Methods of Mathematical Physics Coursework

Wasim Rehman: CID: 01439484

The first section provides an outline of the project and the approach used. The following sections contain the code in a modular format so one can step through each part of the code and evaluate results, please note the first section of code sets up the problem by generating the operator matrices and input variables. Finally, there is an Appendix which contains details of other numerical approaches that have improved calculation speed.

Overview

Differential equations in matrix form arise frequently in Quantum Mechanics, especically when a density operator approach is taken to represent a statistical ensemble of state vectors. The solution of such differential equations can require the evaluation of matrix exponentials, which when represented via their Taylor expansions result in infinite sums that may be slow to converge. The complex trapezium rule is a numerical approximation that is computationally lighter than the evaluation of a matrix exponential. In this coursework we will use the complex trapezium rule to evaluate the solution to a differential equation from Wasim Rehman's Masters Project which builds upon previous research in Atom-Molecule Conversion in a Bose Einstein Condensate [1] and to assess whether the complex trapezium is more efficient.

Background

Masters Project Outline:

The experimental realisation of Bose-Einstein condensates (BECs) was one of the major achievements in physics in the late 20th century. Recent progress in confining and manipulating BECs in optical potentials has led to a variety of spectacular results. One interesting problem is the combination of condensed atoms into molecules, which themselves form condensates, and the inverse process of splitting molecules into atoms. The mathematical description of interacting atoms and molecules, which can be converted into one another, and possible dynamical schemes for an effective conversion, is challenging and interesting at the same time. In particular the influence of noise and particle losses, which is of huge experimental relevance, is theoretically not well understood.

Project. In this project you will familiarise yourself with different approximations for the description of interacting atoms and molecules in the ground state of an external potential, leading to classical dynamics on unnusual phase spaces. You will analyse the approximation and the resulting dynamics in detail, and then explore the influence of noise and dissipation described by Lindblad equations.

The system can be described by a Hamiltonian of the form:

$$\hat{H} = \epsilon_a \hat{a}^\dagger \hat{a} + \epsilon_b \hat{b}^\dagger \hat{b} + \frac{\nu}{2\sqrt{N}} (\hat{a}^\dagger \hat{a}^\dagger \hat{b} + \hat{a}\hat{a}\hat{b}^\dagger)$$

where $\hat{H}, \hat{a}, \hat{a}\dagger, \hat{b}, \hat{b}\dagger$ represent the Hamiltonian, atom annihilation, atom creation, molecule annihilation, molecule creation operators respectively, $\epsilon_a, \epsilon_b, \nu$ are parameters that represent the atom energy level, molecule energy level and converting strength respectively, Nrepresents the total number of particles.

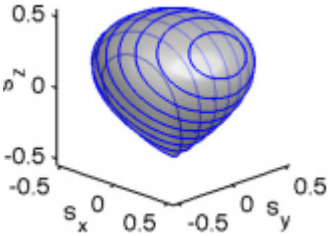
There is a convenient transformation one can apply to express the Hamiltonian in operators that are similar to angular momentum parameters for the Bose-Hubbard model. However, unlike Bose-Hubbard, they now obey a deformed SU(2) algebra due to 1:2 nature of the switching. For this system the Hilbert Space has a dimension of $\frac{N}{2} + 1$

$$\hat{K}_x = \frac{\hat{a}^\dagger \hat{a}^\dagger \hat{b} + \hat{a}\hat{a}\hat{b}^\dagger}{2\sqrt{N}},$$
$$\hat{K}_y = \frac{\hat{a}^\dagger \hat{a}^\dagger \hat{b} - \hat{a}\hat{a}\hat{b}^\dagger}{2i\sqrt{N}},$$
$$\hat{K}_z = \frac{\hat{a}^\dagger \hat{a} - 2\hat{b}^\dagger \hat{b}}{4}.$$

By shifting the zero energy level and introducing $\epsilon = 2\epsilon_a - \epsilon_b$, we have the following symplified Hamiltonian:

$$\hat{H} = \epsilon \hat{K}_z + \nu \hat{K}_x$$

This system has been studied extensively in [1]. The system results in a simple matrix differential equation with sparse matrices. Although not directly relevant to this project, it is interesting to note that in the mean-field limit where $N \to \infty$, the dynamics are constrained to 'tear drop' shape, where s_x, s_y, s_z are the mean-field limits of K_x, K_y, K_z



Extending the model to include Atom Losses

The Masters Project builds upon [1] to study the effect of noise on the system, in particular via atom losses. To model open Quantum systems, a standard approach is to introduce a density operator and work in the space of density matrices which are statistical ensembles of pure quantum states. The evolution of the density matrix is governed by the 'master equation in Lindblad form':

Wikipedia:

In quantum mechanics, the Gorini–Kossakowski–Sudarshan–Lindblad equation (GKSL equation, named after Vittorio Gorini, Andrzej Kossakowski, George Sudarshan and Göran Lindblad), master equation in Lindblad form, or Lindbladian is the most general type of Markovian and time-homogeneous master equation describing (in general non-unitary) evolution of the density matrix ρ that preserves the laws of quantum mechanics (i.e., is trace-preserving and completely positive for any initial condition). The Schrödinger equation is a special case of the more general Lindblad equation, which has led to some speculation that quantum mechanics may be productively extended and expanded through further application and analysis of the Lindblad equation.

$$\dot{\rho} = -\frac{i}{\hbar}[\hat{H}, \rho] + (\hat{L}\rho\hat{L^\dagger} - \frac{1}{2}\hat{L^\dagger}\hat{L}\rho - \frac{1}{2}\rho\hat{L^\dagger}\hat{L})$$

where ρ *is the density operator for the states,* $\hat{H}$ *is the system Hamiltonian,* $\hat{L}$ *is the Lindblad operator which models the interaction with the environment, or 'noise' in the system. Further details on its assumptions on the system/environment interaction can be found here: https://en.wikipedia.org/wiki/Lindbladian*

For this coursework we will introduce the Lindblad operator $\sqrt{\gamma}\,\hat{a}$, which models an environment causing atom losses.

As a result we have the following differential equation for the density operator:

$$\dot{\rho} = -\frac{i}{\hbar}[\hat{H}, \rho] + \gamma(\hat{a}\rho\hat{a^\dagger} - \frac{1}{2}\hat{a^\dagger}\hat{a}\rho - \frac{1}{2}\rho\hat{a^\dagger}\hat{a})$$

As we have allowed the system to include atom losses, we have further possible states and as such the dimension of the sytem Hilbert Space is now $M := (\frac{N}{2} + 1)^2$.

To recap, here is a summary of the variables and their matrix sizes.

$\rho, \hat{H}, \hat{a}, \hat{a^\dagger}$ - MxM matrices, where $M = (\frac{N}{2} + 1)^2$

Note: we will set $\hbar = 1$.

Reformulating the master equation in Lindblad form to a simple matrix differential equation

A Matrix Differential Equation of the form $\dot{X} = AXB$ can be expressed as $vec(\dot{X}) = (B^T \otimes A)vec(A)$ where $vec(X)$ is the vectorised form of X and $\otimes$ denotes the Kronecker product of the matrix. In the case of our equation the derivation is as follows:

$$vec(\dot{\rho}) = A(vec(\rho))$$

where $A = -i(I_M \otimes H) + i(H \otimes I_M) + \gamma((\hat{a\dagger})^T \otimes \hat{a}) - \frac{1}{2} I_M \otimes (\hat{a\dagger}\hat{a}) - \frac{1}{2}(\hat{a\dagger}\hat{a})^T \otimes I_M)$, I_M *is the M-dimenstional identity matrix and* A *is a* $M^2 x M^2$ *matrix. recalling our original system of N particles, this results in A having* $o(N^4)$ *elements. As a result, evaluating the matrix exponential is computationally expensive as we have an infinite sum of a matrix that grows as* $o(N^4)$.

The solution to this differential equation is.

$$vec(\rho) = vec(\rho(0))e^{At}$$

The matrix exponential is computationally demanding to compute, especially via Taylor Series. We will utilise the complex trapezium rule via the Cauchy Integral formula to provide a numerical approximation.

Complex Trapezium Rule for Evaluating Matrix Functions

Cauchy's integral formula enables us to calculate a function via a contour integral in the complex space. In the case of a matrix exponential the Cauchy Integral formula replaces the task of evaluating a matrix exponential with evaluating the expoential of a complex number, which is computationally easier. However, the Cauchy Integral Formula for matrix functions results in a matrix inversion which may be be computationally expensive.

We can extend Cauchy Integral Formula to apply to matrix elements as follows. The contour Γ is chosen such that it surrounds the spectrum of A.

$$f(A) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{f(\zeta)}{\zeta I_M - A} dz$$

Letting $f(A) = e^A$ and denoting quadrature rules with nodes: $\zeta_1, ..., \zeta_n$ and weights $w1, ..., w_n$ then we can approximate a complex integral as follows:

$$e^{At} \approx \frac{1}{2\pi i} \sum_{j=0}^{n-1} w_j e^{\zeta_j} (\zeta_j I - At)^{-1}$$

For the complex trapezium rule we have:

$$w_j = \frac{2\pi}{n} \Gamma'(\theta)$$

Choosing the Contour by using Gershgorin's Circle Theorem

To choose a suitable contour Γ that contains the spectrum of A, we will use Gershgorin's Theorem. Given the nature of the system, the eigenvalues of A are clearly bounded (the explanation is beyond the scope of this project), so we will use a circular contour with equally spaced nodes. For the complex trapezium rule with a circular contour we have:

$$w_i = \frac{2\pi}{n} rie^{\frac{2\pi i}{n}}$$

$$\zeta_i = z_o + re^{\frac{2\pi i}{n}}$$

where z_o, r, n are the the contour centre, contour radius and number of nodes respectively.

To choose the radius and ensure the spectrum of A is encapsulated we utilise Gershgorin's Circle Theorem to find the set of discs that contains the eigenvalues in the complex plane. We can use the outputs of the Gershgorin theorem to calculate $\xi_R = max \bigcup_{k=1}^{M} (Re(a_{kk} + R_k))$, $\xi_L = min \bigcup_{k=1}^{M} (Re(a_{kk} + R_k))$, $\eta_+ = max \bigcup_{k=1}^{M} (Im(a_{kk} + R_k))$ $\eta_- = min \bigcup_{k=1}^{M} (Im(a_{kk} +$ Then we choose $z_o = (\xi_R + \xi_L)/2 + i(\eta_+ + \eta_-)/2$ and $r = ((\xi_R - \xi_L)^2 + (\eta_+ - \eta_-)^2)/2 + m$, where m is a margin, an input for the calculation.

Using our contour and complex trapezium formula, we can now proceed to numerically approximate e^{At}.

Results

In this section we will compare the solutions of the master equation in Lindblad form via the Kronecker Representation using:

i) Matlab's matrix exponential function '**expm**'. It is worth noting that Matlab's 'expm' function utilises the Scaling and Squaring Method for approximating matrix exponentials as outlined in [2] and [3]

ii) A Matlab function '**expm-Taylor**' which calculates the matrix exponential via the Taylor Expansion

ii) A numerical approximation using the **complex trapezium rule**

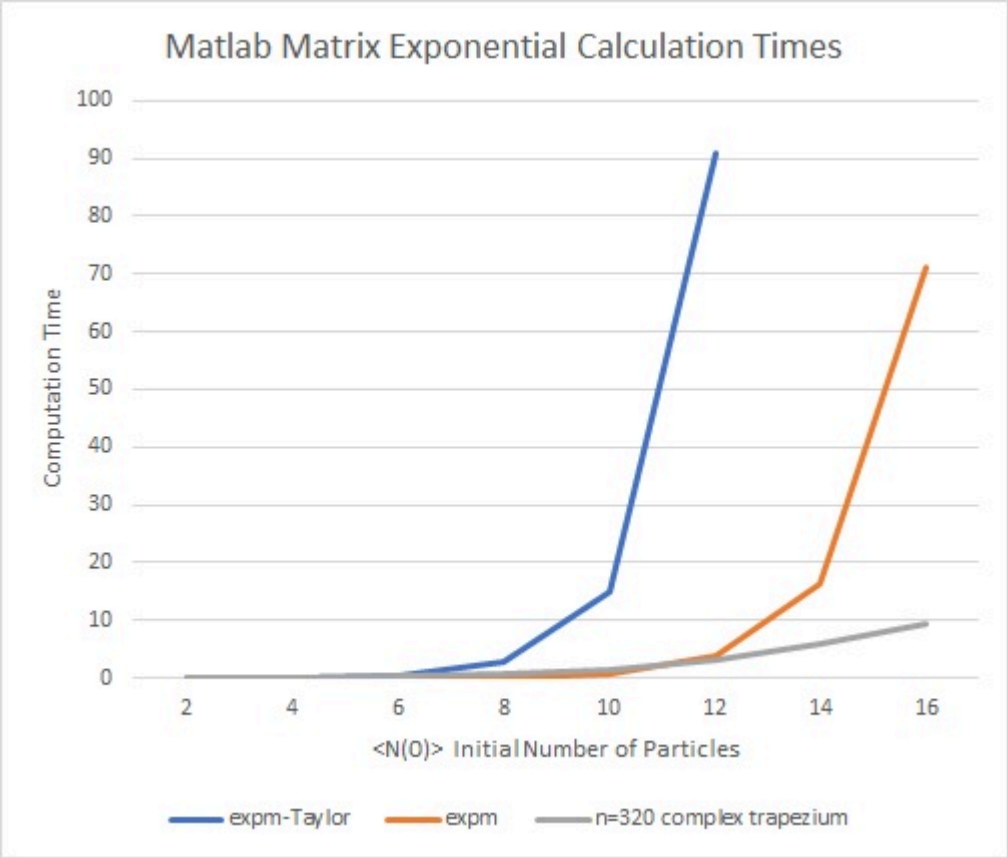
We investigated the speed of calculation, the accuracy and the number of nodes required. To evaluate the density matrix output we calculated the expectation value of the Number operator $< \hat{N} >= Tr(\rho \hat{N})$ at the end time of the simulation, which represents the expected number of particles at the end time. This is simpler and more intuitive than comparing two density matrices

Results are presented below. The cells highlighted in green show the values of N and n where the complex trapezium is quicker to calculate than expm and the error is less than 10%. The cells highlighted in bold the values of N and n where the complex trapezium rule is quicker to calculate than expm-Taylor and the error is less than 10%

<N(0)>	#entries A		Expm - Taylor	Expm - Scaling Squaring	Complex trapezium approxi					
					n=5	n=10	n=20	n=40	n=80	n=1
2	16	error vs expm			5.5E-01	5.8E-02	6.0E-02	2.7E-03	6.4E-03	6.5E
		time	0.009	0.008	0.016	0.016	0.016	0.016	0.025	0.0
4	81	error vs expm			1.3E+01	3.3E-01	**4.9E-03**	**4.6E-03**	**4.7E-03**	**4.7E**
		time	0.054	0.010	0.016	0.018	**0.019**	**0.026**	**0.032**	**0.0**
6	256	error vs expm			2.4E+00	6.0E+01	1.1E-02	**1.4E-03**	**1.3E-03**	**1.3E**
		time	0.500	0.022	0.018	0.023	0.031	**0.046**	**0.086**	**0.1**
8	625	error vs expm			3.8E+04	2.4E+04	8.9E+02	**1.1E-03**	**1.1E-03**	**1.1E**
		time	2.820	0.122	0.025	0.038	0.056	**0.099**	**0.185**	**0.3**
10	1296	error vs expm			5.8E+06	2.6E+06	1.1E+06	6.3E-01	**4.6E-04**	**4.6E**
		time	15.032	0.870	0.057	0.063	0.110	0.207	**0.398**	**0.8**
12	2401	error vs expm			2.2E+09	1.1E+09	1.0E+09	2.1E+05	**2.5E-04**	**2.6E**
		time	90.786	3.763	0.070	0.110	0.207	0.392	**0.766**	**1.6**
14	4096	error vs expm			6.7E+11	3.3E+11	1.9E+11	3.9E+09	**3.0E-04**	**1.8E**
		time		16.338	0.107	0.188	0.366	0.706	**1.377**	**2.7**
16	6561	error vs expm			2.5E+14	1.3E+14	3.4E+13	1.8E+13	2.6E+01	1.5E
		time		71.080	0.170	0.312	0.629	1.206	2.415	4.7

For the above analysis we have $N = 2, \epsilon = 1, \nu = 1, \gamma = 0.05, m = 0.1$ *and the system is evolved to t = 30.*

The chart below compares the calculation times for the three approaches. As can be seen, the complex trapezium rule is superior to 'expm' which is superior to 'expm_Taylor'.



Conclusions and Further Areas of Investigation

The results show that given a suitable number of nodes, the complex trapezium approximation is accurate and computationally faster than evaluating the matrix exponential via a Taylor Series and via Matlab's built in 'expm' function/algorithm.

For a system of greater than 12 particles, the Taylor Series method failed to calculate on Matlab and for greater than 16 particles the 'expm' would not calculate. For such a system, the complex trapezium rule could be used to solve the differential equation.

The complex trapezium method has the added advantage that the calculation could be parallelised giving further speed improvements, code for this is included in the Appendix.

To further improve the efficiency use we could investigate the following areas:

- How to choose the number of nodes
- Investigating different contours and quadratures

This project has been a useful in establishing that the complex trapezium rule is an accurate and fast numerical approximation to the matrix exponential that arises in the Masters Project differential equation.

1. Provide input parameters & generate matrices for the master Lindblad equations and the master equations in Kronecker form

Provide input parameters

```
clear
%define input parameters here >> enter here
N = 8;  % number of Atoms
Epsilon = 1;  % energy level difference parameter
Nu = 1; % switching energy parameetr
gamma = 0.05; %atom loss Lindblad coefficient
margin = 0.1; % to ensure the contour of integral surrounds the spectrum of A
end_time = 30;

%calculate dimension of Hilbert Space
Hil_Dim = ((N/2)+1)^2;

%define an initial state vector - case all molecule state  >> enter here
psi = zeros(Hil_Dim,1);
psi(end) = 1;

%define an initial state vector - case: single atom state
% psi = zeros(Hil_Dim,1);
% psi(2) = 1;
```

Create state vectors and index vectors

```
%size is ((N/2)+1)^2)  - we allow a (0,0) state
% The Hilbert space is all combinations of atoms and molecules
% where total number of particles <=N. This results in the (N/2+1)^2 Dimension space

%create columns vectors: at = atom number in state, mol = molecule number
%in state
counter = 0;
at=zeros(Hil_Dim,1);
mol =zeros(Hil_Dim,1);
for m=0:N/2
    for a = 0:(N -2*m)
        counter = counter+1;
        at(counter) = a;
```

```
        mol(counter) = m;
    end
end

%a vector of total particle number
num = at+2*mol;
```

Calculate $\hat{K}_x, \hat{K}_y, \hat{K}_z, \hat{a}, \hat{a}\dagger, \hat{H}, \hat{L}, \hat{L}\dagger, A$

```
Kz =zeros(Hil_Dim,Hil_Dim);
for i =1:Hil_Dim
    Kz(i,i) = (at(i)-2*mol(i))/4;
end

Kx = zeros(Hil_Dim,Hil_Dim);
for i =1:Hil_Dim
    output = Kx_hat_ext(N,at(i),mol(i));
    index_an = intersect(find(at ==output(2,1)),find(mol==output(3,1)));
    Kx(index_an,i) = output(1,1);
    index_cr = intersect(find(at ==output(2,2)),find(mol==output(3,2)));
    Kx(index_cr,i) = output(1,2);

end
Kx = sparse(Kx);

Ky = zeros(Hil_Dim,Hil_Dim);
for i =1:Hil_Dim
    output = Kx_hat_ext(N,at(i),mol(i));
    index_an = intersect(find(at ==output(2,1)),find(mol==output(3,1)));
    Ky(index_an,i) = -output(1,1)/1i;
    index_cr = intersect(find(at ==output(2,2)),find(mol==output(3,2)));
    Ky(index_cr,i) = output(1,2)/1i;

end
Ky = sparse(Ky);

%Calculate a - atom annihilation operator
a = zeros(Hil_Dim,Hil_Dim);
for i =1:Hil_Dim
    if isempty(intersect(find(at ==at(i)-1),find(mol==mol(i))))
        ind = 1;
    else
        ind = intersect(find(at ==at(i)-1),find(mol==mol(i)));
    end
    a(ind,i)=sqrt(at(i));
end


%Hamiltonian
H = Epsilon*Kz + Nu * Kx;
H = sparse(H);
```

```
%Number of particles Operator
N_hat =2*a'*a - 4*Kz;
N_hat = sparse(N_hat);

%Linblad Operator
L = sqrt(gamma)*a;
L = sparse(L);
L_dag = sqrt(gamma)*a';
L_dag = sparse(L_dag);

%Kronecker formulation matrices
Id = speye([Hil_Dim Hil_Dim]);
A = sparse(kron(Id,(-1i*H -0.5*L_dag*L)) + kron((1i*H.' - 0.5*(L_dag*L).'),Id) +kron(L_
rho = psi*psi'; %initial density matrix
rho_init = psi*psi'; %initial density  matrix
```

Complex Trapezium Derivations & Gershgorin's Theorem to find a suitable contour that encompasses all of the eigenvalues

```
% Create a vector which will denote the centre of the balls referenced by
% Gershgorin
a = diag(A);

% Create R vector which will denote the size of the balls located at
% $a_{kk}$
R = zeros(Hil_Dim^2,1);

for i = 1:Hil_Dim^2
    R(i) = sum(abs(A(i,[1:i-1 i+1:(Hil_Dim)^2])));
end

%find the upper bounds of where the eigenvlaues can sit using Gershgorin.
%This maps a square in the complex plane
maxReal = max(real(a)+R);
minReal = min(real(a)+R);
maxImag = max(imag(a)+R);
minImag = min(imag(a)+R);

% define the contour as the circle that is in the middle of this square
% with a radius that is from the centre to a vertex
contourCentre = mean([maxReal,minReal])+1i*mean([minImag,maxImag]);
contourRadius = ((((maxReal - minReal)/2)^2 + ((maxImag - minImag)/2))^2 )^0.5 + margi

%check the Circle Contour surrounds spectrum
clf
hold on
plot(eigs(A),'x');
circle(mean([maxReal,minReal]),mean([minImag,maxImag]),contourRadius)
hold off
```

2. Evaluate the exact solution of the master equation in Lindblad form via the Kronecker differential equation representation using Matlab Matrix Expononential Methods 'expm' and 'expm Taylor'

To compare the results between the matrix exponential methods and complex trapezium rule approximation, we will evaluate the expectation of the Number operator at the 'end time'.

```
tic()
rho_vec_exact = expm(A*end_time)*rho_init(:);
rho_exact = reshape(rho_vec_exact,[Hil_Dim Hil_Dim]);
expm_time = toc;
N_endtime_expm = real(trace(rho_exact*N_hat));
disp(['Matrix Exponential: The expected number of particles <N>  at end_time =', num2st
```

```
Matrix Exponential: The expected number of particles <N>  at end_time =5.6988
```

```
disp(['Matrix Exponential expm: time taken =', num2str(expm_time)])
```

```
Matrix Exponential expm: time taken =0.12184
```

```
tic()
rho_vec_exact = expm_Taylor(A*end_time)*rho_init(:);
expm2_time = toc();
N_endtime_expm_taylor = real(trace(rho_exact*N_hat));
disp(['Matrix Exponential Taylor : The expected number of particles <N>  at end_time =
```

```
Matrix Exponential Taylor : The expected number of particles <N>  at end_time =5.6988
```

```
disp(['Matrix Exponential Taylor expm_Taylor: time taken =', num2str(expm2_time)])
```

```
Matrix Exponential Taylor expm_Taylor: time taken =2.8991
```

3. Complex Trapezium Approximation using a Circular Contour

```
tic()
n = 40; % define the number of points on contour

output = zeros(Hil_Dim^2,1);
```

```
inputParam = A*end_time;


parfor j = 0:n-1
    zeta = parametrize(contourCentre,contourRadius,j,n);
    w = ((2*pi)/n)*(contourRadius * 1i*exp(1i*2*pi*j/n));
    f = exp(zeta);
    output = output + (1/(2*pi*1i))*w*f*((zeta*speye(size(A))- inputParam)\rho_init(:))
end
final_density = reshape(output,[Hil_Dim Hil_Dim]);
ct_time = toc();
N_endtime_ct = real(trace(final_density*N_hat));
disp(['Complex Trapezium Approximation: The expected number of particles <N> at end_tim
```

```
Complex Trapezium Approximation: The expected number of particles <N> at end_time = 5.6924
```

```
disp(['Complex Trapezium Approximation: time taken =', num2str(ct_time)])
```

```
Complex Trapezium Approximation: time taken =0.102
```

```
error = abs(real(trace(final_density*N_hat)) - real(trace(rho_exact*N_hat)))/real(trace
disp(['error of complex trapezium= ',num2str(error)])
```

```
error of complex trapezium= 0.0011219
```

Appendix

As well as implementing the complex trapezium rule, we have used the following techniques to increase computation efficiency:

i) The system Hamiltonian and the key operators are represented by sparse matrices so we have used 'sparse' and 'speye' where possible, which has improved speed.

ii) The Cauchy Integral formula results in a step that requires a matrix inversion. However it is more efficent to evaluate the inverse on the input parameter by utilising a 'left divide' function in Matlab rather than computing the complete inverse matrix.

iii) The complex trapezium rule could be improved further by use of parallel processing, however this has not been fully evaluated in this project.

Complex Trapezium Approximation using a Circular Contour using Parfor function

```
tic()
n = 40; % define the number of points on contour

output = zeros(Hil_Dim^2,1);
inputParam = A*end_time;


parfor j = 0:n-1
    zeta = parametrize(contourCentre,contourRadius,j,n);
    w = ((2*pi)/n)*(contourRadius * 1i*exp(1i*2*pi*j/n));
    f = exp(zeta);
    wf(:,j+1) = (1/(2*pi*1i))*w*f*((zeta*speye(size(A))- inputParam)\rho_init(:));
end
output = sum(wf,2);
final_density = reshape(output,[Hil_Dim Hil_Dim]);
ct_time = toc();
N_endtime_ct = real(trace(final_density*N_hat));
disp(['Complex Trapezium Approximation: The expected number of particles <N> at end_tim
disp(['Complex Trapezium Approximation: time taken =', num2str(ct_time)])
error = abs(real(trace(final_density*N_hat)) - real(trace(rho_exact*N_hat)))/real(trace
disp(['error of complex trapezium= ',num2str(error)])
```

References

[1] EM Graefe, M Graney, A Rush - Physical Review A, 2015 - APS

[2] Higham, N. J., “The Scaling and Squaring Method for the Matrix Exponential Revisited,” *SIAM J. Matrix Anal. Appl.*, 26(4) (2005), pp. 1179–1193.

[3] Al-Mohy, A. H. and N. J. Higham, “A new scaling and squaring algorithm for the matrix exponential,” *SIAM J. Matrix Anal. Appl.*, 31(3) (2009), pp. 970–989.


```
function output = Kx_hat_ext(N,num_at,num_mol)

%coefficient of the operator that annihilates atom
%new num atoms %new num mol
if num_mol == N/2
    output(2,1) = 0;
    output(3,1) = 0;
elseif num_at < 2
    output(2,1) = 0;
```

```
    output(3,1) = 0;
else
    output(1,1) = 1/(2*(sqrt(N)))*(sqrt(num_at*(num_at-1)*(num_mol+1)));
    output(2,1) = num_at - 2;
    output(3,1) = num_mol + 1;
end


%coefficient of the operator that annihilates atom
%new num atoms %new num mol
if num_at == N
    output(2,2) = 0;
    output(3,2) = 0;
elseif num_mol ==0
    output(2,2) = 0;
    output(3,2) = 0;
else
    output(1,2) = 1/(2*(sqrt(N)))*(sqrt((num_at+1)*(num_at+2)*num_mol));
    output(2,2) = num_at + 2;
    output(3,2) = num_mol -1;
end

end


function zeta = parametrize(contourCentre,contourRadius,j,n)
    zeta = contourCentre + contourRadius*exp((2*pi*j*1i)/n);
end

function E = expm_Taylor(A)
% Matrix exponential via Taylor series.
%   E =  expm_Taylor(A) illustrates the classic definition for the
%   matrix exponential.

% Taylor series for exp(A)
E = zeros(size(A));
F = eye(size(A));
k = 1;
while norm(E+F-E,1) > 0
   E = E + F;
   F = A*F/k;
   k = k+1;
end
end


function rho_dot_vec = Master_Lindblad_Kron(rho_vec,H,Id,L,L_dag)


    %master equation for atom losses
    rho_dot_vec =  ((-(1i*(kron(Id,H) - kron(H.',Id)))) + kron(L_dag.',L) -0.5*(kron(Id

end
```

```
function circle(x,y,r)
%x and y are the coordinates of the center of the circle
%r is the radius of the circle
%0.01 is the angle step, bigger values will draw the circle faster but
%you might notice imperfections (not very smooth)
ang=0:0.01:2*pi;
xp=r*cos(ang);
yp=r*sin(ang);
plot(x+xp,y+yp);
end
```