SUPERCONDUCTING QUANTUM MATERIALS & SYSTEMS CENTER

SQMS Summer Internship Report

Optimising HEP Simulations on 3D Architecture: Primitive Gates for BT

Andrea Maestri[1,3], Hank Lamm[2], Doga Kurkcuoglu[1,4], and Judah Unmuth-Yockey[2]

[1]*Northwestern University, Department of Physics and Astronomy, Center for Applied Physics and Superconducting Technology*
[2]Theory Division, Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
[3]University of Amsterdam
[4]Fermilab Quantum Institute, Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

October 2022

Abstract

In this report, the first attempt for an alternative approach for the simulation of the non-abelian group **BT** has been done. First, we analysed why SNAP and Displacement decomposition is not feasible for this problem with the current technology, and then we explored a new approach for the optimisation of the pulses. In particular, we studied how the expansion of the pulses in Chebyshev Polynomials could affect the optimisation process. Even if the study is not conclusive, we believe that it might be worth continuing to study this approach in the future, in order to be able to efficiently implement quantum circuits on computers.

Contents

1 Introduction

Quantum Information Science makes use of quantum mechanics to realise algorithms and operations that are intractable by standard computers. Theoretical and experimental investigations in high-energy physics is a prime driver for pushing the frontiers of computation paradigms and computing power. In particular, simulating the dynamics of lattice field theories offers clear potential for quantum advantage [3].
This work starts by analysing the implementation of the simulation of the non-abelian group **BT**, a subgroup of **SU**(2). Using pulse optimisation, we aim to improve on the implementation of the quantum algorithms which utilise the usual SNAP and Displacement decomposition [1]. In particular, the aim of this work is to find if, using pulse optimisation, it is possible to decrease the amount of time required for the simulation of the theoretical operations on the group and therefore, if it is possible to approach the simulation of groups like **BT** with the current technologies.
While the theory for lattice field theory or intricacies of **BT** is not treated in this report, a brief description of the quantum algorithms used to simulate this theory, and why we are interested in pulse optimisation, is given in the following sections. For a more in-depth theoretical analysis of the HEP problem, we always refer to [8].
There are multiple approaches to gate implementation. However, on current quantum computers, with the current decoherence times, most problems are still intractable. In particular, in the simulation of **BT**, the number of the SNAP and Displacement gates required for a single quantum circuit is far beyond what can be implemented [1].
Finding a general way to do pulse optimisation can be beneficial not only for this problem, but in principle it can reduce the cost of implementing any quantum algorithm. Moreover, for systems whose final state we know, the development of the quantum algorithm can be skipped and the gate can be found directly by the optimisation algorithm when using pulse optimisation.
The starting point for this study was the simulation of the pulses using the python library QUTIP[10]. However, as the dimension of the gate corresponded to a 48×48 matrix in our study case, this approach resulted infeasible and optimal pulses were impossible to find running the simulations on a laptop[1]. Different approaches spanned from using binary pulses [6] to quantum reservoir computing [7]. Other approaches include a combination of different optimisation algorithms, such as genetic and gradient descent algorithms [13]. While the first two are not meant to be tested by us, the last two are algorithms that are future directions of this paper.
All of these approaches have something in common: the single pulse to optimise corresponds to a time series $\vec{u}$ of dimension N_{steps}, where N_{steps} is the total number of time steps required by the optimisation, a parameter that has to be tuned depending on the problem. In this notation, each value of the vector corresponds to the value of the pulse at time t_k with $k \in [0, N_{steps}]$. The values of the pulses are completely uncorrelated, and therefore the final optimisation process could correspond to an amplitude that is difficult to implement on an actual computer.
Moreover, for high dimensional problems such as the simulation of **BT**, a large number of time steps is required to reproduce a gate with high fidelity, resulting in thousands of parameters to optimise.
In the current report, we explore a different approach, consisting of finding an analytical expression of the pulses by expanding them into Chebyshev polynomials. In section 2, a

[1] the specs of the laptop are: Apple M1 chip, 8-core CPU with 4 performance cores and 4 efficiency cores

brief introduction to the HEP problem will be given, as well as the current implementation approach. In section 3 we give an overview of the literature on pulse optimisation and propose our new approach. In the remaining sections, we discuss the advantages and drawbacks of this approach, as well as suggesting future steps to solve this problem.

2 Simulation of BT

The increasing complexity of simulating quantum field theories naturally brings up the possible advantage of simulations on quantum computers. In particular, quantum computers seems to fit extremely well the simulation of High Energy Physics problems. Easier said than done, performing such a simulation has multiple requirements and difficulties. In this section, we will show what is the nature of the problem that we are focusing on and what are the complications that emerge.
Time evolution on quantum computers requires efficient implementation of the unitary operator $U(t) = e^{-iHt}$. Various approaches exist, but all of them require implementing the key group theoretic operations as quantum circuits in the case of lattice gauge theories [9]. Moreover, a digitisation method needs to be defined. One promising method is the discrete group approximation [4] which reduces resources substantially. For nonabelian gauge groups, we only have a few subgroups. **SU**(2) has 3 and in this report, we are going to focus on **BT** which has proven to be a good trade-off between computational complexity and accuracy compared to **SU**(2).

2.1 Primitive gates for BT

Without discussing in depth the theory behind **BT** that can be found in [8], it has to be said that this theory requires nonabelian bosonic degrees of freedom and that each of those bosons needs 5 qubits, or alternatively for a $3D$ architecture, a *quicosotetrit* (qudit with $d = 24$), that is the alternative on which we are going to focus on.
The simulation of lattice gauge theories requires the definition of a register where one can store the state of a bosonic link variable which we call a $G-$register. In order to construct the $-$register in terms of integers, it is necessary to construct a mapping between the 24 elements of the group and the integers $[0, 23]$. A clean way to obtain this is to write every element of BT as an ordered product of four generators[2] with exponents written in terms of the binary variables m, n, o, p, q:

$$g = (-1)^m \mathbf{i}^n \mathbf{j}^o \mathbf{l}^{p+2q}, \tag{1}$$

with

$$\mathbf{l} = -\frac{1}{2}(1 + \mathbf{i} + \mathbf{j} + \mathbf{k}) \tag{2}$$

and $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are the unit quaternions which in the 2d irreducible representation (irrep) correspond to Pauli matrices. With the construction of Eq. (1), the $-$register is given by a binary encoding of the qubits with the ordering $|qponm\rangle$. This same mapping can be utilized for . For example, using $\eta = 1 + i$ one element in the real 2d irrep is

$$\frac{1}{2}\begin{pmatrix} \eta & \eta^* \\ -\eta & \eta^* \end{pmatrix} = (-1)^1 \mathbf{i}^1 \mathbf{j}^1 \mathbf{l}^{0+2\times 1} \to |10111\rangle = |23\rangle \tag{3}$$

For general gauge groups, it is possible to define any quantum circuit with sets of primitive gates[11]. One choice of primitive gates is: inversion $\mathcal{L}_{-1}$, multiplication $\mathcal{L}_\times$, trace $\mathcal{L}_{Tr}$ and Fourier transform $\mathcal{L}_F$. The inversion gate, $\mathcal{L}_{-1}$, is a single register gate which takes a group element to its inverse:

$$\mathcal{L}_{-1} |g\rangle = |g^{-1}\rangle .$$

[2]The minimal set of generators for is two, but we have been unable to find an ordered product with less than three. The choice of three generators is the same as the one with four generators where $(-1)^m \mathbf{i}^n \to \mathbf{i}^{2m+n}$. Nevertheless, the qubit costs cannot go below the current formulation's value of $\lceil \log_2(24) \rceil = 5$.

The group multiplication gate acts on two quicosotetrits, it takes the target and changes the state to the left product with the control:

$$\mathcal{L}_{\times} |g\rangle |h\rangle = |g\rangle |gh\rangle .$$

The trace of group elements:

$$\mathcal{L}_{Tr} |g\rangle = e^{i\theta ReTrg} |g\rangle .$$

And finally the Fourier transform:

$$\mathcal{L}_F \sum_{g\in G} f(g) |g\rangle = \sum_{\rho\in\hat{G}} \hat{f}(\rho)_{ij} |g\rangle |\rho, i, j\rangle ,$$

where $\hat{f}$ denotes the Fourier transform of f[3]. As will be seen in the following sections, the cost of **BT** is still far from what can be simulated on current quantum computers, and as mentioned before, the goal of this report was to study if this cost can be reduced. This led to the analysis of only the multiplication gate and the inversion gate which are, in order, the simplest non-trivial gate, and the most expensive.

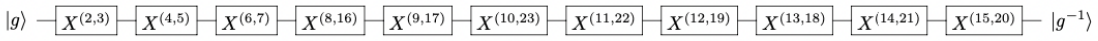

Figure 1: A quicosotetrit implementation of $\mathcal{L}_{-1}$ using the $X^{(a,b)}$ gate.

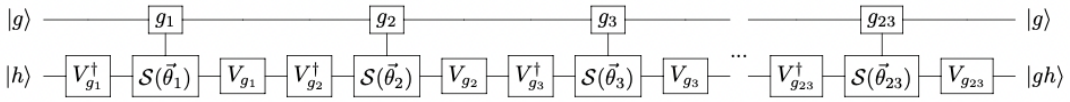

Figure 2: A quicosotetrit implementation of $\mathcal{L}_{\times}$. The subscript g_i indicates the i^{th} element of the group.

2.2 Cost analysis for SNAP and Displacement decomposition

One possible approach for the implementation of the primitive gates is to decompose each elementary operation into a set of gates which are well implemented on the computer. In particular, for a qudit, it can be shown that SNAP and Displacement gates are universal (i.e. each elementary operation can be reproduced as a combination of them). SNAP gates can arbitrarily phase the qudit states:

$$\mathcal{S}(\vec{\theta}) = \sum_{a=0}^{N-1} |a\rangle \langle a| e^{i\theta_a}$$

where the sum is over the computational basis states $a \in [0, N-1]$ of an N-state qudit and $\vec{\theta} = \{\theta_0, \theta_1, ..., \theta_{N-1}\}$ are a set of tunable parameters analogous to a rotation of θ around the z-axis in the block sphere for the qubit.
The Displacement gate coherently changes the cavity's photon number. In terms of Fock-operators a, $a^\dagger$, it is

$$\mathcal{D}(\alpha) = e^{i\alpha\left(a^\dagger + a\right)}.$$

[3]See [8] for better theoretical explanation of the gates

Now, in order to obtain a universal set of gates, we require an entangling gate. The one used in [8] is the controlled SNAP which phases a target qudit based on if a second qudit is in a state $|\alpha\rangle$. With the current algorithms developed at SQMS, it is possible to find the parameters for these two gates efficiently. However, the problem with this approach consists in the number of gates required to implement one trotter step.
As we can see from table 1, the number of gates to simulate a single trotter step for dimension d is high. Moreover, this cost refers to only one link and at least 4 links are needed[4] to simulate the dynamics of the group[8]. This pushed us to explore different ways of implementing the gates. The natural alternative to SNAP and Displacement is pulse optimisation which will be described in the next sections.

Gate	$c\mathcal{S}(\vec{\theta})$	$c\mathcal{S}(\vec{\theta})$	$c\mathcal{D}(\alpha)$
$\mathcal{L}_{-1}$	0	24	25
$\mathcal{L}_{\times}$	23	575	575
$\mathcal{L}_{Tr}$	0	1	0
$\mathcal{L}_{F}$	0	24	25
$e^{-iH_I\delta t}$	$598d - 506$	$15215.5d - 12771.5$	$15225d - 12775$

Table 1: SNAP and Displacement decomposition's cost analysis per link where d represents the dimension.

3 Pulse Optimisation

The other possible approach for the synthesis of the gates is pulse optimisation. This approach consists of the application of control hamiltonians with an intensity that is modulated by pulses that vary accordingly to the gate that has to be implemented

$$H = H_0 + \sum_i u_i(t) H_i. \tag{4}$$

The drift hamiltonian H_0 (i.e. a hamiltonian that does not depend on time and it's constant for the system) is the one that doesn't vary with time. It can be the Jaynes-Cummings hamiltonian or a different one depending on how the qubit and the cavity are coupled[5]. On the other hand, the control hamiltonians usually are a combination of the sigma matrices and a combination of the creation and annihilation operators. In this report, different operators will be used, but they will be always specified depending on the problem.
Once the hamiltonian of the problem is defined, in order to implement the gate, the evolution operator is constructed as follow:

$$U_T = \prod_{k=0}^{N} e^{-iH(t_k)\delta t} \tag{5}$$

in order to construct the adiabatic evolution of the system. The goal of the pulse optimisation, given an arbitrary gate X_{tar}, is to find an optimal value for the pulses such

[4] 4 is needed at minimum, but realistically we need at least $d(4)^d$ of them where d represents the dimension

[5] This can be extended to other types of systems, for example to non-transom qubit. The evolution of those systems is of course governed by a different hamiltonian

that

$$U_T \approx X_{tar}.$$

One possible way to do that is to minimise the following infidelity function

$$\hat{f} = 1 - |Tr\left(U_T^{\dagger} X_{tar}\right)|. \tag{6}$$

The difference in the various approaches for pulse optimisation is in how the amplitudes are defined and in the possible optimisation algorithms. The usual approach consists of the utilisation of the LBFGS [12] algorithm for the optimisation and on a piece-wise expansion of the pulses.

3.1 Qutip's pulse optimisation

QUTIP is a library developed for multiple purposes for quantum computing [10]. Its utilisations span from the solution of Schrödinger equations to pulse optimisation for the gates synthesis. In this part of the report, we are going to explore briefly how QUTIP performed on the problem we are interested in, and why we decided to explore new optimisation strategies.
In this optimisation process, the drift hamiltonian used was the Jaynes-Cummings hamiltonian. After a change of variables it can be written as

$$H_0 = -2\pi a^{\dagger} a \sigma_z,$$

while the control hamiltonians were a linear combination of $a^{\dagger}$, a, σ_+, σ_-. The sigma matrices act on the qubit system while the creation and annihilation operator on the photons states.

3.1.1 Trace Gate

The first gate that we were interested in optimising was the trace gate. However, this gate is already optimised using SNAP and Displacement decomposition. In fact, it can be written as only one SNAP gate. This led us to focus more on the remaining ones.

3.1.2 Inversion Gate

The second gate that was tried, and the last one using Qutip, was the Inversion gate. We ran the *pulse_optimisation* algorithm with different combinations of the hyperparameters in order to get the best possible result in a reasonable time.
The best combination of parameters resulted to be[6]:

- pulses type: sinusoidal,
- pulse duration: $1ms$,
- time slots: 5000,
- maximum number of iterations: 10000,
- maximum time of execution: $8h$.

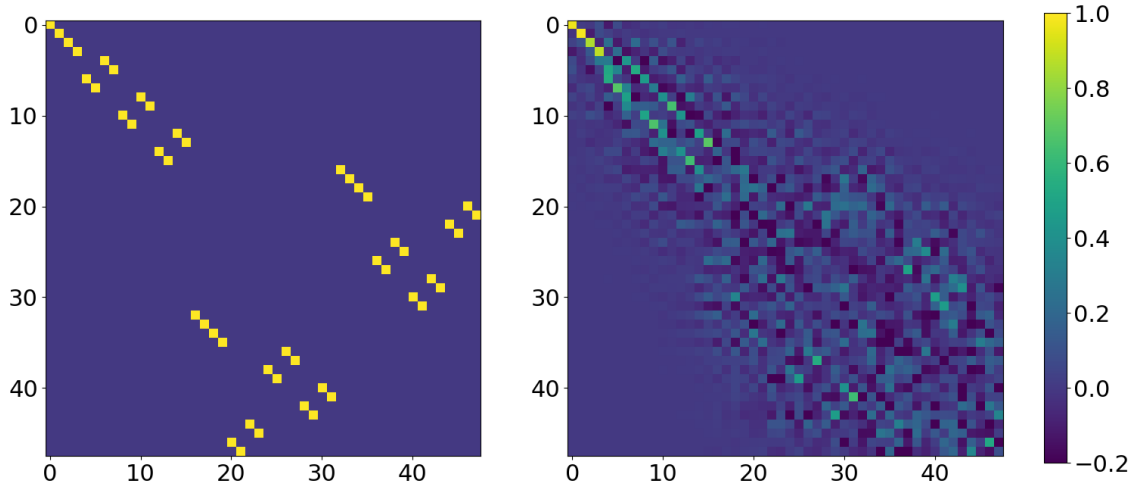

Figure 3: Inversion gate found using Qutip's pulse optimisation.

With this combination, the result got from the optimisation was a gate implemented with an infidelity $\hat{f} = 0.66$. The resulting matrix is depicted in fig 3.
It is worth mentioning that whatever the infidelity is, it is unfortunately not the one that is going to be obtained by the machine. On top of the error due to the optimisation process, there is the error due to the actual physical implementation of the pulses, and the accuracy managed to achieve in the laboratory. Putting everything together, the result obtained is not something we should even try to implement on a quantum computer. Moreover, since the parameter tuning has been done already, the final value for the infidelity is not easy to improve [7].
This pushed us to the exploration of new methods to handle this problem. In literature, there are various approaches that are well documented, however, due to the high dimensionality of our problem, we thought that the number of parameters to optimise required would have been too big and that such approaches would not be scalable for future problems. This led to the idea of expanding the pulses into Chebyshev polynomials as it will be described in the next section.

3.2 Pulse optimisation using Chebyshev expansion

The main disadvantage of using a piecewise approach for the control amplitudes is that for each control hamiltonian, a specific value of the pulse has to be found for each time step. For the optimisation of a qubit gate, this is not a problem since with a number of steps that is between 10 and 100, it is always possible to find an optimal gate. On the contrary, for the problem we are focusing on, the number of time steps required for the optimisation is obviously higher due to the higher complexity of the final gate. From multiple runs for simplified problems (Fock space of size 8 for example), it was noticed that at least 1000 steps are required to obtain an infidelity of 10^{-3}. This approach is therefore not feasible for the simulation of **BT** since, considering the fact that we have 4 control hamiltonians for this problem, we will have a number of parameters that is somewhere between 10^3 and 10^4.

[6] All the simulations in this report were running on a laptop with the following specs: Apple M1 chip 2020 8-core CPU with 4 performance cores and 4 efficiency cores

[7] If it is possible at all

However, we found a way to reduce the number of parameters. The idea that we had was an expansion of the pulses in order to find an analytical expression of them. The choice was between a Chebyshev and a Fourier expansion. The first was preferred due to the smaller error with the name number of terms in the sum.
More precisely, given the hamiltonian in equation 4, we wrote the pulses in the following form:

$$u_i(t) = \sum_{k=0}^{K} c_k T_k(t),$$

where the $\{c_k\}$ are the parameters to optimise and the $\{T_k\}$ are the Chebyshev polynomials [2] of the first kind defined as

$$T_k(cos\theta) = cos(k\theta),$$

and the first terms are shown in fig 4. With such an approximation, it can be shown that

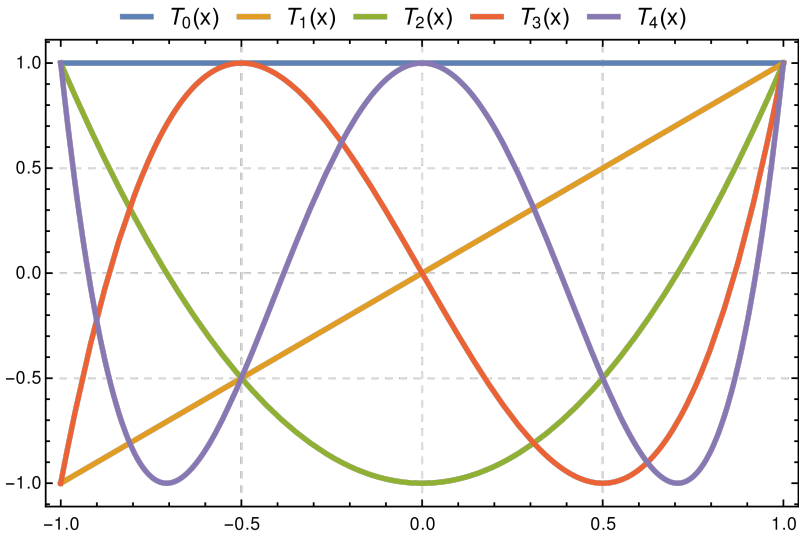

Figure 4: Plot of the first 4 Chebyshev polynomials.

knowing the coefficients exactly, the error in approximating a function follows this formula [2]

$$E(x) = |f(x) - P(x)| \leq \frac{1}{2^K(k+1)!} \max_{-1\leq t\leq 1} |f^{(n+1)}(t)|.$$

Therefore, already with 100 parameters, the error in the expansion would be smaller than what we will ever need for our problems.
After expanding the pulses, the resulting propagator that needs to be optimised is the following:

$$U_T = \prod_{k=0}^{N} e^{-i(H_0 + \sum_i \left(\sum_{k=0}^{K} c_k T_k(t_k)\right) H_i)\delta t}.$$

Given this, the parameters that have to be optimised are the elements of a matrix of size $N \times K$ where K is the number of Chebyshev polynomials and N is the number of control hamiltonians. It is worth noticing that the domain for the Chebyshev polynomials goes from -1 to 1. However, this is not a problem since we can find the form of the pulses and then just rescale them to the time interval that we need. The time interval can be shortened such that the synthesis of the gate that we need can be implemented before the decoherence time of the qubit.

3.3 Optimisation algorithms

Multiple optimisation algorithms were tested in order to run the simulation as fast as we could. Knowing that for the SNAP and Displacement decomposition the Stochastic Gradient Descent worked very well for the same HEP problem, we started experimenting with this type of algorithm. The first algorithm that was tried was actually the Stochastic Gradient Descent Algorithm whose scheme is depicted in alg. 1.

Algorithm 1 Stochastic Gradient Descent algorithm.

$it = 0$
$\vec{x} = rand(n)$
while $it \leq N$ **do**
    $grad = \Delta f$
    $\vec{x} = \vec{x} - \alpha \cdot grad$
    $it = it + 1$
end while

With this algorithm, preliminary results were produced. In particular, after implementing the optimisation algorithm and the calculation of the cost function in Julia [5], it was tested if the expansion in Chebyshev polynomials could produce simple gates such as the Hadamard gate.
In particular, setting the drift hamiltonian $H_0 = \sigma_z$ and the control hamiltonian $H_1 = \sigma_x$, it was found that expanding the control pulse to 5 terms, we reached an infidelity of $\hat{f} \approx 10^{-16}$. Such low infidelity is not the only benefit that we found from this approach. Indeed, the optimisation algorithm resulted to be between 10 and 100 faster than QUTIP's pulse optimisation (which performed equivalently in terms of infidelity). In addition to this, the pulses found using the Chebyshev expansion were smoother than the one obtained with Qutip 5, which is an important feature when we try to actually implement the pulses in the laboratory because that will allow us to change the intensity of the fields less rapidly.

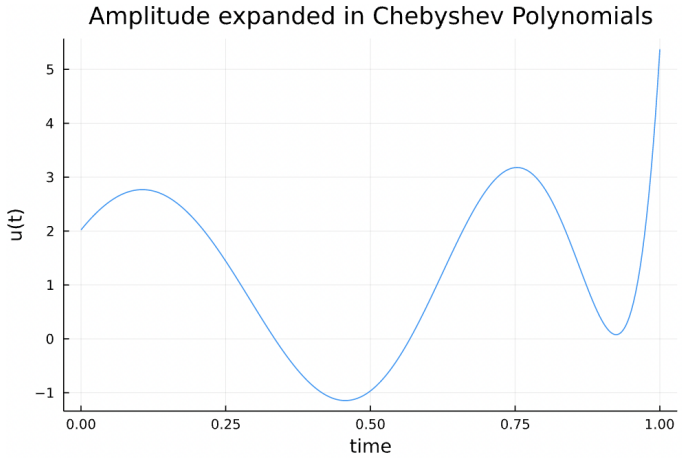

Figure 5: Pulses for the Hadamard gate using Chebyshev expansion for the optimisation of the pulses.

Since the performance of the algorithm seemed promising, we tried to perform the optimisation process for a Fock space of size 5 reproducing a simple X-gate, resulting in a matrix of dimension 10×10. For this system, the Hamiltonian was the same as the

one used in Qutip, the number of terms in the Chebyshev expansion was 200 for each control hamiltonian and the number of time steps was 300. The algorithm was run with the following set of parameters: $N_{it} = 5000$ and $\alpha = 0.01$. As it is possible to see from

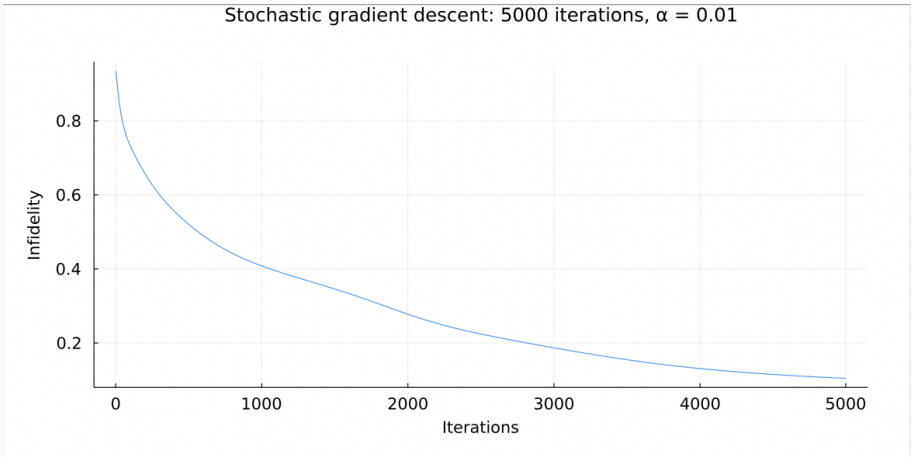

Figure 6: Infidelity vs Iteration number for the stochastic gradient descent algorithm.

fig 6, the cost function rapidly decreased in the first iterations, and slowly got stuck in a local minimum. The final infidelity achieved resulted in $\hat{f} = 0.12$, moreover, the number of iterations required to reach such a result seemed to indicate the fact that the landscape of the cost function is more complicated than what we were expecting, and that therefore a gradient descent methods might not be perfect for the optimisation process.
In order to give more stochasticity to the process, we thought of combining the stochastic gradient descent and the simulated annealing algorithms. The idea was to give the algorithm the possibility of getting out of a local minimum once there with a certain probability. The algorithm is the one described in alg 2. However, we barely tested this approach because we preferred to try different algorithms as well. As it is possible to see in fig 7, for only 1000 iterations, the final infidelity was not promising, in addition to this, we can see that at the beginning the infidelity remained constant, due to the high temperature of the simulated annealing algorithm, and at the end started decreasing, probably due to the fact that at that point the only algorithm that was giving an impact to the optimisation process was the stochastic gradient descent.
Putting everything together, we decided to move on from this algorithm, although keeping in mind that with a finer tuning of the parameters or a different decreasing scheme for the temperature, it could have given better results.
After testing with these 2 algorithms, we tried the more common LBFGS algorithm whose description can be found in [12]. This algorithm is the one used by Qutip and it was already implemented in Julia in the *Optim* library. Before running the optimisation for the full problem, we tested different combinations of the parameters in order to gain more knowledge about the problem. First, we started with a target infidelity of

$$\hat{f}_{tar} = 10^{-3}.$$

Then, after fixing the size of the Fock space, we tested how many iterations and how long did it take for the algorithm to reach that value of infidelity, changing the number of time steps for each simulation.
We started from a Fock space of size 2 to 6, and we tested 400, 600 and 1000 time steps. In all the simulations the hamiltonian was the same as the one used for the Qutip's

Algorithm 2 Stochastic Gradient Descent algorithm combined with simulated annealing.

$it = 0$
$\vec{x} = rand(n)$
while $it \leq N$ **do**
  $grad = \nabla f$
  $\vec{x} = \vec{x} - \alpha \cdot grad$
  $\vec{x'} = \vec{x} + \beta \cdot rand(n)$ ▷ β decreases with the number of iterations
  $p = e^{\frac{\hat{f}(\vec{x}) - \hat{f}(\vec{x'})}{kT}}$
  if $\hat{f}(\vec{x'}) \leq \hat{f}(\vec{x})$ **then**
    $\vec{x'} = \vec{x}$
  else if $rand \leq p$ **then**
    $\vec{x'} = \vec{x}$
  end if
  decrease T, decrease β
  $it = it + 1$
end while

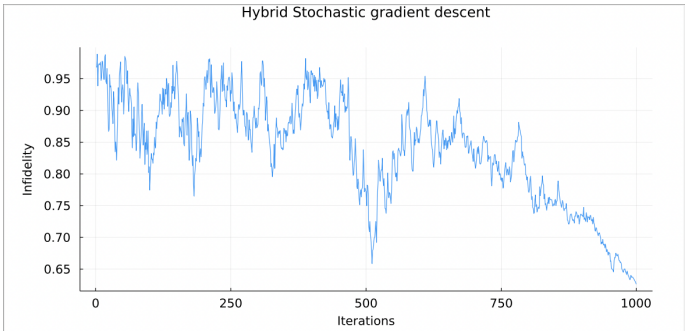

Figure 7: Infidelity vs Iteration number for the stochastic gradient descent + simulated annealing algorithm.

optimisation, and the number of terms in the Chebyshev expansion was 200 for each control hamiltonian. As it is possible to see in table 2, the optimisation algorithm seems to work better for a smaller number of time steps. However, as we noticed from the pulse optimisation in Qutip, a small number of time steps that corresponds to fewer pulses applied to the system might not be enough to reach an optimal result. Therefore a balance needs to be found for this parameter, especially for higher dimensional systems like the simulation of **BT**.
In the next sections, we will embark on a discussion about the increasing complexity of the optimisation process for a bigger number of time steps.

4 Results

Since the LBFGS algorithm gave good results for certain values of the N_{steps}, we decided to run the full optimisation of the inversion gate using that algorithm. The goal was to see if it was possible to obtain a better result than Qutip pulse optimisation, and we will see that this is actually possible. In particular, running the optimisation for $N_{steps} = 100, 200, 300$, we got the following results with a computation time of 4hrs for each simulation:

	$N_{steps} = 400$	$N_{steps} = 600$	$N_{steps} = 1000$
$d_{F.S.} = 2$	$(4', 86, \checkmark)$	$(4', 27, \checkmark)$	$(28', 26, \checkmark)$
$d_{F.S.} = 3$	$(4', 68, \checkmark)$	$(4', 48, \checkmark)$	$(9', 69, \checkmark)$
$d_{F.S.} = 4$	$(13', 179, \checkmark)$	$(19', 207, \checkmark)$	$(33', 210, \checkmark)$
$d_{F.S.} = 5$	$(34', 323, \checkmark)$	$(62', 392, \checkmark)$	$(1h, 234, 0.006)$
$d_{F.S.} = 6$	$(45', 322, \checkmark)$	$(3h, 834, \checkmark)$	$(1h, 181, 0.07)$

Table 2: Results for the optimisation process for different dimensions of the Fock space and for different values of the Number of time steps. The elements in the brackets in each term of the table are in order: computation time, the number of cost function calls, and whether the target infidelity was reached or not.

	$N_{steps} = 100$	$N_{steps} = 200$	$N_{steps} = 300$
$\hat{f}$	0.65	0.55	0.78

Table 3: Results for the optimisation process for the inversion gate for different values of the number of time steps.

Even if the results are very far from being useful for the synthesis of the gates, we still got an improvement over Qutip's optimisation. The resulting amplitudes for the best result are shown in fig 8.

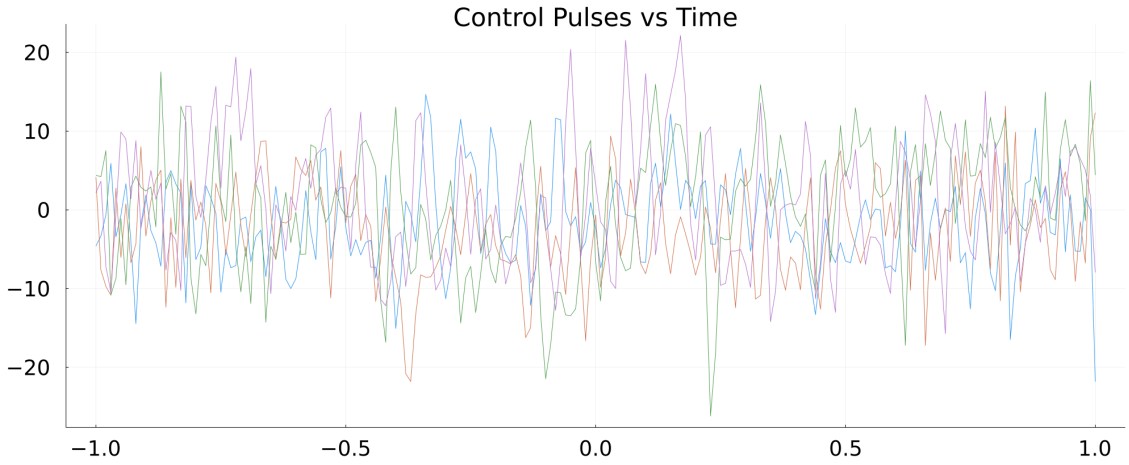

Figure 8: Pulses obtained from Chebyshev decomposition optimisation, using the LBFGS algorithm and $N_{steps} = 200$

5 Discussion

Even if the results obtained for the inversion gate resulted to be better than what we previously had using Qutip, we are still far from having a good optimisation algorithm. In this section, we are now going to try to provide an explanation of why, even reducing the number of parameters drastically, the optimisation process was not much better than what is usually in the literature.
First, it is worth mentioning that, for gradient descent approaches, the gradient has of course to be computed. If the number of parameters is small, then this is not a problem,

but in this case, where we had $N \in [400, 800]$, the cost function has to be called N times for each iteration (if we are not computing the Hessian matrix). This reason alone is already a hint in the utilisation of algorithms that are gradient-free such as simulated annealing or genetic algorithms. Moreover, looking at the expression of the operator that we wanted to optimise:

$$U_T = \prod_{k=0}^{N} e^{-i(H_0 + \sum_i \left(\sum_{k=0}^{K} c_k T_k(t_k)\right) H_i)\delta t},$$

we can immediately see that, compared to a piecewise approach, where each parameter counts only for one time step, here, all of them are in every term of the product. This makes the infidelity function way more complicated to optimise. Putting everything together, using gradient descent approaches for such a problem, might not only be slow but also will probably always get stuck in a local minimum. Lastly, we know that finding optimal pulses is expensive, however, we restricted ourselves to a laptop to find fast algorithms. An obvious continuation of what has been done is using a workstation or a cluster to do more intense calculations.

6 Conclusion and Future work

The aim of this report was initially to compare the difference between SNAP and Displacement decomposition and Pulse optimisation for the simulation of HEP problems, specifically the simulation of the non-abelian group **BT**. However, since this comparison was meant to be done using QUTIP for the pulse optimisation, it was found that the optimisation process resulted to be very slow, and gave infidelity values far from the ones given by the SNAP and Displacement decomposition. Therefore we decided to move the goal of this work from a comparison between methods, to the implementation of a new one.
Initially, we studied multiple approaches that are well known in the literature, however, in order to reuse part of the code written in Julia for the SNAP and Displacement decomposition we had the idea of expanding the pulses into Chebyshev polynomials.
As we have seen in the past sections, reducing the number of parameters to be optimised does not immediately lead to the solution of the problem, but some improvements were found anyway compared to the usual approaches applied to our scenario.
Now, knowing that the work is far from being conclusive, we still want to point out some conclusions that have been drawn. The pros of this approach are:

- It will in principle give pulses that are smoother than random pulses generated with a library like QUTIP and this will lead to an easier implementation on the machine.
- The number of parameters was drastically reduced, and with this approach, it will remain in the same order of magnitude no matter the problem we are going to face.
- The pulses can be made shorter, and this will open to the simulation of problems that are impossible to simulate right now.

However, there are drawbacks that need to be faced. For example, the very complicated landscape of the cost function when the number of time steps is increased, and the fact that with such a landscape a gradient descent algorithm will probably get stuck in a local minimum every time.
In future work, it might be worth exploring different approaches that are different from the one used in this paper. The idea that we had, was to further explore hybrid algorithms

such as the stochastic gradient descent method combined with the simulated annealing, as well as developing genetic algorithms to explore a wider part of the landscape instead of getting stuck in a local minimum immediately. In conclusion, we believe that this problem is worth studying further. The reason why is that, regardless of the level of technology developed for quantum computers, being able to implement the gates with high accuracy and within a shorter time, will give us the possibility of implementing more operations before the decoherence of the qubit, as well as possibly go from one state to another one, without developing the quantum algorithm that lies behind this operation.

Acknowledgements

Having the possibility of studying what you like is a privilege, doing that together with scientists at Fermilab such as my supervisors Hank, Doga and Judah is something that can not even be described.
This would not have been possible without Silvia Zorzetti, and the guys with which I shared my life here: Roberto Menta, Francesco Prince Cioni, Tommaso Bonaccorsi and Filippo Antola.
Without you it would not have been the same.

References

[1] M. Sohaib Alam, Sergey Belomestnykh, Nicholas Bornman, Gustavo Cancelo, Yu-Chiu Chao, Mattia Checchin, Vinh San Dinh, Anna Grassellino, Erik J. Gustafson, Roni Harnik, Corey Rae Harrington McRae, Ziwen Huang, Keshav Kapoor, Taeyoon Kim, James B. Kowalkowski, Matthew J. Kramer, Yulia Krasnikova, Prem Kumar, Doga Murat Kurkcuoglu, Henry Lamm, Adam L. Lyon, Despina Milathianaki, Akshay Murthy, Josh Mutus, Ivan Nekrashevich, JinSu Oh, A. Barış Özgüler, Gabriel Nathan Perdue, Matthew Reagor, Alexander Romanenko, James A. Sauls, Leandro Stefanazzi, Norm M. Tubman, Davide Venturelli, Changqing Wang, Xinyuan You, David M. T. van Zanten, Lin Zhou, Shaojiang Zhu, and Silvia Zorzetti. Quantum computing hardware for hep algorithms and sensing, 2022.

[2] Walter Van Assche. Chebyshev polynomials in the 16th century. *Journal of Approximation Theory*, 279:105767, jul 2022.

[3] Christian W. Bauer, Zohreh Davoudi, A. Baha Balantekin, Tanmoy Bhattacharya, Marcela Carena, Wibe A. de Jong, Patrick Draper, Aida El-Khadra, Nate Gemelke, Masanori Hanada, Dmitri Kharzeev, Henry Lamm, Ying-Ying Li, Junyu Liu, Mikhail Lukin, Yannick Meurice, Christopher Monroe, Benjamin Nachman, Guido Pagano, John Preskill, Enrico Rinaldi, Alessandro Roggero, David I. Santiago, Martin J. Savage, Irfan Siddiqi, George Siopsis, David Van Zanten, Nathan Wiebe, Yukari Yamauchi, Kübra Yeter-Aydeniz, and Silvia Zorzetti. Quantum simulation for high energy physics, 2022.

[4] Julian Bender, Erez Zohar, Alessandro Farace, and J Ignacio Cirac. Digital quantum simulation of lattice gauge theories in three spatial dimensions. *New Journal of Physics*, 20(9):093001, sep 2018.

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[6] Xinyu Fei, Lucas T. Brady, Jeffrey Larson, Sven Leyffer, and Siqian Shen. Binary control pulse optimization for quantum systems, 2022.

[7] Sanjib Ghosh, Tanjung Krisnanda, Tomasz Paterek, and Timothy C. H. Liew. Realising and compressing quantum circuits with quantum reservoir computing. *Communications Physics*, 4(1), may 2021.

[8] Erik J. Gustafson, Henry Lamm, Felicity Lovelace, and Damian Musk. Primitive quantum gates for an su(2) discrete subgroup: Bt, 2022.

[9] Daniel C. Hackett, Kiel Howe, Ciaran Hughes, William Jay, Ethan T. Neil, and James N. Simone. Digitizing gauge fields: Lattice monte carlo results for future quantum computers. *Physical Review A*, 99(6), jun 2019.

[10] J.R. Johansson, P.D. Nation, and Franco Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, 2013.

[11] Henry Lamm, Scott Lawrence, and Yukari Yamauchi and. General methods for digital quantum simulation of gauge theories. *Physical Review D*, 100(3), aug 2019.

[12] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528, 1989.

[13] Yiduo Wang, Bin Chen, Zhengqiu Zhu, Rongxiao Wang, Feiran Chen, Yong Zhao, and Laobing Zhang. A hybrid strategy on combining different optimization algorithms for hazardous gas source term estimation in field cases. *Process Safety and Environmental Protection*, 138:27–38, 2020.