28-03-2022

D4.2 – Report on algorithms for exascale robustness (fault tolerance and large-scale communications) in QMC flagship codes

Version 1.0

GA no 952165

Dissemination Level

- ☒ PU: Public
- ☐ PP: Restricted to other programme participants (including the Commission)
- ☐ RE: Restricted to a group specified by the consortium (including the Commission)
- ☐ CO: Confidential, only for members of the consortium (including the Commission)

Document Information

Project Title	Targeting Real Chemical accuracy at the EXascale
Project Acronym	TREX
Grant Agreement No	952165
Instrument	Call: H2020-INFRAEDI-2019-1
Topic	INFRAEDI-05-2020 Centres of Excellence in EXascale computing
Start Date of Project	01-10-2020
Duration of Project	36 Months
Project Website	https://trex-coe.eu/

Deliverable Number	D4.2
Deliverable title	D4.2 – Report on algorithms for exascale robustness (fault tolerance and large-scale communications) in QMC flagship codes, as seen in GA
Due Date	M18 – 31-03-2022 (from GA)(from GA)
Actual Submission Date	28-03-2022

Work Package	WP4 – Workflows for HTC and HPDA solutions, algorithms, and toolkits
Lead Author (Org)	Ali Alavi (MPI)
Contributing Author(s) (Org)	Anthony Scemama (CNRS), Vijay Gopal Chilkuri (CNRS), Abdallah Ammar (CNRS)
Reviewers (Org)	Ivan Stich (IPSAS), Pablo de Oliveira Castro (UVSQ)
Version	1.0
Dissemination level	PU
Nature	Report
Draft / final	Final
No. of pages including cover	vii (front matter), 17 (main text), II (back matter)

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Disclaimer

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

The content of this document does not represent the opinion of the European Union, and the European Union is not responsible for any use that might be made of such content.

Versioning

Version	Date	Authors	Notes
1.0	28-03-2022	Ali Alavi (MPI)	First Official Release

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Abbreviations

API	Application Programming Interface
CPU	Central processing unit
GPU	Graphical processing unit
CI	Configuration Interaction
DMC	Diffusion Monte Carlo
HPC	High Performance Computing
MO	Molecular Orbital
MPI	Message passing interface
PDMC	Pure Diffusion Monte Carlo
QMC	Quantum Monte Carlo
TCP	Transmission Control Protocol
TC	Transcorrelated
VMC	Variational Monte Carlo
WP	Work Package

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Table of Contents

Document Information i

Disclaimer ii

Versioning iii

Abbreviations iv

Table of Contents v

List of Figures vi

List of Tables vii

1 Introduction 1

2 Fault-tolerance 2
- Implementation in existing codes 2
- Exascale applications 4

3 Asynchronous Diffusion Monte Carlo algorithm 5
- Real space QMC 5
- Determinant-space QMC 10
  - Bosonic wavefunctions 10
  - Fermionic wavefunctions 11

4 Low-memory algorithm for wavefunction optimization 13

5 Summary 17

References I

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

List of Figures

1 Design of the QMC=Chem application. A *data server* collects the results computed by the *compute processes* (qmc). One *forwarder* process per node aggregates the data to reduce the communications. 2
2 Distributed Quantum Monte Carlo (QMC) simulation among the LAL cloud, the IPHC cloud and the CALMIP supercomputer. The plot represents the number of computed blocks, proportional to the number of Monte Carlo samples, as a function of time. A network failure was simulated by stopping the tunnelling process between CALMIP and the internet. 3
3 Design of the sparse matrix-vector multiplication in the Davidson algorithm implemented in Quantum Package. A *master* OpenMP simulation runs on a single node. Multiple *slave* MPI/OpenMP simulations can be connected using ZeroMQ to the master simulation to increase the computational power. 4
4 Communication pattern of the asynchronous DMC algorithm. A compute node requests walker coordinates (dashed line), the server fulfills the request (dashed line). Then, when branching occurs the compute node sends coordinates to the server (plain line). The request of the next batch of coordinates is made in advance, and the walker coordinates are transferred while the compute node is computing the trajectories of the previous batch. 9
5 Statistical errors of $\left\langle \frac{D_I e^{-J}}{\Psi} E_L \right\rangle$ and $\left\langle \frac{D_I e^{-J}}{\Psi} (E_L - E_0) \right\rangle$ in a simulation of the water molecule with 400 000 Slater determinants. 16

List of Tables

1 Convergence of the energy (a.u.) for the water molecule and the fluorine dimer........ 16

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

1 Introduction

We expect exascale machines to enable QMC applications on larger systems than those that can be treated today. This implies that systems will have larger numbers of electrons, and/or larger Configuration Interaction (CI) expansions. In this Work Package (WP), we investigate ways to overcome new difficulties that will arise when running exascale simulations.

Exascale machines will often be used to run simulations that can't run on smaller systems. So the computed data will be particularly valuable to users, and it should not be lost by accident during the simulation. In addition, an exascale machine will be such a complex piece of hardware and software that it is not reasonable to neglect system failures in the design of dedicated software. The first section of this document discusses different strategies used to make simulations robust to system failures.

QMC methods can be trivially parallelized by running independent random walks to reduce the statistical noise, and with the present supercomputers, the statistical error bars can already be reduced below the chemical accuracy by taking advantage of embarrassing parallelism. However, before including the statistical samples in the computation of the averages, the trajectories need to have reached the ergodic regime in which the sampled density has converged to the target stationary density. The time required to reach the ergodic regime is proportional to the length of each independent trajectory, so using more trajectories in parallel is not a solution to this issue. One should instead aim at reducing the number of walkers per node as much as possible when the size of the system increases, and possibly to use single-node parallelism within a single trajectory.

Exascale systems will be hybrid architectures, and exascale simulations will need to take advantage both of the Central processing units (CPUs) and Graphical processing units (GPUs). The main difficulty in using efficiently such machines is data transfer between the main memory to the memory of the GPUs. If some transfer is required at each Monte Carlo step, the time required for data transfer is likely to be comparable to the time required for the computation of the step, leading to poor performance. Two solutions are envisioned in the high-performance implementation of the QMCkl library (in WP 3). The first solution is the use of asynchronous task-based parallelism using the StarPU[1] library, handling dynamically the scheduling of tasks on CPUs and GPUs. This solution will be adopted when the system under study is so large that very few trajectories can be run on each compute node. The second solution, which will be adopted for most common simulations, is to consider the GPUs as autonomous computing entities that will be able to run a complete QMC simulation without any communication with the host's memory. In this scheme, a distributed simulation will be composed of pure CPU workers, and pure GPU workers. As these computing units are different hardware, the trajectories will not evolve at the same speed, and we will need to introduce the possibility to handle de-synchronized trajectories.

Another problem that will arise is the saturation of the memory. Indeed, when the number of parameters in the wavefunction grows (number of Molecular Orbitals (MOs), number of CI coefficients), the memory required to sample the derivatives required for wavefunction optimization will grow accordingly. In addition, the memory on GPU accelerators is relatively small compared to the memory that is available on a CPU host. If one aims at avoiding transfer from the host memory to the accelerator, one needs to design algorithms with reduced memory requirements. In the last section, we present such an algorithm.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

2 Fault-tolerance

Implementation in existing codes

Most of the algorithms in QMC simulations can be expressed as completely independent Monte Carlo simulations, which can take advantage of embarrassing parallelism. If all the simulations are different realizations of the same process, the stochastic averages are computed as averages over all the independent simulations. Hence, if some arbitrary processes crash during the simulation the number of Monte Carlo samples will be reduced but the statistical averages will not be biased. In theory, such algorithms are expected to be very robust to technical failures, making them excellent candidates for extreme scale simulations.

The communication standard for High Performance Computing (HPC) is the Message passing interface (MPI) Application Programming Interface (API) which was initially designed for more tightly coupled algorithms. As a consequence, some choices in the design of MPI justified by performance considerations prevent the natural exploitation of fault tolerance which is possible with Monte Carlo algorithms. For example, if a single MPI process is killed the whole simulation dies. In the second version of MPI, some primitives have been added to enable client-server communications (`MPI_Open_port`, `MPI_Connect, ...`), allowing to couple multiple independent MPI jobs to communicate together. However, this feature has not been used extensively and offers very limited possibilities compared to what can be achieved with the conventional Transmission Control Protocol (TCP) socket API.

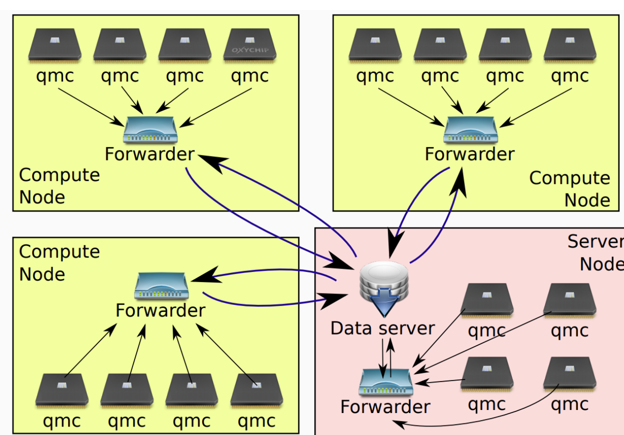

Figure 1: Design of the QMC=Chem application. A *data server* collects the results computed by the *compute processes* (`qmc`). One *forwarder* process per node aggregates the data to reduce the communications.

In 2011, when the QMC=Chem code was being designed,[2] a client/server model was chosen (Figure 1) using usual TCP socket communications to enable fault tolerance and to take advantage of grid computing facilities. Although the latency of TCP communications is more than an order of magnitude higher than with MPI, this has not proven to be a severe problem since all the critical communications can be made non-blocking. A few years later, the TCP sockets were replaced by using the more advanced ZeroMQ library.[3] The use of ZeroMQ is quite unusual in the domain

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

of HPC, but in other communities it is a widely adopted standard for building scalable distributed applications.

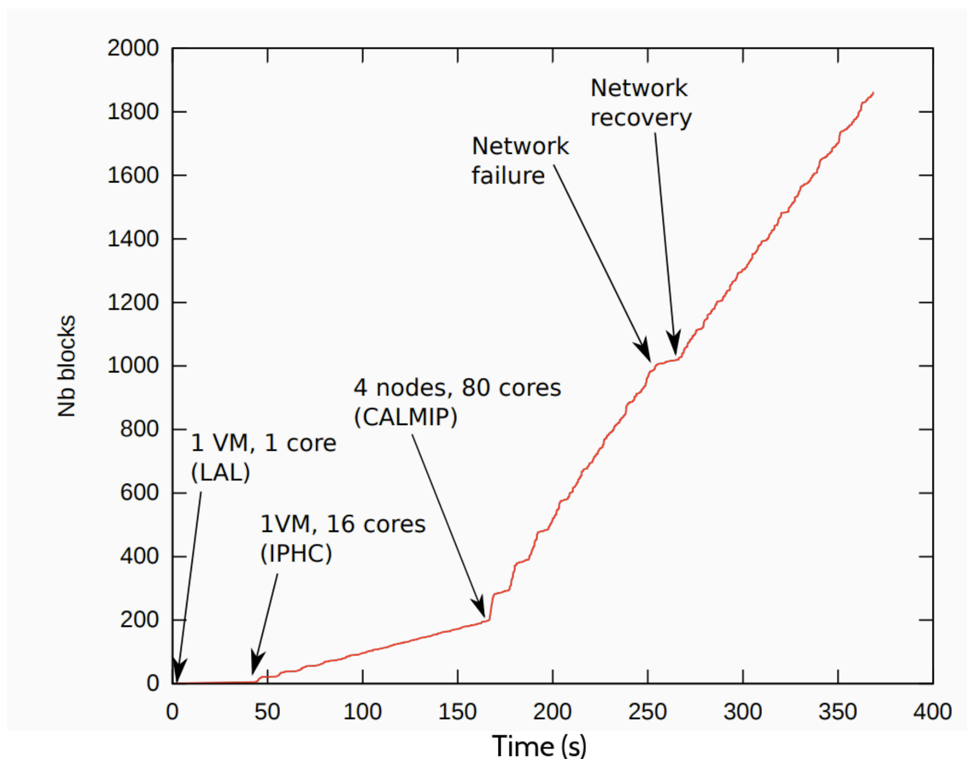

Figure 2: Distributed QMC simulation among the LAL cloud, the IPHC cloud and the CALMIP supercomputer. The plot represents the number of computed blocks, proportional to the number of Monte Carlo samples, as a function of time. A network failure was simulated by stopping the tunnelling process between CALMIP and the internet.

In 2015, a demonstrator QMC calculation was distributed between a supercomputing center and two cloud providers in three different cities in France.[4] We have shown that the calculation ran successfully, that the amount of resources could be adjusted dynamically on demand, and that the simulation could also survive failures, albeit at the expense of the loss of some Monte Carlo samples (Figure 2).

A few years later, we have experimented the same technique in the Quantum Package code.[5] Quantum Package is not a QMC code, but a code used for deterministic wave function calculations. The two hot spots are i) the application of second-order perturbation theory to select relevant Slater determinants and ii) the diagonalization of the Hamiltonian with Davidson's algorithm.[6]

For the perturbative selection, we have designed a stochastic algorithm to enable massive parallelism and fault tolerance,[7] and used a client/server approach similar to the one used in QMC=Chem. For the Davidson diagonalization, the memory is replicated on the compute nodes and the large sparse matrix-vector product at the heart of the Davidson method is split into tasks. The data server holds the list of tasks, provides them to the compute processes and collects the pieces of the resulting vector computed by the compute processes. In this context, the amount of data to be exchanged is

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

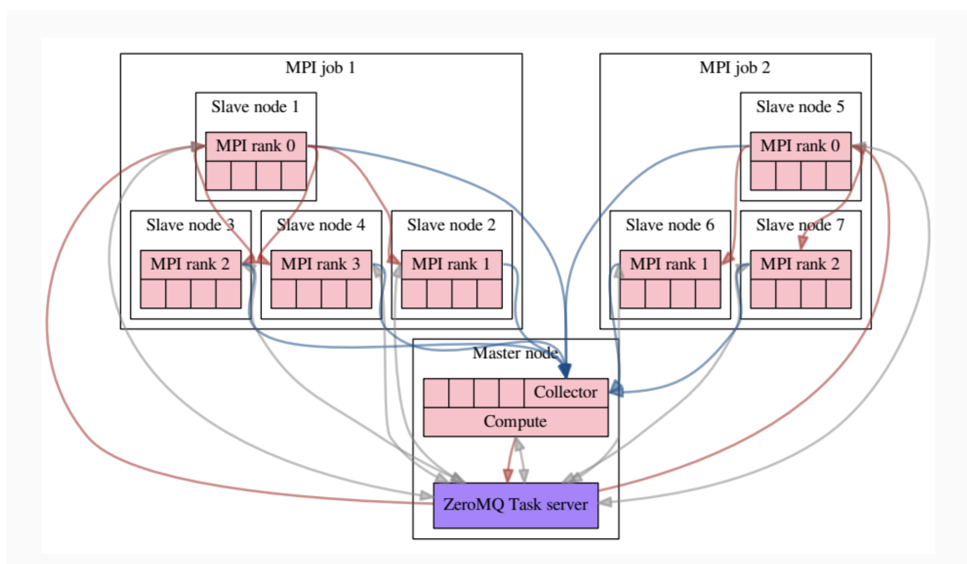

Figure 3: Design of the sparse matrix-vector multiplication in the Davidson algorithm implemented in Quantum Package. A *master* OpenMP simulation runs on a single node. Multiple *slave* MPI/OpenMP simulations can be connected using ZeroMQ to the master simulation to increase the computational power.

large, and a *client* is an autonomous multi-node simulation interconnected with MPI for fast broadcasting of the replicated data (Figure 3). Fault-tolerance is not yet implemented for the Davidson diagonalization, but we are working on a mechanism to re-queue the tasks that have not returned a result after some time.

Recently, we have performed a calculation with Quantum Package distributed on two supercomputers, one in Rouen (north of France) and one in Toulouse (South of France).[8] The results have demonstrated that our 3-layer OpenMP/MPI/ZeroMQ implementation of task-based parallelism enables the use of Quantum Package on distributed cloud infrastructures, and completely solves the problem of the latency of data transfer.

Exascale applications

Our recommendation for designing exascale applications is to use asynchronous task-based parallelism. At the level of the compute node, task-based parallelism provides automatic load balancing between the CPU cores and GPUs, which run at different speeds. Moreover, it addresses the problem of the expensive data transfers from the memory of the host to the memory of the GPU, which can be identified as a high latency issue. There exists many possibilities to implement task-based parallelism within a compute node. OpenMP is a good choice as it is a widely adopted standard, and version 5 allows to offload asynchronously computations to accelerators with the `nowait` clause. Another interesting candidate is the StarPU library[1] as it proposes smart schedulers that can take advantage of multiple CPUs and GPUs, and can run multiple kernels on a single GPU. Moreover, StarPU can distribute tasks among compute nodes with MPI, making it a single framework for the expression of the parallelism in applications. For applications such as Monte Carlo simulations where fault

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

tolerance can be implemented, we don't recommend to use MPI for the distribution of tasks on the whole supercomputer, since a single process failure will crash the whole simulation. Instead, we propose to design the application such that it can aggregate multiple smaller MPI simulations using an open-source high-performance asynchronous message passing library which implements fault tolerance. If a low-latency network is the target hardware, we recommend using GPI/GASPI[9] to interconnect the MPI simulations. If the application is also meant to be run on cloud infrastructures, we recommend to interconnect the MPI simulations with ZeroMQ.

3 Asynchronous Diffusion Monte Carlo algorithm

Real space QMC

In real-space quantum Monte Carlo algorithms, trajectories are built by moving random walkers with a drifted diffusion random process. In the family of Diffusion Monte Carlo (DMC) algorithms, a birth/death process is introduced. In this context, we don't consider any more independent trajectories but we consider instead a *population* of walkers with a fluctuating number of walkers. The DMC algorithm is usually expressed as:

```
E = 0.
for kStep in range(nSteps):
    newCoordinates = []

    for iWalker in range(nWalkers):
       x_old = coordinates[iWalker]
       x = DiffusionDrift(x_old)

       eWalk[iWalker] = Energy(x)
       E += eWalk[iWalker]
       w = exp(-timeStep * (Energy(x) - E_ref))

       if w < 1.: # Random death of the walker
          if random.uniform(0.,1.) < w:
             newCoordinates.append(x)

       else: # Random duplication
          newCoordinates.append(x)
          if random.uniform(0.,1.) < w-1.:
             newCoordinates.append(x)

    # end for iWalker
    coordinates = newCoordinates
    E_ref = f(eWalk)
# end for kStep
```

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

```
return E / nSteps
```

In this algorithm, the loop over the Monte Carlo steps is the outer loop and the loop over the walkers is the inner loop. Hence, all the walkers of the population are synchronized in time. This choice is motivated by the fact that the variable `E_ref` is adjusted dynamically so that the number of walkers in the population stays roughly constant. As the number of walkers is variable, distributing the population on the cluster leads to load balancing problems since the number of walkers per compute node varies as the algorithm evolves. Moreover, it implies a synchronization barrier to ensure that all the walkers are synchronized and that `E_ref` is computed and broadcast. There has been multiple works in the literature to reduce the impact of the load balancing problem in DMC.[10, 11, 12]

An alternative algorithm is Pure Diffusion Monte Carlo (PDMC), where the branching step is replaced by carrying a weight:

```
E = 0.
sumWeight = 0.
for kStep in range(nSteps):
    w[iWalker] = 1.

    for iWalker in range(nWalkers):
       x_old = coordinates[iWalker]
       x = DiffusionDrift(x_old)
       coordinates[iWalker] = x

       E += w[iWalker] * Energy(x)
       sumWeight += w[iWalker]
       w[iWalker] *= exp(-timeStep * (Energy(x) - E_ref))

    # end for iWalker
# end for kStep
return E / sumWeight
```

This algorithm is unstable because at some point the multiplicative weight will asymptotically go to either infinity or to zero. For this reason, the DMC algorithm with a branching process is the standard implementation in the QMC codes. With the PDMC scheme, `E_ref` can be kept fixed and the two loops can be interchanged, leading to a de-synchronization of the walkers:

```
E = 0.
sumWeight = 0.
for iWalker in range(nWalkers):
    w = 1.
    x = coordinates[iWalker]

    for kStep in range(nSteps):
       x = DiffusionDrift(x)
```

```
        E += w * Energy(x)
        sumWeight += w
        w *= exp(-timeStep * (Energy(x) - E_ref))

    # end for kStep
    coordinates[iWalker] = x
# end for iWalker
return E / sumWeight
```

Therefore, PDMC can be run with a single walker per compute node, and more statistics can be obtained by running independent PDMC simulations, making this algorithm well suited to the fault-tolerant model presented in the previous section. In addition, as the walkers don't need to be synchronized, some trajectories can run independently on CPUs and others ones on GPUs. But this algorithm still needs to be stabilized to be usable.

We propose to add a branching step to PDMC to make it as efficient in practice as DMC:

```
E = 0.
sumWeight = 0.
while (continueRun):

    newCoordinates = []
    for iWalker in range(nWalkers):
       w = 1.
       x = coordinates[iWalker]
       for kStep in range(nSteps):      # <- Maximum number of steps
          x = DiffusionDrift(x)

          E += w * Energy(x)
          sumWeight += w
          w *= exp(-timeStep * (Energy(x) - E_ref))

          if w < 0.5:                   # <- Death threshold
             if random.uniform(0.,1.) < w:
                newCoordinates.append(x)
             break
          if w > 2.0:                   # <- Birth threshold
             newCoordinates.append(x)
             w -= 1.
       # end for kStep

       if w > 1.:
           newCoordinates.append(x)
           w -= 1.
       if random.uniform(0.,1.) < w:    # <- Handle remaining part of the weight
```

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

```
            newCoordinates.append(x)

    # end for iWalker
    coordinates = newCoordinates
# end while
return E / sumWeight
```

When the weight of a walker becomes greater than 2, we create a new walker with a weight of 1, and we remove 1 to the current weight. When the weight is lower than 0.5, we draw a random number to decide whether to kill the walker or not. If the walker is not killed, its weight goes back to 1. To avoid having a too large discrepancy between the number of steps performed by all the walkers, we impose a maximum length of the trajectory, and a branching step is realized at the end of the trajectory to discretize the weight. This algorithms converges to the DMC energy, and one can remark that all the walkers are not any more at the the same point in time because of the branching steps. Hence, it is possible to combine different lists of walkers coming from different compute nodes running at different speeds without introducing any bias. If the branching events occur rarely enough, the complete list of walker coordinates can be stored on a distant server, and transferred asynchronously leading to a perfect load balancing with only a few walkers per node. The rate of branching can be tuned by adjusting the birth and death thresholds (here 0.5 and 2.0), and also by increasing the quality of the wave function: better wave functions branch less because the weight fluctuates less. The final asynchronous DMC algorithm is then:

```
E = 0.
sumWeight = 0.
promise = asyncFetchWalkers(server, nWalkers) # -> Non-blocking coordinates request
coordinates = asyncWait(promise)              # -> Wait for coordinates to arrive
while (continueRun):
    promise = asyncFetchSomeWalkers(server, nWalkers) # -> Request next coordinates

    for iWalker in range(nWalkers):
       w = 1.
       x = coordinates[iWalker]

       for kStep in range(nSteps):
          x = DiffusionDrift(x)

          sumWeight += w
          E += w * Energy(x)
          w *= exp(-timeStep * (Energy(x) - E_ref))

          if w < 0.5:
             if random.uniform(0.,1.) < w:
                asyncSendCoordinates(server, x)        # <- Send coordinates
             break
```

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

```
            if w > 2.0:
                asyncSendCoordinates(server, x)            # <- Send coordinates
                w -= 1.

        # end for kStep
        if w > 1.:
            asyncSendCoordinates(server, x)                # <- Send coordinates
            w -= 1.
        if random.uniform(0.,1.) < w:    # <- Handle remaining part of the weight
            asyncSendCoordinates(server, x)                # <- Send coordinates

    # end for iWalker
    coordinates = asyncWait(promise)      # -> Wait for new coordinates to come
# end while
return E / sumWeight
```

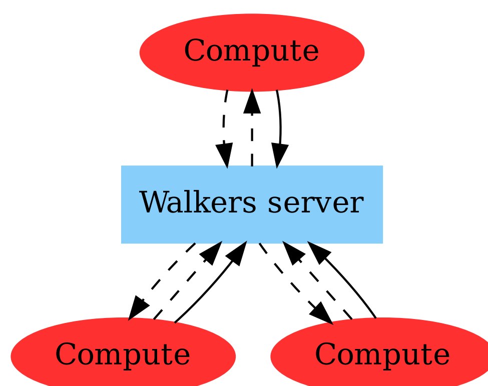

Figure 4: Communication pattern of the asynchronous DMC algorithm. A compute node requests walker coordinates (dashed line), the server fulfills the request (dashed line). Then, when branching occurs the compute node sends coordinates to the server (plain line). The request of the next batch of coordinates is made in advance, and the walker coordinates are transferred while the compute node is computing the trajectories of the previous batch.

A prototype of this algorithm was implemented in QMC=Chem, and we confirmed numerically that it converges to the same energy as the DMC algorithm. We plan to write a small proof of concept in Python or Julia using the QMCkl library developed in WPs 1 and 3. This mini-application will be a nice illustration of the usefulness of having a library containing all the required functions for performing QMC trajectories, and it will also demonstrate the experimental scaling of the algorithm with the number of compute nodes, and the possible recovery from crashes.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

Determinant-space QMC

Similar to the case of real-space QMC, in determinant-space QMC trajectories are built of random occupations on Slater determinants in the Hilbert space. However, in contrast to real-space QMC, the total possible space is finite and denoted as the full Hilbert space of determinants. The objective of the QMC algorithm is then to estimate the Energy and optionally the coefficients of the determinants corresponding to the ground state wavefunction. Here one can distinguish between two types of wavefunctions, first, *bosonic* wavefuntions that constitute only non-negative coefficients which can be interpreted as weights. Second, ground states that exhibit a *fermionic* character with both positive and negative coefficients. The former can be treated with either PDMC or DMC algorithms, however, the latter needs special consideration and will be treated separately.

Bosonic wavefunctions

The algorithm for strictly non-negative wavefunction determinant QMC is similar to that of real space i.e the Pure Diffusion Monte Carlo method. The main difference is the projection operator

$$\mathcal{G}(H) \equiv 1 - \tau\left(H - E_T\right). \tag{1}$$

Similar to the real-space QMC, the target wavefunction can be obtained by a repeated application of the projection on a trial wavefunction Ψ_T,[13]

$$\lim_{n\to\infty} \mathcal{G}(H)^n\, \Psi_T = \Psi_0. \tag{2}$$

The main steps in the Determinant-space QMC algorithm are shown below:

```
E = 0.
for kStep in range(nSteps):
    newDeterminant = []

    for iWalker in range(nWalkers):
       jDet = determinants[iWalker]
       # Spawning Step
       iDet = SpawnDeterminant(jDet)

       eWalk[iWalker] = Energy(iDet)
       E += eWalk[iWalker]

       if iDet == jDet: # Copy current determinant
            w = (1 - timeStep*(H(iDet,iDet) - E_ref)) \
              / (1 - timeStep*(H(iDet,iDet) - Energy(iDet)))
       else:             # Spawn a new determinant
            w = 1.

       if w < 1.:        # Random death of the walker
```

```
            if random.uniform(0.,1.) < w:
               newDeterminant.append(iDet)

         else:              # Random duplication
            newDeterminant.append(iDet)
            if random.uniform(0.,1.) < w-1.:
               newDeterminant.append(iDet)

      # end for iWalker
      determinants = newDeterminants
      E_ref = f(eWalk)
   # end for kStep
   return E / nSteps
```

The two main steps in the algorithm are spawning and branching. During the spawning step, walkers on new determinants are born. In the birth/death step, walkers can be copied/deleted. The main difference compared to the real-space QMC is in the calculation of the weight w_{ij}.

Fermionic wavefunctions

In the case of fermionic wavefunctions, the spawning probability $p_s(j|i)$ given by

$$p_s(j|i) = \frac{\tau \, H_{ij}}{p_{\text{gen}}(j|i)}, \tag{3}$$

where $p_{\text{gen}}(j|i)$ is a positive value and relates to the total probability of spawning on an new determinant $j \neq i$.[14] Thus, $p_s(j|i)$ can become either positive or negative and hence cannot be used as a probability distribution. This is due to the fact that the off-diagonal Hamiltonian matrix-elements H_{ij} can be either positive or negative in sign. Consequently, one has to separate the population of walkers into two families of determinants, one with positive sign (i.e. positive walkers) and another with negative sign (i.e. negative walkers). In order to accumulate fermionic statistics, an *annihilation* step has to be introduced in order to correlate the two populations of walkers. The annihilation step removes pairs of walkers of opposite signs on the same determinant. Hence, the walkers become correlated and completely independent QMC simulations with independent sets of walkers can no longer be carried out. This is in stark contrast to the case for bosonic wavefunctions.

In the NECI code, the walkers are stored in a distributed hash table where the keys are the determinants and the values are their occupation.This design is motivated by the fact that the number of occupied determinants can become too large to be stored on a single compute node. The key point to realize is that in this context the annihilation step involves a global communication, and this could become a bottleneck for exascale applications.

Here we propose a different algorithm which involves non-blocking asynchronous communications of the local walker populations in order to hide the latency of the communications. The idea originates from the realization that in order to accumulate fermionic statistics, it is sufficient to perform the annihilation step regularly enough. Crucially, it can be shown that the different walkers do not need to be synchronized for the annihilation to be effective.[15, 16, 17]

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

The main steps of the algorithm follow that of the asynchronous DMC algorithms described in the previous section, in which the storage of the complete list of walkers is separated from the computational code (see figure 4). If the list of walkers is too large to fit on a single node, a distributed hash tables may be used. We will call here *walkers server* the process which is the entry point for interacting with the distributed storage of walkers. The complete algorithm for a computing process is described as follows:

1. The computing process obtains a batch of walkers from the walkers server, in the form of pairs (determinant, signed occupation). It receives also the E_{ref} constant required to adjust the growth of the population.
2. Every walker of the batch spawns new walkers, stored in a temporary list of signed walkers,
3. Every walker of the batch performs the birth/death step,
4. The walkers of the temporary list are added to the batch, including an annihilation process.
5. The wall-clock time is measured from step 1 to now.
   (a) If the wall-clock time is long enough or if the local number of walkers has become critically large, the local signed list of pairs (determinant, signed occupation) is sent to the walkers server, together with a contribution to the energy:

$$\sum_{i=1}^{N_{\text{walkers}}} \frac{n_i}{N_{\text{walkers}}} \langle 0|H|i\rangle$$

   We then go back to step 1.

   (b) Otherwise, we go back to step 2.

The role of the walkers server is to provide batches of pairs (determinant, signed occupation) to the computing processes, and to receive new batches from them. During the storage of the new data, an annihilation step is performed in the global population. The walkers server also computes the value of E_{ref} which has to be common to all compute processes, and estimates the total energy by combining contributions coming from all the compute processes.

The non-blocking algorithm for the fermionic determinant-space QMC is then similar to the real-space asynchronous QMC. A simple implementation of the above algorithm is given below:

```
promise = asyncFetchWalkers(server, nWalkers) # -> Non-blocking determinants request
determinants = asyncWait(promise)             # -> Wait for determinants to arrive

wallTime0 = getWallTime()                   # -> Initial wall-clock  time
while (continueRun):
    promise = asyncFetchNewWalkers(server, nWalkers) # -> Request next batch

    while getWallTime() - wallTime0 < MaxTime:
```

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

```
        newDeterminants = []
        for (jDet, jN) in determinants:
            if   jN < 0:  jSign = -1
            else       :  jSign =  1

            for k in range(jSign * jN):

                # Spawning Step
                (iDet, iSign) = SpawnDeterminant(jDet)
                newDeterminants.append( (iDet, -iSign*jSign) )

                # Birth/Death Step
                w = getBirthDeathProb(jDet)
                if w < 1.: # Random death of the walker
                   if random.uniform(0.,1.) < w:
                      newDeterminants.append( (jDet, jSign) )

                else: # Random duplication
                   newDeterminants.append( (jDet, jSign) )
                   if random.uniform(0.,1.) < w-1.:
                      newDeterminants.append( (jDet, jSign) )

            # end for
        # end for
        determinants = mergeWalkers(newDeterminants)
    # end while
    E = Energy(determinants)
    asyncSendDeterminants(server, determinants, E)  # <- Send walkers (non-blocking)

    determinants = asyncWait(promise)  # Block until new batch has arrived
    wallTime0 = getWallTime()

# end while
```

A proof-of-principle algorithm has been implemented and has been tested with the conventional FCIQMC[14] algorithm in order to verify that there is no bias in the energy estimates. The next step will be to test the algorithm in a large scale simulation.

4 Low-memory algorithm for wavefunction optimization

Some exascale QMC applications will contain both a large number of correlated electrons and a large number of Slater determinants. CI coefficient optimization is one of the most memory consuming approach in QMC, and we propose here a method which both reduces the memory requirements and

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

takes advantage of zero-variance estimators which exhibit smaller statistical errors than the usual ones, thus requiring shorter simulation times.

Consider a ground-state wavefunction $\Phi(\mathbf{R})$ expressed in a basis of N_{det} Slater determinants $\{D_I(\mathbf{R})\}$:

$$\Phi(\mathbf{R}) = \sum_{I=1}^{N_{\text{det}}} c_I \, D_I(\mathbf{R}) \tag{4}$$

The linear coefficients are initially obtained by solving the standard CI problem. In the QMC framework, a relatively cheap and efficient way of increasing the amount of electron correlation described by the wave function is to introduce a Jastrow factor $e^{J(\mathbf{R})}$, capturing particularly well short-range correlation effects that can't be described by the finite determinant basis set.

$$\Psi(\mathbf{R}) = \Phi(\mathbf{R}) \, e^{J(\mathbf{R})} = \left(\sum_{I=1}^{N_{\text{det}}} c_I D_I(\mathbf{R}) \right) \, e^{J(\mathbf{R})}. \tag{5}$$

The most natural way of re-optimizing the coefficients c_I under the presence of the Jastrow factor is to express the CI problem in the basis $\{\mathcal{D}_I(\mathbf{R}) \equiv D_I(\mathbf{R}) \, e^{J(\mathbf{R})}\}$. This basis is not orthonormal, so in addition to the Hamiltonian matrix elements $H_{IK} = \langle \mathcal{D}_I | \hat{H} | \mathcal{D}_K \rangle$ the overlap matrix elements $S_{IK} = \langle \mathcal{D}_I | \mathcal{D}_K \rangle$ need to be computed and the CI problem can be solved as

$$\mathbf{H} \, \mathbf{C} = E \mathbf{S} \, \mathbf{C}. \tag{6}$$

As the forms commonly used for the Jastrow factor are too complicated to integrate analytically, these $3N$-dimensional integrals are sampled using a Variational Monte Carlo (VMC) sampling:

$$H_{IK} = \left\langle \frac{\mathcal{D}_I}{\Psi} \frac{\hat{H} \, \mathcal{D}_K}{\Psi} \right\rangle_{\Psi^2} \text{ and } S_{IK} = \left\langle \frac{\mathcal{D}_I}{\Psi} \frac{\mathcal{D}_K}{\Psi} \right\rangle_{\Psi^2} \tag{7}$$

where $\langle \dots \rangle_{\Psi^2}$ denotes the stochastic average over the Monte Carlo samples drawn with the $3N$-dimensional density Ψ^2.

Such an approach requires the sampling of two $N_{\text{det}} \times N_{\text{det}}$ matrices, and is therefore only applicable to a few thousand parameters. State-of-the art algorithms employ Krylov methods,[18] such that the sampling can be reduced to a small number m of vectors, with a memory requirement scaling as $\mathcal{O}(m \times N_{\text{det}})$. This memory reduction enables the optmization of hundreds of thousands of parameters, but is obtained at the cost of multiple matrix-vector multiplications during the sampling.

The algorithm we propose here requires exactly N_{det} random variables, with no additional computational cost. First, we eliminate the need for the overlap matrix by expressing the optimization in the Transcorrelated (TC) formalism, where the correlation is incorporated in the Hamiltonian through a similarity transformation[19]

$$\hat{H}_J \equiv e^{-\hat{J}} \, \hat{H} \, e^{\hat{J}}. \tag{8}$$

Therefore, solving exactly the Schrödinger equation with an ansatz wavefunction $\Psi(\mathbf{R}) = \Phi(\mathbf{R}) \, e^{J(\mathbf{R})}$

$$\hat{H} \, \Psi = E \, \Psi, \tag{9}$$

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

is equivalent to solve the TC differential equation

$$\hat{H}_J \, \Phi = E \, \Phi \tag{10}$$

which is expressed in the orthonormal basis of Slater determinants, suppressing the need for sampling the overlap matrix.

By projecting Eq. (10) in the basis of Slater determinants, we obtain

$$\sum_{K=1}^{N_{\text{det}}} \langle D_I | \hat{H}_J | D_K \rangle \, c_K = E \, c_I \tag{11}$$

We define the operator $\hat{\Delta} = \hat{H}_J - \hat{H}$, and the projected equations can be rewritten as

$$\sum_{K=1}^{N_{\text{det}}} \langle D_I | \hat{H} | D_K \rangle \, c_K + \frac{\langle D_I | \hat{\Delta}_J | \Phi \rangle}{c_I} c_I = E \, c_I \tag{12}$$

such that the effect of the Jastrow factor is expressed as a diagonal dressing of the the Hamiltonian. As the dressed Hamiltonian is now parameterized by the wave function, the equations will be solved iteratively,

$$\Phi^{(i)} = \sum_{I=1}^{N_{\text{det}}} c_I^{(i)} D_I, \tag{13}$$

using the solution of the standard CI problem as an initial guess. The dressing term is approximated using the coefficients of the previous iteration:

$$\frac{\left\langle D_I \middle| \hat{\Delta}_J \middle| \Phi^{(i)} \right\rangle}{c_I^{(i)}} \approx \frac{\left\langle D_I \middle| \hat{\Delta}_J \middle| \Phi^{(i-1)} \right\rangle}{c_I^{(i-1)}}. \tag{14}$$

Note that the division by c_I generates numerical instabilities when c_I is small, so we reformulated the problem using a column dressing such that the denominator is always the largest of all $|c_I|$ as proposed in [20].

At each iteration, Davidson's method[6] can be used to extract the ground state of the dressed Hamiltonian. This method requires to apply the dressed Hamiltonian to the trial vector at each step of Davidson's algorithm,

$$\left\langle D_I \middle| \hat{H}_J \middle| \Phi^{(i)} \right\rangle = \left\langle D_I \middle| \hat{H} \middle| \Phi^{(i)} \right\rangle + \left\langle D_I \middle| \hat{\Delta}_J \middle| \Phi^{(i)} \right\rangle. \tag{15}$$

One can remark that the first term can be computed exactly with a standard quantum chemistry code, such as Quantum Package. Only the second term needs to be sampled in VMC, and this term is implemented in QMC=Chem as

$$\frac{1}{\int \left[\Psi^{(i)}(\mathbf{R})\right]^2 e^{-2J(\mathbf{R})} \mathrm{d}\mathbf{R}} \times \int \left[\Psi^{(i)}(\mathbf{R})\right]^2 \frac{D_I(\mathbf{R}) e^{-J(\mathbf{R})}}{\Psi^{(i)}(\mathbf{R})} \left(\frac{\hat{H}\Psi^{(i)}(\mathbf{R})}{\Psi^{(i)}(\mathbf{R})} - \frac{\hat{H}\Phi^{(i)}(\mathbf{R})}{\Phi^{(i)}(\mathbf{R})} \right) \mathrm{d}\mathbf{R} \tag{16}$$

$$= \frac{1}{\left\langle e^{-2J} \right\rangle_{\Psi^{(i)2}}} \left\langle \frac{D_I \, e^{-J}}{\Psi^{(i)}} (E_L - E_0) \right\rangle_{\Psi^{(i)2}}. \tag{17}$$

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

This term has low statistical fluctuations, since most of the fluctuations of $E_L = \frac{\hat{H}\Psi^{(i)}(\mathbf{R})}{\Psi^{(i)}(\mathbf{R})}$ and $E_0 = \frac{\hat{H}\Phi^{(i)}(\mathbf{R})}{\Phi^{(i)}(\mathbf{R})}$ cancel out, as shown on figure 5, and as a consequence shorter simulation times can be used for the VMC sampling.

We have also proposed an alternative estimator with even smaller fluctuations using as a reference the transcorrelated Hamiltonian of Giner[21] instead of the standard Hamiltonian. We have observed a reduction of a factor of two in the error bars, but this transcorrelated Hamiltonian requires the computation of three-electron integrals which will not be technically applicable to large systems. To cure this problem, we take as a reference the transcorrelated Hamiltonian where the three-body terms are neglected, leading to the same reduction in fluctuations but with a deterministic computation of the reference which is tractable for larger systems.

Wave function	Water	Fluorine dimer
Hartree-Fock	-16.94804	-198.76828
Full CI	-17.16467	-199.361
Selected CI	-17.16452	-199.32215
N_{det}	400 478	91 988
CI/Jastrow	-17.2053(7)	-199.4455(5)
CI/Jastrow iteration 1	-17.2376(1)	-199.4669(4)
CI/Jastrow iteration 2	-17.2376(1)	-199.4672(5)

Table 1: Convergence of the energy (a.u.) for the water molecule and the fluorine dimer.

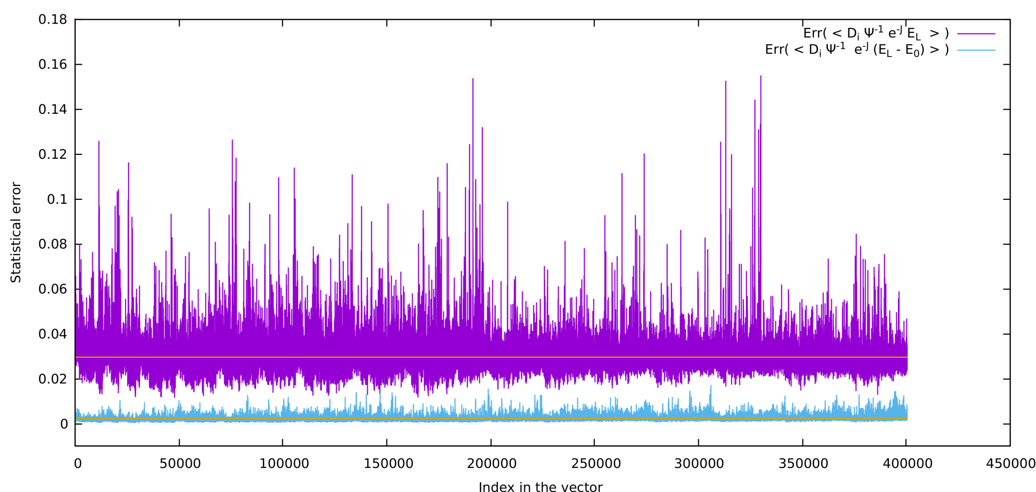

Figure 5: Statistical errors of $\left\langle \frac{D_I e^{-J}}{\Psi} E_L \right\rangle$ and $\left\langle \frac{D_I e^{-J}}{\Psi} (E_L - E_0) \right\rangle$ in a simulation of the water molecule with 400 000 Slater determinants.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

As a proof of concept, we have applied the proposed method to the water molecule using effective core potentials with more than 400 000 Slater determinants, and to the fluorine dimer using all-electron calculations with around 100 000 determinants. Table 1 shows that after a single iteration the energy minimization has converged to a sub-milliHartree precision, demonstrating the efficiency of our method. An article is under preparation.

5 Summary

Using efficiently exascale machines will require to take advantage of parallelism first at the level of a single compute node, and then at the level of a large cluster of nodes. Single-node efficiency is a problem taken care of in WP3, and the present WP focuses on distributing the work on multiple nodes, keeping the efficiency high.

Monte Carlo algorithms have the particularity that they are able to return an unbiased result when some parts of the computation have been removed, if the removed parts are random samples. We can exploit this advantage to design fault-tolerant applications, since there is no need for book keeping the computed tasks or handling a complex recovery procedure. The only condition is that the simulation survives upon failure of some processes, which is not the standard behavior of applications relying on the MPI library. Hence, we propose to use alternative libraries to limit the impact of failures of MPI processes.

The heterogeneity of the hardware imposes the usage of asynchronous algorithms to keep a good balance of the work load, and to hide the latency of communications. This requires algorithmic changes in the most common QMC methods. We have proposed some developments which combine PDMC with DMC to benefit both from the asynchronous character of PDMC and the stable character of DMC thanks to the branching process.

Finally we address the problem that the available memory will not grow as much as the computing power, and low-memory implementations of the algorithms will be required for applications requiring a computational power at the exascale. We have proposed a new method for wave function optimization of hundreds of thousands of parameters with a minimal memory footprint.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

References

[1] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, “StarPU: a unified platform for task scheduling on heterogeneous multicore architectures,” *Concurrency Computat.: Pract. Exper.*, vol. 23, no. 2, pp. 187–198, Feb 2011.

[2] A. Scemama, M. Caffarel, E. Oseret, and W. Jalby, “Quantum Monte Carlo for large chemical systems: Implementing efficient strategies for petascale platforms and beyond,” *J. Comput. Chem.*, vol. 34, no. 11, pp. 938–951, Apr 2013.

[3] “ZeroMQ,” Feb 2022, [Online; accessed 4. Feb. 2022]. [Online]. Available: https://zeromq.org

[4] Dec 2015, [Online; accessed 4. Feb. 2022]. [Online]. Available: http://irpf90.ups-tlse.fr/files/succes2015.pdf

[5] Y. Garniron, T. Applencourt, K. Gasperich, A. Benali, A. Ferté, J. Paquier, B. Pradines, R. Assaraf, P. Reinhardt, J. Toulouse, P. Barbaresco, N. Renon, G. David, J.-P. Malrieu, M. Véril, M. Caffarel, P.-F. Loos, E. Giner, and A. Scemama, “Quantum Package 2.0: An Open-Source Determinant-Driven Suite of Programs,” *J. Chem. Theory Comput.*, vol. 15, no. 6, pp. 3591–3609, Jun 2019.

[6] E. R. Davidson, “The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices,” *J. Comput. Phys.*, vol. 17, no. 1, pp. 87–94, Jan 1975.

[7] Y. Garniron, A. Scemama, P.-F. Loos, and M. Caffarel, “Hybrid stochastic-deterministic calculation of the second-order perturbative contribution of multireference perturbation theory,” *J. Chem. Phys.*, vol. 147, no. 3, p. 034101, Jul 2017.

[8] A. Scemama, “A quantum chemistry calculation distributed among computing facilities with Quantum Package,” *Zenodo*, Oct 2019.

[9] C. Simmendinger, M. Rahn, and D. Gruenewald, “The GASPI API: A Failure Tolerant PGAS API for Asynchronous Dataflow on Heterogeneous Architectures,” in *Sustained Simulation Performance 2014*. Cham, Switzerland: Springer, Nov 2014, pp. 17–32.

[10] C. D. Sudheer, S. Krishnan, A. Srinivasan, and P. R. C. Kent, “Dynamic load balancing for petascale quantum Monte Carlo applications: The Alias method,” *Comput. Phys. Commun.*, vol. 184, no. 2, pp. 284–292, Feb 2013.

[11] K. P. Esler, J. Kim, D. M. Ceperley, W. Purwanto, E. J. Walter, H. Krakauer, S. Zhang, P. R. C. Kent, R. G. Hennig, C. Umrigar, M. Bajdich, J. Kolorenč, L. Mitas, and A. Srinivasan, “Quantum Monte Carlo algorithms for electronic structure at the petascale; the Endstation project,” *J. Phys. Conf. Ser.*, vol. 125, no. 1, p. 012057, Jul 2008.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.

[12] M. T. Feldmann, J. C. Cummings, D. R. Kent, R. P. Muller, and W. A. Goddard, "Manager-worker-based model for the parallelization of quantum Monte Carlo on heterogeneous and homogeneous networks," *J. Comput. Chem.*, vol. 29, no. 1, pp. 8–16, Jan 2008.

[13] R. Assaraf, P. Azaria, M. Caffarel, and P. Lecheminant, "Metal-insulator transition in the one-dimensional su (n) hubbard model," *Physical Review B*, vol. 60, no. 4, p. 2299, 1999.

[14] G. H. Booth, A. J. Thom, and A. Alavi, "Fermion monte carlo without fixed nodes: A game of life, death, and annihilation in slater determinant space," *The Journal of chemical physics*, vol. 131, no. 5, p. 054106, 2009.

[15] D. Arnow, M. Kalos, M. A. Lee, and K. Schmidt, "Green's function monte carlo for few fermion problems," *The Journal of Chemical Physics*, vol. 77, no. 11, pp. 5562–5572, 1982.

[16] M. Kalos and F. Pederiva, "Exact monte carlo method for continuum fermion systems," *Physical review letters*, vol. 85, no. 17, p. 3547, 2000.

[17] R. Assaraf, M. Caffarel, and A. Khelif, "The fermion monte carlo revisited," *Journal of Physics A: Mathematical and Theoretical*, vol. 40, no. 6, p. 1181, 2007.

[18] E. Neuscamman, C. Umrigar, and G. K.-L. Chan, "Optimizing large parameter sets in variational quantum monte carlo," *Physical Review B*, vol. 85, no. 4, p. 045103, 2012.

[19] N. Handy, "The transcorrelated method for accurate correlation energies using gaussian-type functions: examples on he, h2, lih and h2o," *Molecular Physics*, vol. 23, no. 1, pp. 1–27, 1972.

[20] Y. Garniron, A. Scemama, E. Giner, M. Caffarel, and P.-F. Loos, "Selected configuration interaction dressed by perturbation," *J. Chem. Phys.*, vol. 149, no. 6, p. 064103, Aug 2018.

[21] E. Giner, "A new form of transcorrelated Hamiltonian inspired by range-separated DFT," *J. Chem. Phys.*, vol. 154, no. 8, p. 084119, Feb 2021.

TREX: Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation program under Grant Agreement No. 952165.