D3.3

Second release of MaX software: Report on the evolution actions taken in each of the codes, and software release

Luigi Genovese, Stefano Baroni, Augustin Degomme, Pietro Delugas, Stefano de Gironcoli, Andrea Ferretti, Alberto Garcia, Paolo Giannozzi, Anton Kozhevnikov, Andrea Marini, Davide Sangalli, Daniele Varsano, and Daniel Wortmann

Due date of deliverable	30/11//2020 (**month 24**)
Actual submission date	02/12/2020
Final version	02/12/2020
Lead beneficiary	CEA (participant number 5)
Dissemination level	PU - Public

Document information

Project acronym	MAX
Project full title	Materials Design at the Exascale
Research Action Project type	European Centre of Excellence in materials modelling, simulations and design
EC Grant agreement no.	824143
Project starting/end date	01/12/2018 (month 1) / 30/11/2021 (month 36)
Website	http://www.max-centre.eu
Deliverable no.	D3.3

Authors	Luigi Genovese, Stefano Baroni, Augustin Degomme, Pietro Delugas, Stefano de Gironcoli, Andrea Ferretti, Alberto Garcia, Paolo Giannozzi, Anton Kozhevnikov, Andrea Marini, Davide Sangalli, Daniele Varsano, and Daniel Wortmann.

To be cited as	Genovese et al. (2020): Second release of MaX software: Report on the evolution actions taken in each of the codes, and software release. Deliverable D3.3 of the H2020 CoE MAX (final version as of 02/12/2020). EC grant agreement no: 824143, CEA, France.

Disclaimer

Contents

1 Executive Summary **4**

2 Introduction **5**

3 Algorithmic Activities **6**
3.1 BigDFT 7
3.2 SIESTA 8
3.3 QUANTUM ESPRESSO 10
3.4 YAMBO 12
3.5 FLEUR 15
3.6 CP2K 16
3.7 SIRIUS software development platform 16

4 Conclusions and ongoing work **18**

References **18**

1 Executive Summary

MAX has an entire work package (WP) devoted to "*Code Evolution: exploiting algorithmic advantages enabled by the exascale transition*". It deals with the implementation of algorithmic solution and functionalities in the MAX flagship codes, in order to best exploit (pre-)exascale architectures. In the previous deliverables of WP3 we have presented the identified "algorithmic advances" in the form of a software development plan (D3.1) and a first year release report (D3.2). The former includes a list of the planned algorithms for each code, a classification of each algorithm according to selected criteria, and a timeline in the form of a Gantt chart.

In this document we present the actual status and the advancement of these actions after two years of development. We do this on a per-code basis. We also highlight how the above activities are expected to provide/receive input to/from other technical WorkPackages of MAX . Since this deliverable is meant to report the activities of the consortium at mid-term, for the sake of consistency, we also remind the reader with some of the concepts that have been introduced in the Software Development Plan and previous reports.

2 Introduction

WP3 addresses the progressive adaptation in terms of newly available algorithms and calculations of physical observables of MAX flagship codes (and in turn of the related research community) to the advent of pre- and exascale machines. As discussed in the Software Development Plan document (D3.1 [1]), the algorithmic advances targeted by WP3 may be (and in many cases are) inter-connected to the activities of other WorkPackages, e.g. via requirements on libraries or code restructuring, or by the extreme scale demonstrators they may enable. Such dependencies represent pre-requisites that need to be taken into account when defining or updating the schedule of the activity for each algorithm/feature (action) to be implemented. To each of these pre-requisites, there is a corresponding level (Stage) of complexity in the activities of WP3. For the sake of completeness, we here briefly list how the stages are defined.

Stage1: No prerequisites, direct implementation. The algorithmic actions to be implemented do not have relevant dependencies on libraries or code restructuring, and can be directly addressed.

Objectives: hitting the "wall" of the limitations of present-day paradigms, showing what can be effectively done with the codes "as they are" today. Such stage is associated to a set of new functionalities that are added *on top* of existing codes, with pre-exascale awareness, yet still *without* significantly altering the underlying infrastructure of the codes.

Stage2: Mild prerequisites. This Stage addresses algorithms and physical features that only have limited dependencies on WP1/WP2 libraries, and may require minor code restructuring at the global level.

Objectives: showing that the consortium is able to identify and promote new user experiences, that connect HPC and novel approaches for the end-user.

Stage3: Strong prerequisites and code restructuring. In this stage, long term and extended actions (algorithms/features) are addressed.

Objectives: here the idea is to bring awareness of how to conceive the software codes in order to simplify its usage and its maintenance. The objectives are prototype examples that should suggest how the codes have to deal with the unavoidable process of fine-tuning to make different libraries/modules working together. Such stage of actions is certainly enabled by WP1 outcomes, in the sense that some actions may explicitly employ the libraries released in WP1.

The Stages presented above are associated to the complexity of the actions. The WP tasks, instead, are associated to the objective to which each activity is related. The tasks are defined as follows:

T3.1 Software-related fault resilience algorithms and solutions: with this task, we intend to *prevent* a potential waste of computing time due to incorrect behaviour of the code or incorrect setup of the input parameters. Actions associated to this task are aiming at, for instance, detecting runtime misbehaviour and providing a fallback solution, or correcting code weaknesses in certain regimes; for example,

by preventing the code to run in a regime for which a selected functionality has not been conceived.

T3.2 Enabling new code functionalities with an exascale mindset: here the goal is to try new solutions that are made possible by the advent of new supercomputers, that have up to now been considered unfeasible or impractical.

T3.3 Exploitation of new algorithms at the pre-exascale: Here the idea is to "think differently" the development actions, knowing the features that are nowadays available in the exascale era. High thread concurrency and increasing computational workload will be the runtimes associated to these algorithms. Actions in this tasks will pave the way towards novel investigation paradigms and solutions.

Of course, there is a interplay between the tasks and the progressive difficulty of the corresponding actions. For example, it is likely that some actions of T3.3 would be associated to Stage 3. We have provided the overview of the panorama of code activities in a table, which has been presented in D3.1 as well as in the project mid-term report.

3 Algorithmic Activities

In the following we present, for each code consortium, a detailed description of the WP3 activities focused on the development and implementation of algorithmic advances in MAX flagship codes and libraries, as occurred in the period M12-M24 (Dic 2019 - Noc 2020). This document is therefore an update of the M12 document D3.2 [2].

It is important to remark that the development of each algorithm or feature identified in WP3 has a strong connection with software development activities from WP1 and WP2. For instance, libraries modularised by WP1 may represent inputs to WP3 or may be the container for the output of the algorithmic developments, while newly developed algorithms may require GPU porting and represent therefore input for WP2 work. Similarly, WP3 actions in some cases may provide new code features useful for the activities of WP5 and WP6, or may receive from these WPs a list of requirements and needs. We have presented and summarised these connections in the revised version of the deliverable D3.1 [1] Software Development Plan.

Besides a description of each planned action, for each code, we provide the relevant portion of the WP3 Gantt chart, which illustrates the status of the algorithmic actions that were organised in the software development plan. When needed, Gantt charts have been updated according to the current status of development of each action. A per-code presentation of the main achievements follows.

3.1 BigDFT

B1 Input-file manipulation and handling: wildcard approaches.

This task has been completed in due time, and its behaviour described in D3.2 [2].

B2 Mixed-precision techniques for convolutions and Poisson Solver.

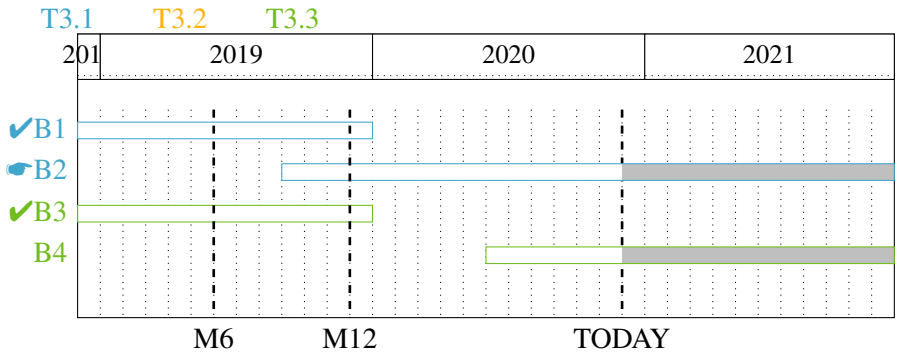
The best way to deal with mixed precision has been identified in the convolution wrapper (a wrapper to libconv). The Poisson Solver has been restructured in its 1.9 version and it is ready to be refactored in such a way to handle mixed precision FFTs. It is still not yet clear how to deal with reduction operations and explicit manipulations that are needed in the intermediate steps of the PSolver workflow. The objective is to reduce code intrusivity by minimising the number of duplicated lines.

In the forthcoming months we plan to have a working wrapper for libconv and identify the most convenient strategies for PSolver. This task will be prolonged until the end of the project as we foresee several back-and-forth reconsideration of the best strategies according to the development experience.

B3 Usage of Pseudo-Fragment approaches for extended systems in the Support Functions formalism.

The task has been completed in advance with respect to the software development plan. Its description can be found in D3.2 [2].

B4 Exact exchange for $\mathcal{O}(N)$ implementation. We have started to prototype the implementation of the exact exchange implementation, thanks to the workflows that can be defined in PyBigDFT. Presently, we plan to employ this workflow to understand if a Support Function basis extracted with a semilocal functional like PBE can be employed in view of searching a ground state for a hybrid functional like PBE0. This workflow will be employed also a reference implementation that can be employed to understand the potential crossover point of a more complicated, fully $\mathcal{O}(N)$ implementation based on explicit four-point integral calculation.



3.2 SIESTA

S1 Basis-set contraction.

The work to implement this feature is still in the planning stages. The actual implementation is being delayed pending the incorporation of a new postdoc, expected for January 2021.

S2 Exact-exchange.

The reference implementation reported in M12 has been considerably improved, and profiled. We have identified some data-redesign actions that would be needed in order to further improve performance.

S3 Convergence improvements

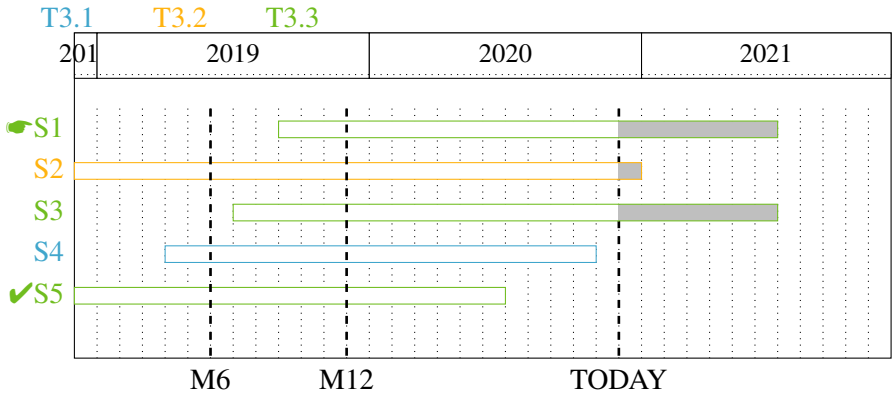
We have in place the basic infrastructure of Lua hooks that can enable arbitrary remediation recipes for lack of convergence or slow convergence, and an associated proof-of-concept recipe. As a production-level demonstrator is implemented in the next stage, it can impact the overall degree of successful completion of calculations, with particular relevance for the high-throughput workflows in WP5.

S4 Break-even points for various solvers and automatic dispatch based on learned heuristics.

Work has already started taking advantage of the recently implemented interface to the ELSI library of solvers. A uniform interface is available to the PEXSI, NTPoly, and OMM (cubic scaling version) solvers, as well as to diagonalization (ELPA). We are designing a new "performance model" along the lines of that developed for Quantum Espresso. Work in this direction can now begin thanks to a newly arrived post-doc.

S5 Re-design of the legacy $\mathcal{O}(N)$ solver.

This task (re-design, implementation, and initial testing) has been completed. Testing for performance in medium-to-large systems is ongoing. This action reimplements the linear-scaling Orbital Minimization Method (OMM) with localized electron orbitals, using the sparse matrix-matrix multiplication library DBCSR from WP1/WP2. The interface has been abstracted through the intermediate gluing library MatrixSwitch. This work can impact the WP6 demonstrators that deal with gapped systems, and also serve as the starting point for the implementation of alternate linear-scaling solvers that reuse the sparse matrix support backend.


T3.1
T3.2
T3.3
201
2019
2020
2021
S1
S2
S3
S4
S5
M6
M12
TODAY

3.3 QUANTUM ESPRESSO

Q1 Improved diagonalization algorithms.

The task has been successfully completed. The ParO (Parallel Orbital) update scheme, as well as a Projected Preconditioned Conjugate Gradient (PPCG) and Residual minimisation-direct inversion in the iterative subspace (RMM-DIIS) scheme, were implemented. Improvements over the previous implementations include: reduction of memory requirement, elimination of redundant evaluations of the Hamiltonian and minimisation of communication time. The efficiency of these additions has been field-tested and verified. These algorithms are being developed in the context of the code-agnostic `KS_Solvers` driver collection. A GPU-accelerated version is available.

Q2 Direct energy minimisation schemes.

High throughput applications rely on the robust performance of the underlying quantum engine. The fast convergence of the self-consistent cycle scheme used in typical use cases is plagued by occasional failures of the procedure in a small fraction of the cases (in the order of a few percent). Alternative, more robust minimisation strategies based on global optimisation of the energy functional are highly desirable as fall-back solutions (even if possibly less efficient). Work on the implementation of global functional optimisation in the QUANTUM ESPRESSO main engine is underway, in collaboration with the CAS group in Beijing mentioned in D3.2 [2].

A preliminary implementation, for norm-conserving pseudopotentials, has been developed. An extension to the more complex ultrasoft and PAW cases is currently underway. Initial tests on non-critical systems (insulators) show good performance of the adopted preconditioning for the wavefunction update; the critical metallic system case, involving the need to optimise additional and very different variables describing orbital occupation, is being addressed. Preliminary tests on a very limited number of metallic systems show promising performance, but the implementation is not yet ready for the release. A number of "difficult" test cases, met during a recent high-throughput materials screening, has been collected and will be used as test-bed for further tuning and development as soon as the ultrasoft and PAW extension of the algorithm becomes available.

Q3 RPA-based advanced exchange and correlation functionals.

Work is on schedule and in line with the workplan. The current QUANTUM ESPRESSO development version has undergone an analysis of the code-refactoring needed to fully exploit high-level parallelisation in the imaginary frequency integration, with the final aim of enabling scalability to pre-exascale machines. Code refactoring has started. Note that this activity was meant to end at M36 (Nov 2021) but was scheduled for M30 in the Gantt table by mistake. Schedule has been updated.

Q4 Extension of the localised inner-projection method to EXX.

Work is ongoing on ideas for improving the convergence properties of the method. In the current implementation, the capability to achieve very small convergence

threshold may be limited by the presence of orbitals having a marginal overlap oscillating over or below the threshold. We are testing different methods to optimise the convergence path. Some limited delays (about 6 months) with respect to the schedule planned at M6 have occurred.

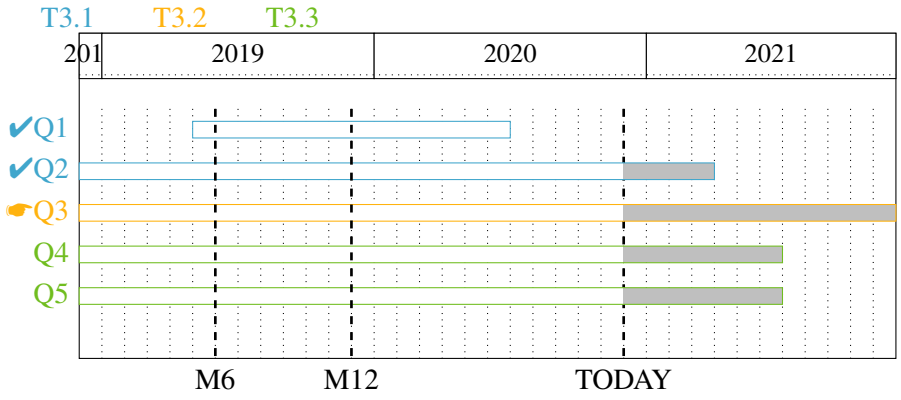
Q5 Adaptive parallelisation schemes.

The activities on adaptive parallelization schemes are ongoing. The first implementation will be done and released as a python utility for the selection at startup of the optimal choice of parallelization parameters. The present algorithm takes into account only a small number of parameters: the machine side is characterised only by the available MPI tasks and the available memory per task; the calculation side is characterised only by the FFT grid dimension, the number of **k**-points, the number of bands and the estimated needed memory per task.

Currently, the application checks that:

- Required and available memory per task are compatible
- The number of used pools is a divisor of the number of **k**-points;
- The number of MPI tasks assigned to each pool allows for an efficient performance of FFT. At the moment the tasks per pool must be at least 1/3 of the z dimension of the FFT grid;
- If the number of MPI tasks is a multiple of 1/3 of the z dimension of the FFT grid, task group parallelism is also used depending on the number of the bands.

More work is ongoing for including more parameters in the model, optimising and making them more flexible using machine-learning techniques. We are also working at the reorganisation of parallelism initialisation and management on MPI groups inside QUANTUM ESPRESSO, in order to be able to implement these algorithms directly at run time.



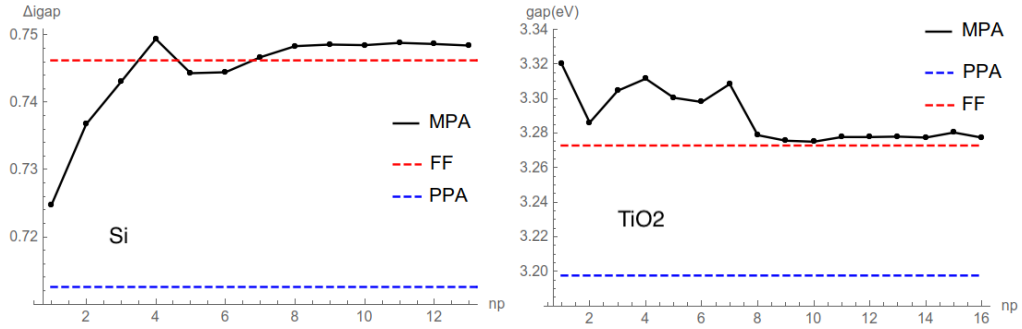


Figure 1: Numerical validation of the multi-pole approximation (MPA) used by YAMBO to represent the dynamical dependency of the screened Coulomb potential W, in order to evaluate the frequency convolution needed to compute the GW self-energy. Results for bulk Silicon and rutile TiO_2 are shown in the left and right panels, respectively.

3.4 YAMBO

Y1 Restart structure, parallel IO and database re-organization. This action was already almost completed at M12. Nevertheless, the activity was taken forward by refactoring the IO of the DIPOLE kernel, now supporting parallel IO via HDF5 and on-the-fly expansion over the full Brillouin zone after being read (instead of before writing as it used to be).

Y2 Exploitation of mixed precision algorithms. This task has been completed in due time, the use of mixed precision having been exploited in different parts of YAMBO as described in D3.2 [2]. At the moment we have considered mixed single/double precision. Nevertheless we plan (1) to further extend the current strategy to newly developed algorithms and (2) to possibly explore the use of even lower precision (e.g. half precision) if suitable algorithms of YAMBO were identified.

Y4 Advanced approaches for full-frequency GW. The activity on the multi-pole approximation (MPA) representation of the screening potential has been developed and implemented in YAMBO. The technique relies in an optimal sampling strategy in the complex plane and it leads to an accuracy in the calculation of quasiparticle energies (electronic gap) comparable with full-frequency methods at much lower costs. The method has been validated, and two examples are shown in Fig. 1, were can be seen that about a number of ten poles (20 frequency sampling) are enough to obtain an accuracy comparable with the full frequency technique which superior than the commonly used plasmon-pole approximation (PPA) for Silicon and TiO_2 crystals. The work is in line with the expected plan, and a scientific paper describing the algorithm in details is in preparation. Work is still ongoing on the validation of the method for different class of systems such as isolated molecules and interfaces, which are prototypical cases where the PPA it is expected to fail and the new implementation will be an valuable solution to reach the desired accuracy at a lower cost than full frequency methods (e.g. real-axis integration or contour deformation techniques) . The scheme will be included in the YAMBO release as soon the last validating tests will be assessed.

Y5 Advanced self-energies from MBPT.

- **Beyond-GW methods.** This task is ongoing. We currently have considered self-energies beyond GW methods, as second Born approximation and multiple flavors of second order screened exchange SOSEX approximations. Such advanced approximations have been implemented and validated calculating ionization potentials for a subset of spherically symmetric atoms. In the next months we plan to validate further the approximations, also considering the calculations of spectral functions and finally, depending on the results obtained, we will explore the feasibility to implement the most advanced self-energies for 3D systems.
- **GW self-energies plus environment.** We have finalized the implementation that allows the calculation of quasiparticle energy in GW framework in presence of a dielectric environment (e.g. solvent) via a polarisable continuum model (PCM). This has been achieved by interfacing YAMBO with the ENVIRON module,[1] already available in QUANTUM ESPRESSO. At the moment we are validating the implementation by considering known approximations as the Onsager reaction field model, later we will move onto validation for realistic cases.

Y7 Real time parallelisation. This activity was completed in advance due to re-prioritization. The implementation put in place at M12 has been extensively tested and numerically validated. Further work along these lines may consider the paralleization over different time-steps in a multi step solution of the equation of motion. Nevertheless, this approach would require a significant amount of coding for a relatively small impact on the performance of the code. As such, this implementation will be performed only if strictly necessary.

Y8 Convergence accelerator for k/q grids: 2D materials. Quasiparticle calculations in the GW approximation are known to be particularly challenging for 2D materials since a very dense $\mathbf{q}$-sampling of the Brillouin zone is needed to obtain properly converged quantities. This difficulty is due to to the sharp $\mathbf{q}$-dependence of the dielectric function in the long-wavelength limit ($\mathbf{q} \to 0$), which is well described only using very large $\mathbf{k}$- and $\mathbf{q}$-grids. We have overcome this issue by developing frm scratch a new algorithm consisting in combining Monte Carlo integration techniques with interpolation schemes of the screened potential W. An example of the performance of this new algorithm is shown in Fig. 2, where the convergence of the gap of a MoS_2 monolayer with respect to the adopted $\mathbf{k}$-point grid of the new algorithm is compared to the traditional approach. A speed-up of at least one to two orders of magnitude in terms of number of $\mathbf{k}$-points used is found. As shown, coarse grids are enough to obtain very accurate results, opening the way to studying materials having a large number of atoms in the unit cell.

The code activities Y3 (*Real–time propagation with atomic motion and interface with QE*) and Y6 (YAMBO *without empty states*) are tightly related to the interfacing of YAMBO with core libraries of QUANTUM ESPRESSO. These activities have been

[1] http://www.quantum-environment.org/

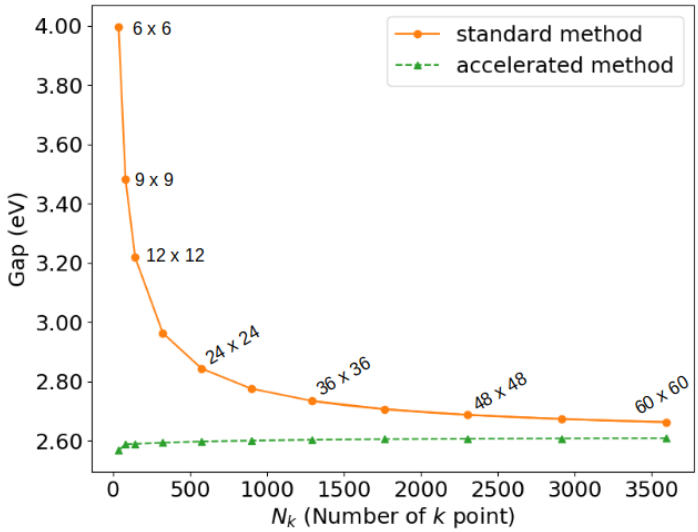
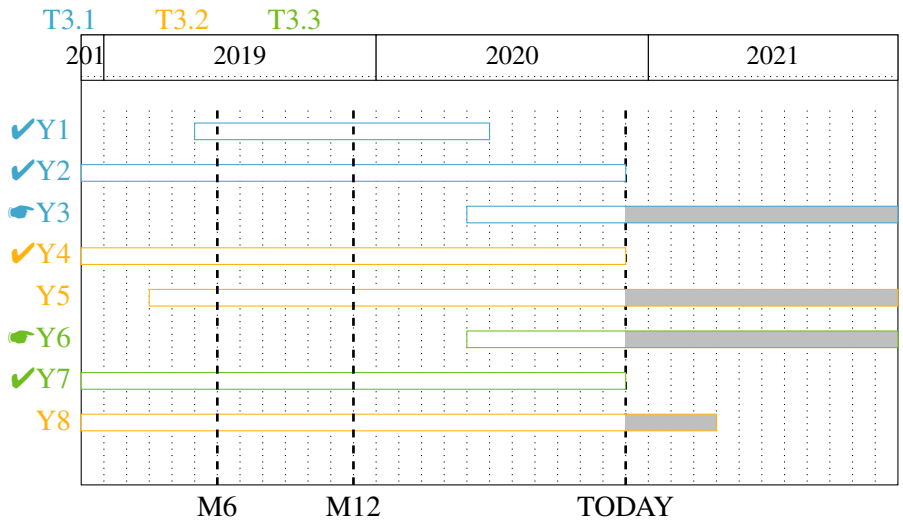


Figure 2: Validation of the newly developed and implemented algorithm to accelerate GW convergence wrt **k**-point sampling. The approach, currently targeting 2D materials but not necessarily limited to them, is applied here to a MoS_2 single layer.

rescheduled and slightly delayed to better synchronize with other YAMBO and QUANTUM ESPRESSO development actions (such as those related to GPU porting, WP2, and code restructuring WP1). The Gantt chart below has been updated accordingly.



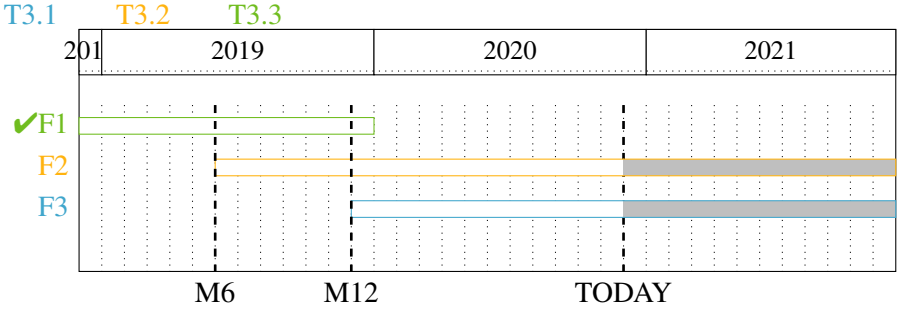
3.5 FLEUR

Out of the implementation tasks related to the FLEUR code, the focus of the current release lies on advances of our activities F2 and F3. While these developments are not finalized yet, we can already provide users with enhanced options for the calculation of hybrid functionals and for controlling the convergence of the self-consistency cycle.

F1 Finishing the restructuring of the Hamiltonian setup into high-level operations.
This task has been completed in due time, and its description can be found in D3.2.

F2 Evaluation of the Coulomb kernel in LAPW. Our refactoring and extraction of fundamental operations used in evaluating the Coulomb kernel has made significant progress. One of the main task in such calculations is the projection of the LAPW wavefunctions onto the optimized product basis used to evaluate the Coulomb matrix. The routines for this task have been substantially reworked and the underlying algorithm was changed to use the convolution theorem and optimized FFTs. Similarly, the code performing the final back-transform from the product basis was redesigned and now maps the operations to explict matrix-matrix operations allowing further encapsulation of the compute intensive operations. As further and final steps we plan to restructure the data handling as well as the interfacing of optimized, performance-portable kernels. Hence, we foresee a slow shift of our activities from the more algorithmic work done here to performance related tasks as described in WP2 and finally an inclusion of the results into the efforts of LAPWlib as described in WP1.

F3 Improved charge-density mixing schemes. As reported in D3.2 we implemented the Kerker preconditioner to increase convergence speed in metallic systems in FLEUR. In addition to the implementation into the bulk code we now extended this scheme for film calculations. This effort proved to include rather difficult as the standard Kerker scheme tends to shift charge into the vacuum region in an unphysical process due to its suppression of long-range charge transfer. Hence, we implemented a modified algorithm which aims at treating the vacuum region in a different manner. However, the resulting difficulties in ensuring the critical property of charge conservation lead to a rather complex preconditioner. While example calculations demonstrated the principal power of the method, details of the correct parameterization for a bigger set of test-systems must still be evaluated and corresponding experience must be acquired. This work might also lead to a possible extension of the method to other inhomogenous systems containing both a conductive and an insulating material.



COSMA: REPRESENTATIVE MULTIPLICATION IN ISOLATION (1 OUT OF 46 PDGEMMs IN CP2K)

BEFORE OPTIMIZATOIN

	total pdgemm	multiply	reshuffle	other
time [s]	9.5	5.88	3.24	0.38
% total time	100.00	61.89	34.11	4.00
total pdgemm = multiply + reshuffle + other				

AFTER OPTIMIZATION

	total pdgemm	multiply	reshuffle	other
time [s]	5.6	5.06	0.53	0.01
% total time	100.00	90.36	9.46	0.18
total pdgemm = multiply + reshuffle + other				

CP2K: FULL CP2K RPA 128 MOLECULES ON 128 NODES (ALL 46 PDGEMMs + OTHERS)

BEFORE OPTIMIZATION

	totl cp2k	total pdgemm	multiply	reshuffle
time [s]	985	439.95	272.75	167.2
% total time	100.00	44.66	27.69	16.97
total pdgemm = multiply + reshuffle				

AFTER OPTIMIZATION

	total	pdgemm	multiply	reshuffle
time [s]	781.6	257.99	233.2	24.79
% total time	100.00	33.01	29.84	3.17
total pdgemm = multiply + reshuffle				

Figure 3: Data reshuffling optimization of COSMA library. The data reshuffling overhead in the 128-node runs on Piz Daint was reduced from 16.97% to 3.17% eliminating the need to work on the native COSMA layout in CP2K.

3.6 CP2K

In the M19-24 of the project the work on CP2K was focused around task C2 (transform CP2K matrices directly to COSMA layout without a need of a ScaLAPACK storage). The results of the work are collected in two tables (Fig. 3). In the initial benchmarks the data reshuffling was taking too much time, rising the concern if the native COSMA data layout has to be supported by CP2K. This was an unfavorable solution due to a complexity of COSMA data partitioning: the communication-optimal algorithm divides data in a non-trivial way, depending on the number of MPI ranks and M,N,K sizes of the matrix multiplication problem. However, it was discovered that the data reshuffling can be considerably optimized. This work is now encapsulated in a standalone project named COSTA[2] and used in the new round of benchmarks. With the COSTA algorithm in place, the data reshuffling overhead dropped from 34.11% to 9.46% in a single *pdgemm* call and from 16.97% to 3.17% in a total CP2K RPA run of 128 water molecules. With this work the data reshuffling becomes inexpensive and the need to implement native COSMA layout in CP2K vanishes.

The other tasks (C1 – integrate COSMA library into CP2K using an intermediate ScaLAPACK matrix layout and C3 – switch to COSMA in RPA calculations) have been successfully finished.

3.7 SIRIUS software development platform

The domain-specific SIRIUS software development platform aims at providing efficient algorithms for the DFT total energy minimization. The work continues to implement the direct solvers for the wave-function optimization. The following optimizers for the plane-wave pseudopotential method have been implemented:

- orbital transformation method for insulators
- direct minimization for ensemble density-functional theory

The optimizers have been successfully tested on few structures where the classical density mixing scheme doesn't converge. The following actions have been completed:

[2] https://github.com/eth-cscs/COSTA

U1 Prototype and implement conjugate gradient method.

This is an implementation of direct minimization for ensemble electronic structure calculations in which the update operator for the electronic orbitals takes the structure of the Stiefel manifold into account. In this method the optimization scheme for the occupation numbers ensures that the constraints remain satisfied. The prototype code was written in Python and tested on several structure where conventional mixing fails.

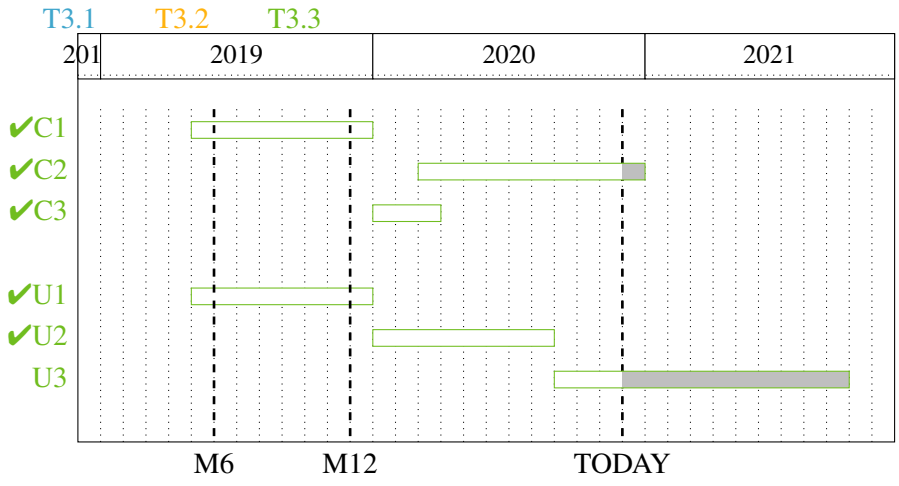
U2 Prototype and implement proximal gradient method.

This is an implementation of proximal gradient method for ensemble density functional theory which is suitable for metallic systems. The prototype will be done in Python using SIRIUS bindings.

The tasks U1, U2 and partially U3 are implemented in the form of a standalone library, written in C++ with the Kokkos backend for performance portability. The library is located on a GitHub repository.[3] The SIRIUS interface to NLCGLIB was also implemented.

The following actions from the U3 task remain to be finished:

U3 Interface advanced density optimizers with QUANTUM ESPRESSO and CP2K. Use already existing interfaces SIRIUS $\Longleftrightarrow$ NLCGLIB and SIRIUS $\Longleftrightarrow$ QE, CP2K. This will allow QUANTUM ESPRESSO and CP2K codes to call the optimizers from NLCGLIB through SIRIUS and find the ground states of systems which cannot be converged using standard mixing techniques.



[3] https://github.com/simonpintarelli/nlcglib

4 Conclusions and ongoing work

In this document we have presented the algorithmic development and implementation activities of the MAX flagship codes and libraries, reporting on their current development status and the related plans for the next year. As already pointed out in the previous deliverables (D3.1 and D3.2, [1, 2]), such activities are deeply connected with the actions of other Workpackages, in particular WP1 and WP2 for what concerns their implementation, and WP5 and WP6 for their outcome or requirement indications. The interested reader may refer to the description presented in the D3.1 deliverable for a more detailed presentation of the connection of each algorithmic action of WP3 with the other WPs.

All code development teams have reported significant achievements concerning the activities of the past year (M12-M24, Dic 2019 - Nov 2020), in some cases completing some of the planned algorithmic improvements, in other cases pushing the development status ahead. A few new algorithmic advances have been identified and added to the development plan. Overall, the activities of WP3 proceed as expected. The pandemic emergency brought some delay for a few actions, but with mild effects. Rather, the code developers had the possibility to have a more focused period working on the improvement of their development approach and strategy (tools and programming models, repositories, development meetings, team communication, to name a few).

In the next year of MAX WP3 activity, we aim at: (i) finalising the actions which are presently ongoing, including interactions with WP1 for what concerns code and library modularisation and WP2 concerning performance portability and porting to accelerated architectures; (ii) evaluating the overall impact of WP3 algorithms in the codes, understanding in particular the adaptation of each code to the newly available HPC architectures; (iii) working in close connection with WP5 (data handling) and WP6 (demonstrator) to make sure the newly developed algorithms and features can be exploited within those activities, e.g. when running high throughput simulations using MAX codes or in the scientific grand challenges selected as MAX demonstrators.

References

[1] Genovese, L. *et al.* Report on identified algorithmic advances, and their software development plan. Deliverable D3.1 of the H2020 CoE MaX (final version as of 31/05/2019; revised version Dic 2020). EC grant agreement no: 824143, CEA, France. (2019).

[2] Genovese, L. *et al.* First release of MAX software: report on the identified actions, update of the software development plan, and software release. Deliverable D3.2 of the H2020 CoE MaX (final version as of 30/11/2019). EC grant agreement no: 824143, CEA, France. (2019).