Computational Linear Algebra
Project 2

Tudor Trita Trita
CID 01199397

December 3, 2019

This is my own work unless stated otherwise.

1 Conditioning and Stability

1.1 Stability of QR Factorisation (Householder)

The implementation and tests for this question is inside of the *question1.py* under the *householder_stability* and *main* function.

After averaging over 5000 iterations, the errors were as follows:

$$\|Q_2 - Q\| \approx 10^{-7} \tag{1}$$
$$\|R_2 - R\| \approx 10^{-7} \tag{2}$$
$$\|Q_2 R_2 - A\| \approx 10^{-15} \tag{3}$$

In lectures, we proved that the QR Householder algorithm was backwards stable. We also saw that using forward stability analysis would not necessarily give the smallest bound on the error on the numerical stability. This is illustrated by the errors above. The errors in computation of (1) and (2) are relatively large, compared to machine-epsilon ($2^{-52} \approx 2.22 \cdot 10^{-16}$ on my machine). If one were to conduct forward analysis of errors, adding the errors up, one would get a bound on the error much greater than machine epsilon. However, this approach neglects the fact that errors are 'correlated' and cancel each other out as the algorithm progresses. Backwards stability analysis provides a tool to give a much smaller bound on the error.

We see that the error in (3) is many times smaller than (1) and (2) and close to machine-epsilon, showing evidence that this algorithm is backwards-stable.

1.2 Stability of Singular Value Decomposition

The implementation and tests for this question is inside the file *question1.py* under the *svd_stability* and *main* functions.

To repeat the exercise in Question 1.1 for SVD, we will need to generate two random, independent orthonormal matrices U & V, as well as a random diagonal matrix Σ independent of U or V. We can obtain the exact (up to very small negligible rounding errors) SVD decomposition of a matrix by multiplying these three matrices together $A = U\Sigma V^T$. We then compute the numerical SVD using the function `numpy.linalg.svd`, returning the desired U_2, V_2, Σ_2 matrices, allowing us to make the comparisons as above. To generate U & V, we can use the QR factorisation function `numpy.linalg.qr` twice, throwing the R matrices away. Σ can be generated by taking just a random vector and inserting it into the diagonal of a matrix of zeros using `numpy.diag`. It is worth noting that to compute $U\Sigma$ one can exploit Numpy's matrix-vector * multiplication, where each row of the matrix is multiplied element-wise by the vector, having the same effect as performing the entire matrix-matrix multiplication, but we avoid unnecessarily computing zeroes. So we don't need to compute the matrix Σ explicitly, and the code never does.

After averaging over 5000 iterations, the errors where found as follows:

$$\|U_2 - U\| \approx 6 \tag{4}$$
$$\|V_2 - V\| \approx 6 \tag{5}$$
$$\|\Sigma_2 - \Sigma\| \approx 10^{-15} \tag{6}$$
$$\left\|U_2\Sigma_2 V_2^T - A\right\| \approx 10^{-15} \tag{7}$$

We can see that the errors in computing U, V are very large compared to machine-epsilon, however the error in the SVD result (7) is very small, and close to machine-epsilon, suggesting that this algorithm is backwards stable.

However, another explanation could be the case, and this is that the matrices $U\&V$ are not unique. This result is presented in Theorem 4.1 in "Numerical Linear Algebra" - Trefethen, Bau. The result says that the the matrices $U\&V$ in the SVD algorithm are unique only up to complex signs (i.e. a complex scalar with norm equal to 1).

This can mean that the output SVD computation that numpy implements can be different to the inputs that we have given them. However, the Σ matrix is unique, and this is reflected by the small value in the norm of sigmas. This is further evidence that this algorithm is backwards stable.

2 LU Factorisation of Banded Matrices

2.1 Implementation of LU Factorisation for Banded Matrices

The implementation of the LU factorisation of Banded Matrices (without pivoting) can be found in the file *question2.py* under the function *bandedLU*.

We have implemented the algorithm as presented in Lecture 14 'LU for Banded Matrices', but modified to work with sparse matrices.

2.2 Tests for LU Factorisation

The following tests are implemented inside of the file *question2.py*:

1. **Errors of entries of LU:** In this test we check the accuracy of the implemented function bandedLU by calculating the result $M - LU$. The test will output the largest single error attained as well as the sum of the absolute errors attained.
2. **Checking Norms:** We check that the norm $\|M - LU\|$ is within a margin of error.
3. **Check that U is Upper-Triangular:** By definition, the resulting U matrix will be upper triangular with upper band equal to mu. We do not need to check entries above the upper band, as these have not been touched in the algorithm. We only need to check that all the entries below the diagonal are within a threshold to zero, i.e. entries that have been changed from M.
4. **Checking Determinants:** We check the fact the following determinants and compute their each element $\det(M) = \det(LU) = \det(L)\det(U) = \det(U)$ - the test passes if this equation gets satisfied within a margin of error.

We do these tests for a variety of matrices, both complex and real, of different bands, same bands, with full entries, etc.

We do not need to check whether L is lower-triangular or whether the diagonal entries of L are equal to 1, because our implementation algorithm never accesses entries of L outside of the lower sub-diagonals 1 to $\max(1, ml)$, where ml is the number of lower bands that the matrix possesses.

3 Exponential Integrators

3.1 Obtaining K

Using the fact that $v = (u_1, \ldots, u_N)^T$ and after simple rearrangement of equations (2) and (3), we obtain the following:

$$\frac{1}{\Delta x^2}\begin{pmatrix} u_0 - 2u_1 + u_2 \\ u_1 - 2u_2 + u_3 \\ \vdots \\ u_{N-1} - 2u_N + u_{N+1} \end{pmatrix} = -Kv = -K\begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix}$$

but we can use the fact that $u_0, u_{N+1} = 0$, reducing the equation to:

$$\frac{1}{\Delta x^2}\begin{pmatrix} -2u_1 + u_2 \\ u_1 - 2u_2 + u_3 \\ \vdots \\ u_{N-1} - 2u_N \end{pmatrix} = -\frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & 0 & \ldots & \ldots & \ldots & 0 \\ -1 & 2 & -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & -1 & 2 & -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 2 & -1 & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 & 2 & -1 \\ 0 & \ldots & \ldots & \ldots & 0 & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix}$$

and so by inspection, we can deduce that K is the matrix

$$K = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & 0 & \ldots & \ldots & \ldots & 0 \\ -1 & 2 & -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & -1 & 2 & -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 2 & -1 & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 & 2 & -1 \\ 0 & \ldots & \ldots & \ldots & 0 & -1 & 2 \end{pmatrix}$$

3.2 Showing $\frac{dU}{dt} = LU$

Introducing $w = \frac{dv}{dt}$ we let

$$
\begin{aligned}
\frac{dU}{dt} &= \frac{d}{dt}(v^T, w^T)^T \\
&= \frac{d}{dt}\left(u_1, \ldots, u_N, \frac{du_1}{dt}, \ldots, \frac{du_n}{dt}\right)^T \\
&= \left(\frac{du_1}{dt}, \ldots, \frac{du_n}{dt}, \frac{d^2u_1}{dt^2}, \ldots, \frac{d^2u_N}{dt^2}\right)^T \\
&= \left(\frac{du_1}{dt}, \ldots, \frac{du_n}{dt}, \frac{u_0 - 2u_1 + u_2}{\Delta x^2}, \ldots, \frac{u_{N-1} - 2u_N + u_{N+1}}{\Delta x^2}\right)^T \\
&= \left(\frac{du_1}{dt}, \ldots, \frac{du_n}{dt}, \frac{-2u_1 + u_2}{\Delta x^2}, \ldots, \frac{u_{N-1} - 2u_N}{\Delta x^2}\right)^T \\
&= \begin{pmatrix}
0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & 1 & 0 & \ldots & \ldots & \ldots & 0 \\
\vdots & \ldots & \ldots & \ldots & \ldots & \ldots & \vdots & 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ldots & \ldots & \ldots & \ldots & \ldots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ldots & \ldots & \ldots & \ldots & \ldots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ldots & \ldots & \ldots & \ldots & \ldots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & 0 & \ldots & \ldots & \ldots & 0 & 1 \\
\frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & 0 & \ldots & \ldots & \ldots & 0 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\
\frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \ddots & \ddots & \ddots & \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots \\
0 & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \ddots & \ddots & \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots \\
\vdots & \ddots & \ddots & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & 0 & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \vdots & \ldots & \ldots & \ldots & \ldots & \vdots \\
0 & \ldots & \ldots & \ldots & 0 & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} & 0 & \ldots & \ldots & \ldots & \ldots & 0
\end{pmatrix}
\begin{pmatrix}
u_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_N \\ \frac{du_1}{dt} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \frac{du_n}{dt}
\end{pmatrix} \\
&= \begin{pmatrix} 0 & I \\ -K & 0 \end{pmatrix}
\begin{pmatrix} u_1 \\ \vdots \\ u_N \\ \frac{du_1}{dt} \\ \vdots \\ \frac{du_N}{dt} \end{pmatrix} \\
&= LU
\end{aligned}
$$

as required.

3.3 Eigenvalues of L

We obtain the eigenvalues of L by solving the characteristic polynomial of L given by

$$\det(L - \lambda_L I_{2N}) = \det \begin{pmatrix} -\lambda_L I_N & I_N \\ -K & -\lambda_L I_N \end{pmatrix} = 0 \tag{8}$$

It is a known result in linear algebra that the determinant of a 2x2 block matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

is given by $\det(AD - BC)$ if A, B, C, D all commute with each other. This is the case for equation (8), and so

$$\begin{aligned} \det(L - \lambda_L I_{2N}) &= \det(-\lambda_L I_N \cdot (-\lambda_L I_N) - I_N \cdot (-K)) && (9) \\ &= \det(\lambda_L^2 I_N + K I_N) && (10) \\ &= \det(K + \lambda_L^2 I_N) = 0 && (11) \end{aligned}$$

We now quote another known result in linear algebra about the eigenvalues of a Toeplitz Tridiagonal Matrix. For a Toeplitz Tridiagonal matrix such as K with three values, a on the main diagonal, b on the diagonal below and c on the diagonal above the main diagonal, the eigenvalues of A are given by

$$\lambda_{A_k} = a - 2\sqrt{bc}\cos\left(\frac{k\pi}{N+1}\right), \quad k = 1, 2, \ldots, N. \tag{12}$$

The eigenvalues of K are the solution to the characteristic polynomial $\det(K - \lambda_K I_N) = 0$. But by equating this equation with equation (11), this implies that the eigenvalues of L are given by

$$\begin{aligned} & \lambda_L^2 = -\lambda_k, \quad k = 1, 2, \ldots, N && (13) \\ \implies \quad & \lambda_L = \pm\sqrt{-\lambda_k} = \pm i\sqrt{\lambda_k}, \quad k = 1, 2, \ldots, N && (14) \end{aligned}$$

which are imaginary as long as $\lambda_k > 0, \forall k$. We can now work out an explicit form for the eigenvalues of L. By using equation (12), we can see that the eigenvalues of K are given by

$$\begin{aligned} & \frac{2}{\Delta x^2}\left(1 + \cos\left(\frac{k\pi}{N+1}\right)\right), \qquad k = 1, \ldots, N. && (15) \\ \implies \lambda_L = \pm i & \sqrt{\frac{2}{\Delta x^2}\left(1 + \cos\left(\frac{k\pi}{N+1}\right)\right)}, \qquad k = 1, \ldots, N. && (16) \end{aligned}$$

and these λ_k are non-negative, so the eigenvalues of L are indeed imaginary. Furthermore, we can obtain an explicit formula for the absolute value of the largest eigenvalue of L by taking $k = N$ which will make the computation in the next parts more efficient.

$$|\max(\lambda_L)| = \frac{1}{\Delta x}\sqrt{2\left(1 + \cos\left(\frac{\pi}{N+1}\right)\right)}$$

3.4 Banded Matrix System

We begin with equation (8) (as block matrices):

$$\begin{aligned} & (\alpha_j I + TL)U_j = U(0) \\ \implies & \begin{pmatrix} \alpha_j I & TI \\ -TK & \alpha_j I \end{pmatrix} \begin{pmatrix} v_j \\ w_j \end{pmatrix} = \begin{pmatrix} v_0 \\ w_0 \end{pmatrix} \\ \implies & \begin{pmatrix} \alpha_j v_j + T w_j \\ -TK \cdot v_j + \alpha_j w_j \end{pmatrix} = \begin{pmatrix} v_0 \\ w_0 \end{pmatrix} \end{aligned}$$

This produces two equations (i): $\alpha_j v_j + T w_j = v_0$, (ii): $-TK \cdot v_j + \alpha_j w_j = w_0$.

From equation (i), we obtain that

$$w_j = \frac{v_0 - \alpha_j v_j}{T}$$

and upon substitution of this into equation (ii) we obtain the following:

$$\begin{aligned} & -TK \cdot v_j + \frac{\alpha_j}{T}(v_0 - \alpha_j v_j) = w_0 \\ \implies & -T^2 K \cdot v_j + \alpha_j v_0 - \alpha_j^2 v_j = w_0 T \\ \implies & (T^2 K + \alpha_j^2 I) v_j = \alpha_j v_0 - w_0 T \end{aligned}$$

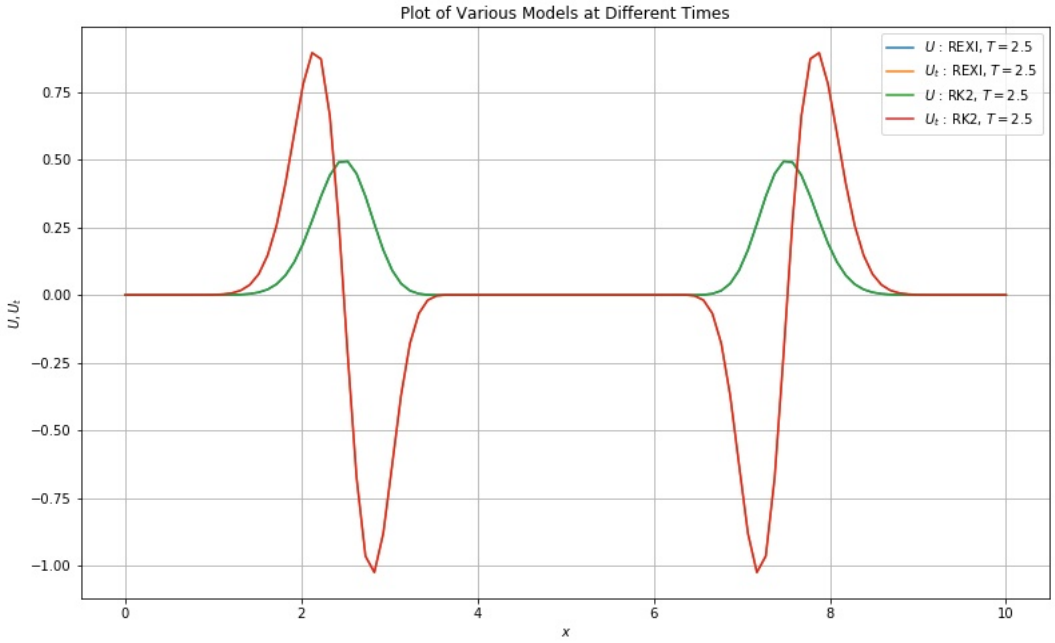
which is a banded matrix system for v_j only, with banded matrix $\mathrm{M} = (T^2 K + \alpha_j^2 I)$, having 1 lower band and 1 upper band.

3.5 Implementation of Exponential Integrator

The implementation and graphs for this question is inside the file *question3.py* inside the class method *Computations.REXI*. I used a class for this section to make the computations of L, K and their storage in memory faster and easier to access throughout the implementation.

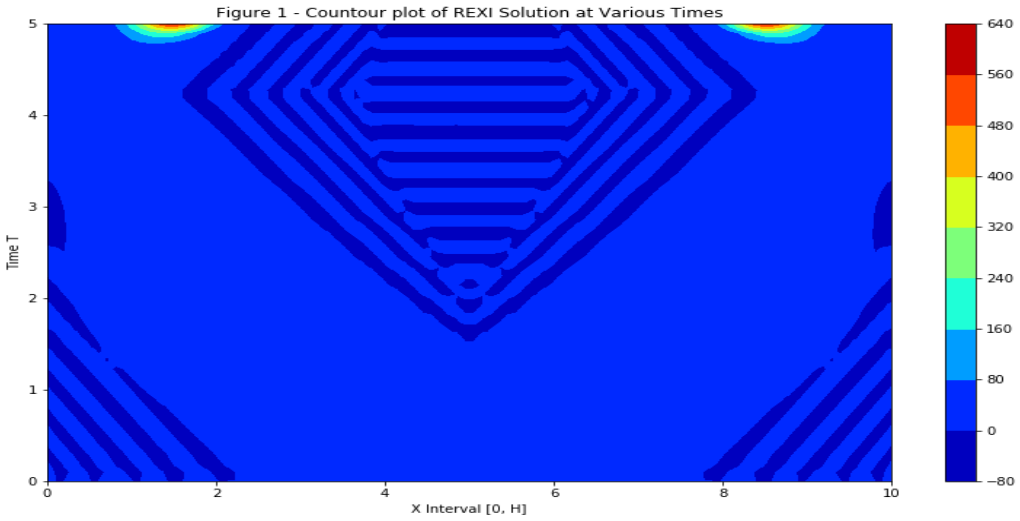
3.6 Comparison with Second-Order Runge Kutta

We begin by displaying the wave at time $T = 2.5$ for the REXI and Runge-Kutta 2nd Order methods:

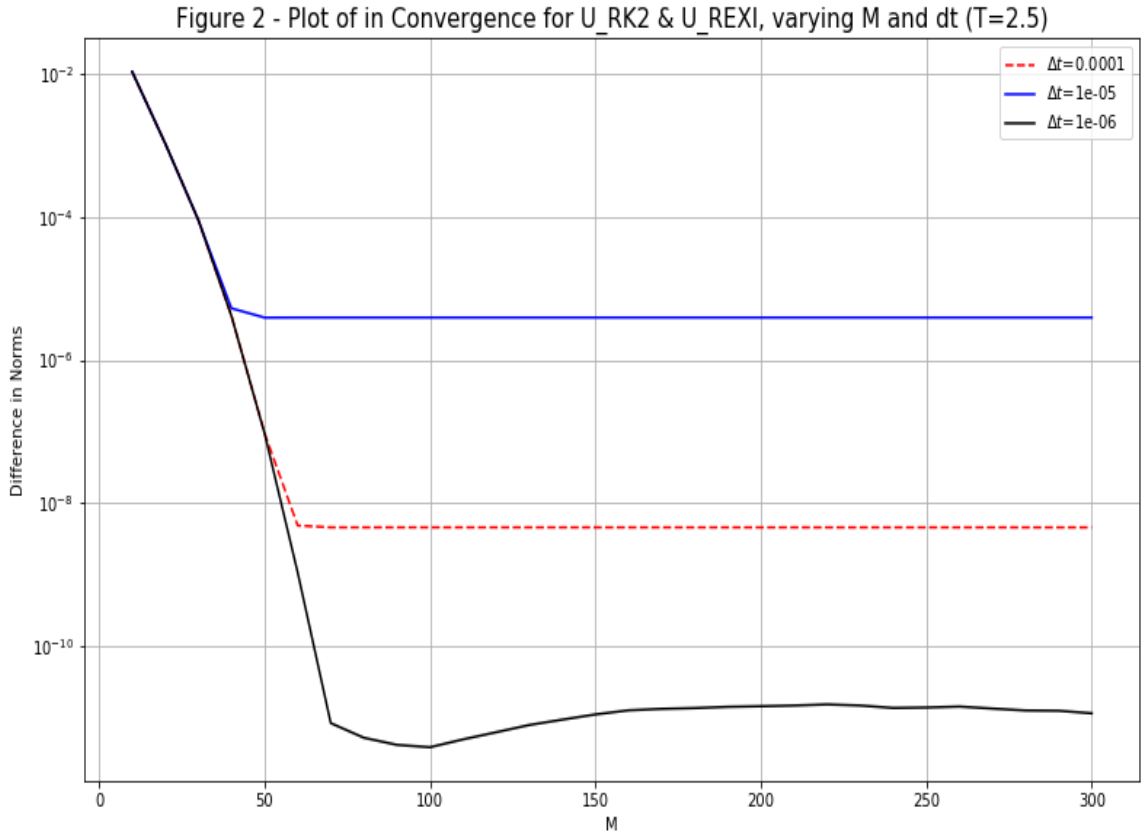


We cannot distinguish both methods, so we can agree that they give very similar results.

We now plot a contour plot of the REXI solution from times 0 to 5:

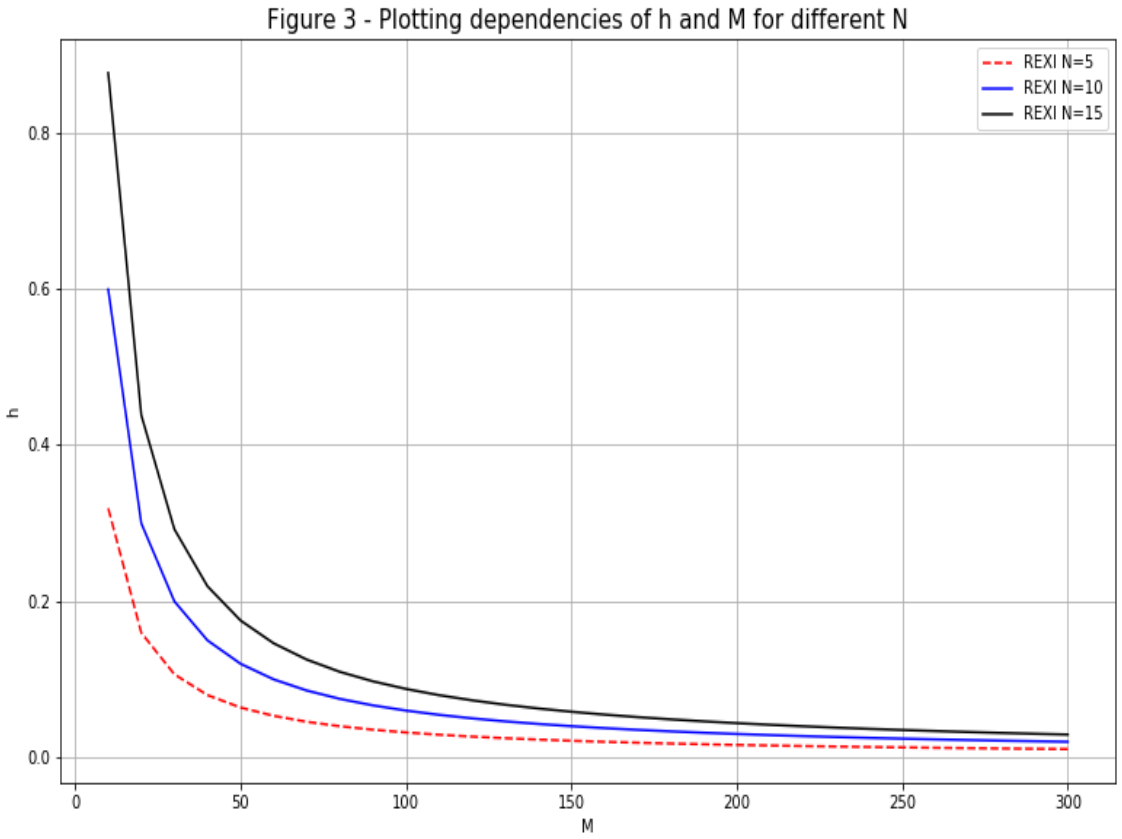


The following plot illustrates the error $\|U_{RK2} - U_{REXI}\|$:



As we can see, there is a clear dependence on M for the error. The larger the value of M is, the more accurate the REXI method becomes, up to a point. This point is due to the fact that the REXI methods converges to the correct solution faster than the RK2 does for large M, and therefore the plateau shown in the graphs are mainly due to the errors in the RK2 method. This is confirmed by the fact that by increasing Δt, we can make this plateau occur at different values.

We now explore the dependency between h and M:



The relationship between h and M is of the form $h \propto 1/M$. This can be seen clearly in the graph, but also from the equation

$$hM = 1.1T|\mu_{max}|$$