Quantum control report

Tobias Rasmussen
*Department of Physics and Astronomy, Aarhus University, Ny Munkegade 120, 8000 Aarhus C, Denmark**
(Dated: March 18, 2021)

I. INTRODUCTION

In the quantum control problem called "shakeup", one works with a quantum state confined in a trapping potential. The state describes either a single particle or a Bose Einstein Condensate (BEC) state of matter and is at $t = 0$ in the ground state of the given trapping potential. The goal is to manipulate the potential using a *control function* in order to excite the state to the first excited state. This problem is explored and simulated using 2 different tools for simulating quantum mechanics. The first one is *Quantum Composer*, a visual program where a quantum system consists of interconnected nodes with different roles e.g. potential, time and space [1]. The second tool is the *QEngine*, a C++ library of high performance functions for simulating quantum mechanics [2]. One of the goals of this report is to use both tools to explore this challenge, mainly to test the limits of what is possible to do with the intuitive, but functionally more limited Quantum Composer, but also to find pros and cons of each tool and describe when to use what tool. Furthermore, based on those considerations it is an objective to formulate a list of suggestions which could make Quantum Composer a more attractive tool to use for expert-level work.

From a quantum physics perspective, the goal of exploring this challenge was to explore the different dynamics that arise when simulating a BEC compared to a single particle and how the dynamics depend on the nonlinearity of the BEC Hamiltonian, including how this affects the so-called Quantum Speed Limit. In addition, this report also explores if and how control functions can be clustered, how optimal solutions look compared to similar ones and how robust they are to system changes.

This report is structured as follows: Sec. II discusses some of the theoretical concepts relevant to the challenge. In Sec. III, a brief introduction to Quantum Composer and the workflow using it is provided. Sec. IV shows the results obtained using Composer when exploring this challenge and Sec. V explains the strategies used in Composer to obtain some of these results. In Sec. VI, some early results from using the QEngine library are shown and in Sec. VII, the 2 tools are compared by discussing their pros, cons and situations where using one, the other, or both are advantageous. The work described above is discussed in Sec. VIII. Finally, Sec. IX concludes the report by summarizing the results and presenting possible next steps for further exploration.

II. BACKGROUND

The mean field of a BEC of mass m, momentum p and interatomic interaction strength β (also called $g_{1\mathrm{D}}$) can be described using the Gross-Pitaevskii equation (GPE)

$$i\hbar\dot{\psi}(x,t) = \left(\frac{\hat{p}^2}{2m} + \hat{V}(x,t) + \beta|\psi(x,t)|^2\right)\psi(x,t) \quad (1)$$

Notice how β scales the nonlinear interaction term and that for the case $\beta = 0$, we recover the time dependent Schrödinger equation for a single particle. This nonlinear term represents interaction that takes place between all of the atoms in a BEC. The factor β is both dependent on the s-wave scattering length $a^s_{1\mathrm{D}}$ as well as the number of atoms N contained in the BEC [3]. It is also possible to tune this interaction parameter using a Feshbach resonance [4].

Depending on the sign of β in Eq. (1) the self interaction will be either attractive ($\beta < 0$) or repulsive ($\beta > 0$) and because of this, we would expect the state to respectively shrink or grow in size as a result of this interaction. This is also what can be seen in Fig. 1, where the ground state of a BEC is plotted as a function of β.

We attempt to excite the BEC from its ground state to first excited state using a control function, usually denoted $u(t)$. The control function describes how the potential evolves in time. A simple example of this is could be that $u(t)$ could describe how a quartic potential is displaced over time:

$$V(x,u(t)) = a(x-u(t))^2 + b(x-u(t))^4 \quad (2)$$

While it is not hard to write out *some* control function, it is much more difficult to find a control that succesfully excites the initial state into the desired state. To quantify how well a given control function has performed, one can calculate the *fidelity* $\mathcal{F} = |\langle\psi_D|\psi\rangle|^2$ to calculate the overlap between the two states $|\psi\rangle$ and $|\psi_D\rangle$, where $|\psi\rangle$ is the current state and $|\psi_D\rangle$ is some desired state. Thus, the fidelity $\mathcal{F}$ is a number between 0 and 1, where 0 means that there is no overlap between the 2 states and 1 means that the states are identical. Usually values of 0.99 and above are set as the benchmark that a control function must satisfy. For plotting purposes it is more convenient to instead show the *infidelity* $1 - \mathcal{F}$

* 201608265@post.au.dk

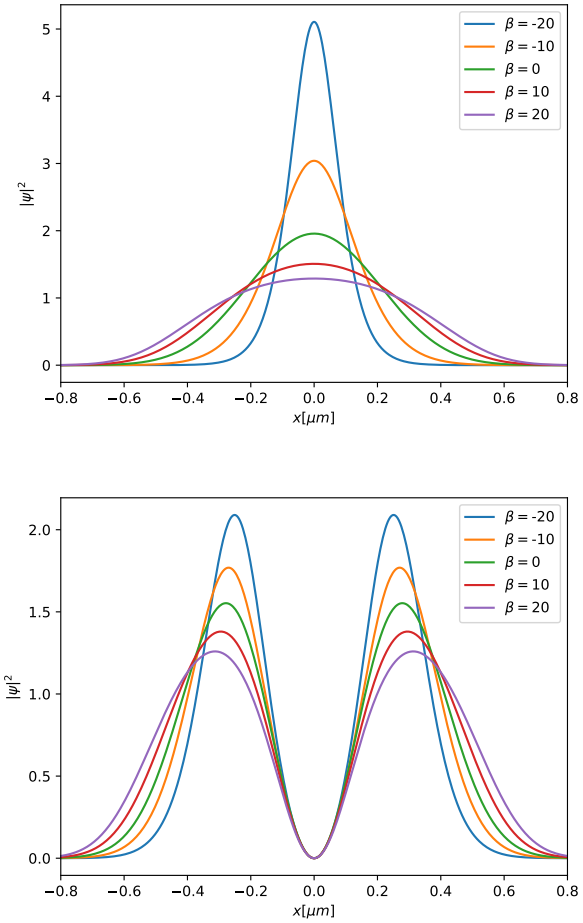


FIG. 1: **Top:** Ground states of a BEC in the quartic trapping potential seen in Eq. (2) with $u(t) = 0$ shown for different self interaction (β) strengths from Eq. (1). **Bottom:** First excited state of the same system. Notice that the state grows in size when $\beta > 0$ and shrinks when $\beta < 0$. Potential parameters are found in Appendix A under system *B1*, here with $n_{\text{points}} = 1024$.

on a logarithmic scale to illustrate how well different control functions performed.

Since it can be very difficult to find a control function that is capable of reaching $\mathcal{F} \geq 0.99$, it is often useful to *optimize* an initial control function (called a "seed" in this case) using different kinds of optimization algorithms. These optimization algorithms are, given enough time, usually capable of optimizing a seed into a control that can reach much higher fidelities. Regardless of which optimization algorithm is used, one must first define the optimization *problem*, which is a function that the given algorithm will attempt to minimize. We define this problem as [5]

$$\min J(u) = \frac{1}{2}(1 - |\langle\psi_t|\psi(T)\rangle|^2) + \frac{\gamma}{2}\int_0^T \dot{u}^2 \mathrm{d}t \qquad (3)$$

where ψ_t is the target state, $\psi(T)$ is the final state evolved with the control $u(t)$ to final time T. The first term minimizes infidelity and the second term penalizes rapid fluctuations of the control function that can be challenging to implement experimentally. γ is called the regularization factor and is usually on the order of 10^{-4} to 10^{-6}. Additionally, a third *boundary* term can be added to Eq. (3) with a factor σ which penalizes controls with values outside a defined interval $\{x_{lower}, x_{upper}\}$.

The optimization algorithm GRAPE (**Gr**adient **A**scent **P**ulse **E**ngineering) [6] is available to use in both Quantum Composer and the QEngine. GRAPE works by calculating the gradient of $\mathcal{F}$ with respect to each point of a given control function on the discretized time scale

$$\vec{\nabla}_{\vec{u}}\mathcal{F}[\vec{u}] \qquad (4)$$

where $\vec{u} = (u(t_0), u(t_1) \ldots, u(T))$. When this calculation is done, GRAPE takes a step of size α_k such that the new discretized control function becomes

$$\vec{u}_{k+1} = \vec{u}_k + \alpha_k \vec{\nabla}_{\vec{u}}\mathcal{F}[\vec{u}] \qquad (5)$$

such that $\mathcal{F}[\vec{u}_{k+1}] \geq \mathcal{F}[\vec{u}_k]$. This process is continued until some convergence criterium is fulfilled, e.g. a target threshold $\mathcal{F}[\vec{u}_k] \geq \mathcal{F}_{target}$, a maximum number of iterations $k \leq N$ or until the step size α_k becomes too small.

Two other optimization algorithms are available to use in the QEngine: GROUP and dGROUP (dressed GROUP). Where GRAPE is a relatively simply gradient ascent algorithm, these two attempt to combine the local optimization that GRAPE does with a global search for promising regions in the fidelity landscape [2, 7]. Note that this does not mean that an optimized seed will always be able to reach $\mathcal{F} \geq 0.99$. One particular hindrance is the *Quantum Speed Limit* (QSL). This speed limit refers to the upper bound of how fast a state is able to evolve in time [8]. This speed limit is (not surprisingly) dependent on the individual system examined and in the context of the Shake Up challenge, it refers to the minimum time required to evolve from the initial state into the target state with a fidelity of at least $\mathcal{F} \geq 0.99$.

The challenge was explored in Quantum Composer by creating different systems and control functions and optimization parameters. The control functions themselves were varied and explored, along with the control duration T and the GRAPE optimization parameters F_{target}, N, γ and σ.

III. QUANTUM COMPOSER

Quantum Composer is an educational tool for visualizing the behavior of a quantum mechanical system [1], either statically or as the system evolves in time. To do this, one combines building blocks (nodes) containing the various components (like a potential $\hat{V}(x)$) to build new nodes (like a Hamiltonian $\hat{H} = \hat{T} + \hat{V}$). From there, one can build a wave function from a superposition of $\hat{H}$ eigenstates using one type of node and connect it to a plotting node to have the state, its norm-square and other relevant quantities visualized. The system can be evolved in time using a time node. The use of Composer in this report can be grouped into three overall groups: System setup, GRAPE optimization and simulation. These groups and what they entice has been visualized in Fig. 2.

To simulate a quantum control problem, one first has to set up a system that describes the problem. This includes choosing boundaries of the Hilbert space and discretizing the space into a number of points. It also includes things like describing the potential of the system and the time interval (t_0, T). Specifically for quantum control problems it also involves defining how the potential depends on one or multiple control functions and how these are defined. From these steps, it is then possible to build the Hamiltonian of the system and from that we can create wave functions, in particular ψ_0 (the initial state, in this case the ground state of the BEC) and ψ_D (the desired/target state, in this case the first excited state) that are relevant for a quantum control problem.

It is then possible to optimize the initial control function using GRAPE. The optimization procedure requires a number of arguments, some optional (γ and σ), others required (N and F_T). The control function will then be updated until a convergence criteria is reached, after which GRAPE outputs the optimized control function. In Composer it is possible to map the fidelity of the current and previous control iterations to show how fidelity improves as GRAPE iterates. Likewise, it is also possible to plot the initial and current control function to see how they differ. A screenshot from Composer showing how this optimization procedure is set up can be seen in Fig. 3.

When the system is simulated, it is possible to analyze the wave function at the current point in time. Thus we are able to calculate and visualize different variables in our system as they are being simulated. This includes calculating the fidelity F as a function of time, but we can also plot how our wave function evolves in time by mapping out $|\psi(x,t)|^2$. When the simulation is finished, the final fidelity $F(T)$ can be recorded.

The data presented in Composer, in most cases, cannot be saved directly through the program, but must instead be manually written to a separate text file. This file containing data can then be further analyzed and plotted. In this case it is done using PYTHON and its plotting module `Matplotlib.pyplot`.

IV. RESULTS

A. Numerical limitations

1. Spatial resolution & performance

One obvious source of numerical inaccuracies is the resolution of the quantized Hilbert space one uses. Throughout the report, the spatial grid was divided into 256 points. We can compare this resolution to other resolutions by calculating the difference between a high resolution wave function with 1024 points and lower resolution wave functions. This difference between the different resolutions for the ground state can be seen in Fig. 4. As can be seen in the figure we find that the lower the resolution of a wave function is, the bigger the difference between that wave function and a higher resolution one.

Considering this, one would think that it would always be better to use as high spatial resolution of a given system as possible. This would of course be true if we had access to unlimited computational power. We instead have to balance the desire for a high spatial resolution and more accurate wave functions against the increased time it takes to do calculations for a higher dimension Hilbert space. As an example of this increased calculation time, a system was set up as described by System *B1* seen in Appendix A, with additional parameters: $\beta = 4$, $u(t) = 0.5\cos(2\pi t/T)$ and $T = 0.8$. This system was then optimized for 100 iterations using GRAPE before the system was simulated with the optimized control function. The results can be seen in Fig. 5. This result confirms that choosing a grid resolution of 256 points gives us a high accuracy compared to lower resolutions while also keeping performance time low.

2. Spatial limitations of controls

One could think, after having defined a spatial grid between 2 points $\{x_{min}, x_{max}\}$, that any given control function is then able to utilize the entirety of this grid. While the control function itself does not cause problems by roaming near the edges of the simulated space, the state being controlled does indeed cause big problems if parts of the wave "spill" out over the edge of the defined space. This phenomenon is demonstrated on Fig. 6, where the left state spills outside the space and "breaks" the simulation while the right shows a regular simulation. This can also happen when the control accelerates the state so rapidly that it can't slow the state down and "catch" it again.

To further investigate this problem, a simulation with no optimization was set up. The purpose of these simulations were not to find the highest fidelity, but to find

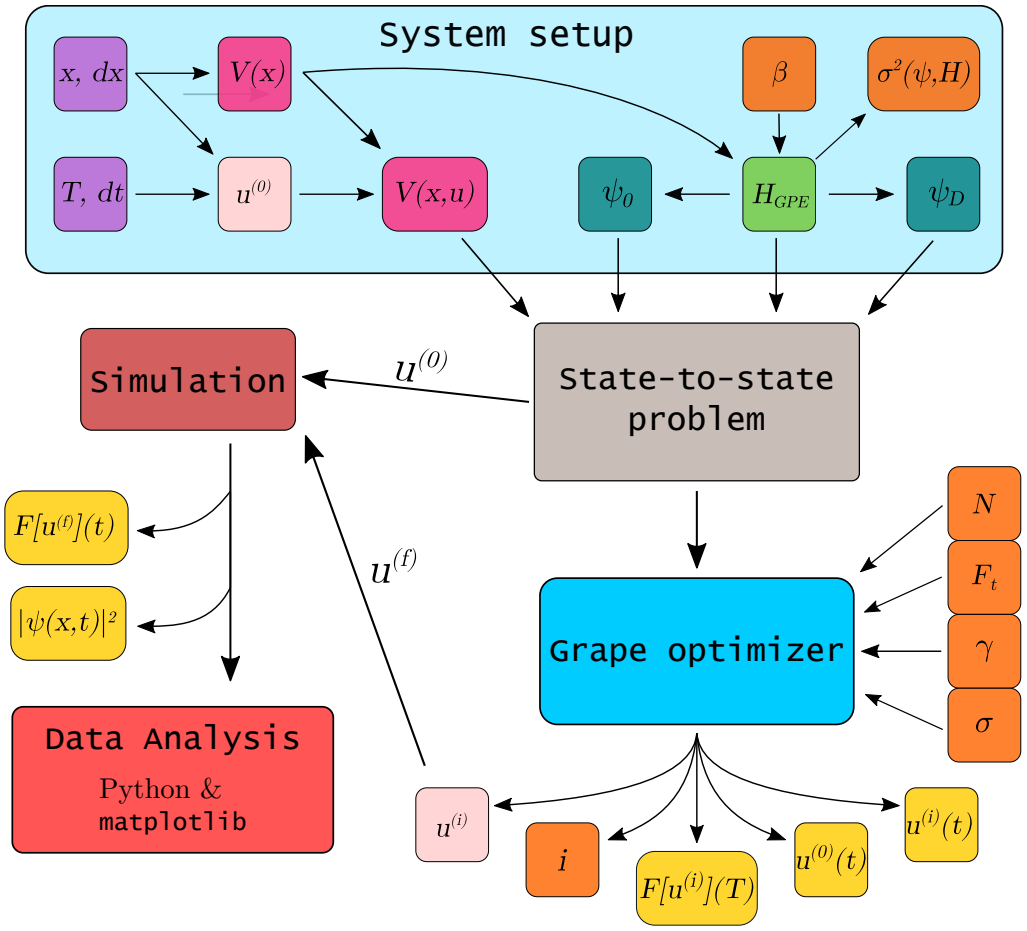


FIG. 2: Diagram illustrating the workflow of using Quantum Composer. The different phases of working are illustrated with the `Phase nodes`, which depend on a number of smaller nodes. Nodes are colored by the type of data they represent, e.g. scalars are orange, graphs are yellow, etc.. Arrows indicate dependencies on other nodes. Note that a node in the diagram does not necessarily translate directly to a node in Composer, although many do.

the highest amplitude A of the control function

$$u_{amplitude}(t) = A\sin(3\pi t/T) \tag{6}$$

possible, where the state did not spill over the space edges and break. For comparison purposes, this was experiment was repeated for three different spatial boundaries: $x \in \pm 2\,\mu\text{m}, \pm 4\,\mu\text{m}$ and $\pm 6\,\mu\text{m}$. The potential and self interaction strength β is noted in Appendix A under *Composer impl. of QM2 Shake up*. The results are shown in Fig. 7, which consists of 2 sub figures. The upper figure shows that for control durations below 1 ms, the highest amplitudes attainable are found in grids with boundaries $x \in \pm 4\,\mu\text{m}$ and for higher durations the most suitable bounderies are $x \in \pm 6\,\mu\text{m}$. The lower sub figure shows the ratio between the maximum amplitude found that the space boundaries. This shows that in terms of amplitude relative to space size, the 3 grid sizes each outperform the others such that increasing boundaries increases with control duration.

These results can be useful in cases where Composer is used as an exploration tool prior to an experiment. If the experimental boundaries are around $\pm 2\,\mu\text{m}$, it seems advantageous to use a boundary slightly larger for simulations with small timescales. This ensures that it is possible to fully explore the relevant control space.

It should be noted that while it has been demonstrated that it can be beneficial to increase the grid spacing, it comes at the cost of increasing the variance of important states (ψ_0, ψ_D). These settings should ideally be tuned to each simulation since it is possible to prioritize the trade off between performance and accuracy to a great extend. This trade off confirms what figures 4 and 5 demonstrate, discussed in the previous subsection.

3. *Eigenstates & eigenvalues*

In order to obtain accurate results when making these computations, one has to ensure that the states that are analyzed are described within acceptable numerical accuracy. We can ensure that we are working with the ground state as our initial state in Composer by calculating the

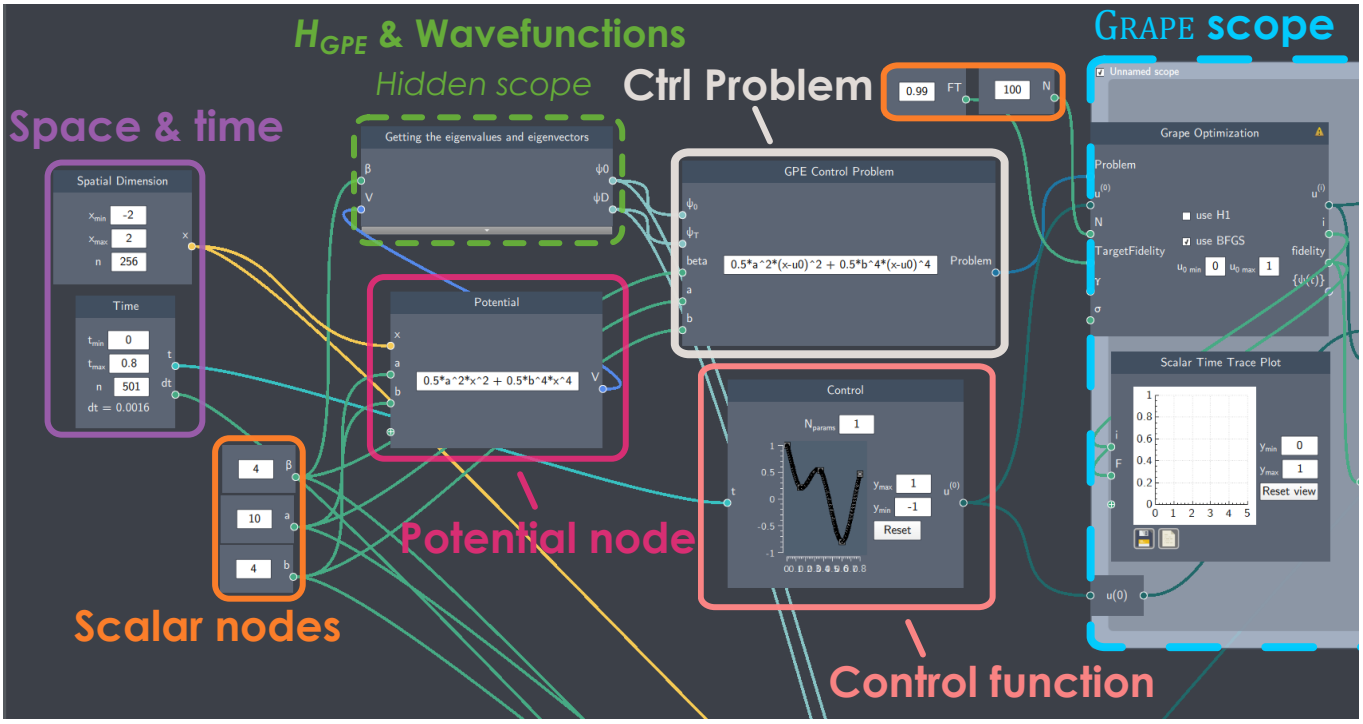


FIG. 3: Screenshot from Quantum Composer partially showing a flowfile that optimizes a control function using GRAPE. The nodes and scopes follow the same color scheme as in Fig. 2. Not shown is the scope where the optimized control is simulated and the resulting $\mathcal{F}(t)$ and $|\psi(x,t)|^2$ graphs are generated.

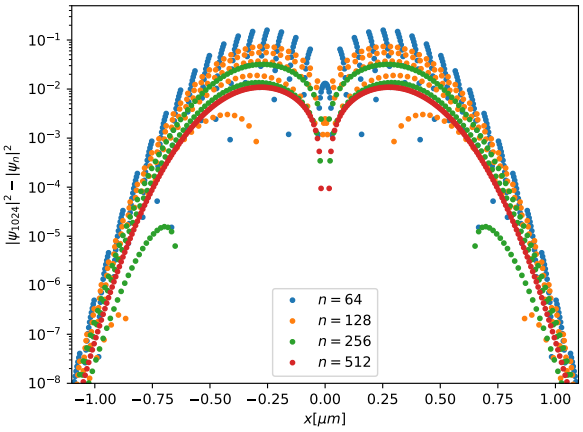


FIG. 4: Difference between different resolutions of the initial wave function compared to a higher resolution wave function with $n = 1024$. Lowering the spatial resolution results in increasingly larger error.

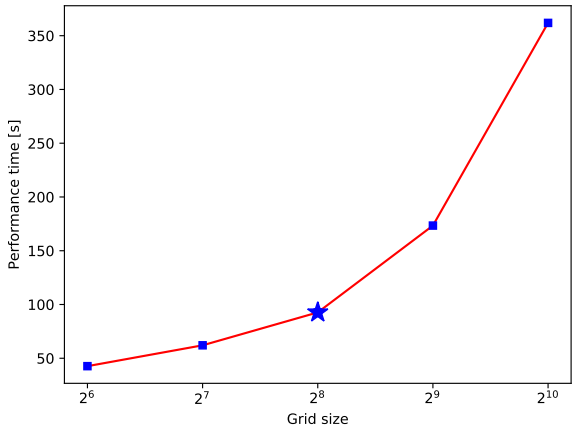


FIG. 5: The time it takes for Quantum Composer to solve a specific problem described in the main text as a function of the spatial grid resolution. The grid size of $n = 256 = 2^8$ was found to be optimal in terms of the accuracy vs. performance time.

variance of our initial state with respect to the Hamiltonian operator. The variance of a state with respect to an operator $\hat{\mathcal{O}}$ is given by

$$\mathrm{Var}(\psi, \hat{\mathcal{O}}) = \langle\psi|\hat{\mathcal{O}}^2|\psi\rangle - \Big(\langle\psi|\hat{\mathcal{O}}|\psi\rangle\Big)^2 \tag{7}$$

This gives us a numerical estimate of how "close" our initial state is to the actual ground state of $\hat{H}_{GPE}$ since

$$\psi \text{ is an eigenstate of } \hat{H}_{GPE} \iff \mathrm{VAR}(\psi, \hat{H}_{GPE}) = 0 \tag{8}$$

The way the energy spectrum of $\hat{H}_{GPE}$ is calculated is not trivial [2], but the variance can generally be

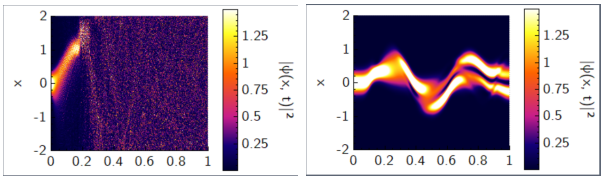


FIG. 6: Comparison of the state evolution for a numerically broken state (left) and a numerically working state (right). On the first axis is normalized control time and on the second axis position in μm.

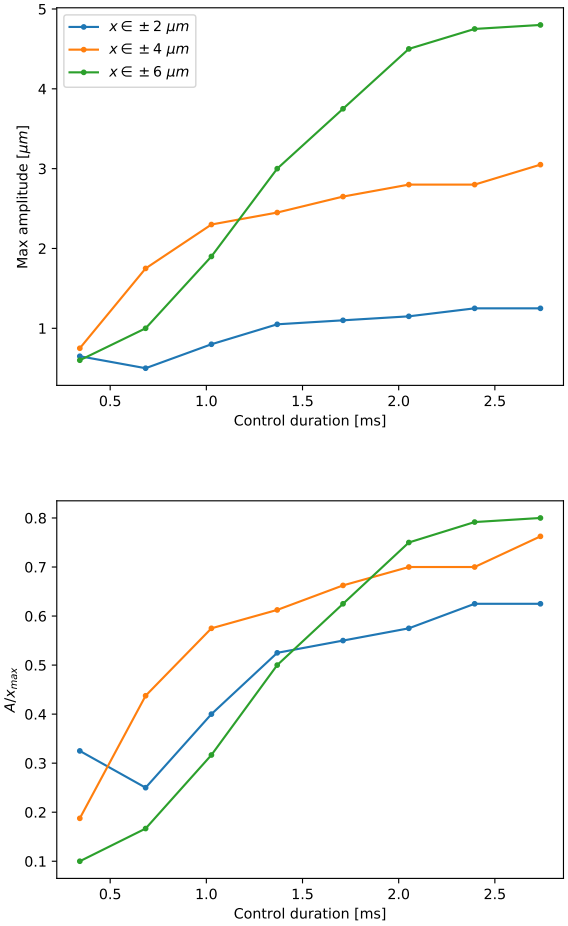


FIG. 7: Maximum possible amplitude A in the control function in Eq. (6) where the simulation does not break, simulated on 3 different grid sized $x \in \pm 2, \pm 4, \pm 6$. **Top:** Value of A_{max} as a function of control duration T. **Bottom:** Maximum amplitude relative to the maximum grid value. Bigger spatial boundaries makes the a bigger portion of the control space available for exploration. In some cases it seems viable to change the spatial boundary size depending on the control duration of the problem in order to maximize the possible control amplitude.

reduced by adjusting different parameters of the system or Composer file, e.g. spatial dimension x or initial conditions for the H_{GPE} spectrum node. Despite these possible fixes, the numerical limitations of Quantum Composer become apparent as the shakeup problem is explored in increasingly greater details and limits and accuracy are pushed further and further. The following paragraphs describe areas where the numerical limitations are significantly limiting the exploration of certain aspects of this problem.

Composer was also used to show how the energy levels of the BEC energy eigenstates changed as the self interaction was varied. The energy levels were studied both for a BEC in a quadratic potential (*System A1* in Appendix A) and a quartic potential (*System B1*). Figure 8 shows the how the energy levels of the 3 lowest energy eigenstates change with self interaction strenght. Below that it shows the energy spacing between these 3 levels for both potentials.

Firstly note that Composer is only capable of calculating the second excited state with great uncertainty outside a narrow interval around $\beta = 0$. This uncertainty can be reduced slightly by changing inititial parameter guesses in the spectrum generating procedure, but it is not reducible to negligible levels as for the uncertainties of the other states shown. It should also be mentioned that minimization of state variances was only done for the quardratic potential, which presumably explains the lower variances for these states compared to the states in the quartic potential. It is expected that the variance levels of the quartic system can reach similar lower values as the quadratic potential.

It should be noted how the self interaction strength β affects the spacing between the energy levels. For non-interacting and repulsive interactions between atoms we see that the spacing is still equally spaced. Due to the uncertainty of the 2nd excited state energy, we are unable to actually conclude if this behavior is true and so we can also say that these results indicate that the equal spacing also holds for repulsive BECs. This is not the case for BECs with attractive self interaction, where the energy spacing between the 2 lowest states rapidly increases compared to the spacing between the 2 highest states. This increase in spacing is so large that it even surpasses the considerable uncertainty of the difference between 1st and 2nd excited state.

B. Harmonic potential optimization

Due to the equally spaced energy levels of a quadratic potential, it is difficult to excite the state exclusively to the first excited state. It is likely that some part of the state is driven into higher excited states and so it is seemingly impossible to reach high fidelities with this kind of potential. Nonetheless, the nonlinear

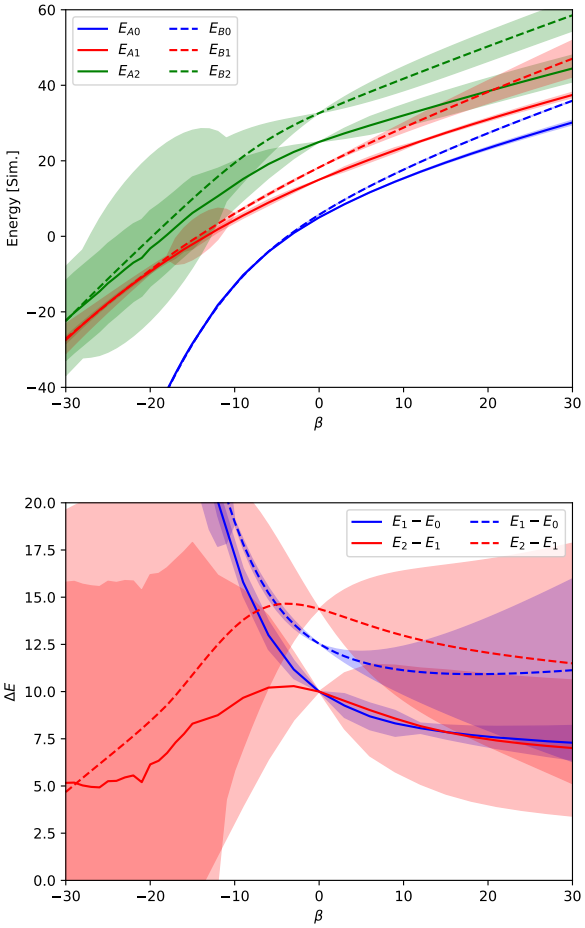


FIG. 8: **Top:** Energies of the ground state and first 2 excited states of a BEC with self interaction strength β for a quadratic (full drawn lines) and quartic (dashed lines) potential with $u(t) = 0$. **Bottom:** Difference between state energies of the energies shown above. Uncertainties marked show the square root of the variance in both figures. Notice in particular that the equally spaced energy levels of the quardratic potential at $\beta = 0$ remains somewhat equal as the interaction becomes more repulsive, while the distance grows rapidly with a more attractive self interaction.

contribution to the GPE Hamiltonian seems to make it possible to improve the fidelity of control functions compared to the levels reachable for a single particle. In a quadratic potential different controls were optimized to estimate the highest reachable fidelity for this kind of potential as a function of the self interaction strength β. These controls were optimized with GRAPE for 100 iterations and the highest reachable fidelity was then noted. The potential parameters are described in Appendix A under *A1*.

The results can be seen in Fig. 9. As can be seen on the figure, it is actually possible to reach higher levels of fidelity for all measured values of $\beta \neq 0$ with control duration $T = 1$. The highest result shows that it is possible to improve the fidelity score with about 50% compared to the reachable score of a single particle. One should also note that controls with duration $T = 7.5$ do not perform as well as the other 2 shown durations. This is suspected to be caused by the long control duration allowing for states to be excited beyond the targeted first excited state, which the smaller control durations are not as susceptible to. Another explanation could be that the increased control duration makes the optimization landscape more complex and so more sensitive to the inital control that is being optimized on.

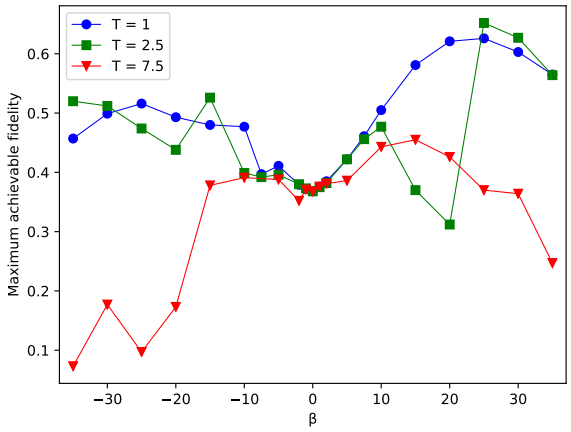


FIG. 9: The result of varying β and T for a BEC in the harmonic potential. There exist combinations where it is possible to achieve above 50% higher fidelity compared to a single particle (at $\beta = 0$), but many also combinations also perform worse than for a single particle.

C. Unoptimized BEC control in a quartic potential

Through simulations it seems that BECs have very different dynamics compared to single particles. Without using GRAPE, one can study how the similarity between single particles and BECs changes depending on the value of β. Keeping the control function and control duration the same, we find that what was a good solution for a single particle reaches a lower and lower fidelity as we change β away from 0, but that it also reaches revivals with a higher fidelity compared to neighboring values. This behavior is shown in Fig. 10 where the non-optimized control

$$u_{\text{non-opt}}(t) = 0.12 \cdot \sin(0.86 \cdot \omega_{01} t) \tag{9}$$

with final time $T = 3.49$ was simulated for different values of β. The potential parameters are described in Ap-

pendix A under *B1*. We define ω_{01} to be the energy frequency between the ground and first excited state. This variation is not unexpected. The function in Eq. (9) was chosen because of its high fidelity score around $\beta = 1$, but expecting this control function to perform well for the wide range of other systems (i.e. values of β) it is tested on would not be expected. While the behavior of similar systems is not vastly different, this small change to the system dynamics does mean that some systems will not reach a high fidelity at the final time, while others will. This can also be seen in Fig. 11, where the fidelity as a function of time has been plotted for 2 different interaction strengths. In conclusion, using a single, non-optimized control for a broad range of systems is not viable, as success boils down to timing whether or not the control terminates at a high fidelity.

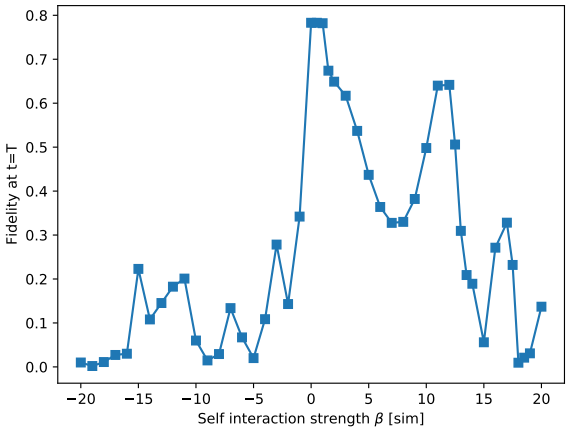


FIG. 10: Fidelity of a non-optimized solution for different values of β in the quartic potential from Eq. (2) with $u(t) = 0$. The fidelity drops off as β is moved away from 0, but there are also revivals where the fidelity reaches a local peak. These revivals can be explained by looking at the fidelity as a function of time, two examples of which are shown in Fig. 11.

D. Quantum speed limit

It was attempted to estimate the Quantum Speed Limit for this control problem and its dependence on the interaction strength of the BEC, β. The system parameters used are described under *System B1* in Appendix A. The methods for determining seeds are discussed in section V. The seeds chosen were then optimized for 100 iterations in order to estimate the duration above which optimized controls reach a fidelity of 0.99. The resulting estimates of the QSL for different values of β can are shown in Fig. 12. Notice how it seems that the speed limit is lower for attractive BECs and increasingly higher for increasingly more repulsive BECs. It was not

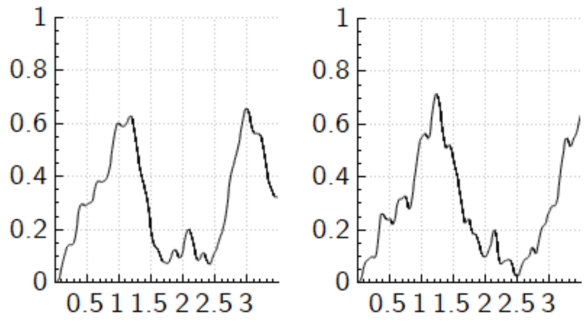


FIG. 11: Fidelity as a function of time for the control function in Eq. (9) for two different values of β. **Left:** $\beta = 7$. **Right:** $\beta = 12$. As can be seen, the overall trend is the same for the 2 systems and so it becomes a matter of "timing" the final time to reach a higher fidelity if one refrains from using GRAPE to optimize controls.

possible to estimate the QSL for values of β below -10 or above 25, since no control function could be optimized to 0.99 within 100 iterations. The speed limit was found to be lowest at the values $\beta \in \{-5, 0, 0.5, 1\}$ where the speed limit was estimated to $0.63\mu_{time} \approx 0.86$ms.

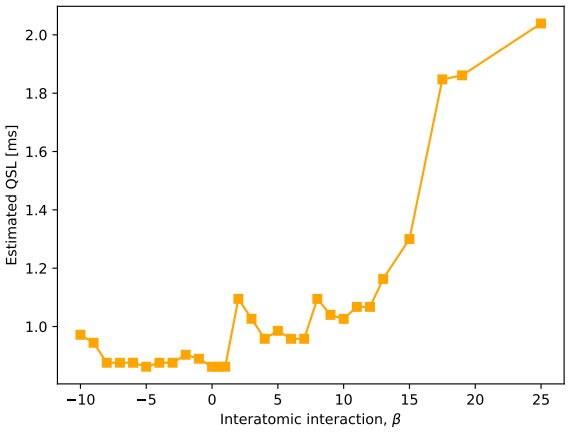


FIG. 12: The estimated Quantum Speed Limit for a BEC in a quartic potential as a function of β. The speed limit was found to generally increase for $\beta > 0$, while it was found to be around the same value for $\beta \leq 0$. Note that the highest speed limit found at $\beta = 25$ was more than twice as high as the lowest found around $\beta \sim 0$.

E. Optimizable lower bound of β

It is clear after having explored the behavior of BECs in different numerical setups that the behavior the BECs

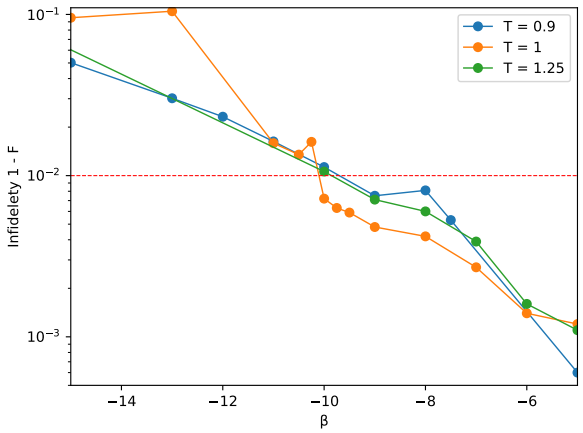


FIG. 13: Lowest reachable infidelities as a function of interatomic attraction β, shown for different timescales. It seems that $\beta = -10$ is the lower limit for reaching optimizable solutions $F \geq 0.99$. This lower limit is, however, suspected to be caused by numerical inaccuracies. The red line marks where the fidelity of a control is 0.99.

are not symmetric with respect to β. This is not unexpected, since the sign of β constitutes whether the atom-atom interaction happening inside the BEC is repulsive($\beta > 0$) or attractive ($\beta < 0$) and so it would not be unjustified to say that the sign of β describe entirely different species of bosons. Since attractive BECs have proven difficult to optimize on for $\beta < -5$, it was explored what the lower limit of β would be with respect to reaching an optimizable fidelity of at least 0.99. The results of this exploration is seen in Fig. 13, where it can be seen that for values of $\beta < -10$ it is no longer possible to reach $F \geq 0.99$. While this lower bound could be seen as a numerical limitation of Composer as discussed in Sec. IV A, it is also relevant to consider that there exists a lower bound for how attractive a BEC can be before the atoms collapse in on themselves in a process known as a Bose-Nova [9].

F. Clustering of solutions

The shakeup problem has also been studied in a recent article [10]. It was found that optimized controls could be clustered by calculating their projection onto cosine functions of increasing frequencies

$$c_k = \frac{1}{T}\int_0^T [u(t) - \langle x(t)\rangle]\cos(k\pi t/T)\mathrm{d}t \qquad (10)$$

where $k = 0\ldots5$. The resulting clusters indicated that the best solutions were dominated by a specific cosine frequency k and that k increased as the control duration T increased. The figure from Ref. [10] describing this is shown in Fig. 14.

To replicate this result using Quantum Composer, the system conditions had to be rescaled to Composer, where the kinetic factor κ is locked at 0.5, while in the original system it was instead 0.36537. Thus, the different quantities of the system had to be reworked. This recalculation can be seen in Appendix A.

It was attempted to replicate this result by letting GRAPE optimize seeds of the form $u(t) = A\cos(k\pi t/T)$ for 750 iterations each. By varying k and A, the frequency and amplitude of the seeds were varied and their optimized solutions and reached fidelity noted. The results can be seen in Fig. 14. They do not replicate those found in Ref. [10] where solutions dominated by different values of k outperform other k's for some interval. Note in particular that the overall highest fidelity was reached for a seed with frequency $k = 2$ at $T \approx 0.93$ms, where $k = 5$ was expected to be the best performing frequency. However, note that both $k = 4$ and $k = 5$ reach solutions that outperform the other seeds for that particular timescale and that they do so in the expected order (4 before 5).

As is also shown on the figure, seeds with different frequencies do not have the same amplitude. It was found that having a too high amplitude caused the state to become numerically unstable and the simulation to not function correctly as described in Sec. IV A 2. This is also the reason that the $k = 4$ data spans a smaller time interval due to numerical errors in their solutions for $T > 0.8ms$.

While it is currently not possible to calculate each feature c_k with composer since the control functions cannot be extracted, it is still possible to analyze the screenshots of the controls. One way of classifying the controls using the screenshots is to count the number of 0-crossings the control performs. Since cosine functions of increasing frequencies perform an increasing number of 0-crossings, we would expect that optimized controls perform a number of 0-crossings proportional to the control duration. An example of this clustering method is shown in Fig. 15 where two similar seeds are compared and the optimized control with highest duration has more 0-crossings.

All optimized solutions shown in Fig. 14 were analyzed as well as some not shown results for the parameters $k = 3$, $A = 0.5$. Some data was not included as the optimized control screenshots were not saved. The results of counting the number of 0-crossings for each control are shown in Fig. 16. The results seem to indicate that as the control duration is increased, so is the number of 0-crossings of the best performing solutions.

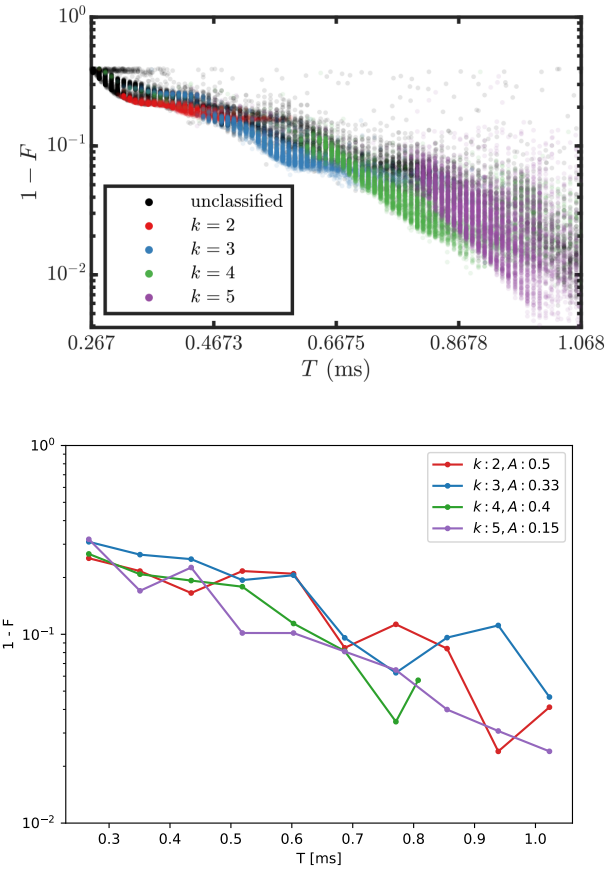


FIG. 14: **Upper**: This figure is taken from [10] and shows how optimized controls can be clustered by their projection onto $\cos(k\pi t/T)$. **Lower**: Infidelity of GRAPE optimized solutions with initial control $u(t) = A\cos(k\pi t/T)$. The first axis shows the control duration T.

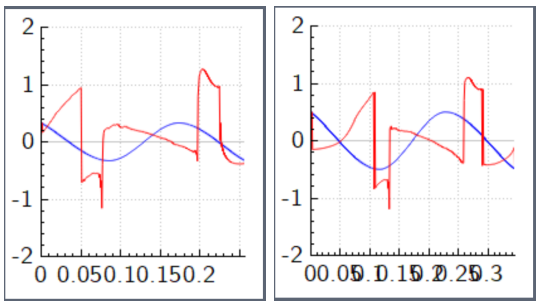


FIG. 15: Screenshots of optimized controls of a similar seed ($k = 3$) for 2 different durations and amplitudes. The axes show control displacement $u(t)$ as a function of time. **Left:** $T \approx 0.35$ms, $A = 0.33$. **Right:** $T \approx 0.47$ms, $A = 0.50$. The optimized seeds are very similar in terms of shape, but differ in the number of 0-crossings, where the left control has 5 and the right has 7.

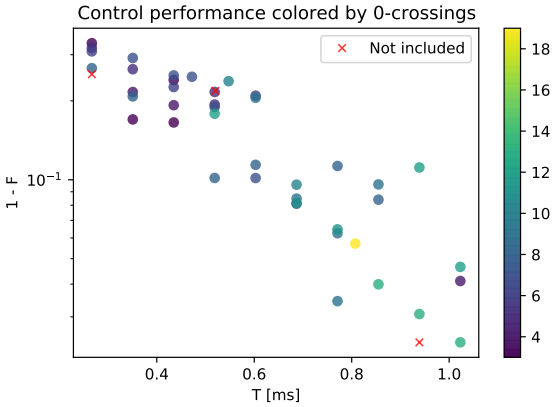


FIG. 16: The performance of the optimized controls from Fig. 14, colored by the number of 0-crossings. The analysis indicates that the number of 0-crossings in optimal controls increases with the control duration.

G. Robustness of solutions

The robustness of optimized solutions with respect to self interaction strength or potential scaling was studied in an article [7]. This is interesting since real experiments are often subject to the inherent uncertainties of their equipment e.g. the frequency width of a laser. Since the self interaction in a BEC is dependent on the number of atoms it consists of, it is not surprising that a specific strength β is difficult to replicate exactly in the lab and may at the same time vary in time. Likewise the trapping potential is created by standing waves of laser light and so it is also subject to some uncertainty in intensity and frequency.

The robustness of optimized solutions was explored for the system *Composer implementation of Quantum Moves 2* described in Appendix A. This replication was done by letting GRAPE optimize the same control seed, $u(t) = 0.15\cos(5\pi t/T)$, for 300 iterations and for different durations $T = 0.5$, $T = 0.875$ or $T = 1.3$, noting that these durations are respectively below, near and above the QSL of this system. These optimized solutions were then replayed in a system with a modified interaction strength $\tilde{\beta} = a_I\beta$ or in a system with a modified potential $\tilde{V}(x, u(t)) = a_V V(x, u(t))$.

The results can be seen in Fig. 17. There seems to be an overall trend where the more optimal a solution is, the more sensitive it is to small changes in either the potential or the interaction, in terms of fractional change from the unperturbed case. Keeping in mind that better solutions are reached for longer durations, this could also be intepreted as how the 'fidelity landscape' becomes increasingly more complex the longer the control duration is. This could make sense since for small durations below the QSL, there could be many 'plateaus' near optimal fidelity. Small changes to the system would

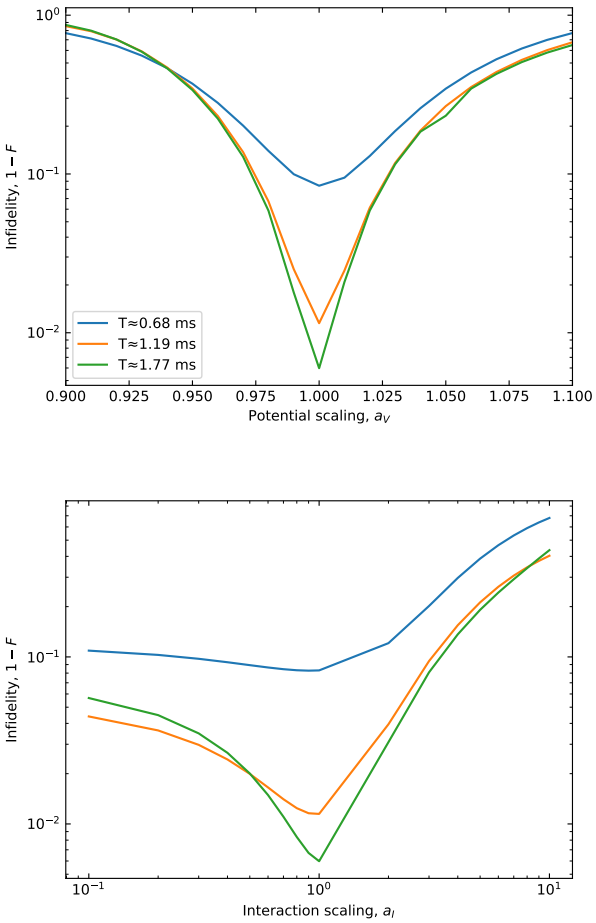
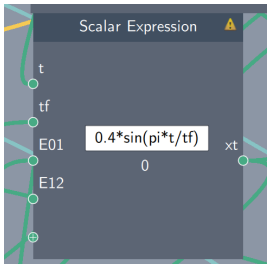


FIG. 17: Examining the robustness of an optimized solution. **Top:** By scaling the potential. **Bottom:** By scaling the BEC self interaction. The three control durations are chosen to be below, near and above the QSL of this system. The higher the control duration, the more relatively sensitive the optimized control is to noise.

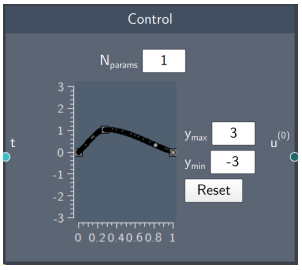
change the shape/height of the plateau, but if the changes are small then the optimized solution should still be somewhere on this plateau. On the contrary, for long durations above the QSL, the landscape is more "spiked" due to some specific solutions being able to reach very high fidelities and this spiked landscape would of course be more affected of changing parameters.

V. STRATEGIES

Throughout working with this project, there has been two ways to determine the control function $u(t)$ in Composer: The control function is determined either by an analytical expression (using the Scalar expression node) or it is made by dragging the desired "path" of displacement on a graph (using the Control node). These different nodes are shown in Fig. 18. One can use both of these approaches when working with quantum control and they both have their advantages and disadvantages.



(a) Scalar expression node



(b) Control graph node

FIG. 18: Both nodes used for making a control function in Quantum Composer. The analytical function method (a) can take a number of input constants like final time $tf = T$, energy spacing between different states ($E01 = E_1 - E_0$) and variables such as time t. The graph method (b) always scales the first axis such that the first point on the graph is at $t = t_0$ and the final point is at $t = T$.

Using an analytical expression as a control function, one gets the possibility of tweaking initial parameters to great accuracy such that the seed GRAPE gets already has a high fidelity. It is preferable to the graph approach in cases where a desired oscillation frequency is wanted e.g. the function $A\sin(E_{01}t)$. These kind of functions are difficult to mimic in the graph approach. When working with expressions independent of the final time T, adjusting this parameter translates into cutting the graph at different times. This is useful when exploring different functions and one finds that higher fidelity occurs at a different time than one currently works with. This method is shown in Fig. 19.

The Control graph node makes it possible to build up the control function as a series of steps. Here one can move one point on the graph and watch how the state evolves until that point, making it possible to optimize each step before going further. Since the graph node always spans the entire time interval $\{t_0, T\}$, this makes using the node useful when wanting to stretch or contract the entire function as shown in Fig. 20. As demonstrated with the BEC in a harmonic potential in Fig. 9, there exists a complex relationship between β and T and so being able to keep the shape of the control function the same while altering the total time makes the node a beneficial tool to use.

When calculating the maximum reachable fidelity for some combination of β and T in the harmonic poten-

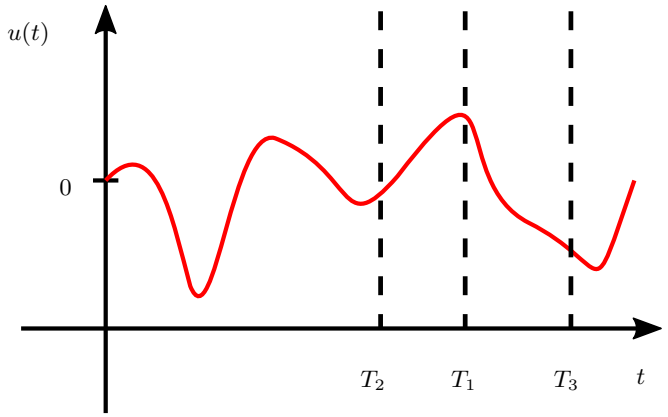


FIG. 19: How an analytical control function can be used to reach a higher fidelity seed by altering the final time T. In this example we use the control function shown in red and the final time T_1. Evolving the system for a shorter (longer) time can yield a higher fidelity, and we can reach these fidelities by replacing the final time with T_2 (T_3). This method only works if the control function is independent of the final time.

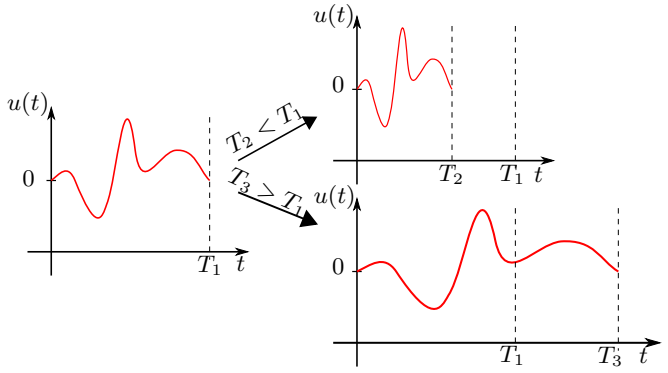


FIG. 20: One method to optimize fidelity with the graph node. Changing the control time T will squeeze or stretch the control function. Altering the final time can be beneficial given the complex relationship between β and T, but also as a tool to regulate the speed at which the potential moves. Using this approach also allows us to fix the control end points, e.g. $u(0) = u(T) = 0$ for any T. This should be compared to the analytical function approach shown in Fig. 19 where this is not always the case and the state could possibly need to be transported back to its initial position.

tial, the graph control node was used to seed the GRAPE algorithm. The overall strategy using this tool involved moving points on the graph into certain shapes and looking at how the initial (i.e. no GRAPE) fidelity looked. I particularly liked the semi-circle shape of a $\sin(\pi t/T)$ function where the amplitude was varied. Another particular shape used in this work was generated by moving a single point around $0.1T$ or $0.9T$ to create a function which had a fast(slow) buildup to a maximum and then a slow(fast) return to the initial position. This wedge-like shape is shown in Fig. 18 (b).

Both of these strategies worked well and provided nice seeds for GRAPE to optimize on. Despite some cases where the fidelity changes significantly from small variations of β, the overall rule seems to be that something that works well for one value of β will have a similar fidelity for nearby values of β, and so this was also used to find a good seed when β was larger than 10, since above this limit the system seemed a lot more sensitive to the initial control function with respect to reaching the highest possible fidelity for that particular combination of β and T.

Measuring the Quantum Speed Limit for the quartic system (*B1* in Appendix A) involved a combination of 2 strategies: The self-dubbed "up-to-down" and "down-to-up" approaches.

UP-TO-DOWN:

1. Pick T_{up} to be a control duration estimated to lie above the QSL of the system
2. Find an optimizable solution capable of reaching F 0.99
   - If unable to do so, return to Step 1 and choose a higher T_{up}
3. Gradually lower the duration in small steps, ensuring at each step that the control can still be optimized to F 0.99
   - If the control is unable to reach F 0.99, one should try minor modifications to the control to see if this fixes it. This could be dragging a point on the control graph slightly up / down to increase fidelity.
4. The lowest reachable T is then the estimate of the QSL for the system. These steps can be repeated for another control function.

The UP-TO-DOWN procedure was my main method for estimating the QSL of the system shown in Fig. 12.

DOWN-TO-UP works in a similar way to UP-TO-DOWN, but in a reversed way.

1. Pick T_{down} such that it is estimated to lie below the QSL of the system.
2. Try a number of different control functions and choose the most promising (i.e. the seed capable of reaching highest fidelity)
3. T is raised in small steps, at each step optimized and modified to check if F 0.99 is reachable
4. The first time F 0.99 is reachable, the current T is the estimated QSL for the system.

This procedure attempts to exploit that the optimization landscape for GRAPE to traverse is hopefully simpler due to the lower control duration that is then gradually raised in the hopes that the seed has found a potential fidelity peak in the landscape that becomes available for some slightly higher control duration.

While reproducing the clustering of optimized controls data seen in Fig. 14, it became apparent that using cosine functions instead of sine functions could be advantageous for small timescales. One can use the fact that cosine is displaced from 0 at $t = 0$ to one's advantage as it initially places the state on a steep slope of the potential. This generates motion of the state much faster than when initiating with a non-displaced control function. Similarly when the control is done, one can displace the potential such that the state feels rapid deceleration, thus reducing the velocity of the state much faster than if the control was constrained to end at $u(T) = 0$. This behavior can also be seen on the optimized controls shown in Fig. **??**, where both end points experience rapid displacement away from $x = 0$. These advantages are similar to the "back-swing" solution for another quantum control problem described in [10].

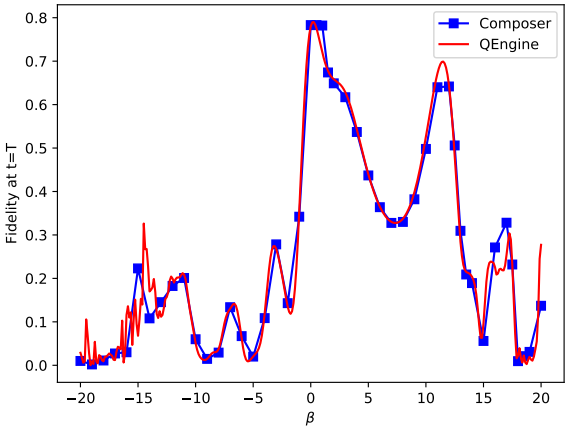


FIG. 21: Comparison of the results obtained using Composer and the QEngine where both tools simulate the unoptimized control from Eq. (9) with a control duration of $T = 3.49$. The two tools reach the same or very similar results for most of the self interaction values, which is also expected as Quantum Composer uses the QEngine for computations. The parts where the 2 tools disagree are suspected to be caused by numerical inaccuracies.

VI. QENGINE

This section describes the work done using the QEngine, a C++ library which Quantum Composer is build on top of [1, 2]. Instead of the GUI that Composer offers, it is library of efficient methods for simulating quantum mechanics. This makes it vastly more flexible and customizable than the graphical interface built onto it. The QEngine library comes with a number of example files that can be modified to set up a desired system and e.g. a control function with optimization using one (or all) of the three available optimizers. The `DataContainer` class can be assigned to save variables, control functions, etc. and can be saved in either `.mat` or `.json` format to later be read by either MATLAB or anything else, respectively. Similarly to the Composer case, the data gathered using the QEngine is further analyzed and plotted using PYTHON.

To start off an exploration into the advantages that QEngine offers compared to Quantum Composer, a previously examined problem is replicated using the QEngine. The replicated problem is the periodically shaken potential shown in Fig. 10 and the QEngine results are included in Fig. 21. The two tools reach the same or very similar fidelities for simulations where $\beta > -13$. Below this limit, the results follow the same trend, but it appears that the system becomes very sensitive to the final time.

A. Clustering of controls

In the following it is demonstrated how the QEngine exceeds Quantum Composer as a data gathering tool. The system is set up in the exact same manner as the system described in Sec. IV F. A seed is set using an analytical expression. From this initial seed, a number of new seeds n are generated by adding random noise to each point of the discretized control. Each of these noisy seeds are optimized, some using only GRAPE, others with GROUP and dGROUP as well.

The resulting infidelity of these optimized solutions are shown in Fig. 22, where the best results from Composer in Fig. 14 have been included for reference. From each initial seed with added noise, it is possible to calculate many more datapoints and optimize seeds to reach higher fidelities than what is possible in Composer, even when GRAPE does the optimization. This is most likely a result of having a high iteration cap $N_{max} = 3000$ on the optimizer in the QEngine. This out performance of Composer does come at the cost of a considerable computation time, though. The exact settings and seeds used are listed in Appendix B.

It was not possible to replicate the clustering of optimized controls similarly to what was done in Ref. [10] using Quantum Composer as mentioned in Sec. IV F. This is, on the other hand, possible using the QEngine, where optimized controls and expectation values are extracted using the `DataContainer` class.

The optimized controls shown in Fig. 22 were chosen

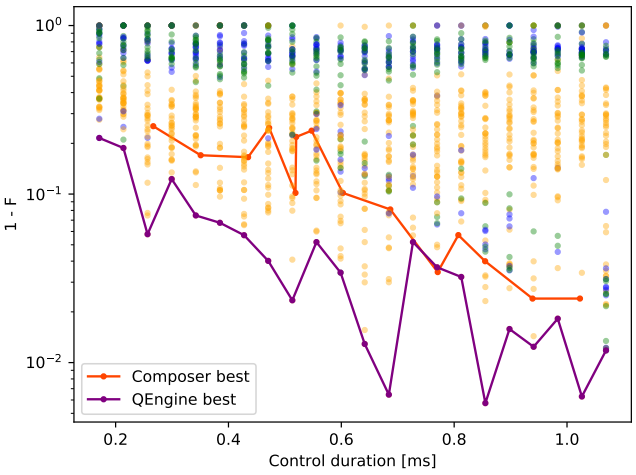


FIG. 22: Infidelities of solutions optimized using each of the 3 optimizers available in the QEngine: GRAPE (yellow), GROUP (blue) and dGROUP (green).

if $\mathcal{F}[u] \geq 0.70$ and clustered by calculating a vector of features $\vec{c} = (c_0, \ldots, c_5)$ using Eq. (10). The feature vectors were clustered using the DBSCAN method from the `sklearn` Python module with parameters $\epsilon = 0.05$, $\text{min}_{\text{samples}} = 10$ and Euclidean metric. This is the same clustering method using in Ref. [10]. DBSCAN works by finding *core samples* defined as containing $\text{min}_{\text{samples}}$ other points withing a distance of ϵ of them in feature space. A collection of these core samples are then considered a cluster. The results of the clustering are shown in Fig. 23.

Based on the results shown, we can conclude a number of things. Firstly it is clear that the clusters found are not concentrated around the lowest infidelity regions, but instead span a large duration and infidelity interval. In order for the algorithm to find clusters in the optimal time-infidelity regions, a much higher density of data points are needed as seen in the original figure from Ref. [10].

If we focus on the clusters that have been found, we see that cluster mean feature vectors have identified clusters dominated by different cosine frequencies. From these clusters we see that Cluster 0, mainly consisting of controls in the upper half of the durations simulated, is dominated by the highest cosine frequencies. Thus, we find the same trend as described in Ref. [10] where the optimal solutions at higher durations could be characterized by a higher frequency.

Considering the relative controls $u(t) - \langle x(t) \rangle$ that emerge, we see that Cluster 1 and 2 (blue and green on the figure) behave similarly. They both begin with $u(t) \neq 0$ and oscillate with a higher relative amplitude than Cluster 0, which is not displaced at $t = 0$ and generally has a lower relative amplitude.

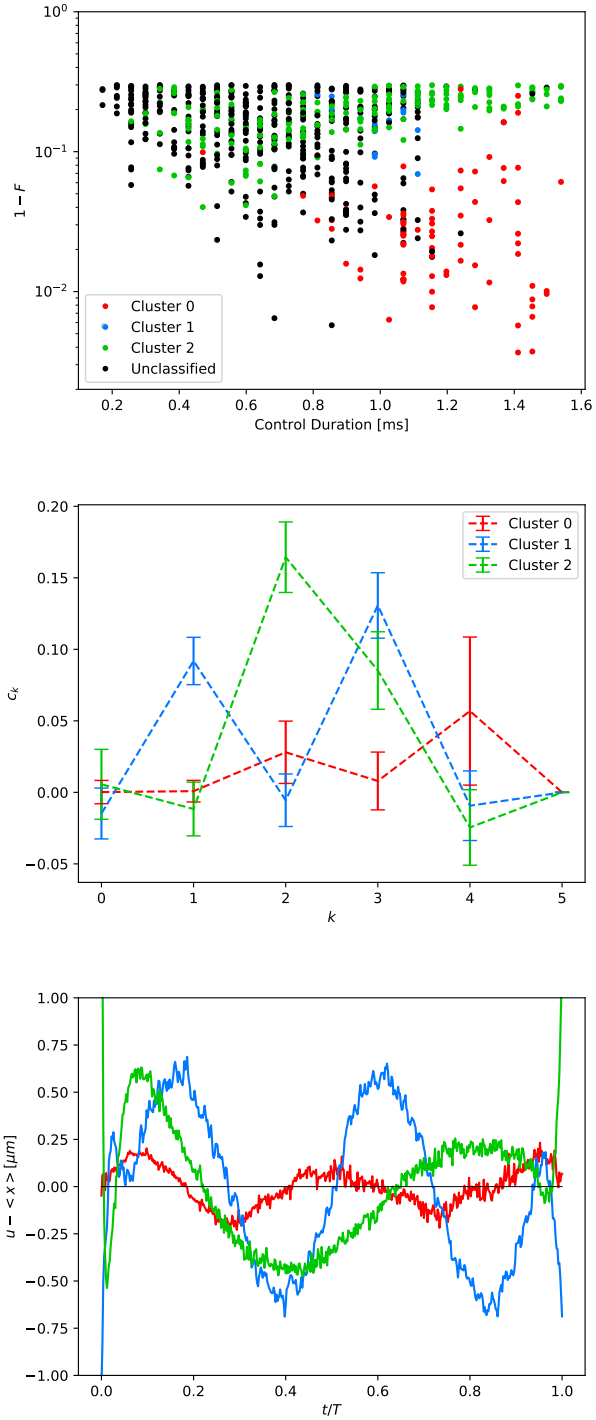


FIG. 23: Clustering of the data presented in Fig. 22 in a similar way to what was done in Ref. [10], seen at the top of Fig. 14. **Top:** Time vs. infidelity plot showing which points belong to which clusters. Black points have not been assigned a cluster. **Middle:** Mean feature vectors of each cluster. Errorbars show the standard deviation. **Bottom:** Average motion of $u(t) - \langle x(t) \rangle$ for each cluster. The thin black line marks $u(t) = \langle x(t) \rangle$ for visual aid purposes. Note that cluster 2 (green) begins and ends at values above 1 µm.

B. Landscape exploration

In a recent group meeting presentation [11] it was attempted to describe the control landscape of a 1D chain of 5 spin-1/2 particles. This was done by calculating a number of characteristics based on many optimization runs: Infidelity, mean step size, no. of unique optimized controls and the mean of the Hessian diagonal. Infidelity offers insight into the height of the landscape and the number of local "traps" in that part of the region. A trap is a local maximum named so since it can "trap" a gradient based optimizer and keep it from reaching the global maximum. The mean step size could hopefully describe how "flat" the landscape is, i.e. a large (in this case around unit length) step size means that we get a similar fidelity score from a control that is similar to the current one. The number of unique controls show the number of different solutions that are found in that simulation. The Hessian diagonal describes tries to describe how sharp the found peaks are in the control space.

A control $u(t)$ is in this case different from another control $v(t)$ if the Euclidean norm between the two controls is larger than ϵ

$$u \text{ different from } v \Leftrightarrow \left(\sum_i [u(t_i) - v(t_i)]^2\right)^{1/2} > \epsilon \quad (11)$$

with $\epsilon = 10^{-16}$.

The Hessian diagonal is calculated numerically using finite differences [12], which states that given a grid with uniform spacing h, the second derivative of a function f at a point x can be approximated to second order in accuracy by

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \quad (12)$$

In practice the Hessian diagonal was approximated using the following procedure.

1. Extract optimized control $\vec{u}$ and fidelity $\mathcal{F}(\vec{u})$.
2. For each timestep in $\vec{u}$:
   (a) Obtain $\mathcal{F}(\vec{v}_{\pm}^{(i)})$ with $\vec{v}_{\pm}^{(i)} = (u_0, \ldots, u_i \pm h, \ldots u_N)$ by simulating the control.
   (b) Approximate the second partial derivative $\frac{\partial^2}{\partial u_i^2}\mathcal{F}(\vec{u})$ using Eq. (12).

The QEngine was set up to run 25 iterations for each final duration. The GRAPE optimizer was set up with parameters $N = 1000$, $\gamma = 10^{-5}$ and boundaries $u(t) \in \pm 2\,\mu\text{m}$ with weight $\sigma = 2 \times 10^3$. In an attempt to search a larger part of the control space, the seeding strategy consisted of using cosine and sine functions with randomized amplitudes and frequencies described in the following equation

$$u_{landscape} = A\cos(k_1 \pi t/T) + B\sin(k_2 \pi t/T) \quad (13)$$

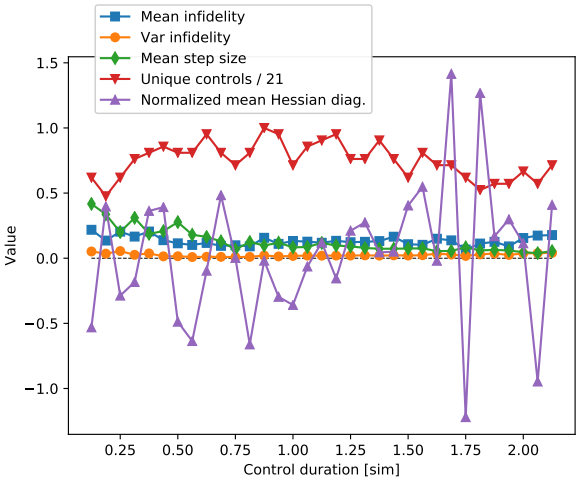


FIG. 24: Calculated quantities of interest for describing the control landscape. The gradual decrease in mean step size as control duration T is increased could indicate that the number of local traps increase with duration. The mean Hessian diagonal was expected to be negative for all times since this would indicate a fidelity "hill". Instead it fluctuates between positive and negative values and even for nearby durations there is a big difference.

with $A, B \in \{-0.5, 0.5\}$ and $k_1, k_2 \in \{-3, 3\}$ being randomly generated for each control.

The resulting controls with $\mathcal{F}(u) \geq 0.1$ were kept for further analysis and the rest discarded. The quantities of interest mentioned above were calculated based on the remaining controls. These have been plotted on Fig. 24. Most of the quantities calculated are not easily interpreted. The mean step size gradually decreases as control duration increases. Could indicate that the control landscape becomes more "rugged" as time increases as described previously. Despite the attempt to thoroughly explore the control landscape using the seeding strategy in Eq. (13) it could be interesting to see how much the results would change from the inclusion of a new set of data.

VII. COMPARISON

As has been demonstrated, QEngine offers several advantages to Quantum Composer with respect to flexibility and computational power. However, this does not mean that the QEngine should be used for every problem. Quantum Composer makes it very easy to set up a desired system, run a simulation on that system and get the system plotted as the system evolves. This makes it preferable to use Composer when one wants to visually and (more or less) playfully explore the behavior of a system as the different parameters are tuned up or down.

Here, I will attempt to list some of the pros and cons

of both tools.

QEngine

+ Flexible and customizable
+ 3 Optimization algorithms
+ Superior data gathering capabilities
− Data must be plotted seperately
− Data gathering can be *very* time consuming
− Steep learning curve
− Difficult installation process

Quantum Composer

+ Many visualization options
+ Intuitive to explore a given system
+ Graph node to drag out a control function
+ Simulating and optimizing is usually fast
− Limited flexibility and customization
− Most types of data must be manually gathered
− Some types of data cannot be exported from the program at all

Based on the considerations above it is concluded that Quantum Composer is preferable to use as an exploration tool. It is fast to set up, relatively easy to use and most importantly, it is very good at visualizing the simulation through its different kinds of plotting nodes. An example of this exploration could be a physicist trying to simulate an experiment to get a feeling for what kind of settings the experimental apparatus should be capable of. With the current state of Quantum Composer, I find it advantageous to use the QEngine for any kind of simulation where more than around 15 datapoints are desired. While the QEngine has a vastly steeper learning curve, it is designed to easily generate large amounts of data using computationally efficient calculations. It should also be mentioned that, in my personal opinion, having worked or used Composer before working with the QEngine might have smoothed the learning curve of using the QEngine. This is because many of the nodes in Composer correspond to a class in the QEngine library (remember that Quantum Composer is built on top of it) and so one can use the intuition of how the nodes are connected in Composer to find out how to set up a system in the QEngine. The example files provided with the library are also of great help to get started. These conclusions about when to use Quantum Composer align well with what the intended strengths of Composer are [1].

It could be advantageous to combine the playful, visual and fast elements that Composer has, with the more systematic, flexible and superior data gathering of the QEngine. One way to possible exploit the best of both these tools could be to use Quantum Composer for globally exploring a given system through the intuitive and playful graph control node. Any promising candidates for good seeds could be exported from Composer in the `.csv` format to be read by a QEngine script, where noise and other more local optimization approaches could be utilized.

These considerations also give rise to a number of suggestions for desirable features for Quantum Composer in order to make it more attractive to use as an expert tool. Most of these points are a result of a brainstorming/discussion session with Shaeema Ahmed (Aarhus University) and Tiantian Zhang (TU Vienna).

Desirable features for Quantum Composer

- Conversion of real world data to dimensionless input scalars
- Exportable control functions to potentially be used in experiments
- Save data even when for-loops have run
- Import control functions from an experiment
- Enhance data gathering capabilities all around
- Select a time range for control durations
- Feed analytical control functions to GRAPE optimizer
- Add noise to the control function
- A more detailed description of how to use the GPE spectrum node
- Undo and redo buttons
- An ability to automate data gathering, including exporting data

VIII. DISCUSSION

As described in section IV A, some of the ways to explore this problem are blocked by numerical instabilities. This could possibly be fixed by increasing the number of timesteps, which is currently 501, since the smaller the timestep used, the better an approximation each step forwards in time becomes. This might mean that the GPE spectrum would have to be recalculated to minimize the variance of important states, but also mean an increased computational cost of each simulation.

The energy spectrum is generated in Quantum Composer to create wave functions, find energy levels and calculate variances wrt. $\hat{H}_{GPE}$. This generation is subject

to some initial guesses that can be tweaked to minimize the variance of any relevant states, thus ensuring that the simulation will be as accurate as possible. However, it seems to be the case that while one can tweak these guesses to reach low variances (on the order 10^{-8} or better) for one set of system parameters (e.g. x_{max}, x_{min}, β), it has not been possible to find a set of initial parameters that keep the variance low as the system is altered. This is a relevant source of error as a substantial part of the results in this report describe how β changes the system behavior.

Thus the accuracy of the results could be improved further, in some cases even greatly, by instead of keeping the initial guesses the same as the system was altered, to instead have the guesses re-tweaked to fit the new system. By recalculating the spectrum for each system, it is suspected that different results could be obtained for some figures, most notably Fig. 10, 12 and 13. This is due to the control function $u_{\mathrm{non-opt}}(t)$ depending on the energy spacing between the first 2 states, ω_{01}, and because the 2 other figures are presumed to be particularly sensitive to sources of error in the calculation of initial and target state.

In addition to the source of error regarding the estimate of the QSL shown in Fig. 12, it is believed that these estimates are higher than what is actually the speed limit for this problem. This is mainly based on the number of GRAPE iterations being set to the relatively low value of 100 iterations, where it would be beneficial to instead have it several times higher, e.g. 750 or 1000 iterations. The number of iterations was set this low due to the lack of automated data gathering in Composer. This is particularly problematic when optimizing systems with a duration around the QSL of the problem, since the optimization landscape is relatively flat and so the step size α_k has to be lower in order for GRAPE to properly traverse it. This means that while it will take a large number of iterations to reach a height of 0.99, it will most likely be possible for values lower than those currently found.

Likewise it should also be kept in mind that only a very limited part of the "control space" has been explored so far, and so it would most likely be a matter of doing a more persistant search to find a lower QSL for one or more of the βs examined. This is one of the places where it would be very interesting to be able to add random noise to a control seed as one can do in the QEngine, as this would most likely increase the exploration of the local optimization landscape that the initial seed has "globally" pointed out. This could hopefully lead to a similar situation as the one seen in Fig. 22, where the optimized "noisy" seeds were able to outperform the controls optimized in Composer.

In Sec. IV F where it was attempted to seed an optimal solution based on the other clustered, optimized solutions, it was found that these seeded cosine functions of increasing frequency did not perform as well as the optimized solutions they attempted to replicate. One should firstly consider that the clusters were not based on the control function $u(t)$ projection, but instead the control minus expected position $u(t)-\langle x(t)\rangle$. Thus, while it would have been an easy way of determining optimal seeds, it should not be unexpected that the seeds would need to be more complicated. To further underline why it is not this easy to find the optimal seed for this problem, the control landscape of the shakeup problem has been found to be particularly complicated compared to other control problems [10, 13].

Since the amplitudes of the control function oscillations were very restricted for this experiment, it could be interesting to see if what was found in Sec. IV A 2 could be used to have a larger span of amplitudes available for seeding.

The primitive clustering method of counting 0-crossings on optimized controls was found to be both a simply and objective way of classifying controls. Another possibility would have been to count the number of times the direction of the control was reversed (or "flipped"), but the number of flips were found to be ambiguous when the analysis was done by hand, looking at screenshots.

The second attempt at clustering optimized solutions using the QEngine showed that the controls were noisy and contained many small oscillations. These solutions, while being correct and optimal for the problem, fail to highlight the interesting physics that can be extracted from them compared to more physically realistic solutions with fewer rapid oscillations. In order to generate solutions with more physical insight, the regularization term γ and the added noise of solutions should be tweaked and the latter possible removed.

Regarding the clustering strategy itself, it can be considered problematic that the solutions that are clustered stem from cosine and sine functions, not from player seeds as in Ref. [10]. This could pose a problem when trying to characterize the solutions by their projection onto the same functions they were made from. Thus, it could be beneficial to these results to change either the seed generation or the features that are clustered.

It is also problematic that the clustering method fails to include the best performing controls below 1 ms. While these points of interest are sparsely populated in the $(t, 1-\mathcal{F})$ representation, it is unknown where they lie and with what density they populate the feature space. Nonetheless it can be concluded that their density is smaller than for the other controls considered and that they are far enough away from the found clusters to be included in them. The problem of not classifying these optimal controls can possibly be solved by using another clustering method than DBSCAN, such as OPTICS, which supposedly performs well with data containing uneven cluster densities unlike DBSCAN.

The exploration of the control landscape using the QEngine did not yield much information about the landscape. The consequently low variance of the infidelity could indicate that the seeding strategy mentioned in Eq. (13) does not exhaust the control space as it was intended to. Furthermore, the calculation of the mean Hessian diagonal might need to be re-done with a higher order of accuracy in the finite difference estimate.

IX. CONCLUSION

The backbone of the results in this report have been generated in Quantum Composer. The workflow of using this amateur tool for data gathering and visualization has been described and visualized. The numerical boundaries of Composer have been demonstrated and their implications for exploring the challenge have been discussed. Likewise the performance and numerical accuracy of different grid sizes have been estimated their results on the performance/accuracy trade-off discussed. Notably, the numerical limits from exceeding grid boundaries have been explored and strategies to improve results or work around problems have been proposed.

It has been demonstrated that the behavior of BECs are sensitive to the parameters β and T. As it has been shown in Fig. 9, this changed behavior compared to a single particle can be beneficial and capable of reaching higher fidelity than is otherwise possible. On the other hand it is also apparent that the non linear contribution to the Hamiltonian can make the optimization more difficult or time consuming as well as increasing variance of the Hamiltonian eigenstates. It has also been shown that the β term in Eq. (1) increases the QSL up to a estimated value of 2 times that of a single particle for the case $\beta = 25$.

It was also attempted to cluster the performance of $\cos(k\pi t/T)$ control seeds to the frequency k, although the data shown in Fig. 14 shows that the seed frequency and optimized performance are not connected in a straightforward manner. What seems more promising is analyzing the number 0-crossings in the optimized solutions and clustering these. This analysis indicates that optimal solutions have an increasing number of 0-crossings as control duration is increased. The robustness of some optimized solutions have been explored both regarding a scaled potential and a scaled atomic interaction, as is shown in Fig. 17. Optimized solutions become increasingly more sensitive to system changes the better they initally perform compared to other controls.

Some strategies involving choosing a control function have also been explored. The differences, advantages and disadvantages of using either an analytical function node or a graph node to determine the control function have been discussed. This includes how varying T affects the control function, as has been illustrated on figures 19 and 20. Finally, 2 concrete strategies for determining the QSL in this problem have been explained, both using the graph node to create a control function.

The QEngine has been used to generate data that illustrate how it differs from using Quantum Composer. It was shown that one can use the QEngine to further improve on results optained in Composer, where seeds with random noise were optimized using all 3 optimizers available in the QEngine. The QEngine is able to match or outperform the best results from the different seeds used in Composer, by using random noise and optimization by different algorithms and a higher iteration cap.

It was also attempted to use the QEngine to characterize the control landscape by calculating quantities of interest such as mean step size, infidelity variance and mean Hessian diagonal. These results are, however, inconclusive.

Based on the work done using both tools, their pros and cons have been discussed and suggestions for where it is advantageous to use one tool over the other have been provided, as well as when it might be useful to use both tools for the same problem.

Furthermore, a list of desirable additions to Quantum Composer has been added. The implementation of these features would hopefully make the tool a more effective tool for expert-use, where customization, flexibility and automated data gathering features are highly valued.

Finally, there are aspects of this work that would interesting to continue to explore and research. These include:

- Test if the harmonic potential is capable of reaching higher fidelities in the time domain around $T \sim 1$, as this was demonstrated to be near the QSL for the quartic potential.
- Find out if there is an upper limit to the self interaction strength similarly to what was found in Sec. IV E.
- Further explorations of the possible seeding strategies in the QEngine in order to develop a more exhaustive search of the control space.
- An analysis of successful ($\mathcal{F} \geq 0.99$) controls in an attempt to distinguish patterns, phases of common / different behaviors, etc.

[1] S. Z. Ahmed, J. H. M. Jensen, C. A. Weidner, J. J. Sørensen, M. Mudrich, and J. F. Sherson, "Quantum composer: A programmable quantum visualization and

simulation tool for education and research," (2020), arXiv:2006.07263 [physics.ed-ph].
[2] J. Sørensen, J. Jensen, T. Heinzel, and J. Sherson, Computer Physics Communications **243**, 135–150 (2019).
[3] S. van Frank, M. Bonneau, J. Schmiedmayer, S. Hild, C. Gross, M. Cheneau, I. Bloch, T. Pichler, A. Negretti, T. Calarco, and et al., Scientific Reports **6** (2016), 10.1038/srep34187.
[4] C. Chin, R. Grimm, P. Julienne, and E. Tiesinga, Reviews of Modern Physics **82**, 1225–1286 (2010).
[5] J. J. Sørensen, *Quantum Control of Ultracold Quantum Systems*, Ph.D. thesis, Aarhus University (2018).
[6] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Journal of Magnetic Resonance **172**, 296 (2005).
[7] J. J. W. H. Sørensen, M. O. Aranburu, T. Heinzel, and J. F. Sherson, Physical Review A **98** (2018), 10.1103/physreva.98.022119.
[8] S. Deffner and S. Campbell, Journal of Physics A: Mathematical and Theoretical **50**, 453001 (2017).
[9] E. A. Donley, N. R. Claussen, S. L. Cornish, J. L. Roberts, E. A. Cornell, and C. E. Wieman, Nature **412**, 295 (2001).
[10] J. H. M. Jensen, M. Gajdacz, S. Z. Ahmed, J. H. Czarkowski, C. Weidner, J. Rafner, J. J. Sørensen, K. Mølmer, and J. F. Sherson, "Crowdsourcing human common sense for quantum control," (2020), arXiv:2004.03296 [quant-ph].
[11] M. Dalgaard, "Magnus expansion, quantum landscapes and uncertainties," Group meeting presentation (2021), accessed: 2021-1-11.
[12] B. Fornberg, Mathematics of computation **51**, 699 (1988).
[13] G. Jäger, D. M. Reich, M. H. Goerz, C. P. Koch, and U. Hohenester, Phys. Rev. A **90**, 033628 (2014).

Appendix A: System parameters

Generally when doing numerical calculations it can be useful to rescale small or large quantities like $\hbar$ or M_{Sun} to a new size in order to avoid problems caused by finite precision memory and rounding errors. In this report the relevant quanteties for rescaling are

$$x_{SI} = \mu_{length} x, \quad t_{SI} = \mu_{time} t, \quad V_{SI} = \mu_{energy} V \tag{A1}$$

Here μ_{energy} is chosen as

$$\mu_{energy} = \frac{\hbar^2}{2\kappa m \chi^2} \tag{A2}$$

where κ is the *kinetic factor*.

One can freely define 2 of the 3 quantities $\{\mu_{length}, \mu_{time}, \kappa\}$ and the remaining quantity will follow. Throughout the report the mass is taken as $\mu_{mass} = m_{Rb87}$ and we require $\mu_{energy} = \hbar/\tau$ in order to get dimensionless equations of motion, where $\hbar = m = 1$. Unless otherwise stated, the grid size used was $n_x = 256$.

System A1: Units are chosen as

$$\mu_{length} = 1\,\mu\text{m}, \quad \kappa = 0.5 \tag{A3a}$$
$$\Rightarrow \mu_{time} = 2\kappa\mu_{mass}\mu_{length}^2/\hbar = 1.3699\,\text{ms} \tag{A3b}$$

$x \in \pm 6\,\mu$m with potential

$$V(u) = a(x-u)^2 \tag{A4}$$

where $a = 50$. The value of β was varied in this system.

System B1: Units are chosen in the same way as in Eqs. (A3a)-(A3b). $x \in \pm 6\,\mu$m. The potential from Eq. (A4) is modified with a quartic term

$$V(u) = a(x-u)^2 + b(x-u)^4 \tag{A5}$$

with $b = 128$. β was also varied in this system.
Shake up level in Ref. [10]:

$$\mu_{length} = 1\,\mu\text{m}, \quad \mu_{time} = 1\,\text{ms} \tag{A6a}$$
$$\Rightarrow \kappa = \hbar\mu_{time}/(2\mu_{mass}\mu_{length}^2) = 0.36537 \tag{A6b}$$

Position restricted to $x \in \pm 2\,\mu$m. The potential used was

$$V(u) = \sum_{r=2,4,6} = p_r(x-u)^r \tag{A7}$$

with coefficients and interaction strength

$$p_2 = 65.8392, \qquad p_4 = 97.6349, \tag{A8a}$$
$$p_6 = -15.3850, \qquad \beta = 1.8299. \tag{A8b}$$

Composer implementation of QM2 Shake up: Since Quantum Composer has a locked choice of $\kappa = 0.5$, it is necessary to rescale the units from the Quantum Moves 2 level *Shake Up* in order to simulate the same system in Composer. The scaling factor is $0.5/0.36537 \approx 1.368$ and yields the following transformation of Eqs. (A6)-(A8) Position still restricted to $x \in \pm 2\,\mu$m.

$$\kappa = 0.5, \qquad \mu_{time} = 1.368\,\text{ms}, \tag{A9a}$$
$$\beta = 2.5042, \qquad p_2 = 90.0994, \tag{A9b}$$
$$p_4 = 133.611, \qquad p_6 = -38.5267 \tag{A9c}$$

Appendix B: QEngine settings

This section will describe the seeding strategies and optimization parameters used when working with the QEngine in this report.

1. Clustering

This refers to section VI A of the report. Control duration was initially 0.125[sim] and incrementally increased. The simulation parameters are listed in Table I.

Seed	iter.	T_max[sim]	T_steps	Noise	Optimizers	N_max	γ	σ	Bounds[sim]
$\sin(2\pi t/T)$	5	0.8125	23	± 0.5	All	3000	1×10^{-5}	2×10^3	± 2
$0.5 \sin(4\pi t/T)$	10	0.8125	23	± 0.4	GRAPE	3000	1×10^{-5}	2×10^3	± 2
$\sin(\pi t/T)$	10	0.8125	23	± 0.7	GRAPE	3000	0	0	
$0.33 \cos(2\pi t/T) + 0.20 \cos(3\pi t/T)$	10	1.125	33	± 0.4	All	3000	1×10^{-6}	0	

TABLE I: Settings used to obtain results described in Sec. VI A.

2. Landscape

Landscape exploration is described in section VI B in the report. Currently, only a single run of data has been done. The seeding strategy for this run involved a linear combination of sine and cosine with random front factors and frequencies as shown in Eq. 13, where 25 seeds where generated for each control duration. This was varied between 0.125 and 2.125[sim.] in 33 steps. Noise was added to each point on the control, the size was between ± 0.1. Seeds were optimized using GRAPE with $N_{max} = 3000$, $\gamma = 10^{-5}$ and $\sigma = 2 \times 10^3$ with bounds $x \in \pm 2\,\mu\text{m}$.