FORSCHUNGSZENTRUM JÜLICH GmbH
Jülich Supercomputing Centre
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

2009 Proceedings of the JSC Guest Student Programme on Scientific Computing

Robert Speck (Ed.)

FZJ-JSC-IB-2009-04

November 2009

(last change: 1. 11. 2009)

Preface

One of the main missions of JSC besides providing and operating supercomputer resources, IT tools, methods and knowhow for the Research Centre Jülich and for European users through the John von Neumann Institute for Computing is the education and encouragement of young and promising scientists in the broad field of scientific computing. As one opportunity for young academics JSC hosted its traditional 10-week guest student programme again during summer 2009. Within this programme students from the natural sciences, engineering, computer science and mathematics had the opportunity to familiarize themselves with different aspects of scientific computing. Mainly supported by staff members of JSC, the participants worked on various topics in computational science including mathematics, physics, chemistry, software development tools, visualization, grid computing, operating systems and communication. Special emphasis was placed on the use of the newly installed JUGENE and JUROPA supercomputers at JSC, which have raised the attraction of the programme this year.

Guest students and their advisers:

Jonathan Groß	Thomas Neuhaus, Michael Bachmann (IFF)
Ricardo Kennedy	Thomas Neuhaus, Marcus Richter, Binh Trieu
Malik Kirchner	Stefan Dürr
Lukasz Kucharski	Jan Meinke, Sandipan Mohanty, Wolfgang Frings
Stilianos Louca	Paul Gibbon, Benjamin Berberich
Stefan Maintz	Thomas Müller
Hannah Rittich	Bernhard Steffen
Martin Rückl	Walter Nadler
Theodros Zelleke	Godehard Sutmann

Zu Beginn ihres Aufenthalts erhielten die Gaststudenten eine viertägige Einführung in die Programmierung und Nutzung der Parallelrechner im JSC. Um den Erfahrungsaustausch untereinander zu fördern, präsentierten die Gaststudenten am Ende ihres Aufenthalts ihre Aufgabenstellung und die erreichten Ergebnisse. Sie verfassten zudem Beiträge mit den Ergebnissen für diesen Internen Bericht des JSC. Wir danken den Teilnehmern für ihre engagierte Mitarbeit - schließlich haben sie geholfen, einige aktuelle Forschungsarbeiten weiterzubringen - und den Betreuern, die tatkräftige Unterstützung dabei geleistet haben. Ein besonderer Dank gilt Wolfgang Frings und Marc-André Hermanns, die den Einführungskurs gehalten haben, Anke Visser, die an der Erstellung dieses Berichtes maßgeblich mitgewirkt hat, und Robert Speck, der mich bei der Organisation in diesem Jahr nach Kräften unterstützt hat. Ebenso danken wir allen, die im JSC und der Verwaltung des Forschungszentrums bei Organisation und Durchführung des diesjährigen Gaststudentenprogramms mitgewirkt haben. Besonders hervorzuheben ist die finanzielle Unterstützung durch den Verein der Freunde und Förderer des FZJ und die Firma IBM. Es ist beabsichtigt, das erfolgreiche Programm künftig fortzusetzen, schließlich ist die Förderung des wissenschaftlichen Nachwuchses dem Forschungszentrum ein besonderes Anliegen. Weitere Informationen über das Gaststudentenprogramm, auch die Ankündigung für das kommende Jahr, findet man unter http://www.fz-juelich.de/jsc/gaststudenten.

Jülich, November 2009 Robert Speck

Content

Parallel Monte Carlo for Condensed Polymers

Jonathan Groß

Universität Leipzig
Institut für Theoretische Physik
Vor dem Hospitaltore 1
04103 Leipzig

E-mail: pge03cnx@studserv.uni-leipzig.de

Abstract:
A simple model of a condensed polymer was simulated using the parallel tempering algorithm. The implementation makes use of multiple cores in modern workstations utilizing POSIX threads. Also there is an outlook to porting the parallel tempering algorithm to GPGPUs using NVIDIA's CUDA.

1 Introduction

One of the most challenging problems of biochemical research is to understand protein folding. All-atom simulations with realistic interactions are extremely difficult. Due to the high complexity of real life proteins, being complex macromolecules, one has to find models to simplify research on such systems. Coarse-grained models are one object of interest in computational polymer physics. Theses models abstract from the influence of quantum chemical details and long range interactions of involved particles. Especially a solvent is no necessary part of simulations. One is able to obtain a more global view of the physical behavior of such systems and calculations are easier and faster.
This paper deals with a particularly simple model: A polymer consisting of not otherwise specified monomers of the same type. The following energy function describes the interaction of the monomers:

$$E = \sum_{i<j} 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] + \sum_{i=1}^{N-1} k(r_{ii+1} - r_0)^2, \tag{1}$$

with $\epsilon = 1$, $\sigma = 1$, $k = 1$ and $r_0 = 1$. The first part is a Lennard-Jones interaction between pairs of monomers of the chain. Between adjacent monomers there is a spring-like potential representing flexible bonds.
The simulated chain consisted of 20 monomers. Even in this simple and small system one is able to see a second-order-like transition – the so-called Θ-collapse. This is a structural change of the conformation from stretched, coil-like configurations at higher temperatures to more compact, globular configurations at lower temperatures.
The focus of this work is to make use of efficient, parallel algorithms to obtain more statistics in a shorter time.

2 Parallel Monte Carlo

2.1 Metropolis Update

Each configuration of the system is assigned a Boltzmann weight

$$w_B(\beta, C) = e^{-\beta E(C)} \tag{2}$$

with $\beta = 1/k_B T$ being the inverse thermal energy. I have set k_B to 1 throughout my simulations.
By running a simulation one obtains a Markov chain which resembles the canonical distribution at *a single* temperature T.

In a regular simulation in the canonical ensemble a configuration of *one* copy of the polymer structure is updated using the Metropolis [1] algorithm. This in my case looks like this:

1. Calculate energy of conformation $\rightarrow E_{old}$
2. Pick random monomer
3. Perform a random shift of each cartesian coordinate
4. Calculate energy of conformation $\rightarrow E_{new}$
5. Accept update with probability $w(C^{old} \rightarrow C^{new}) = \min(1, e^{-\beta\Delta E})$

2.2 Parallel Tempering

In parallel tempering one generates a system of N non-interacting copies of the polymer at different temperatures T_i:

$$\mathcal{C} = \{C_1, C_2, \ldots, C_n\}, \tag{3}$$

where $\mathcal{C}$ is a state the compound system and C_i is the i-th copy of the polymer.

Because the copies do not interact with each other one can assign a weight to a configuration of the artificial system as follows:

$$w_{PT}(\mathcal{C}) = e^{-\sum_i^N \beta_i E(C_i)} = \prod_i^N w_B(\beta_i, E(C_i)). \tag{4}$$

One can assume that the inverse temperatures are in order $\beta_1 < \beta_2 < \ldots < \beta_N$.

To construct a Markov chain of the compound system two steps are necessary.

1. A *local* update.
   Updating only one copy of the system. Since the copies do not interact such an update is accepted or rejected with the Metropolis algorithm as described above.

$$w(\mathcal{C}^{old} \rightarrow \mathcal{C}^{new}) = w(C_i^{old} \rightarrow C_i^{new}) = min(1, e^{-\beta_i \Delta_i E}) \tag{5}$$

2. A *global* update.
   Exchanging conformations of two copies i and j. Taking (4) into account one can see that the probability to accept such an update is

$$w(\mathcal{C}^{old} \rightarrow \mathcal{C}^{new}) = min(1, e^{\Delta\beta\Delta E}) \tag{6}$$

with $\Delta\beta = \beta_j - \beta_i$ and $\Delta E = E(C_j) - E(C_i)$.

The choice of pairs of neighboring inverse temperature is very important. One should choose small enough intervals of inverse temperature differences, because the accept rate will decrease exponentially with $\Delta\beta$.

The idea behind parallel tempering is not only to parallelize the work, but also to improve sampling of the conformational space. If a system is described by a more "realistic" energy function this can result in a rough energy landscape. Conformations might get trapped in one of the many local minima. The exchange of conformations can help to overcome high energy barriers.

3 Multithreading

3.1 Motivation

Over the last years we have seen a change in the CPU market. Up until a few years ago speed of computers increased with increasing clock speed of CPUs. We have reached the "GHz era". But due to high power consumption, physical limits in chip development and heat dissipation, further speed improvements by increasing the clock speed are very limited.
The solution to that problem were so-called multi-core processors – chips that have more than one processing unit. It seems the future of CPU design and development will take this step even further and we will move from the "multi-core era" to even more cores on one chip – the "many-core era".
To make use of this power, available in nearly every PC these days, programming habits also have to change.

There is a "native" way to program multi-core CPUs – *threads*.

3.2 Threads

3.2.1 What is a Thread?

One can think of a thread as a task run by a program. It is possible to have multiple tasks or threads running per program or process. There are several advantages of using multiple threads over multiple processes:

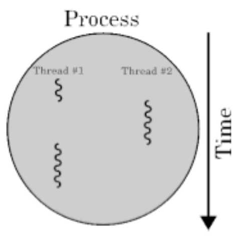


Figure 1: Visualization of threads in a process.

- Threads share memory with main process
- Creation of a thread faster than creating a fork (child process)

Since threads share the memory with the main process there is no need for message passing between them. Also it is faster to create a thread within a running process than forking the process and creating child processes.
Every POSIX-like[1] operating system is capable of threads. There is a standard API to program on these systems called *POSIX threads*.

[1]This includes AIX, *BSD, Linux, Mac OS X and even Windows

3.2.2 POSIX Threads

POSIX threads (Pthreads) is a collection of tools for the C programming language which enable the programmer to create and manipulate threads. It is also possible to synchronize threads using signals and mutexes.

4 Results

4.1 Simulation Parameters

1. Single Thread Monte Carlo
   The serial implementation of the model was run sequentially at 28 different temperatures. Every single run consisted of $5 \cdot 10^5$ thermalization sweeps followed by $5 \cdot 10^6$ measurement sweeps.
2. Parallel Tempering Monte Carlo
   The parallel implementation was run with all 28 temperatures at once using 28 threads on a 4 core CPU. Again, there $5 \cdot 10^5$ thermalization and $5 \cdot 10^6$ measurement sweeps per temperature. Additionally an exchange of copies was attempted every 50 local sweeps.

4.2 Interpretation of Results

Figure 2 shows the specific heat per monomer given by:

$$\frac{c_V}{N} = \frac{1}{N}\frac{\partial < E >}{\partial T} = \beta^2(< E^2 > - < E >^2) \tag{7}$$

As one can see there is a monotonic change in the run of the curve at $T \approx 1.1$. At temperatures below $T \approx 1.1$ the specific heat is nearly constant. At higher temperatures it declines nearly linear. The peak is an indicator for a phase-like transition in the system.

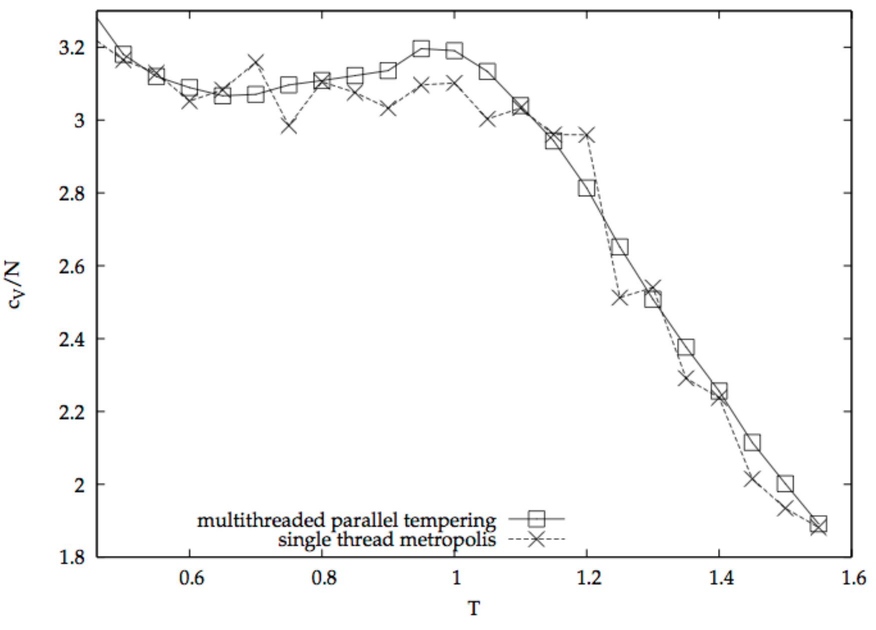


Figure 2: Comparison of results for the specific heat obtained by single thread MC and Parallel Tempering MC.

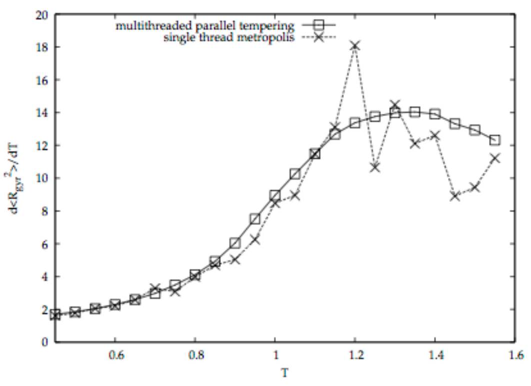
There are more physical quantities one can inspect to see this indication: the end-to-end distance

$$R_{ee} = (\mathbf{r}_N - \mathbf{r}_1) \tag{8}$$

and the square radius of gyration

$$R_{gyr}^2 = \frac{1}{N}\sum_{k=1}^{N} (\mathbf{r}_k - \mathbf{r}_{mean})^2 \,, \tag{9}$$

where $\mathbf{r}_{mean} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{r}_k$ is the center of the polymer. So one can think of the radius of gyration as the mean distance of a monomer to the center of the polymer.



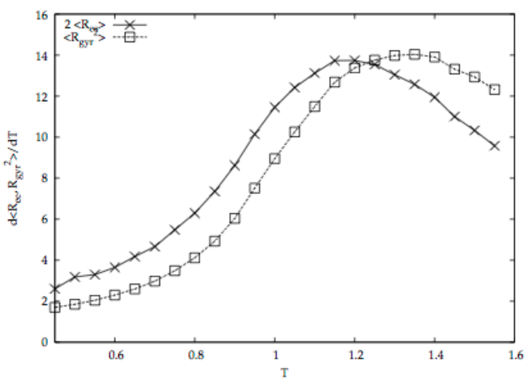


Figure 3: The left figure shows a comparison between the fluctuation of the radius of gyration obtained by single thread MC and Parallel Tempering MC. On the right one can see the fluctuation of R_{ee} and R_{gyr} given by eq (10).

$$\frac{\partial < R_{ee,gyr} >}{\partial T} = \beta^2 (< R_{ee,gyr} \cdot E > - < R_{ee,gyr} > \cdot < E >) \tag{10}$$

Both quantities, the mean end-to-end distance and the mean square radius of gyration, can be used to evaluate the volume expansion of the polymer.

The fluctuation in the curve on the right in the results from single thread MC are due to too less statistical data. Longer runs would have been necessary. This on the other hand shows the advantage of parallel tempering.

Another interesting thing one can see in the figure on the right is that the maxima of the curves are not at the same temperature. This is a finite size effect. Since my chain is only 20 monomers long one can not see a sharp transition temperature but a section or an interval of temperature where a change happens. In this case approximately $1.1 \lesssim T \lesssim 1.3$.

Figure 4: On the left is shown a typical conformation at temperature $T = 0.5$. Conformations at lower temperatures are of a globular structure. The right picture shows a conformation at temperature $T = 1.5$. For higher temperatures this polymer forms coil-like structures.

Figure 4 now shows the expected structures. Below a certain temperature the polymer collapses to a compact, globular shape. However, at higher temperatures the polymer is stretched.

4.3 Speed Results

Figure 5 illustrates the total run times for different number of threads while keeping the total work for the whole program constant. 10^5 sweeps in total were run. The run time for 2 threads is 1.8 times faster

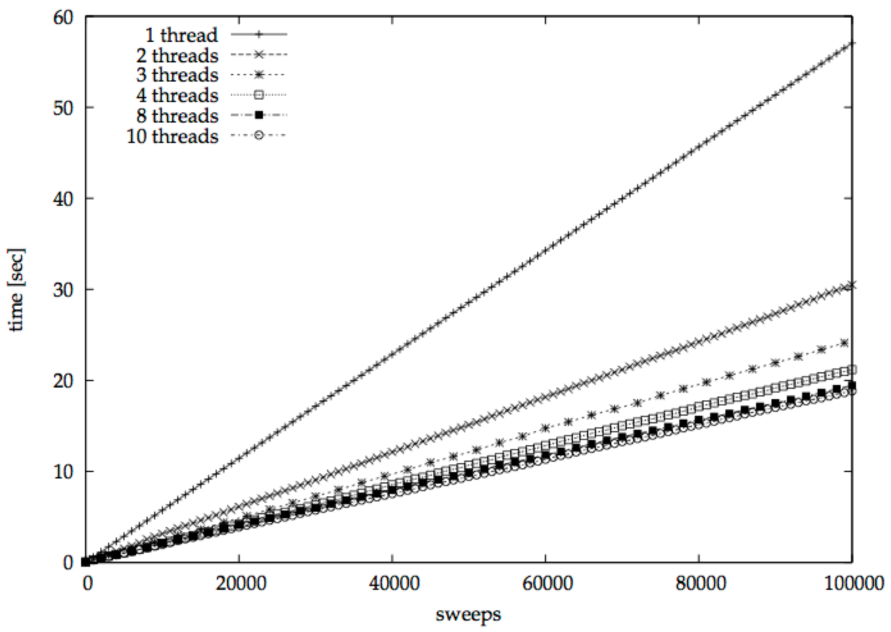


Figure 5: Total time consumption per 100.000 sweeps with different number of threads.

than the run with 1 thread. For 3 threads the speedup is 2.3. From 4 threads up the speedup saturates around a value of 3.

It is easy to understand that the speed does not increase anymore with more threads, because these test

were run on a 4 core machine. On the other hand it is more complicated to explain why there is not a naive speedup factor equal to the number of threads. This is caused by the fact that I wait for all threads to finish their work, sync them and then attempt to exchange conformations. This can be optimized by trying to exchange neighboring conformations as soon as both are finished with their work, so that they do not need to wait for all other threads to finish.
Also there are processes from the operating system running, so that not all of the power of all 4 cores are available to my program.

5 Outlook

Another trend arises in high performance computing these days: *General-purpose computing on graphics processing units* (GPGPU).

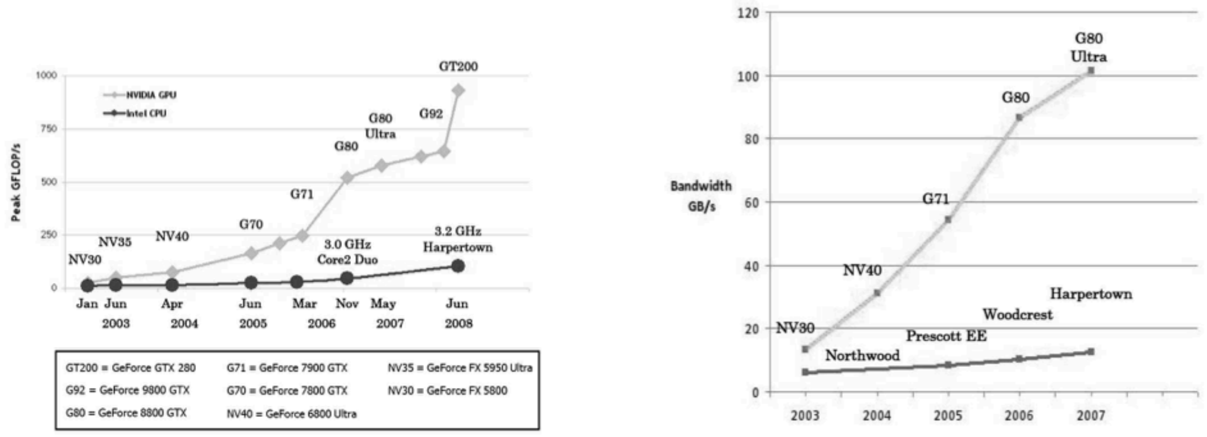


Figure 6: Speed comparisons CPU vs GPU.

The figures above show a discrepancy between peak floating-point operations and memory bandwidth on CPUs and GPUs. This is is due to highly specialized chip design of GPUs aiming for high performance. Highly parallel computation of floating-points. More transistors are dedicated to data processing than data caching or flow control like in CPUs (see below).

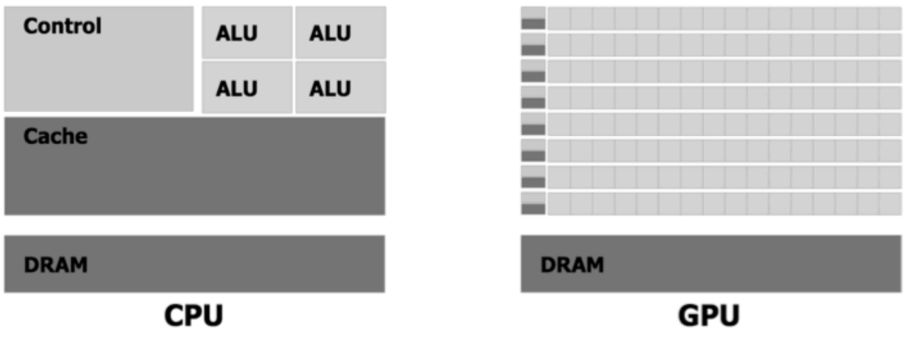


Figure 7: Comparison of CPU and GPU architecture.

Problems with many data elements which can be processed in parallel can really profit from GPGPUs. Modern graphic cards can have up to 240 cores. Arranged in blocks called multiprocessors they have

a shared memory for processing data within this block. This local memory is fast cache memory. In addition to that there is a global memory already in the range of gigabytes.
NVIDIA is providing an API (CUDA) to make use of the power of GPUs. There is also the Khronos Group[2] which puts much effort into developing an independent and open standard for programming hybrid architectures (CPU+GPU) – OpenCL.

The next step will be an implementation of the parallel tempering algorithm using CUDA, since it is more mature than OpenCL at this point of time.

6 Acknowledgments

I would like to thank my advisors at Forschungszentrum Jülich, Michael Bachmann and Thomas Neuhaus, for supporting me during my stay with advice and knowledge. Also I would like to thank Robert Speck who beautified the last weeks by making the German bureaucracy so much more bearable and uncomplicated.

References

1. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. H. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087
2. U. Hansmann, Chem. Phys. Lett. 281 (1997) 140
3. M. Bachmann, H. Arkin, W. Janke, Phys. Rev. E 71, 031906 (2005)
4. B. Barney, Livermore Computing, https://computing.llnl.gov/tutorials/pthreads
5. NVIDIA Corporation, NVIDIA CUDA Programming Guide 2.2.1

[2]http://www.khronos.org/opencl/

Threshold Determination for the Fault-Tolerant Implementation of the CNOT-Gate

Ricardo Kennedy

Universität zu Köln
Albertus-Magnus-Platz
50923 Köln

E-mail: r.kennedy@uni-koeln.de

Abstract:
The controlled-NOT ($CNOT$) gate is implemented fault-tolerantly using the Juelich Massively Parallel Ideal Quantum Computer Simulator (JUMPIQCS)[1] and its performance is tested under the influence of two different error sources: For decoherence errors, we find that the threshold for effective error correction ($10^{-6} < p_{thr} < 10^{-5}$) is comparable to the single qubit threshold found in [1]. In the case of operational errors, we find that the threshold is of the same order of magnitude as in the single qubit case, but dependent on the initial state. It is determined to be $\sigma_{thr1} = (5.5 \pm 0.3) \cdot 10^{-2}$ for a separable initial state $|11\rangle$ and $\sigma_{thr2} = (2.0 \pm 0.3) \cdot 10^{-2}$ for the maximally entangled state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$.

1 Introduction

Classical computing uses bits with two values, 0 or 1. In a quantum computer, these values are encoded as the eigenstates of a two-level quantum system, $|0\rangle$ and $|1\rangle$, such that all complex superpositions are possible and the quantum bit (qubit) is generally in a state

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \tag{1}$$

with $\alpha_1, \alpha_2 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

In analogy to a classical register, a quantum register comprises a set of N qubits in a general superposition with 2^N complex amplitudes:

$$|\psi_N\rangle = \sum_{i=0}^{2^N-1} \alpha_i|i\rangle. \tag{2}$$

Here the binary representation is used, so for example $|6\rangle = |110\rangle = |1\rangle \otimes |1\rangle \otimes |0\rangle$ corresponds to the first qubit (from right to left) being in the $|0\rangle$ state and the second and third qubit being in the $|1\rangle$ state.

The manipulation of this register is called a quantum algorithm and involves a general unitary operation on the entire set of qubits followed by a measurement to read out the result of the computation, shown schematically in the circuit diagram of figure 1.

There is a wealth of algorithms [2] that usually tackle very specialised problems. The most well-known ones are the Grover algorithm for searching databases and the Shor algorithm for factorising large numbers. Both of them give a considerable speed-up over all known classical algorithms [4, 5]. A further

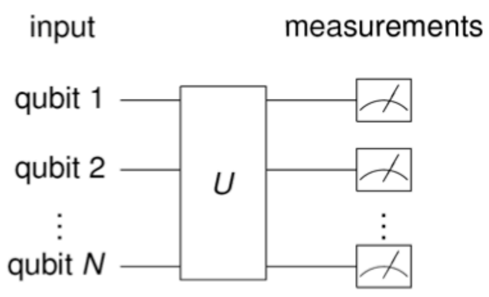


Figure 1: General quantum algorithm: A unitary operation on the 2^N dimensional Hilbert space of the input qubits is followed by measurements.

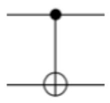
motivation to study quantum computation is the possibility of simulating quantum systems themselves, which was first proposed by Richard Feynman in 1982. While this kind of simulation is inefficient on a classical computer, it has been shown [7] that it can be done just as efficiently on a quantum computer as a classical system can be simulated on a classical computer.

Another useful analogue between a quantum computer and a classical one is the notion of universal gates. While classically either the NAND or the NOR gate are universal, more gates (unitary operations) are required to form a universal set for a quantum computer. Figure 2 shows a common choice, which is a combination of single qubit gates (Hadamard gate H, phase gate S and $\pi/8$-gate T) and a two-qubit gate to create correlations (the controlled-NOT gate $CNOT$). An arbitrary algorithm U (as in figure 1) can therefore be implemented using these gates only.

$$\text{Hadamard} \quad -\boxed{H}- \quad \equiv \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\text{Phase} \quad -\boxed{S}- \quad \equiv \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$$\pi/8 \quad -\boxed{T}- \quad \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

CNOT 

$$\equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2: Universal set of gates for quantum computing.

2 Quantum Error Correction

The introduction to quantum computing given in the last section makes a crucial assumption: The qubits constitute a completely isolated quantum system in which perfect unitary operations are performed. This assumption obviously cannot hold in any physical realisation of a quantum computer and it is therefore valid to ask whether or not any of the mentioned algorithms can be achieved in reality. Inspired by this question the field of quantum error correction developed and showed that errors can indeed be corrected for.

2.1 Error Model

The error model used for the simulations introduces two error sources: Decoherence errors and operational errors. The former are modelled by the so-called *depolarising channel*, which represents a random interaction with the environment. This picture is equivalent to the following: Given an arbitrary qubit state, it will experience a random application of one of the Pauli gates with probability $p/3$ each and will be left alone with probability $1-p$. An initial density matrix ρ will therefore evolve according to

$$\rho \rightarrow (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z). \tag{3}$$

The interaction with the environment is assumed to take place over the entire duration of an algorithm. Accordingly, the decoherence step (3) is performed at every time step in the simulation. The other error source, operational errors, are caused by the experimental imprecision when manipulating the qubits, which needs to be taken into account whenever a unitary operation is performed. They are modelled by re-expressing the action of the quantum gate matrices U (in the $\{|0\rangle, |1\rangle\}$-basis) in terms of planar rotations $R(\theta)$ and polarisations $P(\phi)$:

$$U = R(\theta)P(\phi) = U(\theta, \phi), \tag{4}$$

where

$$R(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \quad \text{and} \quad P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}. \tag{5}$$

To model the experimental imprecision, Gaussian errors $\epsilon_{\theta/\phi}$ are added to the angles θ and ϕ, giving an erroneous operation $U_\epsilon = U(\theta + \epsilon_\theta, \phi + \epsilon_\phi)$. The most significant erroneous gates in the context of the later sections are the Hadamard gate H_ϵ and controlled-NOT gate $CNOT_\epsilon$:

$$H_\epsilon = R\left(\frac{\pi}{4} + \epsilon_\theta\right) P(\pi + \epsilon_\phi) \tag{6}$$

and

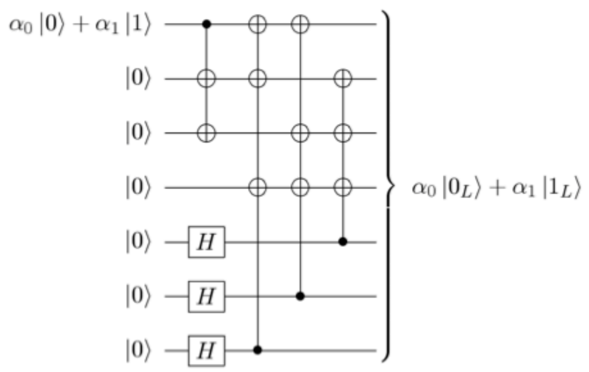
$$\begin{aligned} CNOT_\epsilon &= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X_\epsilon \\ &= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes R\left(\frac{\pi}{2} + \epsilon_\theta\right) P(\pi + \epsilon_\phi). \end{aligned} \tag{7}$$

2.2 Steane Code

Classically, it is straightforward to correct for errors by simply adding redundancy to a bit-value and performing a majority vote to detect any errors (that occur with low probability). In quantum computing, more sophisticated encoding and error detection methods are needed as a quantum state cannot simply be copied (in general) and direct measurements destroy any quantum superposition in the data.

In this project we have employed a 7-qubit encoding scheme called the Steane code [3, 4, 5], i.e. 7 physical qubits represent one logical qubit. Figure 3 shows the encoding circuit using gates from the universal set, as discussed previously, where the basis states $\{|0\rangle, |1\rangle\}$ are encoded into a logical basis $\{|0_L\rangle, |1_L\rangle\}$ according to

$$\begin{aligned} |0_L\rangle = \frac{1}{\sqrt{8}}(&|0000000\rangle + |0001111\rangle + |0110011\rangle + |0111100\rangle \\ &|1010101\rangle + |1011010\rangle + |1100110\rangle + |1101001\rangle) \end{aligned} \tag{8}$$



$\alpha_0 |0\rangle + \alpha_1 |1\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
H
H
$\alpha_0 |0_L\rangle + \alpha_1 |1_L\rangle$

Figure 3: Encoding circuit for the Steane code. Multiple $CNOT$ operations with the same control qubit commute and are therefore combined. The decoding circuit is the same circuit applied inversely.

and

$$|1_L\rangle = \frac{1}{\sqrt{8}}\big(|1111111\rangle + |1110000\rangle + |1001100\rangle + |1000011\rangle \\ |0101010\rangle + |0100101\rangle + |0011001\rangle + |0010110\rangle\big). \tag{9}$$

Once encoded, some of the logical gates from the universal set of gates can be applied transversally on the encoded qubit (Hadamard, Phase and controlled-NOT). In other words, manipulating the logical qubit corresponds to a manipulation of the constituent physical qubits, such that the following identities hold for the logical gates:

$$H_L = H \otimes H \otimes \ldots \otimes H = H^{\otimes 7} \tag{10}$$

$$CNOT_L = CNOT_{1,8} CNOT_{2,9} \ldots CNOT_{7,14}, \tag{11}$$

where the numbers in the subscript denote control and target qubit of the two-qubit $CNOT$-operation. Transversality is a clear advantage over other common encoding schemes (e.g. 5- or 9-qubit codes [4, 5]), as there is no need for intermittent decoding and re-encoding to allow for the application of gates. Once encoded, single errors per block can be detected by using syndrome measurements as depicted in figure 4. Note that this is done indirectly using $CNOT$-gates to entangle auxiliary qubits to the original system, so as to protect the encoded information.

2.3 Fault Tolerance

Taking the Steane code as an ideal correction algorithm guarantees that any single error per logical qubit block and correction step can be corrected with certainty. However, in reality, the circuits used for encoding, correction and decoding are made from the same building blocks as the error-prone algorithm to be protected. This means that the correction mechanism is erroneous itself and the next question arises: Can quantum error correction help at all? The answer to this question is given by the so-called threshold theorem that states that if the error probability is below a certain value, the correction procedure corrects more errors than it introduces itself. For this to work, the entire correction scheme needs to be fault-tolerant, meaning that any single error per block of 7 qubits and per correction step needs to be detectable. If this is the case and independent errors are assumed as in our error model, together with a single error probability p, a block will only fail if two or more errors occur within the correction step, which happens with probability $\mathcal{O}(p^2)$. The encoding and decoding (figure 3) are necessarily non-fault-tolerant (as they are not error corrected), but the syndrome measurement in figure 4 can be made fault-tolerant by removing the vulnerability to error-propagation in the ancilla qubits: If these are affected by a

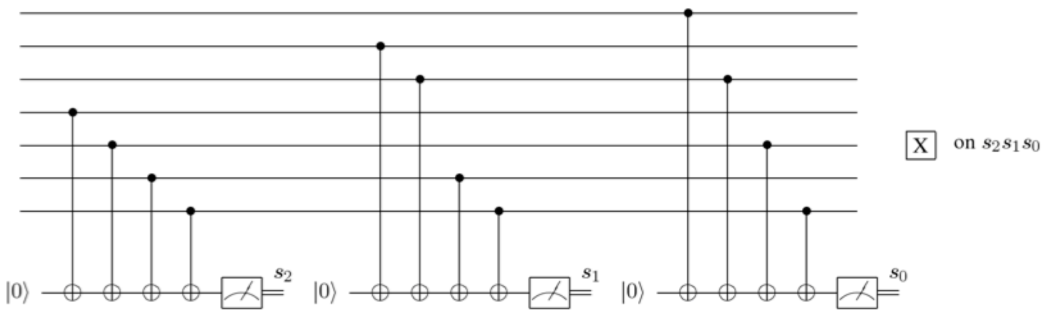


Figure 4: Indirect syndrome measurement for the Steane code using three ancilla qubits. The location of a Pauli-X error can be detected through the measurement results s_i, such that it can be undone by applying a Pauli-X matrix to qubit number $s_2s_1s_0$ (in binary notation). Repeating this circuit with logical Hadamard gates on either side (change of basis) enables the detection of a Pauli-Z error and therefore also a combination of the two, which is a Pauli-Y error. Note that this circuit is non-fault-tolerant. More ancilla qubits are needed for a fault-tolerant implementation.

single error, this error may propagate through the block undetected and may eventually be transferred to further blocks in the next correction steps. A fault-tolerant implementation is obtained by using 4 ancilla qubits in an entangled state called the *Shor state* and a further qubit to detect errors in the preparation of this state[1]. Accordingly, 5 qubits need to be added to the 7 qubits that encode the logical qubit, giving a total of 12 physical qubits per logical qubit.

3 Simulation

The Juelich Massively Parallel Ideal Quantum Computer Simulator (JUMPIQCS) [1] was used to find the threshold values based on the previously introduced error models. All 2^N complex amplitudes of the N-qubit quantum system are distributed among the available processes of the distributed memory parallel computer that is used and gates are performed as functions which update these amplitudes on the fly. The software includes the entire set of universal gates (figure 2) enabling simulations of arbitrary algorithms.

3.1 Error-Prone Simulations

In general, the introduction of errors requires the notion of mixed states with corresponding density matrices. However, as the memory needed to store the state vector of an N-qubit state is already 2^{N+4} bytes (using 8 byte double precision arithmetics) and would increase to 2^{2N+8} for a density matrix, an approach is implemented that uses pure states only. In this scheme, a general N-qubit density matrix ρ is sampled over M statistical runs, each yielding a resulting pure state $|\psi_i\rangle$. The final result is an approximation ρ_M according to

$$\begin{aligned}\rho_M &= \frac{1}{M}\sum_i m_i|\psi_i\rangle\langle\psi_i| \\ &\xrightarrow{M\to\infty} \sum_i p_i|\psi_i\rangle\langle\psi_i| = \rho,\end{aligned}$$

[1]See [1, 6] for details

where m_i/M is the relative frequency approximating the probability p_i of state $|\psi_i\rangle$.

3.2 Involution Algorithm

To study the effects of quantum error correction, an artificial algorithm is used. It is based on involutions of either the Hadamard gate (single qubit case, $H^{2k} = \mathbb{I}$, $k \in \mathbb{N}$) or, as studied here, the $CNOT$ gate (two qubit case, $CNOT^{2k} = \mathbb{I}$). This model gives the possibility of examining different algorithm lengths while providing a direct comparison to the ideal state, which simply remains unchanged.

In previous work by Trieu [1] the fault-tolerant implementation of the 7-qubit code was tested on single qubit algorithms of the form H^{2k}. The threshold in the case of decoherence errors was found to be $p_{thr} = (5.2 \pm 0.2) \cdot 10^{-6}$ and in the case of operational errors $\sigma_{thr} = (4.31 \pm 0.02) \cdot 10^{-2}$, where both thresholds were determined in absence of the other error source respectively.

4 Results

For the sake of saving memory space it is possible to use the 5 ancilla qubits for both logical qubits in the simulation of the $CNOT^{2k}$-algorithm, giving a total of $7 + 7 + 5 = 19$ qubits. As a measure for the similarity of the resulting erroneous state to the ideal state at the end of the algorithm, a quantity called the *fidelity* is used. The fidelity F between two mixed states ρ_1 and ρ_2 is defined as

$$F = \left[Tr\left(\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}\right)\right]^2, \tag{12}$$

which can be sampled with pure states according to

$$F \approx \frac{1}{M}\sum_{j=1}^{M} |\langle\psi_{init}|\psi_j\rangle|^2, \tag{13}$$

where M is the total number of iterations, $|\psi_j\rangle$ are the resulting states and $|\psi_{init}\rangle$ is the initial, unchanged state.

4.1 Decoherence

For the decoherence simulation perfect gates were assumed in order to extract the influence of decoherence errors only. In the uncorrected case of two qubits, the loss of fidelity can be determined analytically by the density matrix ρ^{2k} after $2k$ iterations, giving

$$\begin{aligned} F_{dec} &= \langle\psi_{init}|\rho^{(2k)}|\psi_{init}\rangle \\ &= \frac{1}{4}\left(1 + 3\left(1 - \frac{4}{3}p\right)^{2k}\right), \end{aligned} \tag{14}$$

where p is the error probability in eq. (3). This result is independent of the initial state and it is therefore convenient for assessing, in the error correction case, the impact of *different* initial states on the resulting fidelity. For this purpose, the separable initial state $|00\rangle$ and the maximally entangled state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ were chosen.

The results in figure 5 show that for both initial states the threshold error probability p_{thr} fulfils the constraint $10^{-5} < p_{thr} < 10^{-6}$. This is in agreement with the result of about $p_{thr} \approx 5 \cdot 10^{-6}$ in the single qubit case. The noise in the $p = 10^{-6}$ case is due to the high number of iterations required to sample the slight decrease in fidelity. On the JUGENE [8] architecture a total of only 8192 iterations took about 6 hours on a rack of 4096 processors.

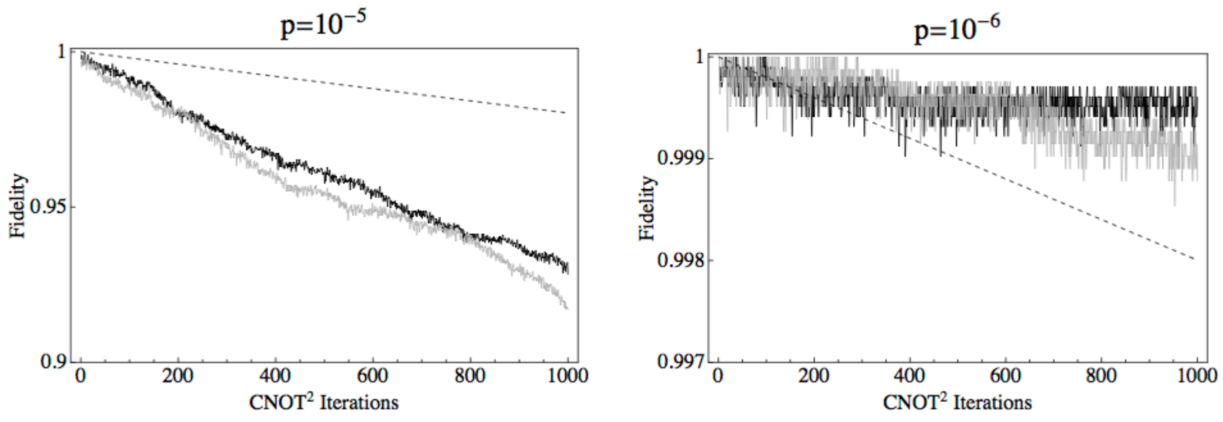


Figure 5: Decrease in fidelity for two unencoded qubits (dotted line) and the 19 qubit error correction case for the initial states $|00\rangle$ (black) and $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ (grey) with error probability $p = 10^{-5}$ (left) and $p = 10^{-6}$ (right). Number of statistical iterations: 2048 (left) and 8192 (right).

4.2 Operational Errors

In a similar fashion to the analysis of decoherence errors, the influence of operational errors was tested in the absence of decoherence, i.e. with $p = 0$. For unprotected qubits, the fidelity can again be calculated analytically by using density matrices and integrating over all possible Gaussian errors. Starting in a state ρ_k, the effect of operational errors in the time step $k \to k+1$ gives a density matrix ρ_{k+1} with

$$\rho_{k+1} = \frac{1}{2\pi\sigma^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} CNOT_\epsilon \rho_k CNOT_\epsilon^\dagger e^{-(\epsilon_\theta^2+\epsilon_\phi^2)/(2\sigma^2)} \mathrm{d}\epsilon_\theta \mathrm{d}\epsilon_\phi. \tag{15}$$

The fidelity is now dependent on the initial state: For the separable state[2] $|11\rangle$, the analytical result after $k\ CNOT^2$ iterations is

$$F_{op1} = \frac{1}{2}\left(1 + e^{-4\sigma^2 k}\right), \tag{16}$$

while for the singlet state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$, the same analysis yields

$$F_{op2} = \frac{3}{8} + \frac{1}{8}e^{-4\sigma^2 k} + \frac{1}{2}e^{-3/2\sigma^2 k}. \tag{17}$$

The results in figure 6 show an influence of the initial state on the utility of quantum error correction. While the order of magnitude confirms the threshold found in the single qubit case ($\sigma_{thr} \approx 0.043$), there is a factor of about 3 between the thresholds for the separable state (left, $\sigma_{thr} \approx 0.055$) and the one of the singlet state (right, $\sigma_{thr} \approx 0.02$). A source of this discrepancy may be the error model that introduces errors only on the target qubit, therefore causing the entanglement of the singlet state to be destroyed, which is accompanied with a decrease in fidelity.

[2] $|00\rangle$ would not be affected at all, which follows from eq. (7)

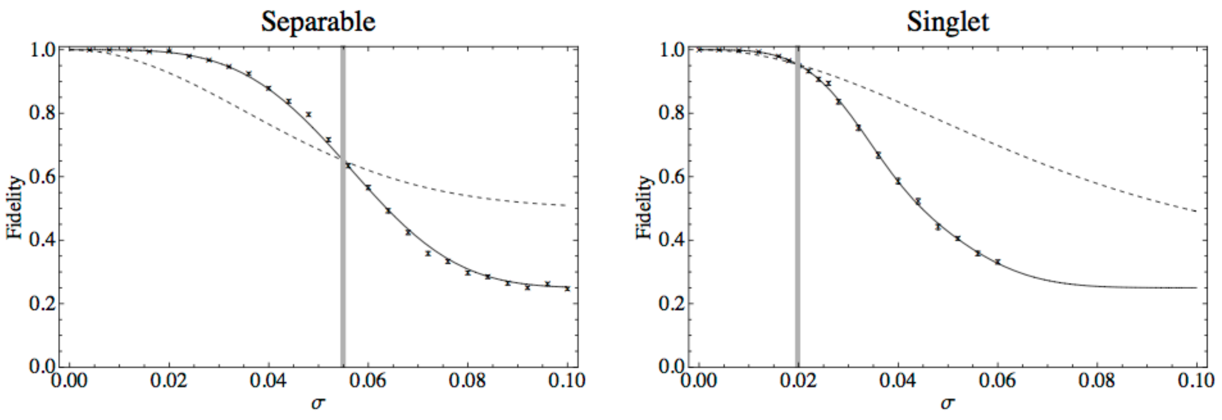


Figure 6: Value of the fidelity after 100 $CNOT^2$ iterations with varying standard deviation σ for operational errors. The dashed curves are the analytical results for two unencoded qubits (left: eq. (16), right: eq. (17)), the data points are the simulation results for error correction with 19 qubit (with spline fits) and the grey vertical line indicates the threshold. Left panel: initial state $|11\rangle$. Right panel: initial state $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. Number of statistical iterations: 4096 per data point.

5 Outlook

5.1 Modification of Error Model

The error model used in the case of operational errors gives an erroneous $CNOT_\epsilon$ matrix

$$CNOT_\epsilon = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\sin(\epsilon_\theta) & \cos(\epsilon_\theta)e^{i\epsilon_\phi} \\ 0 & 0 & \cos(\epsilon_\theta) & \sin(\epsilon_\theta)e^{i\epsilon_\phi} \end{pmatrix}, \tag{18}$$

which clearly shows that there is a bias on the target qubit, as only the lower right part does not correspond to the identity matrix. A next goal in the investigation of a fault-tolerant implementation of the $CNOT$-gate would therefore be a generalisation of the error model as a rotation similar to the single qubit case (see eq. (6)). As proposed in [9], a possible physical implementation of the $CNOT$-gate can be based on the NMR Hamiltonian

$$H_{\text{NMR}} = -JS_1^zS_2^z - h_1^zS_1^z - h_2^zS_2^z, \tag{19}$$

where J is the coupling strength between the two qubits and $h_{1/2}$ represents the strength of the applied magnetic fields. The time evolution $e^{itH_{\text{NMR}}}$, together with some additional rotations, yields the $CNOT$-gate, so a more general error model can be formed by incorporating errors into the Hamiltonian. As a result, a possible modification of the $CNOT$-gate can be the following:

$$CNOT_\epsilon \equiv e^{i\epsilon_1\vec{n_1}.\vec{\sigma_1}}e^{i\epsilon_2\vec{n_2}.\vec{\sigma_2}}CNOTe^{-i\epsilon_1\vec{n_1}.\vec{\sigma_1}}e^{-i\epsilon_2\vec{n_2}.\vec{\sigma_2}}. \tag{20}$$

Here $\vec{n_i}$ are random unit vectors chosen isotropically and $\vec{\sigma_i} = (X_i, Y_i, Z_i)^T$ is the vector of Pauli matrices for qubit i. The errors ϵ_i now correspond to fluctuations in the spin operators of the Hamiltonian (19).

5.2 Investigation of Arbitrary Algorithms

For the assessment of fault-tolerant error correction, a test of useful algorithms (rather than the artificial ones used here) would give more relevant results concerning the threshold. Unfortunately, the number of qubits that can be simulated to this day is very limited. The largest run as of yet only simulated 40 qubits, and the requirement of at least 7 qubits to encode a single logical qubit therefore currently limits the total number of logical qubits to about 5. With the necessary sampling of the density matrix and the associated large number of iterations, this number is reduced even further, making it impossible to find thresholds for larger algorithms. One possible approach to tackle this problem is to reduce the algorithm size to 2 or 3 qubits in order to infer the threshold for larger systems, for example with a 2 qubit quantum Fourier transform. However, this algorithm requires non-transversal gates like the $\pi/8$-gate, which would require even more resources when implemented. These problems are clearly overcome with simply more computing power and more memory space, so more progress can be made in the future.

6 Conclusion

With the previously mentioned restrictions in mind, we have shown that fault-tolerant quantum error correction works, within the introduced error model, for both decoherence and operational errors. Independent of the initial state, the threshold was shown to be approximately of the same order using either single qubit Hadamard or two qubit $CNOT$-gates. Once this threshold is reached, there is the possibility of refining the error correction by concatenation, i.e. by adding additional levels of error correction to the already encoded qubits ($7 \cdot 7 = 49$ physical qubits per logical qubit as a first concatenation step). While the threshold for operational errors can be reached with devices already existing today, the decoherence threshold is still out of reach for quantum computation. Nonetheless, once the thresholds are reached in the future, the methods for establishing arbitrarily good quantum error correction will be readily available.

7 Acknowledgements

First of all I would like to thank my advisor Binh Trieu for his constant support during the project. I would also like to thank Prof. Dr. Kristel Michielsen and Prof. Dr. Hans De Raedt for very thought-provoking discussions and, finally, Robert Speck for his superb organisation of the guest student programme.

References

1. D. B. Trieu, PhD-Thesis: *Large-Scale Simulations of Error-Prone Quantum Computation Devices*, Bergische Universität Wuppertal (2009)
2. S. Jordan, *Quantum Algorithm Zoo*, http://www.its.caltech.edu/∼sjordan/zoo.html, as of 09/10/2009
3. A. Steane, Proc. Roy. Soc. Lond. A **452** 2551-2577 (1996)
4. M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press (2000)
5. M. Nakahara, T. Ohmi, *Quantum Computing: From Linear Algebra to Physical Realizations*, CRC Press (2008)
6. H. K. Lo, T. Spiller, S. Popescu, *Introduction to quantum computation and information*, World Scientific Pub. Co. (1998)
7. S. Lloyd, Science **273**, 1073 (1996)
8. JUGENE (Jülicher BlueGene/P) Supercomputer, http://www.fz-juelich.de/jsc/jugene, as of 09/10/2009
9. H. De Raedt, K. Michielsen, A. Hams, S. Miyashita, K. Saito, Eur. Phys. J. B, **15-28** (2002)

Algorithmic optimization issues in quenched lattice gauge theory

Malik Kirchner

Humboldt University Berlin
Department of Physics,
Theory of Elementary Particles – Phenomenology
Newton Str. 15, 12489 Berlin

E-mail: malik@physik.hu-berlin.de

Abstract:
Calculations in lattice QCD are very time consuming and need an enormous amount of memory and computing power. Inverting the Dirac-operator is very costly. Many architectures compute faster in single than double precision. It is possible to speed up the inversion of the Dirac-operator by iterative "mixed precision" solvers by a factor of almost two. The Monte-Carlo integration is done by a multi-hit Metropolis algorithm based on random uniformly distributed SU(N) elements.

1 QCD on the lattice

Quantum Chromo Dynamics QCD is the modern theory of quark gluon interactions. It is part of the Standard Model of elementary particle physics. The early construction of QCD and the Standard Model itself were based on observed symmetries in the spectrum of particles. These symmetries were coded into a Hamiltonian, including all known interactions beside gravity. It is possible to transform this Hamiltonian into a Lagrangian form which is, quite apparently, Poincaré invariant (Legendre transformation). The part concerning strong interactions was the starting point of my investigations. In Minkowski space it takes the form

$$\mathcal{L}(\Psi, \bar{\Psi}, \partial\Psi, \partial\bar{\Psi}, A, \partial A) = \bar{\Psi}(x)\left(\mathrm{i}\not{D}(x) - m\right)\Psi(x) - \frac{1}{4}\mathrm{tr}\left(F^{c}_{\mu\nu}(x)F^{c,\mu\nu}(x)\right). \tag{1}$$

The fields Ψ and $\bar{\Psi}$ are fermionic and hence anticommuting. These are the elementary quark fields. The field strength tensor

$$F^{c}_{\mu\nu}(x) = \partial_\mu A^c_\nu(x) - \partial_\nu A^c_\mu(x) + g f^{cab} A^a_\mu(x) A^b_\nu(x) \tag{2}$$

contains the bosonic gluon fields. The last term vanishes in the Abelian case, where the structure constants f^{cab} are trivial. The coupling to the fermionic fields is expressed in the covariant derivative

$$D_\mu = \partial_\mu \otimes \mathbb{1}_{N\times N} - \mathrm{i}g A^c_\mu \tau^c, \tag{3}$$

with $\not{D} = \gamma^\mu D_\mu$. The so-called gauge Lie group has generators τ^c. The vector potential A_μ lives in the algebra of the gauge group and is therefore a linear combination $A_\mu = A^c_\mu \tau^c \in \mathfrak{su}(N)$ of the generators.

It has been shown[1], that the only possible choices of gauge groups[2] are $\mathrm{U}(1)$ and $\mathrm{SU}(N)$. The gauge group of the electro weak interaction is $\mathrm{SU}(2) \times \mathrm{U_Y}(1)$, which are spontaneously broken via Higgs's mechanism. The strong interaction is described by a $\mathrm{SU}(3)$ local gauge group.

The coupling of the theory depends on the renormalization scale. In some cases this is indirectly the energy scale of the problem. QCD is an asymptotically free theory, hence it has a vanishing coupling for ultra high energies and an extremely high coupling for very small energies. It is possible to do asymptotic expansions in the coupling g to describe scattering amplitudes for high particle energies. In the case of low energies one can expand in the inverse coupling g^{-1}. This is only possible with static fermions. The Noether charge corresponding to the $\mathrm{SU}(3)$ gauge symmetry is called colour. It cannot be observed in experiments directly – bound states of quarks must be colour singlets because of Fermi's principle. This property of QCD is called confinement and cannot be generated within perturbation theory. Therefore one needs non-perturbational methods to describe real bound states like Mesons and Baryons.

The N-point correlation functions of the theory can be used to do spectroscopy of a N-particle bound state and can be expressed by Feynman path integrals:

$$\langle \bar\Psi(x_1)\cdots\Psi(x_N)\rangle = \frac{1}{Z_0}\int \mathcal{D}\bar\Psi\mathcal{D}\Psi\mathcal{D}A\ \bar\Psi(x_1)\cdots\Psi(x_N)e^{-iS[\Psi,\bar\Psi,\partial\Psi,\partial\bar\Psi,A,\partial A]}, \tag{4}$$

where the action S is the 4-integral over $\mathcal{L}$. Path integrals of this type are not well defined. They need a regulator. Here enters lattice QCD . It is possible to define a spacetime lattice of finite size, calculate the correlations functions and do the continuum limit again with the help of phase transitions. One can determine e.g. the mass ratios of different bound states.

The non-Abelian nature of the interaction results in the self interaction of the gluons. They can bind[3] without fermions! My calculations are done in pure gauge theory. I have neglected the dynamics of the fermions, which reduces the simulation to only gluodynamics. It is still possible to determine some fermionic observables, which are astonishingly realistic.

The discretization of the spacetime and fermionic fields is done as follows.

$$\begin{array}{llll} x^\mu & \longrightarrow x_n^\mu = an^\mu & \int \mathcal{D}\Psi & \longrightarrow \prod_{n=0}^{N}\int \frac{d\Psi_n}{\sqrt{2\pi a^4}} \\ \Psi(x) & \longrightarrow \Psi_n = \Psi(x_n) & \int \mathrm{d}^4x & \longrightarrow a^4\cdot\sum_n \end{array} \tag{5}$$

Let us assume the number of lattice sites is finite, this allows an estimate of the problem's size. The are as many integrals as lattice sites to evaluate by eqn. (5). Each of these integrals covers the whole lattice. To achieve a physically realistic situations the lattice volume needs to be large and the lattice spacing small against the scale of the events to observe. Doing these integrals by Monte-Carlo approximation is the only known way to compute them in reasonable time. The most time is spend to matrix multiplications of the gauge group elements and spinors. Let N_C be the number of colours and L be the number of lattice sites in one direction. The number of FLOPS will scale approximately like $L^4 \times N_C^2$. Realistic dimensions for today's workstations are $L = 16\ldots32$ and $N_C = 2\ldots5$.

The covariant derivative was constructed under the assumption of an arbitrary small distance between two neighbouring points in spacetime. This is no longer true, because of the finite lattice spacing. The

[1]The Coleman-Mandula theorem is based on following assumptions:

1. The S-Matrix is based on a local, relativistic quantum field theory in 4-spacetime;
2. there are only a finite number of different particles associated with one-particle states of given mass and
3. there is an energy gap between the vacuum and the one particle states.

There exists a supersymmetric extension to graded $\mathrm{SU}(N, M)$ groups by Haag-Lopuszanski-Sohnius theorem.

[2]More exactly: the symmetry Lie-group of the S-Matrix is a direct product of the Poincaré-group and a semisimple internal gauge Lie-group.

[3]Bound states of gluons only are called "glue-balls".

mathematical structure[4] of a Yang-Mills theory like QCD allows to define a parallel transporter:

$$U(x_2, x_1) = \mathcal{P}e^{\mathrm{i}g\int_{x_1}^{x_2} A(x(t))\mathrm{d}t}, \tag{6}$$

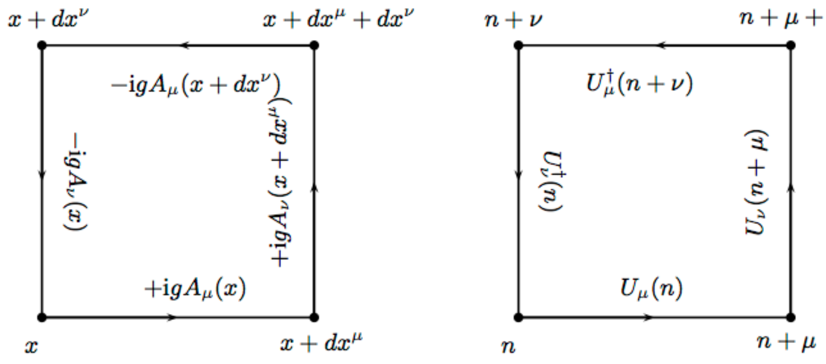


Figure 1: The plaquette is the smallest loop along the links of a lattice. The gauge fields are assigned to the links of the lattice. They take the role of an affine connection in the continuous case. In the discretized version they are parallel transporters along the links.

where $x(t)$ is a continuously differentiable path from x_1 to x_2. The pathordering symbol $\mathcal{P}[\cdot]$ makes sure that the path is monotonously parameterized in t. This parallel transporter simplifies in terms of its Taylor expansion to A_μ itself if the path is arbitrary short. Let us define the so-called Wilson-link $U_\mu(n) \equiv U(x_n + ae_\mu, x_n) = U^\dagger(x_n, x_n + ae_\mu)$, where γ is the direct line from x_n to $x_n + ae_\mu$. The symmetric discrete covariant derivative of Ψ is written using the Wilson-link.

$$\nabla_\mu \Psi_n = \frac{1}{2a}\left(U_\mu(n)\Psi_{n+\mu} - U_\mu^\dagger(n-\mu)\Psi_{n-\mu}\right) \tag{7}$$

The dual lattice corresponds to the momentum space. A regular lattice has a Brillouin zone of finite size. At the surface of the Brillouin zone and at the origin the lattice momentum vanishes. This would lead to non physical states, because the Green's function of the Dirac-operator would have extra residues for each solution of $p^2 + M^2 = 0$ with bare quark mass M. But one can shift these solutions by adding a second order derivative, which is vanishing in the continuum limit. The content of only one Brillouin zone is sufficient to describe a system on a lattice with periodic boundaries. With this in mind a lattice analogue of the Dirac operator can be defined:

$$D^W_{mn} = (4r + aM)\delta_{mn} - \frac{1}{2a}\sum_\mu \left[(r - \gamma_\mu)U_\mu(n)\delta_{m,n+\mu} + (r + \gamma_\mu)U_\mu^\dagger(n-\mu)\delta_{m,n-\mu}\right]. \tag{8}$$

The superscript 'W' stands for Wilson who first proposed this operator. The constant $r(\equiv 1)$ shifts the residues.

2 Mixed Precision Solver

The fermion propagator contains the solution of the equations of motion of the system. For lattice calculations it is necessary to do a Wick rotation of the spacetime. This transforms a Minkowski metric to a Euclidean metric by analytic continuation to the complex plane. It is now possible, e.g., to determine the

[4]The gauge fields living in the vectorbundle of the gauge algebra over the spacetime manifold.

mass of bound states. To do that one must invert the Dirac operator. The fermionic part of the action is integrated[5] and set to a constant:

$$\langle \mathcal{T}\left\{\bar{\Psi}(x_1)\Psi(x_0)\right\}\rangle = \frac{\det D}{Z_0}\int \mathcal{D}U\ \bar{\Psi}(x_1)\Psi(x_0)e^{-S[U]} = \langle D^{-1}[U](x_1, x_0)\rangle \tag{9}$$

$$\det D \overset{!}{\approx} \text{const.} \tag{10}$$

This is connected to the Euclidean Green's function of the system in the means of contraction [5, p. 32] and [5, p. 109]. The time ordering symbol $\mathcal{T}[\cdot]$ makes sure causality is preserved. Propagators are not gauge invariant – they need gauge fixing.

The Dirac operator is linear and local. Hence it can be written as sparse matrix eqn. (8). The linear system

$$Ax = b \qquad\qquad r_i = b - Ax_i \tag{11}$$

stands for that. The vector x_i is the best available guess and r_i its residual. One is interested in the application of the inverse Dirac operator to a source vector, to determine its propagation. There are several iterative solver which project the inverse matrix application on a vector to a Krylov subspace, which is the span of monomes of the matrix itself applied the same vector:

$$A^{-1}b = a_0 b + a_1 A^1 b + a_2 A^2 b + a_3 A^3 b + \cdots . \tag{12}$$

A very efficient choice is the Conjugate Gradient algorithm (CG) which was implemented as described in [6, p. 176]. It comes with the drawback that it is only applicable to hermitian and positive definite matrices. The Dirac operator itself is not, but $(D^\dagger D)^{-1}D^\dagger = D^{-1}$ is a workaround for that[6]. An alternative to CG is the Minimal Residual algorithm (MinRes). To solve the problem directly[7] one can use the bi-Conjugate Gradient Stabilized algorithm (BiCGStab) [6, p. 217] or the Generalized Minimal Residual algorithm (GMRes).

The desired accuracy of these solvers is of order 10^{-13}, which can be done in at least double precision. In real world simulations this is a problem. The partial derivatives require at least nearest neighbour knowledge, which means on distributed memory systems one must communicate these at each application of the Dirac operator. There is a possibility to reduce the amount of data by using "mixed precision" solvers. An outer solver refines the solution with simple iteration steps in double precision, alg. 1, and an inner solver uses one of the mentioned algorithms to calculate the necessary corrections t_i in single precision, where a is a single precision version of A. A second benefit from that is that on systems

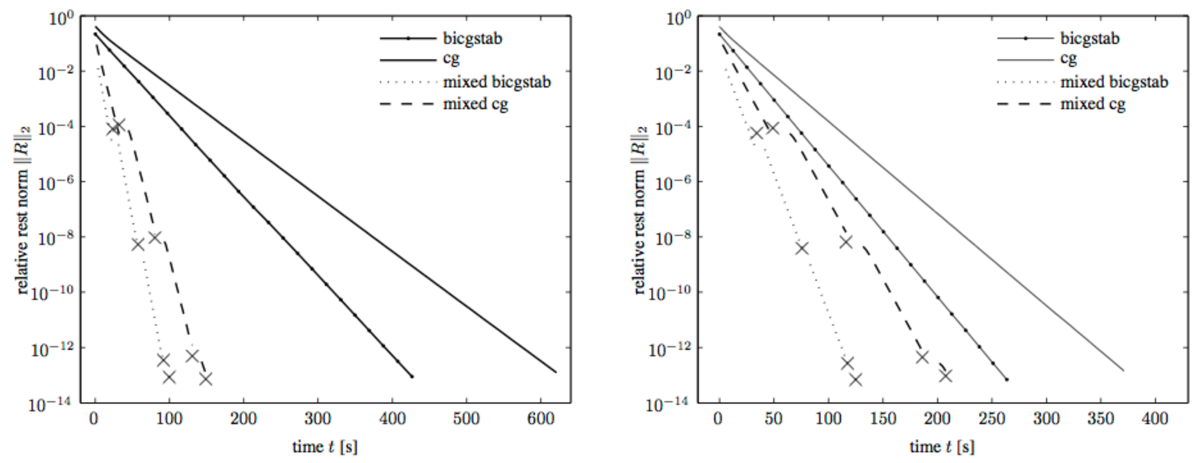
Algorithm 1 simple iteration

```
repeat
    compute $r_i = b - Ax_i$
    solve $at_i = r_i$ in sgl. prec. with, e.g., CG
    update $x_{i+1} = x_i + t_i$
until $\|r_i\| \leq \epsilon\|b\|$
```

which have SIMD[8] extensions one can do twice as many floating point operations as before. But there are drawbacks, too. For all used fields there must be a single precision copy.

The residuals for CG ($A = D^\dagger D$), BiCGStab ($A = D$) and their mixed versions are shown in fig. 2. On the test machine[9] single and double precision are processed at same speed. To emulate the effect

[5]It is a gaussian path integral of the shape $\int \mathcal{D}\bar{\Psi}\mathcal{D}\Psi\ \exp\left(-\bar{\Psi}M\Psi\right) = \det M$, where Ψ and $\bar{\Psi}$ are grassmannian.
[6]$A = D^\dagger D$
[7]$A = D$
[8]On Intel/AMD platforms the SSE series and AltiVec on PPC platforms.
[9]An Intel Core2 quad system.



(a) $\mathrm{SU}(2),\ 32^4,\ \beta = 2.4$ (b) $\mathrm{SU}(3),\ 24^4,\ \beta = 6.0$

Figure 2: The residual of a CG, BiCGStab and their "mixed versions" is shown for a thermalized $\mathrm{SU}(2)$ configuration on a 32^4 lattice and a thermalized $\mathrm{SU}(3)$ configuration on a 24^4 lattice. The mixed versions are at least $1.7\times$ as fast as their double precision counterparts. In the $\mathrm{SU}(2)$ a much higher speed up is observed.

64bit double and 128bit long double were used instead of 32bit float and 64bit double. The much higher speed up for the $\mathrm{SU}(2)$ case might be a memory bandwidth or cache/prefetch effect. This is not clear up to now, because the used compilers do some vectorizing optimizations, which are not apparent to the programmer.

The speed up my "mixed precision" is quiet significant and might be even higher on distributed memory systems. To evaluate the path integral one needs to average over many inversions on different configurations (gauge fields).

3 Multi-hit Metropolis

The Metropolis algorithm [4] is a simple but effective way to achieve a Gibbs sampling of phase space. Regions which bring a significant contribution to the pathintegral are sampled more dense, than others. The Metropolis algorithm constructs a Markov chain of gauge fields. A single Metropolis update is expensive, because one must calculate the difference of the gauge action involving six plaquettes:

$$S[U] = \beta \sum_{\mu<\nu,\, n} \left[1 - \frac{\Re\,\mathrm{tr}}{N_C} \left(U_\nu^\dagger(n) U_\mu^\dagger(n+\nu) U_\nu(n+\mu) U_\mu(n) \right) \right] \tag{13}$$

The inverse temperature β is not related to the physical temperature! It is the temperature of the Gibbs ensemble. The physical temperature is zero if the box length is sufficiently large ($\gg 1$fm) in each direction.

Some of the needed matrix products can be reused if one does many update tries at each link. The justification for that can be found in [3]. A possible algorithm is shown in alg. 2. The resulting configurations can be slightly more correlated for many hits, but the speed up is significant. In fig. 3 the thermalization phase and the resulting distribution of gauge fields is shown. The resulting distributions after the thermalization are independent of the number of hits.

The logarithmic scale of the number of effective[10] updates implies, that linear graphs in the plot show

[10]Rescaled to the equivalent number of one hit sweeps.

	CG	mixed CG	BiCGStab	mixed BiCGStab
$SU(2), 32^4, \beta = 2.4$	1.00	4.1(7)	1.4(6)	6.2(3)
$SU(3), 24^4, \beta = 6.0$	1.00	1.7(9)	1.4(1)	2.9(7)

Table 1: This table shows the speed up of different algorithms compared to CG for different setups.

	1	2	3	4	5
$SU(2), \beta = 2.4$	1.00	1.5(7)	1.9(3)	2.1(7)	2.4(9)
$SU(3), \beta = 6.0$	1.00	1.5(3)	1.9(2)	2.1(7)	2.4(0)

Table 2: The table above lists the speed up for an elementary update step against the number of hits.

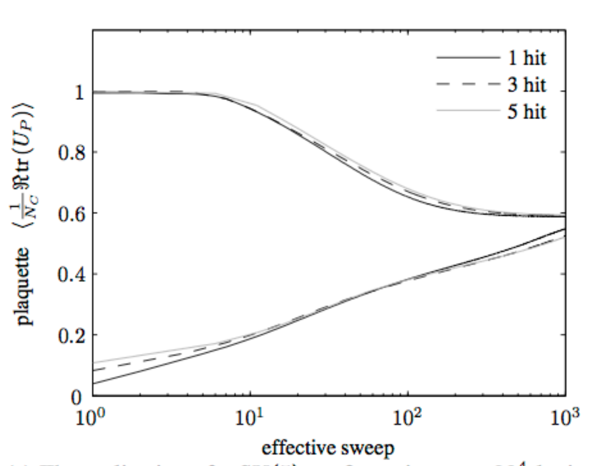
a polynomial behaviour. For low ensemble temperatures a start from a constant configuration is wise. The effective number of samples is dependent on the integrated autocorrelation time $N_{\text{eff}} = N/(2\tau_{\text{int}})$, as described at [2, p. 432]. There might be an optimal number of hits, hence the autocorrelation rises a bit with the number of hits. To estimate it one must measure the autocorrelation for different lattices, temperatures, step sizes and gauge groups. A further benefit of the multi-hit Metropolis algorithm is, that the stepsize can be higher the more hits are done. This lowers the autocorrelation again.

4 SU(N) Random Matrices

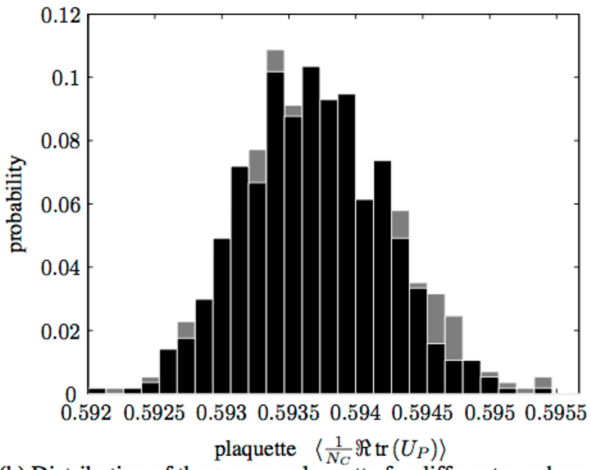
The last thing missing to take the expectation values are random gauge group elements. The use of high quality pseudo random numbers is crucial to Monte-Carlo simulations to keep autocorrelation low. I have used a Mersenne twister (mt19937) random number generator, because this type of generators has a huge period and produces highly uncorrelated and evenly distributed random numbers. But that is not sufficient to guarantee detailed balance in the Metropolis algorithm. Evenly distributed SU(N) elements are needed. The SU(N) Lie groups are semi simple. Especially U(1) and SU(2) are simple. It is therefore

Algorithm 2 multi-hit Metropolis, one sweep

```
1: U = given, const. or random configuration
2: for all links $U_\mu(n)$ do
3:   $S = \sum_{\mu\neq\nu} U_\nu^\dagger(n)U_\mu^\dagger(n+\nu)U_\nu(n+\mu) + \sum_{\mu\neq\nu} U_\nu(n-\nu)U_\mu^\dagger(n-\nu)U_\nu^\dagger(n-\nu+\mu)$
4:   $s_{\text{old}} = 6 - \frac{1}{N}\Re\,\text{tr}\,[S \cdot U_\mu(n)]$
5:   for number hits do
6:     $A = \text{uni}\,[\text{SU}(N)]$ close to $\mathbb{1}_{N\times N}$
7:     $U'_\mu(n) = AU_\mu(n)$
8:     $s_{\text{new}} = 6 - \frac{1}{N}\Re\,\text{tr}\,[S \cdot U'_\mu(n)]$
9:     if $\text{uni}(0,1) \leq \exp[-\beta(s_{\text{new}} - s_{\text{old}})]$ then
10:      $s_{\text{old}} = s_{\text{new}}$
11:      $U_\mu(n) = U'_\mu(n)$
12:    end if
13:  end for
14: end for
```



(a) Thermalization of a SU(5) configuration on a 20^4 lattice with $\beta = 18.0$



(b) Distribution of the average plaquette for different numbers of hits.

Figure 3: The left side plot shows the average plaquette in the thermalization phase of the SU(5) theory on a 20^4 lattice for $\beta = 18.0$ against the number of effective sweeps. The right side plot states, that the distribution of average plaquette values is independent of the number of hits.

possible to compose a random SU(N) element from random SU(2) elements in a very efficient way:

$$U \in \mathrm{SU}(N)\,, \quad S_i \in \mathrm{SU}(2)\,, \quad n = \frac{1}{2}N(N-1),$$
$$U = \bigotimes_{i=1}^{n} S_i. \tag{14}$$

Hence the problem is reduced to sampling SU(2) as even as possible. It is a simple linear group, a general SU(2) group element is therefore written

$$S_i = s_0 \mathbb{1}_{2\times2} + \mathrm{i}s_1\sigma^1 + \mathrm{i}s_2\sigma^2 + \mathrm{i}s_3\sigma^3, \tag{15}$$

where σ^i are ordinary Pauli matrices. The algebra $\mathfrak{su}(2)$ is isomorphic to $\mathfrak{so}(3)$, the groups are only homomorphic by some homomorphism

$$\varphi: \quad \mathrm{SU}(2) \longrightarrow \mathrm{SO}(3) \qquad\qquad \ker\varphi = \{-1,\, +1\}\,. \tag{16}$$

That is why the parameters s_i can be mapped to a 3-sphere and vice versa:

$$\begin{aligned} s_0 &= \cos\alpha & s_1 &= \sin\alpha\sin\theta\cos\phi \\ s_2 &= \sin\alpha\sin\theta\sin\phi & s_3 &= \sin\alpha\cos\theta \end{aligned}$$

with

$$\phi \in [0, 2\pi), \qquad\qquad \alpha, \theta \in [0, \pi]\,.$$

The Jacobian of 3-spherical coordinates reads $r^3 \sin^2\alpha \sin\theta$. To sample S^3 evenly ϕ and $\cos\theta$ are sampled evenly. The distribution in α is not that apparent. The distribution function of α is not invertible as such:

$$F(\alpha) = \frac{2}{\pi}\int_0^\alpha \mathrm{d}\beta\, \sin^2\beta = \frac{\alpha}{\pi} + \frac{1}{2\pi}\sin 2\alpha. \tag{17}$$

Algorithm 3 SU(N) random matrices

```
1: if $N = 1$ then
2:     return $\exp[\mathrm{i} \cdot \mathrm{uni}(0, 2\pi)]$
3: end if
4: $U = \mathbb{1}_{N\times N}$
5: for $i = 0$ to $N-2$ do
6:     for $j = i+1$ to $N-1$ do
7:         $\alpha = F^{-1}[\mathrm{uni}(0, \pi/8)]$
8:         $\phi = \mathrm{uni}(0, 2\pi)$
9:         $\cos\theta = \mathrm{uni}(-1, 1)$
10:        $\sin\theta = \sqrt{1-\cos^2\theta}$
11:        $s_0 = \mathrm{uni}(-,+)\sqrt{1-\sin^2\alpha}$
12:        $s_1 = \sin\alpha \sin\theta \cos\phi$
13:        $s_2 = \sin\alpha \sin\theta \sin\phi$
14:        $s_3 = \sin\alpha \cos\theta$
15:        $S = s_0\mathbb{1}_{2\times 2} + \mathrm{i}s_1\sigma^1 + \mathrm{i}s_2\sigma^2 + \mathrm{i}s_3\sigma^3$
16:        $U' = U$
17:        for $k = 0$ to $N-1$ do
18:            $U'_{ik} = S_{00}U_{ik} + S_{01}U_{jk}$
19:            $U'_{jk} = S_{10}U_{ik} + S_{11}U_{jk}$
20:        end for
21:        $U = U'$
22:    end for
23: end for
24: return $U$
```

The inverse distribution function creates the needed distribution in α from a uniform distribution in $[0,1]$. There are different ways to invert it on an interval, e.g. an expansion of the inverse.

$$\alpha = F^{-1}(a) = \frac{4}{\pi}\left[6^{\frac{1}{3}}a^{\frac{1}{3}} + \frac{2}{5}a + \frac{12}{175}6^{\frac{2}{3}}a^{\frac{5}{3}} + \frac{16}{175}6^{\frac{1}{3}}a^{\frac{7}{3}} + \mathcal{O}\left(a^3\right)\right] \tag{18}$$

An expansion of this kind works well[11] for $a \in [0, 1/2]$ which is mapped to $\alpha \in [0, \pi/2]$. To cover the second half of the interval $\alpha \in [0, \pi]$ one chooses a random sign in front of $\cos\alpha$ terms. This works, because $\cos\alpha$ is point symmetric to $\alpha = \pi/2$. Alternatively one might solve $F^{-1}[\cdot]$ numerically e.g. using Newtons rule. Based on that an algorithm alg. 3 can be formulated, based on [1, p. 17]. The step size in the SU(N) group can be modified by restricting $\cos\alpha$ to values close to one. This results in small values of $\sin\alpha$ and group elements close to the identity. This step size can be used to modify the acceptance rate of the Metropolis algorithm. This is totally fine if this is done in the thermalization phase only and only a few times. A reasonable acceptance rate for elementary updates is $0.4\ldots0.5$.

5 Summary

Mixed precision solver bring significant speed up to lattice QCD simulations. They yield the same precision as their ordinary variants by using lower precision for time critical parts. This reduces the amount of data to be sent by a factor of two. Processors with SIMD extension can also do up to twice the number of FLOPS. The production of configurations in pure gauge theory can be accelerated by reuse of matrix

[11] A relative error of order 10^{-4} of an expansion of tenth order.

products in Metropolis update steps. A five hit update results in at least twice the number of elementary update steps in the same time. The generation of uniformly distributed gauge group elements as desired for a justified Metropolis algorithm is expensive. The described algorithm generates them with a rate of $\mathcal{O}(200mb/s)$ on the test workstation. The used algorithms enable an ordinary todays workstation to do pure gauge lattice QCD simulations in reasonable times. Especially the "mixed precision" solvers can bring great speed up on distributed memory systems.

References

1. Massimo Di Pierro. An algorithmic approach to quantum field theory. *Int. J. Mod. Phys.*, A21:405–448 (2006).
2. Wolfhard Janke. Statistical Analysis of Simulations: Data Correlations and Error Estimation. In *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, NIC Series, pages 423–445. NIC (2002).
3. J. S. Liu, F. Liang, and W. H. Wong. The use of multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134 (2000).
4. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092 (1953).
5. Heinz J. Rothe. *Lattice Gauge Theories, An Introduction*, volume 74 of *World Scientific Lecture Notes in Physics*. World Scientific Publishing Co. Pte. Ltd., 3^{rd} ed. edition, (2005).
6. Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Cambridge University Press, 2^{nd} ed. edition (2000).

Single-File File System
A Library for Massively Parallel I/O

Łukasz Kucharski

Adam Mickiewicz University in Poznań, Faculty of Physics
Umultowska 85, 61-614 Poznań

E-mail: luk32@o2.pl

Abstract:
This work focuses on integrating *SIONlib* into ProFASi by introducing a new middle layer library called *Single-File File System*. *SFFS* aim is to extend *SIONlib* capabilities to make it a general parallel I/O replacement for standard *C/C++* paradigms. This paper introduces *SFFS* concepts and discusses achieved ProFASi performance gains and the integration process.

1 Introduction

1.1 The Problem

There are some applications that are inherently parallel and can achieve good scalability even to systems with hundreds of thousands processes. It is possible and tempting to extend such application by using MPI (Message Passing Interface) just for communicating between processes and distributing the work among them, when the rest of the code base is portable. With such a simple approach certain phenomena may manifest when the problem is put onto large computing systems. One such problem is to deal with massive amounts of files created by all the tasks during a run. An application where a task manages all of its files locally can easily use about $10 - 15$ files per process. Such a situation is perfectly fine for a serial application or small parallel systems. However, on the scale of hundreds of thousands tasks the number of files automatically grows to millions. Even though file systems on supercomputers are designed to perform well in parallel environments, a locking on working directory occurs each time a new file (or directory) is created. This leads to serialization of the tasks as long as they operate on the same directory which usually is the case. Measurements have shown that the time used to create the file structure can exceed 15 minutes on 16384 processes, which equals to 2730 CPU hours.

Another problem is that usually such file systems use rather large basic block size, e.g. 1 MB on JuRoPA's LUSTRE and 2 MB on JUGENE's GPFS. This means that every atomic operation on the I/O server actually involves multiples of 1 or 2 MB (respectively for given examples) regardless of the actual size of the data. This can lead to performance degradation if one does not take special care about the I/O operations structure.

1.2 ProFASi

ProFASi stands for Protein Folding and Aggregation Simulator [2]. It is a good example of an application hit by the aforementioned problems.

It was parallelized using the parallel tempering method. In parallel tempering independent copies (replica) of the system are simulated at different values of a control parameter, most often, the temperature. After some time an exchange of two replica at neighbouring temperatures T_i and T_j is attempted. The acceptance probability is given by $P_{\mathrm{PT}}(i,j) = \min(1, \exp((\beta_i - \beta_j)(E_i - E_j))$, where $\beta_i = \frac{1}{k_B T}$ with k_B the Boltzmann constant and E_i is the total energy of replica i. The exchange move results in a random walk of the replica in temperature space. By moving to higher temperatures, replicas can escape local minima more easily.

It is tempting to collect the statistical data including histograms, averages, and conformations in a separate directory for each replica. This keeps the logic of the code simple and also provides a clean structure for post processing. Each process creates about 15 files of different types including human-readable alphanumeric data for convenient viewing of computed observable values and binary data types meant to be snapshots for restarts or to be used as input for visualization software.

Parallel tempering makes ProFASi scalable even to the largest existing parallel computers such as JUGENE. Because the parallel version was originally developed for small work-station class PC it used the standard *C/C++* paradigms and methods for I/O operations. This became a concern of the developers. As they started to use ProFASi on larger and larger number of cores, I/O introduced a considerable negative impact on the performance. Moreover used computing methods and the application itself are being constantly improved so the relative influence of the file access on overall performance is rising to an unacceptable level.

1.3 SIONlib

SIONlib (Scalable I/O library for Native parallel access to binary files) [1] combines file output from each task into one large shared file. This alone solves the file creation problem. It is optimized to fully utilize the underlying parallel file system and provides an easy to use interface to make the adaptation of the existing code as easy as possible.

Usually, if one uses standard *C posix* functions, only *fopen() / fclose()* need to be changed to their *SIONlib* counterparts which are *sion_paropen_mpi() / sion_parclose_mpi()* respectively. After opening the *SION* file the user is given a standard *FILE** pointer and can perform write operations using unmodified *fwrite()* call. *SIONlib* assigns every process a chunk of a shared *SION* file, which is always aligned to base file system block. There is no need for any locking between writing tasks as each process "owns" and is allowed to write only to its allocated chunk (thus file system blocks). This allows effective parallelization of I/O operations almost transparently to the user. The only collective calls needed are for opening and closing the *SION* file.

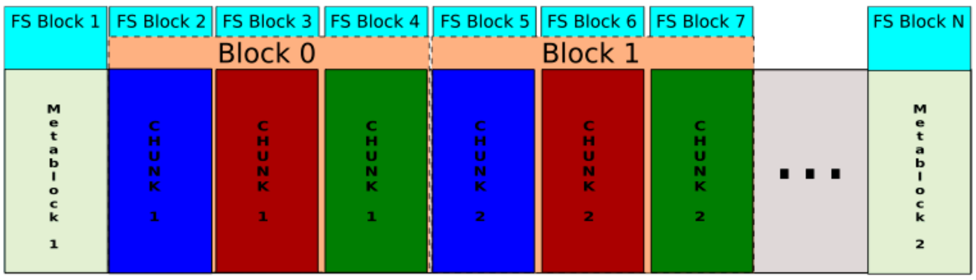


Figure 1: Typical *SION* file structure. The figure shows how data chunks of each process are organized inside the collective file. One can see they are aligned and sized to match exactly the underlying base file system structure [3].

Although the outline of *SIONlib* advantages seems to make it a perfect solution for ProFASi and similar applications it still introduces some drawbacks. Under certain conditions they can even make the portability goal impossible to achieve. If one would like to use unmodified *fwrite()* function special care needs to be taken not to violate the process' chunk boundaries. *SIONlib* provides the necessary functions and wrappers to ensure the correct behaviour but it implies further changes to the existing code and may introduce some unused space in the chunks. When the user wants to write data of size greater than the space left in the chunk, the current solution is to simply allocate next chunk for the process and write in there. This fragmentation can be dealt with by a utility provided with *SIONlib* as a post-process step. It is not possible to write a block of size larger than the specified chunk size, as even allocating a new empty one would not be sufficient. The user's application would have to split and buffer the data before writing it to a *SION* file, obviously making the whole code adjustment process more complicated.

Another peculiarity of *SIONlib*, which might make it hard to utilize it, is the fact, that the current implementation allows a process to have only one binary stream (task local file) per *SION* file. This might be easily overcome by using several *SION* files within a process, one per a filetype that the process uses. This solution is not always an available and wise approach to take. It can be very troublesome to fully utilize *SIONlib* capabilities if local files are too small to fill up a chunk, which often is the case with statistical alphanumeric data. In general, the user could be left with several *SION* files and lots of wasted space, at least during the run.

The last major concern we encountered during our initial evaluation of *SIONlib* capabilities and ProFASi needs is that *SIONlib* does not support random write access, even assuming that the user takes the responsibility of preserving *SION* file consistency, especially not violating chunk ownership. This means that one can treat the "logical" local file given by the *FILE** pointer from *SIONlib* only as a serial output stream. This issue usually implies a redesign of the I/O model if the application rewrites files during the run as it is not possible to provide this functionality using *SIONlib* routines without major performance hits. All of the tasks would have to collectively close the file and rewrite its data.

It may seem that solving presented problems might be a cumbersome process. It should be clearly stated that the library is still in an early stage of development and its original goal was to introduce an easy to use and efficient mechanism to do checkpoints and restarts. For this purposes *SIONlib* performs exceptionally well since stated troublesome use-cases are rare. After consultations with ProFASi and *SIONlib* developers it was decided that the library has the potential to be efficiently utilized as a general I/O replacement to standard calls for massively parallel systems with the introduction of a middle layer.

2 SFFS

2.1 Motivation and General Ideas

SFFS is an acronym for Single-File File System. The main purpose and starting point of the library was to fully utilize *SIONlib* functionality without the need to alter existing I/O operations structure. This was to be done by combining all the task-local files into one binary stream, which would act and be treated as task-local and then be written using *SIONlib*.

Because *SFFS* is a middle layer between user and *SIONlib*, it was possible to introduce some design concepts and provide an API that make the new library more general and reusable. The initial idea was to provide a simple, easy to use extension to *SIONlib* that would cope with as many potential disadvantages as possible allowing it to be used as a real general purpose I/O replacement for parallel applications. Ideally the middle layer should negligible overhead.

The introduction of a new user API led to the idea to completely separate the user from the real back-end I/O calls. This gave me a lot of freedom in designing the user's interface. Also a generic back-end I/O concept was provided, making it easier to implement different back-end systems for handling the combined user files. In general *SFFS*'s conceptual credo could be stated as: "Keep it easy to use, extensible and fast."

2.2 SFFS Design Concepts

The two main concepts are the back-end *Input/Output Interface* and a design unit called FileSystem model.

The latter consists of two entangled classes *File* and *FileSystem*. They are separate because one *FileSystem* object handles multiple *File* instances. Their implementation will depend on each other so much that they cannot be treated separately. This is due to performance reasons.

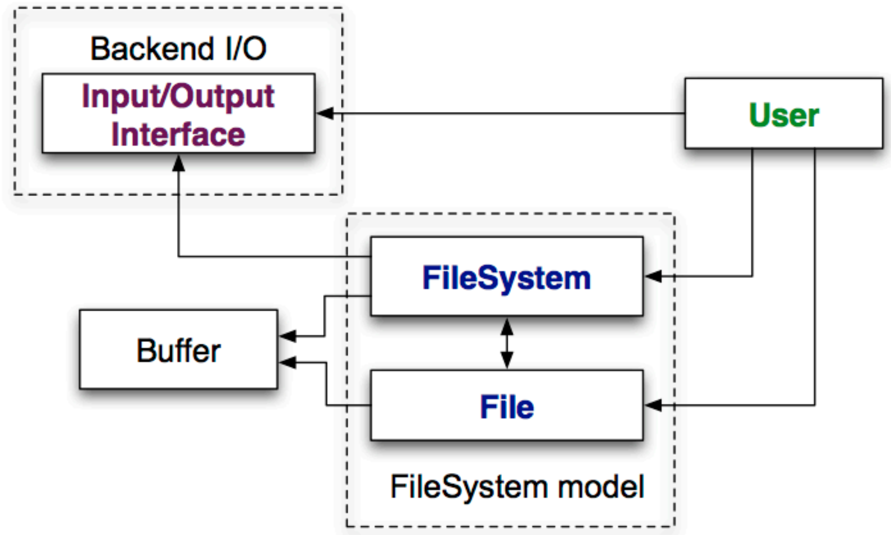


Figure 2: The figure shows a usage diagram in *SFFS*. The arrows point from the object that is a user to the object being used. The green "user" represents the end-user of the *SFFS* library. The rest are the classes *SFFS* consists of.

2.2.1 FileSystem

An instance of this class will handle the virtual file system and provides methods for manipulating *File* objects. *FileSystem* is also attached to a back-end *Input/Output Interface* responsible for communicating with the data sink/source. The most important methods are:

- `Constructor` - to create an object
- `isOpen()` - to check if the object is accessible
- `closeFS` - [Output mode] to manually close the *FileSystem*, after this call system is closed, fully written to the back-end *Output Interface* and might be safely destroyed
- `attachInterface` - [Output mode] to connect the object to a back-end *Output Interface* after the creation of the object

- `isInterfaceAttached()` - [Output mode] to check if the *FileSystem* is connected to an *Output Interface*
- `createFile()` - [Output mode] to create a *File* for writing
- `openFile()` - [Intput mode] to open a *File* for reading

It should be noted that there might be cases when a *FileSystem* itself is opened and fully accessible to the user without an OutputInterface attached. This means that one can create and write to the logical files before choosing and specifying how the binary stream will be written. This might be realized when a FileSystem model uses internal buffering. The functionality, even though provided, should be avoided or used with extreme care. The *FileSystem* eventually will need to write its data to the sink and the interface for that should be accessible, otherwise fatal or undefined behaviour is inevitable.

This functionality was implemented in the standard approach to allow creation of the objects in the global space or when the back-end *Input/Output Interface* could not be initialized beforehand. It is particularly useful when one would like to redirect the standard streams to logical files or write some small debug data before setting up the whole output system. For example *SIONlib* needs `MPI_Init()` to be called before opening a file in the parallel mode, so no writes would be possible before this point without this feature.

2.2.2 File

This concept represents a file object, which the user will use most often. Its purpose is to provide data manipulation functions that resemble the behaviour of the standard *C++* `fstream` binary mode. The most important methods exposed to the user are:

- `isOpen()` - to check if the object is accessible
- `write()` - [Output mode] to write a binary stream of data in Output mode
- `read()` - [Input mode] to read a binary stream of data
- `seek()` - [Input mode] to set the reader position in *File* at specified byte
- `tell()` - [Input mode] to get the reader position
- `eof()` - [Input mode] to check if the end of the file is reached
- `getSize()` - [Input mode] to get data size stored in the opened *File*

It should be noted that the user is not meant to explicitly create instances of this object. A logical file should always exist within a *FileSystem* context. For any *File* object manipulation one should call appropriate *FileSystem* method, especially for those which involves creation of new *File* instances.

2.2.3 Input/Output Interface

The purpose of *InputInterface* and *OutputInterface* is to provide a standard concept and methods of handling the binary stream that a FileSystem model implementation will need. Implementations of the interfaces will define how the *FileSystem* data will be really split and communicated to the back-end I/O device. The interfaces implement methods that allow *FileSystem* and *File* classes to access the data as a contiguous binary stream. This is a very convenient approach, that eases the design and the process of implementing a FileSystem model.

A particular example may be *SIONlib*, where a process' data is split amongst chunks. *SIONlib* at the time of creation of *SFFS* did not provide a function to map a logical position in the process' stream of data to the chunk number and the position in the chunk in a *SION* file. Those values are needed for the `sion_seek()` function, thus for the purpose of random access of the saved data. In case of *SIONlib InputInterface* has to implement the functionality. If the back-end I/O system already provides the necessary functions an *InputInterface* implementation needs just to wrap them. An example of such simple situation would be the standard *C* library. Its `seek()` function behaves exactly as required.

From the user's point of view the *Input/Output Interface* cannot be called really generic. It is for a FileSystem model. Even though this might contradict one of the main goals - a complete separation of the user from the back-end I/O, one does not need to know any internals. The only thing the user has to do is to construct the object and sometimes close the interface. Moving the responsibility of the object creation to the user allowed more freedom in specifying working parameters, such as filename and other back-end specific settings. It would be very hard to provide a standard constructors and parameters that would not limit the user. It was decided that it is better to make the user explicitly call one or two more functions for the sake of versatility.

The most important method is the `Constructor`.Through it one can specify where and how the binary stream will be written or read. If the implementation needs it, the *OutputInteface* can be closed before the object destruction by calling the `close` method. This is the case, for example, with *SIONlib* where a *SION* file has to be closed before calling `MPI::Close`.

3 General Achievements and Failures

3.1 Accomplished Goals

- custom file creation; *File* objects can be created at any place in the code by calling appropriate *FileSystem* method
- custom write pattern; the user does not need to care about the optimal data output nor split it manually as this is moved onto FileSystem model
- ease of use; *File* methods closely resemble those used in *C++* `fstream` binary mode; usually the only change needed is just the type of the object, method names and parameters are designed to stay compliant
- custom and seamless reading with `seek` support; each task local file can be treated as a separate and contiguous binary data stream as a standard `fstream` object would be
- almost complete user separation from the back-end I/O
- possibility to use any back-end I/O system as long as it properly implements Input/Output Interface
- possibility to implement different FileSystem models if needed

3.2 Unaccomplished Goals

- random write; no method that would resemble `seek` in Output mode

- complete user separation from the back-end I/O; the user still needs to create and sometimes close the object
- support for mixed Input and Output operations; *FileSystem* and *File* objects can be used for either Input or Output mode at a time

4 Implementation of a FileSystem model - Shared-Buffer

4.1 Motivation

The first FileSystem model implemented assumed that the only ProFASi problems were large amount of small writes and combining files into one binary stream. Files where considered to be large enough to fill the underlying file system base block. *File* object had its own buffer of the size of a basic block size. Though during initial consultations it turned out that in general not even some files but whole process might have trouble filling up the specified space. This made the model impractical and unacceptable in some situations.

4.2 Solution Outline

To deal with this problem the Shared-Buffer FileSystem model was designed and implemented. In this model the buffer is owned by the *FileSystem* and shared among the *File* objects. This made all the data that should be written with one call to the back-end *OutputInterface* kept at one contiguous memory block. Each *File* instance is given a virtual buffer which is in fact a piece of *FileSystem*'s buffer.

4.3 Data Layout

A binary stream containing user files consists of a file blocks part, a file descriptors table and a number of files stored in this particular order. This is because the model has to conform to the *SIONlib* limitation that writing is serial. The data written physically to the disk have to be known at the point of the write call and no further update is possible. The complete file system structure is known when it is closed. Especially the number of files, thus the number of file descriptors needed. To handle arbitrary number of files the implementation stores the file descriptors in memory and writes it at the end of the *SFFS* stream followed by the number of files stored.

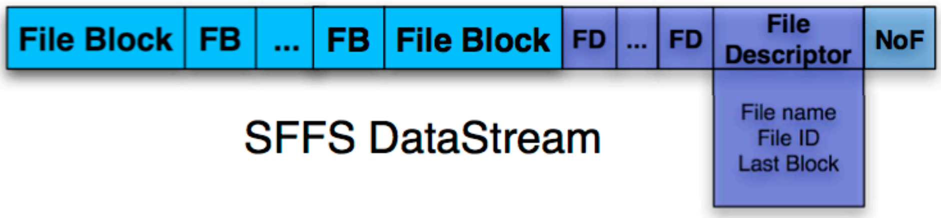


Figure 3: The figure depicts the structure of a SFFS file system's binary stream. NoF stands for "Number of Files". A group of file blocks is sent to a back-end I/O interface whenever they fill the shared buffer. File descriptors and the number of the files are written upon closing of the *FileSystem* object.

A *File descriptor* is a structure holding the name of the file, its ID number and the address of the file's last block in the *SFFS* stream.

Each *File block* consists of a header and a data space. The header is placed at the beginning of the chunk and the data space immediately after. The header contains the *File ID*, the chunk size, the data size and the address of the previous block with the same *File ID*. The *File ID* allows the data in chunk to be assigned to the appropriate file when reading the *SFFS* file. The chunk size is used to easily locate the next chunk in the *SFFS* file. The data size indicates how much of the data space in the chunk contains the relevant user file data. This field is needed as sometimes the data space might not have been used completely. The rest would contain some random information that was in the buffer at the flush point. This should be discarded while reading the data. The previous block address helps to navigate fast through the user file.

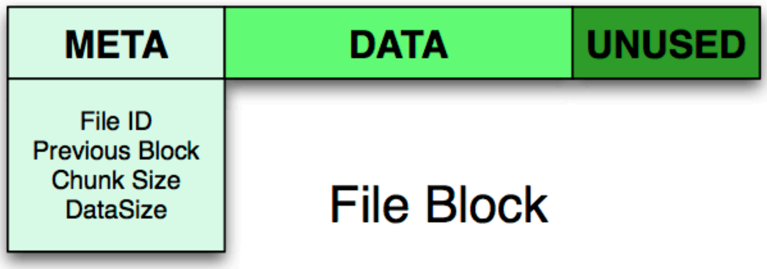


Figure 4: An illustration of data layout within a file block structure.

4.4 Working Internals

The model is designed to keep memory operations at minimum while allowing effective use of *SIONlib* by performing write calls with data size exactly matching the specified chunk size. *File* object's virtual buffer is the exact data that will be written to the *OutputInterface*. The shared buffer contains a number of different *File blocks* that will be flushed in one write call. Its binary structure exactly represents what will be sent to the data sink.

The only draw back is a possibility of wasting some data space. If a *File* does not manage to fill its buffer space before the flush, then nothing can be done with the unused space in *File*'s virtual buffer. Though the amount of space lost due to this effect is usually very low.

Whenever a *File* object fills its buffer, it asks the owning *FileSystem* for another piece providing the size it would prefer to get. The *FileSystem* may respect this but does not need to. *File* implementation should be able to work with any buffer size granted. This way it is possible to implement some sophisticated strategies of assigning buffer chunks to optimize lost space and number of requests made. When a *FileSystem* has no more buffer left to assign, it means that the whole shared buffer is full. Then the *FileSystem* requests all of the owned *File* objects to update their header data and flushes the shared buffer through the assigned *OutputInterface*.

5 Real Usage

5.1 Current Implementation Status

Currently implemented and working features include parallel version of *SIONlib* as both Input and Output Interfaces, so the files can be written and then read by parallel applications. The same status applies to standard Input/Output Interfaces. By standard I mean those that use *C/C++/posix* routines as back-end I/O operations. In a such case one physical file per process is created and it contains a binary stream of the whole *SFFS FileSystem*.

The only FileSystem model implemented for now is *Shared-Buffer*.

There is a number of applications originally intended for testing purposes. They were adapted to be used as external tools for tasks like splitting the whole file system from a *SFFS* file, listing files saved in a *SFFS* file or extracting just a particular one. These are provided for *SFFS* files that were written using standard OuputInterfaces.

5.2 SFFS Integration

With such designed and working library integrating *SFFS* with parallel version of *SIONlib* as back-end into ProFASi was a fairly easy task. The whole process was split in two phases due to ProFASi's I/O operations structure. Files that wrote text data used a dedicated class that was responsible for write calls. Therefore simply by switching functions of creation and writing to the files from standard ones to *SFFS* counterparts took care of all text files from whole application.

Second phase was to redirect writing of the binary files. This part was more inconvenient because the file creations and writings were spread among different source files. Nevertheless it was sufficient just to find every occurrence of file creation (`fopen`) and switch it to `OFileSystem::createFile()`. Then every `fwrite` to `OFile::write`. It was also necessary to overload functions which took *FILE** pointer as an argument to accept *OFile&* reference. The whole phase could have been automated using regular expressions.

6 Performance

6.1 JuRoPA

JuRoPA is a cluster consisting of 3288 computing nodes, each with 2 Intel Xeon X5570, which gives in total 26304 computing cores [4]. The filesystem used to support JuRoPA is Sun's Lustre File System [5, 6].

Tests run on this machine used 64 up to 2048 processes with a typical ProFASi settings file. The purpose of the tests was to check if there is a reduction of the total runtime in typical run conditions. The runs were performed using both the original unmodified version and the one enhanced with *SFFS/SIONlib* for I/O handling. The settings were exactly the same as well as the results. The only difference was the data output method.

The modified version gave a performance gain up to 10% over the original one. However the performance gain scales with number of processes although at diminishing rate. One should keep in mind that these plots do not show if the time was saved by reducing file creation time or during the run due to buffering and better usage of parallel I/O filesystem. Nevertheless it has been clearly shown that implementing *SFFS/SIONlib* improved the over-all runtime in general conditions.

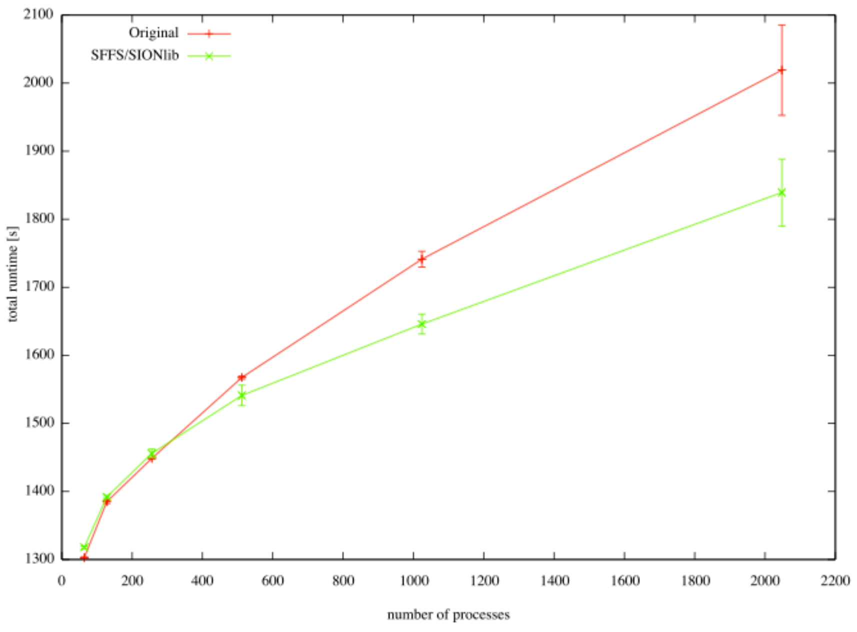


Figure 5: The figure presents a plot of total runtime of original ProFASi version and enhanced against number of processes. A run using twice the number of processes produces twice the amount of data.

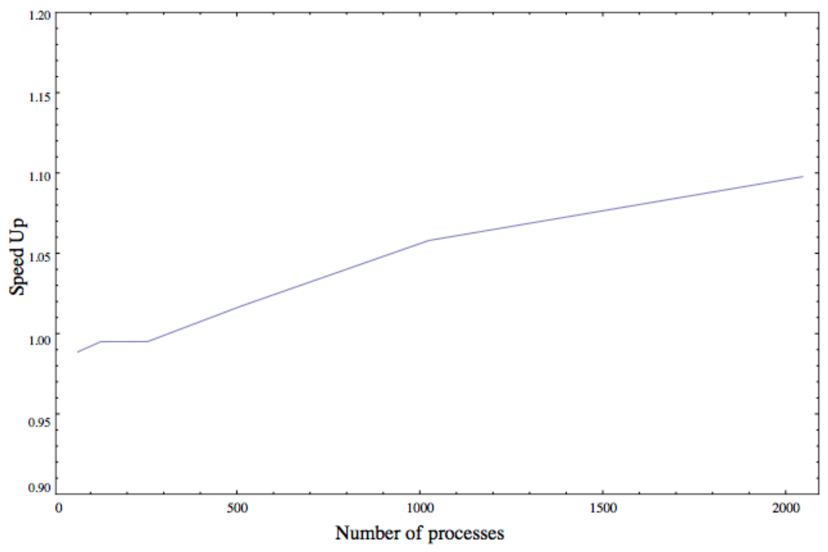


Figure 6: The figure presents speed gained by using modified version. "Speed Up" in this plot is the ratio of the total runtime of the version which utilized *SFFS/SIONlib* and the original one which used standard *C* functions for I/O.

6.2 Nicole

Nicole is a small cluster where tasks up to 256 processes are launched. The filesystem supporting Nicole is NFS.

A number of tests were run with different I/O operations load to draw more conclusions about performance gains. A second settings file was prepared to make ProFASi output unusually large amounts of data and additional time checks were included to measure the file creation time.

Performed tests have shown that there is almost no difference in file creation nor run time between original and modified version on instances under 128 processors. When using 256 processor file creation time increased from about 1 second up to 3 while modified version consistently kept under 1 second. The difference in this part of the application started to be visible although it is very small compared to the total run time of about 5 minutes.

The real difference could be observed with heavy I/O settings used. File creation times stayed at the same levels. The total time went up to 22 minutes for original version and 17 for modified. This behaviour was stable and reproducible. This is a difference of about 20% gained almost completely in the runtime. The results suggest that modified version scales better not only with number of tasks but I/O operations load as well.

6.3 JUGENE

JUGENE [7] is an IBM BlueGene/P with 73728 computing nodes. Each computing node has four PowerPC 450 processor. This gives 294912 processors for the whole system. The supporting filesystem is IBM's General Parallel File System [8].

The tests were designed to measure differences in the file creation part as well as total runtime. The number of processes used ranged from 512 to 8192.

Although not all tests with the original version have been run, performed tests clearly show that file creation times scales with the number of processes going from 24 seconds to 40 and over 170 for 512, 1024 and 2048 processes respectively. Modified version took 2.2 seconds for 512 processes and went up to 3.8 seconds with 8192. All total runtime differences were equal to file creation time variances.

The results show a clear advantage of the modified version in the file creation time, although it is a difference that does not scale with runtime and amount of I/O operations application makes. Further tests with more application data output should to be performed to check if the behaviour observed on Nicole also applies to this system.

7 Summary

7.1 Conclusions

The main goal of this project can be regarded as achieved. The combination of *SFFS/SIONlib* has indeed shown to be a good replacement to standard file handling functions. *SIONlib* allows to take advantage of parallel filesystems with almost no effort. *SFFS* makes it even more user friendly extending *SIONlib*'s original functionality and removing some responsibilities from the user.

The integration process of the proposed solution into ProFASi was fairly easy to perform. Most changes could be done almost automatically. The performance compared to the original version was also improved. Although gains in most typical conditions were not critical they were considered good enough. Also the tests performed strongly suggest that the speed gain scales with both problem size and the amount of output operations. Another clear advantage of this approach coming directly from *SIONlib* is only one file is created for the whole run. This removes filesystem load caused by the file creation and also makes it possible to fit into quota limits. One can easily imagine that on a machine such as JUGENE the number of the files can go into millions. The file number quota limit for the group my account was assigned to was 2 million. That means reducing whole output into a single file allows easier management of runs by abolishing the necessity of file removal in-between. Moreover in certain, really large scale cases *SFFS*/*SIONlib* makes the simulations possible to be performed.

7.2 Future Work

To evaluate pure *SFFS* quality synthetic tests which would measure performance of pure *SIONlib* and a version wrapped with *SFFS*. In fact this comparison is very interesting and important as it would explicitly show how much of overhead does *SFFS* produce, though unfortunately this was not completed due to time shortage.

Another thing of interest would be implementing different parallel back-end I/O Interfaces especially ones which would allow random writes.

8 Acknowledgements

I would like to sincerely thank Jan Meinke and Sandipan Mohanty, the developers of ProFASi, for being always available for consultation and providing me with much help during my stay at Juelich; also for the amount of trust they have put into my person and ideas letting me implement and realize my own visions. I would also like to express my gratitude to Wolfgang Frings, the developer of SIONlib, for helping me resolving issues and implementing new features for sake of my work despite being on vacation time. I would like to thank Professor Grzegorz Kamieniarz for giving me the opportunity to apply for the program.

Great amount of thanks goes to my fellow guest students of the programme, especially: Jonathan Gross, Stilianos Louca, Ricardo Kennedy, Malik Kirchner and Martin Rueckl for helping me in numerous every-day life situations in Germany, being always available for discussions and of course for the meetings. Finally I would like to express my gratefulness and honour to Robert Speck, who had to put in an extraordinary amount of work to make it possible for me to participate in the programme. Thank you.

References

1. Frings W., Wolf F., and Petkov V., "Scalable massively parallel I/O to task-local files." Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (Portland, Oregon, USA. 14 - 20.11.2009). SC '09. ACM, New York, NY.
2. Irbäck A., Mohanty S., "ProFASi: A Monte Carlo simulation package for protein folding and aggregation.", J. Comp. Chem. 27, p. 1548 (2006).
3. Petkov V., Jülich Supercomputing Centre Guest Student Program 2008. FZJ-JSC-IB-2008-07, p. 93 (2008). Web. 03.12.2009. URL: http://www.fz-juelich.de/jsc/files/docs/ib/ib-08/ib-2008-07.pdf
4. JuRoPA System Configuration. Jülich Supercomputing Centre. 23.06.2009. Web. 03.12.2009. URL: http://www.fz-juelich.de/jsc/juropa/configuration/
5. Lustre File System. Sun microsystems. n.d. Web. 03.12.2009. URL: http://www.sun.com/software/products/lustre/
6. Schwan P., "Lustre: Building a File System for 1,000-node Clusters." Proceedings of the Linux Symposium. Ottawa, Ontario, Canada. 23 - 26.06.2003. p. 380. Web.03.12.2009. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.1062&rep=rep1&type=pdf#page=380
7. IBM Blue Gene/P JUGENE. Jülich Supercomputing Centre. 23.06.2009. Web. 03.12.2009. URL: http://www.fz-juelich.de/jsc/service/sco_ibmBGP
8. IBM General Parallel File System. IBM. n.d. Web. 03.12.2009. URL: http://www-03.ibm.com/systems/clusters/software/gpfs/index.html

Simulation of Drift-Kinetic Dynamics of Charged Particles in Magnetized Plasmas

Stilianos Louca

Friedrich Schiller Universität, Jena

E-mail: stilianos.louca@uni-jena.de

Abstract:

In particle codes for fusion plasma simulations, an upper bound for the integration time-step is set by the high frequency gyro-motions ($\sim$ THz) of electrons. A method for replacing the exact electron positions by their guiding centers is discussed and tested on single particle simulations. It is shown, that for typical electromagnetic fields found in fusion experiments as the TOKAMAK, the time-step can be increased by a factor of up to 100, while still maintaining deviations of the guiding centers in the order of 10^{-2} gyro-radii. Moreover, a divergence-free interpolation method is introduced for cylinder-symmetric magnetic fields defined on triangular meshes, which results in an almost everywhere smooth field. The interpolation is reduced to 1-dimensional interpolations and can include an arbitrary number of base-points.

1 Introduction

Particle codes simulating the behavior of plasmas all share a common concept: Particle positions and velocities are integrated sequentially by a time-step sufficiently small, using the forces, or electromagnetic fields for that matter, calculated as acting on the particles [1]. This is in particular true for the PEPC tree code, used among others, to simulate the behavior of fusion plasmas in TOKAMAK experiments [7, 9].

Charged particle motion in plasmas is classically governed by the so called Lorentz-Force[1]

$$\ddot{\mathbf{r}} = \frac{q}{m}\mathbf{E} + \frac{q}{m}\dot{\mathbf{r}} \times \mathbf{B} \tag{1}$$

with q, m and $\mathbf{r}$ being the charge, mass and position of the given particle respectively. Throughout this article, $\mathbf{E}$ and $\mathbf{B}$ will signify the electrostatic field and magnetic flux density respectively. As is known [2], the presence of a magnetic field, forces charged particles on spiral orbits about the magnetic field lines with local frequency $\omega_{\mathrm{g}} = qB/m$. This motion can be interpreted as a circular motion (*gyration*) around a so called *gyration center* superimposed on a drag **approximately** along magnetic field lines.

In fusion plasmas, where magnetic field values are of the order ~ 1 T, electron gyro-periods T_{g} can reach down to a few ps. This sets an upper limit for the integration time-step δt used in conventional integrators like the Boris Solver [1], which requires time-steps $\delta t \lesssim T_{\mathrm{g}}/10$ in order to lead to reasonable particle orbits [5].

Different methods for increasing the possible time-step have been suggested [6], which usually attempt to deliver the *actual* particle orbits. The solution presented in this report, is based on the idea of replacing the exact electron position with its gyration center, alias *guiding center*. By dropping any detailed information on the exact particle gyro-motion, the following assumptions have to be made:

[1]Gravitational forces shall be neglected in the following context, as they can be assumed to be included in $\mathbf{E}$.

- Electromagnetic field values vary only weakly within a gyro-period and gyro-radius.
- Particle interactions take place at scales greater than the typical gyro-radius.
- Radiation effects due to electron accelerations are neglectable.

This report begins with some insight into guiding center motions and their driftings in magnetized plasmas. The implementation of a guiding center integration scheme into PEPC is outlined and accuracy test results for the calculated guiding center orbits presented. Furthermore, an interpolation scheme for cylinder-symmetric, divergence-free magnetic fields on triangular meshes is presented. The necessity for the latest arose from the need of the magnetic field-gradient for the guiding center integrations.

2 The Guiding Center Motion

2.1 Defining the Guiding Center

Abstractly, the guiding center should resemble some sort of geometrical center of the gyration motion. Formally, we use

$$\mathbf{R} := \mathbf{R}(\mathbf{r}, \dot{\mathbf{r}}) := \mathbf{r} - \underbrace{\frac{1}{\omega_{\mathrm{g}}}\mathbf{b} \times \left(\dot{\mathbf{r}} - \frac{\mathbf{E} \times \mathbf{B}}{B^2}\right)}_{\boldsymbol{\rho}} \tag{2}$$

as an exact definition as provided by Northrop in [4]. Here, $\mathbf{r}$ is the exact particle and $\mathbf{R}$ the guiding center position respectively, ω_{g} the local gyration frequency and $\mathbf{b} := \mathbf{B}/B$. In the simple case of homogenic $\mathbf{B}$ and $\mathbf{E}$-fields this definition incites with our geometrical expectations.

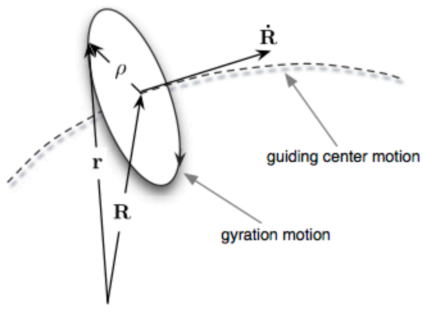


Figure 1: On the definition of the guiding center.

Note that the introduction of a *gyration center* assumes the presence of a $\mathbf{B}$-field sufficiently far from 0.

2.2 The Guiding Center Equation of Motion

In order to directly calculate the orbit of a guiding center, the need for a differential equation describing its motion naturally arises. One such differential equation is provided by Northrop [4], expanded with respect to the parameter

$$\varepsilon := \rho_{\mathrm{g}} \cdot \max\left\{\frac{\|\nabla\mathbf{B}\|}{B}, \frac{\|\nabla\mathbf{E}\|}{E}\right\} , \tag{3}$$

basically expressing the ratio between the initial gyration radius and the characteristic field variation-scales[2]. Expanding the electromagnetic fields to first order in ε and using $\nabla \cdot \mathbf{B} = 0$, one obtains the equation of motion in zero order accuracy

$$\boxed{\ddot{\mathbf{R}} = \frac{q}{m}\left[\mathbf{E} - \frac{\mu}{q}\nabla B\right] + \frac{q}{m}\dot{\mathbf{R}} \times \mathbf{B} + \mathcal{O}(\varepsilon)} \tag{4}$$

with

$$\mu := \frac{m}{2}\frac{\dot{\mathbf{r}}_\perp^2}{B} \tag{5}$$

as the *magnetic moment* of the particle circulating around its gyration center and $\dot{\mathbf{r}}_\perp$ as the exact particle velocity perpendicular to $\mathbf{B}$. Furthermore, it can be easily shown [4] that $\dot{\mathbf{R}}$ is to zero order given by

$$\dot{\mathbf{R}} = \dot{\mathbf{r}}_\parallel + \underbrace{\frac{\mathbf{E} \times \mathbf{B}}{B^2}}_{\mathbf{v}_E} + \mathcal{O}(\varepsilon) \tag{6}$$

with $\dot{\mathbf{r}}_\parallel$ as the particle velocity parallel to $\mathbf{B}$.

Note: Equation (4), describing the movement of the guiding center in zero order, actually resembles the EOM of the particle, within a modified electrical field $\tilde{\mathbf{E}} := \mathbf{E} - \frac{\mu}{q}\nabla B$. Omitting higher order terms, leads to an orbit consisting of the usual, highly-frequent gyrations superimposed on a central movement approximately along the B-field lines. In fusion plasmas, the initial, perpendicular speed component $\mathbf{E} \times \mathbf{B}/B^2$ is typically orders of magnitudes smaller than the actual particle velocity, leading to a significantly smaller *gyration* radius. This allows for an increase of the time-step usable for the integration of this motion (see section 4.1) [5].

2.3 Guiding Center Drifts

Crossing EOM (4) with $\mathbf{b}$ gives an insight into the guiding center velocity perpendicular to $\mathbf{B}$. One obtains [4]:

$$\dot{\mathbf{R}}_\perp := \dot{\mathbf{R}} - \underbrace{\langle \mathbf{b}, \dot{\mathbf{R}} \rangle \cdot \mathbf{b}}_{\dot{\mathbf{R}}_\parallel} = \underbrace{\frac{\mathbf{E} \times \mathbf{B}}{B^2}}_{\mathbf{v}_E} + \frac{\mu}{qB}\mathbf{b} \times \nabla B + \frac{m}{qB}\mathbf{b} \times \ddot{\mathbf{R}} + \mathcal{O}(\varepsilon^2) \ . \tag{7}$$

The first term is the known $\mathbf{E}\times\mathbf{B}$-*drift*, caused by a variation of the particle speed during its gyro-motion.

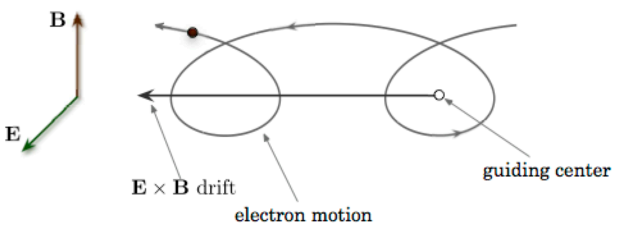


Figure 2: On the $\mathbf{E} \times \mathbf{B}$-drift of the guiding center.

[2] Assumed to be sufficiently small.

The second term, known as *magnetic-* or *gradient-B-drift*, results from a variation of the particle's gyro-radius during its gyro-motion. The last one is the so called *acceleration drift* [4] Expanding it to

$$\frac{m}{qB}\mathbf{b}\times\ddot{\mathbf{R}} \overset{(6)}{=} \frac{m}{qB}\mathbf{b}\times\left[\frac{d}{dt}\left(\dot{\mathbf{R}}_\parallel + \mathbf{v}_E + \mathcal{O}(\varepsilon)\right)\right] \tag{8}$$

$$= \underbrace{\frac{m}{qB}\mathbf{b}\times\frac{d\mathbf{v}_E}{dt}}_{\mathbf{v}_\mathrm{p}} + \underbrace{\frac{m}{qB}(\dot{\mathbf{R}}\mathbf{b})\cdot\mathbf{b}\times\frac{d\mathbf{b}}{dt}}_{\mathbf{v}_\mathrm{i}} + \mathcal{O}(\varepsilon) \tag{9}$$

yields the so called *polarization drift* $\mathbf{v}_\mathrm{p}$ and *inertial drift* $\mathbf{v}_\mathrm{i}$, which in turn can be expanded to

$$\mathbf{v}_\mathrm{i} = \frac{m}{qB}(\dot{\mathbf{R}}\mathbf{b})\cdot\mathbf{b}\times\left[\frac{\partial\mathbf{b}}{\partial t} + (\nabla\mathbf{b})\mathbf{v}_E\right] + \frac{m}{qB}(\dot{\mathbf{R}}\mathbf{b})^2\cdot\mathbf{b}\times[(\nabla\mathbf{b})\mathbf{b}] + \mathcal{O}(\varepsilon) \tag{10}$$

with the last term being the known *curvature drift* [3].[3]

2.4 Average Kinetic Energy

Equation (4) can not be solved in a closed way, since it requires exact knowledge of the magnetic moment $\mu = \mu(\dot{\mathbf{r}}_\perp, B)$. But this in turn would, so it seems, require knowledge of $\dot{\mathbf{r}}_\perp$, whose elimination was one of the purposes for this theory in the first place.
It can be shown, that the *average kinetic energy* $\mathcal{E}_\mathrm{kin}$ over one gyration period is in first order given by

$$\mathcal{E}_\mathrm{kin} = \frac{m}{2}\dot{\mathbf{R}}_\parallel^2 + \frac{m}{2}\mathbf{v}_E^2 + \mu B + \mathcal{O}(\epsilon^2) \tag{11}$$

fulfilling the differential equation

$$\frac{d}{dt}\mathcal{E}_\mathrm{kin} = q\langle\dot{\mathbf{R}},\mathbf{E}\rangle + \mu\frac{\partial B}{\partial t} + \mathcal{O}(\epsilon^2), \tag{12}$$

where all fields are evaluated at $\mathbf{R}$. The term $q\langle\dot{\mathbf{R}},\mathbf{E}\rangle$ results from the work performed by the electric field on the guiding center, while the term $\mu\frac{\partial B}{\partial t}$ represents the action of the curl of $\mathbf{E}$ on the gyrating particle, resulting from the induction due to a time-depended magnetic field[4]. Together with eq. (4), this equation allows for an integration of the EOM of the guiding center, at least to zero order [4].

3 Interpolating Divergence-Free Fields

The need for a meaningful $\mathbf{B}$- and $\nabla\mathbf{B}$-field for integrating the guiding-center motion of particles, naturally leads to the necessity of interpolating or fitting the externally provided $\mathbf{B}$-field data (see section 4.2) over a given set of point-value pairs $(\mathbf{x}_i, \mathbf{B}_i)$, while securing the fundamental differential equation $\nabla\cdot\mathbf{B} = 0$. The following section proposes some interpolation methods developed within the context of this project.

The method of choice turned out to be the so called *component-separated* interpolation scheme, mainly due to its simplicity and low field variation within the convex hull of the interpolation base-points.

[3]Note that $(\nabla\mathbf{b})\mathbf{b}$ is the vector pointing towards the curvature center of the magnetic field-line with the inverse curvature radius as norm.

[4]Note that the magnetic field its self does not perform any actual work on the particle, which is why the term $(\nabla\mathbf{B})\dot{\mathbf{R}}$ is not included.

3.1 2D-Case

Let $\mathbf{x}_1, .., \mathbf{x}_N \in \mathbb{R}^2$ be pairwise different base-points and $\mathbf{F}_1, .., \mathbf{F}_N \in \mathbb{R}^2$ arbitrary base-field-values. Sought is a field $\mathbf{F} : \mathcal{C}^1(\mathbb{R}^2 \to \mathbb{R}^2)$ such that:

1. $\nabla \cdot \mathbf{F} = 0$,
2. $\mathbf{F}(\mathbf{x}_i) = \mathbf{F}_i, \;\; i = 1, .., N$.

3.1.1 Component Separated Interpolation

As can be easily seen, any vector field of the form

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} F^1(x^2) \\ F^2(x^1) \end{pmatrix} \tag{13}$$

is by construction divergence-free. Naturally, one is tempted to interpolate the 1st component F^1 using only the 2nd coordinates and the 2nd component F^2 using only the 1st coordinates of the base-points $\mathbf{x}_1, .., \mathbf{x}_N$. A prerequisite for this, is that all base-points differ in their 1st as well as 2nd coordinate. Obviously, this is always true in some cartesian coordinate system.

More precisely, given base-points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^2$ and base-values $\mathbf{F}_1, .., \mathbf{F}_N \in \mathbb{R}^2$, the field $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$ defined by

$$\mathbf{F}(\mathbf{x}) := \begin{pmatrix} f(x^2;\; x^2_1, .., x^2_N;\; F^1_1, .., F^1_N) \\ \\ f(x^1;\; x^1_1, .., x^1_N;\; F^2_1, .., F^2_N) \end{pmatrix} \tag{14}$$

with $f(\cdot\,;\, t_1, .., t_N;\;\, f_1, .., f_N)$ being some arbitrary interpolation scheme over the point-value pairs $\{(t_i, f_i)\}_{i=1}^N \subseteq \mathbb{R}^2$, actually satisfies both conditions 3.1 (1) and (2). Thus, using eq. (14), one is able to reduce the problem of 2D divergence-free interpolation to two 1D interpolations (see fig. 3). Note that f can be really arbitrary and as simple as piecewise-linear.

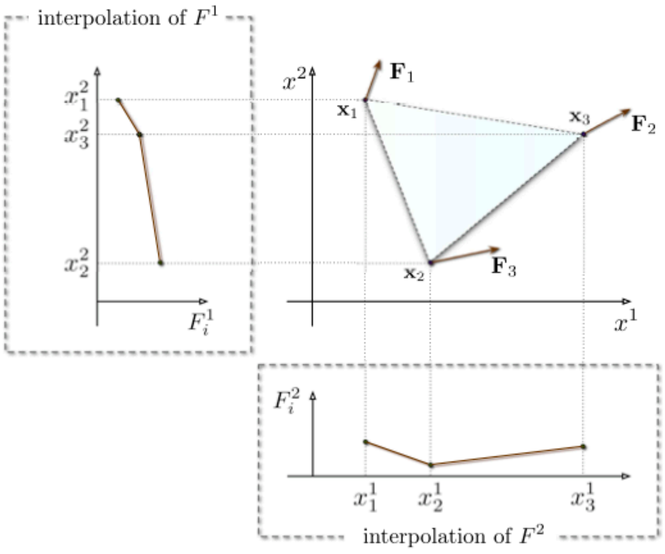


Figure 3: Component-separated interpolation over 3 base-points. 2D divergence-free interpolation is reduced to two 1D interpolations (here: linear).

The field-Jacobian is given in the form

$$\nabla \mathbf{F} = \begin{pmatrix} 0 & \partial_2 f(\cdot\,;\, x_1^2, .., x_N^2;\; F_1^1, .., F_N^1) \\ \partial_1 f(\cdot\,;\, x_1^1, .., x_N^1;\; F_1^2, .., F_N^2) & 0 \end{pmatrix} . \tag{15}$$

Note: In case the interpolation is performed in some other coordinate system, with U being the transformation from original to new coordinates, the field $\mathbf{F}$ will generally not be of the form (13), that is, both components will depend on both coordinates. Nonetheless, as the field in the new coordinates simply takes the form $\tilde{\mathbf{F}}(\mathbf{x}) = U\mathbf{F}(U^{-1}\mathbf{x})$, with

$$\nabla \cdot \mathbf{F} = \operatorname{trace}(\nabla \mathbf{F}) = \operatorname{trace}(U^{-1} \cdot (\nabla \tilde{\mathbf{F}}) \cdot U) = \operatorname{trace}\left(UU^{-1} \cdot \nabla \tilde{\mathbf{F}}\right) = \nabla \cdot \tilde{\mathbf{F}} \tag{16}$$

it is clear that this does not affect the divergence-freeness of $\mathbf{F}$. It should be noted that the resulting interpolated field actually depends on the coordinate system chosen!

3.2 Cylinder-Symmetric, Divergence-Free 3D-Field Interpolation

In view of the magnetic fields used for TOKAMAK plasma simulations, we shall only consider the interpolation of cylinder-symmetric $\mathbf{B}$-fields over base-points defined on the plane $\{\varphi = 0\}$. That is, for given base-points (ρ_i, z_i) and base-values $(B_i^\rho, B_i^z, B_i^\varphi),\ \ i = 1, .., N$, sought is a cylinder-symmetric vector-field $\mathbf{B} : \mathbb{R}^3 \to \mathbb{R}^3$ so that:

1. $\mathbf{B}(\rho_i, z_i) = B_i^\rho \mathbf{e}_\rho + B_i^z \mathbf{e}_z + B_i^\varphi \mathbf{e}_\varphi\ ,\ \ i = 1, .., N,$
2. $\nabla \cdot \mathbf{B} = 0.$

3.2.1 Reduction to 2D-Case

For cylinder-symmetric vector-fields $\mathbf{B} : \mathbb{R}^3 \to \mathbb{R}^3$, condition $\nabla \cdot \mathbf{B} = 0$ reduces to the differential equation

$$\frac{\partial}{\partial \rho}(\rho B^\rho) + \frac{\partial}{\partial z}(\rho B^z) \overset{!}{=} 0\ . \tag{17}$$

In particular, the B^φ-component has no influence on the field's divergence whatsoever. Thus, by setting

1. $\mathbf{x} := (\rho, z)$ and $\mathbf{x}_i := (\rho_i, z_i),\ \ i = 1, .., N,$
2. $\mathbf{F} := \rho \cdot (B^\rho, B^z)$ and $\mathbf{F}_i := \rho_i \cdot (B_i^\rho, B_i^z),$

the interpolation can be reduced to the 2-dimensional case (see section 3.1) for the B^ρ, B^z components, while B^φ may be interpolated in any arbitrary (e.g. linear) way. Note that this interpolation method defines the magnetic field (B^ρ, B^z, B^φ) and its $(\partial_\rho, \partial_z, \partial_\varphi)$-derivative in cylindrical coordinates.

Note: Due to substitution 3.2.1(2), the interpolated field can display divergent behavior for $\rho \to 0$. Even more, one needs to assume that all base-coordinates ρ_i differ from 0. As particle simulations typically take place within the TOKAMAK torus and not its center, this should not pose any problem.

4 Implementation in PEPC

4.1 Guiding Center Integration

As the gyro-periods of the electrons are 3 orders of magnitude smaller than those of the ions, only electrons are replaced by their respective guiding center[5]. In particular, ions are integrated using the original integration scheme.[6]

Start positions and velocities of the electron guiding centers are set at the beginning of the simulation according to eq. (2) and (6), using the originally provided start data, after calculating necessary $\mathbf{E}$- and $\mathbf{B}$-fields at the real electron positions. Similarly, the start magnetic moment and kinetic energy are set using definition (5) and expression (11) respectively.

The similarity of the particle EOM (1) and guiding center EOM (4) allowed the use of the already implemented, leapfrog integration scheme for electron guiding-centers. The velocity of the guiding center is pushed forward using the Boris Solver [1] just as a regular particle, with the only difference lying in the $\mathbf{E}$-field provided, modified by the additional ∇B term.

As the magnetic moment μ is needed for the integration of the guiding center EOM (4), but not explicitly available from $\mathbf{R}$ and $\dot{\mathbf{R}}$, it has to be integrated separately along the simulation. Equivalently, the average kinetic energy $\mathcal{E}_{\text{kin}}$ could be integrated and used to calculate μ (see eq. (11)).

Thus, in view of EOM (4) and (12), the integration step

$$\left(\mathbf{R}^n, \dot{\mathbf{R}}^{n-\frac{1}{2}}\right) \overset{\text{Boris}}{\underset{}{\xrightarrow{\text{Solver}}}} \left(\mathbf{R}^{n+1}, \dot{\mathbf{R}}^{n+\frac{1}{2}}\right) \tag{18}$$

$$\mathcal{E}_{\text{kin}}^{n+\frac{1}{2}} := \mathcal{E}_{\text{kin}}^{n-\frac{1}{2}} + \delta t \cdot q \cdot \left\langle \mathbf{E}^{n+1}, \dot{\mathbf{R}}^n \right\rangle + \delta t \cdot \frac{\mu^{n-\frac{1}{2}}}{q} \cdot \nabla B^{n+1} \tag{19}$$

with time-step δt was used, whereas

$$\dot{\mathbf{R}}^n := \frac{1}{2}\left(\dot{\mathbf{R}}^{n-\frac{1}{2}} + \dot{\mathbf{R}}^{n+\frac{1}{2}}\right) \tag{20}$$

with the $\mathbf{E}$-field evaluated at $\mathbf{R}$ using the PEPC tree code and $\mathbf{B}$ provided externally. The magnetic moment $\mu^{n-\frac{1}{2}}$ is connected explicitly to the kinetic energy by

$$\mu^{n-\frac{1}{2}} := \frac{1}{B}\left[\mathcal{E}_{\text{kin}}^{n-\frac{1}{2}} - \frac{m}{2}\left(\dot{\mathbf{R}}^{n-\frac{1}{2}}\right)^2\right]. \tag{21}$$

As the guiding center drifts (a few cm/s) are typically orders of magnitude smaller that the actual particle velocities (a few $c/10$), the guiding-center gyro-radii are accordingly smaller than the particle gyro-radii. Thus, the time-step δt for the guiding-center integration could be increased noticeably (see test results 4.3) without significantly affecting the accuracy of the calculated orbits [5].

4.2 Magnetic Fields

In contrast to the electrostatic field calculated internally using the PEPC tree code [7], the magnetic field used in the simulation is provided externally on a triangular mesh. Due to the assumed cylindrical

[5]Increasing the time-step even by a factor of $\times 100$, ions would still be integrated using a time-step much smaller than their gyro-period.

[6]Ions and *electrons* are in this context distinguished based on the charge/mass ratio, which essentially defines their gyration frequency in a given $\mathbf{B}$-field.

symmetry of the field, the mesh is defined solely on the $\{\varphi = 0\}$ plane[7], with the field value given in cylindrical coordinates $(B^{\rho}, B^{z}, B^{\varphi})$ on each triangle [8].

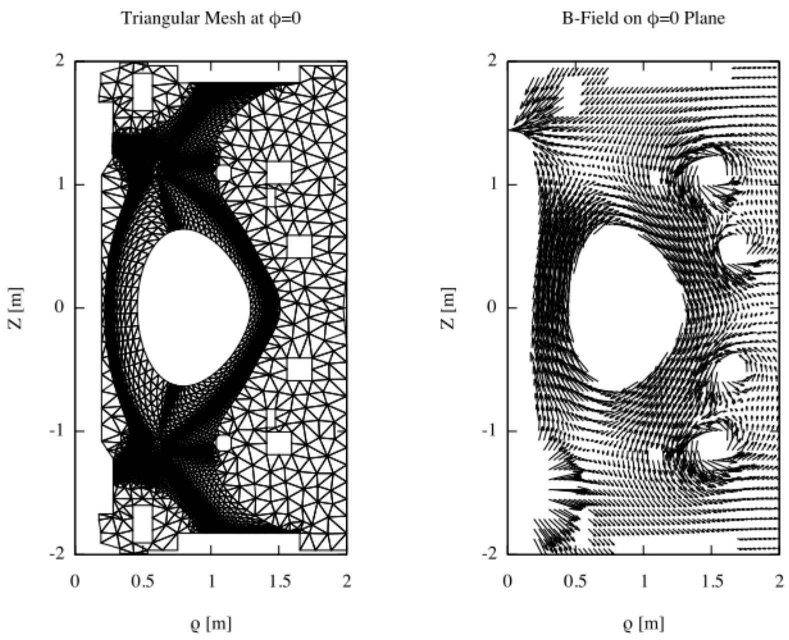


Figure 4: Triangular mesh and projection of $\mathbf{B}$-field on $\{\varphi = 0\}$-plane [8].

4.2.1 Mesh Refinement

The need for a meaningful $\mathbf{B}$-gradient for the integration of the guiding center implied that the $\mathbf{B}$-field values would somehow need to be interpolated or fitted, at least on a local basis. This in turn made the availability of $\mathbf{B}$-field values on a pointwise basis a necessity. Ideally, these values should be defined on the vertices forming the triangular mesh.

In view of these requirements, a new mesh with *new* field-data was constructed from the original mesh and field-data. Using the geometrical center of each triangle and its three vertices, three new triangles were defined, as depicted in fig. 5. The $\mathbf{B}$-field value on the center point was set to be the field-value at the original triangle, while the remaining points were ascribed the arithmetic mean of triangle-field-values of all triangles including them.

[7]Actually, merely using ρ, z coordinates.

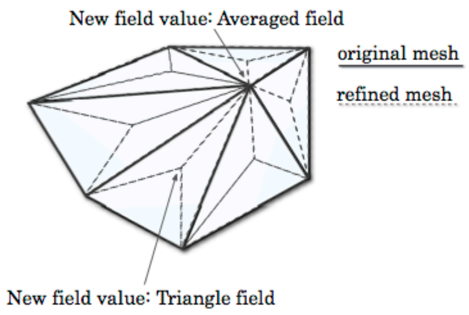

New field value: Averaged field
original mesh
refined mesh
New field value: Triangle field

Figure 5: On the refinement of the mesh triangles. Each original triangle is subdivided into three sub-triangles around its geometrical center.

Loading and evaluation of the new data, is up to the interpolation, performed in PEPC in the same way as originally. In particular, the simulation space[8] is divided into a rectangular $N \times M$ grid, of which each box points to all triangles containing points in that box. This enables a quicker locating of the triangle containing a given point. The magnetic field and its Jacobian for that point is then determined, by interpolating the three vertex field-values of the triangle using component-separated interpolation for (B^ρ, B^z) and simple linear interpolation for B^φ.

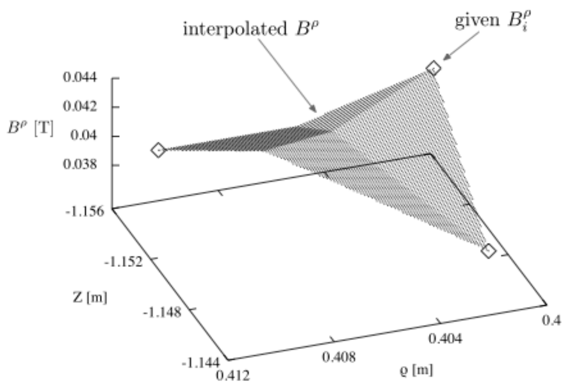


Figure 6: Example interpolation of B^ρ component within a mesh triangle.

4.3 Results

Though the gyro-radius of he guiding center is typically much smaller that the gyro-radius ρ_g of the actual electron, their gyro-periods T_g are actually the same. Thus one might expect, that the need for an accurate calculation of the former, would restrict the time-step δt to similar values as before. Luckily, as was shown by Vu and Brackbill [5], the resulting drifts stay accurate even for time-steps much bigger that the gyro-periods. As the calculated gyro-radius only grows linearly with δt, it was expected that electron guiding centers would remain *sufficiently close* to the real guiding centers of the electron orbits even for $\delta t \gg T_\mathrm{g}$.

The following section presents some typical results of guiding center convergence tests performed on

[8]At $\{\varphi = 0\}$.

PEPC. For clarity purposes, electromagnetic fields were defined externally[9]. As the main subject of these tests was the accuracy of the calculated drifts for increased time-steps, only single-particle tests were performed. *Exact* particle orbits were calculated using the Boris Solver with a time-step $\delta t = T_g/100$.

All units are in SI. In particular, electrostatic field-units are $\mathrm{N \cdot C^{-1}}$, spatial units are m and magnetic field units are T. Particle orbits always start at the origin, with a start velocity of $(0.1, 0, 0.1) \cdot c$. Gyro-radii ρ_g and gyro-periods T_g used for axis-scaling always refer to their values at the origin.

Fig. 7 shows a typical electron orbit in a toroidal (i.e. cylinder-symmetric $\sim \mathbf{e}_\varphi$) magnetic field, together with the calculated guiding center orbit.

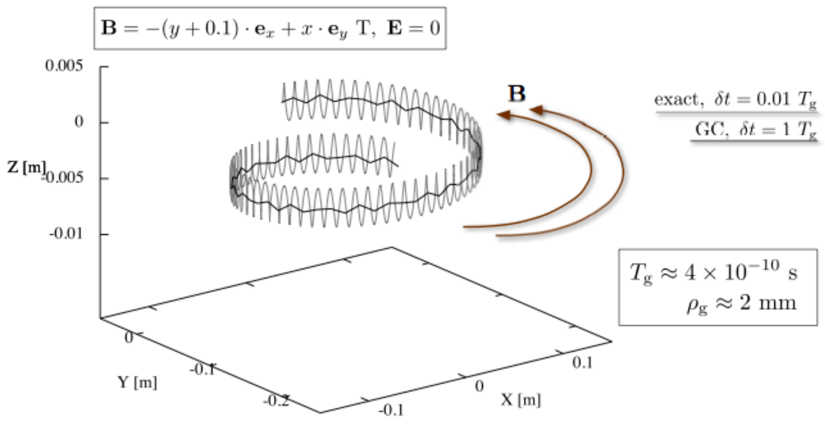


Figure 7: Example simulation of electron orbit and guiding center in toroidal $\mathbf{B}$-field. Start velocity was $\mathbf{v}_0 = (0.1, 0, 0.1) \cdot c$. Notice the vertical curvature drift of the guiding center along the z-axis.

Fig. 8 shows the pointwise distance of calculated guiding centers from the *exact* electron orbit over runtime, in a constant $\mathbf{B}$-gradient and homogenic $\mathbf{E}$-field. Ideally, the guiding center distance to the electron orbit is expected to be constant, namely one gyro-radius ρ_g.[10] As can be seen, the guiding center deviation from this ideal orbit, is in the order of $10^{-3}\rho_g$ for a time-step up to $\delta t = 5T_g$ and $10^{-2}\rho_g$ for a time-step up to $\delta t = 50T_g$!

[9]As internal electrostatic fields are typically of the order 10^{-1} $\mathrm{N \cdot C^{-1}}$, the dominant resulting drifts were indeed (over)represented in the tests performed.

[10]Though ρ_g actually changed along the electron path due to perpendicular drifts, its relative variation within the given runtimes was of the order 10^{-4}.

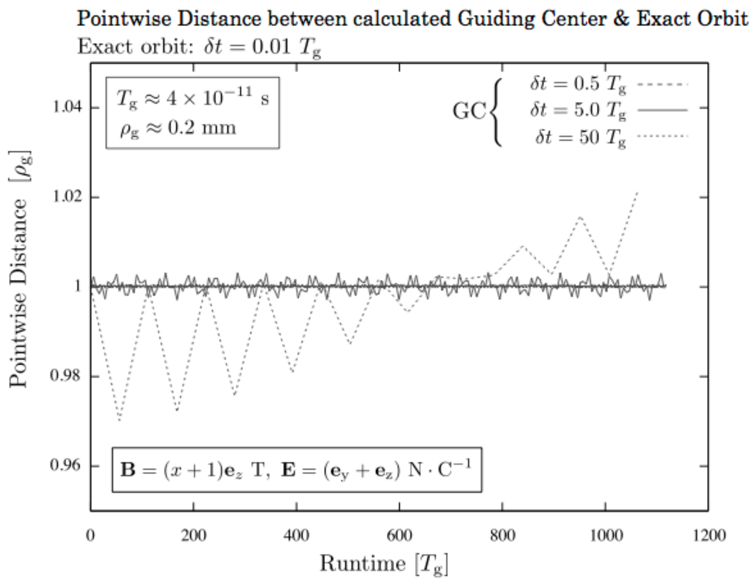


Figure 8: Pointwise distance of calculated guiding-center orbits to *exact* particle orbit over runtime. Start velocity was $\mathbf{v}_0 = (0.1, 0, 0.1) \cdot c$.

Fig. 9 illustrates the spatial *supremum-distance*[11] of calculated guiding center orbits to the exact electron orbits for various time-steps and various field-configurations. Notice the relative stability of calculated guiding center orbits for homogenic fields or constant ∇B even for time-steps $\delta t \sim 100T_g$. In contrast, the toroidal $\mathbf{B}$-field leads to significant instabilities for time-steps $\delta t \gtrsim 10T_g$, which gives an indication for just how much one can increase the guiding center time-step without noticeable deviations.

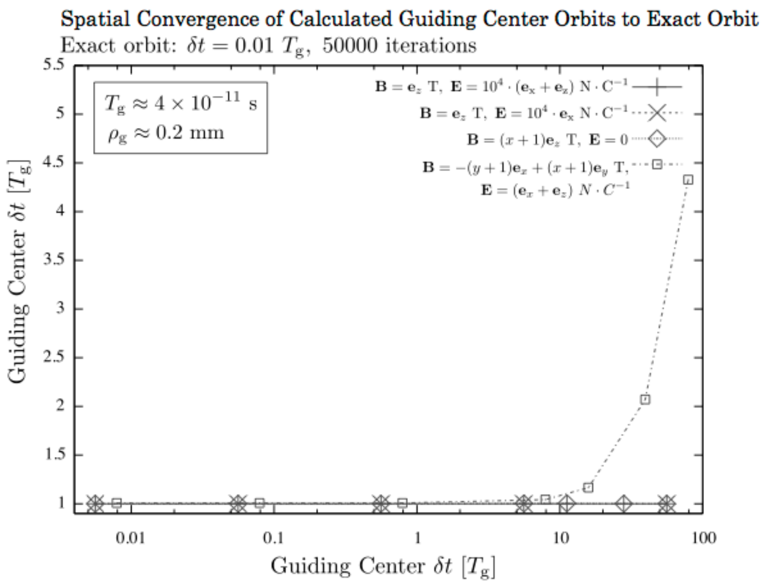


Figure 9: Spatial convergence of calculated guiding-center orbits to *exact* particle orbit. Start velocity was $\mathbf{v}_0 = (0.1, 0, 0.1) \cdot c$.

[11]Defined as $\max_i \|\mathbf{R}(t_i) - \mathbf{r}(t_i)\|$ with $t_1, t_2, ..$ as the time-samples for the guiding center orbit and $\mathbf{r}(t_i)$ as linear interpolation of the exact electron orbit.

5 Summary and Conclusions

The convergence tests in the previous section indicate that the time-step for the integration of the electron guiding centers can be increased to up to 10 gyro-periods while still maintaining an accuracy of 10^{-2} gyro-radii. As these tests where performed on single particle configurations in externally provided fields, tests remain to be performed for *real-life* configurations, in particular with high particle numbers.

As any conventional particle motion integrator (e.g. Runge-Kutta) can be naturally applied to integrating the guiding center by merely modifying the applied electrostatic field, stability for even bigger time-steps could be achieved.

Furthermore, improvements have been achieved in the way the external magnetic field is defined. A divergence-free, 3-point interpolation method has been introduced for the available triangular mesh, which in contrast to the previous method used, not only provides a magnetic field continuous at the mesh-vertices, but more importantly the complete field Jacobian $\nabla\mathbf{B}$. The resulting field is continuously differentiable almost everywhere[12].

Due to the simplicity and flexibility of the interpolation method, it can easily be improved in the following ways:

- Local (e.g. triangle-specific) interpolation data such as coordinate-rotation angle or even interpolation matrices could be stored along with the mesh and loaded directly upon startup. While this would increase the amount of data needed to be stored, it would also significantly reduce the time needed for the runtime interpolations.
- As the key of this interpolation is its reduction to 1-dimensional interpolations, other 1D-schemes (e.g. splines) can also be easily used. This would secure the differentiability of the field at least within the whole triangle. Additionally, more mesh vertices (e.g. from neighboring triangles) may be used for the interpolations. As an extreme case, the whole mesh could be used for a single interpolation. The latest would secure the continuity of the field within the whole simulation space.

6 Acknowledgments

I would like to thank Benjamin Berberich and Dr. Paul Gibbon for their support and advices, Robert Speck for his great effort at organizing the Guest Student Programme 2009 and Lukasz Kucharski for his numerous suggestions on improving this report.

References

1. *Plasma Physics via Computer Simulation*, C.K. Birdsall, A.B. Langdon, Adam Hilger (1991)
2. *Plasmaphysik*, R.J. Goldston, P.H. Rutherford, Vieweg (1998)
3. *The Physics of Plasmas, Lecture notes*, R. Fitzpatrick, The Universiy of Texas at Austin
4. *The Adiabatic Motion of charged Particles*; T.G. Northrop, Interscience Tracts on Physics and Astronomy (1963)
5. *Numerical Error in Electron Orbits with Large $\omega_c\Delta t$*, S.E. Parker, C.K. Birdsall, Journal of Computational Physics, Vol. 97, P. 91-102 (1991)
6. *Accurate Numerical Solution of Charged Particle Motion in a Magnetic Field*, H.X. Vu, J.U. Brackbill, Journal of Computational Physics, Vol. 116, P. 384-387 (1995)
7. *Pretty Efficient Parallel Coulomb-Solver*, P. Gibbon, Technical report, FZJ GmbH (2003)
8. *The Eirene and b2-Eirene Codes*, D. Reiter, M. Baelsmans, P. Börner, Fusion Science and Technology, 47:172 (2005)
9. *Many-Body Tree Methods in Physics*, S. Pfalzer, P. Gibbon (1996)

[12] In the Lebesgue sense.

Electronic Structure of Transition Metal Complexes with Phthalocyanines

Stefan Maintz

RWTH Aachen University
Faculty of Mathematics
Computer Science and Natural Sciences
Templergraben 55, 52062 Aachen

E-mail: stefan.maintz@rwth-aachen.de

Abstract:
Density functional theory (DFT) calculations have become an elemental tool for various natural sciences over the past decades. In this work calculations on $3d$-transition metal complexes with phthalocyanine ligands have been carried out using the B3-LYP hybrid functional, in order to evaluate the performance of DFT, applied to open-shell complexes that contain many electronic excited states in the vicinity of the ground state. The results have been compared with predictions and expectations arising from ligand field theory (LFT). It was found that DFT apparently has inherent errors as not even expected trends within the $3d$-period were obtained.

1 Motivation

Transition metal (TM) complexes of the porphyrin type are of central importance in bioinorganic chemistry as they frequently form the catalytic or active center of the prosthetic group of an enzyme, e.g. Haem [1].

Almost all TM cations form very stable complexes with phthalocyanines (Pc). These porphyrin-like complexes are easy to synthesize and crystallize, making these compounds well accessible for experiments.

Due to the size of the TM-porphyrin complexes, the number of most advanced quantum chemical methods that are capable of treating open-shell systems, i.e. many closely spaced low-lying electronic states, reasonably reliable is quite limited: MRCI is generally capable of treating these systems, but suffers from size-extensivity errors when a large number of electrons is correlated, and its computational scaling well beyond $O(N^7)$ makes its usage highly unfavorable. CASSCF deals with non-dynamic electron correlation only, whereas CASPT2 is presumably the most practical and accurate approach to these compounds, but also suffers from high computational costs and the set-up of such calculations is not a black-box scheme. Conventional single-reference methods, such as Hartree-Fock (HF), Møller-Plesset perturbation theory, and coupled cluster theory are not applicable to the present case of near-degenerate, low lying states. The density functional theory (DFT) along with associated response methods, namely time-dependent DFT (TDDFT), offer a viable approach to treating this compounds economically and are widely used in the respective communities. Apart from predictions of molecular structures, the reliability of these methods with respect to the electronic spectra is yet uncertain.

At least in the currently prevailing implementations, DFT is a single-reference method that is commonly applied to multi-reference problems. Hence, inaccuracies are to be expected, but so far DFT has been extensively used for TM complexes and has produced reasonable structural data.

TDDFT is expected to describe the singly excited states relative to the ground state quite reasonable, while charge-transfer (CT) excitations are typically described poorly. Alternatively, symmetry constraints can be employed in order to access the excited states, i.e. the ground states within each irreducible representation.

Therefore, the objectives of this study were to clarify whether DFT is indeed a feasible method to describe the electronic structure of TM complexes, and to analyze possible systematic defects.

2 Theoretical Background

2.1 Density Functional Theory

DFT is based on a theorem by Hohenberg and Kohn [2], which states that the ground state energy E and the associated properties of an electronic, non-degenerate many-body system are uniquely defined by the electron density $\rho(r)$ of that system, where r denotes a point in real space. Hence, the energy is a functional of the electron density $E[\rho(r)]$. Furthermore in a second theorem, Hohenberg and Kohn showed the applicability of the variational principle to that problem, i.e. the electron density of the ground state minimizes the energy.

Nevertheless, DFT methods were made feasible for computational chemistry, not until a formalism was proposed by Kohn and Sham [3] that can be described by the following Hamiltonian, where a parameter λ satisfies $0 \leq \lambda \leq 1$:

$$H_\lambda = T + V_{ext}(\lambda) + \lambda V_{ee}. \tag{1}$$

Therein T describes the kinetic energy of the system and V_{ee} the electron-electron repulsion. $V_{ext}(\lambda = 1)$ equals the real electrostatic potential caused by the interaction between the nuclei and the electrons. For the case of $\lambda = 0$ a hypothetical system with non-interacting electrons is depicted. For the values $0 < \lambda < 1$ the external potential $V_{ext}(\lambda)$ is adjusted in a way, so that the obtained electron densities for $\lambda = 1$ and $\lambda = 0$ match. While wavefunction based methods restrict the functional form of the wavefunction, in DFT the Hamiltonian is approximated in practice [4].

For the hypothetical system with N non-interacting electrons ($\lambda = 0$) it is known that the exact solution to the Schrödinger equation is given by a Slater determinant Δ that is built from molecular orbitals ϕ_i, and thus the exact kinetic energy T_Δ is given by [5]:

$$T_\Delta = \sum_{i=1}^{N} \left\langle \phi_i \left| -\frac{1}{2}\nabla^2 \right| \phi_i \right\rangle. \tag{2}$$

As this approximation neglects parts of the total kinetic energy, an additional term E_{XC} is introduced into the expression describing the DFT energy, whereas $J[\rho]$ describes the Coulomb energy of the inter-electronic energy and $E_{ne}[\rho]$ the energy arising from nuclear-electronic interaction:

$$E_{\mathrm{DFT}}[\rho] = T_\Delta[\rho] + E_{ne}[\rho] + J[\rho] + E_{XC}[\rho]. \tag{3}$$

The biggest problem to deal with is that the exact form of the so called exchange-correlation functional $E_{XC}[\rho]$ is unknown. However, many approximations and estimations are available. It is common to divide this functional into two discrete functionals E_X for the exchange and E_C respectively for the correlation related part:

$$E_{XC}[\rho] = E_X[\rho] + E_C[\rho]. \tag{4}$$

From equation (1) and the expression for $E_{XC}[\rho]$ resulting from equating equation (3) to the exact energy it can be shown [5] that E_X can be given by:

$$E_X = \int_0^1 \langle \Psi_\lambda | V_X(\lambda) | \Psi_\lambda \rangle \, \mathrm{d}\lambda. \tag{5}$$

A linear approximation for the dependency between V_X and λ gives:

$$E_X \approx \frac{1}{2} \langle \Psi_0 | V_X(0) | \Psi_0 \rangle + \frac{1}{2} \langle \Psi_1 | V_X(1) | \Psi_1 \rangle \, . \tag{6}$$

According to the considerations above, the first term of equation (6) corresponds to the case, where only exchange energy and no correlation energy exists. Hence, the exact exchange energy for that part of E_{XC} can be calculated, in the same way it is done in the HF method. Of course the above oversimplification is an example to demonstrate the principle, i.e. the actual amount a of exact exchange energy (E_X^{exact}) that needs to be included in so called hybrid functionals, is not known. Consequently, the functional can be given by:

$$E_{XC}^{\text{hybrid}} = (1-a)E_X^{\text{exact}} + a\,(E_X + E_C)\, . \tag{7}$$

The formerly mentioned Kohn-Sham orbitals ϕ_i do not have the same significance as those known from the HF theory (Koopmans' theorem). The only valid interpretation is that the ionization energy is given by the energetic difference between the N-electron and the $(N-1)$-electron system:

$$-I = E(N) - E(N-1) = \int_0^1 \epsilon_N(n)\mathrm{d}n \approx \epsilon_N\left(\frac{1}{2}\right) \approx \epsilon_N, \tag{8}$$

where $\epsilon_N(n)$ is the energetic eigenvalue of the N^{th} Kohn-Sham orbital with the according occupation number $0 \leq n \leq 1$. For extended systems, $\epsilon_N(n)$ does not change significantly with the variation of n from 0 to 1. Thus the negative eigenvalue of the highest occupied molecular orbital is approximately equal to the ionization energy I [6], like shown by equation (8). This needs to be kept in mind for evaluating the results.

2.2 Ligand Field Theory

The ligand field theory (LFT), as published by F.E. Ilse and H. Hartmann [7, 8], is a concept to describe the spectroscopic data and by that the electronic structure of coordination compounds in a semi-quantitative manner. It is based on symmetry considerations and greatly simplifies the structure of the ligands.

The fundamental simplification in LFT is that only the d-orbitals of the central atom, i.e. a metal ion, interact with a set of negative point charges representing the actual ligands coordinated to the metal [9].

An isolated metal ion has five energetically degenerate d-orbitals. Assume an uniform, spherically symmetric charge distribution surrounding the metal ion such that the interaction energy equals those of the sum of the charges that are brought in by, for now, six ligands. The electrostatic repulsion between the electrons in the orbitals and the surrounding spherical charge distribution elevates the energy of orbitals while retaining their degeneracy.

When the six aforementioned ligands are then considered as point charges to represent the positions of real ligands coordinated octahedrally around the metal ion, different interactions between each distinct d-orbital and the point charges are conceivable. Generally it is assumed that the point charges lie on the axes of the coordinate system. The relative orientation of ligands and d-orbitals results in more repulsion between the ligands and those d-orbitals pointing parallel to the coordinate axes, namely d_{z^2} and $d_{x^2-y^2}$.

Concerning the other three orbitals it is the other way round. This breaks their degeneracy by splitting them into the two groups e_g (higher energy) and t_{2g} (lower energy). This issue is clarified by figure 1. The notation to describe the splitting of the energy levels results from the group theory: in octahedral symmetry, i.e. in point group O_h, the respective orbitals transform according to the so called irreducible representations e_g and t_{2g}. Conventionally, the energy difference between those split-up levels is denoted as $10D_q$.

The actual size of $10D_q$ is influenced by a couple of factors. First, increasing the metal oxidation state attracts the negatively charged ligands more strongly, resulting in a stronger interaction. Hence, the value of $10D_q$ would be increased. Another source of influence is the geometric arrangement of the complex, including the ionic radius of the metal, the distance between ligand and metal, and the structure of the ligand. With a decreasing ionic radius the influence of the point charges on the orbitals would also be expected to decrease. This would result in a smaller energetic splitting. The last and presumably largest factor is the electronic nature of the ligand itself.

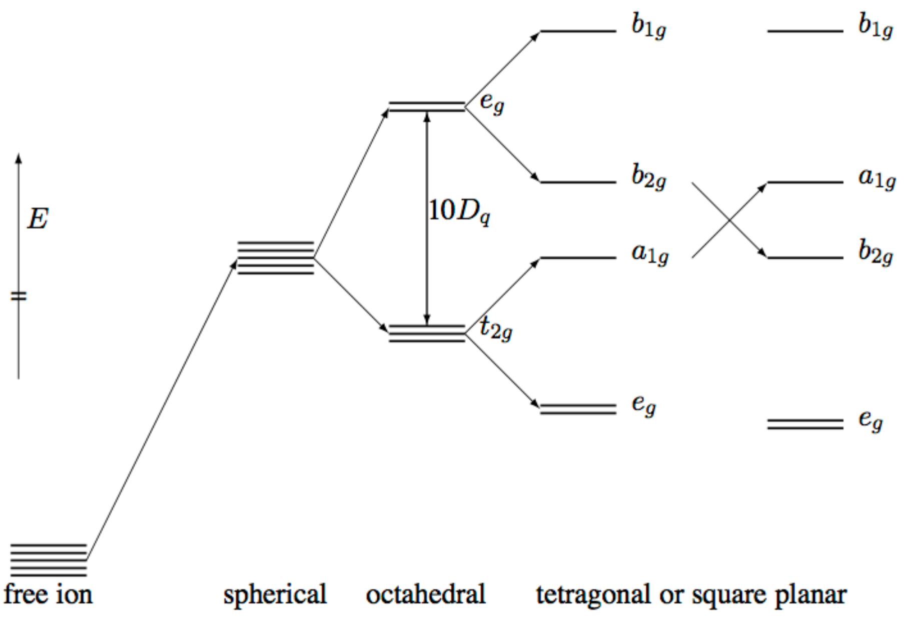


Figure 1: Energetic splitting of the d-orbitals in the different ligand environments. Actual splitting energies will differ. The order of the levels in the square planar case depends on the ligand field strength. Only the single electron case is shown, excluding d-d-electron interaction.

In this work TM complexes of square planar geometries were studied. Those belong to the point group D_{4h}, what applies to tetragonal complexes as well. The latter ones can be described as distorted octahedral geometries with an elongated metal-ligand distance on the z-axis, for example. Consequently, the influence of those ligands to the orbitals with a z-component decreases. Along with that their energetic levels decrease (in D_{4h} symmetry these orbitals correspond to e_g and b_{2g}), what means that their initial degeneracy is lifted. This consideration also applies to the square planar geometry because the its structure can be interpreted as a tetragonal structure, where the ligands on the z-axis are infinitely far away.

The d-orbitals are filled with electrons according to the Aufbau principle. Depending on the size of the actual energetic splittings, distinct, nearly degenerate energy levels arise from various d-orbital occupations. Since two electrons with alike spin cannot occupy the same spatial region, they experience less electron-repulsion than a pair of electrons with opposite spin (spin-pairing energy). Thus it is possible that more energy would be consumed by pairing two electrons in the lower lying orbital, in comparison to the case where one of these electrons occupies the lower and the other one the next higher lying orbital. Thereby it appears that so called high-spin complexes with maximum number of unpaired electrons are energetically favored as compared to their low-spin complements.

Beyond these considerations, huge sets of empirical data exist that describe the splitting parameters. Therefore even predictions for unexamined compounds are possible, as long as they match the previously recorded parameters.

A different approach to describe the electronic structure of molecules is given by the molecular orbital (MO) theory. MOs are represented as linear combinations of the metal and ligand orbitals. Similar to the H_2 case, the interaction of two such orbitals results in two linear combinations (MOs) thereof, one lying energetically lower than the separated ones ("bonding") and one lying higher ("antibonding"). The latter ones are usually marked by a * (c.f. figure 2). The energy difference of the resulting orbitals is proportional to the overlap between the participating orbitals, i.e. only orbitals belonging to the same irreducible representation will interact. Orbitals that cannot interact give so called "non-bonding" MOs retaining their energetic level.

When applied to TM complexes, MO theory generally comes to the same results as LFT. But in contrast, no simplifications are made because it treats the ligand orbitals in the same, and by that complete, manner like it does for the metal orbitals. The in figure 2 depicted MO scheme shows that LFT and MO theory agree about the relevant orbitals for the prevailing case of a D_{4h}-symmetry complex. The order and the gaps shown both visualizations highly depend on the actual system.

Within the underlying theory, interactions between ligand π-orbitals and metal d-orbitals must be treated. Hence, it is also possible to explain and computationally cover effects like π-donation and π-backdonation. I.e. depending on the occupation, it is possible that empty π^* MOs of the ligand interact with appropriate metal d-orbitals and form bonding orbitals. But concerning the prevailing Pc ligand, these phenomena are not to be expected. Thus, these effects cannot explain possible differences between DFT and LFT predictions here.

Strictly speaking, DFT is not an actual MO theory method, even though the computational implementation is based on the construction of MOs. Though it treats both atom and ligand on the same footing and should be expected to describe these interactions whereas LFT has to cope with the crude representation of the ligands.

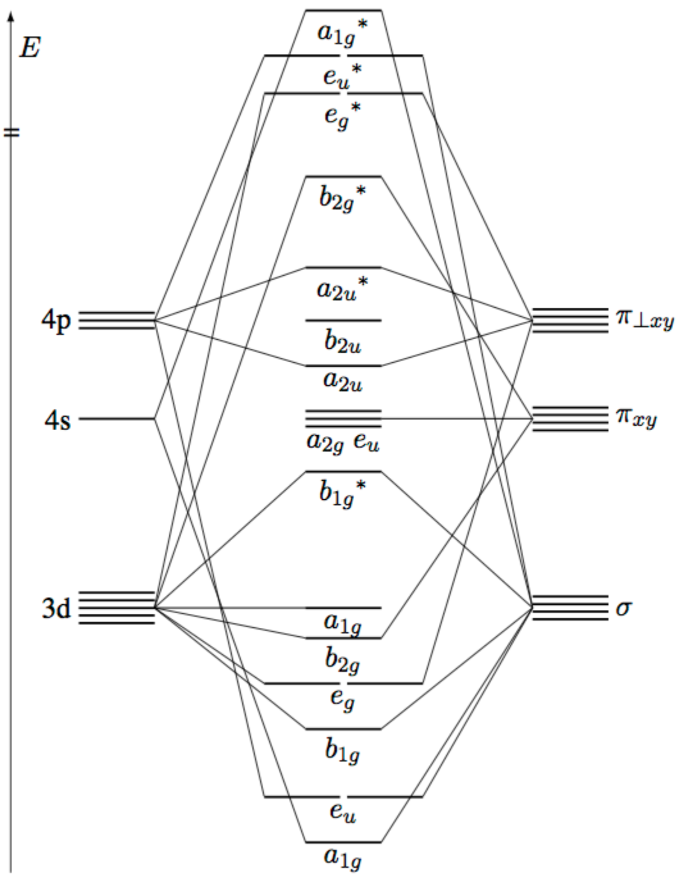


Figure 2: Molecular orbital diagram that qualitatively shows the interaction between the atomic orbitals of the metal ($3d, 4s, 4p$) and the appropriate molecular orbitals of the ligand. In relation to the ligand-metal bond, π_{xy} denotes those π-orbitals lying in the xy-plane and $\pi_{\perp xy}$ those perpendicular to it. The actual order and the energetic gaps of the levels vary with the system. The lowest four MOs (a_{1g}, e_u, b_{1g}) are occupied with electrons of the ligand.

3 Methodology

As shown above, LFT, which was intensively used in the sixties, enables a semi-quantitative treatment of TM complexes. By this, a comparison between the traditional expectations from LFT and the results of DFT calculations is possible and represents a practicable method for examining the accuracy of DFT calculations in the implementation used here (c.f. section 4).

The TMs considered in this study were restricted to the $3d$-block. When considering e.g. $5d$-metals one would have to take special care of the effects arising from the larger and more diffuse $5d$-orbitals. Furthermore, even with the use of appropriate pseudopotentials, relativistic effects are hard to deal with in DFT calculations, but should have little impact on the $3d$-block. In order to be able to observe the course of the electronic structure with increasing number of electrons, i.e. d^1 to d^{10} configurations, all TMs were assumed to be in oxidation state $+2$, even if the probability for the state to be formed is known to be very low. A fact to support the usefulness of this restriction is that the charge of the TM highly influences the energetic splitting of the orbitals, a phenomenon that was to be examined. Restricting the charge to $+2$ makes the according evaluation much less error prone.

Another origin for influences on the ligand field is the geometry and the type of the ligand. Hence, this was also kept the same, except for minor reoptimizations of the structure. These were necessary because the structure must lie in a minimum of the potential energy hypersurface, so that the calculations give physically meaningful results.

4 Computational Details

All DFT calculations were carried out using the TURBOMOLE V6.0 software suite [10] on the "Jülich Multiprocessor" IBM Power6 supercomputer JUMP. TURBOMOLE became the software of choice because it has the ability to make use of all finite point groups. In contrast, most of its competitors only allow the usage of the respective Abelian groups. This was of highest importance as the D_{4h} point group, which describes the symmetry of the TMPc complexes, is not Abelian. Otherwise the excited states could not have been accessed under the above described symmetry constraints.

The structure of the Pc-ligand (see figure 3) was initially generated by manually building the respective Z-Matrix with help of the MOLDEN utility. This more or less arbitrarily oriented structure was translated to fit its center of mass with the coordinate origin. Suitable rotations made the innermost, and thus presumably TM bond building, nitrogen atoms reside on either the X- or Y-axis, in order to conform to the discussed conventions in LFT (c.f. section 2.2).

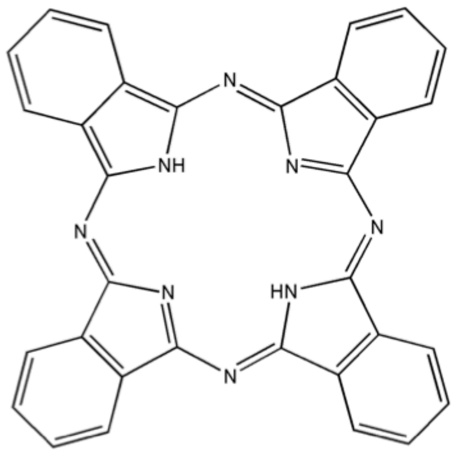


Figure 3: Schematic structure of the neutral Pc.

The structure of the ligand was optimized at a level of theory that consisted of unrestricted (Kohn-Sham-) DFT using the B3-LYP hybrid functional [11, 12, 13] and the def-TZVP basis set [14], assuming D_{4h} symmetry. For the integration of the exchange-correlation functional a grid size defined by the TURBOMOLE parameter `m4` was chosen.

The calculations of the TM complexes were carried out in the same manner, as described for the ligand, after removing the two innermost hydrogen atoms and placing the respective TM

ion at the coordinate origin. On the TMs the pseudopotential
ecp-10-mdf [15] was applied, in order to reduce calculation time and take care of potential relativistic effects, by replacing the ten innermost electrons by an effective potential.

The structures were reoptimized for every distinct TM complex. Those optimized structural parameters were held constant for the calculation of different electronic configurations within the same complex. Whenever a calculation with reduced symmetry was performed, the structure was reoptimized for the complex under the assumption of D_{2h} symmetry.

Mulliken [16] population analyses were carried out by the MOLOCH program from the TURBOMOLE suite. Because unrestricted DFT calculations were performed, the analyses were done for the MO sets of α and β spin separately.

For every complex, the configuration that was found to represent the ground state was also structurally reoptimized with a reduced symmetry of D_{2h}. Thus on the one hand, it was possible to check for consistency within different point groups. On the other hand, that step was necessary for complexes to which the Jahn-Teller theorem applies and a structural distortion is expected hence. Whenever the results from the D_{2h} calculation were in accordance with those assuming D_{4h} symmetry, the results are not printed explicitly to avoid redundancy.

For completeness, calculations displaying substantial spin-contamination, i.e. more than 10% deviation from the expectation value, are also listed as results (struck through in the tables), but were excluded from the evaluation.

5 Results

5.1 Occupation of Molecular Orbitals

To clarify the occupation of the MOs the results of the TMPc structure optimizations were used as input for electronic calculations with an enabled `$fermi` option. This permitted smearing of occupation numbers and thus TURBOMOLE reoptimized the occupations and spins. These calculations were repeated until the occupation numbers were not changing any more.

For ScPc the occupation numbers gave a single unpaired electron as expected. In further calculations, it was put into every shell that transforms according to the irreducible representation that describes the MO built from the d-orbital of the TM and the appropriate orbital from the innermost ligand-atom. In accordance to the considerations from LFT and MO theory (c.f. section 2.2) these refer to the irreducible representations e_g, a_{1g}, b_{2g} and b_{1g}. The procedure was repeated until enough configurations were calculated to identify ground and low-lying excited states.

Within the framework of LFT, the energy of the electronic states of the TM cation interacting with the surrounding point charges is considered. To check for consistency with the DFT predictions, the order and the nature of the electronic states is compared to the LFT predictions. Since bonding effects between ligand and TM cation are absent at LFT level of theory, the σ-bonding effect will additionally raise the energy of the b_{1g}* MO. The empty ligand π-orbitals may lower the energy of the b_{2g} and e_g MOs, while the a_{1g} MO is less affected relative to the LFT model. On the other hand especially open-shell systems are rather sensitive to the form of the exchange-correlation functional. It is of interest to answer the question whether DFT is capable of producing consistent results.

The automated DFT calculations gave the actual ground states only in very few cases. This was noticed, as many electronic calculations with manually specified occupation numbers to access excited states gave lower energies in comparisons to the automatically proposed ground state. Even if the automatic findings of DFT depend highly on the start vectors for the MOs (here, taken from extended Hückel theory calculations), this shows that DFT calculations are not always a black-box scheme. So special attention and manual intervention are necessary if one is interested in molecular properties besides structure.

The Mulliken population analyses showed for all calculated complexes that for a d^n configuration always $n + m$ electrons were assigned to the metal, where m satisfies $0.1 \leq m < 0.7$. This indicates that electron density is shifted from the ligand towards the metal cation consistent with the empirical finding that ligands of this kind are weak π-acceptor ligands.

5.2 Comparison of DFT Results with Expectations from LFT

In the following the results of the calculations for different electronic states using DFT will be presented and discussed. Furthermore, the results have been compared to so-called Tanabe-Sugano diagrams [17, 18], which display the energy of the d^n electron states as a function of the octahedral ligand field splitting parameter $\Delta = 10D_q$, including configuration interaction among the multitude of possible d^n states that can be obtained by distributing n electrons among the d-orbitals. As discussed in section 2.2, the square planar geometry is the limiting case of the tetragonal elongation of the octahedral geometry, so that Tanabe-Sugano diagrams for the tetragonal case [19] were used. Here, the energy of a d^n state is expressed relative to the octahedral case by introducing two additional parameters D_s and D_t that are related to in-plane and out-of-plane interaction with the surrounding point charges. The diagrams give the energy in terms of the parameters D_q for a fixed ratio $\frac{D_s}{D_t}$. In the limiting square planar case the ratio of these two parameters is given by

$$\frac{D_s}{D_t} = R_{ML}^2 \frac{\langle r^2 \rangle}{\langle r^4 \rangle} \approx R_{ML}^2 \frac{1}{4} \geq 0, \tag{9}$$

where R_{ML} denotes the metal-ligand distance and $\langle r^n \rangle$ the n^{th} moment of the metal cation. The ratio is approximately $\frac{1}{4}$ for the metal cations at the HF limit and R_{ML} is about 2 Å, so that $\frac{D_s}{D_t}$ is expected to result in about 4. For the d^1 case the actual energies are given by

$$\begin{aligned}
E(B_{2g}) &= -4D_q + 2D_s - D_t \approx -4D_q + 7D_t & (10)\\
E(E_g) &= -4D_q - D_s + 4D_t \approx -4D_q & (11)\\
E(B_{1g}) &= 6D_q + 2D_s - D_t \approx 6D_q + 7D_t & (12)\\
E(A_{1g}) &= 6D_q - 2D_s - 6D_t \approx 6D_q - 14D_t. & (13)
\end{aligned}$$

The octahedral t_{2g} orbitals will split by $7D_t$, while the octahedral e_g orbitals will split by $21D_t$. Additionally, the square planar b_{2g} orbital will be separated by $10D_q - 21D_t$ from the a_{1g} orbital in this simplified one-electron picture of the d^1 case [20].

By inserting the computed energies, the linear equations can only be solved in a least squares sense. Actually the resulting parameters D_s, D_t are widely different.

To estimate the strength of the ligand field, the splitting parameters D_q, D_s and D_t were calculated by evaluating the set of equations (10)–(13), filled with the appropriate energy differences from the Scandium calculations because that system directly gives the necessary energies, as no pairing energies or similar exist in this case (c.f. section 5.2.1). This lead to $D_q = 2917.72\ \text{cm}^{-1}$, $D_s = 6351.05\ \text{cm}^{-1}$ and $D_t = 2038.46\ \text{cm}^{-1}$. The latter result was confirmed by calculating it two times mixing different equations and by that different states. These values show that the above made estimation for the ratio $\frac{D_s}{D_t}$ seems reasonable, as it fits the order of magnitude. While a value of $10D_q \approx 29000\ \text{cm}^{-1} \approx 340\text{nm}$ is not unreasonable the size of the splitting parameter appears to be rather large.

In the following subsections the results for the calculations will be discussed and for completeness tabulated. The occupations for the orbitals that transform according to an irreducible representation of ungerade symmetry are omitted in those tables because for D_{4h} symmetry they are invariably given by $(a_{1u})^{2\uparrow}_{2\downarrow}$ $(a_{2u})^{5\uparrow}_{5\downarrow}$ $(b_{1u})^{2\uparrow}_{2\downarrow}$ $(b_{2u})^{3\uparrow}_{3\downarrow}$ $(e_u)^{58\uparrow}_{58\downarrow}$. Besides that, the energetic order and the occupations of the MOs arising from the metal d-orbitals are illustrated for the found ground state of each considered complex. It needs to be remarked that the illustrations do not present any energetic scale and serve visualization purposes only.

5.2.1 Scandium d^1

Depending on the strength of the ligand field and by that how much the e_g and t_{2g} levels from the octahedral symmetry will split up, the LFT considerations would expect the energetic order of the levels in square planar geometry to either be $e_g < a_{1g} < b_{2g} < b_{1g}$ for the larger splitting or $e_g < b_{2g} < a_{1g} < b_{1g}$ for the smaller splitting case.

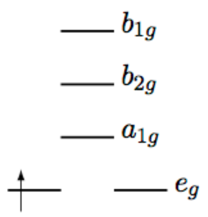


Table 1 shows that DFT predicts the ground state for ScPc to be ^2E_g. The calculation of excited states by the use of symmetry restrictions by shifting the unpaired electron from $e_g \rightarrow X$, where $X = b_{2g}, a_{1g}, b_{1g}$ shows that the energetic order of the orbitals indeed matches the expectations for the case with smaller D_q and larger D_t splitting. Since strong ligand fields cause large level splittings, this observation can be taken as a hint for a strong prevailing ligand field.

ScPc	d^1: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
^2E_g	0.0000	0.752936	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
$^2A_{1g}$	0.3027	0.750906	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$
$^2B_{2g}$	1.0986	0.750761	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$
$^2B_{1g}$	4.7162	0.751958	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$

Table 1: Results for ScPc $E_0 = -1712.868803$ Ha.

The above calculated ground state is degenerate. As mentioned in section 2.1, precisely DFT does not apply to these cases. The results actually represent an arbitrary linear combination of the degenerate states. Furthermore the Jahn-Teller theorem states that the symmetry will be reduced and the degeneracy of the E_g state will be lifted. The reoptimization of the structure is in accordance with that and the results are shown in table 2.

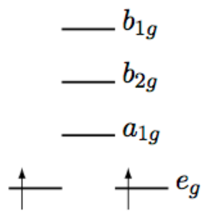
The expected structural distortion was observed. Assuming D_{4h} symmetry, the shortest Sc-N distances and N-N distances were given by 2.0977 and 2.9666 . Under D_{2h} symmetry restriction the N-N distance was reduced to 2.9059 and the Sc-N distances split up to 2.0603 and 2.0493 . Even if the distortion was small, this is noteworthy for the inflexible Pc-ligand.

ScPc	d^1: D_{2h} symmetry		
state	$\Delta E/\mathrm{eV}$	$\langle S^2 \rangle$	occupation
$^2B_{3g}$	0.0000	0.757935	$(a_g)^{33\uparrow}_{33\downarrow}(b_{1g})^{24\uparrow}_{24\downarrow}(b_{2g})^{5\uparrow}_{5\downarrow}(b_{3g})^{\mathbf{6}\uparrow}_{5\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$
$^2B_{2g}$	0.2394	0.757138	$(a_g)^{33\uparrow}_{33\downarrow}(b_{1g})^{24\uparrow}_{24\downarrow}(b_{2g})^{\mathbf{6}\uparrow}_{5\downarrow}(b_{3g})^{5\uparrow}_{5\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$
^2A_g	0.8175	0.750912	$(a_g)^{\mathbf{34}\uparrow}_{33\downarrow}(b_{1g})^{24\uparrow}_{24\downarrow}(b_{2g})^{5\uparrow}_{5\downarrow}(b_{3g})^{5\uparrow}_{5\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$
$^2B_{1g}$	1.6819	0.750758	$(a_g)^{33\uparrow}_{33\downarrow}(b_{1g})^{\mathbf{25}\uparrow}_{24\downarrow}(b_{2g})^{5\uparrow}_{5\downarrow}(b_{3g})^{5\uparrow}_{5\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$

Table 2: Results for ScPc under D_{2h} symmetry. $E_0 = -1712.896597$ Ha.

5.2.2 Titanium d^2

According to LFT, one expects the ground state of TiPc to be $^3A_{2g}$ arising from an $(e_g)^2$ configuration, regardless of the ligand field strength. This is nicely reproduced by the according DFT calculation as shown in table 3. Furthermore, the table shows that the first excited state was found to be ^3E_g with a $(a_{1g})^1(e_g)^1$ occupation, followed by one with $(b_{2g})^1(e_g)^1$ occupation.



Following the corresponding Tanabe-Sugano diagram, in the range of small splitting energies (i.e. $D_q < 500$ cm^{-1}) it might be possible that the first excited ^3E_g state (arising from $(a_{1g})^1(e_g)^1$ occupation) is energetically favored in comparison to the one arising from $(b_{2g})^1(e_g)^1$ occupation, like it is proposed by the DFT results. But as other considerations above have shown that a strong ligand field is much more likely and the calculation of D_q showed an a lot greater splitting parameter D_q, these results raise first doubts that DFT is able to treat the electronic structure of the complexes in the right manner for states that lie closely together.

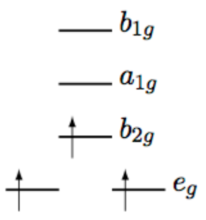
In quite some energetic separation the ^3E_g state describing the $(b_{1g})^1(e_g)^1$ occupation is found, about what DFT and LFT regardless of ligand field strength agree about.

Furthermore this is in agreement with the order of the energy levels found for the ScPc compound, what is important to notice as no rigorous changes to the parameters that determine the ligand field strength (c.f. section 2.2) were made.

TiPc	d^2: D_{4h} symmetry		
state	$\Delta E/\mathrm{eV}$	$\langle S^2 \rangle$	occupation
$^3A_{2g}$	0.0000	2.024056	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
^3E_g	0.4697	2.014221	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
^3E_g	0.7311	2.017528	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
^3E_g	4.3540	2.016453	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$

Table 3: Results for TiPc $E_0 = -1724.353365$ Ha.

5.2.3 Vanadium d^3



For a prevailing strong ligand field, i.e. $D_q \geq 570\ \text{cm}^{-1}$, LFT predicts a ground state of $^4B_{1g}$ for VPc. The DFT results listed in table 4 agree with that prediction.

Concerning the next exited state, DFT predicts this to be a $^4A_{2g}$ state. The course of the appropriate Tanabe-Sugano diagram on the opposite unambiguously shows that the mentioned state lies higher in energy than an ^4E_g state, for every ligand field stronger than the above mentioned boundary for D_q. DFT found both those states, but in inverted order. The results from DFT for the states $^2B_{2g}$ and ^2E_g on the contrary are in accordance with the LFT predictions.

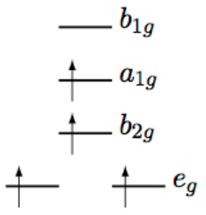
By comparing the DFT results for the $^4B_{1g}$ and $^4A_{2g}$ state it is observed that the a_{1g} and the b_{2g} levels have swapped their energetic order relative to the order that was determined by the examination of the exited states of the former TM complexes (c.f. tables 1 and 3). The only reasonable way to explain the observed phenomenon is the variation of the ionic radius, as it is the only parameter to influence the ligand field strength that changes in the course of the $3d$-metals. Due to the decreasing ionic radius also the influence of the ligand to the metal orbitals decreases. By that there is a smaller energetic splitting of the MOs, what explains the observation.

However, for the doublet states, the ordering of states is wrong: instead of a ^2E_g ground state we find $^2B_{2g}$ and a poorly described ^2E_g state suffering from spin-contamination.

VPc	d^3: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
$^4B_{1g}$	0.0000	3.773118	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^4A_{2g}$	0.4545	3.781388	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
^4E_g	0.6166	3.805412	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
$^2B_{2g}$	1.3483	0.756045	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$
~~^2E_g~~	~~1.5988~~	~~1.303159~~	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
$^2A_{1g}$	1.8515	0.752674	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$
^2E_g	2.5323	0.762440	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
$^4B_{2g}$	3.3047	3.766706	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$

Table 4: Results for VPc $E_0 = -1737.610993$ Ha.

5.2.4 Chromium d^4



For the CrPc complex the Tanabe-Sugano diagram predicts a ground state of $^5B_{1g}$ up to a splitting parameter of $D_q \approx 2700\ \text{cm}^{-1}$. If one assumes that the initial calculations for D_q holds, the decreasing ionic radius must be responsible for a possible drop in ligand field strength. Thus, the ground state predicted by DFT can be brought into accordance with the LFT. A point needed to be stressed here is that all the ^3E_g states suffer from not marginal spin contamination, what can be seen in table 5.

Ignoring the ^3E_g state, LFT predicts $^3A_{2g}$ to be the second excited state as long as the ligand field strength is described by $D_q \geq 1900\ \text{cm}^{-1}$. This is in compliance with any previous findings for that parameter. For a next excited state of $^1A_{2g}$ the two theories could still be in accordance. But for the following state $^5A_{1g}$ is predicted by LFT, when E_g states are also ignored there. Table 5 shows various states in between. Thus, agreements between the two theories appear to be more coincidental than regular, even if the first excited states seem to match.

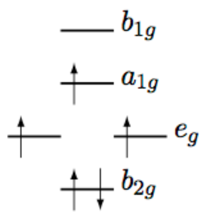
In this complex, it appears energetically favored to occupy even two higher lying orbitals than to pair even one into the e_g orbitals. This can be taken as an indicator for states that really lie closely together.

CrPc	d^4: D_{4h} symmetry		
state	ΔE/eV	$\langle S^2 \rangle$	occupation
$^5B_{1g}$	0.0000	6.027355	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
~~^3E_g~~	~~2.2992~~	~~2.439196~~	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
$^3A_{2g}$	2.4221	2.016674	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
~~^3E_g~~	~~2.8083~~	~~2.883126~~	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
$^1A_{1g}$	3.0377	0.000006	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{10\uparrow}_{10\downarrow}$
$^3A_{2g}$	3.2553	2.031221	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^1A_{1g}$	3.3510	0.000000	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^5A_{1g}$	3.6220	6.013236	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$

Table 5: Results for CrPc $E_0 = -1752.834411$ Ha.

5.2.5 Manganese d^5

Again it was observed that the ^2E_g states suffer from spin contamination and their calculations, hence, were excluded from considerations. Though, it does attract attention that these problems occur mostly on E_g states. This might be caused by the discussed non-applicability of DFT to degenerate states.



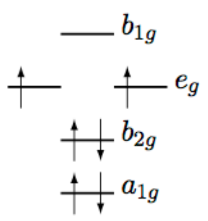
But nonetheless DFT delivers some interesting results here. Regardless of the Tanabe-Sugano diagrams one would expect the ground state to either be of a configuration, where every electron singly occupies an orbital or where the double occupation is placed in the lowest lying orbitals, i.e. e_g. Funnily enough neither of these cases is given by DFT for the ground state. But for the first excited state the mentioned high spin state is found, like shown in table 6. Anyhow, the illustration has the doubly occupied orbital drawn as the lowest lying one.

Another noteworthy observation is the swapped back energetic order of the b_{2g} and the a_{1g} orbitals and the drop of the b_{2g} level even below the e_g orbital. This contradicts any previously made explanation concerning the decreasing ionic radius as cause for the initial swap. Therefore it is not surprising that the comparison with LFT gives a likewise disastrous result for DFT. The Tanabe-Sugano diagram is in accordance with the theoretical considerations above and proposes either $^6A_{1g}$ for low values of D_q or ^2E_g for stronger ligand fields.

MnPc	d^5: D_{4h} symmetry		
state	ΔE/eV	$\langle S^2\rangle$	occupation
$^4A_{2g}$	0.0000	3.775470	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^6A_{1g}$	0.1307	8.760284	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^4B_{1g}$	0.2673	3.782345	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^2B_{2g}$	1.5982	0.757790	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
^2E_g	2.5086	0.817771	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{11}\uparrow}_{10\downarrow}$
~~^2E_g~~	~~2.7047~~	~~0.891918~~	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
~~$^2A_{1g}$~~	~~2.7223~~	~~2.761327~~	$(a_{1g})^{18\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^4B_{2g}$	3.3530	3.791979	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^4B_{2g}$	4.0505	3.760476	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^2B_{1g}$	4.5880	0.756538	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^4A_{2g}$	5.8157	3.773826	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^4B_{1g}$	6.8794	3.761856	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$

Table 6: Results for MnPc $E_0 = -1770.062054$ Ha.

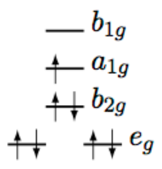
5.2.6 Iron d^6



The d^6 complex shows the same unexpected behavior that was already observed at MnPc concerning the unexplainable deviation from the Aufbau principle. In this case it is inevitable that one electron in the d-orbitals must be paired with another. Even for the high spin case one absolutely expects the pairing to take place in the lowest lying orbitals. DFT proposes instead that two electrons are paired in the energetically higher lying a_{1g} and b_{2g} orbitals, while the two degenerate e_g-orbitals only hold a single unpaired electron each. Also the orbital energies are swapped back the same way as observed at MnPc. As this fault occurs more than once it may be deduced that this must be an inherent error of DFT or the functional used. Again, the illustration was drawn to appear reasonable, even if the described occupation shifting method states a different order.

By means of the Tanabe-Sugano diagrams, LFT again complies to the expectations by predicting the high spin state ^5E_g for ligand field splittings of D_q above 750 cm^{-1} and the low spin state $^1A_{1g}$ for strong ligand fields above 1750 cm^{-1}.

5.2.7 Cobalt d^7



The energetic order of the orbitals remains unclear here because in this case one cannot compare the shifting of one electron from the lower orbitals to the highest one because of spin pairing or unpairing.

Under the assumption that the problem of back swapping does not occur in the calculation of this complex, the problems concerning the Aufbau principle encountered

FePc	d^6: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
$^3A_{2g}$	0.0000	2.018911	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^5A_{1g}$	0.9699	6.018013	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^5B_{2g}$	1.3628	6.013521	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
$^1A_{1g}$	1.4194	0.000000	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^1A_{1g}$	4.3843	0.000000	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^3A_{2g}$	6.4884	2.031745	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$

Table 7: Results for FePc $E_0 = -1789.535064$ Ha.

above have vanished here. A possible explanation for this observation might be that not as much unoccupied, near-degenerate states, which also compete with spin pairing energies, exist here anymore.

For strong ligands fields, meaning $D_q \geq 1600\,\text{cm}^{-1}$, the Tanabe-Sugano diagram predicts a $^2A_{1g}$ ground state. This matches the prediction by DFT. But, the results for the exited states cannot be brought into accordance, since the order of the $^4B_{2g}$ and the $^2B_{2g}$ state (c.f. table 8) is only correct for an assumed ligand field with a splitting parameter in the range from 1600 to 2000 cm^{-1}. The above findings estimated that parameter higher.

CoPc	d^7: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
$^2A_{1g}$	0.0000	0.759668	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^4B_{2g}$	0.3486	3.758280	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
~~^2E_g~~	~~1.3371~~	~~0.877428~~	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
$^2B_{2g}$	1.6636	0.761701	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$

Table 8: Results for CoPc $E_0 = -1811.336399$ Ha.

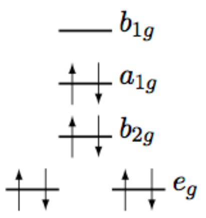
When the automatically by DFT proposed ground state $^4B_{2g}$ was reoptimized under D_{2h} symmetry for consistency checks, it was observed that the total energy E_0 of the D_{2h} complex was even lower than E_0 of the actual, i.e. manually found, ground state $^2A_{1g}$ in D_{4h} (c.f. tables 8 and 9). The e_g orbitals split up by only 0.0057 ev, but even that is not expected because the ground state is not degenerate. A possible explanation is that the resulting energetic advantage from D_{2h} over D_{4h} symmetry is derived from structural distortions in the ligand that have no direct effect on the d-orbitals. But nonetheless another error in DFT or the functional has been discovered, since these distortions are not reasonable. For completeness it is noted that the D_{2h} calculations suffer from spin contamination that might affect the results, even though it is below the above mentioned threshold of 10%.

CoPc	d^7: D_{2h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
${}^2B_{3g}$	0.0000	0.813786	$(a_g)^{34\uparrow}_{34\downarrow}(b_{1g})^{25\uparrow}_{25\downarrow}(b_{2g})^{6\uparrow}_{6\downarrow}(b_{3g})^{\mathbf{6}\uparrow}_{5\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$
${}^2B_{2g}$	0.0057	0.801988	$(a_g)^{34\uparrow}_{34\downarrow}(b_{1g})^{25\uparrow}_{25\downarrow}(b_{2g})^{\mathbf{6}\uparrow}_{5\downarrow}(b_{3g})^{6\uparrow}_{6\downarrow}(a_u)^{4\uparrow}_{4\downarrow}(b_{1u})^{8\uparrow}_{8\downarrow}(b_{2u})^{29\uparrow}_{29\downarrow}(b_{3u})^{29\uparrow}_{29\downarrow}$

Table 9: Results for CoPc $E_0 = -1811.351709$ Ha.

5.2.8 Nickel d^8

The calculations on this complex allowed the clarification of the energetic order of the orbitals again and gave the results that are in accordance with the considerations made above, so except for some outliers the initially observed and expected trend seems to hold.



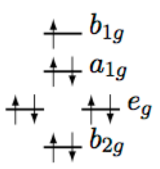
The isolated DFT results (c.f. table 10) appear reasonable because no previous ground state included an unpaired electron in the b_{1g} orbital. The energetic gap to this orbital appears relatively high. LFT on the other hand predicts a triplet ground state ${}^3B_{1g}$ for all $D_q \geq 600\ \text{cm}^{-1}$ since the splitting of the b_{1g} and b_{2g} orbitals is less than the spin-pairing energy. MO theory would — in agreement with the DFT results — predict a higher energetic gap to b_{1g} (cf. Fig. 2). Whether this is sufficient to have a singlet ground state cannot be judged from this comparison.

NiPc	d^8: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2 \rangle$	occupation
${}^1A_{1g}$	0.0000	0.000000	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
${}^3B_{1g}$	0.8209	2.004724	$(a_{1g})^{\mathbf{19}\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
${}^3A_{2g}$	1.7696	2.009919	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
${}^3E_g$	1.8942	2.044732	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{15\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{11\downarrow}$
${}^3A_{2g}$	3.8179	2.008883	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{10\downarrow}$
${}^1A_{1g}$	5.8517	0.000000	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{13\uparrow}_{13\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
${}^1A_{1g}$	6.5411	0.000000	$(a_{1g})^{18\uparrow}_{18\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$

Table 10: Results for NiPc $E_0 = -1836.189916$ Ha.

5.2.9 Copper d^9

In this d^9 complex the b_{2g} level has energetically dropped even below the e_g orbitals, as can be seen from table 11. This is expected for a further growing ligand field strength. But on the opposite, that the latter one is decreasing with increasing atomic number, as justified above. This raises doubts to the accurateness of the applied method, namely DFT with a hybrid functional.

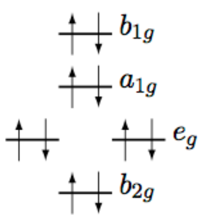


Though it must be said, even with no Tanabe-Sugano diagram available, the ground state is highly probable correct. But the excited states can be assumed to be incorrect due to the same reasoning used for the considerations of the energetic orbital order.

CuPc	d^9: D_{4h} symmetry		
state	$\Delta E/\text{eV}$	$\langle S^2\rangle$	occupation
$^2B_{1g}$	0.0000	0.753348	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{15\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
$^2B_{2g}$	2.9098	0.752804	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{13\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$
^2E_g	3.1985	0.762890	$(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{11}\downarrow}$
$^2A_{1g}$	3.2742	0.751806	$(a_{1g})^{18\uparrow}_{\mathbf{19}\downarrow}(a_{2g})^{11\uparrow}_{11\downarrow}(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$

Table 11: Results for CuPc $E_0 = -1862.581962$ Ha.

5.2.10 Zinc d^{10}



As the d^{10} Zinc-complex has a completely filled d-shell, there are no possible configurations or distortions to be examined. Therefore the results of the calculation are given by the ground state energy $E_0 = -1892.281690$ Ha and the expectation value for the spin operator $\langle S^2\rangle = 0.000000$. The associated state is the totally symmetric one of $^1A_{1g}$ arising from a configuration of $(a_{1g})^{\mathbf{19}\uparrow}_{\mathbf{19}\downarrow}$ $(a_{2g})^{11\uparrow}_{11\downarrow}$ $(b_{1g})^{\mathbf{16}\uparrow}_{\mathbf{16}\downarrow}$ $(b_{2g})^{\mathbf{14}\uparrow}_{\mathbf{14}\downarrow}$ $(e_g)^{\mathbf{12}\uparrow}_{\mathbf{12}\downarrow}$.

6 Conclusion

In this study, it was shown that DFT apparently has some major problems treating open-shell TM complexes. This is especially true for the excited states that were calculated under symmetry restrictions and becomes most evident when there are many states that lie very closely together. Only in the fewest cases DFT was able to automatically find the ground state within its scope. So manual consideration of the states is inevitable if one is interested in more than just complex geometries or rough energy estimates.

The greatest problem that was revealed here, certainly is that not even expected trends inside the $3d$-period were detectable. The comparison to another theory (LFT) describing the same quantities shows different predictions throughout the whole period. But also LFT has some drawbacks. E.g. π-bonds are not taken into account and in the prevailing complexes a delocalized π-system exists, what might have an impact on its predictions. Comparison with the MO diagram (fig. 2) indicates, that the description of the e_g, b_{2g}, a_{1g} and to lesser extent b_{1g} metal ion orbitals as well as the relative energetic location to the ligand σ and $\pi_{\perp xy}$ orbitals is crucial for the description of the electronic states. While the uniform treatment of ligand and metal ion is superior over LFT, the lack of systematic trends among the calculated states at DFT level suggests that DFT based on hybrid functionals is not yet accurate enough to resolve the electronic structure to this level of detail. This applies in particular to d^3 to d^7 transition metal ions.

Acknowledgments

I want to thank my adviser Dr. T. Müller for his support and the huge amount of time he could spare for my project. Furthermore, I want to express my gratitude to everybody involved in instantiating the programme, but in especially to Mr. R. Speck and Mr. B. Berberich for their work as organizers, and also to my professor Dr. R. Dronskowski for his recommendation and to Dr. B. Eck, who initially suggested the programme to me. I really appreciated the working atmosphere I encountered at the JSC what was mainly, but not only, caused by my fellow guest students.

References

1. A. Lehninger, D. L. Nelson, M. M. Cox, *Lehninger Principles of Biochemistry* 5th ed., W. H. Freeman (2008)
2. P. Hohenberg, W. Kohn, *Phys. Rev.*, *136*, B864 (1964)
3. W. Kohn, L. J. Sham, *Phys. Rev.*, *140*, A1133 (1965)
4. J. Reinhold, *Quantentheorie der Moleküle: eine Einführung*, Teubner, Stuttgart (1995)
5. F. Jensen, *Introduction to Computational Chemistry*, John Wiley & sons, Chichester (2001)
6. R. G. Parr, W. Yang, *DFT of Atoms and Molecules*, Clarendon Press, Oxford (1989)
7. F. Ilse, H. Hartmann, *Z. Phys. Chem.*, *197*, 239 (1951)
8. F. Ilse, H. Hartmann, *Z. Naturforsch.*, *6a*, 751.(1951)
9. J. E. Huheey, *Anorganische Chemie: Prinzipien von Struktur und Reaktivität*, de Gruyter, New York (1988)
10. TURBOMOLE V6.0 2009, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from `http://www.turbomole.com`.
11. A. D. Becke, *Phys. Rev. A*, *38*, 3098 (1988)
12. C. Lee, W. Yang, R. G. Parr, *Phys. Rev. B*, *37*, 785 (1988)
13. A. D. Becke, *J. Chem. Phys.*, *98*, 5648 (1993)
14. C. H. A. Schäfer, R. Ahlrichs, *J. Chem. Phys.*, *100*, 5829 (1994)
15. M. Dolg, U. Wedig, H. Stoll, H. Preuss, *J. Chem. Phys.*, *86*, 866 (1987)
16. R. S. Mulliken, *J. Chem. Phys.*, *23*, 1833 (1955)
17. Y. Tanabe, S. Sugano, *J. Phys. Soc. Japan*, *9*, 753 (1954)
18. Y. Tanabe, S. Sugano, *J. Phys. Soc. Japan*, *9*, 766 (1954)
19. S. K. E. Koenig, *Ligand Field Diagrams*,Plenum Press, New York (1977)
20. H. Schäfer, G. Gliemann, *Einführung in die Ligandenfeldtheorie*, Akademische Verlagsgesellschaft, Frankfurt a. M. (1967)

Analysis of a parallel preconditioner

Hannah Rittich

Bergische Universität Wuppertal
Fachgruppe Mathematik und Informatik
Gaußstraße 20, 42119 Wuppertal

Email: hannah@rittich.net

Abstract:
In this report we will analyse the quality and performance of the library Hypre, a library wich provides highly parallel preconditioners and solvers for large sparse systems of linear equations. First we will give a short introduction into a certain class of iterative methods and preconditioners. Afterwards we will present results from different experiments analysing different parts of Hypre.

1 Introduction

Solving large sparse linear systems is a task which often occurs in practical applications, for example when discretizing partial differential equations or in the finite element method. So the ability to solve these systems efficiently would be desirable. Some well known iterative methods like the Conjugate Gradient Method [1] or the Generalized Minimal Residual Method [1] can be used. But to work efficiently these methods need to be preconditioned in an appropriate way. The library *Hypre*, which is developed at the Lawrence Livermore National Laboratory, provides a set of parallel preconditioners and solvers. It enables us tomake practical use of these methods on large parallel computers. We will now analyse the quality and scalability of the methods provided by Hypre.

2 General Theory

In this section we will give a brief introduction into projection methods which are a very general class of iterative methods for solving systems of linear equations. After that we will describe the two known Krylov subspace methods; the Conjugate Gradient Algorithm (CG) and the Generalized Minimal Residual Method (GMRES), which are special projection methods.

Let $A \in \mathbb{R}^{n\times n}$ a square matrix, $b \in \mathbb{R}^n$ a vector and $x \in \mathbb{R}^n$ the solution of

$$Ax = b. \tag{1}$$

Our task is to find an approximation $\tilde{x}$ of the solution x from (1). We are especially interested in the case, where A is a large sparse matrix.

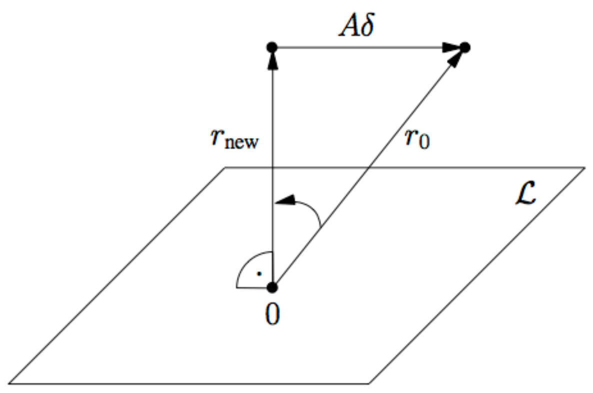


Figure 1: Interpretation of the orthogonality condition.

2.1 Projection methods

Projection methods provide the basic theory for most of the iterative methods known today. In this section we will give a short introduction into the topic.

The main idea of this method is to search for an approximative solution $\tilde{x}$ in a m dimensional subspace, say $\mathcal{K} \subset \mathbb{R}^n$, which we will call *the search subspace*. To uniquely determine $\tilde{x}$ we need to supply at least m constraints for the choice of $\tilde{x}$. We do this by requesting m different orthogonality conditions. We can formulate this, as the approximation $\tilde{x}$ will be chosen, so that $(b - A\tilde{x}) \perp \mathcal{L}$, where $\mathcal{L} \subset R^n$ is an other m dimensional subspace.

If we want to supply an initial guess x_0 to our method, we will need to search $\tilde{x}$ in the subspace $x_0 + \mathcal{K}$. So we will

$$\text{choose } \tilde{x} \in x_0 + \mathcal{K}, \quad \text{so that} \quad (b - A\tilde{x}) \perp \mathcal{L}. \tag{2}$$

Definition 1. If $\tilde{x} \in \mathbb{R}^n$ is an approximation of the solution x of the equation (1), we call

$$r = b - A\tilde{x}$$

the *residual* of the approximation $\tilde{x}$. The *relative residual* is defined by

$$\hat{r} = \frac{r}{||b||}.$$

Using the fact, that we can write $\tilde{x}$ as

$$\tilde{x} = x_0 + \delta, \qquad \text{where } \delta \in \mathcal{K}$$

and by defining r_0 as the residual of x_0, the orthogonality condition leads to

$$b - A(x_0 + \delta) = (r_0 - A\delta) \perp \mathcal{L}.$$

This is shown in figure 1. For a more detailed view on projection methods see [1].

2.2 Krylov Subspace Methods

The previous section raises the question, what good choices for the subspaces $\mathcal{K}$ and $\mathcal{L}$ are. It turns out that Krylov subspaces are quite a good one.

Definition 2. A subspace $\mathcal{K}_m(A, v) \subset \mathbb{R}^n$

$$\mathcal{K}_m(A, v) := \text{span}\left\{v, Av, A^2v, \cdots, A^{m-1}v\right\}$$

is called a *Krylov subspace*. For brevity of notation, we will also write $\mathcal{K}_m$ in unambiguous situations.

We will call a projection method a *Krylov subspace method*, if $\mathcal{K} = \mathcal{K}_m(A, r_0)$. Many of theses methods, as the ones we will show later, start with a Krylov subspace of dimension one and increase the dimension with every step. So we obtain a sequence of approximations

$$x_0,\ x_1,\ x_2,\ \ldots x_{n'},$$

since we add an additional constraint in every step. We note that there exists an $n' \leq n$, so that $x_{n'}$ is the exact solution of the linear system (1) [1, Proposition 5.6 and proposition 6.1].

2.3 Conjugate Gradient Method

The Conjugate Gradient Method is a Krylov subspace method, which works for a symmetric and positive definit matrix A. We obtain the method by choosing

$$\mathcal{L} = \mathcal{K} = \mathcal{K}_m(A, r_0).$$

CG is optimal in the sense, that it minimizes the error in the A-norm

$$\|x - x_m\|_A = \min_{\tilde{x} \in x_0 + \mathcal{K}_m} \|x - \tilde{x}\|_A.$$

See [1, Proposition 5.2].

2.4 Generalized Minimal Residual Method

The other Krylov subspace method we want to present is the *Generalised Minimal Residual Method* (GMRES). In contrast to the CG method, GMRES can be used with any arbitrary square matrix A. This method chooses the space of constraints as follows.

$$\mathcal{L} = A\mathcal{K} = A\mathcal{K}_m(A, r_0)$$

GMRES is also optimal. It minimizes the residual in the 2-norm

$$\|b - Ax_m\|_2 = \min_{\tilde{x} \in x_0 + \mathcal{K}_m} \|b - A\tilde{x}\|_2.$$

See [1, Proposition 5.3].

2.5 Preconditioners

The main idea behind preconditioning is to manipulate the linear system in such a way, that the convergence of the iterative method is speed up. So instead of solving the system (1), we solve the linear system

$$M^{-1}Ax = M^{-1}b. \tag{3}$$

Ideally M could be chosen as $M = A$, so that $M^{-1}A = I$, which would result in a linear system which solved in one iteration step. But obviously we do not want to compute the inverse of A for preconditioning. So we are searching for a matrix M which is as close to A as possible, but where it is still easy to solve linear systems with M.

3 Hypre

The library Hypre, as mentioned above, is a library providing different parallel preconditioners and solvers. The preconditioners we are interested in are

- Euclid [7],
- ParaSails [6],
- Algebraic Multi Grid (AMG) [5]

and we will use the iterative solvers

- CG [1],
- GMRES [1],
- AMG .

Hypre also supports specialized preconditioners and solvers, but we will focus on the general ones.

Parallelization in Hypre is realized by the Message Passing Interface (MPI). The sparse matricies are stored in the Compressed Row Storage format (CRS).

4 Numerical Experiments

In this section we will show different results of several experiments we have carried out. First we will describe our test setup.

4.1 Setup

The following experiments were run on the *IBM Power6 575 Cluster JUMP* at the Jülich Forschungszentrum. A system with 14 SMP nodes where each node consists of 32 SMT processors (total 448). More details are given in table 1.

Processortype:	IBM Power6, 4.7 GHz
Overall peak performance:	8.4 Teraflops
Linpack:	5.4 Teraflops
Main memory:	14×128 Gbytes (aggregate 1.8 TB)
Networks:	InfiniBand (MPI communication)
	10 Gigabit Ethernet (I/O)
	1 Gigabit Ethernet (cluster management)
Operating system:	AIX 5.3

Table 1: Hardware specification of the Cluster JUMP.

We have several preconditioners in combination with different solvers and matricies. Therefore we calculated

$$b := A \cdot e, \qquad \text{where } e = (1, 1, 1, \ldots, 1)^T$$

and solved the linear system

$$Ax = b. \tag{4}$$

4.2 Matricies

To test the library we used different matricies from real applications.

Kurbel This matrix results from a finite element analysis of a Crankshaft.

Dimentsion	: 192858×192858
Nnz	: 12 226 189
Max. nnz per row	: 408
Avg. nnz per row	: 63.4
Max. diagonal value:	$2.64684e + 07$

BoneS10 This matrix also results from a finite element analysis. It is the analysis of a trabecular bone.

Dimentsion	: 914898×914898
Nnz	: 55 468 422
Max. nnz per row	: 81
Avg. nnz per row	: 60.6
Max. diagonal value:	18803.4

This matrix was obtained from the University of Florida Sparse Matrix Collection [2].

Reso_50x50x50 Calculating the eigenmodes of a box-shaped resonator cavity leads to this matrix.

Dimentsion	: 375000×375000
Nnz	: 6133288
Max. nnz per row	: 20
Avg. nnz per row	: 16.4
Max. diagonal value:	82.0511

4.3 Convergence

First of all, we want to analyse the convergence of the different algorithms. So we used different preconditioners and solvers to solve the linear system (4). The iteration should stop, when the relative residual $\hat{r}$ fulfills

$$\hat{r} \leq 10^{-9}$$

or after 1000 iterations.

Figure 2 shows the relative residual norm in each iteration of GMRES. Different preconditioners where used respectively.

If we were only looking at the total number of iterations then the Euclid preconditioner would be the best one followed by AMG and ParaSails. But when looking at the total runtime, we get a different picture. Table 2 and figure 3 show, that the preconditioner Euclid is very slow. Even though it yields good results in terms of quality, it is way to slow to be of any practical use.

As we can see in figure 4 the ParaSails preconditioner is the best choice in terms of runtime in this example. AMG as preconditioner is even a little bit slower than solving without preconditioning. So we want to look at a slightly more complicated example and see how AMG and ParaSails perform.

In figure 5 we observe that the convergence of GMRES with ParaSails as preconditioner is considerably slower than with AMG. So to get more accurate results it might be required to use AMG in certain situations. On the other hand it can also be the case, that ParaSails converges faster, than AMG, as one can see in figure 6.

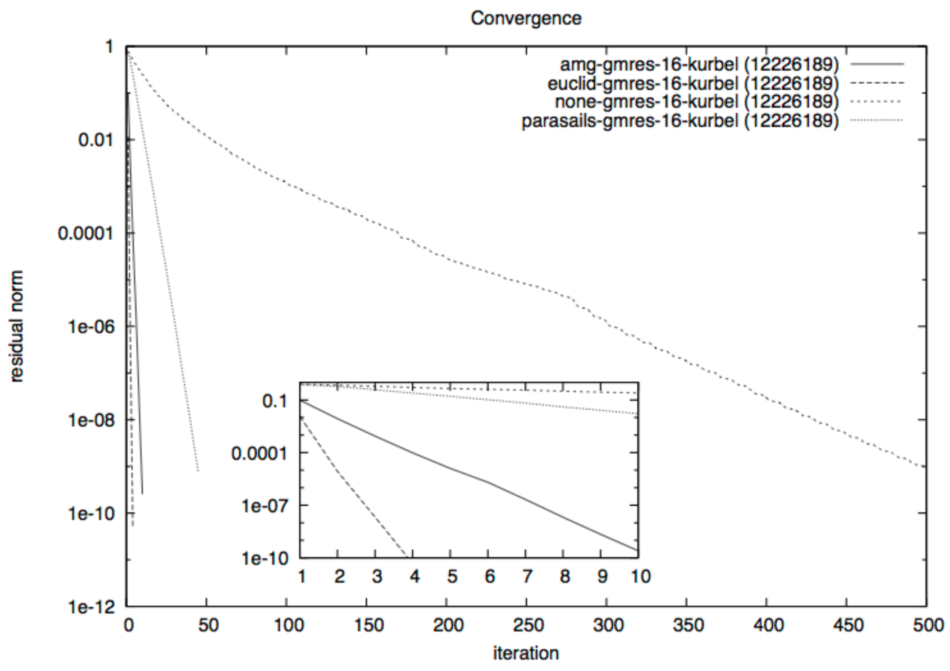


Figure 2: “Kurbel”: Convergence of GMRES with different preconditioners.

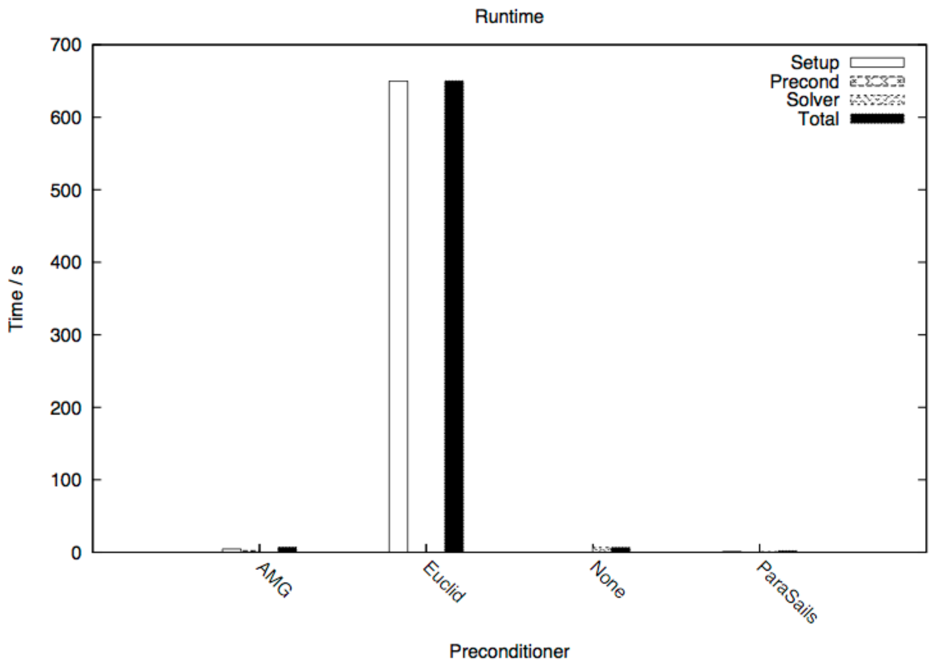


Figure 3: “Kurbel”: Runtime, GMRES with different Preconditioners.

Prec.	Solv.	Iter.	Abs. Err.	Rel. Err.	Time
AMG	CG	35	2.79625e-07	1.4499e-12	16.4836
AMG	GMRES	10	7.72657e-08	4.00635e-13	8.57119
Euclid	CG	6	6.56035e-08	3.40165e-13	649.483
Euclid	GMRES	4	2.65662e-08	1.3775e-13	650.575
None	AMG	690	NaNQ	NaNQ	131.35
None	CG	1000	13218.8	0.0685417	26.6161
None	GMRES	499	1.00843e-05	5.22887e-11	13.8277
ParaSails	CG	1000	65262.2	0.338395	29.2915
ParaSails	GMRES	45	2.87627e-07	1.49139e-12	2.5333

Table 2: “Kurbel”: Runtime.

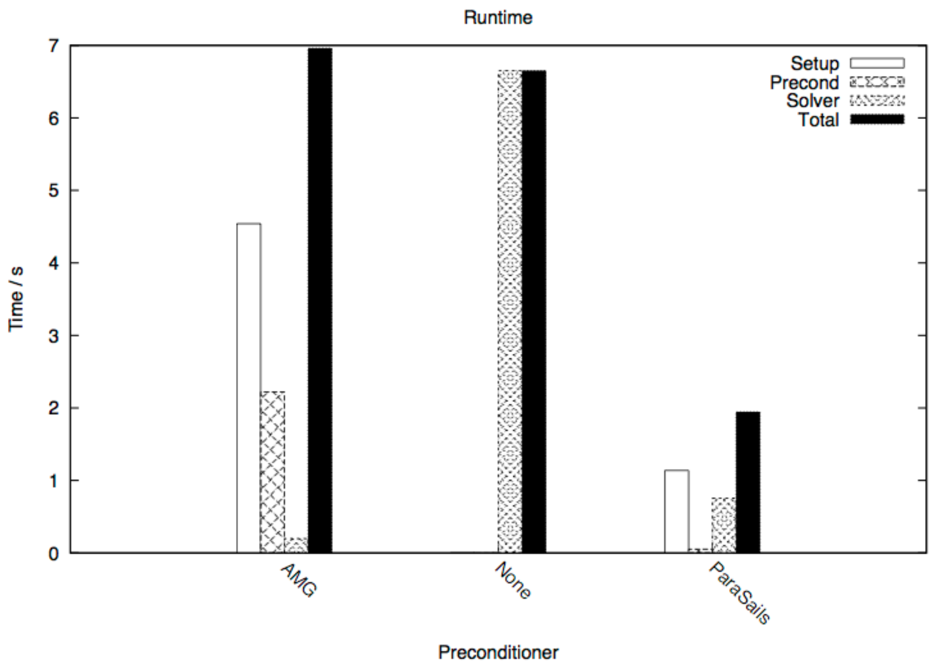


Figure 4: “Kurbel”: Runtime, GMRES with different Preconditioners.

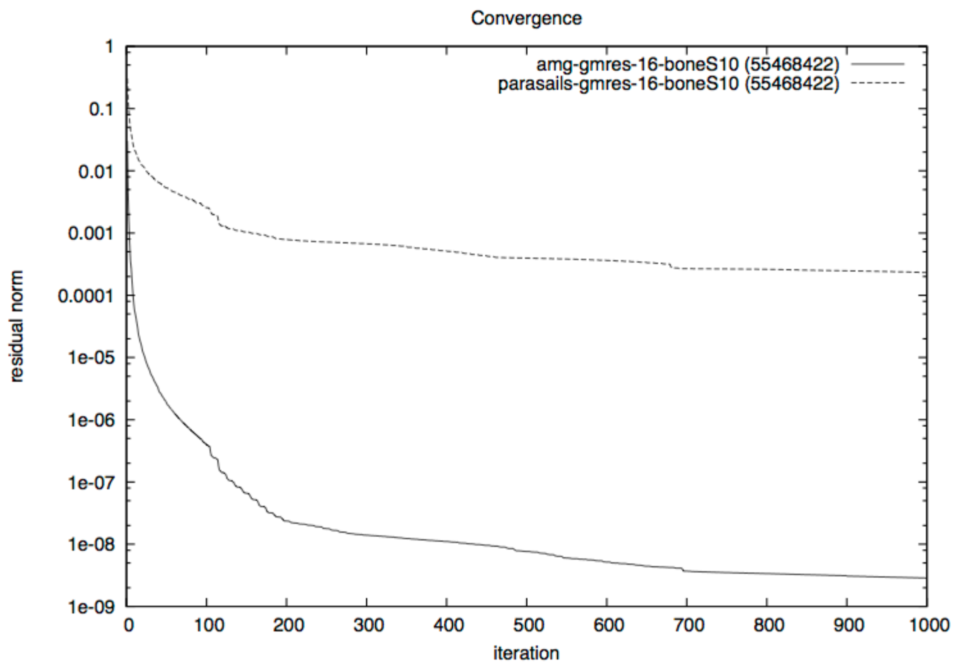


Figure 5: “BoneS10”: Convergence of GMRES with different preconditioners.

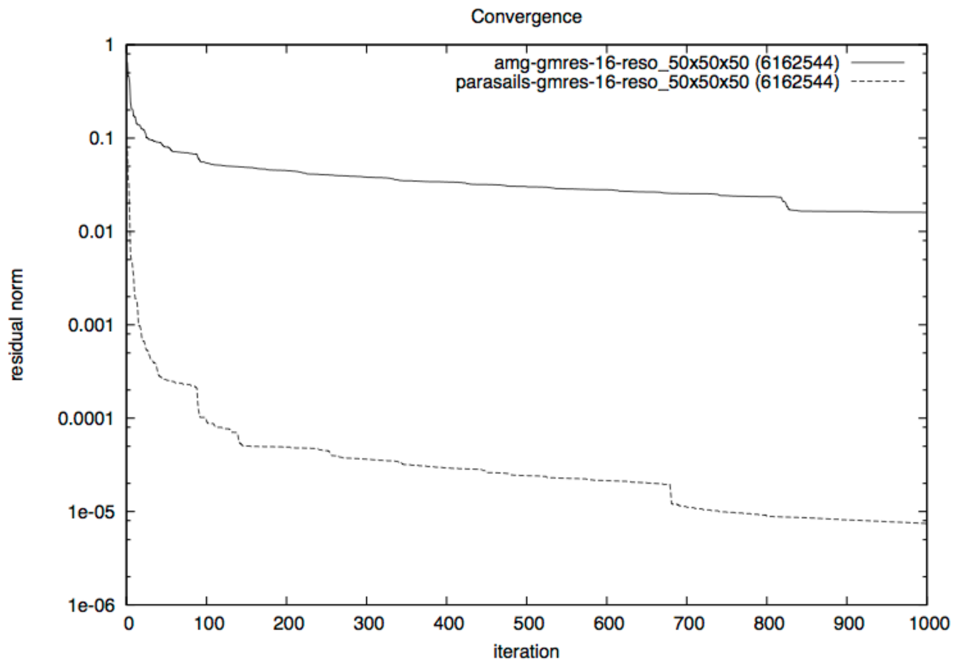


Figure 6: “Reso_50x50x50”: Convergence og GMRES with different preconditioners.

4.4 Scalability

One of the most important information for judging the practical usage of a parallel library, beside the fact that it works, is how well it scales. So in this section we will draw our attention to this aspect of Hypre.

In figure 7 we can see, that ParaSails scales nearly optimal up to 256 processors in terms of runtime. The memory usage scales quite well up to 32 processors, as we can see in figure 8. The memory usage gets higher when using more than 32 CPUs.

AMG does not scale so well as ParaSails does. There is no further benefit from using more than 128 processors, as one can see in figure 9. Also the total memory usage is quite high and memory scaling is not that efficient, as we can see in figure 10. We can conclude, that memory is more or less the critical point with this implementation of AMG. So with AMG as preconditioner we can only solve problems which fit well into memory.

One thing we should note about AMG is, that there exist certain parameters which decrease the memory usage of AMG. But these parameters also decrease the quality of preconditioning, so we did not used them. But if we are running out of memory, we maybe can still use AMG with some fine tuning of some parameters.

5 Conclusion

In conclusion we can say, that although Hypre does a good job, it does not provide the perfect preconditioner for all tasks. ParaSails scales very well, but it might not be able to solve the problem. So if our problem is "easy" enough, ParaSails is probably a better choice as far as scalability is concerned. On the other hand AMG scales not so well, but might be able to solve problems where ParaSails fails.

References

1. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelpha (2003)
2. The University of Florida Sparse Matrix Collection, `http://www.cise.ufl.edu/research/sparse/matrices/`
3. Hypre-2.0.0, `http://computation.llnl.gov/casc/linear\_solvers/`
4. Hypre-2.0.0 User's Manual (2006)
5. J. W. Ruge and K. Stüben, *Algebraic multigrid in: Multigrid methods*, Frontiers in Applied Mathematics 3,pp. 73–130, SIAM, Philadelphia (1987)
6. Edmond Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM Journal on Scientific Computing 21(5), pp. 1804–1822 (2000)
7. D. Hysom and A. Pothen, *A scalable parallel algorithm for incomplete factor preconditioning*, SIAM Journal on Scientific Computing 22(6), pp. 2194–2215 (2000)

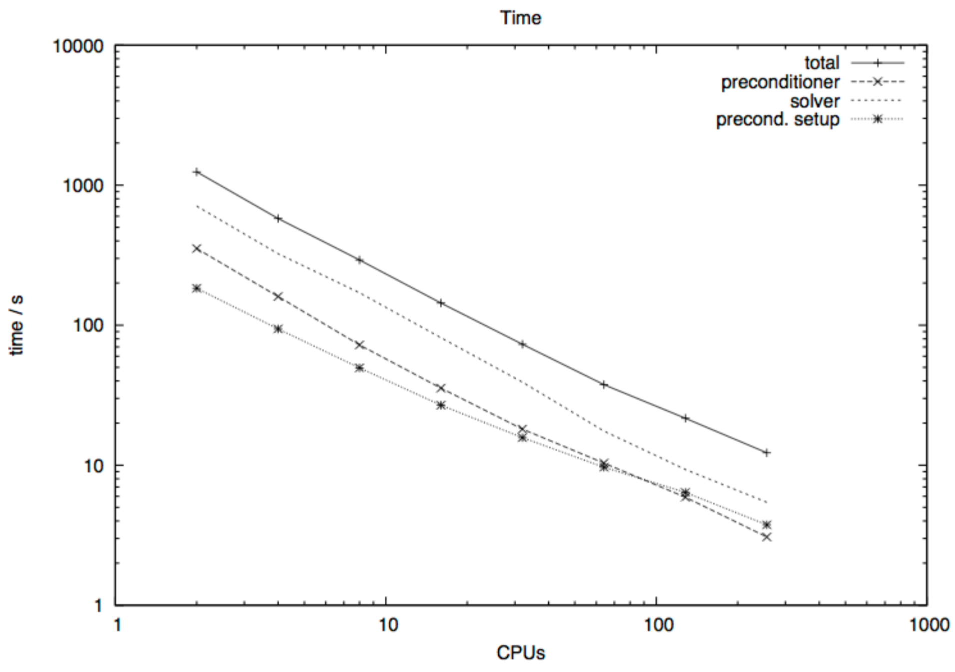


Figure 7: “BoneS10”: Runtime of GMRES with ParaSails.

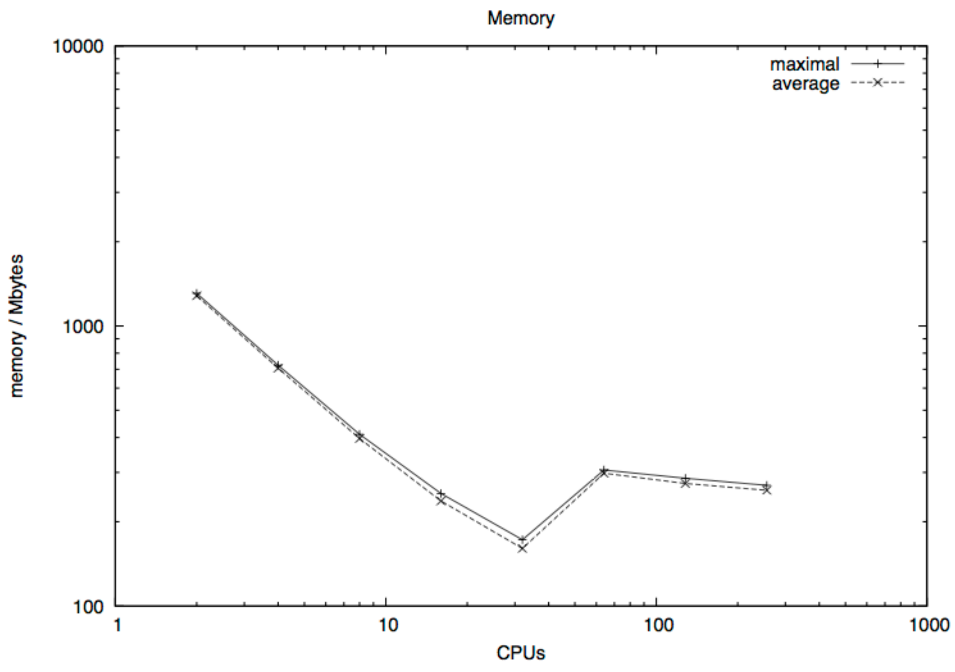


Figure 8: “BoneS10”: Memory usage of GMRES with ParaSails.

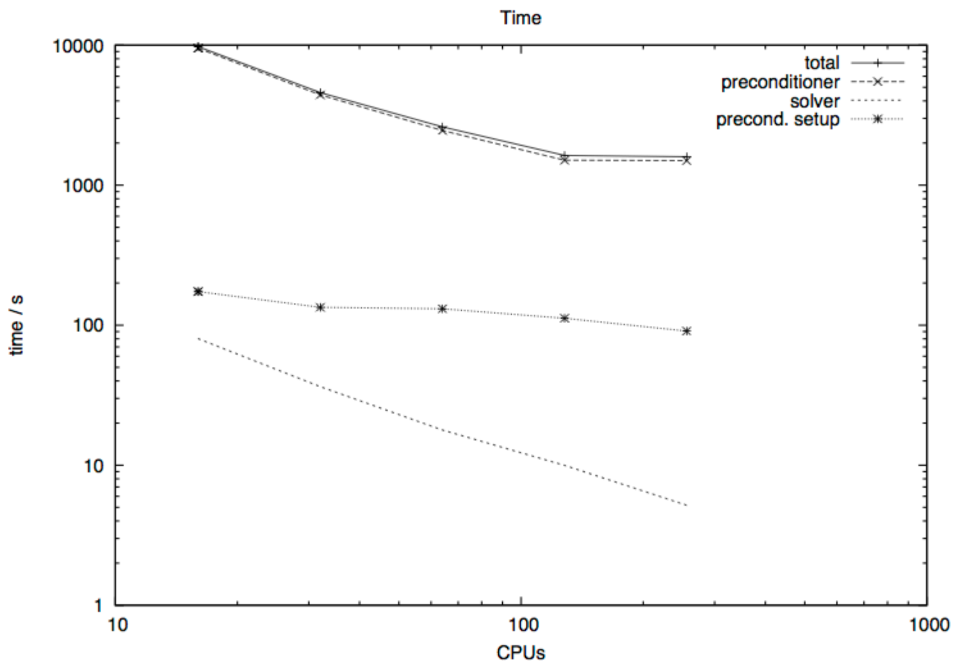


Figure 9: “BoneS10”: Runtime of GMRES with AMG.

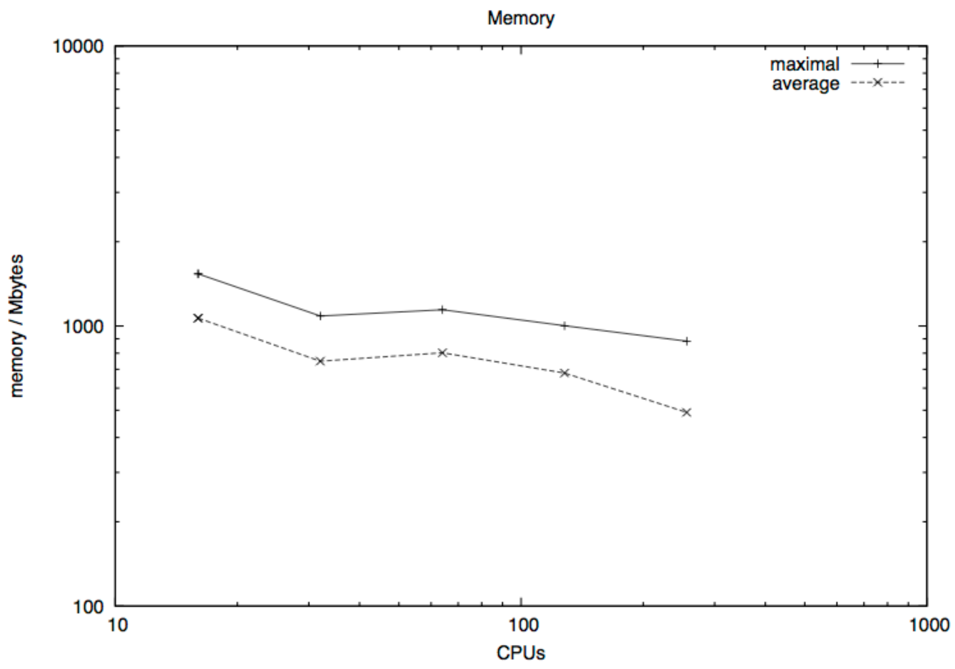


Figure 10: “BoneS10”: Memory usage of GMRES with AMG.

Simulation of NMR signal formation

Martin Rückl

E-mail: mrueckl@physik.uni-wuerzburg.de

Abstract: Constrictions of heart arteries called stenosis can be detected by analyzing the configuration of capillaries in myocardium using nuclear magnetic resonance (NMR). In this work a simulation model for the NMR signal formation was developed to check the influence of capillary configurations on the transversal relaxation processes. It has been found that faster diffusion strongly reinforces geometric influence.

1 Motivation

In today's medicine image formation allows very diversified diagnostics of diseases. Unfortunately detecting stenosis, a common reason for heart failure, with NMR-image formation is not that easy. A possible workaround would be to obtain information about the properties of capillaries in the myocardium, the hearts muscle tissue.
If some major arteries are blocked by stenosis, the capillaries in the tissue try to take over blood supply. This causes changes in areal distribution and radii of these capillaries. Because of the paramagnetic desoxyhemoglobin within a large fraction of this capillaries, inhomogeneities are induced to an external homogeneous magnetic field. This influence is called blood oxygen level dependency (BOLD). Those inhomogeneities furthermore have strong influence on transverse relaxation of NMR signals.
The goal of this work is to check whether and to what extend NMR signals allow conclusions about the tissue's capillary configuration. For that purpose the algorithm already developed in [1] was extended to simulate the signal formation for different tissue models.

2 Theory

2.1 Quantum mechanical background

Atomic nuclei with an odd number of nucleons posses a nuclear spin $\vec{I}$ which differs from zero. For the hydrogen nucleus, the one commonly used in medical NMR imaging, the spin's magnetic quantum number can only take the values $-1/2$ and $1/2$ and its two corresponding energy levels are degenerated. In addition the spin is associated with a magnetic dipole moment

$$\vec{\mu} = \gamma\hbar\vec{I} \;\text{ with }\; \gamma = \frac{g\mu_k}{\hbar} \text{ and } \mu_k = \frac{e\hbar}{2m_p} \tag{1}$$

here γ is the gyromagnetic ration, g is the dimensionless so called g-factor, μ_k is the nuclear magneton and m_p is the proton mass.
By applying a magnetic field $\vec{B}_0 = (0, 0, B_0)$, the coupling of $\vec{\mu}$ with the field splits up the degenerate levels. This is called nuclear Zeeman effect and can be described with the following Hamiltonian

$$\vec{H} = -\vec{\mu}\vec{B}_0 = -\gamma m B_0 \tag{2}$$

where m is the value of the magnetic quantum number. The energy eigenvalues and the energy difference between $m = -1/2$ and $m = 1/2$ for the Hamiltonian are then

$$E = -\hbar\gamma m B_0 \quad \text{and} \quad \Delta E = \hbar\gamma B_0 = \omega_L \hbar \tag{3}$$

This energy difference causes a discrepancy in the thermal occupation probability of the two eigenstates and, hence, induces an overpopulation ΔN of the energetically lower state. One can show that, at room temperature, this overpopulation is of the order of ppm. and grows approximately linear with B_0 [6].

2.2 Classical simplification

Even if quantum mechanics is the proper way to describe the physical problem of a single two-level system, one can move to a much simpler quasi-classical description of the magnetization. The huge amount of hydrogen nuclei in even a microscopic fraction of the smallest resolvable volume element (voxel) of a NMR scanner renders it possible to concatenate all of the magnetic moments contained in this fraction to a mesoscopic magnetization M. This mesoscopic magnetization has lost the quantum mechanic discretized properties. The discretized alignment of magnetic dipole moments can now be interpreted as a continuous magnetization vector which precesses around the magnetic field with frequency ω_L (fig. 1a). It is possible to manipulate this precession due to the application of a second oscillating field $\vec{B}_1(t) = (B_1 \sin\omega t, B_1 \cos\omega t, 0)$. If the oscillation frequency matches ω_L one can move from a static coordinate system to a rotating one in which both, M and B_1 stand still. In this coordinate system now the same effect like in the static one without B_2 takes place: M starts precessing around $\vec{B}_0 + \vec{B}_1 = (B_1, 0, B_0)$. In the static coordinate system M rotates towards the transversal plane (fig. 1b). This is then called a 90°-pulse.

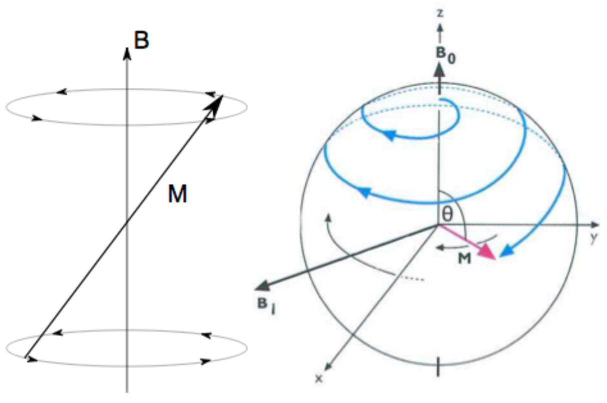


Figure 1: (a) Precession of the magnetization around the magnetic field. (b) 90°-pulse tilting the rotation into the transversal plane[2].

By longer pulse durations one can also achieve 180°-pulses, which results in a reversal of the precession direction. The amplitude of the projection of $\vec{M}(t)$ to the transversal plane is a signal this work is all about.

2.3 Signal decay and spin-echo

If the magnetic field would be totally homogeneous in the whole voxel the signal would not decay at all. But because of inhomogeneities each concatenated subset of particles has its own frequency ω_L and the precession of the mesoscopic M starts to dephase. If there is no further excitation of the system this is called Free induction decay (FID).
In a static case, applying another 180°-pulse at time $t = T_E/2$ now flips the rotation directions and the signal will be totally rephased at $t = T_E$. In the other case, when there is diffusion every spin experiences a particular history of different rotation speeds on its way which is not reversible, hence they cannot completely rephase. The maximum of the regained signal is called echo. It is obvious that this process can be repeated several times in a row what leads to a multiple spin-echo sequence (MSE).

3 Simulation approach

Because of the parallel alignment of capillaries in the hearts tissue the first approximation is to reduce the simulation to two dimensions. Assuming further capillaries of infinite length (or $l \gg r$) the frequency inhomogeneity of a single capillary according to [7] is

$$\delta\omega\,(r,\phi) = \delta\omega_0 R_0^2 \frac{\cos 2\varphi}{r^2} \quad \text{with} \quad \delta\omega_0 = \frac{\Delta\chi}{2} B_0 \sin^2\Theta \tag{4}$$

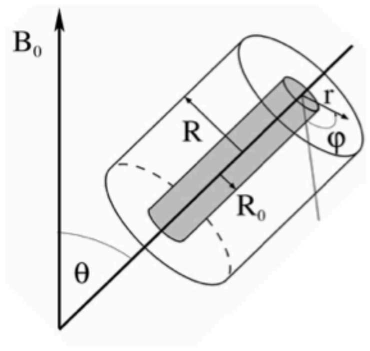


Figure 2: Coordinate system. The capillary and the surrounding tissue are tilted towards B_0.[7].

The next step is to replace the diffusion by a random walk. Referring to [1] and [8] this can be done by choosing step lengths to be Gaussian distributed:

$$P(\vec{r} \rightarrow \vec{r} + \Delta\vec{r}) = \frac{1}{4\pi D\Delta t} \exp\left[\frac{\Delta\vec{r}^2}{4D\Delta t}\right] \tag{5}$$

where Δt is the size of a single time step. To avoid errors at the boundary conditions it is important to choose the time steps so small, that the mean step size $\sigma = \sqrt{2D\Delta t}$ of the random walk is at least about one magnitude below the capillary diameter.

3.1 Signal formation

Once the random walk has generated a trajectory in the simulation environment, the history of experienced fields $\omega(t)$ is known. Now the overall phase $\Phi(t) = \int_0^t \omega(t)dt$ can be accumulated. For a large amount of sampled of trajectories N_T building the mean value $\langle\cos(\Phi(t))\rangle(t)$ for every time step produces a signal according to the macroscopic magnetization $M(t)$ which is rotating in the transversal plane.

$$M(t) = \left\langle \cos\left(\int_0^t \omega(t')dt'\right)\right\rangle \tag{6}$$

For simulating spin echo sequences it is now sufficient to just adapt the phase accumulation routine to the desired sequence. For a periodic MSE sequence for example one needs to switch the sign of $\omega(t)$ with each $180\frac{1}{2}$o-pulse, or just split the integral $\int_0^t \omega(t')dt'$ to a sum of several sub-integrals like $\int_0^{1/2T_E} \omega(t')dt' - \int_{1/2T_E}^{3/2T_E} \omega(t')dt' + \int_{3/2T_E}^{5/2T_E} \omega(t')dt' - ...$

3.2 Field discretisation

To avoid long runtimes of the simulation on the one hand, and to allow a large simulation environment with lots of capillaries on the other, for complex capillary distributions the inhomogeneous field was calculated in advance. Therefore the random walk environment gets divided into a square grid and between the grid the values are obtained by linear interpolation. Important for this method is to chose the resolution for the grid large enough. Otherwise the singularities of the field in the inner of capillaries will even introduce larger interpolation errors in the diffusion area.
This also allows more realistic simulations with cyclic boundary conditions: one can build a lattice of capillaries with the environments configuration as elementary cell, then calculate the field induced by this lattice only for a small central area and run the random walk in there with cyclic boundaries.Finally this approach even resolves the problem of discontinuities at boundary crossings discussed in [1].

3.3 Error checks

The functionality of the implemented algorithm has been extensively checked:

- Does the field discretisation affect FID-signals? How large does the resolution have to be?
  No deviations larger than statistical errors were found for discrete fields with a distance of interpolation points one order of magnitude below the smallest capillary diameter.

- Do numerical errors affect the calculations in a reasonable amount?
  Even for large N_T (number of sampled trajectories) statistical errors shrunk with $\sqrt{N_T}$ like expected. This leads to the conclusion that in the scope of the performed simulations numerical errors can be neglected.

4 Results

Fig. 3a - 3c shows the three most intensely analyzed geometries. For all of them the volume fractions

$$\eta = \frac{A_{\text{Capillary}}}{A_{\text{Environment}}} \tag{7}$$

0.05, 0.10 and 0.15 where simulated. The different η were achieved by constant environment size, so the radii of capillaries were $R_c = 10\sqrt{\eta}$ for all geometries. For the simulations of Kroghs Model reflective boundaries were used, the square shaped environments had cyclic boundary conditions. The difference between fig. 3b and fig. 3c is not only the orientation of the capillary, in fig. 3c also a periodic lattice consisting of approx. 120 000 single surrounding capillaries was used to calculate the field. The commonly used diffusion constants were in the range of 0 to 3, $\delta\omega_0$ was set to 1 for all simulations. The size of time steps and hence the mean step size of the random walk always were far beyond limits mentioned before. The statistical errors for all signals are of order of 0.003.

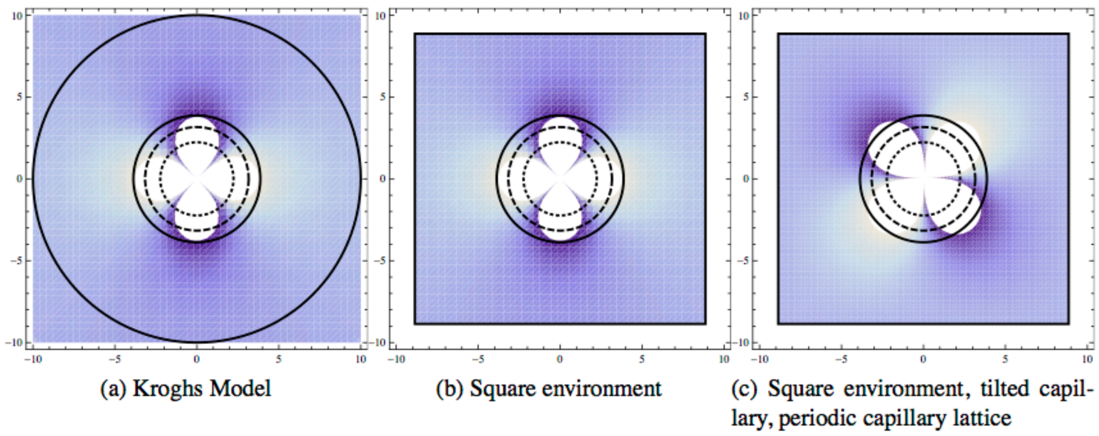


Figure 3: Most used simulation environments.

4.1 Free induction decay

Fig. 4a and fig. 4b show the results for the most simple realized simulations. It can easily be seen that for smaller η the dephasing process is much slower. This directly results from the lower average inhomogeneities (eqn. (4)).

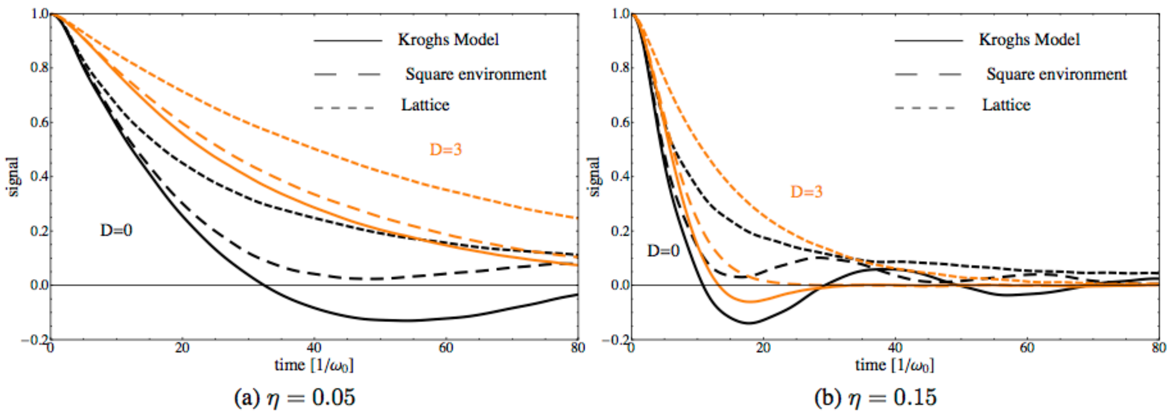
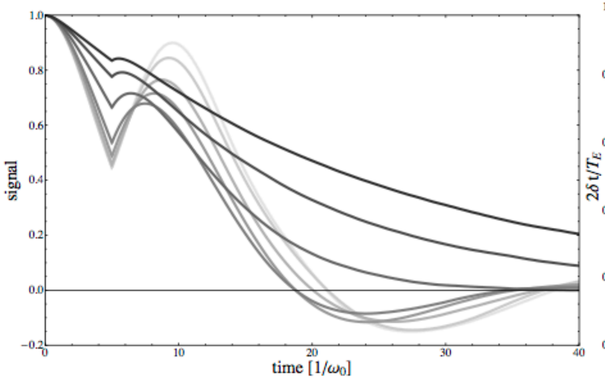


Figure 4: Signal of free induction decay for different η and D. ($N_T = 50\,000$, $\Delta t = 0.008$)

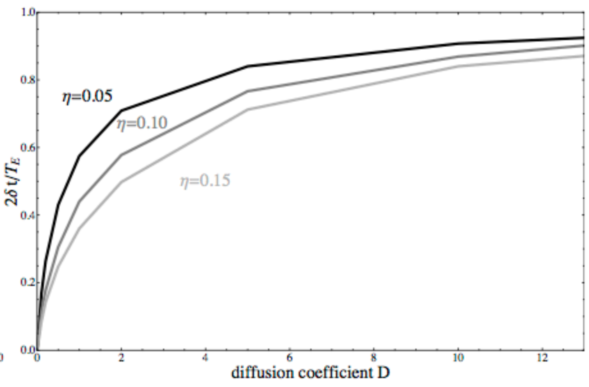
The algorithm was tested with the simulations for Kroghs Model, because for $D = 0$ analytical solutions exist [8]. It turned out, that the statistical errors shrunk with $\sqrt{N_T}$ up to $N_T > 2\,000\,000$ like expected, whereas for simulations with different Δt and constant N_T no influence on the deviation to theoretical solutions were found. This leads to the conclusion that the statistical error widely exceeds influence of the time steps' size.

4.2 Spin-Echo: Timeshift

With increasing D, a shift of the echoes positions $\delta t = T_E - t_{measured}$ towards lower times than expected has been found (fig. 5a). This can also be explained theoretically, by considering a further dephasing caused by the diffusion even after the 180° rephasing pulse. For $D = 0$ of course no shift will occur.



(a) Time shift of echoes for increasing diffusion constants($D = 0.1$ to $D = 15$) from light to dark. ($\eta = 0.15$)



(b) Dependency of shift $2\delta t$ of diffusion, normed to T_E

Figure 5: Single spin-echo simulation of Kroghs Model. ($T_E = 10$)

For Kroghs Model this effect has been reviewed in single spin echo simulations for $\eta_1 = 0.05$, $\eta_2 = 0.10$ and $\eta_3 = 0.15$, $\Delta t = 0.004$ and different diffusion constants ($D_1 = 0.1, D_2 = 0.2, D_3 = 0.5, D_4 = 1.0$, $D_5 = 2.0$, $D_6 = 5.0$, $D_7 = 10.0$,$D_8 = 15.0$) to get a qualitative overview of this process (fig. 5a). As one easily can see, for all curves the signal at the expected echo position $t = 10$ is no local maximum. From fig. 5b, one can also see that for lower η the diffusion has a larger effect on the dephasing because here the shift of the echoes grows faster with D. Comparing this with numerical calculations for low diffusion rates form [5] (eqn. (35)) at least qualitative accordance is shown: δt decreases with rising η but also shows some $D^{1/3}$ dependency which tolerably fits to fig. 5b for small D. For $D \to \infty$ of course δt nears 1 asymptotically.

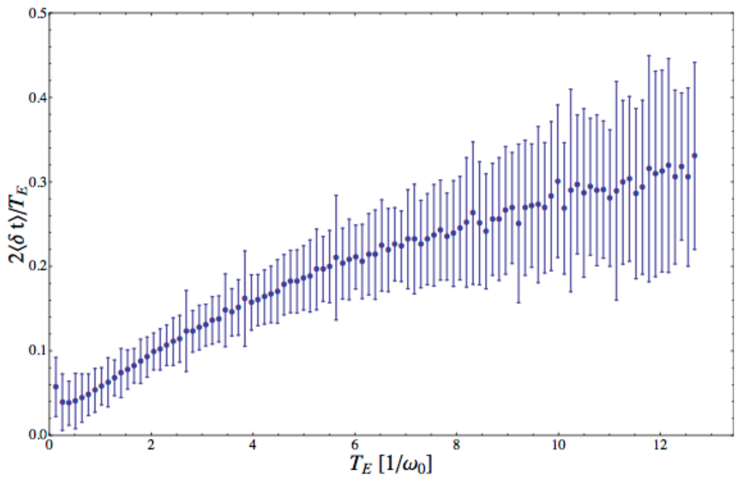


Figure 6: Dependency of δt on T_E, the error bars show the standard deviation of the mean values.

For a higher resolved MSE simulation it was also possible to get detailed information about the dependency of δt on echo times T_E. Therefore the δt of the MSE signal for each echo number n was measured and the mean value $\langle \delta t(n) \rangle$ over all echoes was calculated (fig. 6). For $T_E \to \infty$, δt in fact also should near 1 asymptotically, this doesn't match to [5] (eqn. (35)) at all.

4.3 Multiple spin-echo sequences

For MSE sequences also the three different geometries were simulated. For each simulation 50 000 trajectories were sampled. The total simulation duration ($t_{max} = 320$) got sampled with 20 000 time steps which leads to $\Delta t = 0.016$. The echo period T_E was increased from 0 to 17.6 in 1100 equally spaced steps.
Fig. 7a and 7b show some of the obtained signals.

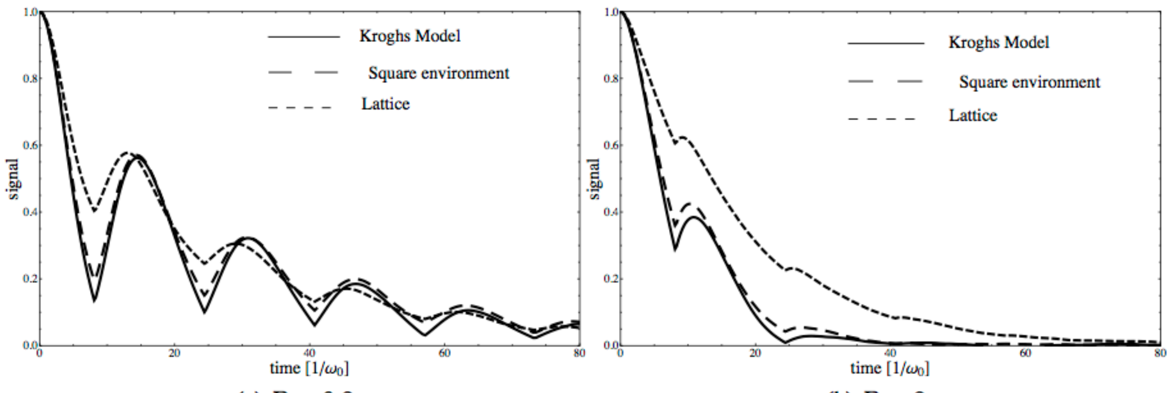


Figure 7: MSE signals for the three different geometries. $\eta = 0.15, T_E = 16.256$ ($N_T = 50000$)

One can simply see, that for $D = 0.3$, the decay of the signals echoes is very similar for all geometries, whereas for $D = 3$ the decay of the capillary lattice simulation shows large deviations to the other ones. Basically, fig. 7a and fig. 7b show the generic behavior of MSE signals. This behavior suggests a more intense analysis of the strongest characteristic of such signals: the decay of the echoes amplitudes. Therefore some different post processing methods come to mind:

- analysis of the decay of really "measured" amplitudes (further referred as "real amplitudes")
- analysis of the decay of signal at the expected echo position (further referred as "expected amplitudes")
- analysis of the decay of echoes amplitudes by echo number
- analysis of the decay of echoes amplitudes by time

For all methods a simple exponential decay of the specific signal S

$$S(t) = Exp\left[\gamma t\right] \quad \text{and} \quad S(n) = Exp\left[\gamma' n\right] \tag{8}$$

was assumed and the negative decay coefficient γ for the different echo periods T_E was determined with *Mathematica*'s *FindFit* function. Fig. 8a and fig. 8b show $\gamma'(T_E)$ and $\gamma(T_E)$. For large echo times both decay coefficients show a similar linear decreasing behavior, whereas for small T_E, γ' has a more complex characteristic. It can also be seen that for the chosen diffusion rate $D = 0.3$ and $\eta = 0.05$ only very small deviations between the real and the expected amplitudes occur.

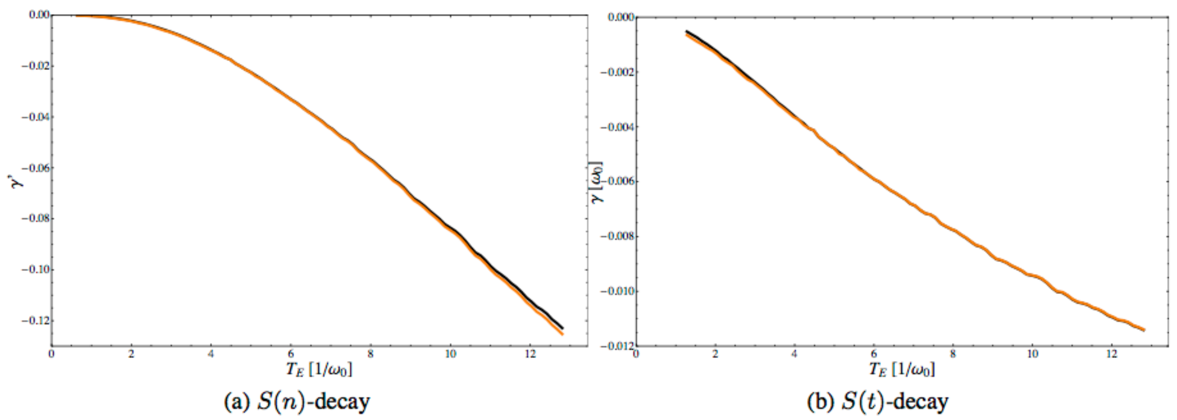


Figure 8: Decay coefficients γ and γ' behavior for different T_E. Black: "real amplitudes", Orange: "expected amplitudes" ($N_T = 50\,000$, $\Delta t = 0.0032$, $D = 0.3$, $\eta = 0.05$, same data set like fig. 6).

To perform also a more quantitative check of fitness of decay according eqn. 8, the root mean square for the deviations of the signal S to the fitted curve was calculated. As it can be seen in fig. 4.3 for real amplitudes decaying with echo number n best fit is obtained.

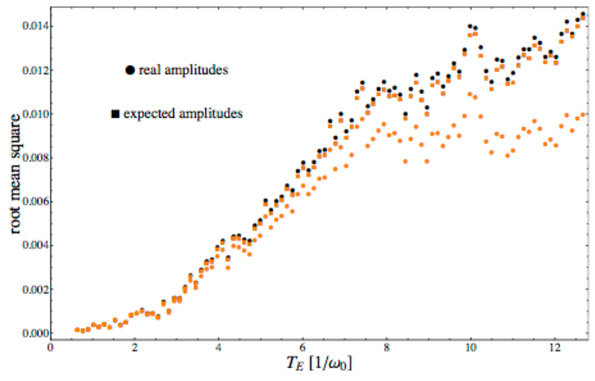
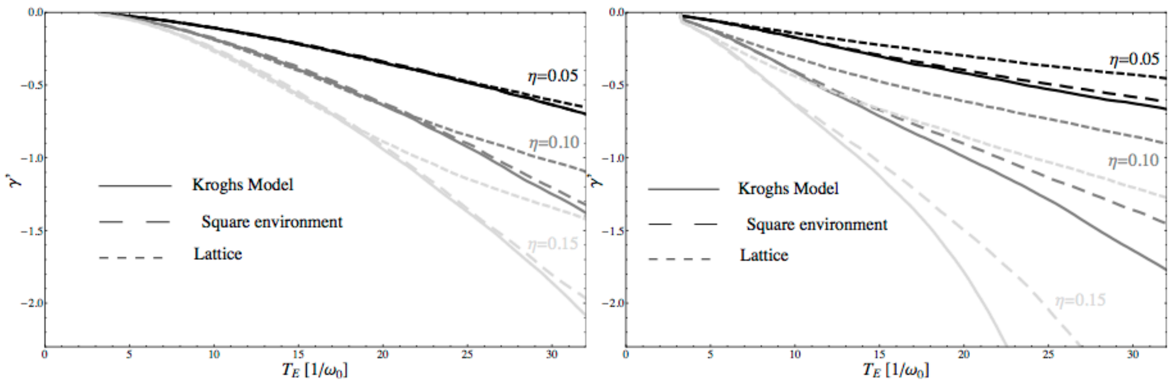


Figure 9: Root mean square of deviation between simulated signal S and its exp. fit. Orange: $S(n)$ Black: $S(t)$. For "expected amplitudes" naturally $S(n)$ and $S(t)$ have same deviations.

Even if fig. 4.3 reflects only a slight difference between the different fit methods, because of the influence of diffusion, volume fraction and T_E on the δt, in the following always the decay of "real amplitudes" according to the echo number is analyzed.
Fig. 10a and 10b shows the characteristics of $\gamma'(T_E)$ for the three focused geometries and different parameter combinations.

From fig. 10a and 10b it is obvious that a faster diffusion highly increases the influence of the environments geometry. But also some time dependent effect for the capillary lattice simulation can be seen: whereas for Kroghs Model and the simple square environment behavior is pretty similar up to high echo times, the deviation to the lattice simulation grows fast. This can be understood by recognizing that for larger T_E, diffusion in the lattice model allows a particle also to be affected by more distant capillaries.
The general conclusion of the MSE simulations performed for these more simple geometries is, that at least for large echo times significant differences in the echo decays can be observed, further increasing diffusion and volume fraction also makes the echoes decay rates more distinguishable.

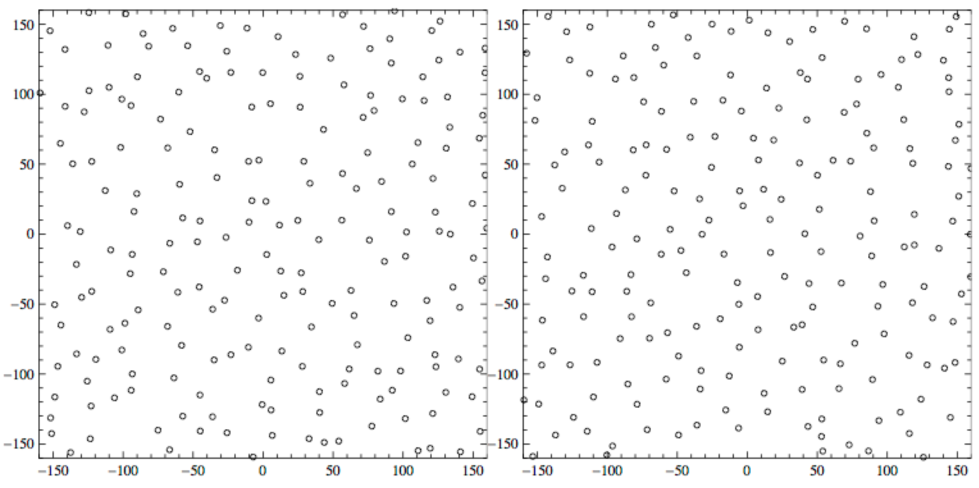


(a) $D = 0.3$ (b) $D = 3$

Figure 10: Decay coefficients (γ') dependency to echo period T_E for different D and η. ($N_T = 50\,000$, $\Delta t = 0.016$)

4.4 Realistic capillary distributions

Beside the analysis of more or less generic geometries discussed until now, also more interesting capillary distributions were simulated. For that purpose capillary distributions according to a two dimensional one component plasma, proposed by [4] and sampled by [3], were simulated. For those distributions Γ is a characteristic parameter, that changes regarding real cardiovascular tissues "health status". To check the influence of parameter Γ on the signal formation of FID and MSE signals, simulations for the distributions showed in fig. 11a and 11b were performed. Γ was chosen taking into account the results from [4].



(a) "healthy", $\Gamma = 3.40$ (b) "sick", $\Gamma = 3.07$

Figure 11: More complex simulated capillary distributions consisting of 200 capillaries. The alignment of the capillaries is realized like proposed in [4] and was generated by calculations form [3]. The size of area is 320^2.

Those distributions were regarded as elementary cells for a periodic square lattice. For the field calculation all capillaries in the range of 300 elementary cells were taken into account to calculate the field within the central cell. The resolution chosen was 4. For different capillary radii, one can now just scale the field values with R_C^2.

Simple FID simulations were performed for uniform capillary radii $R_{C,1} = 1$, $R_{C,2} = 2$ and $R_{C,3} = 3$ and different diffusion constants in the range of 0 to 3. This different capillary radii lead to overall volume fractions $\eta_i = 200R_{C,i}^2\pi/320^2$. Unfortunately for none of the combinations of η and D significant deviations between the signals for both distributions were found.
For MSE simulations only simulations with $R_C = 3$ and $D = 1$ were done. These signals showed deviations of the decay coefficient γ' only for larger echo periods (fig. 12). But even there they are small.

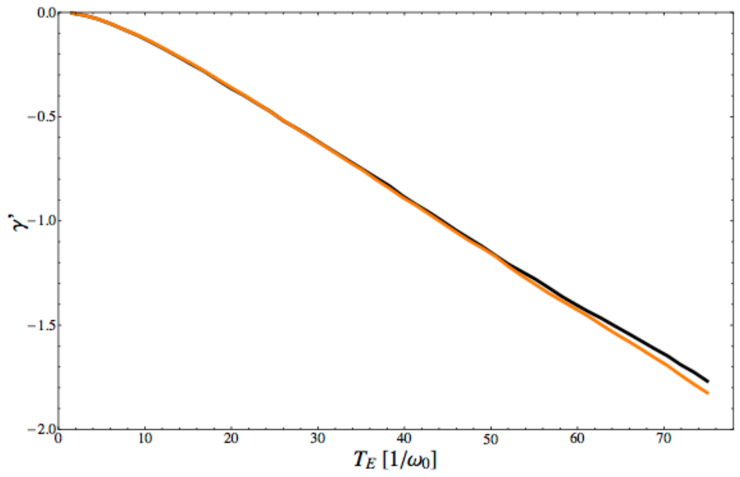


Figure 12: Decay coefficient γ' for capillary distributions shown in 11a and 11b.$D = 1$, $R_C = 3$, $\eta = 0.0552$, $\Delta t = 0.015$, $N_T = 100\,000$

5 Conclusion and outlook

Overall the timeshifts of the echo signals have been examined in a very detailed way and it turned out that simply assuming the echo signal at time T_E might introduce errors. In addition it has been shown that the geometrical properties of capillaries can have large impact to the MSE decay coefficients. Therefore, the original idea of detecting stenosis via the change it affects to cardiovascular tissue surely is justified. Although the simulations for the two complex distributions didn't show any applicable result, this will probably change with a better adaption of the geometry to real tissue. Particularly considering the different capillary densities and deviations of radii for different kinds of cardiomyopathy found in [4] promise some more significant results.
In addition during performing the simulations a large and easy to extend and adapt parallel simulation framework has been developed. This would e.g. allow to do further simulations with arbitrary shaped 2D blood vessels or additional diffusion conditions.

6 Acknowledgment

My greatest gratitude goes to Dr. W. Nadler for having patience with my questions and good advices in many situations. In addition I want to thank R. Speck and B. Berberich for the great organization of the guest student program. Also I'd like to thank all my fellow guest students for the nice atmosphere and creative advices. Finally thanks go to T. Kampf in Wuerzburg for providing data on realistic capillary distributions.

References

1. N. Fricke. Nmr signal formation in capillary networks. In Matthias Bolten, editor, *Ergebnisse des Gaststudentenprogramms 2008*. 2008.
2. Philips Healthcare, editor. *Basic principles of MR imaging*. Philips Healthcare, June 2008.
3. T. Kampf. Capillary distributions, simulated 2d one component plasma. private communications, September 2009.
4. R. Karch, F. Neumann, R. Ullrich, J. Neumueller, B. K. Podesser, M. Neumann, and W. Schreiner. The spatial pattern of coronary capillaries in patients with dilated, ischemic, or inflammatory cardiomyopathy. *Cardiovascular Pathology*, 14:135–144, 2005.
5. V.G. Kiselev and S. Posse. Analytical model of susceptibility-induced mr signal dephasing: Effect of diffusion in a microvascular network. *Magnetic Resonance in Medicine*, 41:499–509, 1999.
6. H. Mueller. Zuslassungsarbeit - magnetische kernresonanz. www.physik.uni-wuerzburg.de, February 1980.
7. C.H. Ziener, T. Kampf, V. Herold, P.M. Jakob, W.R. Bauer, and W. Nadler. Frequency autocorrelation function of stochastically fluctuating fields caused by specific magnetic field inhomogeneities. *The journal of chemical physics*, 129, 2008.
8. C.H. Ziener, T. Kampf, G. Melkus, V. Herold, T. Weber, G. Reents, P.M. Jakob, and W.R. Bauer. Frequency autocorrelation function of stochastically fluctuating fields caused by specific magnetic field inhomogeneities. *Phys. Rev. E*, 76, 2007.

Domain-Force-Decomposition for Load-Balancing Molecular Dynamics

Theodros Zelleke

Faculty of Chemistry and Biochemistry
Department of Theoretical Chemistry
Ruhr-Universität Bochum
Universitätsstraße 150, 44801 Bochum

E-mail: theodoros.zelleke@theochem.ruhr-uni-bochum.de

Abstract: A new parallel algorithm for classical Molecular Dynamics is presented. The algorithm is an extension to a common domain decomposition algorithm and is suitable for simulation systems with short range forces and an inhomogenous distribution of the particles and for low density systems. The implementation is presented in the context of the combined Molecular Dynamics/ Multiparticle Collision Dynamics program *MP2C*.

1 Motivation

Computer simulation methods play an evermore important role in the physical sciences and two important fields are Molecular Dynamics (MD) and Computational Fluid Dynamics (CFD)[1]. Whereas MD describes molecular systems on the atomic level CFD is intended to describe hydrodynamic effects in liquids on a mesoscopic scale. A combination of methods from both fields has recently been implemented for a parallel environment in the program *MP2C*[2]. This program allows to simulate molecules dissolved in a liquid and explicitly take into account the hydrodynamic effects at relatively low computational cost. However it became apparent that the program shows non-optimal scaling properties especially for systems with inhomogenous distribution of solute particles. The reason for this behaviour could be traced to inefficient load-balancing in the MD part of the program. Therefore the goal set for this project was to implement a parallelization scheme with a better load-balance in the MD part of the *MP2C*-code.

2 Introduction

2.1 Molecular Dynamics

Classical Molecular Dynamics treats each of the N particles in a simulation system as a point mass and numerically integrates Newton's equations of motion to compute a trajectory for the system. These equations are given by

$$m_i \frac{d\mathbf{v}_i}{dt} = \sum_j \mathbf{F}_2(\mathbf{r}_i, \mathbf{r}_j) + \sum_j \sum_k \mathbf{F}_3(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \ldots \tag{1}$$

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i \tag{2}$$

where m_i is the mass of particle i, $\mathbf{r}_i$ and $\mathbf{v}_i$ are its position and velocity vectors. $\mathbf{F}_2$ is a force function describing the pairwise interactions between the particles while the further terms on the left side of equation 1 describe additional many-body interactions. The force terms are derivatives of potential energy functions which altogether describe the complete physics of the simulation model. In practice the left side of equation 1 is confined to only a few terms and the corresponding potential energy functions are constructed so as to include many-body and quantum effects. A given potential can be classified to be either long-range or short-range in nature. An important characteristic is the fact that short-range potentials are limited in range, which means that each particle interacts only with other particles that are geometrically nearby. Therefore the computational effort in calculating the forces for short-range potentials scales as N, the number of particles[3, 4]. Since particles can undergo large displacements, both in total as well as relative to other particles, this $O(N)$ scaling can only be implemented in connection with an efficient scheme to continually track the neighbors of each particle. In the *MP2C* program all interactions are of short-range nature.

The evaluation of particle interaction, i.e. force evaluation, is in general the most time comsuming part in a MD simulation[3]. A useful construct to illustrate the computational work in force evaluation is the $N \times N$ force matrix $\mathbf{F}$, where the $\mathbf{F}_{ij}$ element gives the force on particle i due to particle j. In case of short-range forces many entries in $\mathbf{F}$ are near to zero. This quality can be used advantageously to significantly reduce the computational cost by introducing a cut-off distance r_c beyond which mutual interactions between particles are neglected. Furthermore $\mathbf{F}$ is skew-symmetric due to Newton's third law $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$.

$$\begin{array}{ccc} & & \left(\begin{array}{cccc} x_1 & x_2 & \ldots & x_N \end{array}\right) \\ & & \begin{array}{cccc} \downarrow & \downarrow & \quad & \downarrow \end{array} \\ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} & \begin{array}{c} \longrightarrow \\ \longrightarrow \\ \\ \longrightarrow \end{array} & \begin{pmatrix} 0 & \mathbf{F}_{12} & \ldots & \mathbf{F}_{1N} \\ \mathbf{F}_{21} & 0 & & \\ \vdots & & \ddots & \\ \mathbf{F}_{N1} & & & 0 \end{pmatrix} \end{array}$$

Figure 1: Concept of the force matrix $\mathbf{F}$. Its entries $\mathbf{F}_{ij}$ represent the force vector acting on particle i due to particle j. x_1 to x_N are the position vectors of particles i to N. The overall force acting on particle i is the sum of all entries in the ith row. Per se its calculation requires knnowledge of x_1 to x_N.

Two different techniques have been reported to speed up force evaluation in MD simulations. The first technique implements *Verlet*-neighbor lists, where at a given timestep for each particle a list of nearby particles is maintained. When a list is formed, all neighboring atoms within a distance $r_s = r_c + \delta$ are stored. When calculating the overall force on a given atom, only the entries in the corresponding neighbor list are tested for possible interactions. A second technique commonly used is known as the linked-cell method. At every timestep all particles in the system are binned into cubic cells. The sidelength L_c of a cell is chosen such that $L_c \geq r_c$. This limits the search for possible interaction partners of a given particle to the bin it is in and the 26 surrounding ones. When appying Newton's third law force evalution can be further enhanced by searching only in half the surrounding bins.

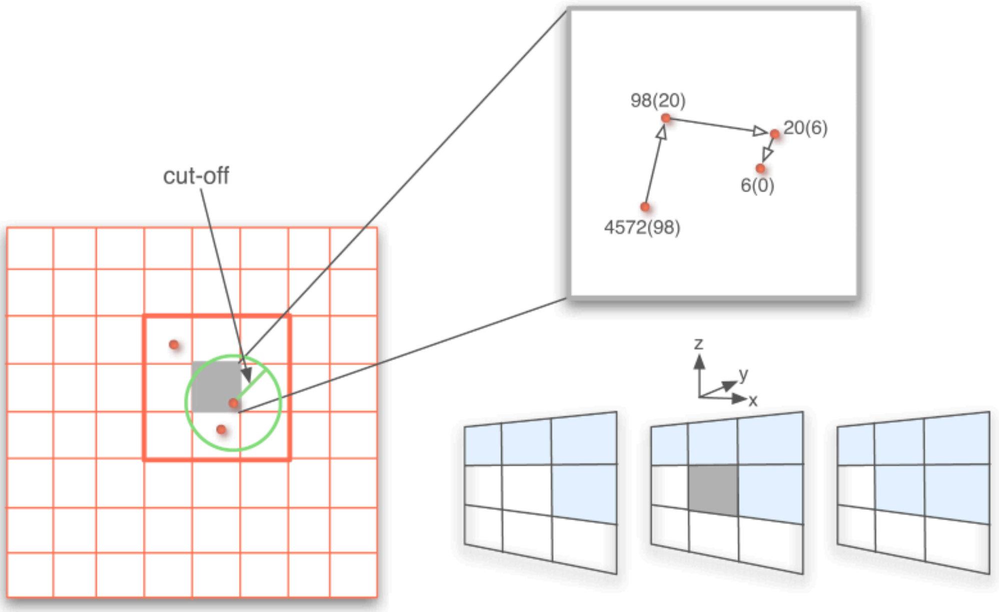


Figure 2: Principle of the linked cell method to speed up force evaluation in MD simulations. Particles belonging to the same cell are linked together in linked list data structure. In three dimension mutual interactions between particles are calculated only once when the search for possible interaction partners of a particle is confined to half of the 26 surrounding bins.

2.2 Multi Particle Collision

The hydrodynamic effects to be modelled arise from momentum exchange between the particles of a liquid. In the MPC method the particles of a liquid are represented by coarse-grained pseudo-particles which carry the hydrodynamic information. Only few parameters like particle density, a scattering angle and the mean free path of a particle are used to describe the physical properties of the liquid. The momentum exchange occurs after collisions of the particles on the condition of conversation of energy and conversation of momentum. Since the microscopic details of the collisions are of no interest they are modeled by a stochastic momentum exchange. In every timestep, which is called a collision step in MPC, the particles are sorted in so called collision cells. For every collision cell a random rotation axis and the center-of-mass velocity of all particles in a collision cell are calculated. Then for every particle its relative velocity to the center-of-mass velocity is calculated and the part of the relative velocity that is orthogonal to the rotation axis is rotated by the characteristic scattering angle. The new relative velocity is added to the center-of-mass velocity to get the new particle velocity after a collision. The principle concept of the random rotation is depicted in figure 3. To conserve Galilean invariance the grid of collision cells is shifted randomly relative to the coordinate system of the particles in every collision step.

One advantage of the MPC method is that a coupling to atomistic simulations is established in a simple way[2]. A program that implements such coupling is *MP2C*.

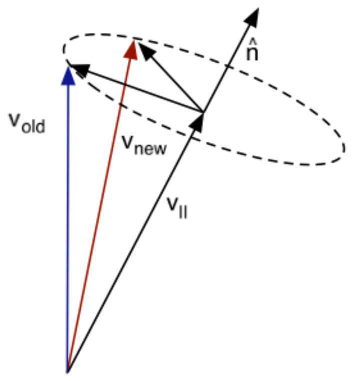


Figure 3: Principle of a stochastic rotation in Multiparticle Collision Dynamics

2.3 MP2C

MP2C combines multiparticle collision dynamics for fluid systems with MD simulations for molecular systems. This allows to simulate molecules dissolved in a liquid and explicitly take into account the hydrodynamic effects. The coupling between the two simulation schemes is done in the following way:

The usual MD moves are performed according to the given force field. Every n_{coll} timesteps a collision step is done, that also includes the solute particles. Both the solvent particles and the solute particles are sorted into collision cells. Then the center-of-mass velocity of the collision cell is calculated and a rotation around the same random axis is performed for both solvent and solute particles. These operations are performed according to equation 3 and 4. This procedure leads to a stochastic momentum exchange of the solutes while conserving the overall momentum and kinetic energy.

$$\mathbf{v}_{\mathrm{c,cm}} = \frac{1}{M_C^{\mathrm{slv}} + M_C^{\mathrm{slt}}} \left(\sum_{a \in C}^{N_{\mathrm{slv}}} m_a \mathbf{v}_a + \sum_{b \in C}^{N_{\mathrm{slt}}} m_b \mathbf{v}_b \right) \tag{3}$$

$$\tilde{\mathbf{v}}_i = \mathbf{v}_i - \mathbf{v}_{\mathrm{c,cm}} \quad , \quad i \in \{\mathrm{slv}, \mathrm{slt}\} \tag{4}$$

2.4 Parallelization

The parallelization strategy in *MP2C* is based on the domain decomposition (DD) scheme, which is used for the MD part as well as for the MPC part. Since the work that is reported in this paper was confined to the MD part the DD scheme will be described in the context of MD. In the DD scheme the physical simulation domain is subdivided into equal sized subdomains and each PE is assigned one of them. The information on a particle is stored on the PE governing the spatial domain the particle is currently in When moving from one subdomain to another the information on that particle is transferred from one PE to another. To calculate the forces on the particles in its domain a PE needs to know the coordinates of nearby particles in neighboring domains. These are particles that are closer to a domain border than the cut-off radius and they are commonly referred to as ghosts. To exchange the coordinate information on the ghosts only local point-to-point communication is required. In a simple approach a PE would need $3^{\mathrm{dim}} - 1$ communication steps to obtain the ghosts from its neighboring PE. With the use of an elaborate communication scheme the number of send/receive operations can be reduced to $2 \times \mathrm{dim}$ steps. In three dimensions the application of this communication scheme allows a PE to obtain all the required information from the 26 surrounding PE with just six communication steps.

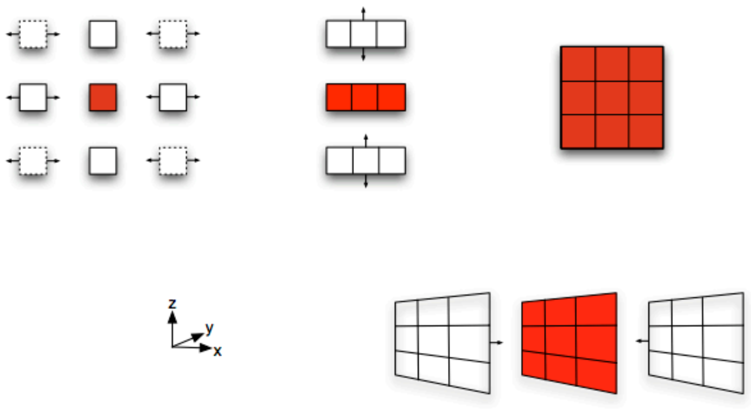


Figure 4: Communication scheme for efficient particle and ghost exchange. In the first step each PE exchanges with both adjacent PE in the first dimension. In the second step the same exchange takes place in a second dimension. Now the previously received data is included. In three dimensions a third exchange step is performed.

2.5 Problem Statement

Due to the homogenous distribution of the coarse-grained solvens particle the hydrodynamic part of the *MP2C*-code shows good scaling properties. However, the MD part shows less optimal scaling performance especially when the solute particles are inhomogenously distributed or their density is low. The reason for this behaviour is that with the domain decomposition a good load-balancing cannot be sustained when scaling to large numbers of PE. What happens is that the effective domain assigned to a PE is evermore shrinking when scaling up the number of PE. Therefore it is increasingly likely that a PE is assigned a domain with relatively little or no solute particles at all. The consequence is that these PE go idle during the force evaluation and the overall performance depends on the PE with the highest workload. As a solution to this problem an improved parallelization scheme has been implemented and given the name domain-force-decomposition.

3 Domain-Force-Decomposition

The Domain-Force-Decomposition scheme (DFD) is an extension to the DD scheme in the sense that it combines the DD-scheme with elements from other decomposition schemes to yield better load-balancing in the computation of the forces. The ground-lying decomposition scheme is the DD, which means that still each PE updates the positions and velocities of all the particles that are within its spatial domain. The difference is in the computation of the forces. In the DFD scheme eight neigboring PE make up a larger domain which will be referenced in the following as a coarse-grid domain (CG-domain). For ease of distinction the basic domain which describes the simulation space that is governed by one PE will be referred to in the following as DD-domain. The tie-up of the eight DD-domains in a CG-domain is done so that the CG-domain forms a cluster that comprises two DD-domains in each direction. Now the workload for computation of the forces on all local particles in a given CG-domain is partitioned as equally as possible among the PEs involved. This gives good load-balancing for each CG-domain.

3.1 Communication Scheme

Balancing the workload for force evaluation requires PEs that govern only a little number of particles take on the force evalution for particles that reside on PEs with a larger number of particles. However, the force evaluation for a particle requires a PE to know the coordinates of all the other particles that interact with that particle, i.e. all the other particles that lie within its cut-off radius. These particles could in turn reside on a third PE. Thus it appears that an approach where particles are locally exchanged to balance the workload for force evaluation would a significant amount of additional communication operations. In the DFD scheme a different approach is taken in which in a first step the overall work is collected and then redistributed. Collection of work means here that all coordinates that are required to evaluate forces on all the particles in a CG-domain are stored in one place. Therefore the DFD scheme introduces two new data structures that are maintained on all PEs:

- A memory buffer is maintained to store coordinates and a global index of all particles in its CG-domain. Additionally it stores coordinates and indices of nearby particles in neighboring CG-domains. This buffer will be referred to as COORD_BUFFER in the following.
- A second memory buffer, which will be referred to as FORCE_BUFFER, stores the force contributions during evaluation of the forces on the particles of a given CG-domain. Finally the contributions to the force on each particle are summed up and written to the local data structure that stores the complete information on each particle.

New collective communication is used to fill these buffers with the local data on each processor and to distribute the final results to each PE. The collective communication takes place on a newly created communicator that includes only the PEs in a given CG-domain. Every CG-domain creates its own communicator which is referenced as COMM_DFD in the following. Now, compared to the DD scheme point-to-point communication to exchange ghosts takes only place across CG-domain borders.

All these changes to the *MP2C*-code are easily implemented. The old routine calls to exchange ghosts and to evaluate forces are substituted by a call to the newly introduced routine *force()*. Algorithm 4 shows the pseudocode for the routine *force()*.

Algorithm 4 subroutine *force()*

```
fill COORD_BUFFER
if local particles $> 0$ then
  sort particles in linked-cells
  divide linked cells to PEs involved
  computes force contributions and stores in FORCE_BUFFER
  sum up force contributions with MPI_ALLREDUCE
  copy forces to local particle structure
end if
```

In the first step a call to *fill_force_coords()* fills COORD_BUFFER with the coordinates of all local particles in a given CG-domain and all ghosts in neighboring CG-domains. Furthermore COORD_BUFFER stores a global index for each particle. This index is used during force evaluation to determine the correct potential parameters for each particle. The data in COORD_BUFFER is distributed in two collective communication steps to all PEs involved. The population of COORD_BUFFER is performed according to the rank of the contributing PE in the specific communicator COMM_DFD. The principle of populating COORD_BUFFER is presented in figure 5. The received ghost's coordinates and indices are appended to COORD_BUFFER after the local particles data.

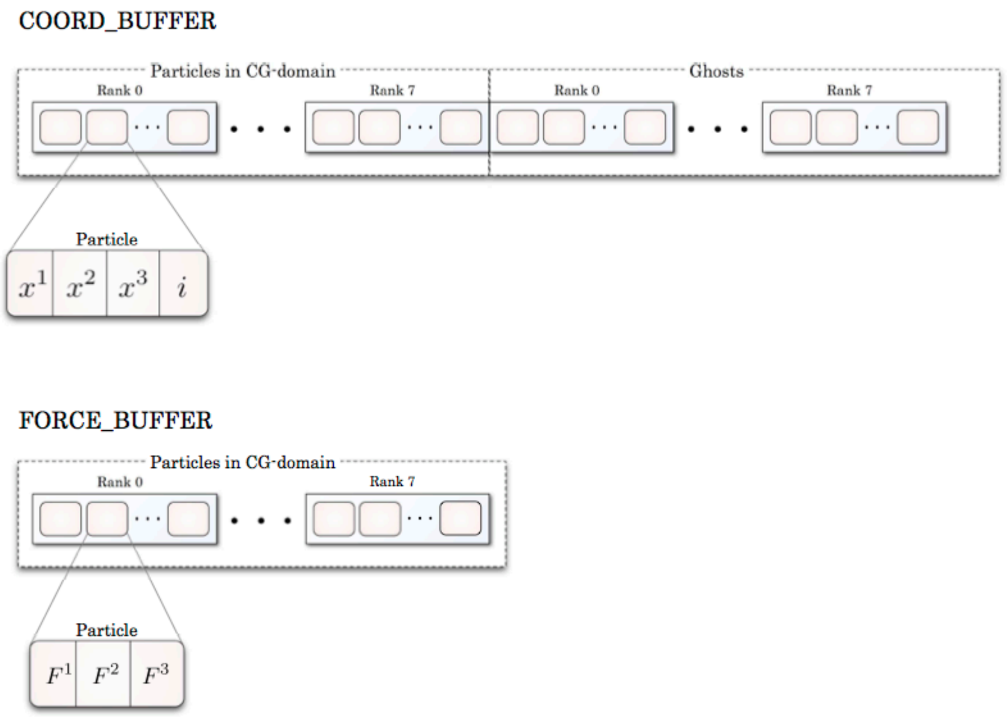


Figure 5: Memory layout for COORD_BUFFER and FORCE_BUFFER.

The detailed steps in the routine *fill_force_coords()* are given in Algorithm 5. This routine starts with all PEs involved communicating their local number of particles on COMM_DFD. Communication is realized with MPI_ALLGATHER. Now that each PE knows the exact numbers of particles residing on the other PEs within that communicator it can calculate the offset to store coordinates and indices of its particles in COORD_BUFFER. The particles are copied in the order of the ranks in COMM_DFD. After each PE has copied its data to COORD_BUFFER this data is synchronized on all PEs involved with MPI_ALLGATHERV. Then a ghost exchange step takes places in the same manner as previously described for the DD scheme. A crucial difference is that now the exchange happens only across CG-domain borders. In this 8-PE cluster set-up each PE does ghost exchange with three surrounding PEs, one in each direction, whereas in the DD scheme it was with all six surrounding PEs. The orientation of the ghost exchange in each direction is fixed for a given PE during initialization of a simulation run. After each PE has received ghosts for its section of the CG-domain border all PEs communicate the number of received ghosts in the same way as beforehand for the local particles. Now the offset for copying the ghosts to COORD_BUFFER is now calculate relative to the position of the last particle data in COORD_BUFFER.

Algorithm 5 subroutine ***fill_force_coords()***

```
  communicate number of local particles across CG-Domain with MPI_ALLGATHER
  calculate offset in COORD_/FORCE_BUFFER
  copy particles coordinates to COORD_BUFFER
  communicate particle coordinates in COORD_BUFFER with MPI_ALLGATHERV
  exchange ghosts with neighboring CG-Domains
  communicate number of ghosts with MPI_ALLGATHER
  calculate local offset in COORD_BUFFER
  copy ghost coordinates to the end ofCOORD_BUFFER
  communicate ghost coordinates in COORD_BUFFER with MPI_ALLGATHERV
```

After the return of *fill_force_coords()* each PE of the CG-domain contains the same data in its instance of COORD_BUFFER.

In the next step each PE sorts all particles in COORD_BUFFER in linked cells. The corresponding routine is named *fill_linked_cells()*. The actual load-balancing is done in the following step when contigous blocks of linked cells are distributed to the PEs involved. Only those linked cells that lie inside the CG-domain are distributed. Linked cells that comprise the border region in the neighboring CG-domains are not explicitly distributed. Each PE divides the total number of particles in the specific CG-domain by the number of PEs in that CG-domain to get N_{av}, the average number of particles on a PE belonging to the specific CG-domain. Next the distribution of linked cells to the PEs is performed in the order of their ranks in COMM_DFD. A rank gets linked cells until the number of particles on that rank equals or exceeds N_{av}. Each PE runs this algorithm up to the point where it has evaluated its share in the linked cells. The PE with the highest rank in COMM_DFD gets the remaining linked cells.

The following step is the force evaluation. Each PE works through the assigned linked cells und evaluates the force contributions for the particles contained within. During the run through the surrounding linked cells of a particle Newton's third law is applied and only half of them are checked for possible interactions. This means that in many cases contributions to the total force on particle are evaluated on various PE in the CG-doain. An exception are the linked cells that belong to the border region of a neighboring CG-domain. They are always included in the screening for possible interactions. The force contributions are summed up in FORCE_BUFFER. The offset of a particle in COORD_BUFFER is also used as offset for that particles in FORCE_BUFFER and hence identical for a given particle on all PEs in the specific CG-domain (see figure 5). To sum up the contributions to the total force on each particle a collective reduction operation is performed on FORCE_BUFFER. The reduction communication is realized by MPI_ALLREDUCE with MPI_SUM specifying the reduction operation.

4 Speedup Analysis

Up to the day that this report has been written the implementation of the DFD scheme in MP2C was still in the process of debugging. Certain parts of the implementation have been checked to work correctly while other are still not working in the desired way. Those parts that still do not work correctly are the process of exchanging ghosts and the application of Newton's third law when scanning surrounding linked cells. In order to assess at least qualitatively the effect of the DFD-scheme compared to the DD-scheme a reduced implementation of *MP2C* was build. On that implementation benchmark runs were performed that went just long enough to get an rough idea of the speedup that can be expected from a fully, correctly working implementation. This reduced implementation does not take full advantage of Newton's third law in evaluating forces since it is applied only to interactions between particles inside the same linked cell. This means that the total force on a given particle is always evaluated on a single PE and interactions between particles in different linked cells are calculated twice. The second important restriction is the fact that during a simulation run no ghost exchange at all takes place. This restricts the possible length of a benchmark run significantly. Therefore the benchmark runs were performed on a specifically configured system of 20000 solute particles on 8 PEs with the starting configuration chosen such that all solute particles were inside a single DD-domain. In this configuration no ghost exchange is needed as long as all solute particles remain in that DD-domain. Therefore the lenght of the benchmark run was chosen to 100 timesteps. Benchmark times for the force evaluation part both for the DD-scheme and the DFD-scheme are given in figure 6. With this configuration of the system in the DD-scheme the evalution of all forces is done by just one PE and no communication at all takes places during the benchmark run. In the DFD-scheme the workload for the evalution of the forces is shared between all 8 PE at the cost of global communication to communicate coordinates and to sum up and communicate forces. The results show that the communication overhead introduced in the DFD-scheme is fairly small compared to the speedup that is achieved. For the DFD-scheme it can be assumed that the total time does not change significantly with a different relative distribution of particles across the 8 PE (one CG-domain in a larger system). The benchmark time for the communication part in the DFD-scheme might change significantly but since communication makes up only a minor part of the total time this effect can be neglected. In the DD-scheme the total time is always the total time of the PE with the highest workload. With a different relative distribution of particles in a first approximation the total time can be assumed to scale linearly with the workload. Therefore the results promise a speedup with the DFD-scheme for more common (but still inhomogenous) particle distributions. Clearly there will be little to no speedup in cases where the system on whole is inhomogenously distributed but the regions of high particle density extend homogenous mentation seems to yiel

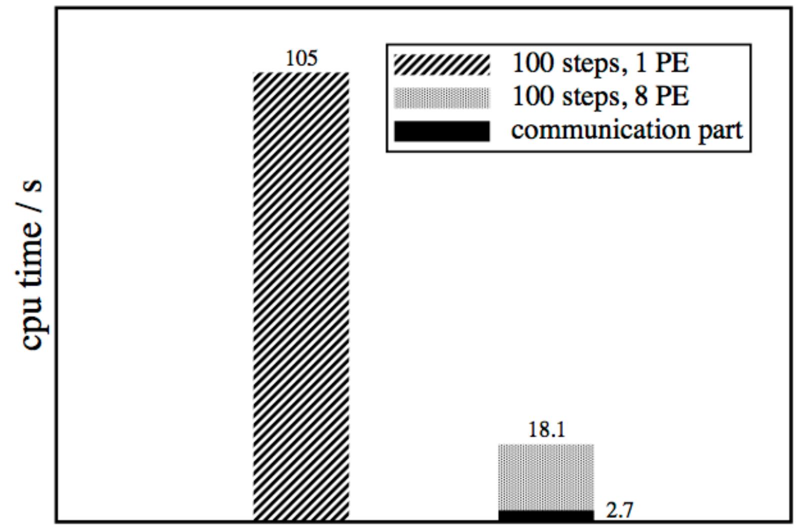


Figure 6: Comparison between the situation where force evaluation is done by just one PE in a CG-domain and the situation where force evaluation is partioned according to the DFD scheme. The latter case includes the all-to-all communication and the cpu time for communication is shown seperately. The benchmark times are taken for 100 timesteps. The system consisted of 20000 solute particles and was decomposed to one CG-domain with 8 DD-domains.

5 Outview

Apart from the above reported bugs that need to be fixed in the first place the here presented implementation of the DFD-scheme could be improved on different issues:

- Different shapes for a CG-domain could be examined, that match the inhomogeneity of the simulation system better.
- A more elaborate exchange scheme involving a combination of ghost and force exchange could be implemented. In the current implementation the force contributions involving ghosts are evaluated on both PEs involved. This means also that the ghosts involved are being exchanged mutually. Instead of ghost exchange taking place in both opposite directions in each dimension in the *fill_force_coords()* routine they could be sent to only one direction in each dimension and at the end of the force evaluation the forces for this ghosts could be sent back the inverse way. This way the force contributions on ghost particles would be evaluated once.

6 Acknowledgement

I would like to express my gratitude toward the JSC for enabling this work. I would like to thank Godehard Sutmann for his patience and support. I am indebted to Prof. Dr. D. Marx at Bochum University for his recommendation, which made my attendance possible in the first place. Special thanks go to Robert Speck and Benjamin Berberich for the great organization of the guest student program. I would like to thank to my fellow students for all the useful advice and the nice atmosphere.

References

1. Griebel, M., S. Knapek, G. Zumbusch und A.. Caglar: Numerische Simulation in der Moleküldynamik. Springer Verlag, 2003.
2. Sutmann, G., R. Winkler und G. Gompper: Multi-particle collision dynamics coupled to molecular dynamics on massively parallel computers. in preparation, 2009.
3. Plimpton, S.: Fast Parallel Algorithms For Short-range Molecular-dynamics. Journal of Computational Physics, 117(1):1–19, 1995.
4. Sutmann, G.: Molecular Dynamics - Extending the Scale from Microscopic to Mesoscopic. unpublished, 2009.