H2020–FETHPC–2014: GA 671633

D5.1
Requirements analysis

April 2017

Document information

Scheduled delivery	2017-04-30
Actual delivery	2017-04-27
Version	1.0
Responsible partner	STFC

Dissemination level

PU — Public

Revision history

Date	Editor	Status	Ver.	Changes
2016-11-11	Iain Duff	Draft	0.1	Draft based on Word template produced by Lennart Edblom
2017-03-24	Iain Duff	Draft	0.2	Draft prepared for internal review
2017-04-27	Iain Duff	Final	1.0	Revision after internal reviews

Author(s)

Iain Duff, STFC

Internal reviewers

Bo Kågström, UMU
Sam Relton, UNIMAN
Olivier Tissot, Inria

Contributors

Laura Grigori, Inria, France
Bernd Klöss, DigSILENT GmbH
Radoslaw Stompor, CNRS, France
Olivier Tissot, Inria, France

Copyright

Acknowledgements

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under the grant agreement number 671633.

Table of Contents

1 Introduction **3**

2 Materials science and chemistry **3**

3 Load flow in large scale power systems **4**

4 Energy solutions and Code Saturne **8**

5 Data analysis, astrophysics, and Midapack **11**

List of Figures

1 Plot of sparsity pattern of typical matrix from power system applications. . 6
2 An example of dynamic reduction of the search directions. 10
3 Communication bottleneck of two algorithms, a dense linear solver based on the LU factorization with partial pivoting (bottom) and a sparse iterative solver applied to the map-making problem in astrophysics (top). 12

List of Tables

1 Influence of threshold parameter, u, on power system matrices. Time in seconds. 5
2 Influence of switching to full code on power system matrices. The switch to dense code happens when the Schur complement has the density shown. Time in seconds. 7
3 Matrices used in our tests, their size, number of nonzeros, and smallest (λ_1) and largest (λ_n) eigenvalues. 9
4 Number of iterations to get the solution. 10

1 Introduction

The *Description of Action* document states for Deliverable D5.1:

> "*D5.1 Requirements analysis*
>
> Report describing the outcome of the requirements analysis for all applications."

This deliverable is in the context of Task–5.1 (Requirements analysis).

The novel numerical algorithms and software developed in NLAFET will be validated and integrated into several challenging real-world applications. This will allow their scalable execution on the emergent hierarchical models of extreme-scale machines, thus facilitating new scientific discoveries and novel industrial solutions.

This deliverable discusses the requirements of the applications described in the proposal in Workpackage 5. For each application, we identify the properties of the linear algebra problems to be solved, the sizes of the problems that will be addressed during the project, their sparsity properties, and the parameters that need to be taken into account at execution. This deliverable serves as input for the implementation in Task 5.2 and also gives input to the work in the research-oriented WPs (WP2–4, WP6). The applications are:

1. Dense solvers/eigensolvers in materials science and chemistry. Collaboration with ETH Zürich, Switzerland
2. Load flow based calculations in large-scale power systems and PowerFactory code. Collaboration with DIgSILENT GmbH, Germany.
3. Energy solutions and Code Saturne. Collaboration with EDF France.
4. Data analysis in astrophysics and Midapack. Collaboration with University Paris 7, France.

We discuss these in more detail in Sections 2 to 5.

2 Materials science and chemistry

The main NLAFET contact is Jack Dongarra and the main applications contact is Thomas Schulthess of ETH Zürich, Switzerland.

We are collaborating with ETH Zürich who in turn interact with researchers at the Swiss National Supercomputing Centre (CSCS) on developing large dense eigenvalue solvers on hybrid architectures. This will provide us with an application platform for validating and integrating novel algorithms in the area of nanostructure materials developed at ETH Zürich.

Scientific computing applications, ranging from computing frequencies that will propagate through a medium, to an earthquake response of a bridge, or energy levels of electrons in nanostructure materials, require the solution of eigenvalue problems. There are many ways to formulate these problems mathematically and solve them numerically.

The Swiss Team has several codes that require a dense eigensystem solution. The main ones are:

- QuantumESPRESSO: nanoscale materials modelling.
- CP2K: molecular dynamics for solid-state/liquid/biological systems.
- VASP: quantum-mechanical molecular dynamics.
- NWCHEM: computational chemistry package.

The major application of our contact at ETH Zürich is in materials design. Here a materials database is used in conjunction with a powerful suite of first principles electronic structure codes that can be run dynamically during the screening process to compute observable quantities specific to the design goals for the material. It is thus important to have very efficient algorithms for the first principle calculation. One way of doing this is the self-consistent field (SCF) approach which requires the repeated solution of the Kohn-Sham differential equation. These are commonly solved using the Linearized Plane Wave (LAPW) method that has, at its core, the solution of a generalized eigenvalue problem with dense complex Hermitian matrices of the form $Ax = \lambda Bx$, where A is a Hermitian dense matrix and B is Hermitian positive definite [1]. That is to say such solvers are needed to solve electronic structure problems in materials science and chemistry. The complexity of the LAPW method is cubic in the number of atoms. Currently, because of work from the Team at ICL in Tennessee [7] systems with over a 1,000 atoms can be solved. For these problems the resulting generalized eigenvalue problem must be solved for a dense, complex Hermitian matrix with dimension of order 10^5. Since in a materials design problem these simulations will have to run on large parallel supercomputers that cannot hold these matrices on individual nodes, the implementation must be designed for distributed memory architectures. The generalized eigenvalue problem, using methods discussed in detail in [5], is solved and the smallest eigenvectors are found. The number of eigenvectors needed ranges from 10% to 50%. It is possible to use an iterative diagonalization technique and reduce the computational cost, however the 'safe default' of major LAPW community codes is to solve the generalized eigenvalue problem with the brute force dense matrix diagonalization. The material scientists would like to solve even larger problems and to solve them faster, as this computation is at the heart of their simulation and must be performed many times.

We will provide new Hermitian-definite eigenvalue problem solvers based on factorization kernels and eigenvalue problem solvers delivered in WP2. We will integrate the software into the application and evaluate the impact on scalability and performance.

3 Load flow in large scale power systems

The main NLAFET contact is Iain Duff and the main applications contact is Bernd Klöss of DIgSILENT GmbH, Germany.

We are collaborating with the Research & Development Department of DIgSILENT GmbH, a leading provider of power system analysis software. With offices in Australia, South Africa, Italy, Spain, France, and Chile, DIgSILENT has software installations and conducts services in more than 130 countries. Their software features, inter alia, calculation functions for the analysis and optimisation of network flows and voltages covering applications in generation, transmission, distribution, industrial systems, and renewable energies.

As more and more power systems are being connected to each other, their users are confronted with growing, large-scale networks. With classical load flow based calculations at its core, this leads to a large number of similar very large linear systems that must be solved fast and accurately. Typically, the systems are highly sparse, unsymmetric, and very ill-conditioned. In order to tackle such problems, their software package PowerFactory features direct and iterative linear solution methods built in-house.

Table 1: Influence of threshold parameter, u, on power system matrices. Time in seconds.

Application	Size	Nonzeros	Nonzeros in LU	Time	Bkwd error
Optimal Power Flow	427069	2378554			
	$u = 0.0001$		6491905	1.21	0.7×10^{-18}
	$u = 0.001$		6808813	1.28	0.1×10^{-18}
	$u = 0.01$		7183064	1.35	0.2×10^{-17}
	$u = 0.1$		7823038	1.55	0.2×10^{-17}
Balanced load flow	197156	805839			
	$u = 0.0001$		1590324	0.23	0.6×10^{-15}
	$u = 0.001$		1645314	0.24	0.4×10^{-16}
	$u = 0.01$		1856003	0.27	0.2×10^{-16}
	$u = 0.1$		1941067	0.28	0.1×10^{-17}
Balanced load flow	19202	128242			
	$u = 0.0001$		201015	.035	0.3×10^{-13}
	$u = 0.001$		210159	.027	0.3×10^{-15}
	$u = 0.01$		227984	.033	0.2×10^{-15}
	$u = 0.1$		250683	.035	0.7×10^{-17}
Unbalanced load flow	202760	2406285			
	$u = 0.0001$		11353348	5.39	0.2×10^{-16}
	$u = 0.001$		10413421	5.29	0.3×10^{-17}
	$u = 0.01$		12260763	6.91	0.1×10^{-16}
	$u = 0.1$		14465918	8.69	0.8×10^{-18}

We will provide new sparse linear system solvers and link them to the PowerFactory code. Their performance and accuracy will then be compared with the existing solvers. Using DIgSILENT's extensive customer database, comparisons will be made based on many real-world power systems for calculations such as classical load flow analysis, AC optimal power flow (OPF) calculations, dynamic time-domain simulations, and modal stability analysis.

There are many different matrix problems arising from this application. These include:

- Eigenvalue computation: generalized ($Ax = kMx$) and standard eigenvalue problems must be solved, where the left and the right eigenvectors for a subset of the eigenvalues are of interest. A simultaneous calculation of both (left and right eigenvectors) would be very interesting for them.

- Solution of nonlinear systems: they must solve many sparse linear systems with the same structure during iterations of a Newton-type algorithm. They currently have both direct and iterative methods available but further enhancement using parallel methods would be welcome.
- Multi-variable optimisation problems: they have to solve linear (linear programming) and non-linear optimisation problems (using various methods, also Newton-type methods). Many sparse linear systems are encountered there as well and a parallel solver would be very welcome, in particular when thinking about areas like security-constrained OPF calculations (SCOPF).
- Solution of linear systems with fixed matrix: moreover, they encounter problems where they have to solve linear systems with a fixed matrix and for a large number of right-hand sides. It would be interesting whether a fast parallel solution process for factorized matrices can compete with their current approaches.
- Solution of large systems of ODEs arising in extensive time-domain simulations. In the integration methods used in this solution, there is a need to solve many sparse linear systems. For example, the simulation might be for 60 seconds with a step size in the millisecond range. A good speed up in this case would be particularly beneficial.

The structure of a typical matrix from this application is shown in Figure 1.

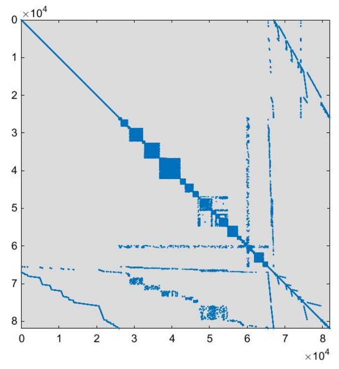

Figure 1: Plot of sparsity pattern of typical matrix from power system applications.

The team at RAL has tested a standard serial code on a range of matrices from these applications. We were interested to see the effect of varying two important parameters, the threshold parameter u and the point at which we switch to using a code for dense matrices. The threshold parameter permits entries to be allowed as pivots only if their modulus is larger than u times the entry of maximum modulus in their column. As the LU factorization of a sparse matrix proceeds, fill-in occurs (that is zeros in the original matrix become nonzero) and at a certain stage it is not beneficial to exploit sparsity so we switch to using a code for dense factorization for the remainder of the factorization. We show the results of these runs on four matrices from different applications in Tables 1 and 2, respectively. All the runs for both tables were performed on an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz which has 4 cores and two threads per core. The code used was HSL_MA48 from the HSL Library, which is a serial code that is essentially a precursor to the parallel code being developed in Task 3.3 of the NLAFET project.

In Table 1 we see that as u is increased, we emphasize stability over sparsity so that the time and number of entries in the factors increase while the backward error (as measured by the scaled residual) decreases although the behaviour is not monotonic. It is noticeable that there is not a huge effect from changing u which differs from other applications, for example for unsymmetric matrices from PDEs, where the effect can be more dramatic. This is even more surprising since the matrices in these tables are very ill-conditioned, with condition numbers around 10^{10} as we saw by comparing the forward error (not in the tables) with the backward error.

Table 2: Influence of switching to full code on power system matrices. The switch to dense code happens when the Schur complement has the density shown. Time in seconds.

Application	Size	Nonzeros	Nonzeros in LU	Time	Size dense matrix
Optimal Power Flow	427069	2378554			
Switch density					
0.2			7513629	1.39	715
0.4			7217808	1.35	399
0.6			7172420	1.36	286
0.8			7144564	1.36	81
1.0			7143398	1.36	14
Balanced load flow	197156	805839			
Switch density					
0.2			1909572	0.28	316
0.4			1863481	0.27	196
0.6			1854652	0.27	153
0.8			1846370	0.27	20
1.0			1846298	0.26	6
Balanced load flow	19202	128242			
Switch density					
0.2			253808	.048	220
0.4			229615	.036	127
0.6			223654	.030	60
0.8			222684	.030	34
1.0			222516	.029	14
Unbalanced load flow	202760	2406285			
Switch density					
0.2			16322021	6.36	2946
0.4			12902340	6.25	1904
0.6			11990647	7.17	1432
0.8			11426628	9.31	227
1.0			11416563	9.29	18

In Table 2 we examine the influence of changing the point at which we change to using a dense code. Again the results for this class of matrices show a different effect from other applications inasmuch as, even with a switch when the density is as low as 0.2 (20% dense) the size of the Schur complement matrix is very small (relative to the order of the original matrix). This is largely due to the fact that, contrary to matrices from discretizations of PDEs, the Markowitz criterion keeps the amount of fill-in very low. One comment can be made on the times: they are generally quite close to constant. This is interesting because the HSL_MA48 analysis and factorization has two quite separate phases. The first factorizes the matrix using a right-looking algorithm but does not keep the factors. Also it stops as soon as we decide to switch to dense code. Thus the lower the switch density, the sooner we switch and thus stop this phase so it becomes faster as the switch density decreases. The second phase uses the pivot order obtained from the first phase to factorize the matrix using a left-looking algorithm and switches to using a dense code at the point determined by the first phase. Thus, as we see from the last column in the table, the size of the matrix on which dense factorization is performed increases monotonically with decreasing switch density. Thus, even though code for dense systems is more efficient, the cost of the dense part of the subsequent factorization increases significantly as it has complexity the cube of the dimension.

4 Energy solutions and Code Saturne

The main NLAFET contact is Laura Grigori and the main application contact is Yvan Fournier from EDF, France.

We are interacting with EDF researchers on validating and integrating our novel algorithms in simulations used at the R&D centre of EDF. The activities at the R&D centre are focused on three top priorities: consolidating a carbon-free energy mix, planning future electricity supply and developing a flexible range of low-carbon energy sources. As part of these simulation activities, EDF develops and distributes a number of codes, many of which are open source. An important fraction of the execution time of these codes is spent in the linear solvers. The progress obtained by our project will potentially impact several simulation activities. For our first test case we will use the CFD solver Code Saturne that solves the Navier-Stokes equations for incompressible flows. This code is developed in-house by EDF and is distributed under the GPL open source licence (available at http://www.code-saturne.org). Code Saturne has been included in the Unified European Applications Benchmark Suite of the PRACE project (see http://www.prace-ri.eu/ueabs). It is used for a wide range of applications, many of which are related to nuclear engineering, but increasingly with applications related to renewables. We will focus on simulations such as the computation of fluid flow in tube bundles, either cross-flow as in steam generators, or tangential as in PWR fuel assemblies, as these applications are numerically quite representative of a broader range of applications, and large-scale meshes for benchmarking are available.

We will provide new sparse linear system solvers and test them in the context of Code Saturne. The global performance and scalability of the code using communication-avoiding methods will then be compared with that using legacy solvers already available in Code Saturne.

During a six-week project, which took place in the context of Cemracs 2016[1], we

[1]For more information please visit `http://smai.emath.fr/cemracs/cemracs16/index.php`

have collaborated with Yvan Fournier and started testing our methods on several small to medium size matrices. In Table 3 we describe these matrices, their size, number of nonzero entries and their largest and smallest eigenvalues.

Table 3: Matrices used in our tests, their size, number of nonzeros, and smallest (λ_1) and largest (λ_n) eigenvalues.

	Size	Nonzeros	λ_1	λ_n
BUNDLE16	16 384	109 184	8.7e-10	9.8e-06
BUNDLE65	65 536	439 552	2.2e-10	9.8e-06
BUNDLE262	262 144	1 763 840	5.5e-11	9.8e-06

We have evaluated the enlarged Krylov subspace methods developed at Inria [2] in the context of WP4. We use a domain decomposition approach to define enlarged Krylov subspaces and the associated *ECG(t)* method. More precisely given a splitting decomposition represented by the operator T,

$$T(x) = \begin{pmatrix} * & & & & \\ \vdots & & & & \\ * & & & & \\ & * & & & \\ & \vdots & & & \\ & * & & & \\ & & \ddots & & \\ & & & * & \\ & & & \vdots & \\ & & & * & \\ & & & & * \\ & & & & \vdots \\ & & & & * \end{pmatrix}$$

and the initial residual r_0, the corresponding enlarged Krylov subspace is defined as

$$\mathcal{K}_{k,t} = \operatorname{span}\{T(r_0), AT(r_0), \dots, A^{k-1}T(r_0)\}, \tag{1}$$

where t is the number of columns of $T(r_0)$ and is called the enlarging factor. Using this definition and the algebraic properties of the Conjugate Gradient method (Petrov-Galerkin condition and A-orthogonality of the search directions), it is possible to derive a so-called *Enlarged Krylov Conjugate Gradient* method [2]. At each iteration, t vectors forming an A-orthonormal basis of the enlarged Krylov subspace are constructed. This increases the arithmetic intensity of the method but can lead to *inexact breakdowns* [6]. It is possible to detect such situations and reduce adaptively the number of search directions without increasing the number of iterations [4].

All the numerical experiments were performed with Matlab 2015b. In Table 4 we summarize the results obtained when running the Preconditioned Conjugate Gradient (PCG) method and the Enlarged Krylov Conjugate Gradient method with an enlarging factor t (ECG(t)). We use the Metis K-Way algorithm to partition the matrix into 1024 domains. The preconditioner is block Jacobi with 1024 blocks. In Table 4 we summarize the results obtained when comparing ECG(t) and PCG without reduction in the number of search directions. ECG(t) is always more effective than PCG in terms of iteration count, almost half the number of iterations when $t = 4$ and as much as a quarter the number of iterations when $t = 32$. As a consequence, ECG(t) scales better than PCG when the matrix size increases. In Figure 2 we show the results obtained when comparing ECG(t) and BCG(t) with dynamic reduction of the search directions. BCG(t) is the block Conjugate Gradient method where the first right-hand side is b and the $t-1$ others are chosen randomly. Although the number of iterations is of the same order for BCG(32) and ECG(32), the number of search directions is more effectively reduced with ECG(32).

Table 4: Number of iterations to get the solution. The method stops when the relative residual is lower than 10^{-6}. PCG is the usual preconditioned Conjugate Gradient method and ECG(t) is the Enlarged Krylov Conjugate Gradient method where t is the enlarging factor. The system is preconditioned with a block diagonal preconditioner with 1024 blocks. When increasing the enlarging factor the number of iterations decreases, half the number of iterations for ECG(32) compared to ECG(4) and around 4 times less compared to PCG.

Matrix	PCG	ECG(4)	ECG(8)	ECG(16)	ECG(32)
BUNDLE16	172	103	79	63	53
BUNDLE65	207	117	92	66	53
BUNDLE262	325	180	127	93	70

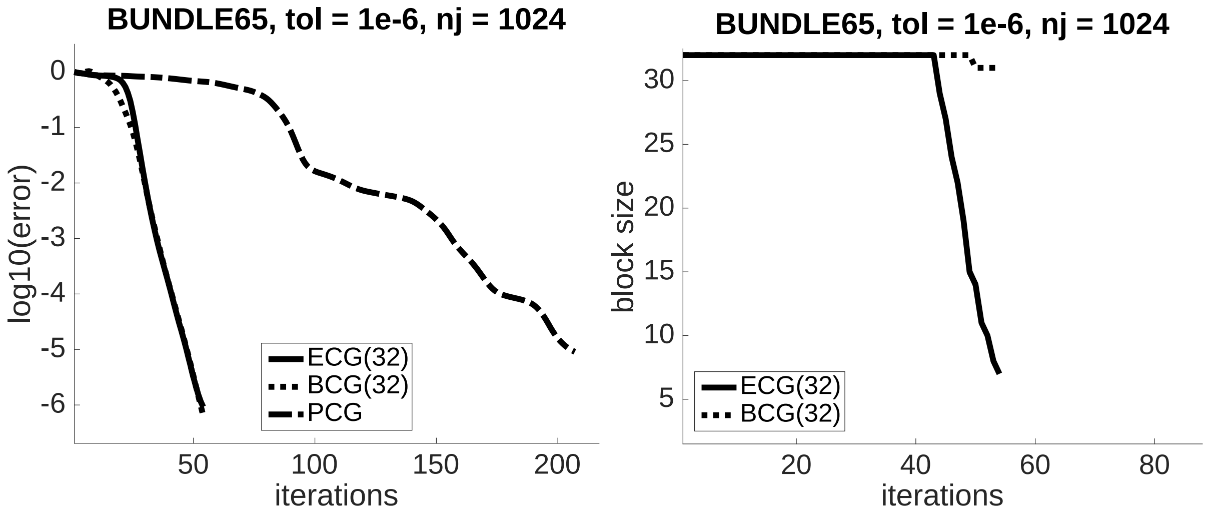

Figure 2: An example of dynamic reduction of the search directions. We terminate the iterations when the relative residual is lower than 10^{-6}. The system is preconditioned with a block diagonal preconditioner with 1024 blocks. BCG(t) denotes a block Conjugate Gradient method with t right-hand sides, the first is b and the other are random vectors. The dynamic reduction of the search directions does not increase the number of iterations (54 against 53) although ECG reduces the number of search directions more effectively.

5 Data analysis, astrophysics, and Midapack

The main NLAFET contact is Laura Grigori and the principal application contacts are Radek Stompor from APC/CNRS, France, and Carlo Baccigalupi of SISSA Italy.

Studies of the minute fluctuations of the intensity and polarization of primordial photons, called CMB for Cosmic Microwave Background, have been one of the main sources of invaluable information about our universe and the fundamental laws of physics. Astrophysicists produce and analyse multi-frequency 2D images of the universe when it was 5% of its current age. The new generation of CMB experiments observe the sky with thousands of detectors over many years, producing overwhelmingly large and complex data sets. For example, Planck is a keystone satellite mission which has been developed under the auspices of the European Space Agency (ESA). Planck has been surveying the sky since 2010. It produces Terabytes of data and requires 100 Petaflops to compute each image of the universe. This next generation of observational efforts promise to revolutionise our understanding in these areas. This information has to be extracted from data sets collected by increasingly more ambitious efforts by observatories, which over the last two decades have kept producing data sets with volumes consistently growing at Moore's Law rate. One of the main computational challenges in the CMB data processing involves reconstructing a 2-dimensional map of the sky from these data sets. This problem, called a map-making problem, is difficult for a number of reasons, including data volumes, long-term noise correlations and presence of parasitic signals, inter alia. With current algorithms, the reconstruction of the sky maps from the data available by early 2020 will require 100 Exaflops. In the analysis of any CMB data, the map-making step may need to be performed hundreds if not thousands of times as, for instance, is the case whenever popular Monte Carlo sampling algorithms are used to characterize the uncertainties of the estimated sky maps.

The map-making problem can be formulated as a generalized least-squares problem with non-trivial weights, leading to the following linear system,

$$(\mathbf{A}^T\mathbf{N}^{-1}\mathbf{A})\,\mathbf{x} = \mathbf{A}^T\mathbf{N}^{-1}\,\mathbf{d}. \tag{2}$$

Here $\mathbf{d}$ stands for the vector of observations, with dimension n_t between 10^{13} and 10^{15}. The weight matrix, $\mathbf{N}^{-1}$, has rank $n_t \times n_t$ and is structured, e.g., block Toeplitz, or diagonal plus low rank correction, and the matrix $\mathbf{A}$, with n_t rows and up to the order of 10^6 columns, is very sparse with only a handful of nonzeros per row.

Given the size of the system matrix, $\mathbf{A}^T\mathbf{N}^{-1}\mathbf{A}$, the solution, $\mathbf{x}$, is most readily obtained using iterative algorithms, and is facilitated by the fact that efficient methods for multiplying a vector by the matrices $\mathbf{A}$ and $\mathbf{N}^{-1}$ exist and have been implemented in the MIDAPACK Library[2]. The volumes of the data and the requirement to process the entire data set in one step in order to deal with and properly account for the long-term noise correlations, requires use of the largest available supercomputers with their complex architectures and communication networks and appropriate numerical algorithms.

The goal of this project is to adapt and validate communication-avoiding iterative methods in the context of the CMB map-making problem and to make them available to the CMB community by integrating them with the MIDAPACK package. Previous experiments show that the communication is the bottleneck that prevents scaling the map making problem to a very large number of processors. Figure 3 displays the performance of preconditioned conjugate gradients used currently for solving the generalized least

[2] `http://www.apc.univ-paris7.fr/APC_CS/Recherche/Adamis/MIDAS09/software/midapack/ver1.1/index.html`

squares problem arising in the map making problem. This performance result is extracted from [3][3] where a more detailed description of the algorithms can be found. It shows the cost of a single iteration of the conjugate gradient iterative solver preconditioned by a block diagonal preconditioner, together with the time spent on computation and communication. These runs were performed on a Cray XE6 system; each node of the system is composed of two twelve-core AMD MagnyCours. It can be seen that the communication becomes quickly very costly, potentially dominating the runtime of the solver when more than 6000 cores are used (each MPI process uses 6 cores).

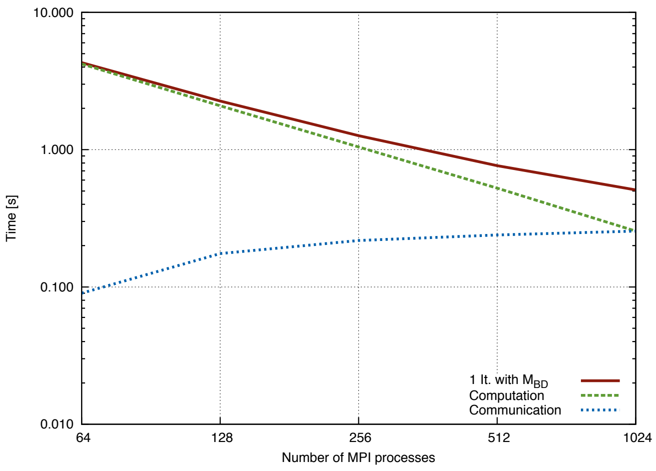

Figure 3: Communication bottleneck of two algorithms, a dense linear solver based on the LU factorization with partial pivoting (bottom) and a sparse iterative solver applied to the map-making problem in astrophysics (top).

We will focus on novel preconditioning techniques, which are suitable for single runs but also on preconditioners appropriate for multiple sequential solves with different right-hand sides, where the information from the previous runs is accumulated and exploited in the forthcoming runs or a precomputation is used to construct a better and more efficient preconditioner. In either case, the methods we are seeking should not merely lead to a decrease in the number of required iterations but should also lead to reducing the time-to-solution. Thus the required preconditioners need to be cheap to construct and apply in a massively parallel context appropriate for modern computer architectures.

References

[1] T. Auckenthaler, V. Blum, H. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. Willems, *Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations*, Parallel Computing, 37 (2011), pp. 783–794.

[2] L. Grigori, S. Moufawad, and F. Nataf, *Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication*, SIAM Journal on Scientific Computing, 37 (2016), pp. 744–773. Also as INRIA TR 8266.

[3] Courtesy of M. Szydlarski

[3] L. Grigori, R. Stompor, and M. Szydlarski, *A parallel two-level preconditioner for cosmic microwave background map-making*, Proceedings of the ACM/IEEE Supercomputing SC12 Conference, (2012).

[4] L. Grigori and O. Tissot, *Reducing the communication and computational costs of Enlarged Krylov subspaces Conjugate Gradient*, *NLAFET Working Note* WN-13, April, 2017. Also as INRIA Research Report 9023, Project Team Alpines, France.

[5] A. Haidar, S. Tomov, J. Dongarra, R. Solcà, and T. C. Schulthess, *A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks*, International Journal of High Performance Computing Applications, 28 (2014), pp. 196–209.

[6] M. Robbé and M. Sadkane, *Exact and inexact breakdowns in the block GMRES method.*, Linear Algebra Appl., 419 (2006), pp. 265–285.

[7] R. Solcà, A. Kozhevnikov, A. Haidar, S. Tomov, T. C. Schulthess, and J. Dongarra, *Efficient implementation of quantum materials simulations on distributed CPU-GPU systems*, in The International Conference for High Performance Computing, Networking, Storage and Analysis (SC15), Austin, TX, 11-2015 2015, ACM, ACM.