Wavepacket Propagation Program (w/ Dr Nadav Avidor)

Lorenzo Basso, Jack Lee, Matthew Zhang, Feiyang Chen

April 2020

Contents

1 WPP Introduction

1.1 About

The Wavepacket propagation project is an extension of a Part III project by Ocean Haghighi-Daly. His paper is attached below. It is a working version of a 3D Split-Operator method of wavepacket propagation. The primary point of improvement to the program is that the propagation algorithm has been completely redone in CUDA C, which has greatly improvement the speed of simulation. Other improvements include various bugfixes of various issues (all of which can be found on the Github Page (Currently inaccesible to public))

1.2 Prerequisites

To begin with, a computer with a CUDA compatible GPU (List of compatible GPUs) is required.The MEX/CUDA dependent files have already been compiled for an x64 Linux and x64 Windows system with CUDA 10.1 compatibility. However, it is still potentially necessary to install a CUDA/C++ compiler depending on the CUDA capability your GPU has as matlab/CUDA can be quite finicky with versions.

Installation Guides:

- Latest CUDA Package Download (If you believe you don't need to follow the installation guides as the Windows install is reasonably simple)
- Windows Installation Guide
- Linux Installation Guide
- Mac OS X Installation Guide

These guides also include the relevant instructions to install CUDA/C++ compilers for your relevant OS. If for some reason the documentation is too tedious, we will (maybe) try to provide a simplified version that produces functionality in the Appendix

Setup MEXCUDA Compiler

Windows:

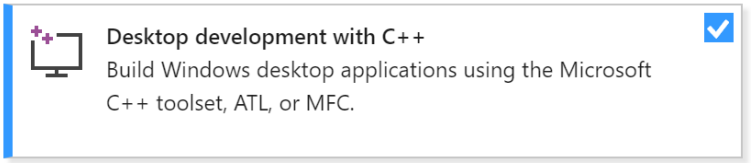
For Windows, Visual Studio must be installed with the Desktop Development Package for C++ installed. If the user has MinGW compiler installer for previous MEX compile operators, then it may be necessary to change compilers. This is because MinGW is incompatible with MEXCUDA operations. This can be done by running

```
mex -setup C++
```

Which will output something like the following:

```
MEX configured to use 'Microsoft Visual C++ 2019' for C++ language compilation.

'To choose a different C++ compiler, select one from the following':
MinGW64 mex -setup:'C:\PATHTO\mexopts\mingw64_g++.xml' C++
MSVS_2019 mex -setup:'C:\PATHTO\mexopts\msvcpp2019.xml' C++
```



Linux:

For a Linux system, the CUDA installation should come with the corresponding NVCC compiler for MEXCUDA. GCC (The standard C++ compiler) is a prerequisite to installing CUDA, and is also necessary for code compilation.

The required Matlab Toolboxes for the program to function are as follows:

- Parallel Computing Toolbox
- Bioinformatics Toolbox

2 Usage

2.1 WavePacketPropagation_Beta4_2

All variables are in SI units. *SetupSIUnits.m* initializes global variables used in picoscale wavefunction propagation:

$$\begin{aligned}
\textbf{hBar} &= 1.054571800 \times 10^{-34} Js \\
\textbf{c} &= 299792458 m/s \\
\textbf{eV} &= 1.6021766208 \times 10^{-19} J \\
\textbf{amu} &= 1.660539040 \times 10^{-27} kg \\
\textbf{A} &= 1.00 \times 10^{-10} m \\
\textbf{ps} &= 1.00 \times 10^{-12} s
\end{aligned}$$

lx, **ly**, **lz** define the length of the box in which you want the wavefunction to propagate, while **nx, ny, nz** denote the grid sizing (how finely you want to splice up essentially). Grid sizing also determines how much memory is used (approximately $nx \times ny \times nz$ bits).

RealTimePlotting displays each figure at the time intervals determined by **numGfxToSave**. Having this setting on also saves the figures when **SavingSimulationRunning = true**. It is greatly advised to either change **numGfxToSave** to 1 or set **RealTimePlotting** to false as it will

greatly improve the simulation speed. This is because to render the images, the program must copy the CUDA C arrays into matlab, which we have noted to be extremely time consuming.

SavingSimulationRunning will create a folder under *SavedSimulation* (unless otherwise specified by the user by changing **saveLocation**) and save the matlab arrays of the wavefunction while the program is running. With **RealTimePlotting** set as true, the figures shown will also be saved. The number of pictures saved is dependent on **numGfxToSave**. **numGfxToSave** breaks up the total iteration time into equally spaced blocks of time. It will also save the initialization data, and additionally, the final wavefunction as a matlab data file. As a result, this will also greatly increase simulation time.

SavingSimulationEnd will make it so that the program only saves the final wavefunction, as well as the initialization data. This can be used if the propagation of the wavefunction is not as important. This does not significantly increase simulation time relative to **SavingSimulationRunning**

DisplayAdsorbateAnimation shows the motion of the adsorbates in an animation. This can be useful useful if you're feeding a custom potential+path to ensure that it is phrased correctly. This setting generally does not signiicantly impact the overall performance of the program.

savingBrownianPaths will save the randomly generated path to a txt file **Browniefile.txt** which can then be fed back to be used as a custom potential later.

propagationMethod dictates the propagation method used for the program. The methods are implemented in matlab unless otherwise specified:

1. Runge-Kutta method of Order 4
2. Split Operator of Order 2
3. Split Operator of Order 3 with K split
4. Split Operator of Order 3 with V split
5. Split Operator of Order 3 with V split, Time dependent
6. MEXCUDA while loop with Split Operator $O(dt^3)$ with Vsplit, Time dependent (This is the fastest implementation of CUDA)
7. Split Operator of Order 3 with V split, Time dependent in Matlab and CUDA (Primarily to compare the two methods, will output the average difference in the Psi tensor for each)
8. Matlab while loop with CUDA Split Operator Order 3, V Split, Time dependent (generally slower than method 6)

numAdsorbates determines the number of adsorbates on the scatter surface. There were previously issues which occured if the number of adsorbates in the custompaths file was different

from that of the adsorbates declared in the program scope, but a fix has been pushed which holds for certainty with 3D wavepacket scattering. It is still unsure whether or not it holds for a 2D wave.

custompaths is the file you feed with a custom potential and path for the adsorbates. The instructions are further detailed below.

decayType is the potential that extends from the corrugation function in one dimension. 1 = exponential repulsive potential. 2 = Morse attractive. 3 = Morse-like, so it can be adjusted between morse and exponential by changing alpha. alpha = 0 is the same as exponential, alpha = 2 is morse. 4 = custom potential: this uses the custom potential in the text file specified by **potfile**, as explained later.

potfile: Text file containing floats for real and imaginary part of potential, seperated by **lz**. The potential should be high to prevent tunelling over cyclic boundary

zOffset translates the entire corrugation function away from the surface by the distance specified. It can be used to prevent quantum tunnelling due to cyclic boundary conditions for **decayType** 1 - 3, or it can be made negative to allow you to specify the potential between the corrugation function and surface for **decayType** = 4.

2.2 Setup Folder

It is likely that the only file in here that needs to be changed at a user level will be *SetupInitialWavefunction.m*. If an alternative wavepacket is necessitated, it is preferable to add an alternative generating function rather than change the existing ones (*InitializeGaussianWavefunction3D.m*). The relevant parameters and functionality is detailed in considerable depth within each file itself.

2.3 Scripts

In the Scripts folders are various Matlab scripts used by Ocean Haghighi-Daly to plot and generate various test cases to demonstrate the program functions as the theory predicts.

Relevant explanations for the various scripts are documented below:

Potential plotting.m is a script used to initialize and generate various potentials forms. It can serve as a reference to the user to verify that relevant potentials are generated correctly within the program body itself

Plot Form Factors for different simulations - Interpolated.m

Not precisely sure

Plot Form Factors for different simulations.m

Optimizing setup 3D.m

I have no idea what this does exactly

Optimizing setup.m

I have no idea what this does exactly

General plotting font and size settings.m

Not very useful for any user. Read ovr the short code as you desire.

Gaussian Random Error plotting.m

Contour and form factor ring plotting.m

autoRun.m

A code body that once allowed various different parameters to be fed into the body. Rendered effectively useless by all the recent improvements.

2.4 Saving

Most of these functions have been roughly explained up about in the WavePacketPropagation. The user can scan through the specific files and see exactly what the functioning code is and adapt as suitable for their own needs.

CreateNewSimulationFolder.m: Functionality has been added to allow for saving to a specific folder/directory rather than the auto one which will simply create a "Saving" folder in the main program body. Aside from that, this file also has the functionality to autogenerate and increment sim—- folders that contain the results of a particular run. Multiple runs can be completed with one job if computation is done with a server and not on-site and the saved results will be sorted out accordingly across sim—- folders.

(Possibly) SavedSimulation: If a custom SavedSimulation folder isn't specified by the user, then the program will by default, save the specified parameters to subfolders titled with sim—-, where —- are incrementing numbers.

2.5 RandomMotion

Nothing of particular use to the user

2.6 PropagationAlgorithms

This folder contains critical files to the functionality of the program. It is these algorithms that pass the wavefunction through timesteps to simulate the wavepacket propagation.

The program predominantly exists to run on the Split-Operator method of wavepacket propagation. The file **RK4step.m** exists as it was used by Ocean Haghighi-Daly to demonstrate that the 3rd order Split Operator method functions within margin of error of a Runge-Kutta 4th order propagation algorithm.

The time-dependent third order V-split operator is the primary one used for wavepacket propagation in this program body. The other matlab split-operator methods were used by Ocean to demonstrate functionality. We have left them as is as they are still available to be used as the primary propagation method for simulation.

The primary method we recommend using is the CUDA C implementation of the third order V-split operator. It is significantly faster than any matlab implementation (up to 7x faster).

mexcudawhile.cu is the a casting of the propagation of the iteration loop done in CUDA C. This saves time in keeping the arrays in purely CUDA C, rather than having to copy it between Matlab arrays and CUDA arrays for every step.

2.7 Print

Section contains files to print various properties about the wavefunction. They are all self-explanatory

2.8 PotentialFiles

This is a folder we used to store custom potential files. There is not much else to say about this.

2.9 Operators

Matlab implementations of the Momentum and Energy Expectation operators.

2.10 MEX_Helpers

This folder contains various helper code/headers for compiling the various CUDA C files used to replace their Matlab counterparts.

print_CUDA_array.cu and **print_complex_CUDA_array.cu** were used by Lorezno Basso to verify that the CUDA code functions near identically to the matlab code it was replacing.

cuda_helper.h and **cuda_helper.cu** function as error checks to help debug the program when first writing the CUDA C implementations.

copy_from_CUDA_complex.cu, **copy_CUDA_complex_array.cu**, and **copy_CUDA_array.cu** all function to copy arrays between CUDA C and Matlab. The second file has effectively superceded the third file, as it includes an implementation of complex numbers.

cmp_complex_matlab_CUDA.cu was also used to cross compare the matlab/CUDA arrays to verify its functionality.

2.11 Graphics

Files for displaying graphics related to the wavefunction propagation. Each file is roughly self-explanatory.

3 Custom Paths and Potentials

Functionality to include custom adsorbate paths and potentials was added by Jack Lee.

1. **Custom potential:** this allows you to specify a custom potential profile in the Z direction. To apply this, set **decayType** to 4 and specify a .txt file (in beta4_2) as a string in potfile. This should contain several numbers separated by spaces (or new lines) which are the values for the potential in Joules in each cell of the simulation out from the corrugation function (+**zOffset**), i.e. the values are separated by a distance of **lz/nz**. The potential is zero for all z not specified by the file. **zOffset** should have a negative value, and can be used to extend the potential back from the gaussians to the surface, as if **zOffset** is 0 the area between the surface and the gaussian peaks will have 0 potential. The potential specified should be very high near the surface to prevent the wavepacket from getting past it, because the propagation algorithm leads to cyclic boundary conditions so any psi that reaches the end will appear at the other side, which is non-physical.

2. **Custom paths:** this allows you to specify the paths that adsorbates take, rather than having them be generated randomly. To enable this, set **custompaths** to true and specify in **pathfile** a text file (in beta4_2) formatted as follows: the first entry on each line should be a time, then for each adsorbate there should be its x position and then its y position at that time, separated by spaces. There can be any number of times, as long as the first is $\leq$ **tStart** and the last is $\geq$ **tFinish**, and they'll be interpolated to get the paths.

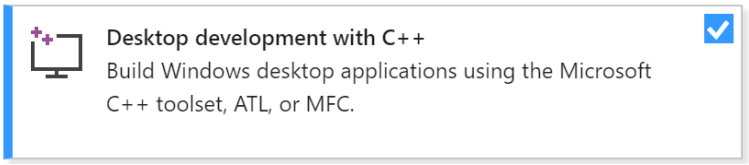
3. **Saving paths:** this allows you to save the randomly generated paths adsorbates take this time in the simulation. It doesn't do anything if custom paths is on. To enable it, set **savingBrownianPaths** to true and name a .txt file in **Browniefile**, which will be overwritten or created in beta4_2. This saves the paths in the same format as custom paths reads them, with one line for each timestep of the simulation. These files can later be read by custom paths to reproduce this simulation (and could be used to vary the potential, detail etc.), although there is a small error introduced here so that the final psi from the custom paths simulation is slightly different to that of the original. Two custom paths simulations from the same source will be the same, however.

4 Appendix

4.1 CUDA + Compiler Install

4.1.1 Windows

Installation for Windows is reasonably straightfoward. It requires first to download the Latest CUDA Package Download and run the relevant executable file.



Next, Visual Studio must be installed with the Desktop Development Package for C++ installed. If the user has MinGW compiler installer for previous MEX compile operators, then it may be necessary to change compilers. This is because MinGW is incompatible with MEXCUDA operations. This can be done by running

```
mex -setup C++
```

Which will output something like the following:

```
MEX configured to use 'Microsoft Visual C++ 2019' for C++ language compilation.

'To choose a different C++ compiler, select one from the following':
MinGW64 mex -setup:'C:\PATHTO\mexopts\mingw64_g++.xml' C++
MSVS_2019 mex -setup:'C:\PATHTO\mexopts\msvcpp2019.xml' C++
```

4.1.2 Linux

CUDA installation for Linux is significantly more annoying than Windows. As the installs for different distros can vary fairly significantly, the best suggestion I have is for the user to follow the instructions in the installation guides.

The corresponding CUDA install should come with the NVCC compiler necessary to compile the MEXCUDA files.

4.2 Ocean Report

The report is displayed on the next pages

Quantum simulations of atom-surface scattering

Candidate Number: 8246Q
Part III Project Plan, Department of Physics, University of Cambridge
Supervisor: Dr Andrew Jardine
Surface, Microstructure and Fracture Group

***Abstract*—Helium-atom scattering is a surface characterisation technique that is particularly sensitive to surface-level effects. Scattered He energies encode the dynamics of surface-adsorbates, the results of which are usually interpreted using the "Kinematic Scattering Approximation", which states that the scattered He intensity can be written as the product of a 'Form Factor' (dictated by adsorbate shape) and 'Structure Factor' (describing adsorbate positions). Previous work done in 2D for purely repulsive surface potentials used quantum-mechanically exact dynamic simulations to verify the Kinematic Approximation for most situations, except when the Form Factor drops to zero. In this project, a selection of numerical algorithms to solve the time dependent Schrödinger equation are compared, of which the 'Split-operator' method is chosen - for its ability to utilise Graphical Processing Units (GPUs) - and implemented. The implementation was verified by propagating model systems and comparing the results to analytic and Runge-Kutta 4th order propagated solutions that preserved norm to within 10^{-6}. Spatial quantisation limits of a model physical system (repulsive Gaussian adsorbate) were tested, enabling physically significant 3D- , and higher fidelity 2D-simulations to be run. In order to determine how different adsorbate shapes, attractive surface potentials, and physical 3D systems (as opposed to the 2D systems analysed previously) affect the Kinematic Approximation, Form Factor shape was analysed for both attractive and repulsive surface potentials while varying adsorbate height and width, potential well depth, and 3D adsorbates size. The effect of using an attractive surface potential, and increasing potential depth, was to increase the depth of the Form Factor minima towards zero, suggesting the Kinematic Approximation will be less successful in these regions. Form Factors were found to be very sensitive to adsorbate height (especially when attractive potentials were used), but less so for adsorbate width. 3D Form Factors were found to be similar in shape to 2D equivalents, but with minima depth reduced for repulsive surface potentials. While minima depth for 3D attractive potentials was also reduced, the effect was small.**

I. Introduction

Helium-atom scattering is a surface characterisation technique that is particularly sensitive to surface-level effects. The low energy, high mass, and neutrality of thermal Helium provides a non-destructive, exclusively surface-sensitive probe [1], unlike electron, neutron, and X-ray scattering. The Cambridge Surface group use a particularly sensitive ^{3}He setup called Spin-Echo [1]. The dynamics of surface-adsorbates are encoded in the energies of scattered ^{3}He atoms. Experimental results are usually interpreted through the "Kinematic Scattering Approximation" which approximates the scattered beam intensity I as the product of a Form Factor F (dictating adsorbate shape) and Structure Factor S (describing adsorbate positions) [1], [2]:

$$I(\Delta\mathbf{k}, \Delta\omega) = S(\Delta\mathbf{k}, \Delta\omega) \cdot F(\Delta\mathbf{k}, \Delta\omega) \tag{1}$$

all terms being functions of ^{3}He momentum change, $\propto \Delta\mathbf{k}$, and energy change, $\Delta\omega$.

Physically, the Kinematic Approximation corresponds to incident plane-wave beams scattering from identical species, analysed in the far-field regime (similar to Fraunhofer diffraction [3]). Like Fraunhofer diffraction patterns, the Form Factors are the amplitude of the plane-wave components making up the scattered wave, but from a single scatterer, in $q \equiv |\mathbf{k}| \cdot \sin(\theta)$ space (which is $\mathbf{k}_x$ space for all setups in this report). While Fraunhofer diffraction patterns are the Fourier-transforms of 2D apertures, the 'aperture' in the Kinematic Approximation is an adsorbate with 3D extent. Form Factors can therefore be thought of as infinite superpositions of Fraunhofer diffraction patterns, caused by apertures corresponding to cross-sections through the adsorbate potential, at different heights in the adsorbate. Further complications arise due to the adsorbate-potential being a graduated ('soft-wall') potential, not the hard-wall boundary that a Fraunhofer aperture would be. The effect of adsorbate size and shape on Form Factors is therefore not trivial.

Verifying the Kinematic Approximation requires running quantum-mechanically exact simulations for dynamic surfaces, and comparing the results to Kinematic Approximation predictions. [4] ran quantum-mechanically exact 2D dynamic simulations for purely repulsive potentials and verified the accuracy of the Kinematic Approximation, except around zero-value points in Form Factors.

The main aim of this project is to better understand the limitations of the Kinematic Approximation, notable due to its prevalence in analysing experimental scattering results. Changes in Form Factor minima position and depth for different physical adsorbate sizes and shapes are of particular interest, since minima can transform into the zero-point values where the Kinematic Approximation breaks down. Previous work has been done that demonstrated that 3D adsorbate Form Factors do differ from 2D 'equivalents' [16], and the same result is found here.

In this project, a computational toolbox capable of running 3D dynamic quantum-mechanics simulations (by numerically propagating a wavefunction according to Schrödinger's equation) on a "Graphical Processing Unit" (GPU) was created, tested for accuracy, and used to investigate static adsorbate Form Factors.

Section II describes the computational background to the project, discussing Hardware and Algorithm choices. III and IV discusses physical models and software implementation, V analyses potential sources or error, VI verifies the implemented code and sets limits of validity for simulations that can be run. VII investigates the physical Form Factors, and IX concludes.

II. Computational background

A. Hardware

Everyday computers use "Central Processing Units" (CPUs) usually consisting of 4–16 'cores' - each handling 1 process at a time - to do computations. CPUs are incredibly fast, achieving $\sim 10^9$ operations per second. GPUs, on the other hand, consist of ~ 1000 cores, capable of $\sim 10^6$ operations per second, and are used for graphical displays where millions of pixels are rendered continuously at $\sim$ 60fps to keep movies/games smooth. GPU programming is becoming more prevalent, specifically for tasks that can manipulate (often highly optimised) graphical processing algorithms to achieve calculations, or tasks that can be parallelised [5].

When considering simulation size and compute time quoted in papers, one should keep Moore's law in mind.

B. Programming Languages

The group's previous numerical propagation algorithm was the 4th order Runge-Kutta method (RK4), implemented on a CPU in Matlab. To create code that would be reused in the group, it was decided to write the program in Matlab because of its prevalence, rather than the more powerful, but complicated, CUDA C [5]. Matlab also provides easy-to-use GPU implementation, with libraries of GPU optimised code that are called automatically for GPU code without any front-end work from the user. The GPU libraries are, however, much more restrictive in their functionality and syntax than the equivalent CPU libraries, leading to code that is more complex than necessary. A characteristic example is the `arrayfun` method that operates a given function on every point in an input array, using separate GPU cores for each point (parallelism). `arrayfun` cannot take constant inputs, only arrays of identical dimension, thus arrays must be created and filled with a desired constant if any constant value is needed. The GPU function called cannot use `global` variables either.

C. Algorithms

Many numerical algorithms exist to solve Ordinary- and Partial-Differential Equations (ODEs and PDEs), including Schrödinger's equation, with different precisions, time-scaling, memory requirements, ability to deal with time-dependent potentials, and implementation difficulties. In surveying the literature, of particular interest is the comparison of different propagators by Leforestier et al. [6], presentation of methods in [10], and discussions in [7].

Mathematically, the problem to be solved is Schrödinger's equation:

$$\begin{aligned} i\hbar\frac{\partial\psi}{\partial t} &= H\psi \\ H &= -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{x},t) \end{aligned} \tag{2}$$

The exact solution is:

$$\psi(\mathbf{x},t_2) = U(t_1,t_2)\psi(\mathbf{x},t_1) \tag{3}$$

where

$$U(t_1,t_2) = \exp\left(\frac{-i}{\hbar}\int_{t_1}^{t_2} H dt\right) \tag{4}$$

and $\exp(...)$ is defined by its Taylor expansion:

$$\exp(A) = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ... \tag{5}$$

Propagation methods differ in how they achieve the evaluation of (3), however most will propagate the wavefunction forward in time, incrementally, by small finite timesteps dt. Presented below are a selection of propagation algorithms. Key features are summarised in TABLE I.

1) ***1st order Euler****:* H is an operator, so $\exp\left(\frac{-i}{\hbar}\int_{t_1}^{t_2} H dt\right)$ is a complicated object. Considering infinitesimal timestep dt, (4) becomes:

$$\begin{aligned} U(t,t+dt) &= \exp\left(\frac{-iH(t)dt}{\hbar}\right) \\ &= 1 + \left(\frac{-iH(t)dt}{\hbar}\right) + \frac{1}{2!}\left(\frac{-iH(t)dt}{\hbar}\right)^2 + ... \end{aligned} \tag{6}$$

with the 1st order truncation of (6) being Euler's method. Errors increase linearly in time, causing results to diverge quickly. Euler is surpassed most other methods.

2) ***Crank-Nicholson****:* Effectively a 2nd order trapezium rule approach, Crank-Nicholson was successfully used in 2004 by NVIDIA to rapidly solve 2D water wave equations on a GPU [11].

3) ***Runge-Kutta 4th order (RK4)****:* A very popular ODE solver, effectively a 4th order Simpson's rule approach. Adaptive step-size RK methods exist that vary step-size during propagation, decreasing effort and improving accuracy for a given computational effort. Methods should be set to return samples uniformly distributed in time, as data analysis is often easier in this format. Implementation of adaptive RK methods is not straightforward, however.

4) ***Leap-frog****:* Implemented in the Cavendish already in Prof. Barnes's lab. The Real and Imaginary parts of ψ are propagated at alternating timesteps i.e. Real at t, Imaginary at $t + dt$, Real at $t + 2dt$, Imaginary at $t + 3dt$, etc... Errors in this method are bounded because Real and Imaginary timestep errors are such that they cancel out. $|\psi|$ usually oscillates randomly around 1 during propagation, but doesn't diverge, causing the error in the final answer to be bounded (unlike 1st and 2nd order Euler methods).

*5) **Split-Operator (SpOp)**:* The Hamiltonian is split into Kinetic and Potential parts, then the time-evolution operator (4) is approximated by the product of separate Kinetic and Potential time-evolution operators. The Kinetic evolution operator has a simple Fourier-space representation, so by repeatedly transforming between Real- and Fourier-space, the Potential and Kinetic evolution operators propagate the wavefunction in staggered steps. SpOp utilises Fourier-transforms, allowing for the potential of speed-increase via hardware capable of fast Fourier-transforms, such as GPUs or dedicated Digital Signal Processors.

*6) **Lanczos**:* Difficult to implement but can be used for time-dependent Hamiltonians [6] and is potentially much faster than SpOp. Spatial requirements could be prohibitive as Lanczos requires the calculation of $H\psi, H^2\psi, H^3\psi, ..., H^n\psi$ per timestep, each term being represented as a large matrix.

*7) **Chebyshev**:* As of 1991 [6] the Chebyshev method does not work on time-dependent Hamiltonians, but was (and still is [10]) one of the fastest and most accurate propagation methods.

*8) **Others**:* For further inspiration, algorithms developed for other physical PDEs, such as the Stömer-Verlet method for solving many-body classical mechanics problems [12], have the potential to be adapted to solve other physical systems.

A summary and comparison of the above algorithms, and more, are given in TABLE I.

III. Propagator implementation

It was decided the split-operator (SpOp) method would be best to pursue, due to its moderate implementation difficulty and potential for speed-up (over the currently implemented RK4 method) from heavy utilisation of Fourier-transforms which can be performed efficiently on GPUs. While SpOp is a lower order algorithm than RK4, implementation and hardware can allow SpOp to produce faster simulations than RK4. SpOp was chosen over Leapfrog because SpOp had not yet been implemented in the Cavendish yet, allowing a new algorithm to be tested. While Lanczos offers a faster alternative to SpOp [6], implementation difficulty and memory requirement concerns weighed against its favour. Chebyshev isn't appropriate as it cannot handle time-dependent Hamiltonians, so cannot run dynamic simulations.

A. RK4

[4] used an RK4 propagation method that can preserve ψ norm to a fractional error of 10^{-6}. The RK4 method from [4] was adapted to 3D, and implemented on a GPU, so it could be used to test and verify the SpOp implementation.

B. Split Operator Implementation

Writing the Hamiltonian H in terms of Kinetic, K, and Potential, V, terms:

$$H = K + V$$

one can approximate the propagator from (6) in several ways:

$$\exp\left(\frac{-iHdt}{\hbar}\right) = \exp\left(\frac{-iKdt}{\hbar} + \frac{-iVdt}{\hbar}\right) \tag{7a}$$

$$\approx \exp\left(\frac{-iKdt}{\hbar}\right)\exp\left(\frac{-iVdt}{\hbar}\right) \tag{7b}$$

$$\approx \exp\left(\frac{-iKdt}{2\hbar}\right)\exp\left(\frac{-iVdt}{\hbar}\right)\exp\left(\frac{-iKdt}{2\hbar}\right) \tag{7c}$$

$$\approx \exp\left(\frac{-iVdt}{2\hbar}\right)\exp\left(\frac{-iKdt}{\hbar}\right)\exp\left(\frac{-iVdt}{2\hbar}\right) \tag{7d}$$

(7b), (7c), and (7d) are approximations as K and V are operators that do not necessarily commute for all time. The Taylor expansion of (7a) includes both $K \cdot V$ and $V \cdot K$ terms, preventing the collection of terms into the specific operator ordering of (7b). (7b) is a 1st order approximation, while (7c) and (7d) are 2nd order [8]. SpOp methods can be extended to 3rd order, as demonstrated in [9].

The key feature of SpOp is that K has a convenient Fourier-space representation. While Real-space $K \propto \nabla^2$, from the Fourier-transform properties of derivatives [13], Fourier-space $K \propto |\mathbf{k}|^2$, where $\mathbf{k}$ is the Reciprocal-space vector. Thus, upon Fourier-transforming, the operation of K becomes that of multiplication. Additionally, since both V and $|\mathbf{k}|^2$ can be represented as matrices, they can be exponentiated explicitly so no truncation of (6) is required.

Fourier transforming large matrices can be done efficiently and quickly on GPUs - the code created utilised this.

SpOp is by definition norm-preserving since the only operations are multiplication by unitary exponentials. $|\psi|$ is

Method name	Algorithm order	Implementation difficulty	Stability	Notes and restrictions
Euler	1st	Easy	Unstable	-
Second-order differencing	2nd	Moderate	Unstable	-
Crank-Nicholson	2nd	Easy	Stable*	-
Leap-frog	2nd*	Moderate	Stable*	-
Split-operator (SpOp)	1st, 2nd, 3rd+*	Moderate	Stable	Unitarity conserved, Energy not
Runge-Kutta 4 (RK4)	4th	Easy	Stable*	Unitarity not conserved
Lanczos	"High" [6]	Difficult	Stable	-
Chebyshev	Exponential	Difficult	Unstable	No time-dependent Hamiltonians

TABLE I: Comparison of different propagation algorithms. Information from [6]–[10]
*implementation dependent

therefore not an appropriate measure of SpOp's simulation accuracy, as it is for RK4. It is thus necessary to test SpOp's limits for different setups, against RK4 with known error, to discover SpOp's accuracy for different ranges of the algorithm parameters (e.g. timestep).

IV. Physical adsorbate potentials

While hard-wall potentials have been studied in depth [14], physical He-surface potentials arise from electromagnetic interactions between the He s-orbital and surface outer electrons [1], which depend on the distance to the surface [4], and are thus 'soft' potentials, $V(x, y, z)$.

Similar to [4], $V(x, y, z)$ is modelled by a 'perpendicular surface potential' $V_{perp}(z)$, dependent only on height z above the surface (describing the He-surface electromagnetic interaction), and a 'lateral corrugation function' $\xi(x, y)$, dependent on the lateral positions x, y. $V(x, y, z)$ is then constructed by vertically shifting $V_{perp}(z)$ depending on the adsorbate height at a particular lateral point - i.e. $V(x, y, z) = V_{perp}(z - \xi(x, y))$.

A. Lateral corrugation functions

The requirements of a corrugation function describing adsorbate shape are that it should fall to 0 smoothly within specified limits, have easily variable height and width, and have a single peak. Two forms that fit this criteria are Hann and Gaussian functions. [4] uses Hann functions, stating they approximate the lateral corrugation well. Gaussian and Hann functions can be parameterised to give very similar shapes (see Fig.1) so the choice of function will likely have little effect on the results. Hann functions are a single oscillation of a sinusoid, starting and ending at 0. Gaussians, on the other hand, tend to 0 at infinity, so will never be exactly 0 in simulations. The choice is arbitrary, with Gaussians being chosen here because physical (Coulomb) potentials never decay exactly to 0.

For 2D space, $\mathbf{r} \equiv (x, y)$:
Hann function, height h, width w, centre $\mathbf{r}_0$:

$$z_{Hann}(\mathbf{r}) = \begin{cases} \frac{h}{2}\left(1 + \cos\frac{2\pi|\mathbf{r}-\mathbf{r}_0|}{w}\right) & |\mathbf{r} - \mathbf{r}_0| < w/2 \\ 0 & |\mathbf{r} - \mathbf{r}_0| > w/2 \end{cases} \tag{8}$$

Gaussian function, height h, standard deviation σ_r, centre$\mathbf{r}_0$:

$$z_{Gauss}(\mathbf{r}) = h \cdot \exp\left(-\frac{|\mathbf{r} - \mathbf{r}_0|^2}{2\sigma_r^2}\right) \tag{9}$$

Using the parameters in [4], $h = 1.61$Å, $w = 5.50$Å, letting $\mathbf{r} \to x$ so $\mathbf{r}_0 \to x_0 = 100$Å, and setting $\sigma_x = \frac{w}{6}$ (6 standard deviations includes 99.7% of z_{Gauss} area) gives Fig.1.

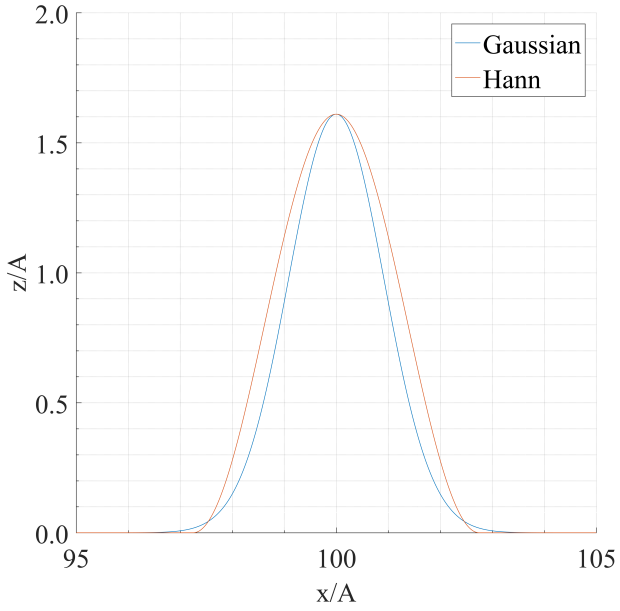


Fig. 1: Gaussian vs. Hann functions.
Height $h = 1.61$Å, Hann width $w = 5.50$Å, Gaussian standard deviation $\sigma_x = \frac{w}{6}$, offset $x_0 = 100$Å

B. Perpendicular surface potentials

1) Exponential potential: [4] used exponentials to represent soft potentials, making the surface potential $V_{exp}(x, y, z)$:

$$V_{exp}(z) = V_0 \cdot \exp\left(-\frac{1}{z_c}(z - z_0)\right) \tag{10}$$

V_0 set equal to the wavefunction's energy, characteristic length scale $z_c = \frac{1}{2.06}$Å, and offset z_0 set arbitrarily to ensure the wavefunction was fully reflected. $z_c = \frac{1}{2.06}$Åwas chosen by [4] to physically represent the He-potential of Na adsorbates on Cu.

2) Morse potential: Exponentials are purely repulsive whereas physical systems are thought to exhibit attractive 'Van der Waals'-style features near the surface [14], [15]. Morse potentials show attractive characteristics with short-range repulsion, providing model potentials for physical surfaces:

$$V_{Morse}(z) = D\left(e^{-2a(z-z_0)} - 2e^{-a(z-z_0)}\right) \tag{11}$$

$D =$ potential well depth, $z_0 =$ well minimum position, and a characterises $V_{Morse}(z = 0)$ magnitude and well width - see Fig.2.

3) Morse-like potentials: To see how features develop when moving from a purely repulsive to attractive potential, it is instructive to develop a potential that 'morphs' from exponential to Morse continuously. While one could start from a generic sum of exponentials:

$$V_{General} = A_1 \cdot e^{-a_1(z-z_1)} + A_2 \cdot e^{-a_2(z-z_2)}$$

and impose turning-point and limiting case criteria, the algebra quickly becomes intractable with 6 free parameters. Instead, if one takes inspiration from the Morse potential, the desired result can be obtained with less difficulty:

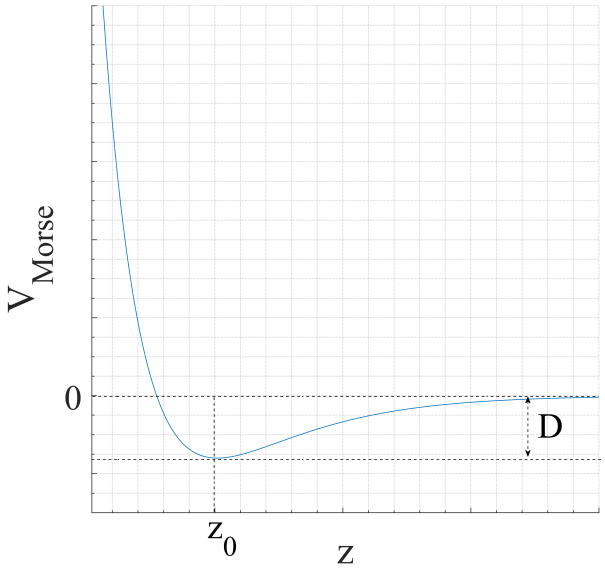


Fig. 2: Morse potential, used to model physical surfaces with attractive features. D characterises the potential well depth and z_0 the well minimum point.

$$V_{MorseLike} = D(e^{-2a(z-z_0)} - \alpha e^{-a(z-z_0)}) \tag{12}$$

where α is a parameter to be varied. $\alpha = 0$ gives $V_{MorseLike} = V_{exp}$, while $\alpha = 2$ gives $V_{MorseLike} = V_{Morse}$, if and only if (see Appendix A):

$$a = \frac{1}{z_0} \ln \left[\frac{1}{2} \left(\alpha + \sqrt{\alpha^2 + 4\frac{V_0}{D}} \right) \right]$$

where the parameters are those from (10) and (11), except V_0 which is now fixed by z_c (or vice versa):

$$V_0 = D \cdot \exp\left(\frac{z_0}{z_c}\right) \tag{13}$$

For $z_0 = 2$Å, $V_0 = 100$meV, $D = 10$meV, and $\alpha = 0 \to 2$ in increments of 0.2, the Morse-like potential is plotted in Fig.3.

V. Errors

A. Floating-point numbers

Two errors that arise when dealing with calculations with errors around machine-precision are representation errors, from not being able to write the desired real numbers exactly in base-2, and rounding errors, from not having enough bits to store the number exactly.

Representation errors in the timesteps dt cause the simulation end-time to differ from the desired end-time. These errors are expected to build up linearly too, however the magnitude of the representation error is difficult to predict, being dependent on how accurately dt can be represented in base-2. Examples are given in TABLE II.

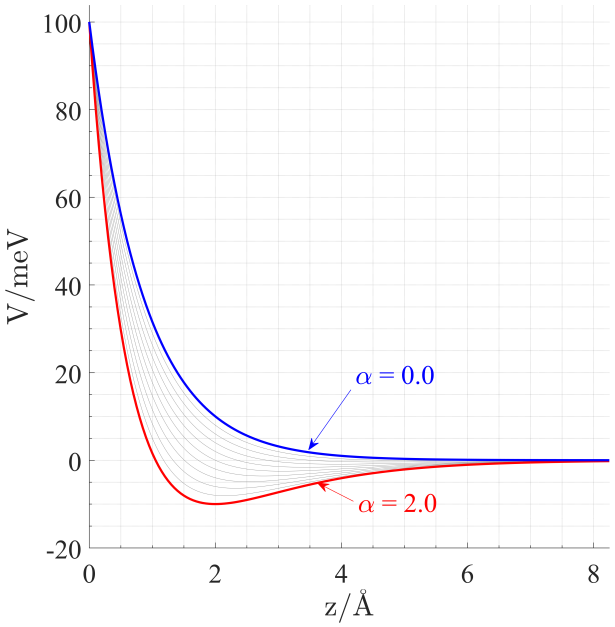


Fig. 3: Morse-like potential
Continuously varying α from $0 \to 2$ takes $V_{MorseLike}$ from $V_{exp} \to V_{Morse}$.

Desired number	Actual stored number
$0.01e-12$	$1.0000000000000000e-14$
$0.001e-12$	$1.0000000000000001e-15$
$0.0001e-12$	$9.9999999999999998e-17$

TABLE II: Numerical precision of floating-point numbers. When desired numbers are not exactly representable in base-2, the numbers are truncated.

B. Fractional errors

The usual method for measuring accuracy is to compute the fractional error between variables, however throughout this project the functions used often contain many values close to zero, causing huge fractional errors for discrepancies which are, objectively, irrelevant. This is demonstrated by generating a Standard Normal Distribution, adding random noise ($\sim 10^{-3}$ magnitude), and plotting the fractional error - see Fig.4.

It is proposed to use a 'normalised error' to characterise differences between variables. When calculating fractional errors ϵ_i^{frac}, where i is the i^{th} grid component of the variables quantised to grids:

$$\epsilon_i^{frac} = \frac{\psi_i^{exact} - \psi_i}{\psi_i^{exact}} \tag{14}$$

the effect of dividing by ψ_i^{exact} is to scale the raw error, representing it as a fraction of the desired value. A similar measure would be to scale by the *mean* value of ψ, which would have a similar effect, representing errors as fractions of

the average value. Here this is named the 'normalised error' ϵ_i^{norm} for point i. Note that division is done by the mean *absolute* value (ordering is important), as any sinusoid will have a mean value of 0, and the aim is to only scale the *magnitude* of the raw error, not the complex phase. Thus:

$$\epsilon_i^{norm} = \frac{\psi_i^{exact} - \psi_i}{\frac{1}{N}\sum_{i=1}^{N} |\psi_i^{exact}|} \tag{15}$$

To characterise the overall discrepancy between two wavefunctions, one considers the mean magnitude normalised error ϵ^{norm}:

$$\epsilon^{norm} = \frac{1}{N}\sum_{i=1}^{N} |\epsilon_i^{norm}| \tag{16}$$

The magnitude is taken before averaging so errors from different points (with different phases) do not cancel. This definition gives a larger value than the average error's magnitude, thereby providing a more conservative accuracy value.

Normalised error should be thought of as a type of fractional error, and as such it is safe to compare normalised and fractional error order of magnitude values directly.

C. Discontinuities

SpOp is very sensitive to discontinuities in the potential gradient (i.e. potentials varying rapidly on the lengthscale of one simulation grid). Discontinuity errors are easily visible in SpOp since they cause small, constant size, high-frequency spatial oscillations throughout the wavefunction (an artefact of the utilisation of Fourier-space, errors introduced at high values in Fourier-space cause oscillations in Real-space).

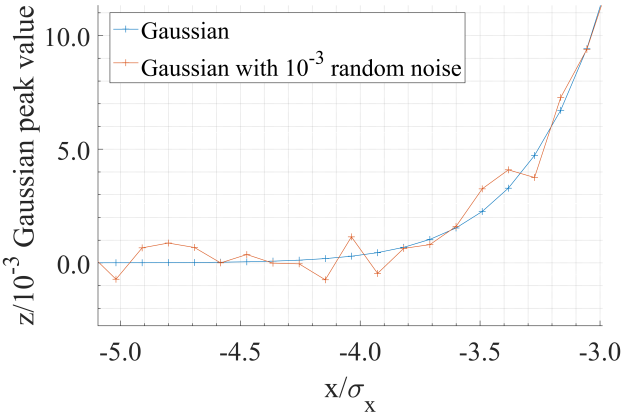
RK4 has no issues with discontinuities, presumably requiring timesteps so small the wavefunction moves less than one pixel per timestep, allowing rapid changes on the scale of one pixel to cause no issues. For such timesteps, SpOp also works fine with discontinuities.

D. Others

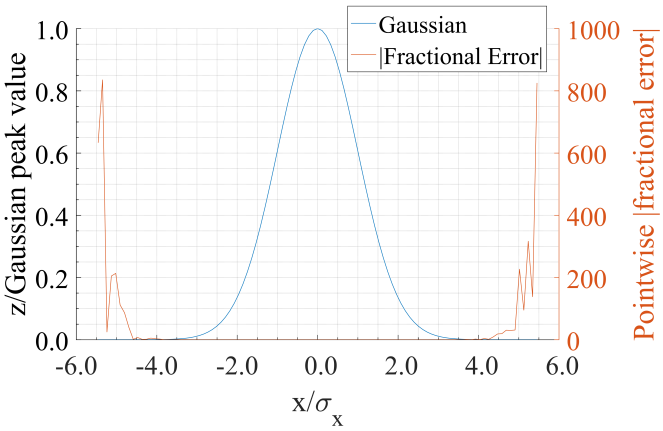
Various other errors (caused physically and computationally) were considered before configuring the simulation setup, but will not be considered here. For example, the effect of wavefunction spreading during simulation would not affect the Form Factor shape, but will prevent a wavefunction being isolated from the potential if the wavefunction spreads out appreciably. Wavefunctions with the largest possible spatial extent (within simulation parameters) were chosen for simulations since they spread out slower.

VI. Verification and Simulation Limits

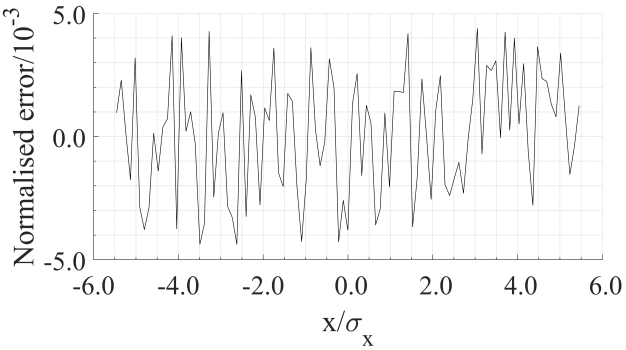
The SpOp propagation method was verified by comparison to analytic plane wave solutions, and RK4-generated solutions to flat and Gaussian repulsive soft-wall potentials. The RK4-generated solutions are kept accurate to 10^{-6} by monitoring $|\psi|$ and restarting the simulation with half the propagation timestep if it differs from 1 by more than 10^{-6}.



(a) Gaussian plotted with and without random noise at a $\sim 10^{-3}$ level of the Gaussian's peak value, $= 1$.



(b) Gaussian plotted with magnitude of fractional error in noisy-Gaussian. Noisy-Gaussian deviations from Gaussian are so small they cannot be seen on the plot.



(c) Normalised error per point. All values $\sim 10^{-3}$ which is the relative error given to this setup.

Fig. 4: Fractional and Normalised errors

A. Plane wave with analytic solution

Plane wavefunctions of the form:

$$\psi(\mathbf{x}, t = 0) = e^{i\mathbf{k}\cdot\mathbf{x}} \tag{17}$$

have analytic solutions:

$$\psi(\mathbf{x},t) = e^{i\mathbf{k}\cdot\mathbf{x}} e^{-i\frac{Et}{\hbar}}$$

where $\mathbf{k}$ is the wavevector, specifying the wavefunction's momentum $\mathbf{p}$, spatial wavelength λ, and energy E via:

$$\mathbf{p} = \hbar\mathbf{k} \tag{18a}$$

$$\lambda = \frac{2\pi}{|\mathbf{k}|} \tag{18b}$$

$$E = \frac{\hbar^2}{2m}|\mathbf{k}|^2 \tag{18c}$$

where m is the particle mass.

The SpOp method is an exact propagation scheme if there is no potential (V) present, since then $\exp(V) \to 1$ and $\approx \to =$ in (7). For the propagation of plane waves in the absence of potentials, errors are therefore expected to rise linearly with the number of timesteps taken, as floating-point precision errors build up.

Propagating an initial wavefunction of the form (17), with $E = 2.5$meV, and $\mathbf{k} = k\hat{\mathbf{z}}$ for a fixed simulation end-time, using SpOp for different timesteps dt, and calculating normalised errors between SpOp and analytic solutions, one obtains Fig.5, which agree with prediction. Analytic solutions were evaluated at the *actual* simulation end-time for each simulation, not the *desired* end-time, so the comparison was between the wavefunction obtained, and the exact wavefunction at that time - see V-A.

The errors of Fig.5 are close in value to, and behave as if caused by, constant floating-point errors building up, rather than systematic implementation errors, suggesting the SpOp code created is propagating wavefunctions in the desired manner, adding weight to the validity of the implementation created.

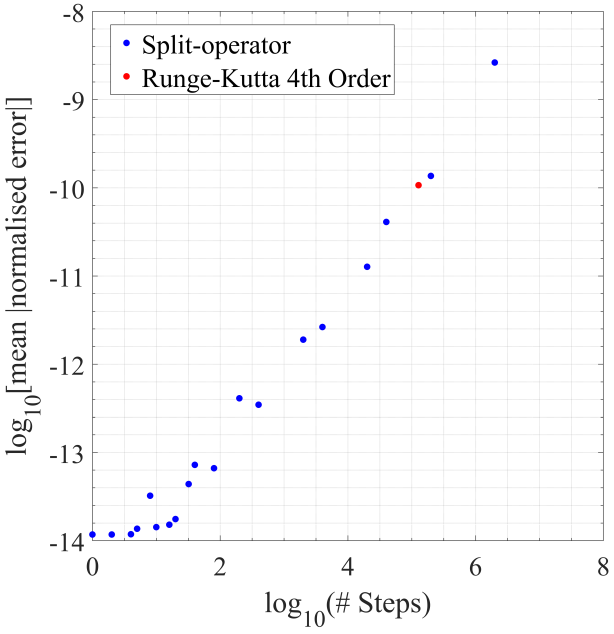


Fig. 5: Mean magnitude of normalised errors for different numbers of steps, given a constant simulation end-time, for Split-operator and RK4 propagation compared to analytic solutions.

B. Flat exponential potential - determining accurate timesteps

$|\psi|$ is conserved using SpOp, so cannot be used to measure propagation accuracy as in RK4. It is therefore necessary to run SpOp for different timesteps, in the presence of a potential with magnitudes and gradients similar to those to be tested, and compare the results to an RK4-propagated solution with a norm fractional error kept below 10^{-6}.

A flat exponential potential setup of the form (10) with $V_0 = 15$eV, $z_c = 1.2$Å, $z_0 = 0$Å, was used with a 10.0meV wavefunction, Energy spread (FWHM) $= 0.75$meV, propagating in a 2D 1024x1024 grid, with side lengths $l_x = l_y = 100$Åfor 14ps, for different timesteps. Results plotted in Fig.6.

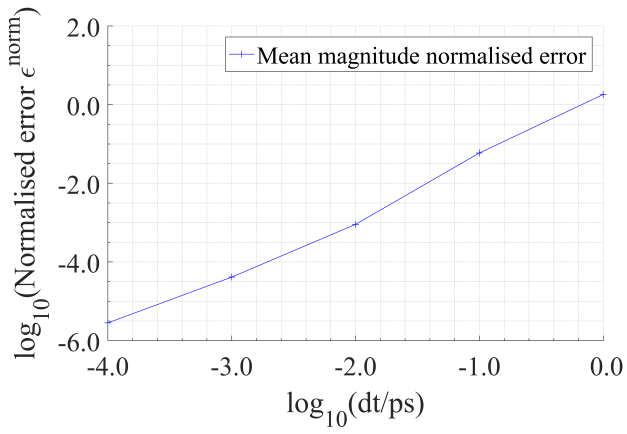


Fig. 6: Normalised errors for split-operator simulations run with different timesteps. Errors calculated relative to an RK4 propagation, accurate to 10^{-6}. As expected, errors decrease when using finer timesteps

C. Spatial quantisation - the Nyquist limit

Since SpOp is implemented through quantising space into a grid of points, saved as matrices, the system is being spatially sampled. Any spatial oscillations will therefore be subject to a Nyquist limit.

Nyquist states at least two sampling points per wavelength of the highest frequency component present are required for complete information of the system to be stored. Highest spatial frequency implies shortest spatial wavelength, so largest wavenumber $|\mathbf{k}|_{max}$. i.e.:

$$\lambda_{min} \geqslant 2dx$$
$$\frac{2\pi}{2dx} \geqslant \frac{2\pi}{\lambda_{min}} = |\mathbf{k}|_{max}$$
$$dx = \frac{l_x}{n_x}$$
$$\frac{l_x|\mathbf{k}|_{max}}{\pi} \leqslant n_x$$

This must be true for all axes $i = x, y, z$, therefore Nyquist's limit is:

$$n_i \geqslant \frac{l_i|\mathbf{k}|_{max}}{\pi} \tag{19}$$

For a Gaussian wavepacket, the Fourier-space representation, and energy, will be Gaussian. Taking the energy-spread to be contained within $5\sigma_E$ of the peak, one obtains:

$$|\mathbf{k}|_{max} = |\mathbf{k}_0| + \frac{5}{\sqrt{2}}\sigma_k. \tag{20}$$

D. Gaussian corrugation function - determining quantisation limits

It was desired to known how sampling more coarsely would affect experimental outcomes. Specifically, the number of pixels used to sample Gaussian potential height and width were investigated. The limits set by these tests allowed higher resolution simulations to be run faster, by optimising the simulation setup.

The measures used for these simulations are 'number of pixels per σ_x' $\equiv px/\sigma_x$ and 'number of pixels per Gaussian height' $\equiv px/h$.

px/σ_x and px/h were investigated from $102.4 \rightarrow 1.6$. This investigation required three simulation setups, each testing a coarser sampling of the potential, as the Nyquist limit was reached before potential sampling effects were noticeable. The limits found are presented in TABLE III, however they are not lower-bounds. px/h testing was prevented further as Nyquist required low-energy (thus low-velocity) wavefunctions, forcing simulation time to be increased, however simulation time was so long that the Gaussian wavefunction envelope spread out significantly, filling up the entire simulation space, preventing a purely 'outgoing' wave to be analysed. The px/h limit found here was 6.4, however this was later found to be caused by quantising the exponential soft-wall potential term VI-E, not the Gaussian, allowing the px/σ_x limit to be revised to 3.2.

E. Exponential potential - quantisation limits

The characteristic length-scale z_c is the distance over which the exponential value drops to a factor of $\frac{1}{e}$. The soft-potential sampling coarseness is characterised by the 'number of pixels per characteristic length-scale' $\equiv px/z_c$. Two setups were used: The first verified (to a normalised error of 10^{-3}) px/z_c was valid for values from $24.6 \rightarrow 0.77$ before Nyquist was reached. The $px/z_c = 0.77$ simulation corresponded to $px/h = 6.4$, the limit set by VI-D.

The second setup verified (to a normalised error of 10^{-3}) $px/z_c = 41.0 \rightarrow 1.28$ values give identical results, however the $px/z_c = 1.28$ simulation had $px/h = 3.2$, lower than the limit set by VI-D, showing the limiting feature of VI-D was not px/h quantisation, but px/z_c quantisation (with a value of 0.77). Hence, we conclude the absolute minimum limit for px/z_c is 0.77, while the new lower limit for px/h is 3.2.

Note, however, that these quantisation limits are for exponentials defined by (10), whereas Morse potentials (11) contain a term with twice this gradient. Therefore it is expected the px/z_c limit should be doubled when working with Morse (11) and Morse-like (12) potentials.

The final sampling limits found are presented in TABLE III.

px/σ_x min. val.	px/h min. val.	px/z_c min. val.	Nyquist condition
1.6	3.4	0.77	$n_i \geqslant \frac{1}{\pi} l_i \lvert\mathbf{k}\rvert_{max}$

TABLE III: Potential quantisation limits. Note, these limits give Form Factors valid to a 10^{-3} fractional error (on both linear and log plots). The simulations that created this data were run with a $dt = 0.01$ps timestep which, via Fig.6, is accurate to a normalised error of 10^{-3}.

VII. Physical investigation

Simulation grid sizes were chosen as coarse as possible within the limits of TABLE III to allow the largest, highest fidelity (in Reciprocal-space), simulations to be run. To plot Form Factors, Fourier-space ψ is interpolated on a 20,000×20,000 grid (the largest allowed by the available hardware), after which the large central specular peak (caused by the large component of ψ normally reflected from the flat adsorbate-free surface areas) is removed, and the amplitude along a ring of constant $|\mathbf{k}|$ - see Fig.7 - (corresponding to wavefunction's energy) is plotted against $\mathbf{k}_x$.

Of particular interest in these investigations are the depths of Form Factor minima (which turn into singularities on a $\log$ plot) since the Kinematic Approximation breaks down around Form Factor singularities [4].

A. Gaussian Adsorbate Form Factors 2D

1) Reference setup: To represent thermal Helium, the reference wavefunction was setup as a 10.0meV plane-wave, windowed by a Gaussian envelope in the direction perpendicular to the surface, incident normally on the surface. The energy spread of the wavefunction is inversely proportional to the width of the Gaussian spatial envelope. For all 2D simulations presented below, the Gaussian spatial envelope had standard deviation $\sigma_z = 15$Å, giving an energy spread (FWHM) of 0.6meV ($\sim 6\%$).

A Gaussian, Morse-like potential as illustrated in Fig.1 and Fig.3 was taken as the reference potential. To enable the Morse-like potential to replicate the physical setup used by [4] as $\alpha \rightarrow 0$, the parameters were chosen as, $z_c = \frac{1}{2.06}$Å, $z_0 = 2.00$Å, $D = 10.0$meV, with V_0 fixed at 615.6meV by

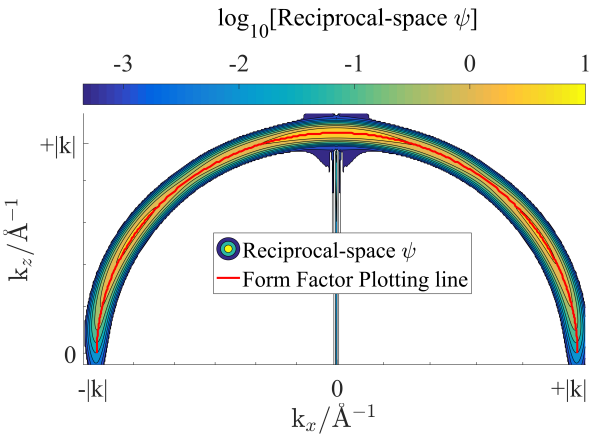


Fig. 7: Contour plot of $\log_{10}$(Reciprocal-space ψ) with the specular peak removed, normalised to 1 and truncated below -2.5 to remove noise. The line along which the Form Factor is measured is plotted in red. Intensity is distributed in a ring of constant $|\mathbf{k}|$ because energy is conserved in this interaction. ψ energy spread is seen by the width of the ring.

(13) - well above the classical reflection point for the reference wavefunction.

The 2D computational setup was that of TABLE IV.

(n_x, n_y, n_z)	(l_x, l_y, l_z)/Å	dt/ps
$(512, 1, 512)$	$(250, 250, 160)$	0.01

px/σ_x	px/h	px/z_c	Nyquist
1.88	5.15	1.55	$n > 327$

TABLE IV:
2D Gaussian simulation parameters and quantisation values - within the limits of TABLE III.

2) Attractive vs repulsive potentials: Simulations were run for Morse-like potentials (12) with $\alpha = 0 \rightarrow 2$ in steps of 0.2, with the resulting Form factors plotted in Fig.8.

The results show that moving from a purely repulsive exponential ($\alpha = 0$) to an attractive Morse potential ($\alpha = 2$) greatly amplifies Form Factor minima, with the outermost minimum moving outward in $|\mathbf{k}_x|$ and the innermost minimum moving inward. The effect of the attractive potential is larger for greater $|\mathbf{k}_x|$, causing Form Factor features at larger $|\mathbf{k}_x|$ to vary most, and create the deepest singularities. The effect of attractive potentials on deepening Form Factor minima was observed for adsorbates of $2, 3$, and $4\times$ the reference adsorbate size (height and width), however the effect became less pronounced for minima that start at lower amplitudes.

3) Increasing adsorbate height: On increasing the Adsorbate's height from $1 \rightarrow 6$ times the reference height h, Fig.9 is obtained.

Repulsive-potential Form Factors are found to be very sensitive to changes in adsorbate height, explaining the significant changes in Form Factor of Fig.9a. The sensitivity with adsorbate height is significantly amplified for attractive potentials - illustrated by the seemingly chaotic changing of Form Factor shape in Fig.9b, and by the effect of a 4% height change, shown in Fig.9c, developing new features throughout the Form Factor.

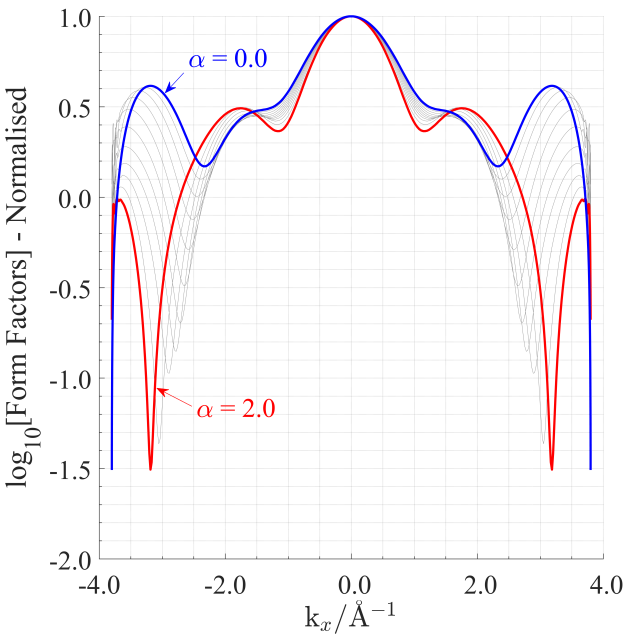


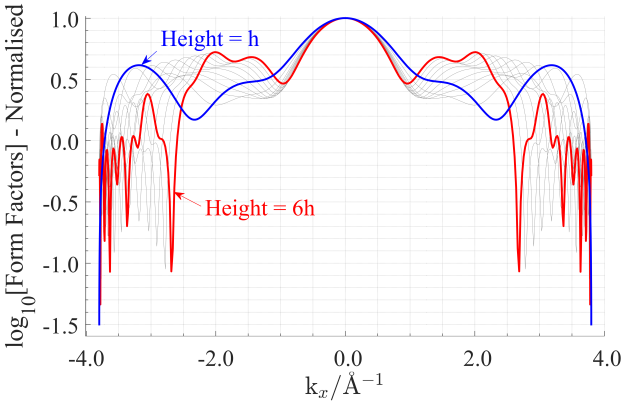
Fig. 8: Form factors reference setup given in VII-A1, varying the potential gradually from a repulsive exponential ($\alpha = 0$) to an attractive Morse potential ($\alpha = 2$).

As height increases, numerous new singularities are introduced at high $|\mathbf{k}_x|$ for both attractive and repulsive potentials. In contrast to varying α, increasing height compresses *all* Form Factor features inward. In Fig.9a a new minimum forms at high $|\mathbf{k}_x|$ between h and $2h$, which moves inward and up as height increases further, finally becoming the small local minimum at $|\mathbf{k}_x| = 1.7\text{Å}^{-1}$. The high sensitivity of attractive potential Form Factors with height make it difficult to determine where and how features develop and change, as seen in Fig.9c.

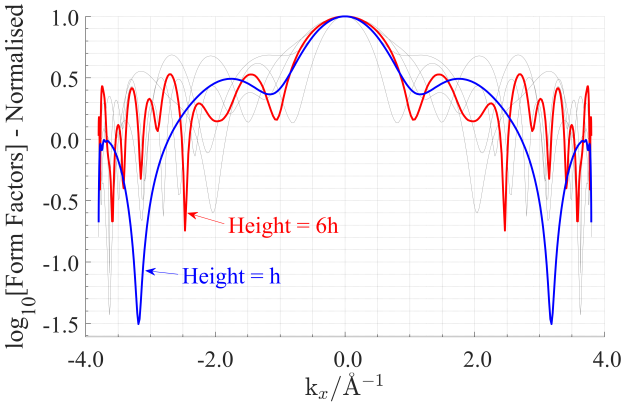
The high number of singularities at large $|\mathbf{k}_x|$, imply the kinematic approximation is likely invalid at large $|\mathbf{k}_x|$ for adsorbates with larger heights - e.g. CO adsorbates.

4) Increasing adsorbate width: On increasing the Adsorbate's width $1 \rightarrow 6$ times the reference width w, one obtains Fig.10.

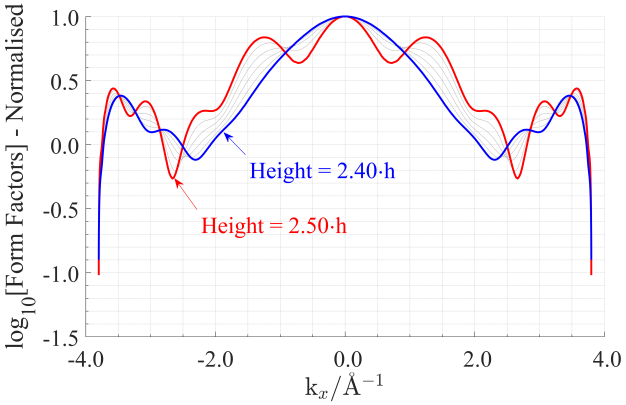
Form Factor shape is less sensitive to width changes than to height changes. For both attractive and repulsive potentials, Form Factor features are shifted inward to lower $|\mathbf{k}_x|$ as width is increased. The effect of the attractive well (VII-A2) decreases for larger adsorbate widths, with attractive and repulsive potential Form Factors tending to one another as width increases (see Width $= 6w$ in (Fig.10a) and (Fig.10b)).



(a) $\alpha = 0.0$ Exponential potential.



(b) $\alpha = 2.0$ Morse potential.

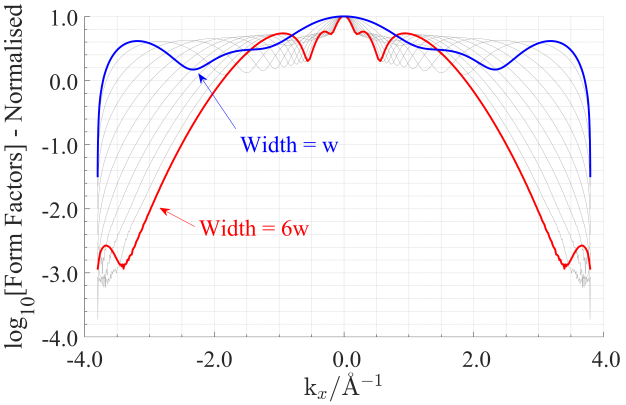


(c) $\alpha = 2.0$ Morse potential. 4% change in height causes significant changes in Form Factor, introducing new turning points throughout.

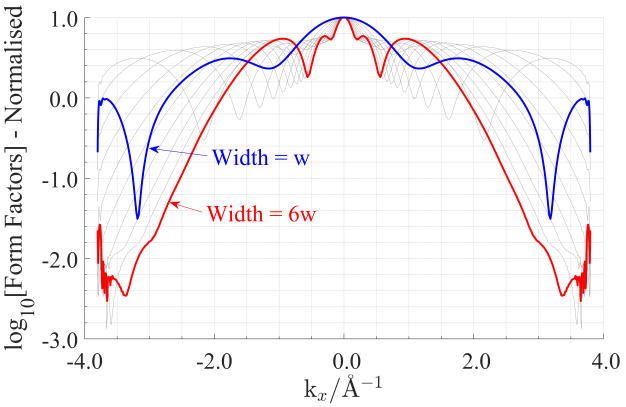
Fig. 9: Form factors for Gaussian adsorbates with height varied uniformly from $1 \rightarrow 6$ times reference height h of VII-A1.

Benzene adsorbates are a physical system exhibiting large lateral extents.

5) Varying well depth: A setup with twice the reference height and width was chosen to investigate the effect of potential well depth since the Form Factor of this adsorbate



(a) $\alpha = 0.0$ Exponential potential.



(b) $\alpha = 2.0$ Morse potential.

Fig. 10: Form factors for Gaussian adsorbates with width $1 \rightarrow 6$ times reference width w of VII-A1.

shows richer features than the reference setup. Well depths, D, of a Morse potential ($\alpha = 2$) were varied from $1 \rightarrow 15$meV in 1meV steps, with V_0 varied to keep z_c at the reference value (VII-A1), and $z_0 = 3.0$Åto ensure total wavefunction reflection at low D. Well depths of $4 - 8$meV are physically representative for He-atom scattering from many metal surfaces [14], [16]. Results plotted in Fig.11.

Similar to VII-A2, increasing adsorbate depth amplifies Form Factor minima, shifting features outward in $|\mathbf{k}_x|$, with the effects greater for larger $|\mathbf{k}_x|$.

B. Gaussian potential form factors 3D

GPU memory limitations put constraints on the simulation sizes that can be run. The largest simulation satisfying Nyquist, and the conditions of TABLE III, are presented in TABLE V.

The setup was identical to VII-A1 except $\sigma_z = 9$Å, giving an energy spread of 1.0meV ($\sim 10\%$), and the wavefunction and adsorbate potential now extend in 3D.

Simulations were run in 2D and 3D for attractive and repulsive potentials, with adsorbate height and width 2 and 3 times the reference setup - results are presented in Fig.12. The 2D simulations were run identically to the 3D setup, except

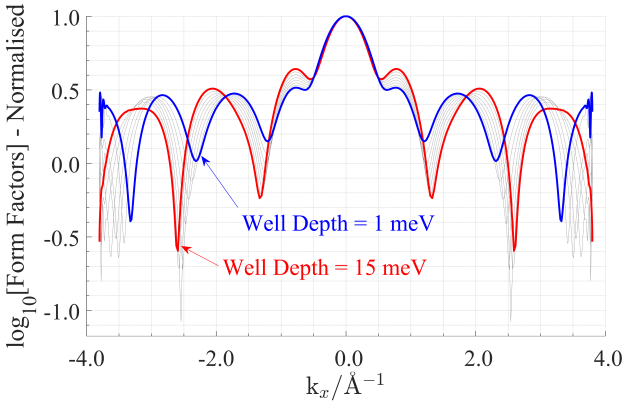


Fig. 11: Form factors for Gaussian adsorbate with Morse potential height = 2(reference height), width = 2(reference height), depth = D

n_y was set to 1. 3D Form Factors were plotted for a 2D cross section through the centre of the Reciprocal-space grid. The setup is cylindrically symmetric, and the Form Factor is also expected to exhibit this symmetry, so the choice of plotting cross-section should be irrelevant.

2D and 3D Form Factors have similar shapes, with peaks and turning points at the same $|\mathbf{k}_x|$. The amplitude of the 3D Form Factor is reduced compared to 2D, which is explained by the amplitude being distributed throughout other Reciprocal-space planes.

For repulsive potentials, the effect of moving to 3D is the reduction of minima depth in the Form Factors. While for smaller adsorbates (Fig.12a) this effectively 'washes-out' the minima at low $|\mathbf{k}_x|$, as adsorbate size is increased (Fig.12b), the minima deepen (as in 2D).

For attractive potentials, moving to 3D has less of an effect than for repulsive potentials, but still appears to reduce minima height for $|\mathbf{k}_x|$ close to 0. For larger $|\mathbf{k}_x|$, the minima depth appears similar to 2D. Minima positions for the larger adsorbate simulation Fig.12d show 3D minima are slightly offset from the corresponding 2D minima.

The Kinematic Approximation is therefore likely to be more valid for smaller adsorbates, as long as the interaction is predominantly repulsive. If a non-negligible attractive potential component is present, Fig.12c and Fig.12d suggests the 2D Form Factors are reasonable approximations (in shape) to 3D. [4] ran 2D dynamic scattering simulations for repulsive potentials only, and therefore one cannot simply conclude that Fig.12c and Fig.12d suggest the results presented in [4] extend directly to this situation, since dynamic simulations with attractive potentials may provide different results.

VIII. Future Work

The potential for future work with the created Toolbox includes, but is not limited to, investigating 2D and 3D Form Factors for oblique incident waves, different Form Factor shape (including more complicated adsorbates such as Benzene), investigating potential 'softness' (varying z_c), and running dynamic scattering experiments in 3D.

3D dynamics experiments will require multiple calls to the 'Generate Adsorbate Potential' function (one per adsorbate per timestep). For 8 adsorbates, and 200 14s simulations (similar to VII-B), it is expected to take 200 hours ($\sim$ 8 days) to run this simulation. The kinematic approximation can therefore be tested exactly in 3D with both attractive and repulsive potentials, including for oblique incidence waves (as physical experiments would be).

(n_x, n_y, n_z)	(l_x, l_y, l_z)/Å	dt/ps
$(128, 128, 512)$	$(90, 90, 90)$	0.01

$px/\sigma_{x,y}$	px/h	px/z_c	Nyquist
2.61	18.3	2.76	$n > 124$

TABLE V:
3D Gaussian simulation parameters and quantisation values - within the limits of TABLE III.

IX. Conclusion

The Split-operator algorithm was implemented to numerically propagate quantum-mechanical wavefunctions in 3D. The implementation was verified as valid by propagating setups and comparing the results to analytic and Runge-Kutta 4th order propagated solutions (that preserved $|\psi|$ to within 10^{-6}). Spatial quantisation limits of a model physical system (repulsive Gaussian adsorbate) were tested, enabling physically significant 3D- and higher fidelity 2D-simulations to be run. The effect of varying adsorbate height, width, and overall size on Form Factor shape was investigated, along with the effect of simulating in 3D.

Appendix A
Morse-like condition

(12) gives:

$$V(z=0) = V_0 = D\left(e^{2az_0} - \alpha e^{az_0}\right). \tag{21}$$

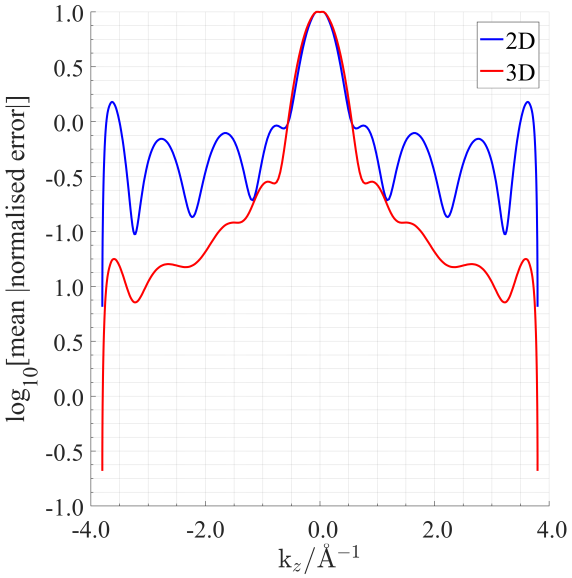
Letting $az_0 = \ln(x)$, one obtains:

$$\frac{V_0}{D} = x^2 - \alpha x \Rightarrow x^2 - \alpha x - \frac{V_0}{D} = 0 \tag{22}$$

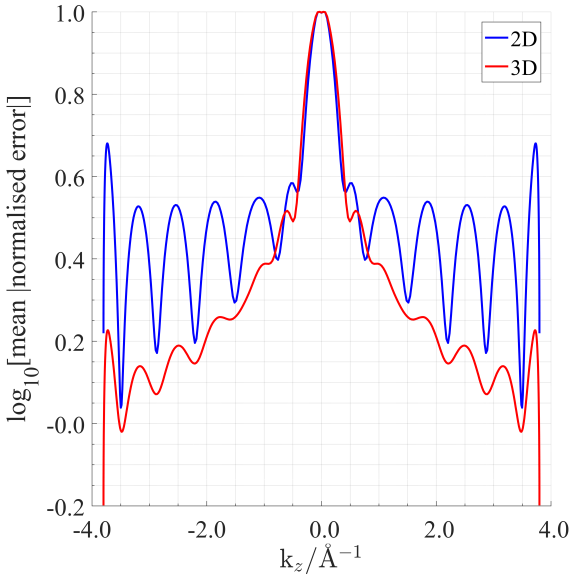
$$\Rightarrow x = \frac{\alpha \pm \sqrt{\alpha^2 + 4\frac{V_0}{D}}}{2}. \tag{23}$$

Since the term in the square root is always greater than α, x_+ must be chosen, since it is requried that $az_0 \in \mathbb{R}$. This gives the condition on a:

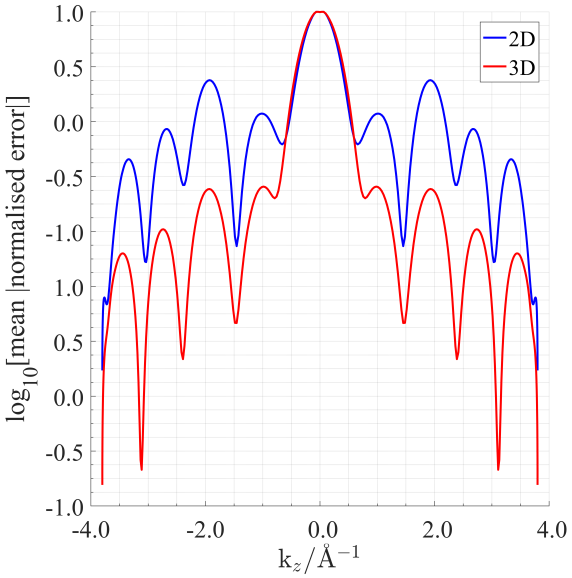
$$a = \frac{1}{z_0} \ln\left(\frac{\alpha + \sqrt{\alpha^2 + 4\frac{V_0}{D}}}{2}\right). \tag{24}$$



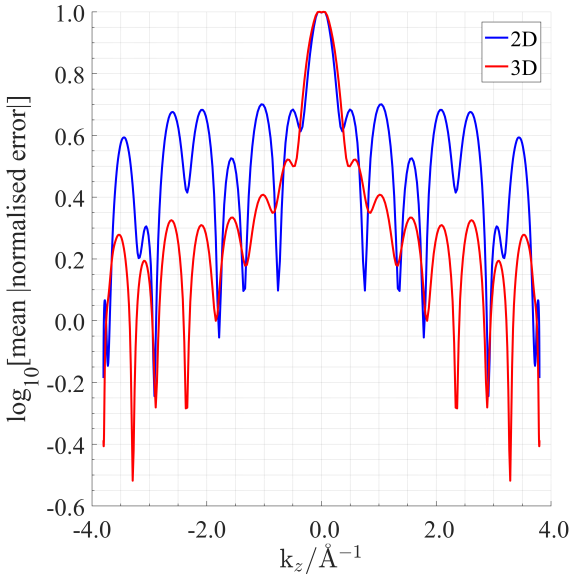
(a) Repulsive potential ($\alpha = 0$) with adsorbate height and width 2× the reference setup of VII-A1.



(b) Repulsive potential ($\alpha = 0$) with adsorbate height and width 3× the reference setup of VII-A1



(c) Attractive potential ($\alpha = 2$) with adsorbate height and width 2× the reference setup of VII-A1



(d) Attractive potential ($\alpha = 2$) with adsorbate height and width 3× the reference setup of VII-A1

Fig. 12: 3D vs 2D Form Factors

REFERENCES

[1] A. Jardine, H. Hedgeland, G. Alexandrowicz, W. Allison, and J. Ellis, "Helium-3 spin-echo: Principles and application to dynamics at surfaces," *Progress in Surface Science*, vol. 84, pp. 323–379, 11 2009.

[2] A. Alderwick, "Instrumental and analysis tools for atom scattering from surfaces," Ph.D. dissertation, University of Cambridge, 9 2009.

[3] A. Lipson, S. Lipson, and H. Lipson, *Optical Physics*. Cambridge University Press, 2010. [Online]. Available: https://books.google.co.uk/books?id=aow3o0dhyjYC

[4] A. R. Alderwick, A. P. Jardine, W. Allison, and J. Ellis, "An evaluation of the kinematic approximation in helium atom scattering using wavepacket calculations," *Surface Science*, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0039602818301511

[5] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st ed. Addison-Wesley Professional, 2010.

[6] C. Leforestier, R. H. Bisseling, C. Cerjan, M. D. Feit, R. Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O. Roncero, and R. Kosloff, "A comparison of different propagation schemes for the time dependent schro¨dinger equation," *J. Comput. Phys.*, vol. 94, no. 1, pp. 59–80, may 1991. [Online]. Available: http://dx.doi.org/10.1016/0021-9991(91)90137-A

[7] H. TalEzer and R. Kosloff, "An accurate and efficient scheme for propagating the time dependent schrdinger equation," *The Journal of Chemical Physics*, vol. 81, no. 9, pp. 3967–3971, 1984. [Online]. Available: https://doi.org/10.1063/1.448136

[8] R. Glowinski, S. J. Osher, and W. Yin, *Splitting Methods in Communication, Imaging, Science, and Engineering*, 1st ed. Springer Publishing Company, Incorporated, 2017.

[9] A. D. Bandrauk and H. Shen, "Improved exponential split operator method for solving the time-dependent schrödinger equation," *Chemical Physics Letters*, vol. 176, no. 5, pp. 428 – 432, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/000926149190232X

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[11] M. Pharr and R. Fernando, *Gpu Gems 2: Programming Techniques for High-performance Graphics and General-purpose Computation*, 1st ed. Addison-Wesley Professional, 2005.

[12] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the störmer/verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[13] G. Arfken, H. Weber, and F. Harris, *Mathematical Methods for Physicists: A Comprehensive Guide*, ser. YBP Print DDA. Elsevier, 2013. [Online]. Available: https://books.google.co.uk/books?id=qLFo_Z-PoGIC

[14] E. Hulpke, G. Benedek, V. Celli, G. Comsa, R. Doak, J. Frenken, B. Hinch, H. Hoinkes, K. Kern, A. Lahee *et al.*, *Helium Atom Scattering from Surfaces*, ser. Springer Series in Surface Sciences. Springer Berlin Heidelberg, 1992. [Online]. Available: https://books.google.co.uk/books?id=oOu2AAAAIAAJ

[15] G. Vidali, G. Ihm, H.-Y. Kim, and M. W. Cole, "Potentials of physical adsorption," *Surface Science Reports*, vol. 12, no. 4, pp. 135 – 181, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016757299190012M

[16] M. Carr and D. Lemoine, "Fully quantum study of the 3d diffractive scattering of he from isolated co adsorbates on pt(111)," *The Journal of Chemical Physics*, vol. 101, no. 6, pp. 5305–5312, 1994. [Online]. Available: https://doi.org/10.1063/1.467384