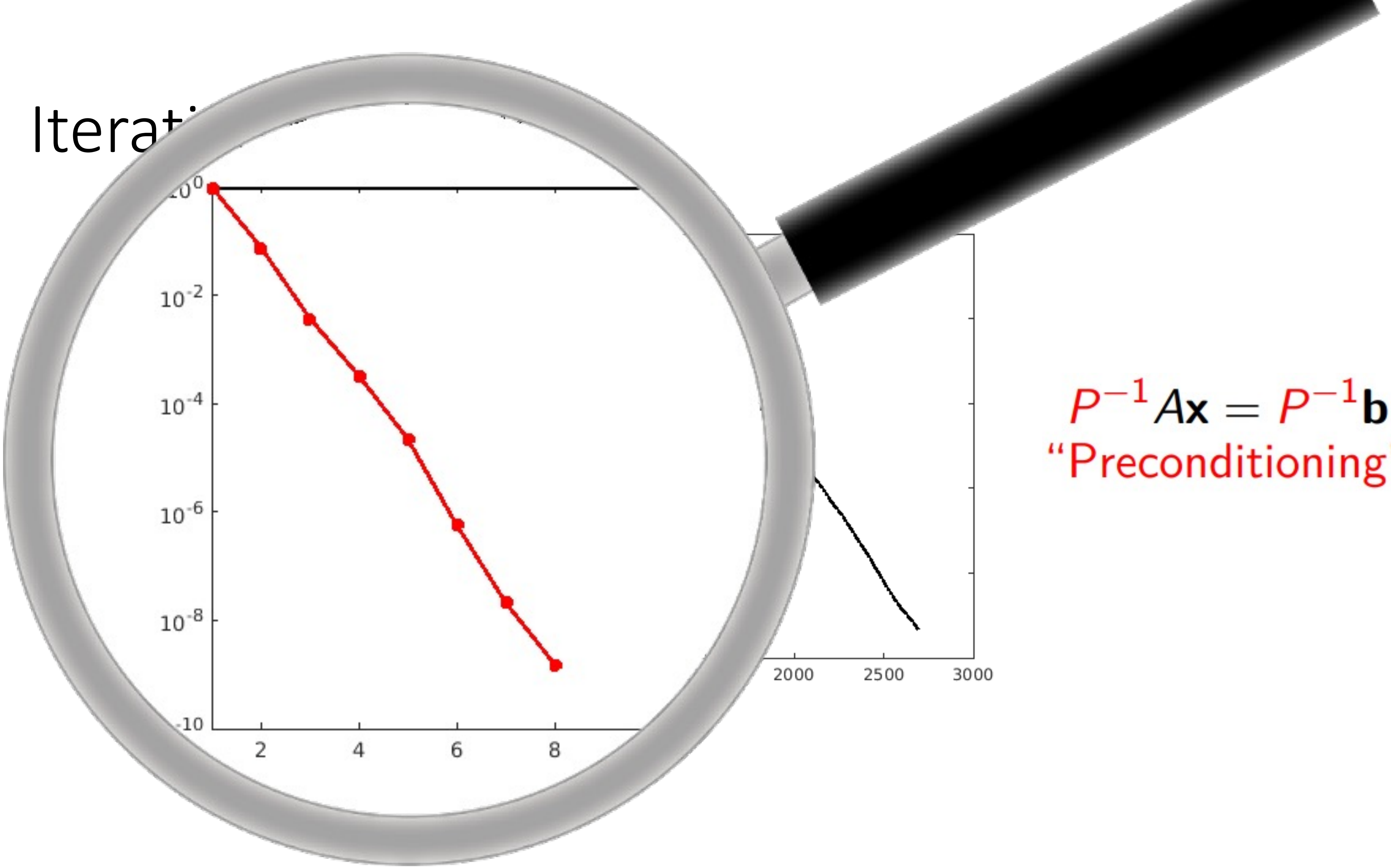
Iterative methods



$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$$

"Preconditioning"

Iterati



$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$$

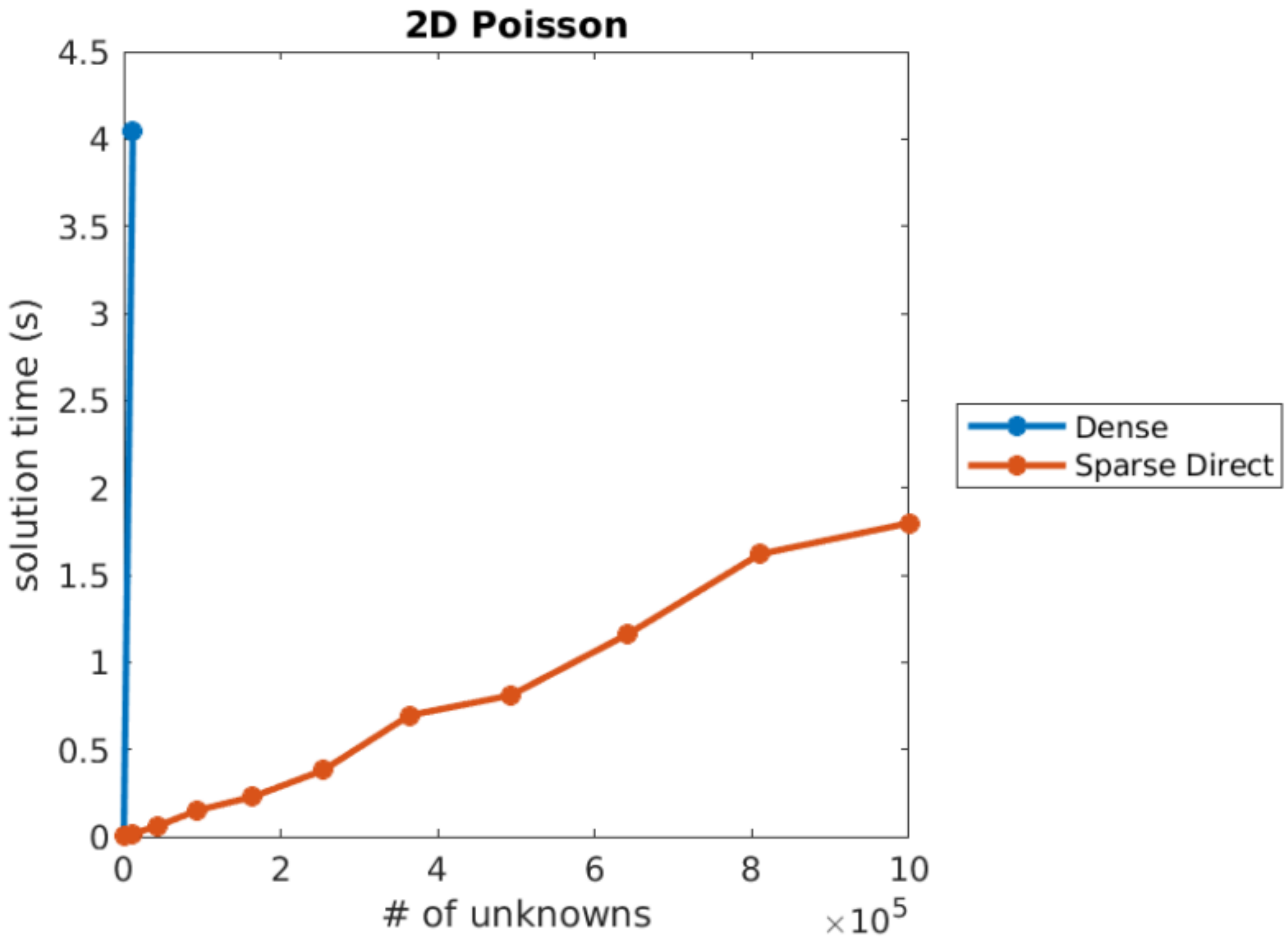
"Preconditioning"

Software

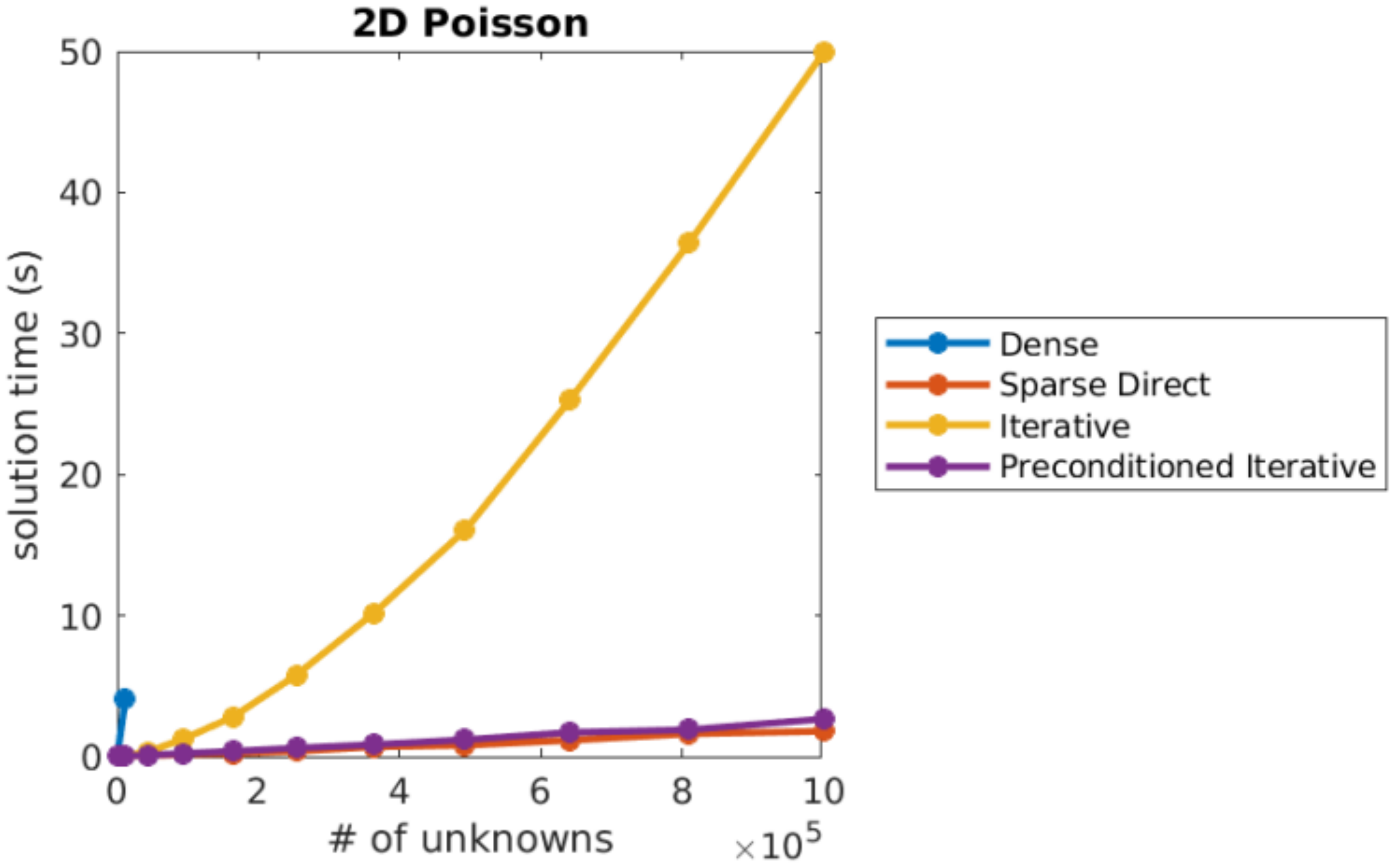
MI: Iterative methods for sparse matrices

HSL_MI02	Symmetric possibly-indefinite system: SYMMBK method
MI11	Unsymmetric system: incomplete LU factorization
MI12	Unsymmetric system: approximate-inverse preconditioner
HSL_MI13	Preconditioners for saddle-point systems
MI15	Unsymmetric system: flexible GMRES
HSL_MI20	Unsymmetric system: algebraic multigrid preconditioner
MI21	Symmetric positive-definite system: conjugate gradient method
MI23	Unsymmetric system: CGS (conjugate gradient squared) method
MI24	Unsymmetric system: GMRES (generalized minimal residual) method
MI25	Unsymmetric system: BiCG (BiConjugate Gradient) method
MI26	Unsymmetric system: BiCGStab (BiConjugate Gradient Stabilized) method
HSL_MI27	Projected preconditioned conjugate gradient method for saddle-point systems
HSL_MI28	Symmetric system: incomplete Cholesky factorization
HSL_MI29	MPGMRES: an extension of GMRES which allows multiple preconditioners
HSL_MI30	Symmetric indefinite saddle-point system: signed incomplete Cholesky factorization
HSL_MI31	Symmetric positive-definite system: conjugate gradient method, stopping according to the A-norm of the error
HSL_MI32	Symmetric possibly-indefinite system: MINRES method
HSL_MI35	Sparse least squares: incomplete factorization preconditioner

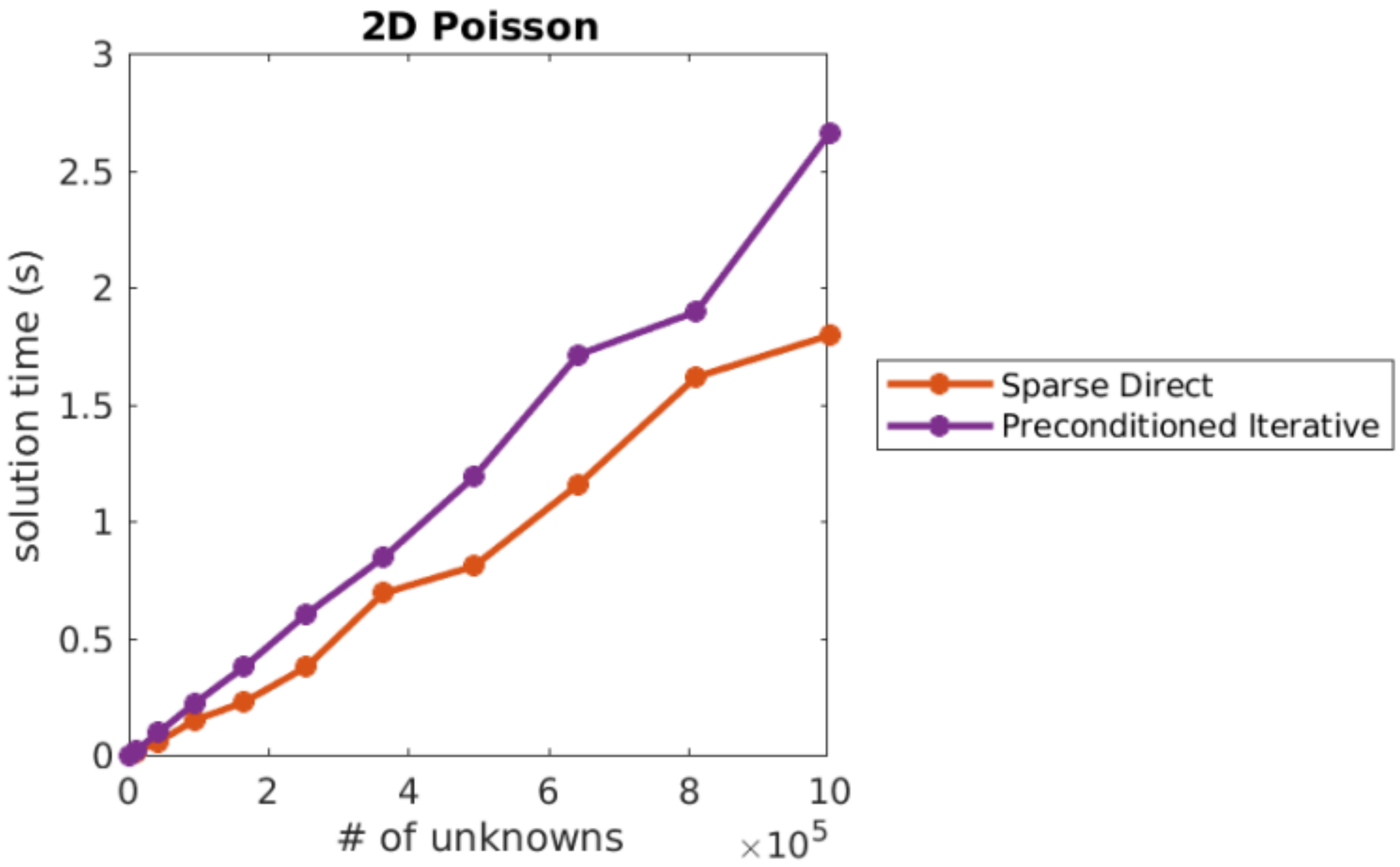
Linear System Solves



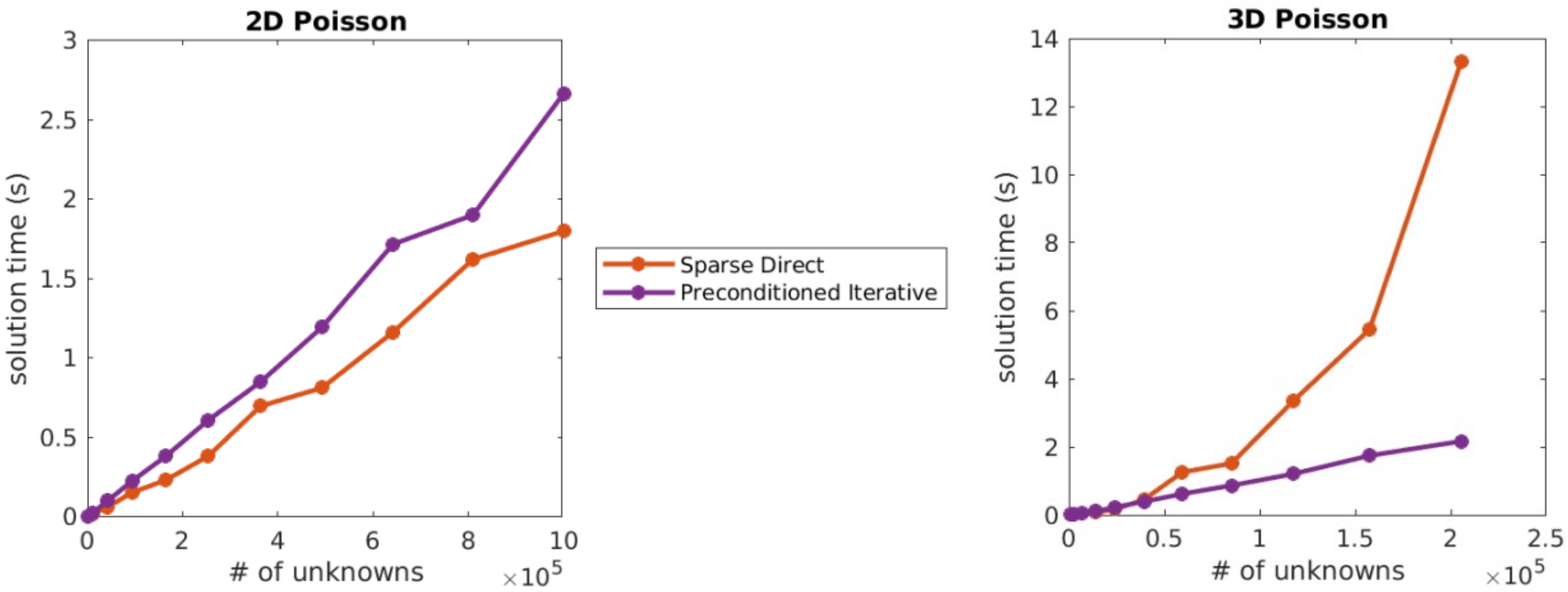
Linear System Solves



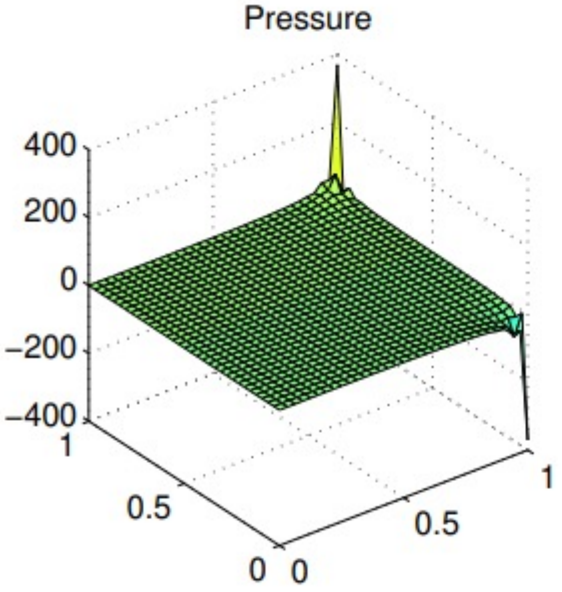
Linear System Solves

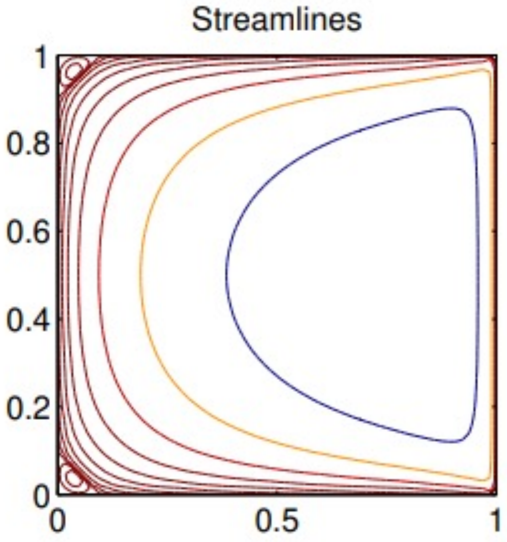


Linear System Solves

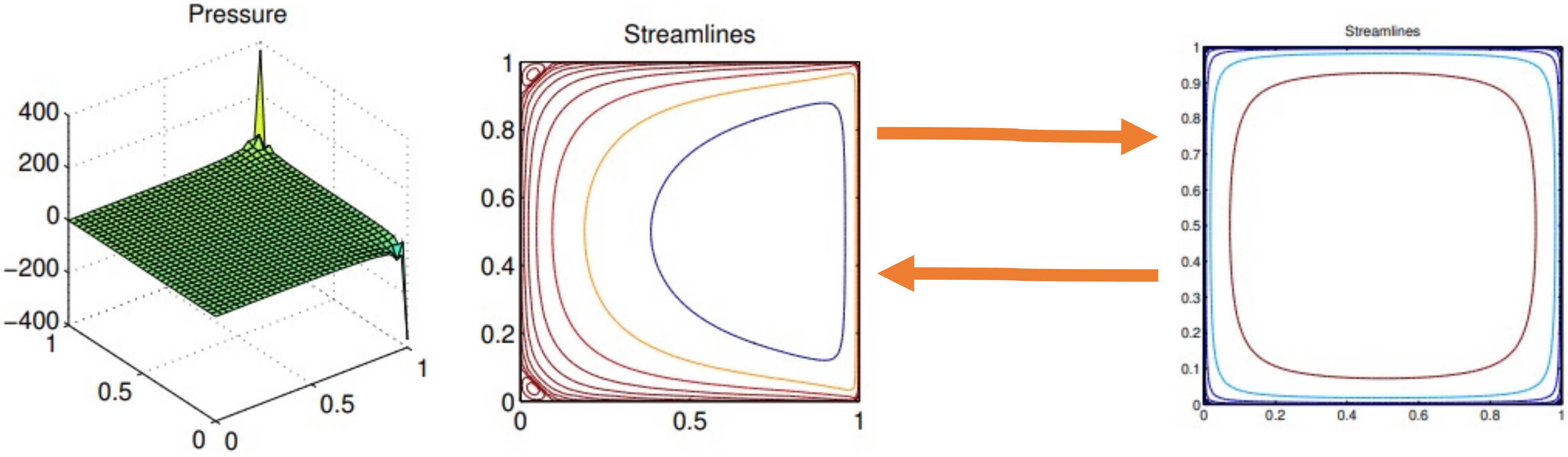


Flow control

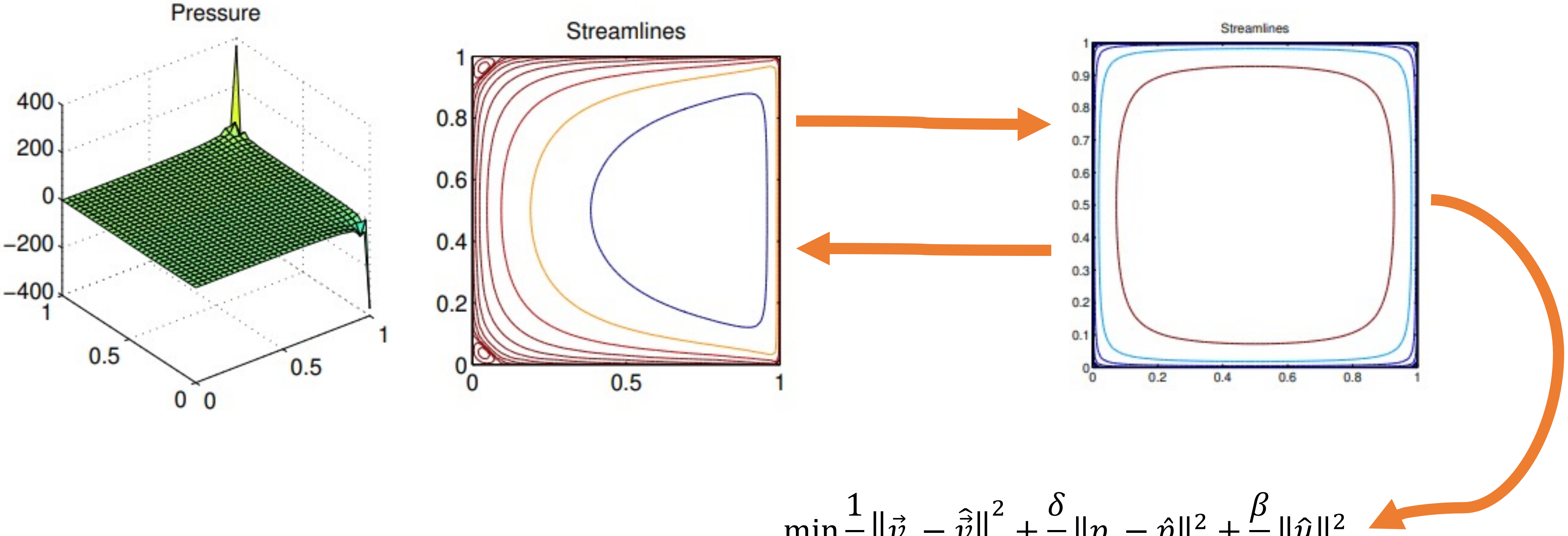




Flow control



Flow control



$$\min_{v,p,u} \frac{1}{2}\left\|\vec{v} - \hat{\vec{v}}\right\|^2 + \frac{\delta}{2}\|p - \hat{p}\|^2 + \frac{\beta}{2}\|\hat{u}\|^2$$

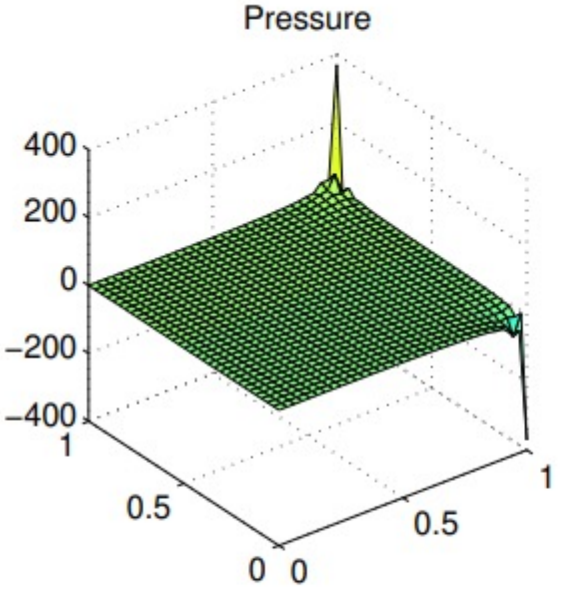
$$s.t. -\nabla^2 \vec{v} + \nabla p = \vec{u} \in \Omega$$

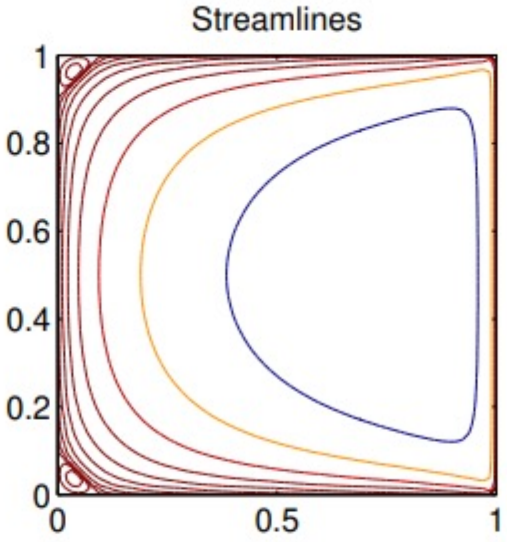
$$\nabla \cdot \vec{v} = 0 \in \Omega$$

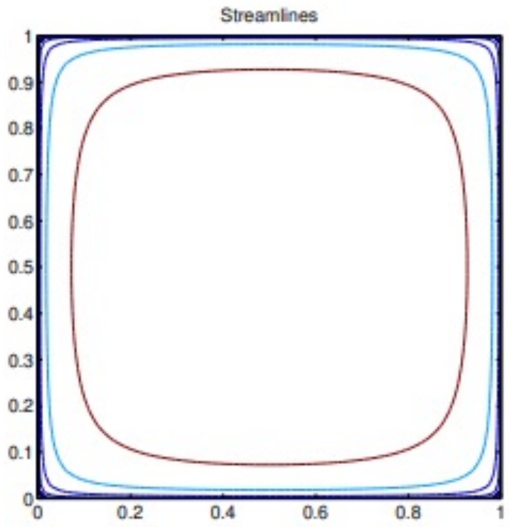
$$\vec{y} = \vec{w} \text{ on } \partial\Omega$$

$$v_l \le \vec{v} \le v_u, \qquad u_l \le \vec{u} \le u_u$$

Flow control







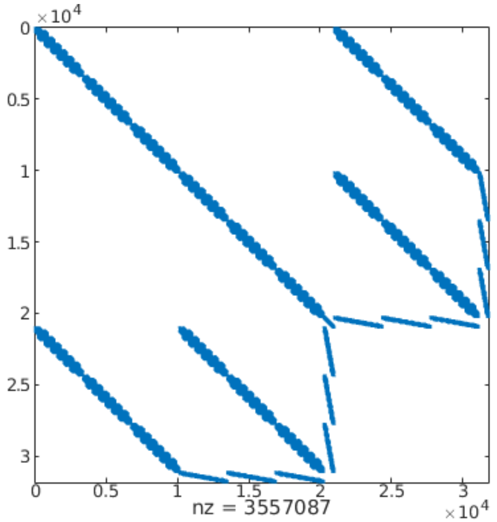
$$\min_{v,p,u} \frac{1}{2}\left\|\vec{v} - \hat{\vec{v}}\right\|^2 + \frac{\delta}{2}\|p - \hat{p}\|^2 + \frac{\beta}{2}\|\hat{u}\|^2$$

$$s.t. -\nabla^2 \vec{v} + \nabla p = \vec{u} \in \Omega$$

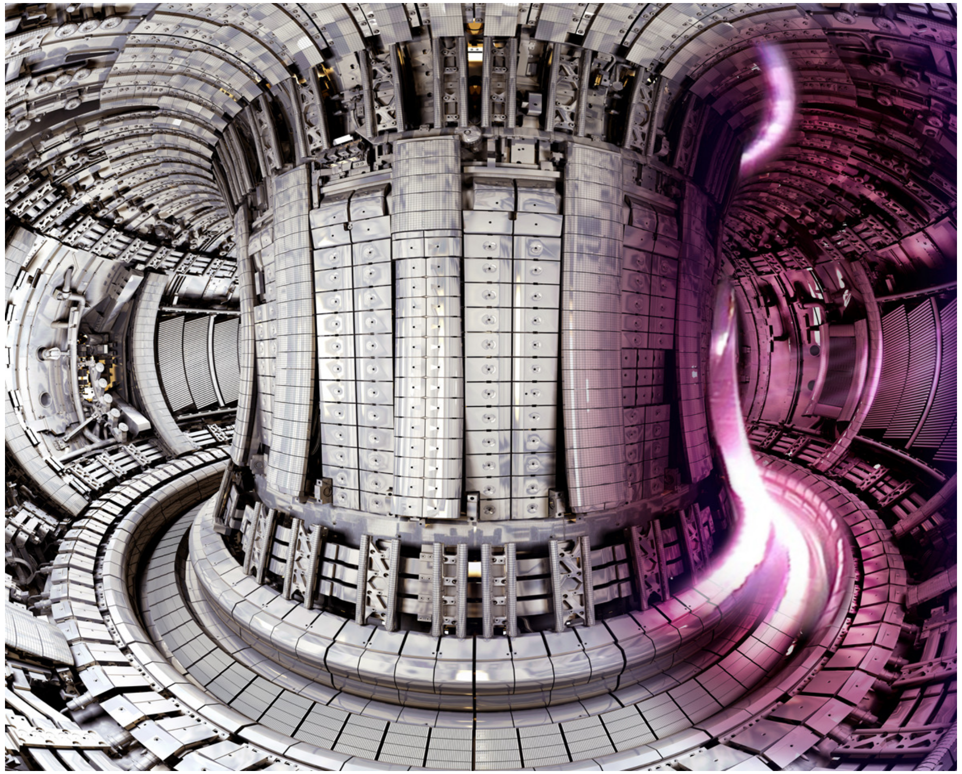
$$\nabla \cdot \vec{v} = 0 \in \Omega$$

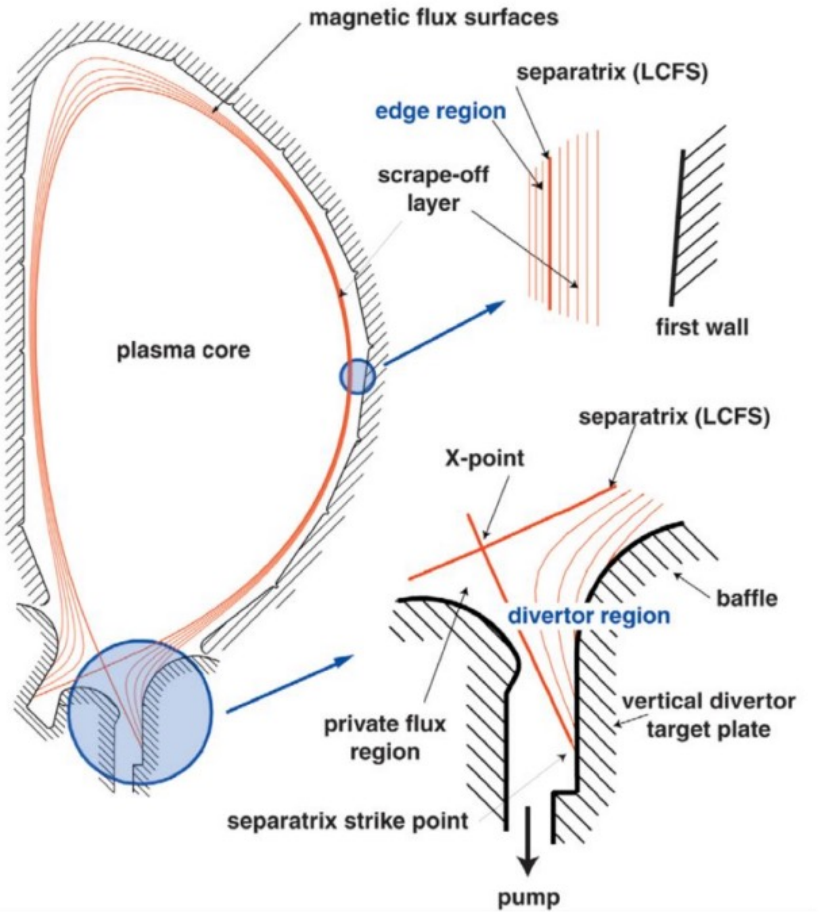
$$\vec{y} = \vec{w} \text{ on } \partial\Omega$$

$$v_l \leq \vec{v} \leq v_u, \qquad u_l \leq \vec{u} \leq u_u$$



NEPTUNE (NEutrals and Plasma TUrbulence Numerics)



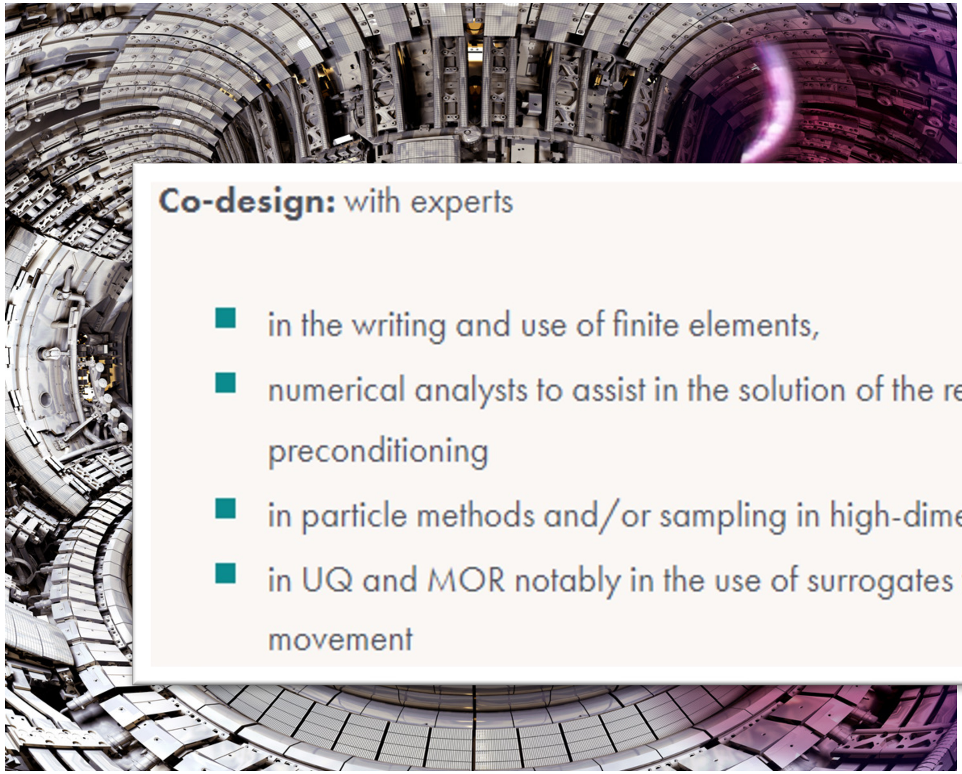


CCFE CULHAM CENTRE FOR FUSION ENERGY

ExCALIBUR

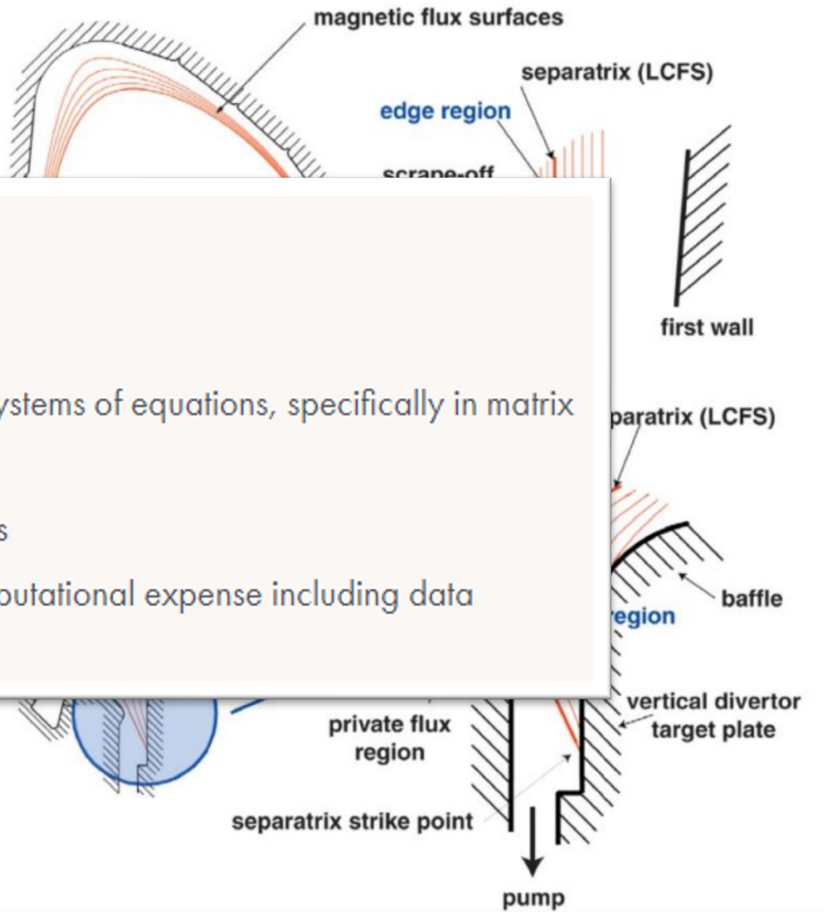
UKRI Science and Technology Facilities Council

NEPTUNE (NEutrals and Plasma TUrbulence Numerics)



Co-design: with experts

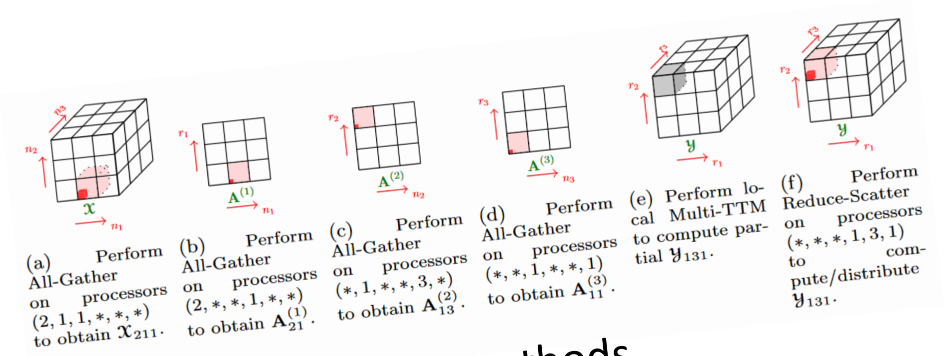
- in the writing and use of finite elements,
- numerical analysts to assist in the solution of the resulting large systems of equations, specifically in matrix preconditioning
- in particle methods and/or sampling in high-dimensional spaces
- in UQ and MOR notably in the use of surrogates to reduce computational expense including data movement



CCFE CULHAM CENTRE FOR FUSION ENERGY ExCALIBUR UKRI Science and Technology Facilities Council

https://excalibur.ac.uk/themes/high-priority-use-cases/

Other Linear Algebra…



Tensor methods

$$\min_x \|Ax - b\|_2^2$$

Linear least-squares

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{B}\mathbf{v}$$

Eigenvalue Problems

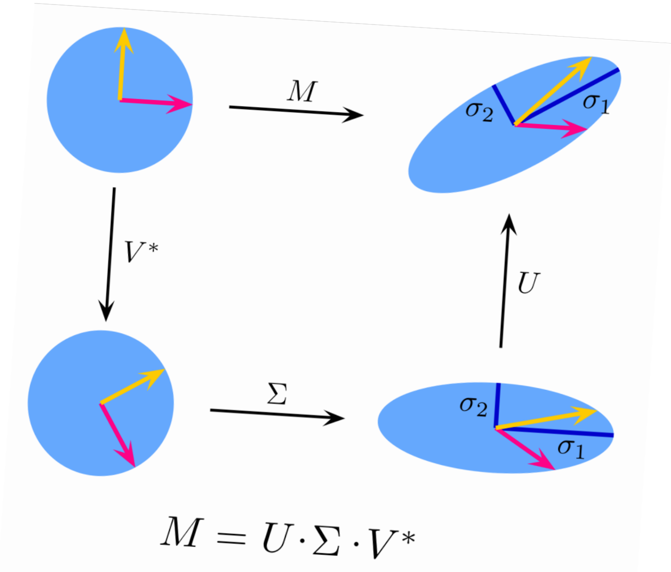
$$f(A)\boldsymbol{b}$$

Functions of Matrices

$$E\dot{x}(t) = Ax(t) + Bu(t),$$

$$y(t) = Cx(t) + Du(t),$$

Model Order Reduction



Singular Value Decomposition/
Principal Component Analysis

Science and Technology Facilities Council

Continuous Optimization

Linear Programming

$$\min_{\mathbf{x}} \Big(x_1$$  $$+ x_2$$  $$+ x_3$$  $$\Big)$$

Linear Programming

$\min_{\mathbf{x}} \Big(x_1$  $+ x_2$  $+ x_3$  $\Big)$

$$
\begin{aligned}
s.t. \quad 6x_1 + 3x_2 + 5x_3 &> 70 \\
x_1 &> 8 \\
x_2 &< 3
\end{aligned}
$$

Linear Programming

$\min_{\mathbf{x}} \Big($ x_1  $+ x_2$  $+ x_3$  $\Big)$

$$
\begin{aligned}
s.t. \quad 6x_1 + 3x_2 + 5x_3 &> 70 \\
x_1 &> 8 \\
{\color{red} -x_2} &{\color{red} > -3}
\end{aligned}
$$

Linear Programming

$\min_{\mathbf{x}} \Bigg(x_1$  $+ x_2$  $+ x_3$  $\Bigg)$

$$\text{s.t.} \quad \begin{bmatrix} 6 & 3 & 5 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} > \begin{bmatrix} 70 \\ 8 \\ -3 \end{bmatrix}$$

Linear Programming

$\min_{\mathbf{x}} \Big(x_1$  $+ x_2$  $+ x_3$  $\Big)$

s.t. $B \, \mathbf{x} \geq \mathbf{g}$

Most work: solve a series $\begin{bmatrix} D & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$

Nonlinear Programming

$$\min f(x)$$

$$s.t.\ Ax > b$$

Software

Camelot

News

Download GALAHAD

Documentation

Contribute

Bugs

Links

Authors of GALAHAD

Welcome!

GALAHAD is a thread-safe library of Fortran 2003 packages for solving nonlinear optimization problems. At present, the areas covered by the library are unconstrained and bound-constrained optimization, quadratic programming, nonlinear programming, systems of nonlinear equations and inequalities, and nonlinear least squares problems. GALAHAD contains the following major packages:

LANCELOT B

is the latest release of LANCELOT, and is designed to solve large-scale optimization problems involving the minimization of a nonlinear objective, subject (perhaps) to linear or nonlinear equality and box constraints. All functions involved are assumed to be group partially separable, and the minimization is based on a Sequential Augmented Lagrangian algorithm. The new release is coded in Fortran 95, allows for a non-monotone descent strategy, Moré and Toraldo-type projections, optional use of Lin and Moré's ICFS preconditioner, structured trust regions, and more.

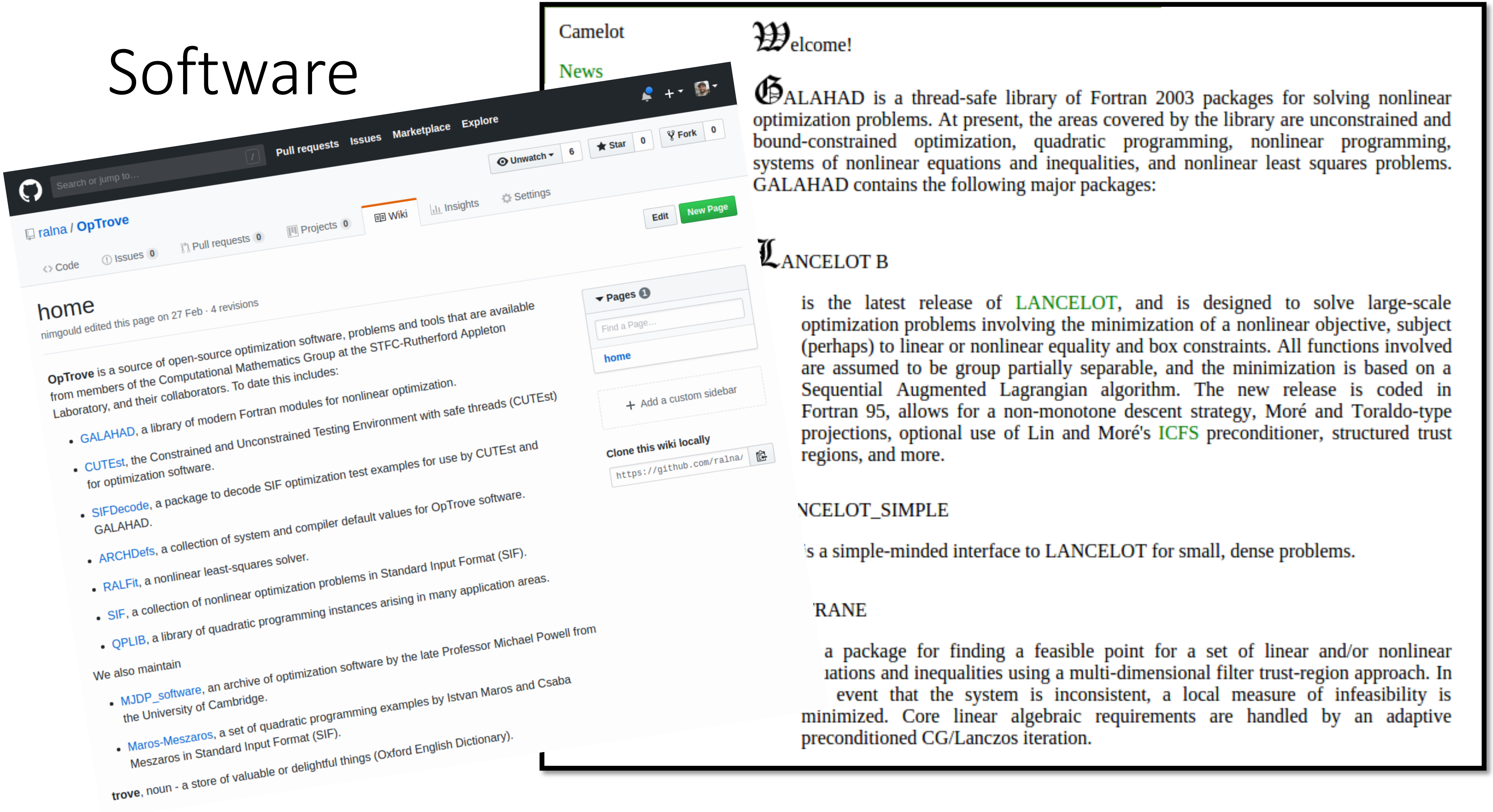
LANCELOT_SIMPLE

is a simple-minded interface to LANCELOT for small, dense problems.

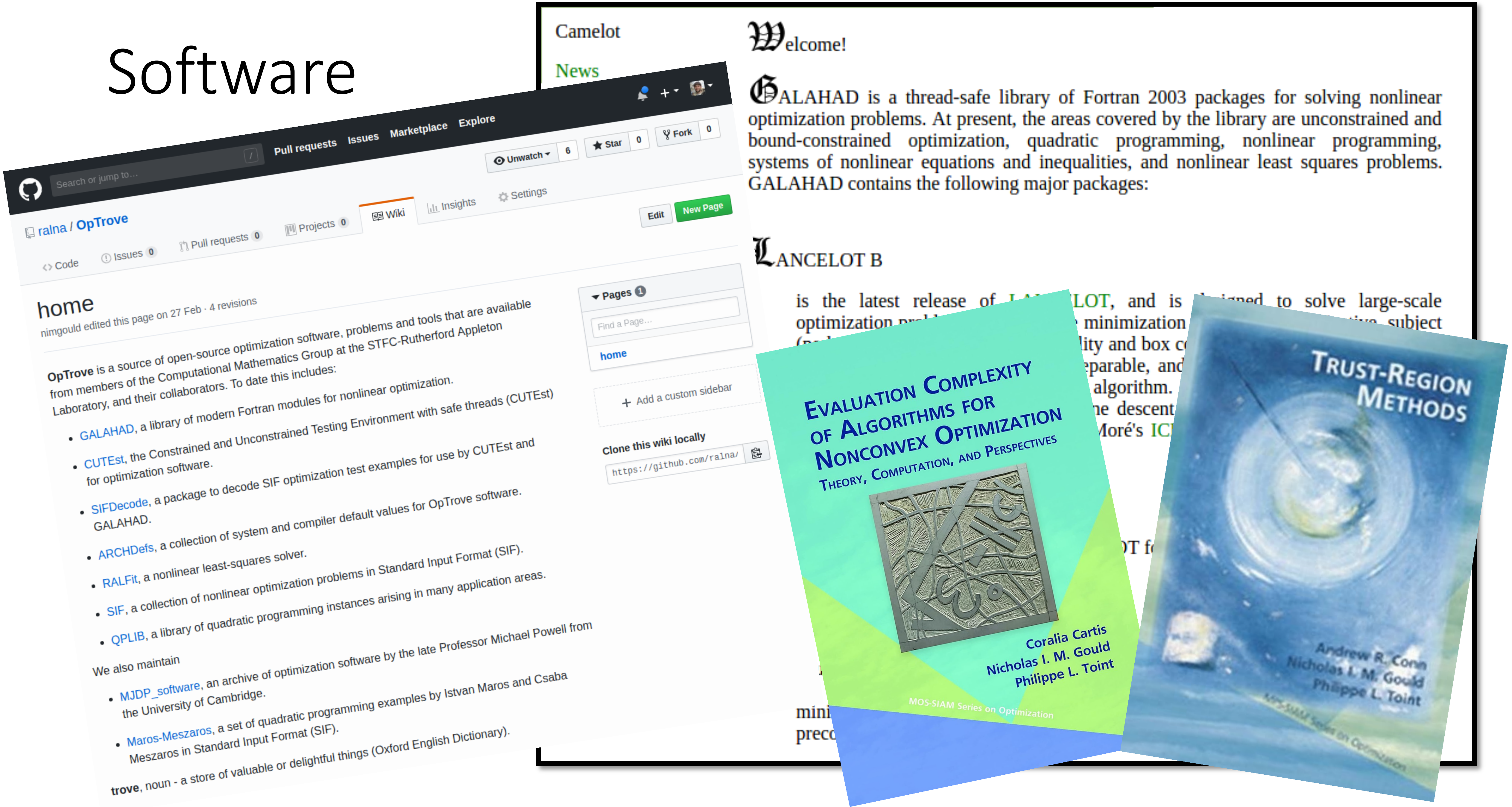
FILTRANE

is a package for finding a feasible point for a set of linear and/or nonlinear equations and inequalities using a multi-dimensional filter trust-region approach. In the event that the system is inconsistent, a local measure of infeasibility is minimized. Core linear algebraic requirements are handled by an adaptive preconditioned CG/Lanczos iteration.

Software

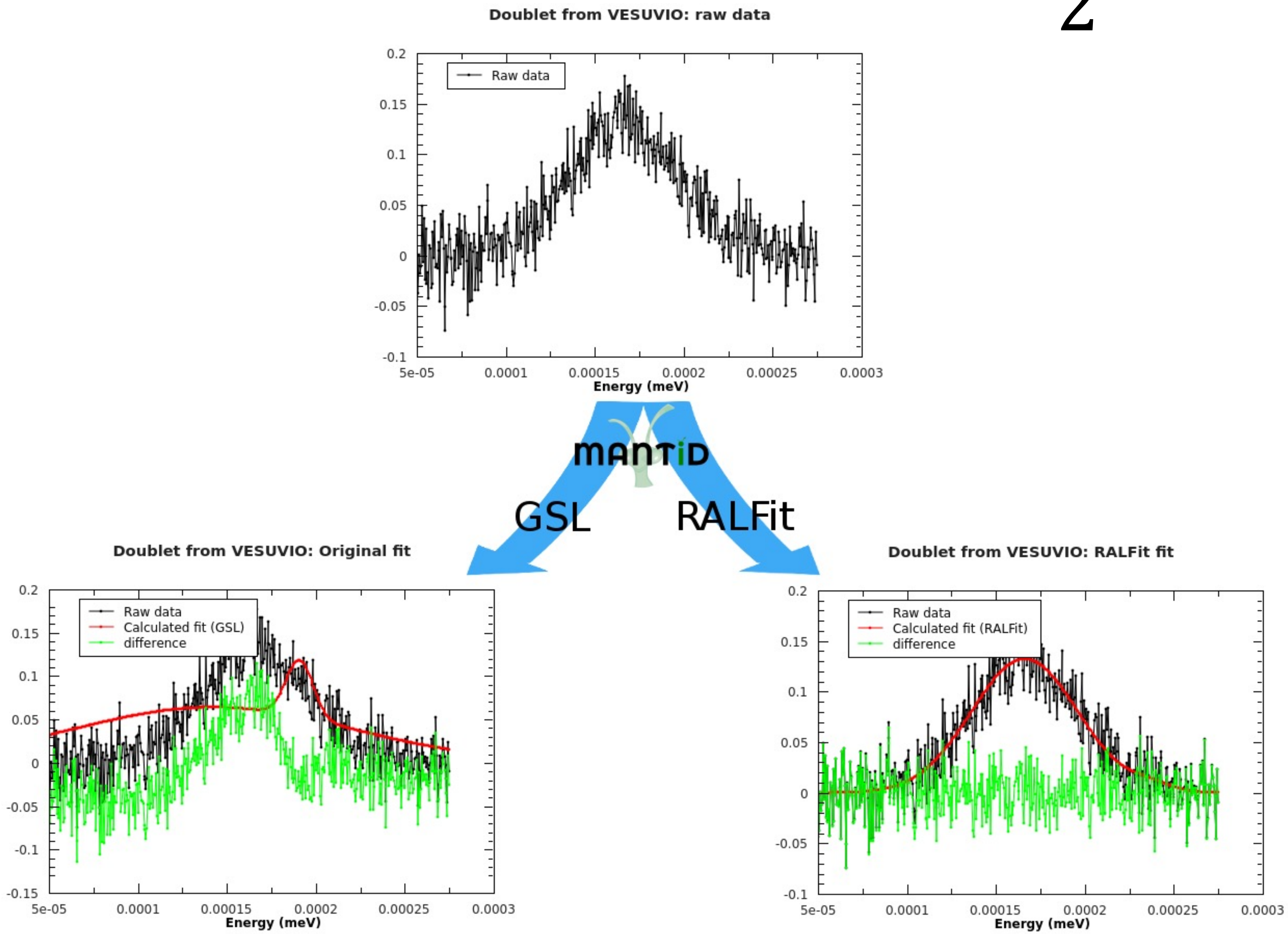

Search or jump to…
Pull requests
Issues
Marketplace
Explore
Unwatch 6
Star 0
Fork 0
ralna / OpTrove
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Edit
New Page
home
nimgould edited this page on 27 Feb · 4 revisions
OpTrove is a source of open-source optimization software, problems and tools that are available from members of the Computational Mathematics Group at the STFC-Rutherford Appleton Laboratory, and their collaborators. To date this includes:
GALAHAD, a library of modern Fortran modules for nonlinear optimization.
CUTEst, the Constrained and Unconstrained Testing Environment with safe threads (CUTEst) for optimization software.
SIFDecode, a package to decode SIF optimization test examples for use by CUTEst and GALAHAD.
ARCHDefs, a collection of system and compiler default values for OpTrove software.
RALFit, a nonlinear least-squares solver.
SIF, a collection of nonlinear optimization problems in Standard Input Format (SIF).
QPLIB, a library of quadratic programming instances arising in many application areas.
We also maintain
MJDP_software, an archive of optimization software by the late Professor Michael Powell from the University of Cambridge.
Maros-Meszaros, a set of quadratic programming examples by Istvan Maros and Csaba Meszaros in Standard Input Format (SIF).
trove, noun - a store of valuable or delightful things (Oxford English Dictionary).
Pages 1
Find a Page…
home
Add a custom sidebar
Clone this wiki locally
https://github.com/ralna/
Camelot
News
Welcome!
GALAHAD is a thread-safe library of Fortran 2003 packages for solving nonlinear optimization problems. At present, the areas covered by the library are unconstrained and bound-constrained optimization, quadratic programming, nonlinear programming, systems of nonlinear equations and inequalities, and nonlinear least squares problems. GALAHAD contains the following major packages:
LANCELOT B
is the latest release of LANCELOT, and is designed to solve large-scale optimization problems involving the minimization of a nonlinear objective, subject (perhaps) to linear or nonlinear equality and box constraints. All functions involved are assumed to be group partially separable, and the minimization is based on a Sequential Augmented Lagrangian algorithm. The new release is coded in Fortran 95, allows for a non-monotone descent strategy, Moré and Toraldo-type projections, optional use of Lin and Moré's ICFS preconditioner, structured trust regions, and more.
NCELOT_SIMPLE
s a simple-minded interface to LANCELOT for small, dense problems.
RANE
a package for finding a feasible point for a set of linear and/or nonlinear
uations and inequalities using a multi-dimensional filter trust-region approach. In
event that the system is inconsistent, a local measure of infeasibility is minimized. Core linear algebraic requirements are handled by an adaptive preconditioned CG/Lanczos iteration.

Software


Pull requests Issues Marketplace Explore
Search or jump to…
Unwatch 6 Star 0 Fork 0
ralna / OpTrove
Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings
Edit New Page
home
nimgould edited this page on 27 Feb · 4 revisions
OpTrove is a source of open-source optimization software, problems and tools that are available from members of the Computational Mathematics Group at the STFC-Rutherford Appleton Laboratory, and their collaborators. To date this includes:
GALAHAD, a library of modern Fortran modules for nonlinear optimization.
CUTEst, the Constrained and Unconstrained Testing Environment with safe threads (CUTEst) for optimization software.
SIFDecode, a package to decode SIF optimization test examples for use by CUTEst and GALAHAD.
ARCHDefs, a collection of system and compiler default values for OpTrove software.
RALFit, a nonlinear least-squares solver.
SIF, a collection of nonlinear optimization problems in Standard Input Format (SIF).
QPLIB, a library of quadratic programming instances arising in many application areas.
We also maintain
MJDP_software, an archive of optimization software by the late Professor Michael Powell from the University of Cambridge.
Maros-Meszaros, a set of quadratic programming examples by Istvan Maros and Csaba Meszaros in Standard Input Format (SIF).
trove, noun - a store of valuable or delightful things (Oxford English Dictionary).
Pages 1
Find a Page…
home
Add a custom sidebar
Clone this wiki locally
https://github.com/ralna/
Camelot
News
Welcome!
GALAHAD is a thread-safe library of Fortran 2003 packages for solving nonlinear optimization problems. At present, the areas covered by the library are unconstrained and bound-constrained optimization, quadratic programming, nonlinear programming, systems of nonlinear equations and inequalities, and nonlinear least squares problems. GALAHAD contains the following major packages:
LANCELOT B
Evaluation Complexity of Algorithms for Nonconvex Optimization
Theory, Computation, and Perspectives
Coralia Cartis
Nicholas I. M. Gould
Philippe L. Toint
MOS-SIAM Series on Optimization
Trust-Region Methods
Andrew R. Conn
Nicholas I. M. Gould
Philippe L. Toint

Nonlinear Least-squares

$$\min \frac{1}{2}\|r(x)\|^2$$



Nonlinear Least-squares - software

User Documentation for RALFit (Python interface)

RALFit computes a solution $\boldsymbol{x}$ to the non-linear least-squares problem

$$\min_{\boldsymbol{x}} \; F(\boldsymbol{x}) := \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_{\boldsymbol{W}}^2 + \frac{\sigma}{p}\|\boldsymbol{x}\|_2^p,$$

where $\boldsymbol{W} \in \mathbb{R}^{m\times m}$ is a diagonal, non-negative, weighting matrix, and $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), ..., r_m(\boldsymbol{x}))^T$ is a non-linear function.

A typical use may be to fit a function $f(\boldsymbol{x}, t)$ to the data $y_i, \; t_i$, weighted by the uncertainty of the data, σ_i:

$$\min_{\boldsymbol{x}} \; \frac{1}{2}\sum_{i=1}^{m}\left(\frac{y_i - f(\boldsymbol{x}; t_i)}{\sigma_i}\right)^2,$$

which corresponds to taking $r_i(\boldsymbol{x}) := y_i - f(\boldsymbol{x}; t_i)$ and $\boldsymbol{W}$ such that $\boldsymbol{W}_{ii} = (1/\sigma_i^2)$. For this reason we refer to the function $\boldsymbol{r}$ as the *residual* function.

Various algorithms for solving this problem are implemented – see Description of the method used.

Nonlinear Least-square

User Documentation for R

RALFit computes a solution $\boldsymbol{x}$ to the non-linear l

$$\min_{\boldsymbol{x}} \; F(\boldsymbol{x}) := \frac{1}{2}\|$$

where $\boldsymbol{W} \in \mathbb{R}^{m\times m}$ is a diagonal, non-negative, w
$\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T$ is a non-line

A typical use may be to fit a function $f(\boldsymbol{x}, t)$ to th
data, σ_i:

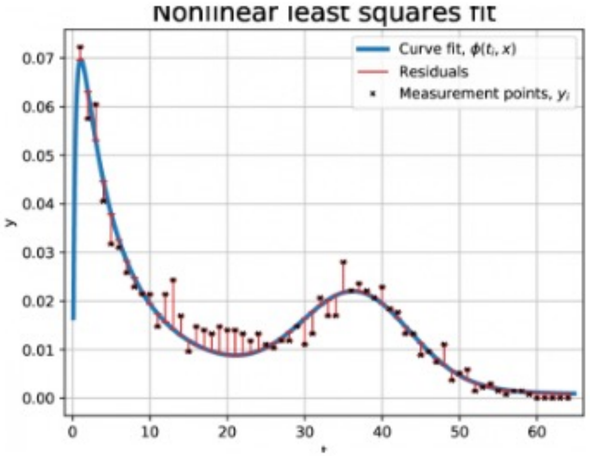
$$\min_{\boldsymbol{x}} \; \frac{1}{2}\sum_{i=1}^{m} \Big($$

which corresponds to taking $r_i(\boldsymbol{x}) := y_i - f(\boldsymbol{x};$
reason we refer to the function $\boldsymbol{r}$ as the *residual* f

Various algorithms for solving this problem are im

Faster Data Fitting Solver

Calibrating the parameters of complex numerical models to fit real-world observations is one of the most common problems found in industries such as finance, physics, simulations, engineering, etc. NAG introduces at Mark 27.1 of the NAG Library a novel nonlinear least squares (NLN-LSQ) trust-region solver handle_solve_bxnl (e04gg) for unconstrained and bound-constrained fitting problems which implements various algorithms and regularization techniques. It is aimed at small to medium-sized fitting problems (up to 1000s of parameters) of the form



$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; \sum_{i=1}^{m} \left(\phi(t_i, x) - y_i\right)^2$$
$$\text{subject to} \quad \ell \le x \le u,$$

where the idea is to find optimal values for the parameters, x, of the model represented by the smooth function $\phi(t, x)$ (blue line) which fit m observed data points (t_i, y_i) (black dots). That is, to minimize the squared error between the model and the data (vertical red bars).

e04gg should present a significant improvement over the current nonlinear least squares solvers in the NAG Library. In addition, this solver fills the gap between unconstrained solvers such as lsq_uncon_quasi_deriv_comp (e04gb) and the fully constrained ones, such as lsq_gencon_deriv (e04us).

e04gg is part of the NAG Optimization Modelling Suite which offers clarity and consistency on the interface of the solvers within the suite.

The new solver stems from a collaboration with the Rutherford Appleton Laboratory [1] and demonstrates NAG's ongoing effort to expand and improve its offering in mathematical optimization.

Nonlinear Least-square

User Documentation for R

RALFit computes a solution $\boldsymbol{x}$ to the non-linear l

$$\min_{\boldsymbol{x}} \; F(\boldsymbol{x}) := \frac{1}{2}\|$$

where $\boldsymbol{W} \in \mathbb{R}^{m \times m}$ is a diagonal, non-negative, w
$\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T$ is a non-line

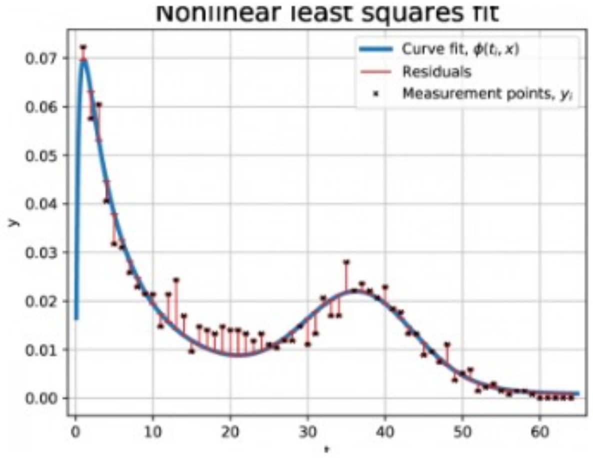
$$\min_{\boldsymbol{x}} \; \frac{1}{2} \sum_{i=1}^{m} \Big(y$$

which corresponds to taking $r_i(\boldsymbol{x}) := y_i - f(\boldsymbol{x};$
reason we refer to the function $\boldsymbol{r}$ as the *residual* f

Various algorithms for solving this problem are im

Faster Data Fitting Solver

Calibrating the parameters of complex numerical models to fit real-world observations is one of the most common problems found in industries such as finance, physics, simulations, engineering, etc. NAG introduces at Mark 27.1 of the NAG Library a novel nonlinear least squares (NLN-LSQ) trust-region solver handle_solve_bxnl (e04gg) for unconstrained and bound-constrained fitting problems which implements various algorithms and regularization techniques. It is aimed at small to medium-sized fitting problems (up to 1000s of parameters) of the form



The new solver e04gg offers unprecedented robustness and a significant speed-up over current alternatives in the Library, namely e04gb for unconstrained nonlinear least squares problems and e04us for problems with simple variable bounds. You are highly recommended to upgrade to the new solver.

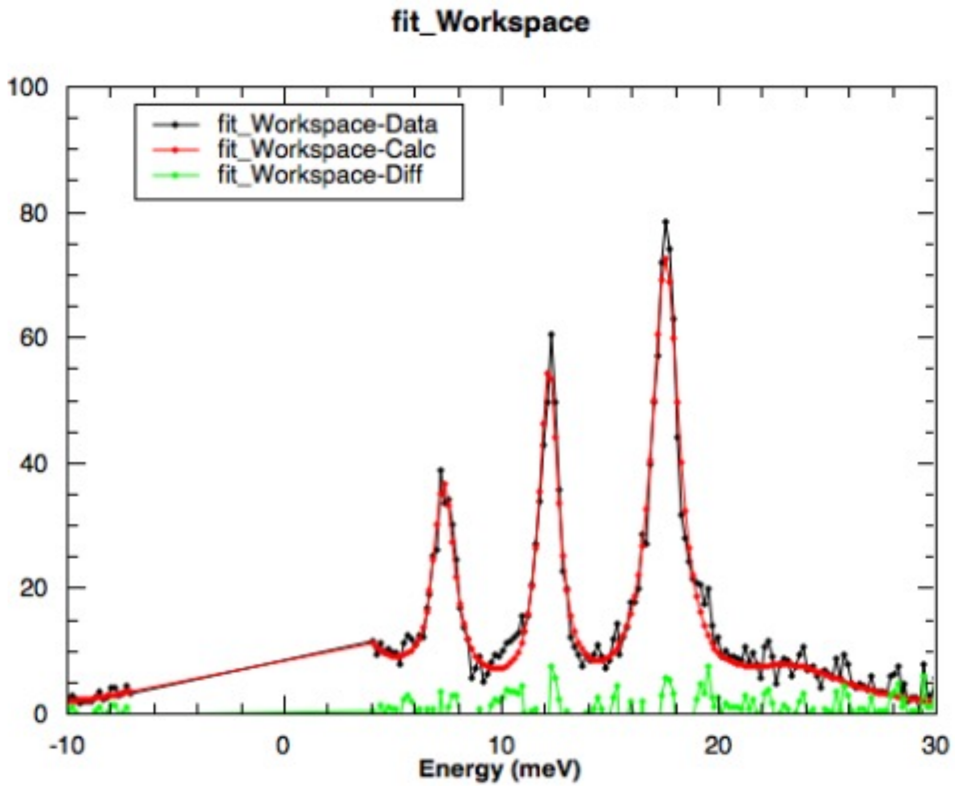
where the idea is to find optimal values for the parameters, x, of the model represented by the smooth function $\phi(t, x)$ (blue line) which fit m observed data points (t_i, y_i) (black dots). That is, to minimize the squared error between the model and the data (vertical red bars).

e04gg should present a significant improvement over the current nonlinear least squares solvers in the NAG Library. In addition, this solver fills the gap between unconstrained solvers such as lsq_uncon_quasi_deriv_comp (e04gb) and the fully constrained ones, such as lsq_gencon_deriv (e04us).

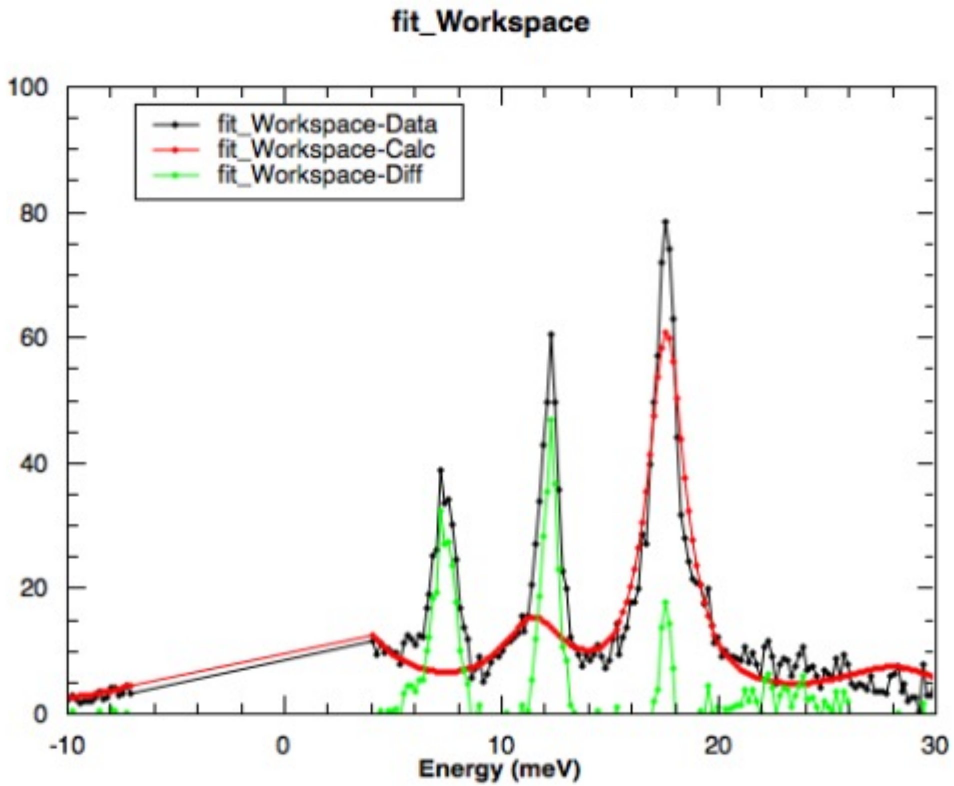
e04gg is part of the NAG Optimization Modelling Suite which offers clarity and consistency on the interface of the solvers within the suite.

The new solver stems from a collaboration with the Rutherford Appleton Laboratory [1] and demonstrates NAG's ongoing effort to expand and improve its offering in mathematical optimization.

Global nonlinear Least-squares



(a) Crystal field fitting with expert-tuned initial parameters.



(b) Crystal field fitting with the expert initial parameters perturbed slightly.

Global Optimization of Crystal Field Parameter Fitting in Mantid

Megan O'Flynn[1], Jaroslav Fowkes[1], and Nick Gould[1]

[1]UKRI-STFC Rutherford Appleton Laboratory, Scientific Computing Department, Computational Mathematics Group

24th February 2022

Abstract

Currently local optimization algorithms are used by the Mantid software package to fit parameters to crystal field data, however such fits are extremely sensitive to the selection of initial parameters. Even introducing a small perturbation to the initial parameters can change the fitting significantly, hence there is a need to consider a global, rather than a local, optimization approach to fit crystal field data. In this report, we propose and test several different global optimization algorithms that could be integrated into Mantid's fitting routines to try and make crystal field fitting more robust. Our numerical results on two real-world datasets demonstrate that there is great benefit to the use of global optimization for crystal field parameter fitting in Mantid.

Contents

1 Introduction 2
1.1 Crystal Field Parameter Fitting in Mantid 2
1.2 Nonlinear Least-Squares Problems 3
1.3 The Crystal Field Model 3

2 Developing Global Optimization Algorithms 5
2.1 Multi-start Optimization Algorithms 5
2.2 Multi-start Selective Search (MS3) 5
2.3 Multi-start with Regularisation 7
2.4 Multi-start with Linesearch 8
2.5 Sine Components Test Problems in High Dimensions 9

3 Fitting Crystal Field Data from Mantid 12
3.1 Crystal Field Test Problems 12

	globalmultistart		mantid									
	l_bfgs: scipy 2-point	scaled_regularisation: scipy 2-point	BFGS: scipy 2-point	Conjugate gradient (Fletcher-Reeves imp.): scipy 2-point	Conjugate gradient (Polak-Ribiere imp.): scipy 2-point	Damped GaussNewton: scipy 2-point	Levenberg-Marquardt: scipy 2-point	Levenberg-MarquardtMD: scipy 2-point	Simplex	SteepestDescent: scipy 2-point	Trust Region: scipy 2-point
Example 1	266.9 (1.018)	317.3 (1.211)	5351 (20.42)[2]	2373 (9.056)[2]	2381 (9.086)[2]	8809 (33.62)[2]	263.3 (1.005)	3.266e+04 (124.7)[2]	inf (inf)[3]	4143 (15.81)[1]	262 (1)[2]
Example 2	272 (1.037)	262.3 (1)	6685 (25.48)[5]	8141 (31.04)[5]	8141 (31.04)[5]	2.487e+17 (9.481e+14)[5]	1673 (6.376)	1.528e+05 (582.6)[2]	inf (inf)[3]	7622 (29.06)[5]	270.6 (1.032)[2]
Example 3	602.8 (1.914)	314.9 (1)	7762 (24.65)[5]	8149 (25.88)[5]	6633 (21.06)[5]	8809 (27.97)[2]	5.243e+04 (166.5)	2580 (8.194)[1]	inf (inf)[3]	8034 (25.51)[5]	1.338e+04 (42.49)[5]
Example 4	681.4 (1.176)	579.5 (1)[5]	7628 (13.16)[5]	6643 (11.46)[5]	6653 (11.48)[5]	6.7e+05 (1156)[5]	4.003e+04 (69.09)	8809 (15.2)[2]	inf (inf)[3]	6754 (11.66)[5]	2836 (4.893)[5]
Example 5	268.7 (1.019)	264.5 (1.004)	7625 (28.93)[5]	6644 (25.21)[5]	6668 (25.3)[5]	3.751e+04 (142.3)[5]	4032 (15.3)	8809 (33.42)[2]	inf (inf)[3]	6678 (25.34)[5]	263.5 (1)[2]

Figure 7: FitBenchmarking accuracy $f(x^*)$ on Crystal Field Problem 1 (values in brackets are relative to the best result). Superscript meanings: 1 - maximum number of iterations exceeded, 2 - unable to converge to solution, 3 - internal error, 5 - solution does not respect parameter bounds.

Global Optimization of Crystal Field Parameter Fitting in Mant

Megan O'Flynn[1], Jaroslav Fowkes[1], and N
[1]UKRI-STFC Rutherford Appleton Laboratory, S
Department, Computational Mathemat

24th February 2022

Abstract

Currently local optimization algorithms are used by package to fit parameters to crystal field data, however s sensitive to the selection of initial parameters. Even int turbation to the initial parameters can change the fittin there is a need to consider a global, rather than a local, o to fit crystal field data. In this report, we propose and global optimization algorithms that could be integrated routines to try and make crystal field fitting more r results on two real-world datasets demonstrate that th the use of global optimization for crystal field paramet

Contents

1 Introduction
1.1 Crystal Field Parameter Fitting in Mantid . .
1.2 Nonlinear Least-Squares Problems . . . . . .
1.3 The Crystal Field Model . . . . .
2 Developing Global Optimization Algorithms
2.1 Multi-start Optimization Algorithms . . . .
2.2 Multi-start Selective Search (MS3) . . . . .
2.3 Multi-start with Regularisation . . . .
2.4 Multi-start with Linesearch . . . .
2.5 Sine Components Test Problems in High Dim
3 Fitting Crystal Field Data from Mantid
3.1 Crystal Field Test Problems . . . .

globalmultistart		mantid									
l_bfgs: scipy 2-	scaled_regularisation:	BFGS: scipy 2-	Conjugate gradient (Fletcher-	Conjugate gradient (Polak-	Damped GaussNewton:	Levenberg-Marquardt:	Levenberg-MarquardtMD: scipy 2-point	Simplex	SteepestDescent: scipy 2-point	Trust Region: scipy 2-point
							6e+04 (.7)[2]	inf (inf)[3]	4143 (15.81)[1]	262 (1)[2]
							8e+05 (2.6)[2]	inf (inf)[3]	7622 (29.06)[5]	270.6 (1.032)[2]
							0 (8.194)[1]	inf (inf)[3]	8034 (25.51)[5]	1.338e+04 (42.49)[5]
							9 (15.2)[2]	inf (inf)[3]	6754 (11.66)[5]	2836 (4.893)[5]
							9 (33.42)[2]	inf (inf)[3]	6678 (25.34)[5]	263.5 (1)[2]

Problem 1 (values in 1 - maximum number 3 - internal error, 5 -

GOFit

Global Optimization for Fitting problems

Release: 0.4

Date: 15 September 2023

Author: Jaroslav Fowkes

GOFit is a package of C++ algorithms with python interfaces designed for the global optimization of parameters in curve fitting, i.e. for nonlinear least-squares problems arising from curve fitting. GOFit was developed with scientific curve fitting problems in mind [1] but is also applicable to general curve fitting problems provided they can be formulated as nonlinear least-squares problems.

FitBenchmarking

interface any data fitting software

NLOPT

GSL

SciPy

Bumps

ParaMonte

Parallel Monte Carlo & Machine Learning Library

Google Ceres

Science and Technology Facilities Council

GOFit

RALFit

nag DFO-GN/LS

Theseus

levmar

CERN Minuit

LMFIT

Gradient Free Optimizers

Matlab

MANTID

NIST Center for Neutron Research

DAVE

Data Analysis and Visualization Environment

CHRNS NIST

SasView

Spin

CUTEst

NIST StRD

Stress Map

HOGBEN

Holistic Optimization for Gaining Better Evidence from Neutrons

to any source of test problems

Key:

FitBenchmarking

Top Colour - Relative Accuracy
Bottom Colour - Relative Runtime

Best (1) — Worst (>100)

Table:

WeightedNLLSCostFunc

	gsl					levmar	
	lmder	lmder: j:scipy 2-point	lmsder	lmsder: j:scipy 2-point	nmsimplex	levmar	levmar: j:scipy 2-point
ENGINX 193749 calibration, spectrum 651, peak 19	inf (inf) inf (inf)[3]	inf (inf) inf (inf)[3]	54.4 (1) 0.6348 (55.1)	54.4 (1) 0.6349 (55.11)	56.45 (1.038) 0.9912 (86.03)[1]	99.41 (1.827) 0.01152 (1)	99.41 (1.827) 0.02359 (2.048)
ENGINX 193749 calibration, spectrum 651, peak 20	56.38 (1) 0.4192 (44.11)	56.38 (1) 0.4101 (43.15)	56.38 (1) 0.1857 (19.54)	56.38 (1) 0.1887 (19.85)	64.84 (1.15) 0.8394 (88.33)[1]	224.5 (3.982) 0.009503 (1)	224.5 (3.982) 0.01961 (2.063)
ENGINX 193749 calibration, spectrum 651, peak 23	76.09 (1) 0.7439 (79)	76.09 (1) 0.7365 (78.22)	76.09 (1) 0.4968 (52.77)	76.09 (1) 0.5093 (54.1)	115.1 (1.512) 0.8105 (86.09)[1]	718 (9.436) 0.009416 (1)	718 (9.436) 0.01942 (2.062)
ENGINX 193749 calibration, spectrum 651, peak 5	14.58 (1.002) 0.2981 (31.86)	14.58 (1.002) 0.2973 (31.78)	14.58 (1.002) 0.7617 (81.41)	14.58 (1.002) 0.8031 (85.84)	16.16 (1.111) 0.8115 (86.73)[1]	24.82 (1.706) 0.009356 (1)	24.82 (1.706) 0.01945 (2.079)
ENGINX 193749 calibration, spectrum 651, peak 6	26.74 (1.141) 0.2948 (25.44)	26.74 (1.141) 0.254 (21.92)	inf (inf) inf (inf)[3]	inf (inf) inf (inf)[3]	25.79 (1.101) 0.7929 (68.43)[1]	40.12 (1.712) 0.01159 (1)	40.12 (1.712) 0.01933 (1.669)
ENGINX 236516 vanadium, bank 1, 10 brk	8.455e+04 (1) 0.9798 (3.102)	8.455e+04 (1) 0.9797 (3.101)	8.455e+04 (1) 1.312 (4.154)	8.455e+04 (1) 1.351 (4.277)	6.362e+06 (75.24) 13.75 (43.53)[1]	8.455e+04 (1) 0.3159 (1)	8.455e+04 (1) 1.303 (4.124)
	2.478e+04		2.478e+04				

	ralfit							
	gn	gn: j:scipy 2-point	gn_reg	gn_reg: j:scipy 2-point	hybrid	hybrid: j:scipy 2-point	hybrid_reg	hybrid_reg: j:scipy 2-point
ENGINX 193749 calibration, spectrum 651, peak 19	54.49 (1.002) 5.651 (490.5)	54.49 (1.002) 5.483 (475.9)	54.52 (1.002) 5.606 (486.6)	54.52 (1.002) 5.735 (497.7)	54.4 (1) 1.533 (133)	54.4 (1) 1.635 (141.9)	54.4 (1) 1.56 (135.4)	54.4 (1) 1.642 (142.5)
ENGINX 193749 calibration, spectrum 651, peak 20	56.38 (1) 0.1787 (18.81)	56.38 (1) 0.1792 (18.85)	56.38 (1) 0.301 (31.67)	56.38 (1) 0.3002 (31.59)	56.38 (1) 0.3705 (38.99)	56.38 (1) 0.3704 (38.98)	56.38 (1) 0.3681 (38.74)	56.38 (1) 0.3839 (40.4)
ENGINX 193749 calibration, spectrum 651, peak 23	76.11 (1) 5.749 (610.6)	76.11 (1) 5.443 (578.1)	76.14 (1.001) 5.444 (578.2)	76.14 (1.001) 5.519 (586.2)	76.09 (1) 0.8743 (92.86)	76.09 (1) 0.8799 (93.46)	76.09 (1) 0.8755 (92.99)	76.09 (1) 0.8725 (92.66)
ENGINX 193749 calibration, spectrum 651, peak 5	14.55 (1) 5.066 (541.4)	14.55 (1) 4.981 (532.3)	14.68 (1.009) 5.435 (580.9)	14.68 (1.009) 5.384 (575.5)	14.91 (1.025) 0.7878 (84.2)	14.91 (1.025) 0.7705 (82.36)	14.91 (1.025) 0.7704 (82.34)	14.91 (1.025) 0.7969 (85.17)
ENGINX 193749 calibration, spectrum 651, peak 6	23.46 (1.001) 5.159 (445.2)	23.46 (1.001) 5.14 (443.6)	23.47 (1.002) 5.197 (448.6)	23.47 (1.002) 5.179 (447)	23.43 (1) 5.41 (466.9)	23.43 (1) 5.334 (460.4)	23.43 (1) 5.413 (467.1)	23.43 (1) 5.347 (461.5)
ENGINX 236516 vanadium, bank 1, 10 brk	8.455e+04 (1) 1.801 (5.7)	8.455e+04 (1) 1.367 (4.328)	8.455e+04 (1) 1.04 (3.293)	8.455e+04 (1) 1.054 (3.337)	8.455e+04 (1) 1.535 (4.858)	8.455e+04 (1) 1.55 (4.907)	8.455e+04 (1) 1.503 (4.759)	8.455e+04 (1) 1.501 (4.751)
	2.478e+04		2.478e+04		2.478e+04			

	scipy-ls						
	dogbox	dogbox: j:scipy 2-point	lm-scipy: j:scipy 2-point	lm	trf	trf: j:scipy 2-point
ENGINX 193749 calibration, spectrum 651, peak 19	62.6 (1.151) 4.427 (384.2)[1]	62.62 (1.151) 4.909 (426)[1]	54.4 (1) 0.4637 (40.24)	54.4 (1) 0.4004 (34.75)	54.4 (1) 0.3025 (26.25)	54.4 (1) 0.3311 (28.74)
ENGINX 193749 calibration, spectrum 651, peak 20	120.5 (2.137) 4.495 (472.9)[1]	104.9 (1.861) 4.801 (505.2)[1]	56.38 (1) 0.4264 (44.87)	56.38 (1) 0.3616 (38.05)	56.38 (1) 0.4852 (51.06)	56.38 (1) 0.5995 (63.09)
ENGINX 193749 calibration, spectrum 651, peak 23	77.48 (1.018) 4.753 (504.8)[1]	76.7 (1.008) 5.074 (538.9)[1]	76.09 (1) 0.7502 (79.68)	76.09 (1) 0.4921 (52.27)	76.09 (1) 0.5049 (53.62)	76.09 (1) 0.5285 (56.13)
ENGINX 193749 calibration, spectrum 651, peak 5	22.86 (1.571) 4.645 (496.5)[1]	22.86 (1.571) 5.21 (556.9)[1]	14.58 (1.002) 0.2998 (32.04)	14.58 (1.002) 0.2639 (28.21)	15.38 (1.057) 0.2291 (24.48)	15.38 (1.057) 0.2896 (30.96)
ENGINX 193749 calibration, spectrum 651, peak 6	29 (1.237) 3.545 (305.9)[1]	28.25 (1.206) 4.163 (359.3)[1]	26.74 (1.141) 0.3064 (26.44)	26.72 (1.141) 0.4746 (40.96)	26.68 (1.139) 0.3492 (30.14)	26.68 (1.139) 0.3576 (30.86)
ENGINX 236516 vanadium, bank 1, 10 brk	8.455e+04 (1) 3.101 (9.817)	8.455e+04 (1) 3.136 (9.929)	8.455e+04 (1) 1.324 (4.19)	8.455e+04 (1) 1.291 (4.086)	8.455e+04 (1) 3.911 (12.38)	8.455e+04 (1) 3.79 (12)
	2.478e+04			2.478e+04	2.478e+04	

Clicking a result in the tables will give more details, such as graphs of the fit against the data and the parameters that the minimizer found.

Clicking the problem names will take you to details of the best minimizer.

Clicking the software name will take you to FitBenchmarking Read the Docs documentation for the selected software.

Hovering over each minimizer name will display its matching algorithm types out of those selected in options.

Key:

FitBenchmarking

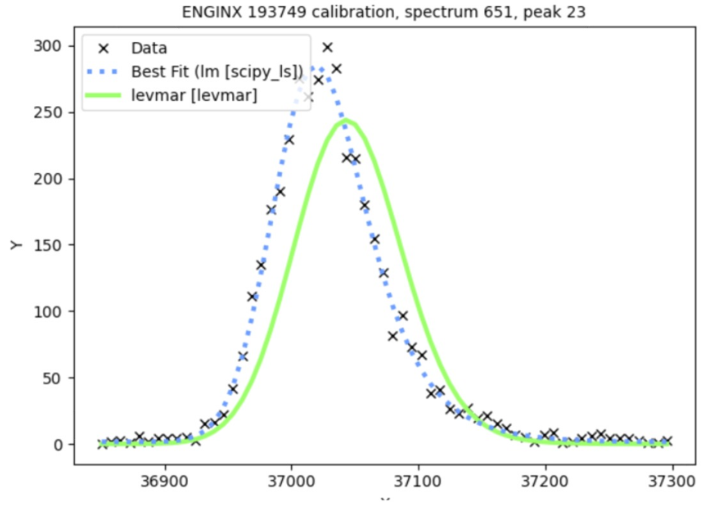
Top Colour - Relative Accuracy
Bottom Colour - Relative Runtime

Best (1) Worst (>1…

Table:

	WeightedNLLSCostFunc																				
	gsl					levmar		ralfit													[illegible]
	lmder	lmder: j:scipy 2-point	lmsder	lmsder: j:scipy 2-point	nmsimplex	levmar	levmar: j:scipy 2-point	gn	gn: j:scipy 2-point	gn_reg	gn_reg: j:scipy 2-point	hybrid	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	trf: j:scipy 2-point
ENGINX 193749 calibration, spectrum 651, peak 19	inf (inf) inf (inf)[3]	inf (inf) inf (inf)[3]	54.4 (1) 0.6348 (55.1)	54.4 (1) 0.6349 (55.11)	56.45 (1.038) 0.9912 (86.03)[1]	99.41 (1.827) 0.01152 (1)	99.41 (1.827) 0.02359 (2.048)	54.49 (1.002) 5.651 [illegible]	…49 (1.002) 5.483 (475.9)	54.52 (1.002) 5.606 (486.6)	54.52 (1.002) 5.735 (497.7)	54.4 (1) 1.533 (133)	54.4 … 1.635 …	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	54.4 (1) 0.3311 (28.74)
ENGINX 193749 calibration, spectrum 651, peak 20	56.38 (1) 0.4192 (44.11)	56.38 (1) 0.4101 (43.15)	56.38 (1) 0.1857 (19.54)	56.38 (1) 0.1887 (19.85)	64.84 (1.15) 0.8394 (88.33)[1]	224.5 (3.982) 0.009503 (1)	224.5 (3.9… 0.019… …3)	56.38 (1) 0.1787 (18.81)	56.38 (1) 0.1792 (18.85)	56.38 (1) 0.301 (31.67)	56.38 (1) 0.3002 (31.59)	56.38 (1) 0.3705 (38.99)	56.3… 0.370…	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	56.38 (1) 0.5995 (63.09)
ENGINX 193749 calibration, spectrum 651, peak 23	76.09 (1) 0.7439 (79)	76.09 (1) 0.7365 (78.22)	76.09 (1) 0.4968 (52.77)	76.09 (1) 0.5093 (54.1)	115.1 (1.512) 0.8105 (86.09)[1]	718 (9.436) 0.009416 (1)	…8 (9.436) …01942 …062)	76.11 (1) 5.749 (610.6)	76.11 (1) 5.443 (578.1)	76.14 (1.001) 5.444 [illegible]	76.14 (1.001) [illegible]	76.09 (1) [illegible] (92.86)	[illegible] 0.8799 (93.46)	[illegible] 0.8755 (92.99)	[illegible] 0.8725 (92.66)	[illegible] 4.753 (504.8)[1]	[illegible] 5.074 (538.9)[1]	[illegible] 0.7502 (79.68)	[illegible] 0.4921 (52.27)	[illegible] 0.5049 (53.62)	76.09 (1) 0.5285 (56.13)
ENGINX 193749 calibration, spectrum 651, peak 5	14.58 (1.002) 0.2981 (31.86)	14.58 (1.002) 0.2973 (31.78)	14.58 (1.002) 0.7617 (81.41)	14.58 (1.002) 0.8031 (85.84)	16.16 (1.111) 0.8115 (86.73)[1]	24.82 (1.706) 0.009356 (1)	24.82 (1.706) 0.01945 (2.079)	14.55 (1) 5.066 (541.4)	14.55 (1) 4.981 (532.3)	14.68 (1.009) 5.435 (580.9)	14.68 (1.009) 5.384 (575.5)	14.91 (1.025) 0.7878 (84.2)	14.91 (1.025) 0.7705 (82.36)	14.91 (1.025) 0.7704 (82.34)	14.91 (1.025) 0.7969 (85.17)	22.86 (1.571) 4.645 (496.5)[1]	22.86 (1.571) 5.21 (556.9)[1]	14.58 (1.002) 0.2998 (32.04)	14.58 (1.002) 0.2639 (28.21)	15.38 (1.057) 0.2291 (24.48)	15.38 (1.057) 0.2896 (30.96)
ENGINX 193749 calibration, spectrum 651, peak 6	26.74 (1.141) 0.2948 (25.44)	26.74 (1.141) 0.254 (21.92)	inf (inf) inf (inf)[3]	inf (inf) inf (inf)[3]	25.79 (1.101) 0.7929 (68.43)[1]	40.12 (1.712) 0.01159 (1)	40.12 (1.712) 0.01933 (1.669)	23.46 (1.001) 5.159 (445.2)	23.46 (1.001) 5.14 (443.6)	23.47 (1.002) 5.197 (448.6)	23.47 (1.002) 5.179 (447)	23.43 (1) 5.41 (466.9)	23.43 (1) 5.334 (460.4)	23.43 (1) 5.413 (467.1)	23.43 (1) 5.347 (461.5)	29 (1.237) 3.545 (305.9)[1]	28.25 (1.206) 4.163 (359.3)[1]	26.74 (1.141) 0.3064 (26.44)	26.72 (1.141) 0.4746 (40.96)	26.68 (1.139) 0.3492 (30.14)	26.68 (1.139) 0.3576 (30.86)
ENGINX 236516 vanadium, bank 1, 10 brk	8.455e+04 (1) 0.9798 (3.102)	8.455e+04 (1) 0.9797 (3.101)	8.455e+04 (1) 1.312 (4.154)	8.455e+04 (1) 1.351 (4.277)	6.362e+06 (75.24) 13.75 (43.53)[1]	8.455e+04 (1) 0.3159 (1)	8.455e+04 (1) 1.303 (4.124)	8.455e+04 (1) 1.801 (5.7)	8.455e+04 (1) 1.367 (4.328)	8.455e+04 (1) 1.04 (3.293)	8.455e+04 (1) 1.054 (3.337)	8.455e+04 (1) 1.535 (4.858)	8.455e+04 (1) 1.55 (4.907)	8.455e+04 (1) 1.503 (4.759)	8.455e+04 (1) 1.501 (4.751)	8.455e+04 (1) 3.101 (9.817)	8.455e+04 (1) 3.136 (9.929)	8.455e+04 (1) 1.324 (4.19)	8.455e+04 (1) 1.291 (4.086)	8.455e+04 (1) 3.911 (12.38)	8.455e+04 (1) 3.79 (12)
	2.478e+04		2.478e+04					2.478e+04		2.478e+04		2.478e+04				2.478e+04			2.478e+04	2.478e+04	

Form	Parameters
2 Functions	f0.A0=0.0, f0.A1=0.0, f1.I=26061.4, f1.A=1.0, f1.B=0.05, f1.X0=37027.1, f1.S=38.7105



Clicking a result in the tables will give more details, such as graphs of the fit against the data and the parameters that the minimizer found.

Clicking the problem names will take you to details of the best minimizer.

Clicking the software name will take you to FitBenchmarking Read the Docs documentation for the selected software.

Hovering over each minimizer name will display its matching algorithm types out of those selected in options.

Key:

FitBenchmarking

Table:

	gsl					
	lmder	lmder: j:scipy 2-point	lmsder	lmsder: j:scipy 2-point	nmsimplex	lev[illegible]
ENGINX 193749 calibration, spectrum 651, peak 19	inf (inf) inf [illegible][3]	inf (inf) inf (inf)[3]	54.4 (1) 0.6348 (55.1)	54.4 (1) 0.6349 (55.11)	56.45 (1.038) 0.9912 (86.03)[1]	99.41 0.0115
ENGINX 193749 calibration, spectrum 651, peak 20	[illegible]6.38 (1) [illegible].4192 [illegible]4.11)	56.38 (1) 0.4101 (43.15)	56.38 (1) 0.1857 (19.54)	56.38 (1) 0.1887 (19.85)	64.84 (1.15) 0.8394 (88.33)[1]	224.5 0.009
ENGINX 193749 calibration, spectrum 651, peak 23	76.[illegible] (1) 0.743[illegible]9)	76.09 (1) 0.7365 (78.22)	76.09 (1) 0.4968 (52.77)	76.09 (1) 0.5093 (54.1)	115.1 (1.512) 0.8105 (86.09)[1]	718 (9 0.009
ENGINX 193749 calibration, spectrum 651, peak 5	14.58 (1.002) 0.2981 (31.86)	14.[illegible] (1.002) 0.2973 [illegible].78)	14.58 (1.002) 0.7617 (81.41)	14.58 (1.002) 0.8031 (85.84)	16.16 (1.111) 0.8115 (86.73)[1]	24.82 0.009
ENGINX 193749 calibration, spectrum 651, peak 6	26.74 (1.141) 0.2948 (25.44)	26.74 (1.141) 0.254 (21.92)	in[illegible]nf) inf (i[illegible])[3]	inf (inf) inf (inf)[3]	25.79 (1.101) 0.7929 (68.43)[1]	40.12 0.0115
ENGINX 236516 vanadium, bank 1, 10 brk	8.455e+04 (1) 0.9798 (3.102)	8.455e+04 (1) 0.9797 (3.101)	8.455e+04 (1) 1.312 (4.154)	8.4[illegible]+04 (1) 1.351 [illegible]77)	6.362e+06 (75.24) 13.75 (43.53)[1]	8.455 (1) 0.3159
	2.478e+04		2.478e+04			

Clicking a result in the tables will give more details, such as graphs of the fit against the data and the [illegible]arameters that

Clicking the problem names will take you to details of the best minimizer.

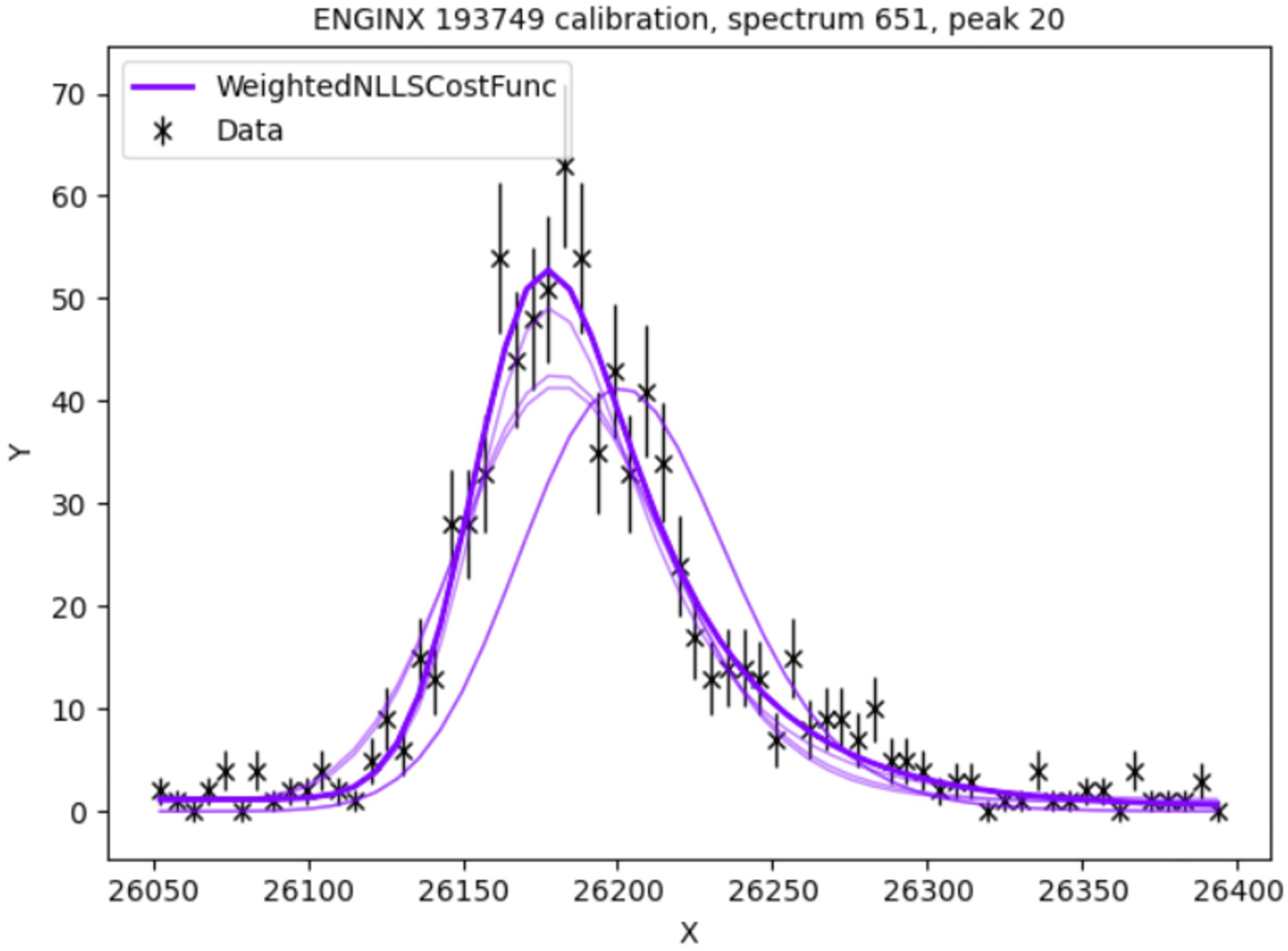
Clicking the software name will take you to FitBenchmarking Read the Docs documentation for the selected so[illegible]are.

Hovering over each minimizer name will display its matching algorithm types out of those selected in options.

› Comparison

This plot shows a comparison of the runs. The minimizer which produced closest fit is in bold, with the other minimizers shown as thinner lines for context.

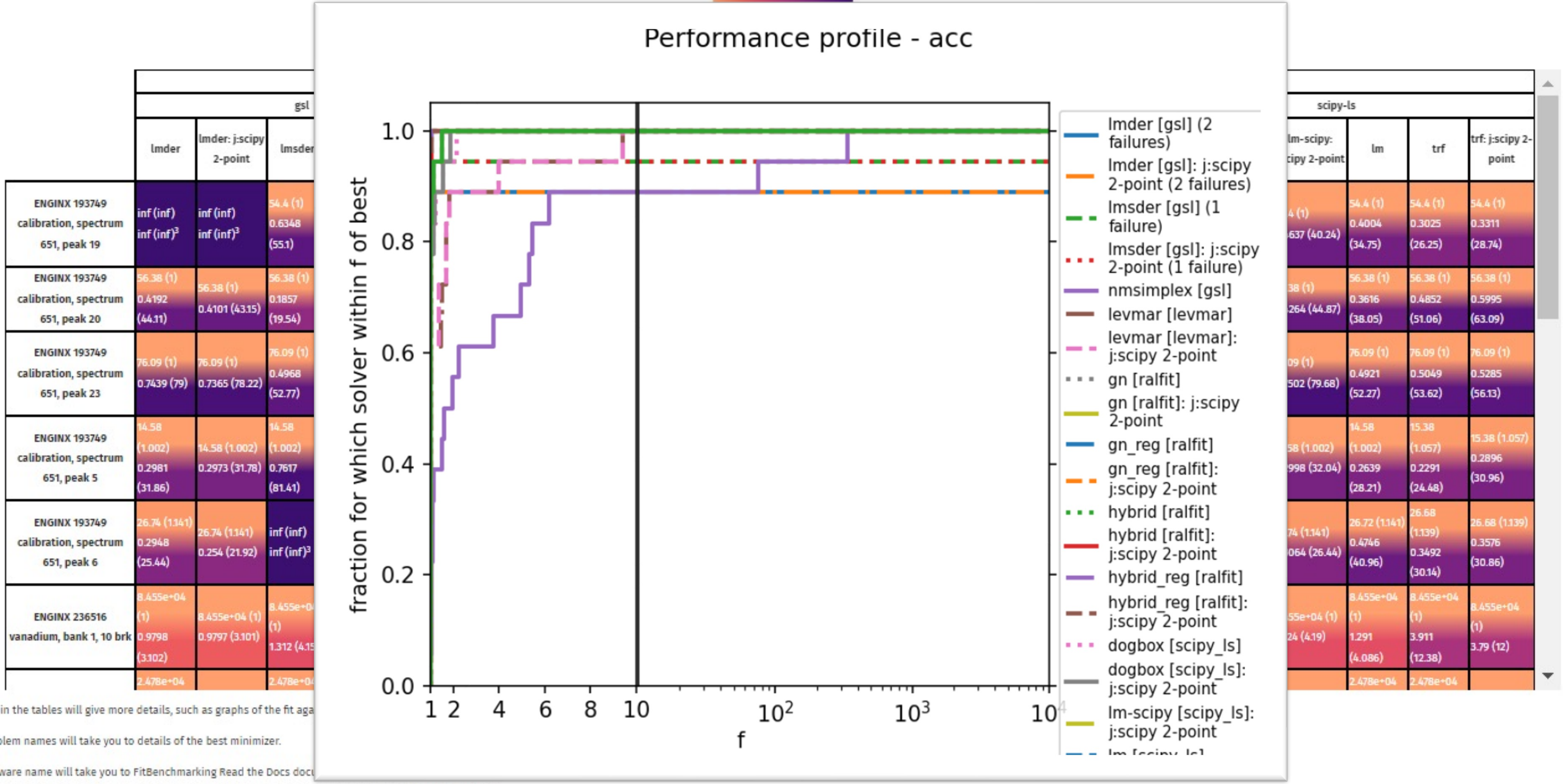
Please note that this plot is intended for use as an indication of the variety of the fits that were obtained for specific details, please consult the individual support pages.



Key:

FitBenchmarking

Table:



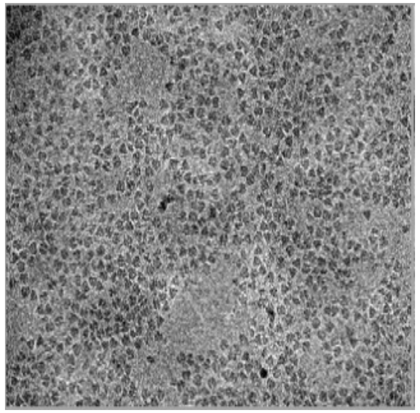
Clicking a result in the tables will give more details, such as graphs of the fit aga

Clicking the problem names will take you to details of the best minimizer.

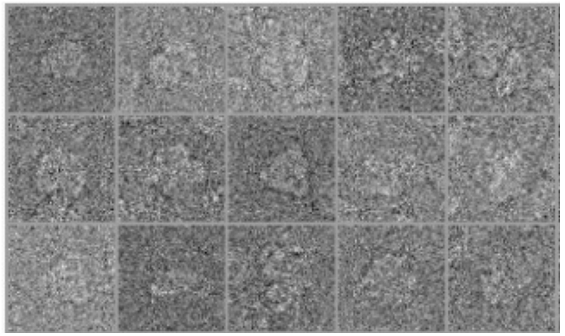
Clicking the software name will take you to FitBenchmarking Read the Docs doc

Hovering over each minimizer name will display its matching algorithm types out of those selected in options.

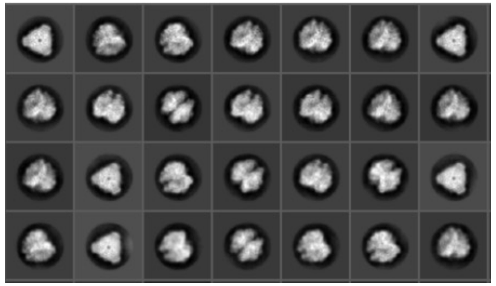
Cryo-EM Cross-Validation Project



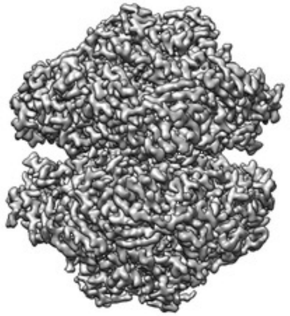
Micrographs



Particle picking and extraction

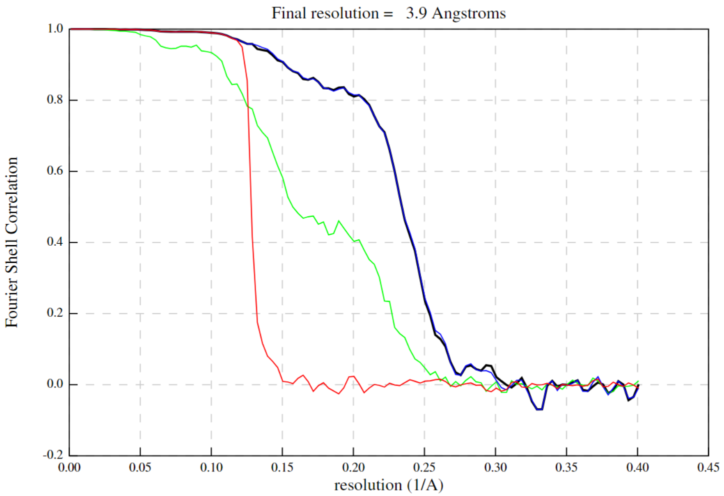


Particle alignment and classification

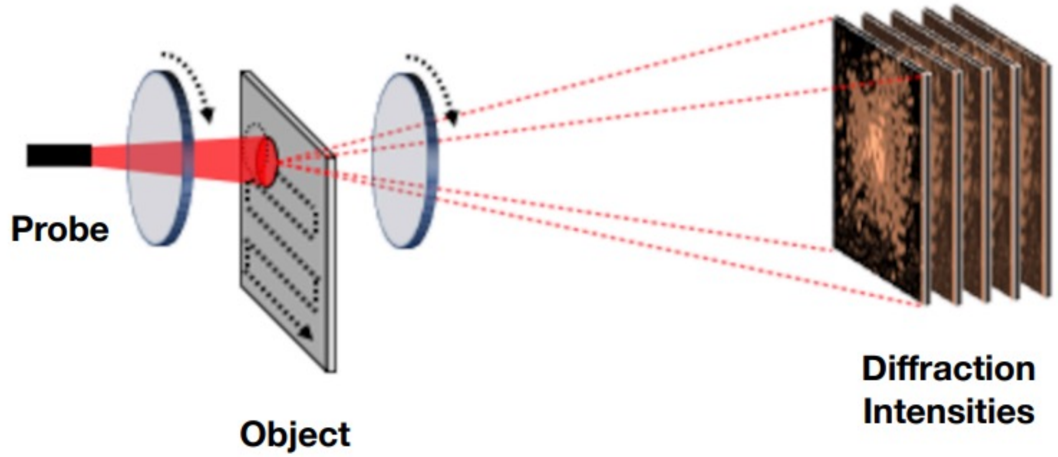


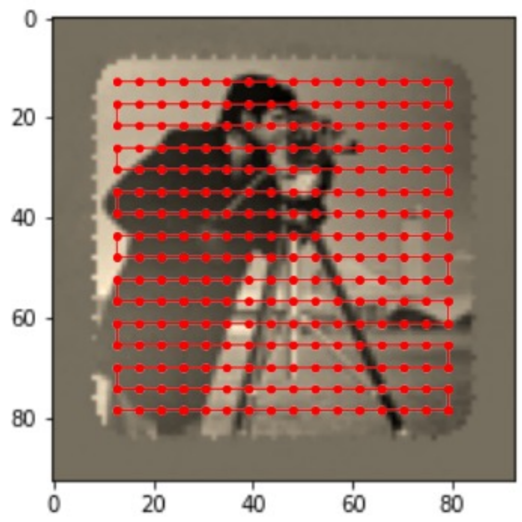
Reconstructed map

$$FSC(r) = \frac{\sum_{r_i \in r} F_1(r_i) \cdot F_2(r_i)^*}{\sqrt[2]{\sum_{r_i \in r} |F_1(r_i)|^2 \cdot \sum_{r_i \in r} |F_2(r_i)|^2}}$$



Ptychography



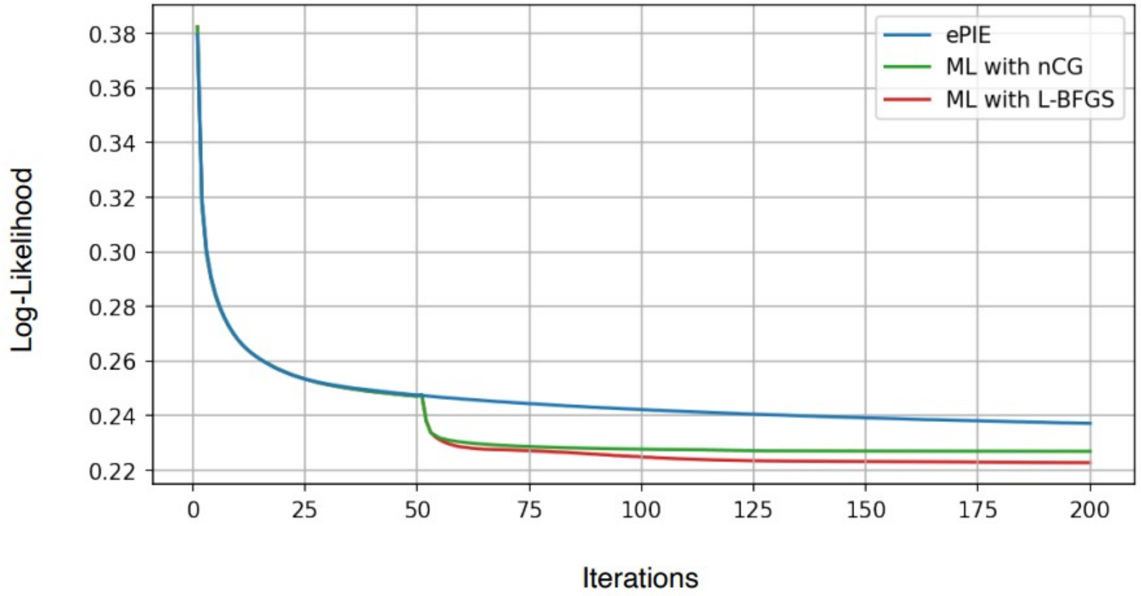


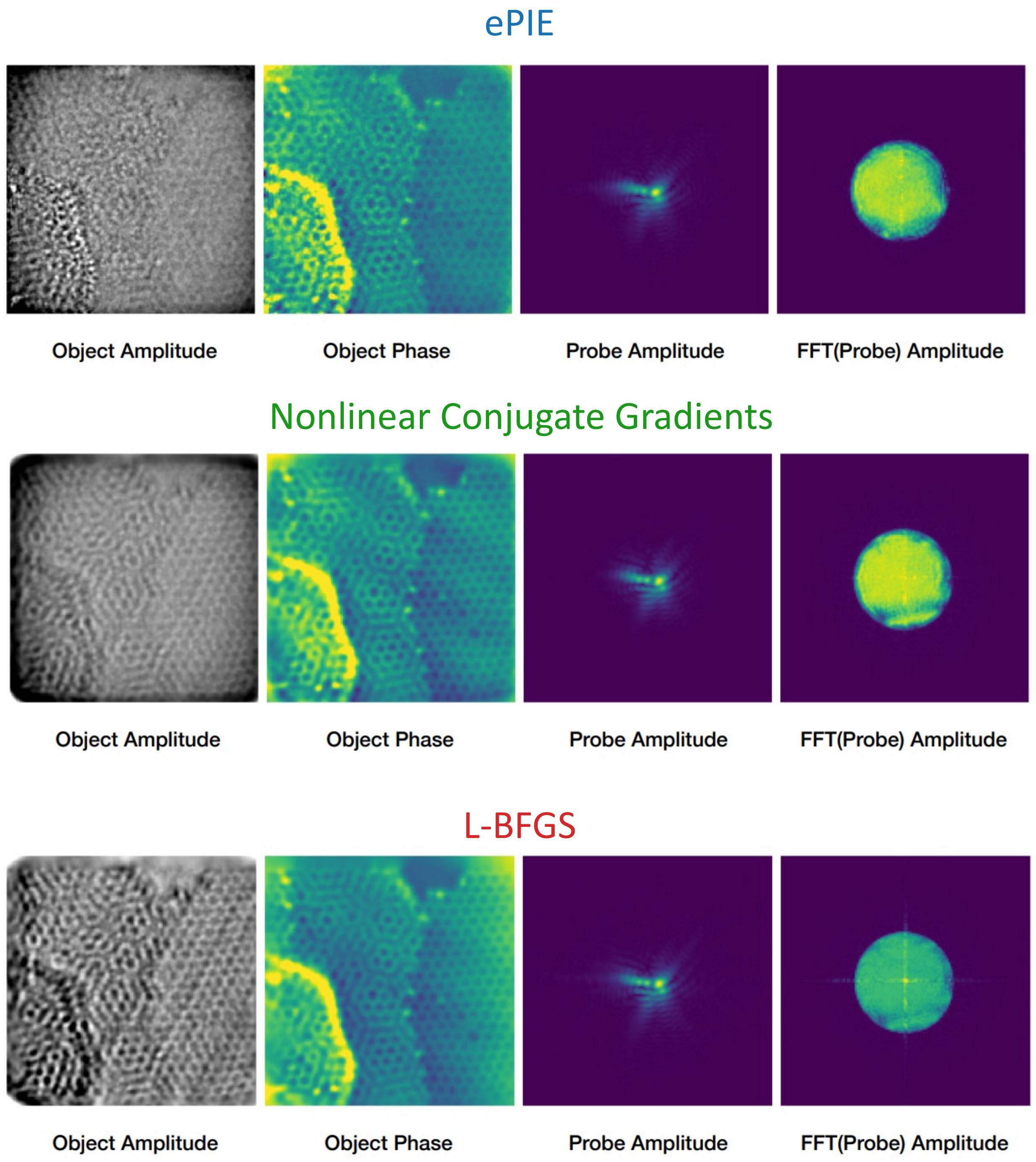
Reconstruct object (and probe) from measured intensities:

$$\min_{o_j\,p} \sum_j \left\| \, |F(o_j\,p)|^2 - i_j \, \right\|^2$$

a sum of nonlinear least-squares problems in $\mathbb{C}^{n \cdot n}$.

Ptychography





Graphene dataset courtesy of Chris Allen (ePSIC)

Summary

Yesterday's algorithms will not be able to analyse tomorrow's data

- Existing expertise in Numerical Linear Algebra and Optimization, well connected in other areas of computational mathematics.
- Linear algebra software
  - HSL: https://hsl.rl.ac.uk
  - SPRAL: https://github.com/ralna/spral
- Optimization software
  - OpTrove: https://github.com/ralna/OpTrove/wiki
  - RALFit: https://github.com/ralna/RALFit
  - GOFit: https://github.com/ralna/Gofit
  - FitBenchmarking: https://fitbenchmarking.github.io/

https://numerical.rl.ac.uk