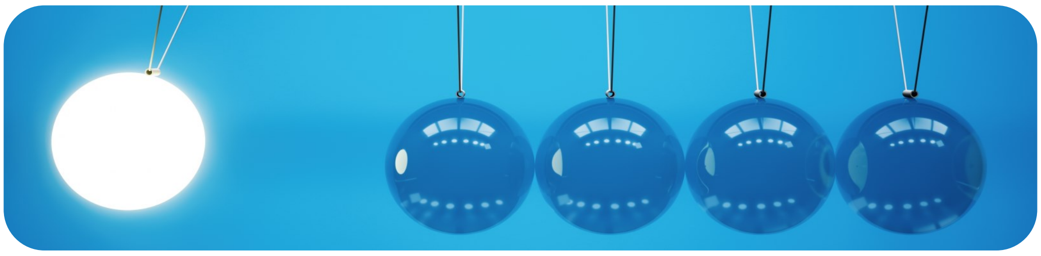


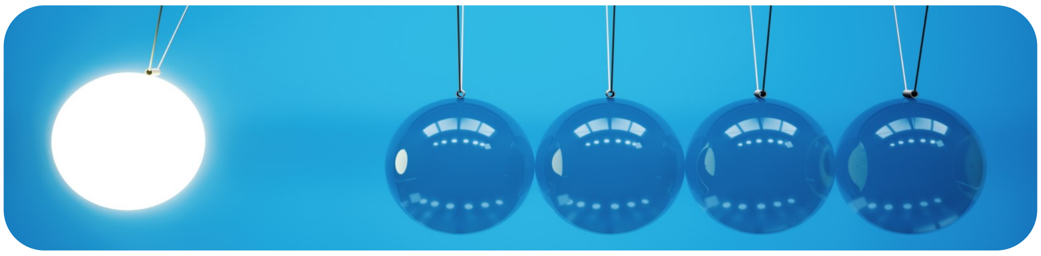
Numerical Algorithms Group

Why Use NAG?

Software Guide

Did you know….?

NAG develops, supports and maintains the NAG Library, a world-renowned Compiler, Algorithmic Differentiation and Optimization software as well as providing specialised HPC and Cloud services



Who is NAG?

The Numerical Algorithms Group (NAG) applies its unique expertise in numerical engineering to delivering high-quality computational software, consultancy and high performance computing services. For 5 decades NAG experts have worked closely with world-leading researchers in academia and industry to provide powerful, reliable and flexible software and solutions relied on by tens of thousands of individual users. Together with the globally renowned NAG Library, NAG experts assist others in many ways, whether that be by parallelizing serial codes, so they optimally perform on HPC hardware to delivering in-depth technical training courses through developing bespoke algorithms.

NAG is a not-for-profit organisation and serves its customers from offices in Oxford, Manchester, Chicago and Tokyo, through staff in France and Germany, as well as via a global network of distributors.

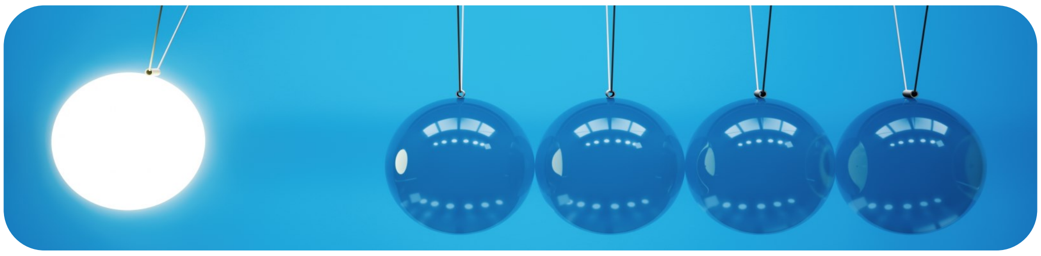
Why use NAG?

All over the world students and their instructors use NAG Library routines in their mathematical, science, computing, and finance courses, to make light work of intensive calculations. They use NAG routines as they are stringently tested, expertly documented, highly flexible and regularly updated. Routines can be accessed through MATLAB, Excel, Visual Basic or via languages such as Java, Python, C, .NET and Fortran. They are also optimized for use in high performance computing and Cloud environments. Unlike many software support centres, NAG Library support is provided by the very people that develop the routines giving users expert help in solving problems and getting the most out of the library.

Read on to discover more on how NAG numerical software can help you.

"I am impressed by the optimization algorithm provided by the NAG Toolbox for MATLAB. It improves the results for my maximum likelihood estimations for situations where the sample size is small causing non-concentrating likelihood and when the likelihood functions have 'ridges'. Ordinary algorithms, for example Newton gradient search, perform poorly in these situations."

Ning Guo, Warwick Finance Research Institute, University of Warwick, UK



The NAG Library

The world's largest collection of robust, documented, tested and maintained numerical algorithms

If you need to add mathematical and statistical functionality to your applications or if you have complex mathematical problems to solve, the NAG Library will provide a host of benefits. The NAG Library provides a solid numerical foundation and serves diverse mathematical areas. It is expertly documented, maintained and supported, and is regularly updated with cutting edge algorithmic capabilities . The NAG Library can be used with: Python, C, C++, Fortran, MATLAB, Java, .NET, Excel and more.

We've selected key highlights from the NAG Library and show in more detail how a particular function or set of functions can be used. To learn more about a specific area/function click on the relevant link below.

What's New?

We have selected the new functionality in the NAG Library and show in more detail how a particular routine or set of routines can be used:

Faster Data Fitting (Calibration): Mini Article, GitHub Examples

Fast Implied Volatilities: Mini Article, GitHub Examples

Solving Convex and Non-convex Quadratically Constrained Quadratic Programming (QCQP) Problems: Mini Article, GitHub Examples

Solving Convex Problems with Second Order Cone Programming (SOCP): Mini Article, **Technical Poster** **&** **GitHub Examples**

Derivative-free Optimization Solver for Calibration Problems: Technical Poster & Mini Article

Flexible Modelling with the NAG Optimization Modelling Suite: Mini-article & Examples

First-order Active-Set Method for Nonlinear Programming: Mini Article, GitHub Examples

Nearest Correlation Matrix: Technical Poster, GitHub Examples **& Mini Article**

Randomized Numerical Linear Algebra (RNLA) Algorithms: Technical Poster

Non-negative Matrix Factorization for Analysing High-dimensional Dataset: Slide Deck & GitHub Examples

More NAG Library key functionality

Algorithmic Differentiation Routines
Derivative-free Optimization for Data Fitting
Struve Functions
Interior Point Method for Large Scale Linear Programming
Interior Point Method for Nonlinear Optimization
Semidefinite Programming (SDP)
Three Body Problem using High-Order Runge–Kutta Interpolation
Mixed Integer Nonlinear Programming
Unscented Kalman Filter
LARS / LASSO / Forward Stagewise Regression
Change Point Analysis
Confluent Hypergeometric Function
Two-stage Spline Approximation to Scattered Data
Multi-start Optimization
Optimization for Non-negative Least Squares
Matrix Functions
Inhomogeneous Time Series
Gaussian Mixture Model
Best subset
Bound Optimization BY Quadratic Approximation
Linear Quantile Regression
Sampling with Unequal Weights
Calling random number generators from a multi-threaded environment
Copulas
Skipping Ahead the Mersenne Twister Random Number Generator
Global Optimization
Partial Least Squares / Ridge Regression
Quantiles
Search routines

NAG develops numerical software for many different computing environments, packages and languages including:

Excel | MATLAB® | Java | Python | R | LabVIEW | C/C++ |

Fortran | SMP | .NET | GPU

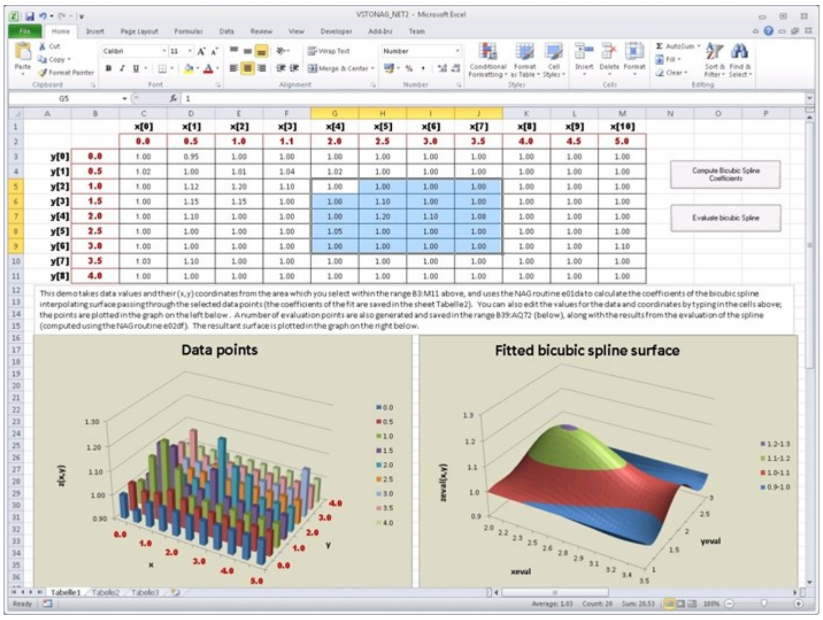
NAG and Microsoft Excel®

expand your capabilities with numerical routines from NAG

- Learn about linking NAG functionality to Excel via the NAG Dynamic Link Libraries
- Open, view and experiment with NAG routine examples from within Excel
- Using the NAG Library for .NET in Excel
- Read white papers and technical reports on using NAG in Excel
- Watch demonstrations showing NAG in Excel functions here

"Accessing my work via the NAG Library gives it a 'seal of approval' and assures people that the code is accurate and correct."

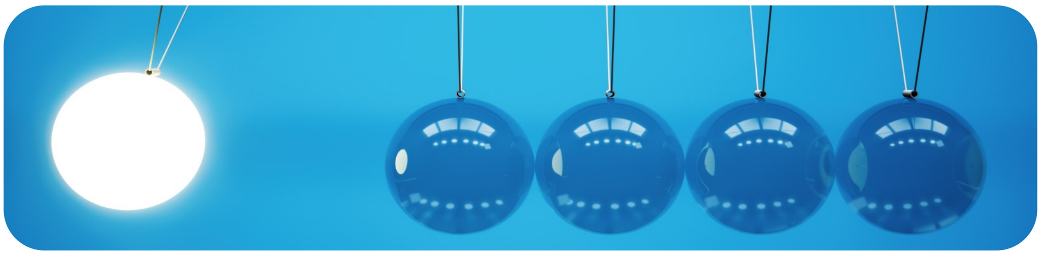
Rebecca Killick,
Lecturer in Statistics,
University of Lancaster



		x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]	x[9]	x[10]
		0.0	0.5	1.0	1.1	2.0	2.5	3.0	3.5	4.0	4.5	5.0
y[0]	0.0	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
y[1]	0.5	1.02	1.00	1.01	1.04	1.02	1.00	1.00	1.00	1.00	1.00	1.00
y[2]	1.0	1.00	1.12	1.20	1.10	1.00	1.00	1.00	1.00	1.00	1.00	1.00
y[3]	1.5	1.00	1.15	1.15	1.00	1.00	1.10	1.00	1.00	1.00	1.00	1.00
y[4]	2.0	1.00	1.10	1.00	1.00	1.00	1.20	1.10	1.08	1.00	1.00	1.00
y[5]	2.5	1.00	1.00	1.00	1.00	1.05	1.00	1.00	1.00	1.00	1.00	1.00
y[6]	3.0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.10
y[7]	3.5	1.03	1.10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
y[8]	4.0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

This demo takes data values and their (x,y) coordinates from the area which you select within the range B3:M11 above, and uses the NAG routine e01da to calculate the coefficients of the bicubic spline interpolating surface passing through the selected data points (the coefficients of the fit are saved in the sheet Tabelle2). You can also edit the values for the data and coordinates by typing in the cells above; the points are plotted in the graph on the left below. A number of evaluation points are also generated and saved in the range B39:AQ72 (below), along with the results from the evaluation of the spline (computed using the NAG routine e02df). The resultant surface is plotted in the graph on the right below.

VSTONAG_NET2 - Microsoft Excel
Compute Bicubic Spline Coefficients
Evaluate bicubic Spline
Data points
z(x,y)
x
y
Fitted bicubic spline surface
zeval(x,y)
xeval
yeval
Tabelle1
Tabelle2
Tabelle3



LabVIEW Using NAG .NET methods in LabView

NAG's numerical routines can be used in the LabVIEW programming environment. There's a lot of help on NAG's website for people wanting to use NAG and LabVIEW together (see http://www.nag.co.uk/numeric/LabView.asp). We've provided various papers, blog posts and 'how to' guides, plus a presentation which gives a self-contained account of how to call NAG Library routines from LabVIEW (http://www.nag.co.uk/numeric/labview/nag_and_labview_general.pdf).

Maths and Stats Routines

for Programming Languages (C/C++, Fortran, Python, R, Java, .NET)

The NAG Library algorithms are inherently flexible – they can be called from an extensive range of languages including C and C++, VBA, Python, Java, .NET and Fortran.

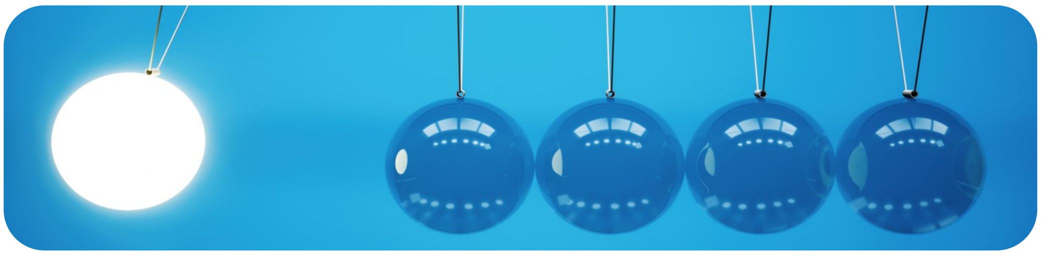
The core NAG Library is provided as a set of generic interfaces:

- The FL interface – the interfaces that utilize only simple types which makes them suitable for calling from multiple languages, including Fortran, C, C++, VBA and others;
- The CL interface – NAG's set of C Library interfaces, and
- The NAG AD Library interfaces to support Algorithmic Differentiation.

The use of NAG Library algorithms enables you to easily switch between programming languages providing heightened flexibility and performance – the algorithms are future-proofed to ensure accuracy and performance.

Available for all these languages and environments

C++ | Python | Fortran | C | Java | MATLAB | C# and .NET | Excel and VBA |
SMP & Multicore | Other languages and environments



Not just libraries—Compilers too!

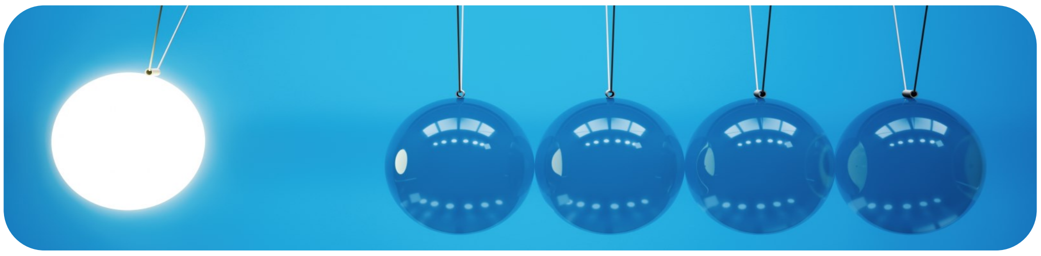
Extensive Fortran Support: parallel programming with coarrays and OpenMP

The NAG Fortran Compiler provides significant support for Fortran 2018 (atomic operations, events and tasks, plus other smaller features), almost all Fortran 2008, complete coverage of Fortran 2003, and all OpenMP 3.1. All platforms include supporting tools for software development: source file polishers, dependency generator for module and include files, call -graph generator, interface builder and a precision unifier.

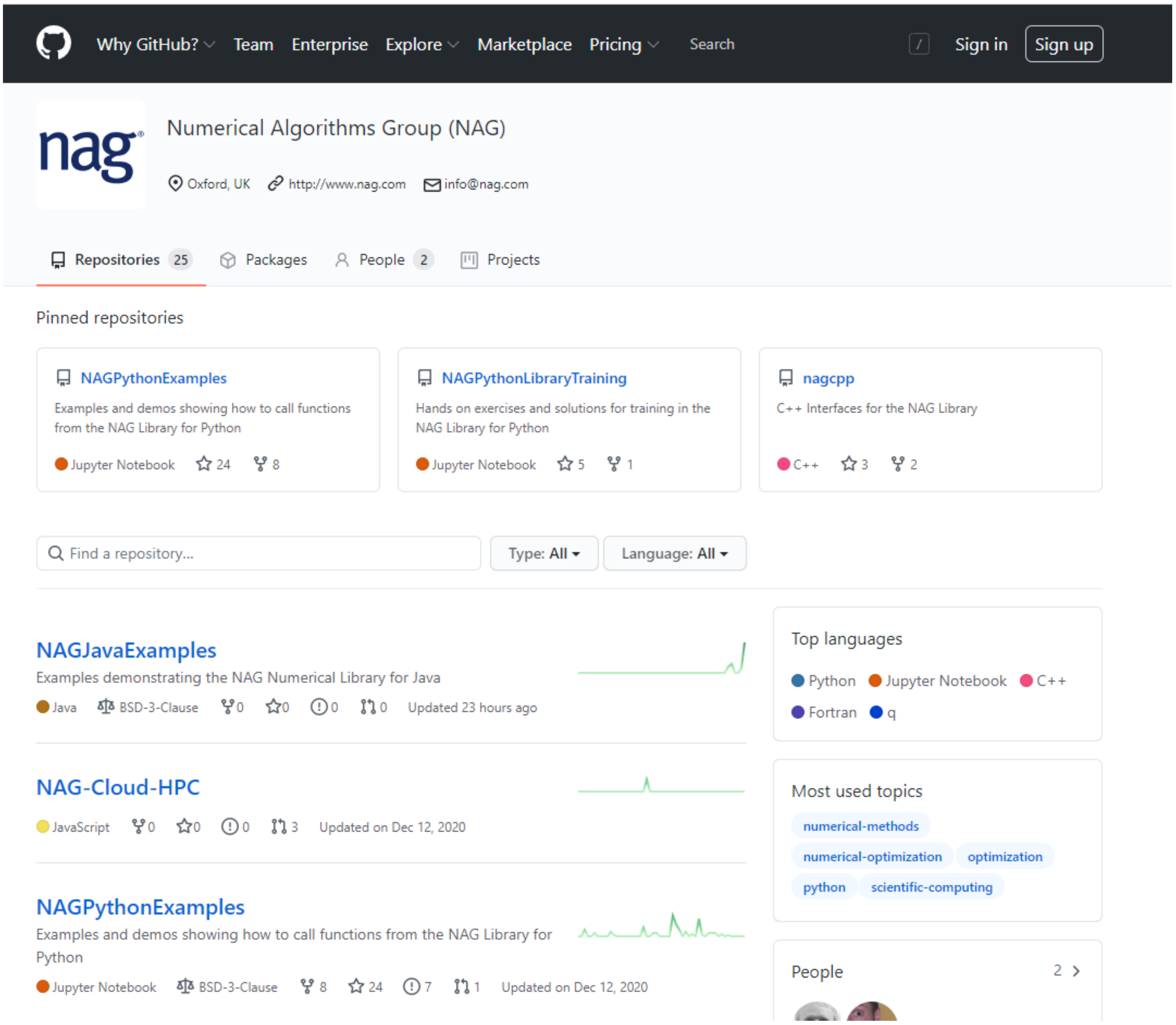
Available on Linux, Windows and macOS, including Apple Silicon Macs. If you prefer using an IDE on Windows or Apple Mac, we recommend using the NAG Fortran Builder.

Latest functionality

- Parallel execution of coarray programs on shared-memory machines;
- Half precision floating-point conforming to the IEEE arithmetic standard, including full support for all exceptions and rounding modes;
- Submodules, a Fortran 2008 feature for breaking large modules into separately-compilable files;
- Teams, a Fortran 2018 coarray feature for structuring parallel execution;
- Events, a Fortran 2018 coarray feature for lightweight single-sided synchronisation;
- Atomic operations, a Fortran 2018 coarray feature for updating atomic variables without synchronisation.
- Plus more….



Visit the NAG GitHub


Why GitHub? Team Enterprise Explore Marketplace Pricing Search Sign in Sign up

Numerical Algorithms Group (NAG)

Oxford, UK http://www.nag.com info@nag.com

Repositories 25 Packages People 2 Projects

Pinned repositories

NAGPythonExamples
Examples and demos showing how to call functions from the NAG Library for Python
● Jupyter Notebook ☆ 24 8

NAGPythonLibraryTraining
Hands on exercises and solutions for training in the NAG Library for Python
● Jupyter Notebook ☆ 5 1

nagcpp
C++ Interfaces for the NAG Library
● C++ ☆ 3 2

Find a repository... Type: All Language: All

NAGJavaExamples
Examples demonstrating the NAG Numerical Library for Java
● Java BSD-3-Clause 0 ☆0 0 0 Updated 23 hours ago

NAG-Cloud-HPC
● JavaScript 0 ☆0 0 3 Updated on Dec 12, 2020

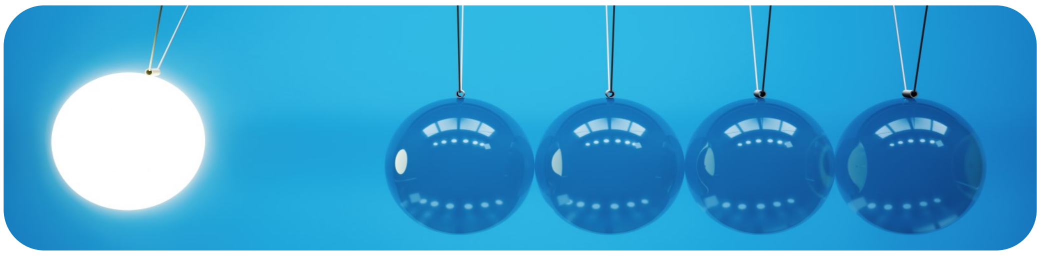
NAGPythonExamples
Examples and demos showing how to call functions from the NAG Library for Python
● Jupyter Notebook BSD-3-Clause 8 ☆ 24 7 1 Updated on Dec 12, 2020

Top languages
● Python ● Jupyter Notebook ● C++ ● Fortran ● q

Most used topics
numerical-methods numerical-optimization optimization python scientific-computing

People 2

https://github.com/numericalalgorithmsgroup



Case Studies

See the benefits that NAG brings to industry and academia

Team Sonnenwagen Optimize Solar Racing Car Aerodynamics Using NAG Algorithmic Differentiation Software Tool - dco/c++

Team Sonnenwagen is a group of students from the Aachen Universities. They participated in the 'World Solar Challenge' (WSC) for the first time in 2017. The WSC is considered the most important race for solar-car teams on the international circuit. 40 teams from five continents compete against each other, all trying to complete a gruelling 3022 km stretch through the Australian outback – all the way from Darwin to Adelaide. The 2019 WSC sees the team bring an additional car to the race in the 'Challenger' class of vehicles, in their quest to capture a Top 3 podium finish. A very intensive development period is currently underway in the hope of the team achieving their mission.

Using the NAG Library to underpin advanced Transport Planning and Optimization.

Andrew Koh, at the Institute for Transport Studies1 , University of Leeds, has long experience in the application of transport planning models and research, both in Europe and Asia, including responsibilities for directing courses on the 'SATURN' network analysis program. He was working in the area of road pricing models when he made use of the NAG Library to solve difficult operational research (OR) problems.

Load balancing issues uncovered by NAG (via the POP project) in a particle tracking application.

The Centre for Environment, Fisheries and Aquaculture Science (Cefas) is a world leader in marine science and technology, providing innovative solutions for the aquatic environment, biodiversity and food security. The centre uses behaviour and transport models to address a range of marine management questions.

They use an off-line particle tracking model that requires velocity fields from a hydrodynamic model, which is known as the Individual Behaviour Model (IBM) GITM (General Individuals Transport Model) code. It includes physical particle advection and diffusion, and biological development and behaviour. The code is implemented in Fortran 90. It was originally sequential then parallelized with OpenMP.

NAG Consulting services quickly provide special quadruple precision routines for satellite sensor simulation

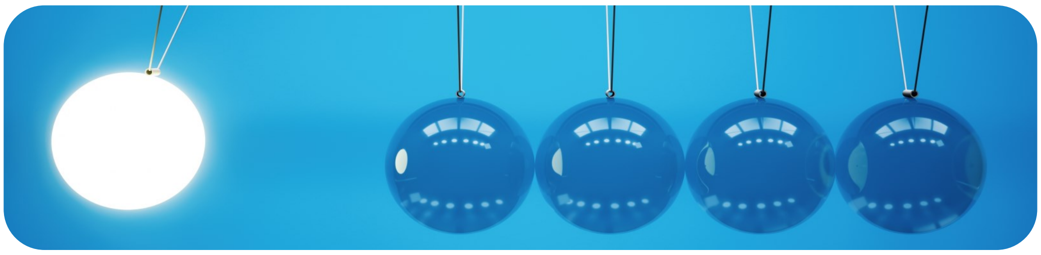
This puzzle to be solved was how to improve the granularity of simulation code to match the greater sensitivity of the new satellite sensors.

NAG HPC Consulting Experts enhance computational fluid dynamics application for optimized ship design.

Cetena is a study centre, based in Italy, that carries out a range of research and consulting work for the maritime industries. NAG, as HPC experts, were asked to investigate the optimal system configuration for the CFD based application to achieve good performance on the chosen HPC cluster platform.

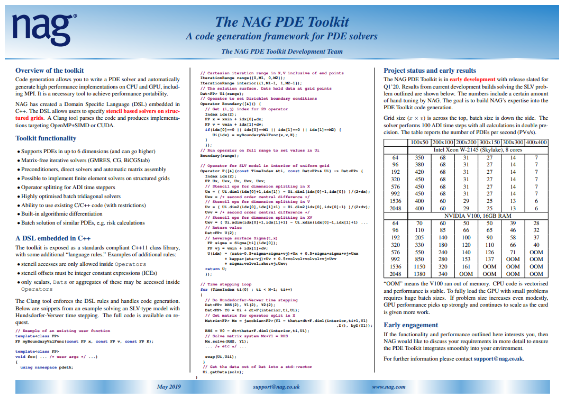
For the full range of case studies visit

https://www.nag.com/content/case-studies-0



Technical Posters

by NAG experts and collaborators



The NAG PDE Toolkit

A code generation framework for PDE solvers

The NAG PDE Toolkit Development Team

Overview of the toolkit

Code generation allows you to write a PDE solver and automatically generate high performance implementations on CPU and GPU, including MPI. It is a necessary tool to achieve performance portability.

NAG has created a Domain Specific Language (DSL) embedded in C++. The DSL allows users to specify stencil based solvers on structured grids. A Clang tool parses the code and produces implementations targeting OpenMP+SIMD or CUDA.

Toolkit functionality

- Supports PDEs in up to 6 dimensions (and can go higher)
- Matrix-free iterative solvers (GMRES, CG, BiCGStab)
- Preconditioners, direct solvers and automatic matrix assembly
- Possible to implement finite element solvers on structured grids
- Operator splitting for ADI time steppers
- Highly optimised batch tridiagonal solvers
- Ability to use existing C/C++ code (with restrictions)
- Built-in algorithmic differentiation
- Batch solution of similar PDEs, e.g. risk calculations

A DSL embedded in C++

The toolkit is exposed as a standards compliant C++11 class library, with some additional "language rules." Examples of additional rules:

- stencil accesses are only allowed inside `Operators`
- stencil offsets must be integer constant expressions (ICEs)
- only scalars, `Dats` or aggregates of these may be accessed inside `Operators`

The Clang tool enforces the DSL rules and handles code generation. Below are snippets from an example solving an SLV-type model with Hundsdorfer-Verwer time stepping. The full code is available on request.

```
// Example of an existing user function
template<class FP>
FP myBoundaryValFunc(const FP x, const FP v, const FP K);

template<class FP>
void foo( ... /* user args */ ...)
{
  using namespace pdetk;
```

```
// Cartesian iteration range in X,V inclusive of end points
IterationRange range({0,M1, 0,M2});
IterationRange interior({1,M1-1, 1,M2-1});
// The solution surface. Dats hold data at grid points
Dat<FP> Ui(range);
// Operator to set Dirichlet boundary conditions
Operator Boundary([&]() {
  // Get (i,j) index for 2D operator
  Index idx(2);
  FP x = xmin + idx[0]*dx;
  FP v = vmin + idx[1]*dv;
  if(idx[0]==0 || idx[0]==M1 || idx[1]==0 || idx[1]==M2) {
    Ui(idx) = myBoundaryValFunc(x,v,K);
  }
});
// Run operator on full range to set values in Ui
Boundary(range);

// Operator for SLV model in interior of uniform grid
Operator F([&](const TimeIndex &ti, const Dat<FP>& Ui) -> Dat<FP> {
  Index idx(2);
  FP Ux, Uxx, Uv, Uvv, Uxv;
  // Stencil ops for dimension splitting in X
  Ux = ( Ui.dim1(idx[0]+1,idx[1]) - Ui.dim1(idx[0]-1,idx[0]) )/(2*dx);
  Uxx = /* second order central difference */
  // Stencil ops for dimension splitting in V
  Uv = ( Ui.dim2(idx[0],idx[1]+1) - Ui.dim2(idx[0],idx[0]-1) )/(2*dv);
  Uvv = /* second order central difference */
  // Stencil ops for dimension splitting in XV
  Uxv = ( Ui.xdim(idx[0]+1,idx[1]+1) - Ui.xdim(idx[0]-1,idx[1]+1) ...
  // Return value
  Dat<FP> U(2);
  // Leverage surface Sigma(t,x)
  FP sigma = Sigma[ti](idx[0]);
  FP vj = vmin + idx[1]*dv;
  U(idx) = (rate-0.5*sigma*sigma*vj)*Ux + 0.5*sigma*sigma*vj*Uxx
         + kappa*(eta-vj)*Uv + 0.5*volvol*volvol*vj*Uvv
         + sigma*volvol*rho*vj*Uxv;
  return U;
});

// Time stepping loop
for (TimeIndex ti(0) ; ti < N-1; ti++)
{
  // Do Hundsdorfer-Verwer time stepping
  Dat<FP> RHS(2), Y1(2), Y2(2);
  Dat<FP> Y0 = Ui + dt*F(interior,ti,Ui);
  // Get matrix for operator split in X
  Matrix<FP> Mx = jacobian<FP>(Y1 - theta*dt*F.dim1(interior,ti+1,Y1)
                                                        ,D(), byD(Y1));
  RHS = Y0 - dt*theta*F.dim1(interior,ti,Ui);
  // Solve matrix system Mx*Y1 = RHS
  Mx.solve(RHS, Y1);
  ... /* etc */ ...

  swap(Ui,Uii);
 }
 // Get the data out of Dat into a std::vector
 Ui.getData(soln);
}
```

Project status and early results

The NAG PDE Toolkit is in early development with release slated for Q1'20. Results from current development builds solving the SLV problem outlined are shown below. The numbers include a certain amount of hand-tuning by NAG. The goal is to build NAG's expertise into the PDE Toolkit code generation.

Grid size ($x \times v$) is across the top, batch size is down the side. The solver performs 100 ADI time steps with all calculations in double precision. The table reports the number of PDEs per second (PVs/s).

	100x50	200x100	200x200	300x150	300x300	400x400
	Intel Xeon W-2145 (Skylake), 8 cores					
64	350	68	31	27	14	7
96	380	68	31	27	14	7
192	420	68	31	27	14	7
320	450	68	31	27	14	7
576	450	68	31	27	14	7
992	450	68	31	27	14	7
1536	400	60	29	25	13	6
2048	400	60	29	25	13	6
	NVIDIA V100, 16GB RAM					
64	70	60	50	50	39	28
96	110	85	66	65	46	32
192	205	140	100	90	58	37
320	330	180	120	110	66	40
576	550	240	140	126	71	OOM
992	850	280	153	137	OOM	OOM
1536	1150	320	161	OOM	OOM	OOM
2048	1380	340	OOM	OOM	OOM	OOM

"OOM" means the V100 ran out of memory. CPU code is vectorised and performance is stable. To fully load the GPU with small problems requires huge batch sizes. If problem size increases even modestly, GPU performance picks up strongly and continues to scale as the card is given more work.

Early engagement

If the functionality and performance outlined here interests you, then NAG would like to discuss your requirements in more detail to ensure the PDE Toolkit integrates smoothly into your environment.

For further information please contact support@nag.co.uk.

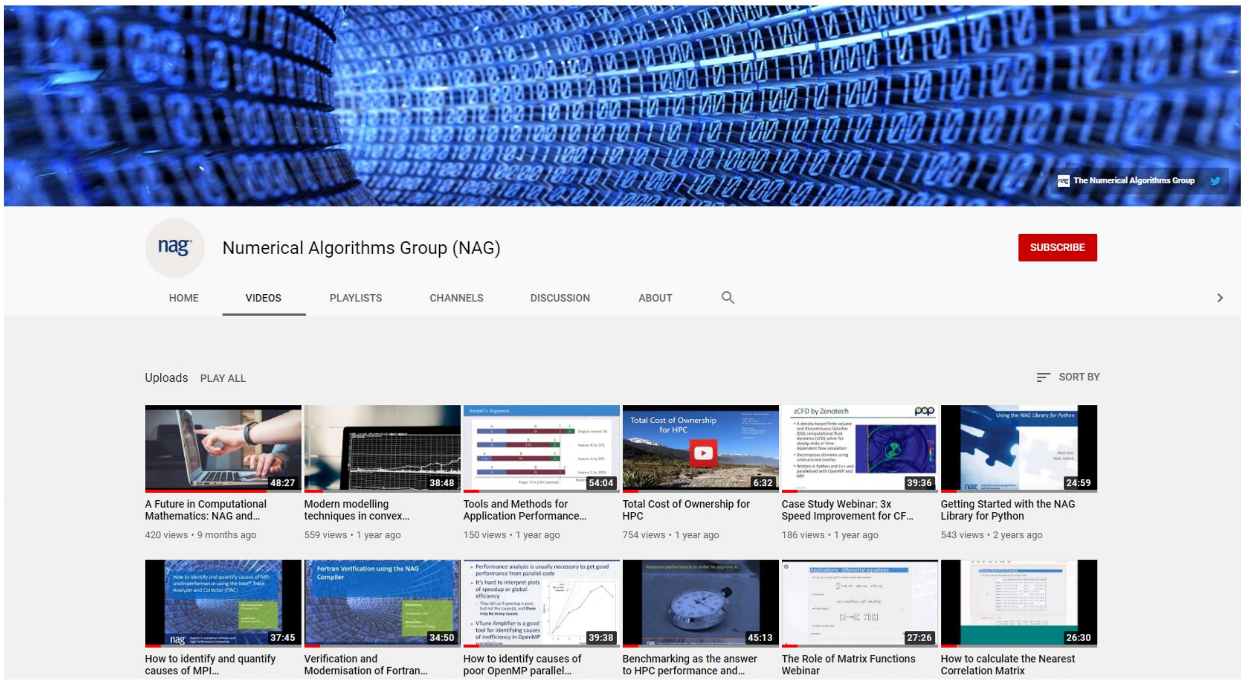
May 2019 *support@nag.co.uk* *www.nag.com*

- The NAG PDE Toolkit
- Nearest Correlation Matrix
- Second Order Cone Programming (SOCP)
- CVA in the Cloud
- Fast Implied Volatilities using Chebyshev Interpolation
- CVA at Scale with Adjoint Sensitivities – Combining the NAG Library with dco/c++ and Origami
- Locally Stationery Wavelet Processes with Trend
- Performance Analysis of GS2 Plasma Turbulence Code
- Performance and Usage Improvements for Multiple OpenFOAM Customers: HPC and Cloud
- Adjoint Algorithmic Differentiation of a GPU Accelerated Application
- High Performance Tape-Free Adjoint AD for C++11
- From Runtime to Compiler Time Adjoints
- Derivative Free Optimization Solver for Calibration Problems
- Matrix Functions and the NAG Library
- Nearest Correlation Matrices
- Algorithmic Differentiation of the Local Volatility Model
- A Finite Volume Scheme for Calibrating Stochastic Local Volatility Models
- Stochastic Grid Bundling Method for Bermudan Swaptions
- Algorithmic Differentiation of Nonsmooth and Discontinuous Functions
- HPC Software Optimization Highlights from the POP Project
- Nonlinear Optimization: A Comparison of Two Competing Approaches
- Batched Least Squares of Tall Skinny Matrices on GPUs
- Parallel Convolution Gridding for Radio Astronomy Applications Running on KNL and GPU
- The Performance Optimization and Productivity Centre of Excellence
- Analysis and Improvement of HPC Software Performance: ADF Modeling Suite

Resources

- [Using the NAG Library *for Python*]
- A Future in Computational Mathematics: NAG and Numerical Analysis
- Modern modelling techniques in convex optimization and applicability to finance and beyond
- Tools and Methods for Application Performance Profiling
- Fortran Verification using the NAG Compiler
- The Role of Matrix Functions
- How to use the NAG Compiler and Fortran Builder
- Working at the Numerical Algorithms Group (NAG)

Visit the NAG YouTube Channel here.



Numerical Algorithms Group (NAG)

48:27 A Future in Computational Mathematics: NAG and... 420 views • 9 months ago

38:48 Modern modelling techniques in convex... 559 views • 1 year ago

54:04 Tools and Methods for Application Performance... 150 views • 1 year ago

6:32 Total Cost of Ownership for HPC 754 views • 1 year ago

39:36 Case Study Webinar: 3x Speed Improvement for CF... 186 views • 1 year ago

24:59 Getting Started with the NAG Library for Python 543 views • 2 years ago

37:45 How to identify and quantify causes of MPI...

34:50 Verification and Modernisation of Fortran...

39:38 How to identify causes of poor OpenMP parallel...

45:13 Benchmarking as the answer to HPC performance and...

27:26 The Role of Matrix Functions Webinar

26:30 How to calculate the Nearest Correlation Matrix