

Time-parallel simulation of the Schrödinger Equation

Hannah Rittich^{a,*}, Robert Speck^a

^a*Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425 Jülich, Germany*

Abstract

The numerical simulation of the time-dependent Schrödinger equation for quantum systems is a very active research topic. Yet, resolving the solution sufficiently in space and time is challenging and mandates the use of modern high-performance computing systems. While classical parallelization techniques in space can reduce the runtime per time step, novel parallel-in-time integrators expose parallelism in the temporal domain. They work, however, not very well for wave-type problems such as the Schrödinger equation. One notable exception is the rational approximation of exponential integrators. In this paper we derive an efficient variant of this approach suitable for the complex-valued Schrödinger equation. Using the Faber-Carathéodory-Fejér approximation, this variant is already a fast serial and in particular an efficient time-parallel integrator. It can be used to augment classical parallelization in space and we show the efficiency and effectiveness of our method along the lines of two challenging, realistic examples.

Keywords: Schrödinger equation, parallel-in-time, rational approximation of exponential integrators, parallel across the method, Faber-Carathéodory-Fejér approximation

1. Introduction

The time-dependent, non-relativistic Schrödinger equation [1] is a complex-valued linear partial differential equation (PDE) that describes the time-evolution of a quantum system. Being able to predict the behavior of a quantum system is important for many applications. Without an analytical, tractable solution, however, numerical methods are needed to evaluate the solution of the PDE. Interest in simulating the time-dependent Schrödinger equation started in the end of the 1950s [2]. With the availability of sufficiently powerful computers, these simulations became increasingly popular for the investigation of molecular structures around 1970 [3, 4, 5] and are still relevant today [6, 7, 8].

*Corresponding author

Email addresses: h.rittich@fz-juelich.de (Hannah Rittich), r.speck@fz-juelich.de (Robert Speck)

For this work we restrict ourselves to the single-particle Schrödinger equation in d dimensions,

$$i\hbar \frac{\partial}{\partial t} \psi(\mathbf{r}, t) = \left(-\frac{\hbar^2}{2m} \Delta + V(\mathbf{r}) \right) \psi(\mathbf{r}, t), \quad (1)$$

where \hbar is the reduced Planck constant, m the mass of the particle, Δ the Laplace operator, $V : \mathbb{R}^d \rightarrow \mathbb{R}$ the potential, and $\psi : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{C}$ is the unknown wave function. Given an initial wave function ψ_0 at some time t , the Schrödinger equation can be used to compute the wave function at any later time. The function ψ encodes the probability distribution of the position and momentum of the particle. More precisely, the probability density of the position of the particle at time t is given by $|\psi(\mathbf{r}, t)|^2$, while the momentum of the particle is, loosely speaking, encoded in the wave-length of ψ via the de Broglie relation $\lambda = h/p$. For more details see, e.g., [9, 10, 11].

The Schrödinger equation is defined on an unbounded domain, which causes problems for numerical computations. Hence, we introduce a finite, but sufficiently large, domain $\Omega \subseteq \mathbb{R}^d$. We then demand that ψ fulfills the Schrödinger equation for $\mathbf{r} \in \Omega$ and conforms to zero Dirichlet boundary conditions, i.e., we require that

$$\psi(\mathbf{r}, t) = 0 \quad \text{for } \mathbf{r} \in \partial\Omega.$$

These boundary conditions imply that the particle leaves the domain Ω with zero probability. If Ω is large enough, this is a reasonable assumption and does not change the outcome of the simulation.

In order to perform such a simulation, the continuous Schrödinger equation has to be discretized both in space and time. Depending on the dimension, the smoothness of the solution and the dynamics of the system, the resulting numerical method may require fine and advanced discretization schemes to resolve the solution adequately and over long time-scales. This mandates the application of parallel numerical algorithms on high-performance computing systems.

Classical parallelization techniques primarily target the spatial domain and are very successful in reducing the time-to-solution per time step. However, this approach can neither mitigate the need for a better resolution in time nor can it scale indefinitely for a fixed-size problem. One promising remedy is the application of parallel-in-time integration techniques. They expose parallelism also in the temporal domain, either within each time step, referred to as *parallelization across the method*, or by computing multiple time steps simultaneously, referred to as *parallelization across the steps* [12].

Parallel-in-time methods have been applied to a multitude of problems, ranging from reaction-diffusion systems [13] and a kinematic dynamo [14] to eddy current problems [15], fusion plasma simulations [16] as well as power systems [17] and robotics [18], to name just a few very recent ones. For further reading, we refer to the comprehensive list of references that is provided on the community website on parallel-in-time integration¹.

Yet, many of these approaches fail for wave-type problems, which includes the non-relativistic Schrödinger equation, we are interested in. For this class of problems, only very specialized and often purely theoretical ideas exist. One promising one, which

¹<https://parallel-in-time.org>

has indeed shown its practical relevance, is the *rational approximation of exponential integrators* (REXI). While it has been well known that certain forms of rational approximations can be used to compute the matrix exponential in parallel [19, 20], it has been first applied for the construction of a parallel-in-time solver for wave-type problems in [21]. This method targets linear problems and forms a parallelizable approximation of the exponential matrix function using rational functions, which can then be used to approximate the solution of the linear PDE. The approximation is designed in a way such that its evaluation consists mainly of the computation of a sum, where the computation of each summand is expensive. The benefit of this structure is that each individual summand in the approximation can be computed in parallel. It can thus be classified as parallel across the method, although its approach allows to take much larger time steps as more classical methods like Crank-Nicolson. REXI has been successfully applied to shallow-water equation on the rotating sphere [22] and to linear oscillatory problems [21, 23], making parallel-in-time integration possible even for these challenging problems.

The rational approximation chosen in the original REXI approach presented in [21], however, involves taking the real part of a complex quantity. While the method can still be applied for complex-valued problems such as the Schrödinger equation, it becomes significantly more expensive.

In this paper, we therefore present a variant of the REXI method specifically targeted toward complex-valued problems. We use a variation of the Faber-Carathéodory-Fejér (Faber-CF) approximation together with a conformal Riemann mapping, which is tailored for the purely imaginary eigenvalues of the semi-discretized Schrödinger equation. This approach reduces the cost of the REXI method substantially, i.e., fewer summands are needed for the rational approximation, thereby increasing the ratio of accuracy per parallel task. For a given accuracy, this method imposes a restriction on the time-step size, which is also discussed in this paper. We note that this restriction is inherent to the REXI approach itself and needs to be considered for the original version as well.

We begin by briefly explaining the finite element discretization in space (Section 2) which leads to a system of ordinary differential equations (ODEs) that needs to be solved. Then, we discuss how to solve this system by approximating a certain matrix exponential (Section 3) and how this computation can be performed in parallel (Section 3.1). For this approximation we need to find a suitable rational approximation to the exponential function, which we construct by using the Faber-Carathéodory-Fejér method (Section 3.2). Finally, we apply the method to two challenging, realistic problems, namely the quantum tunneling and the double-slit experiment, analyzing the performance of the method (Section 4) and finally discuss the applicability of the numerical method (Section 5) beyond the Schrödinger equation.

2. Space Discretization

To simulate the Schrödinger equation, we need an appropriate discretization of the equation. We start by applying the method of lines approach to turn the PDE into a system of ODEs, by applying the finite element method [24, 25, 26] to the spatial part of the PDE.

The finite element discretization is based on the weak formulation of the PDE. For a domain Ω , let $H^1(\Omega) \subset \mathbb{C}^\Omega$ be the Sobolev space of order one and let $\dot{H}^1(\Omega)$ be the subset of $H^1(\Omega)$ that consists of functions whose trace vanishes, i.e., $\dot{H}^1(\Omega) := \{u \in H^1(\Omega) : u|_{\partial\Omega} = 0\}$, where $u|_{\partial\Omega}$ is the trace of u [25]. The weak formulation of the Schrödinger equation (1) is to find ψ such that

$$\psi(t, \cdot) \in \dot{H}^1(\Omega) \quad \text{and} \quad \text{i}b\left(\phi, \frac{\partial\psi}{\partial t}(t)\right) = a(\phi, \psi(t)) \quad \text{for all } \phi \in \dot{H}^1(\Omega),$$

where a and b are the bilinear forms given by

$$a(\phi, \psi) := \int \left(\frac{\hbar^2}{2m} \nabla \phi \cdot \nabla \psi + V \phi \psi \right) \mathrm{d}\mathbf{r} \quad \text{and} \quad b(\phi, \frac{\partial\psi}{\partial t}) := \hbar \int \phi \frac{\partial\psi}{\partial t} \mathrm{d}\mathbf{r}$$

[27, 28].

We can turn the weak formulation into a discrete problem using the famous Ritz approach [29], which approximates the solution of the weak form of the PDE. We select a suitable subspace $V_h \subseteq \dot{H}^1$ and replace \dot{H}^1 by V_h in the weak formulation. In our case, we choose $V_h = \{u \in \dot{H}^1(\Omega) : u|_T \in \mathcal{P}_2 \text{ for all } T \in \mathcal{T}\}$, where \mathcal{T} is a triangulation of the domain Ω . More precisely, we use Lagrange finite elements of order 2 on each triangle [26, 25]. Introducing basis vectors χ_1, \dots, χ_n for V_h , we can write the modified weak formulation as

$$\text{i}B \frac{\partial u}{\partial t}(t) = Au(t), \tag{2}$$

where $A, B \in \mathbb{R}^{n \times n}$ with $A_{jk} = a(\chi_j, \chi_k)$, and $B_{jk} = b(\chi_j, \chi_k)$. The vector $u \in \mathbb{C}^n$ contains the basis coefficients of the approximation of the solution. Together with suitable initial conditions $u(0) = u_0$, this system of ODEs defines the initial value problem (IVP) that we intend to solve using an efficient, parallel-in-time integrator.

3. Time Discretization

Some of the most efficient time integration methods for the Schrödinger equation are based on the approximation of the matrix exponential [30]. Classical time integration schemes require the step size of the method to be a fraction of the shortest wavelength that is present in the problem. Methods based on the computation of the matrix exponential usually do not have this restriction and can in principle use much larger step sizes.

The matrix exponential can be used to compute the solution of the IVP. Since the matrices A and B do not depend on the time variable t , the solution $u(\tau)$ for $\tau > 0$ of the IVP (2) is given by

$$u(\tau) = \exp(\tau M) u_0 \quad \text{with} \quad M := (\text{i}B)^{-1}A, \tag{3}$$

where \exp is the matrix exponential [31, 32]. Thus, one way to solve the IVP is to compute an approximation to the product of the vector u_0 and the matrix exponential of M .

3.1. Rational Approximation of Exponential Integrators

We want to use rational approximations to compute the matrix exponential numerically. There are various ways to compute the exponential of a matrix [33, 34], however, we are interested in methods that use rational approximations, because these methods can be constructed in a way that allows for parallelizing the time integration scheme itself, increasing the parallelism of the overall solution process [21, 23, 22].

It can be shown that the matrix $\tau M = \tau(iB)^{-1}A$ is diagonalizable with purely imaginary eigenvalues. Hence, for simplicity, we restrict ourselves in the following to the computation of exponentials of matrices that have these properties. The method, however, can be applied in cases where these two assumptions do not hold.

The matrix exponential is a special case of a matrix function, which is a way to extend a scalar function $f : \mathbb{C} \rightarrow \mathbb{C}$ to the set of matrices, i.e., to a function $f : \mathbb{C}^{n \times n} \rightarrow \mathbb{C}^{n \times n}$ [35]. If G is diagonalizable, i.e., $G = X\Omega X^{-1}$, $\Omega = \text{diag}(\omega_1, \dots, \omega_n)$, the matrix function of f is given by $f(G) = X \text{diag}(f(\omega_1), \dots, f(\omega_n)) X^{-1}$. Diagonalizing a large matrix, however, is computationally expensive, and thus this formula is usually not useful for computing the function of a matrix.

We can reduce the computational costs by replacing the direct computation of the matrix function by a suitable approximation. By using the diagonalization of G , we see that if \tilde{f} is a function which approximates f in the eigenvalues of G then the matrix function $\tilde{f}(G)$ is close to $f(G)$. Thus, if $\tilde{f}(G)$ is cheap to compute, we have a practical way for evaluating the matrix function $f(G)$ numerically.

Matrix functions of rational functions can be computed without the need of explicitly computing the diagonalization of the matrix. Assume that $r = p/q$, and p, q are polynomials such that r approximates f in the eigenvalues of G . Computing $r(G)$ to approximate $f(G)$ is a feasible approach for the numerical evaluation of the matrix exponential on its own, however, by making an additional assumption, we can derive a time-stepping scheme that intrinsically allows for the simultaneous execution of certain parts of the computation.

Assume that $\deg p < \deg q$ and that the roots of p are distinct. In this case, we can use the partial fraction decomposition to obtain that

$$r(z) = \sum_{j=1}^K \frac{\beta_j}{z - \sigma_j} \quad (4)$$

for proper shifts $\sigma_j \in \mathbb{C}$ and coefficients $\beta_j \in \mathbb{C}$, $j = 1, \dots, K$, $K \in \mathbb{N}$, and the corresponding matrix function

$$r(G) = \sum_{j=1}^K \beta_j (G - \sigma_j I)^{-1},$$

which approximates $f(G)$.

Using the rational approximation of the matrix function $f(G)$, we can define the REXI time stepping scheme. Let $f = \exp$, $G = \tau M$, and r a rational approximation as discussed above, i.e., $r(\tau M) \approx \exp(\tau M)$. Then, the exponential formula for the solution

of the ODE (3) implies that $u(\tau) \approx r(\tau M)u_0$. Hence, we define one time step of the REXI method by $u_1 = r(\tau M)u_0$, which can be computed by

$$\begin{aligned} u_1 &= \sum_{j=1}^K \beta_j (\tau(\mathrm{i}B)^{-1}A - \sigma_j I)^{-1} u_0 = \sum_{j=1}^K \beta_j ((\mathrm{i}B)^{-1}(\tau A - \sigma_j \mathrm{i}B))^{-1} u_0 \\ &= \sum_{j=1}^K \beta_j (\tau A - \sigma_j \mathrm{i}B)^{-1} (\mathrm{i}B) u_0, \end{aligned} \quad (5)$$

where we used the definition of M .

The benefit of computing matrix exponential times vector by evaluating the rational approximation via (5) is that the evaluation of this approximation can be readily parallelized, because each summand can be evaluated independently. Thus, each of the K linear systems can be solved independently, using K different parallel tasks. We refer to this particular splitting of the computation into tasks as *time-parallelization*, because it uses only properties that are inherent to the time-stepping scheme itself. Note that each of these K temporal tasks can be parallelized themselves, since they involve a set of vector and matrix routines, which can be executed in parallel as well. We call this second splitting *space-parallelization*, because the vectors and matrices describe the spatial dimension of the problem. Using time-parallelization into P_t tasks and then applying space-parallelization into P_s sub-tasks to each of the temporal tasks yields $P_t \cdot P_s$ sub-tasks that can be executed simultaneously.

To be able to implement and apply this method, it remains to derive proper shifts σ_j and coefficients β_j (for $j = 1, \dots, K$). Note that, in general, we only need to compute these shifts and coefficients once, because they do not depend on the initial values or the time step. In the following, we will describe the derivation of these parameters in detail using the Faber-Carathéodory-Fejér (Faber-CF) approximation, introduced in [36]. The intention here is to allow interested readers to comprehend and reproduce the steps necessary to obtain the σ_j and β_j and thus the full algorithm.

We point out that the use of the Faber-CF approximation is a key difference to the REXI approach in [21] and [23]. There, an approximation of the form

$$e^{\mathrm{i}x} \approx \operatorname{Re} \left(\sum_{j=1}^K \frac{\gamma_j}{\mathrm{i}x + \mu_j} \right)$$

for certain $\gamma_j, \mu_j \in \mathbb{C}$ is used. If all eigenvalues of M are purely imaginary, as in our case, and all eigenvectors can be chosen to be real, then

$$e^{\tau M} \approx \operatorname{Re} \left(\sum_{j=1}^K \gamma_j (\tau M + \mu_j I)^{-1} \right),$$

where Re denotes the element-wise real-part of the matrix.

The problem when applying this approximation to the Schrödinger equation is that we want to compute $e^{\tau M} u_0$ where u_0 has complex entries without explicitly computing

the matrix $e^{\tau M}$. If u_0 would be real, then u_0 could just be moved inside the computation of the real part. Since u_0 is complex, however, we need to compute

$$e^{\tau M} u_0 \approx \operatorname{Re} \left(\sum_{j=1}^K \gamma_j (\tau M + \mu_j I)^{-1} \operatorname{Re}(u_0) \right) + i \operatorname{Re} \left(\sum_{j=1}^K \gamma_j (\tau M + \mu_j I)^{-1} \operatorname{Im}(u_0) \right),$$

which is twice as much work as in the real case. The Faber-CF approximation that we use does not have this drawback.

Furthermore, the shifts used in the method derived in [21, 23] form conjugate pairs. Using properties of the real numbers, the method only needs to solve one linear system for each conjugate pair. Since the matrix M of the discretization of the Schrödinger equation has complex entries and the right-hand side of the linear systems are complex valued, such a simplification is not possible in the setting we consider in this paper.

There exists another difference between the two approaches. The Faber-CF approximation computes the approximation essentially in one step, while the method in [21] involves a two step approximation. First, a rational approximation to a Gaussian function is constructed. Then, this approximation is used, to approximate the function e^{ix} . This procedure has the benefit that it is easy to compute approximations that are accurate over large intervals and thus allow for the large time steps (see Section 3.3). In our experience, however, using the same accuracy and same approximation interval, the Faber-CF approximation requires fewer poles and therefore fewer linear systems to solve, as detailed in Section 3.3 below.

3.2. Faber-Carathéodory-Fejér Approximation

The Faber-Carathéodory-Fejér approximation is based on the Carathéodory-Fejér (CF) approximation introduced in [37]. The latter computes an approximation to holomorphic functions on the unit disc. The former uses the Faber transform, to generalize the Carathéodory-Fejér approximation to almost arbitrary approximation domains.

3.2.1. The Carathéodory-Fejér Approximation

The CF approximation is a rational approximation to a holomorphic function on the unit disc $D := \{z \in \mathbb{C} : |z| \leq 1\}$. The resulting rational approximations are only close to the best approximation, but easier to obtain.

Let us start by introducing the following notation. Let R_{mn} be the set of rational functions $r(z) = \frac{p(z)}{q(z)}$ with $\deg(p) \leq m$ and $\deg(q) \leq n$ that are holomorphic in D . Furthermore, let $S := \{z : |z| = 1\}$ denote the unit circle. We define the uniform norm of a complex valued function u on the unit disc by $\|u\|_D := \sup\{|u(z)| : z \in D\}$ and the uniform norm of a complex valued function u on the unit circle by $\|u\|_S := \sup\{|u(z)| : z \in S\}$.

With this notation at hand, our next step is to simplify the problem. Assume that $r \in R_{mn}$ is an approximation to a function f which is holomorphic on D . In this case, the error $e(z) := f(z) - r(z)$ is also holomorphic on D . We want that the size of the error to be as small as possible, i.e., we want $\|e\|_D$ to be small. Since e is holomorphic on D , its maximum is located on the boundary of D . Thus, to minimize $\|e\|_D$ we just

have to minimize $\|e\|_S$, i.e., we just have to find a rational function that approximates f well on the sphere S .

It is difficult to find the best approximation to f in R_{mn} . The key idea of the CF approximation is to find the best approximation to f in a larger space \tilde{R}_{mn} with respect to the $\|\cdot\|_S$ norm and then approximate the best approximation from \tilde{R}_{mn} with a function from R_{mn} . The space \tilde{R}_{mn} is defined as follows. Let G be the set of functions that are analytic and bounded in $1 < |z| \leq \infty$ and zero at $z = \infty$. Then define $\tilde{R}_{nn} := R_{nn} + G$ and $\tilde{R}_{mn} := z^{m-n} \tilde{R}_{nn}$. One can show that the space \tilde{R}_{mn} consists of the functions

$$\tilde{r}(z) = \frac{\sum_{k=-\infty}^m d_k z^k}{\sum_{k=0}^n e_k z^k}, \quad (6)$$

where the poles of the numerator lie inside the unit disc and the roots of the denominator lie outside the unit disc.

Once we have obtained the best approximation $\tilde{r}^* \in \tilde{R}_{mn}$ in the form of (6) we can use it to find an approximation in R_{mn} that is close to \tilde{r}^* —the CF approximation. Consider the asymptotic analysis of approximating a function $z \mapsto f(\epsilon z)$ for $\epsilon \rightarrow 0$, $\epsilon > 0$ where f is smooth. In [37] it was shown that for small enough ϵ , the best approximation \tilde{r}^* gets arbitrarily close to a rational function. This behavior motivates the construction of the CF approximation r^{cf} : we compute \tilde{r}^* in the form of (6) and discard the summands with negative indices from the numerator, i.e.,

$$r^{\text{cf}}(z) = \frac{\sum_{k=0}^m d_k z^k}{\sum_{k=0}^n e_k z^k}.$$

We thus need to find the best approximation \tilde{r}^* of f in \tilde{R}_{mn} . First of all note that f can be written in Maclaurin series form, because f is holomorphic. In case the Maclaurin series is not known, it can be computed via the fast Fourier transform (FFT). Since the Maclaurin series converges, we can find an $L \in \mathbb{N}$ such that the polynomial h of degree L that we get by truncating the series after $L + 1$ terms approximates f with negligible error. Thus, the problem simplifies to finding the best approximation \tilde{r}^* to a polynomial h . The theorem below enables us to compute the best approximation in \tilde{R}_{mn} of a polynomial.

Theorem 1 (Trefethen). *The polynomial $h(z) = a_0 + \dots + a_L z^L$ has a unique best-approximation \tilde{r}^* out of \tilde{R}_{mn} . Let*

$$H_{mn} := \begin{pmatrix} a_{m-n+1} & a_{m-n+2} & a_{m-n+3} & \cdots & a_L \\ a_{m-n+2} & a_{m-n+3} & \cdots & a_L & \\ a_{m-n+3} & \cdots & a_L & & \\ \vdots & \ddots & & & \\ a_L & & & & 0 \end{pmatrix} \in \mathbb{C}^{(K+n-m) \times (K+n-m)},$$

where we define $a_k = 0$ for $k < 0$. The error of the approximation \tilde{r}^* is

$$\|h - \tilde{r}^*\|_S = \sigma_{n+1}(H_{mn}),$$

where $\sigma_{n+1}(H_{mn})$ is the $(n+1)$ -st singular value of the matrix H_{mn} . Furthermore, \tilde{r}^* is given by

$$h(z) - \tilde{r}^*(z) = \sigma_{n+1} z^L \frac{u_1 + \dots + u_{K+n-m} z^{L+n-m-1}}{v_{L+n-m} + \dots + v_1 z^{L+n-m-1}} \quad (7)$$

where $u = (u_1, \dots, u_{L+n-m})^T$ and $v = (v_1, \dots, v_{L+n-m})^T$ are the $(n+1)$ st columns of U and V , respectively, in the singular value decomposition $H_{mn} = U\Sigma V^H$.

Proof. See [37, Theorem 3.2]. \square

This theorem provides us with a formula for the error of the approximation and we can now work backwards from the error to obtain the approximation $\tilde{r}^* \in \tilde{R}_{mn}$ via (7). From \tilde{r}^* we then obtain the CF-approximation $r^{\text{cf}} \in R_{mn}$ by dropping the terms with negative indices from the numerator. Since we want to be able to write the rational approximation in partial fraction decomposition form (4), we restrict ourselves to the case $n = m + 1$ in the following. The whole procedure for this case is listed in Algorithm 1.

Algorithm 1 Computation of the Carathéodory-Fejér Approximation

```

1: function CF( $(a_i)_{i=0}^L, n$ )
2:    $h(z) := \sum_{j=0}^L a_j z^j$  ▷ the polynomial to approximate
3:    $H := \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_L \\ a_1 & a_2 & & & \\ a_2 & & \ddots & & \\ \vdots & & & \ddots & \\ a_L & & & & 0 \end{pmatrix}$ 
4:    $U, \Sigma, V := \text{SVD}(H)$  ▷  $U\Sigma V^H$  is the singular value decomposition of  $H$ 
5:    $\sigma_{n+1} := \Sigma_{n+1, n+1}$  ▷ the  $(n+1)$ -st singular value of  $H$ 
6:    $u := U_{*, n+1}$  ▷ the  $(n+1)$ -st column of  $U$ 
7:    $v := V_{*, n+1}$  ▷ the  $(n+1)$ -st column of  $V$ 
8:    $p(z) := u_1 + u_2 z + \dots + u_{K+1} z^L$ 
9:    $q(z) := v_{K+1} + v_K z + \dots + v_1 z^L$ 
10:   $\tilde{r}^*(z) := h(z) - \sigma_{n+1} z^L \frac{p(z)}{q(z)}$ 
11:   $z_1, \dots, z_K :=$  the roots of  $q$  whose absolute modulus is larger than one
12:   $q_{\text{out}}(z) := (z - z_1) \cdot (z - z_2) \cdot \dots \cdot (z - z_K)$  ▷ the denominator of the CF-approximation
13:   $(d_j)_{j=-\infty}^\infty := \text{LAURENTSEQUENCE}(q_{\text{out}} \tilde{r}^*)$  ▷  $q_{\text{out}}^* \tilde{r} = \sum_{j=-\infty}^\infty d_j z^j$ 
14:   $r^{\text{cf}}(z) := \frac{d_0 + d_1 z + \dots + d_{n-1} z^{n-1}}{q_{\text{out}}(z)}$  ▷ drop the coefficients with negative indices
15:  return ( $r^{\text{cf}}, (z_1, \dots, z_K)$ )
16: end function

```

The algorithm starts by computing the quantities used in Theorem 1 (ll. 3–9). Rearranging the error equation (7) yields \tilde{r}^* , the best approximation in \tilde{R}_{mn} (l. 10). Unfortunately, in this form \tilde{r}^* does not provide us with the coefficients of the numerator and denominator. Since h is a polynomial, it is easy to see that the poles of \tilde{r}^* are the roots of the polynomial q . We want to write \tilde{r}^* in the form of (6). By definition, all roots of the denominator lie outside of the unit disc, while the poles inside the unit disc are part of the numerator. Thus, we obtain the denominator of \tilde{r}^* by multiplying all linear factors of q corresponding to roots outside the unit disc (ll. 11–12). Finally, we obtain the coefficients of the numerator by computing the Laurent series of $q_{\text{out}}\tilde{r}^*$ and then dropping the terms with negative indices.

Note that $K \leq n$ and in general we expect K to be equal to n . In the case where a root of the denominator q lies on the unit circle, K can be less than n . In this case, the root in the denominator is canceled by a root in the numerator p . For all practical purposes, however, we can assume that by choosing n we can choose the degree K of the rational approximation [37].

3.2.2. Using the Faber Transform

In many practical applications it is desirable to compute approximations to functions that are defined on domains other than the unit disc. In our case we are interested in computing an approximation that is accurate at the eigenvalues of the matrix M (3). Since the eigenvalues of M are purely imaginary we can restrict the approximation domain to an interval on the imaginary axis. While it would be simple to compute approximations on a disc with a radius large enough to include the desired interval, being able to choose the approximation domain more precisely, and hence smaller, often leads to a better approximation accuracy.

Using the Faber transform, the CF approximation can be extended to allow for the approximation of functions defined on more general domains. Key to this modification is the observation in [36] that the Faber transform maps a rational function onto a rational function. We shall discuss the method introduced in [36], which we modify to compute the rational approximation in partial fraction decomposition form (4).

The Faber transform is based on the fact that Faber polynomials can be used to derive a series expansion of analytic functions. More precisely, let $E \subset \mathbb{C}$ be a compact set such that the complement E^c of E is simply connected in the extended complex plane. Then, an argument involving the Riemann mapping theorem [38] shows that there exists a conformal map η that maps the complement D^c of the closed unit disc conformally onto E^c such that $\eta(\infty) = \infty$ and $\lim_{z \rightarrow \infty} \eta(z)/z = c$. Using this function η we can construct a family of polynomials p_j (for $j = 1, 2, \dots$) with $p_0(w) = 1$ and $\deg(p_j) = j$, such that every analytic function f on E can be written as

$$f(w) = \sum_{j=0}^{\infty} a_j p_j(w) \quad \text{where} \quad a_n = \frac{1}{2\pi i} \int_{|z|=1+\varepsilon} f(\eta(z)) z^{-n-1} dz, \quad (8)$$

where $\varepsilon > 0$ has to be chosen small enough [39, 36] such that g is analytic on $\eta(D_{1+\varepsilon})$, where $D_{1+\varepsilon} := \{z : |z| \leq 1+\varepsilon\}$. These polynomials p_j are called the *Faber polynomials* of E . Note that they only depend on η and not on f .

Let g be analytic on the unit disc, i.e., $g(z) = \sum_{j=0}^{\infty} c_j z^j$. The Faber transform of g is given by

$$[\mathcal{T}g](w) = \sum_{j=0}^{\infty} a_j p_j(w).$$

In other words, the *inverse* Faber transform $\mathcal{T}^{-1}f$ of f is given by replacing p_j by z_j in the Faber series (8) of f . Furthermore, the Faber coefficients a_j can be computed without knowing the Faber polynomials.

As already mentioned, the Faber transform maps rational functions onto rational functions. Hence, we can obtain an approximation to a function f defined on E by computing the CF approximation of the inverse Faber transform of f and then computing the Faber transform of the resulting rational approximation. The whole method is given in Algorithm 2.

Algorithm 2 Computation of the Faber-CF-Approximation

```

1: function FABERCF( $f, \eta, L, n$ )
2:    $(a_j)_{j=0}^{\infty} := \text{FABERSEQUENCE}(f)$  ▷  $f(w) = \sum_{j=0}^{\infty} a_j p_j(w)$ 
3:    $(r^{\text{cf}}, (z_1, \dots, z_K)) := \text{CF}((a_j)_{j=0}^L, n)$  ▷ use Alg. 1 here
4:    $\sigma_j := \eta(z_j)$  for  $j = 1, \dots, K$ 
5:    $(b_j^{(k)})_{j=0}^{\infty} := \text{FABERSEQUENCE}(w \mapsto \frac{1}{w - \sigma_k})$  for  $k = 1, \dots, K$ 
▷  $\sum_{j=0}^{\infty} b_j^{(k)} p_j(w) = \frac{1}{w - \sigma_k}$ 
▷  $\tilde{r}(z) = \sum_{j=0}^{\infty} c_j z^j$ 
6:    $(c_j)_{j=0}^{\infty} := \text{MACLAURIN}(r^{\text{cf}})$ 
7:   solve  $\begin{pmatrix} b_0^{(1)} \\ b_1^{(1)} \\ \vdots \\ b_{K-1}^{(1)} \end{pmatrix} \mid \dots \mid \begin{pmatrix} b_0^{(K)} \\ b_1^{(K)} \\ \vdots \\ b_{r-1}^{(K)} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{K-1} \end{pmatrix}$ 
8:   return  $((\sigma_1, \dots, \sigma_K), (\beta_1, \dots, \beta_K))$  ▷  $f(w) \approx \sum_{j=1}^K \frac{\beta_j}{w - \sigma_j}$ 
9: end function

```

First, the algorithm computes the coefficient of the Faber series (l. 2). These are the coefficients of the Maclaurin series of $\mathcal{T}^{-1}f$. The algorithm then uses the first $L + 1$ coefficients to compute a CF approximation for this analytic function (l. 3). It has been shown in [36] that the poles of $\mathcal{T}r^{\text{cf}}$ are $\eta(z_j)$, where z_j for $j = 1, \dots, K$ are the poles of r^{cf} . These poles are computed in the next step (l. 5). At this stage of the algorithm we know that the approximation takes the form $w \mapsto \sum_{j=1}^K \frac{\beta_j}{w - \sigma_j}$ and it remains to determine the coefficients β_j for $j = 1, \dots, K$. Since the Faber transform is linear, considering the Maclaurin series of both sides of the equation

$$r^{\text{cf}} = \mathcal{T}^{-1} \left(w \mapsto \sum_{j=1}^K \frac{\beta_j}{w - \sigma_j} \right)$$

yields a linear system for the coefficient β_j . This computation is the final step of the algorithm (l. 6–7).

For the purpose of applying this algorithm to the Schrödinger equation, we need to find a suitable mapping η . As mentioned before, the eigenvalues of the matrix M are all purely imaginary. Hence, we choose the conformal mapping

$$\eta(z) := \frac{R_1}{2} \left(z - \frac{1}{z} \right), \quad (9)$$

which maps the unit sphere S onto the interval $i[-R_1, R_1]$ imaginary axis, where all eigenvalues of the matrix M are located. It is here where the problem at hand needs to be taken into account. Specifically, if the matrix M has eigenvalues in a different domain, the Riemann mapping η needs to be chosen differently.

To summarize, with η tailored for the IVP (2), we can a priori compute shifts σ_j and coefficients β_j , $j = 1, \dots, K$, to compute the rational approximation $r(M)$ in partial fraction decomposition form (4). This allows us to approximate the matrix exponential in order to evaluate the matrix exponential (3) for a given time τ .

3.3. Step-Size Requirements

In principle, the exponential formula (3) allows us to compute arbitrary large time steps. There are, however, practical limitations. Assume, we choose a large time-step size τ . The solution of the IVP at time τ is given by $\exp(\tau M) u_0$. When τ is large, the spectral radius of τM is large as well. As discussed in Section 3.1 the rational approximation should be close to the true function values at the eigenvalues of τM , which makes the computation of the rational approximation more challenging.

Computing the CF approximation for a large domain is more expensive than computing it for a smaller domain. For a larger domain the degree of the rational approximation needs to be larger, and the number of terms of the Maclaurin series that we need also becomes larger, which makes the computation of the singular value decomposition more and more expensive. Furthermore, the computation of the CF approximation is already expensive on its own. Hence, we would like to compute the approximation only once and then reuse it. This choice, however, limits the step size that our method is able to perform, as we shall see.

Let us examine the approximation error of the REXI method. For this purpose, let $f : \mathbb{C} \rightarrow \mathbb{C}$ be a given function. We would like to compute the matrix function $f(G)$ for some matrix G . For an approximation r to f we define the error function e by $e(\omega) := f(\omega) - r(\omega)$. Computing $r(G)$ instead of $f(G)$ results in the error $f(G) - r(G)$ and it is easy to see that $f(G) - r(G) = e(G)$. Hence, to compute a bound on the approximation error, we need to find a bound on the norm of $e(G)$. It turns out that if we can bound the error function e in the eigenvalues of G , we can bound $e(G)$.

Proposition 2. *Let $G \in \mathbb{C}^{n \times n}$ and assume that $G = X \text{diag}(\omega_1, \dots, \omega_n) X^{-1}$. Furthermore, assume that there exists $\epsilon > 0$ such that*

$$e(\omega_j) < \epsilon \quad \text{for } j = 1, \dots, n.$$

Then, $\|e(G)\|_\infty \leq \epsilon \text{cond}_\infty(X)$ where $\|v\|_\infty := \max_j |v_j|$, $\|G\|_\infty$ is the corresponding operator-norm, $\|G\|_\infty = \max_k \sum_{j=0}^n |G_{kj}|$, and $\text{cond}_\infty(X) = \|X\|_\infty \|X^{-1}\|_\infty$.

Proof. Follows from [35, Theorem 4.25]. □

Table 1: A parameter choice for the Faber-CF-Approximation with corresponding properties.

K	η	R_1	Approx. Domain	$\ e\ _\infty$
16	see eq. (9)	10	$i[-10, 10]$	2.38×10^{-9}

We can apply this proposition to the case of solving the Schrödinger equation using the REXI method. We know that all eigenvalues of M lie on the imaginary axis. Assume that $e(\omega) = f(\omega) - r(\omega) < \epsilon$ for $\omega \in i[-R_1, R_1]$. Then, if we set

$$\tau = \frac{R_1}{\text{sr}(M)}, \quad (10)$$

where $\text{sr}(M) := \max_j |\lambda_j|$ is the spectral radius of M , we have that the eigenvalues of τM , which are $\tau \lambda_j$, fulfill that $|\tau \lambda_j| < R_1$. Hence, then the assumptions of Proposition 2 are satisfied. Thus, to guarantee proper simulations results we should make sure that (10) is satisfied. This restriction is inherent to all variants of the REXI methods.

Note that we only need a rough estimate for the largest eigenvalue of M in order to ensure that the accuracy condition (10) is fulfilled. Such an estimate can be obtained by running only a few iterations of a sparse eigensolver [see, e.g., 40, and the references therein]. Especially when running many time steps, the time for estimating the largest eigenvalue can be neglected, since it only needs to be computed once.

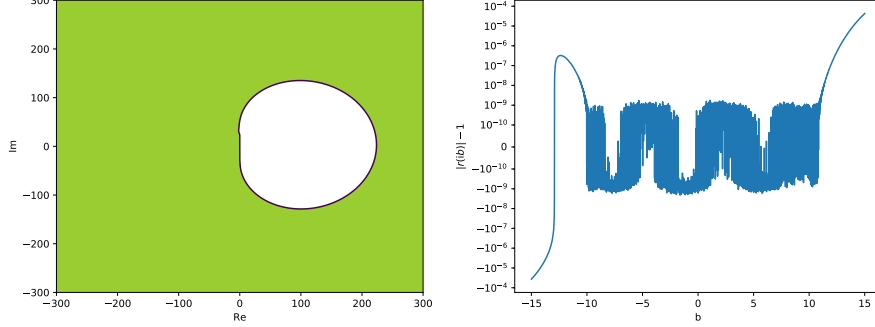
Furthermore, note that in order to compute larger time steps, we can choose larger values for R_1 in the conformal map (9). Choosing larger values of R_1 , however, is likely to increase the approximation error ϵ . Hence, to compensate for an increase in error, one needs to increase the degree K of the rational approximation, which increases the overall cost of the method, because it requires more linear systems to be solved.

In the remainder of this paper, we shall use the Faber-CF-Approximation defined by the parameters listed in Table 1. This choice leads to an approximation which has an error of roughly 1×10^{-9} on the approximation interval $i[-10, 10]$. This interval contains about three periods of $\exp(i\omega)$. Hence, using this approximation one time step of the REXI method can contain up to three oscillations of the solution (at a specific point in the spatial domain).

Comparing this approximation to the approximation derived in [21, 23], we constructed an approximation using the method from these publications with a comparable accuracy on the approximation interval $i[-10, 10]$. To the best of our knowledge, this approximation requires at least 34 poles to achieve the same accuracy. Thus, while the REXI method using the Faber-CF-Approximation needs to solve 16 linear systems per time step, the method from these publications needs to solve 68 linear systems for the same time-step size, because it has to solve two linear systems per pole, as discussed in Section 3.1.

3.4. Stability

Since every step of the computation introduces rounding errors and an approximation error, it is important that these errors are not amplified in the following steps.



(a) Stability domain (shaded region) of the REXI method. (b) $|r(z)| - 1$ for z on the imaginary axis. The points where these values are smaller than zero belong to the stability domain.

Figure 1: Stability of the REXI method using the Faber-CF approximation specified in Table 1.

Amplification of these errors would cause a run-away effect, which leads to an exponential growth of the error and needs to be avoided. Methods that are stable damp the error and thus keep the error under control.

A standard way to assess the *stability* of a method is to apply it to Dahlquist's test equation [41, 42], which is the ODE

$$\begin{cases} \frac{\partial u}{\partial t}(t) = \omega \cdot u(t) & \text{for } t > 0 \\ u(0) = 1 \end{cases} \quad (11)$$

for $\omega \in \mathbb{C}$. Let u_ω^n be the value computed by the method under consideration after n steps for a particular choice of ω . We define the *domain of absolute stability* of the method by

$$S = \{\omega \in \mathbb{C} : |u_\omega^n| \rightarrow 0 \text{ for } n \rightarrow \infty\}.$$

The process of solving our ODE of interest (2) using a method with stability domain S and time step τ is considered stable, if $\tau\lambda \in S$ for all eigenvalues λ of the matrix $M = (iB)^{-1}A$ [cf. 42, Chapter IV]. Hence, to analyze the stability of the REXI method, we compute the domain of absolute stability of the method.

To compute the stability domain, we need to solve the test equation (11) using the REXI method, which yields the iteration $u_\omega^{n+1} = r(\omega) \cdot u_\omega^n$, where r is the chosen rational approximation of the exponential. Thus, $u_\omega^n = (r(\omega))^n \cdot u_\omega^0$, and as a consequence, $|u_\omega^n| \rightarrow 0$ for $n \rightarrow \infty$ if and only if $|r(\omega)| < 1$. Therefore, the stability domain of the REXI method is given by

$$S = \{\omega \in \mathbb{C} : |r(\omega)| < 1\}.$$

Turning to the specific REXI method considered in this paper, we see in Figure 1(a) that the method has a large stability domain. The eigenvalues of the matrix M , however, are located on the imaginary axis, which is not fully contained in the stability domain

of the method. Taking a closer look at the values of $|r(z)| - 1$ for $z \in i\mathbb{R}$, as shown in Figure 1(b), reveals that $|r(z)|$ exceeds one by roughly 1×10^{-9} on the approximation interval $i[-10, 10]$. That $|r(z)|$ is close to one on the approximation interval is no surprise, because $r(z)$ approximates $\exp(z)$, and $|\exp(z)| = 1$ on the imaginary axis. The absolute value of the approximation, however, exceeds one on the approximation interval by at most $\|e\|_\infty$, the size of the approximation error. Hence, the method can be stabilized by multiplying all coefficients β_j used in rational approximation (4) by a factor of $1 - \epsilon$, where ϵ is slightly larger than $\|e\|_\infty$. Since this modification introduces an error which is of the same order of magnitude as the approximation error, the overall accuracy of the method is only slightly reduced. Recall that in order for all eigenvalues of M to be contained in the approximation interval, the accuracy condition (10) needs to be fulfilled.

There might be cases where the largest eigenvalues of the matrix M is not known and estimating its size is too expensive. Since, the degree of the numerator is smaller than the degree of the denominator of the rational approximation r , $|r(z)| \rightarrow 0$ for $|z| \rightarrow \infty$ and thus large eigenvalues have a good chance of falling into the region of stability. Hence, if the corresponding mode is irrelevant for the solution process, usable results might still be obtained. Nevertheless, we recommend to compute a rough estimate for the largest eigenvalue and use the accuracy condition (10) to choose the step size. Note, this behavior is in contrast to polynomial approximations, like the Chebyshev method [43], we will discuss below. For a polynomial (of degree larger than zero), $|p(z)| \rightarrow \infty$ for $|z| \rightarrow \infty$, and hence the method becomes almost immediately unstable if one of the eigenvalues of M lies outside of the approximation region.

Turning to the specific approximation as specified by Table 1, we observe that $|r(z)|$ exceeds one by only 10^{-9} , which results in an error growth proportional to $(1 + 10^{-9})^n = e^{n \cdot \log(1 + 10^{-9})} \approx e^{n \cdot 10^{-9}}$. Even though the error grows exponentially, since the base is only slightly larger than one, it requires a large number of time steps for the exponential effect to dominate. For example, for an error amplification of a factor of 2 we need to run the method for 7×10^8 time steps. Here, we do not plan to run our method for such a large number of time steps. Hence, in the following, we do not stabilize the method for the benefit of a higher accuracy.

4. Numerical Experiments

We carry out numerical experiments to study the potential, effectiveness and efficiency of the method presented above. To test the algorithms in a realistic setting, we simulate two different quantum mechanical systems that feature two famous quantum mechanical phenomena that cannot be explained by classical mechanics.

To this end, we use an implementation of the REXI method written in C++ and parallelized using MPI². For the finite element discretization we use libMesh [44] and all matrix and vector operations are implemented using the PETSc library [45]. All computations are performed on the JURECA³ cluster [46], which consists of 1872 nodes

²Message Passing Interface

³Jülich Research on Exascale Cluster Architectures

Table 2: Parameters of the quantum tunneling simulation.

Parameter	m	V_{\max}	C_{barr}	Σ	σ	$\bar{\mathbf{r}}$	$\bar{\mathbf{p}}$
Value	$1 m_e$	$15 E_h$	$0.005 a_0$	σI	$4 a_0^2$	$-3 a_0$	$5 \frac{\hbar}{a_0}$

connected via InfiniBand. The nodes we use contain a two-socket board equipped with two Intel Xeon E5-2680 v3 and 128GiB of memory.

In many applications we want to start the simulation with a particle at a certain location and with a certain momentum and see how it evolves in time. Due to the Heisenberg uncertainty relation, however, we can either give a quantum particle a defined position or a defined momentum, but not both. Hence we choose a Gaussian-wave package as initial condition.

A Gaussian wave-package is defined as

$$\psi(\mathbf{r}) := C_{\text{norm}} \cdot e^{-\frac{1}{2}(\mathbf{r}-\bar{\mathbf{r}})^T \Sigma^{-1}(\mathbf{r}-\bar{\mathbf{r}})} \cdot e^{i\langle \bar{\mathbf{p}}, \mathbf{r}-\bar{\mathbf{r}} \rangle / \hbar} \quad (12)$$

where $\Sigma \in \mathbb{R}^{d \times d}$ is a symmetric, positive definite matrix and $C_{\text{norm}} \in \mathbb{R}$ is chosen such that

$$\int_{\Omega} \bar{\psi} \psi \, d\mathbf{r} = 1.$$

The wave-package describes a particle ensemble with position expectation value $\bar{\mathbf{r}}$ and momentum expectation value $\bar{\mathbf{p}}$. The matrix Σ describes the uncertainty of the particle position. At the same time the matrix influences the uncertainty of the momentum—the smaller the uncertainty of the position the larger the uncertainty of the momentum and vice versa.

For simplicity, all quantities are measured in Hartree atomic units [47, 48], i.e., we choose a system of measurement in which the electron mass m_e , the elementary charge e , the reduced Plank constant \hbar , and the inverse Coulomb constant $4\pi\epsilon_0$ are all equal to one. In this system, length is measured in *bohr*, a_0 , i.e., the Bohr radius, and energy is measured in Hartree, $E_h = \hbar/(m_e a_0^2)$.

4.1. Quantum Tunneling

We start our numerical investigation by considering a simulation of quantum particle tunneling through a step-potential barrier. Quantum tunneling describes a phenomenon in which a quantum particle passes through a potential barrier even though its kinetic energy is smaller than the height of the barrier [see, e.g., 9, 10, 11]. This behavior is in contrast to classical mechanics where such a behavior is not possible.

We consider the tunneling process defined as follows. An electron moves in the step-potential given by

$$V(\mathbf{r}) = \begin{cases} V_{\max} & \text{if } |r_1| \leq \frac{C_{\text{barr}}}{2} \\ 0 & \text{otherwise} \end{cases},$$

which has a barrier at $x = 0$. The electron starts at the left of the barrier and moves to the right, with a speed typical for electrons that are emitted by electron guns. When the electron reaches the barrier it has a certain probability of being reflected from the

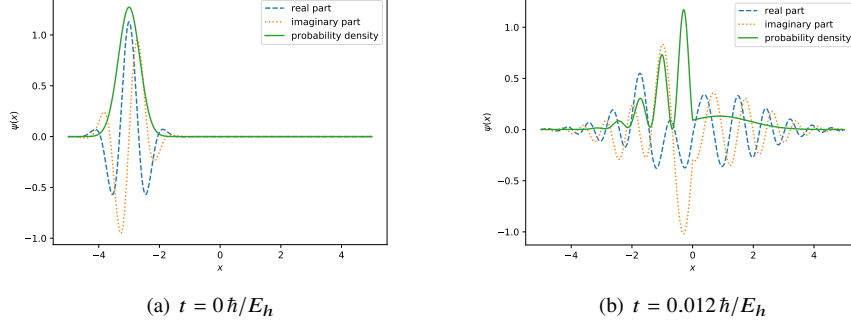


Figure 2: A 1D tunneling problem. Simulation of the Schrödinger equation using the parameters from Table 2 and a Gaussian wave-package as initial values at two different times.

barrier or tunneling through the barrier. We simulate this process by numerically solving the Schrödinger equation (1) with the step-potential, a Gaussian wave-package (12) as initial values, and the parameters listed in Table 2.

Figure 2 shows the state of the quantum system at two different times. In the initial state (Figure 2(a)) the probability density is concentrated at the left of the domain. At a later time (Figure 2(b)) a part of the particle ensemble has passed through the barrier at $x = 0$, which can be seen by a raise of the probability density for $x > 0$. A large portion of the particles, however, is reflected from the barrier, which results in the interference pattern caused by the superposition of the incoming and reflected waves.

For the numerical simulation, we discretize the equation using finite elements as described in Section 2. We choose 4000 equally sized finite elements of order two, which results in an ODE system with 8001 degrees of freedom. We then simulate this system using the REXI time-stepping scheme (Section 3.1).

In our first experiment, we compare the serial execution time and accuracy of the REXI method to other ODE solvers. This comparison is important, because we later want to investigate the parallel performance of the REXI method, and thus we need to know the fastest serial method as a reference point, to get a realistic impression of the effectiveness of the parallelization.

Using REXI requires the construction of a rational approximation of the exponential function. We compute the Faber-CF approximation (Section 3.2.2) as defined by the parameters listed in Table 1 and discussed earlier.

The first method that we compare the REXI method with is the Chebyshev method [43]. It works by approximating $\exp(z)$ via $p(z) := \sum_{k=0}^N a_k T_k(-iz/R)$ on the interval $-i[-R, R]$, where T_k ($k \in \mathbb{N}$) is the Chebyshev polynomial of degree k [49, 50] and $a_k \in \mathbb{R}$. The coefficients a_k can be efficiently computed using the FFT [51]. The polynomial is then used to approximate $\exp(\tau B^{-1}A)u_0$ by evaluating $p(\tau B^{-1}A)u_0$, which can be done using the Clenshaw algorithm [52]. In contrast to the REXI method, the Chebyshev method only allows for spatial parallelization. Note that because of the mass matrix B , stemming from the finite elements approach, the application of the Chebyshev method also involves the solution of linear systems, making it significantly

Table 3: Tunneling problem timings, LU decomposition in space. Comparison of different time-stepping schemes in serial. The time in parenthesis is the time needed for solving linear systems involving the mass matrix B .

	Δt	error	time/s
Chebyshev	2×10^{-4}	3.17×10^{-6}	7.49 (3.08)
REXI	2×10^{-4}	6.66×10^{-7}	2.59
Rosenbrock 4	2×10^{-4}	1.11×10^{-5}	4.50
	2×10^{-5}	2.43×10^{-6}	44.80
	5×10^{-6}	4.23×10^{-7}	179.56

more expensive than in the case of $B = I$. For the sake of a meaningful comparison, we match the approximation quality of the Chebyshev polynomial with the one of the Faber-CF approximation. We choose the approximation interval $i[-10, 10]$ and a polynomial of degree 26, which leads to an error of the same order of magnitude as the rational approximation.

The second method that we compare the REXI method with is a fourth order Rosenbrock method. More precisely, we choose the L -stable method listed in [42, Section IV.7, Table 7.2]. Rosenbrock methods are diagonally implicit Runge-Kutta methods, and hence the method requires the solution of four linear equations per time step. We use the implementation of this method provided in PETSc [45].

There are further methods for numerically simulating the ODE system arising from the discretization of the Schrödinger equation, e.g., the Crank-Nicolson [4] or the leapfrog [53] method (see also [30]). These methods, due to their low order, however, require very small time steps. Hence, we do not consider them in the comparison.

We simulate the 1D tunneling problem using the three different methods for a time period of $0.2\hbar/E_h$, and measure the time the different methods require for the time stepping. All linear systems are solved using the LU decomposition, implemented in the SuperLU_DIST software package [54], and all computations are performed sequentially. The results are listed in Table 3. Note that we choose Δt for the REXI and Chebyshev method such that the accuracy condition (10) is fulfilled.

Considering the results, we see that the REXI method is the overall fastest method for this simulation. Furthermore, when taking accuracy into account, the Rosenbrock method needs substantially more time steps to reach the same accuracy as the REXI method. Note that the Rosenbrock method has to solve only four linear systems, while the REXI method has to solve 16. Hence, we would expect that the Rosenbrock method would be four times faster per time step than the REXI method, while we measure it to be two times slower. We assume that the Rosenbrock method shows this behavior, because we used the generic implementation provided by PETSc, and a more specialized implementation would perform better. Nevertheless, due to the smaller step-size requirement of the Rosenbrock method, the REXI method would still be the fastest method. Justified by these results, we shall use the time of the serial execution of the REXI method as the reference point for computing the parallel speedup of the method.

Table 4: Tunneling problem timings, LU decomposition in space, four refinements.

nodes	cores	time/s				speedup	efficiency
		total	rhs	local	reduce		
1	1	39.92	1.38	38.51	0.01	1.00	1.00
1	2	26.12	1.49	23.87	0.76	1.53	0.76
1	4	19.20	1.70	16.29	1.21	2.08	0.52
1	8	16.21	2.74	12.01	1.45	2.46	0.31
1	16	15.06	3.33	7.82	3.90	2.65	0.17
2	2	21.64	1.36	19.52	0.75	1.84	0.92
4	4	12.11	1.31	9.73	1.07	3.30	0.82
8	8	8.13	1.31	4.92	1.89	4.91	0.61
16	16	5.68	1.27	2.37	2.03	7.03	0.44

In the second experiment, we want to investigate the parallelization potential of the REXI method. In this experiment we simulate the same quantum system. To obtain more meaningful time-measurements and to give the method enough work that can be distributed along multiple processors, we increase the number of degrees of freedom by refining the mesh four times. Each refinement split one mesh cell into two and thus roughly doubles the degrees of freedom each time. Bearing in mind the accuracy condition (10), we have to reduce the step size of the REXI method. We use a time step size of $5 \times 10^{-7} \hbar / E_h$ and simulate the system for $5 \times 10^{-4} \hbar / E_h$. Recall that we have two types of parallelization that we can use—time and space-parallelization. For now we restrict ourselves to inspecting the time-parallelization only.

Note that the time-parallelization is limited by the number of poles of the rational approximation, because the number of poles determine the maximum number K of linear systems in (5) that can/need to be solved simultaneously. Hence, using the approximation described above allows us to split the computation of one time step into 16 independent tasks.

The results of the experiment are listed in Table 4, which contains the time measurements for running the method with different numbers of nodes. In the first half of the table, we keep the number of compute nodes constant, which means that all tasks are running on the same two-socket system and thus have direct access to the same memory. In the second half of the table, we use one compute node per tasks, which means that each tasks has its own CPU⁴ and memory. In addition to the total runtime, the times for the individual phases of the algorithm are also given. The REXI method evaluates (5) in three phases. First, the method has to compute the right-hand side (rhs) of the linear systems by multiplying the matrix iB and the vector u_0 . Note that from the time-parallelization perspective, this is a sequential part of the algorithm. Second, each process performs the local part of the computation, i.e., it solves the

⁴Central Processing Unit

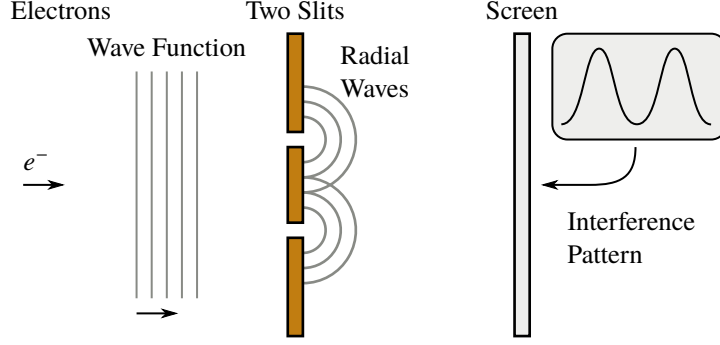


Figure 3: Double-slit experiment.

local linear systems and the local sums. Third, all processes compute the the global sum and distribute the result to all processes (reduce), which is the step that involves communication. In addition to the time measurements, the table contains the speedup and the parallel efficiency.

Inspecting the table, we see that the efficiency when running on one node is low. The time for the computation of the right-hand side increases with increasing number of cores. The time for the local computation achieves only a speedup of 5 when running on 16 cores. Furthermore, the time needed for the reduction increases. This behavior is due to the fact that modern CPUs can run at higher clock speeds when only a few cores are used, and that all cores on one node share the same memory interface, which becomes a limiting factor.

Using multiple nodes, the method scales better. We observe that the time for the right-hand side computation remains constant, which is expected, because it is the sequential part of the algorithm. The time for the local operations scales perfectly. The time for the global summation, however, increases. Hence, due to the sequential part and the increased communication cost, we only achieve an efficiency of 0.44. Note however, that this algorithm is not meant to be used alone. It should be used to provide additional parallelism in the situation where increasing spatial parallelism is not feasible anymore. With respect to these considerations the speedup is promising. As a next step, we considered a larger problem and combine temporal and spatial parallelism.

4.2. Double Slit Experiment

For the purpose of applying the REXI method to a larger problem, we consider the simulation of a double-slit interference experiment with electrons. This experiment demonstrates the wave-like character of matter particles and shows the limits of classical mechanics [10]. Assume that we are shooting electrons at a wall that has two thin slits, each of which can be closed. Most electrons will hit the wall, some, however, will pass through one of the slits. If we place a fluorescent screen at the other side of the wall, we can record the probability density of the incidenting electrons. We repeat this experiment three times. Once with both slits open, once where the first slit is closed, and once where the second slit is closed.

Table 5: Parameters of the double-slit experiment simulation.

Parameter	m	Σ	σ	$\bar{\mathbf{r}}$	$\bar{\mathbf{p}}$
Value	$1 m_e$	σI	$10^{-4} a_0^2$	$(0, -350) a_0$	$(0, 0.1) \frac{\hbar}{a_0}$

Classical mechanics would predict that the probability density we measure with both slits open is just the sum of the probability density that we obtain in the two cases where just one slit is open. It turns out, however, that we observe an interference pattern in the case of two open slits. This interference pattern can be explained using quantum mechanics. We describe the incidenting electrons by a planar probability wave that moves in the direction of the screen. When the planar wave hits the wall, each of the two slits emits radially outward going waves. These waves interfere, and when the electrons hit the screen, the probability density that results from this interference becomes visible. Figure 3 shows a schematic overview of the experiment.

Note that this experiment has never been actually performed in precisely this way. It resembles, however, the essential features of many experiments that have been performed without the technical complications they involve. Yet, Tonomura et al. conducted an experiment very close⁵ to the one that we described [55] and that we shall simulate.

For the simulation, we need to determine the appropriate parameters of the Schrödinger equation. We can model the wall with the two slits by choosing the domain of the PDE appropriately (see Figure 4), imposing zero Dirichlet boundary conditions. These conditions imply that the particle has a zero possibility of reaching the boundary of the domain and, hence, must be contained within. Since the electron is supposed to move freely within the domain we choose the zero potential, $V(\mathbf{r}) = 0$. Furthermore, to fulfill the condition (10) we chose a step size of $10 a_0$. The remaining parameters are listed in Table 5.

The state of the simulation is shown in Figure 4 at two different times. At the beginning the wave package is located in the lower part of the domain and moving towards the wall. At the second time most of the particles have been reflected from the wall, but a fraction of them have passed through either of the slits. At the other side of the wall an interference pattern forms.

We discretize the equation using the finite element method as described in Section 2. We use a suitable triangulation to obtain a discretization using 179620 finite elements of order two, leading to an ODE system with 288796 unknowns. Note that in this case the finite element method is by far the preferred discretization due to the complicated geometry of the domain.

Our first numerical experiment for this setting aims at comparing the space- and the time-parallelization. We use, again, SuperLU_DIST to solve all linear systems in the REXI method. The results can be found in Table 6. We see that the space-parallelization achieves a speedup of barely 1.36 when using 16 cores. This is due to the fact that SuperLU_DIST does not seem to parallelize well for this problem size, but we did

⁵Instead of a wall with two slits an electron biprism was used.

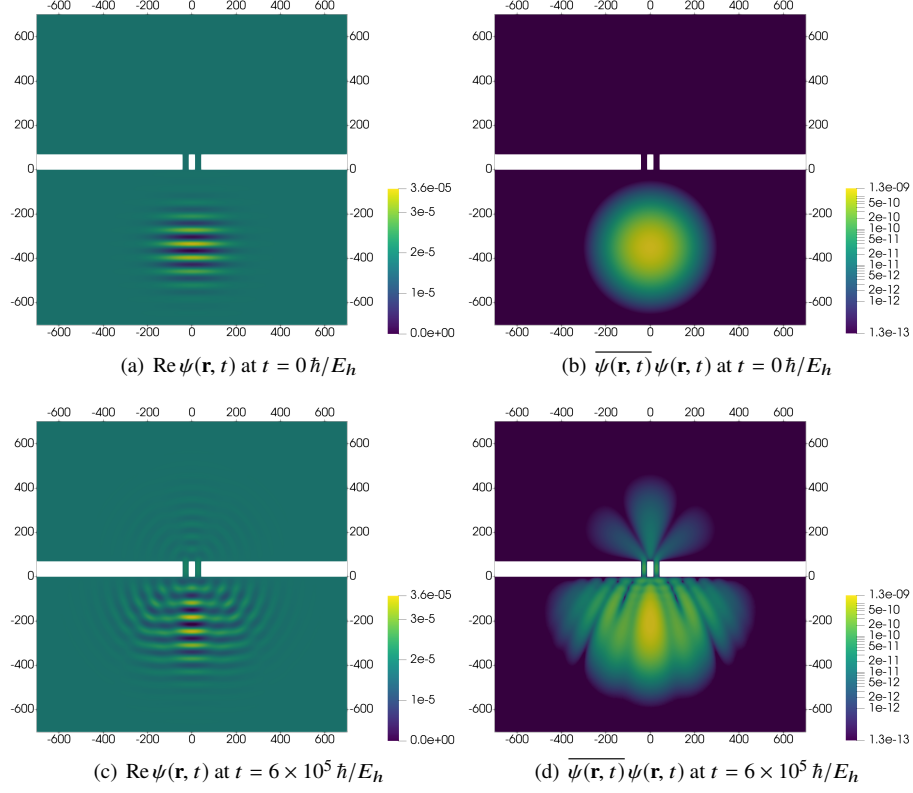


Figure 4: Simulation of the Schrödinger equation using parameters from Table 5 at two different times. The real part (left) and the probability density function (right) are shown.

Table 6: REXI with LU				
time \times space	cores		time / s	
	total	reduce	speedup	efficiency
1×1	132.85	0.00	1.00	1.00
1×2	123.77	0.00	1.07	0.54
1×4	114.43	0.00	1.16	0.29
1×8	99.70	0.00	1.33	0.17
1×12	94.68	0.00	1.40	0.12
1×16	97.45	0.00	1.36	0.09
2×1	78.00	0.04	1.70	0.85
4×1	46.74	0.07	2.84	0.71
8×1	27.14	0.20	4.90	0.61
16×1	17.42	1.51	7.63	0.48

Table 7: Double Slit, Solving with GMRES

cores		time / s			
time \times space	total	total	reduce	speedup	efficiency
1×1	1	299.74	0.00	1.00	1.00
1×12	12	46.62	0.00	6.43	0.54
1×24	24	21.29	0.00	14.08	0.59
1×48	48	7.81	0.00	38.40	0.80
1×96	96	2.97	0.00	100.87	1.05
1×192	192	2.01	0.00	149.08	0.78
1×384	384	1.98	0.00	151.15	0.39
8×24	192	2.84	0.09	105.72	0.55
8×48	384	1.18	0.02	254.50	0.66
8×96	768	1.05	0.04	286.56	0.37
16×12	192	3.12	0.13	96.04	0.50
16×24	384	1.57	0.11	190.62	0.50
16×48	768	0.89	0.08	335.15	0.44

not investigate this issue further. Furthermore, we can see that the time-parallelization is much more effective than the space-parallelization: using 16 cores in time gave a speedup of about 7.63, about 5 times as high as before. To avoid the complications induced by the LU solver and to compare to a more practical choice, for the next numerical experiment we replace the LU solver by the GMRES method [56], which is an iterative solver.

The results of the second numerical experiment can be found in Table 7. We increase the number of cores that we use for the space-parallelization until the speedup saturates. This happens at about 192 cores. Then we start adding more cores by increasing the time-parallelization. Instead of using 384 cores in space, we compute 8 summands of the rational approximation in parallel, each using 48 cores for the matrix-vector operations. This way, the same problem is solved, but the speedup is increased from 151.15 to 254.50. Doubling the number of cores in time to the maximum of 16 parallel summands, we get a maximum speedup of 335.15 on 768 CPU cores. Note that already when going from 1×48 to 8×48 instead of 1×384 (time \times space) the speedup is better. We see that by combining the space-parallelization with the time-parallelization we can increase the speedup substantially.

Note that an interesting effect is observed when using 1×96 cores—the efficiency is larger than one. We assume that this effect is caused by a better CPU utilization. The smaller the problem per node gets the less data needs to be stored on one node. Thus, at some point, the whole problem fits into the CPU cache of the node. Adding more (spatial) nodes to the problem, however, degrades efficiency severely.

5. Conclusions and Outlook

In this paper we have derived and applied a new variant of the rational approximation of exponential integrators (REXI) approach for the non-relativistic, single-particle Schrödinger equation. This time-integration scheme, being already more efficient than the standard integrators used for the examples in this work, can be parallelized efficiently. Each summand of the approximation can be computed in parallel, thus implementing a parallel-across-the-method approach, which augments a classical parallelization strategy in space. With this approach, scaling limits of distributed matrix- and vector-operations that correspond to operations in the spatial domain can be overcome. While parallel-in-time techniques are rather successful for problems of parabolic-type, propagations of waves like in the case of the Schrödinger equation are hard to tackle. With the REXI variant presented here, solving wave-type problems in a time-parallel manner is indeed possible, making efficient fully space-time parallel simulations of quantum systems with the Schrödinger equation possible for the first time.

We have derived and explained the rational approximation strategy chosen for this problem in detail, making use of the Faber-Carathéodory-Fejér approximation to compute the shifts and coefficients of the rational approximation of the matrix exponential. The derivation of the approximation algorithm in Section 3 can be used as a single-source reference to reproduce or potentially improve the numerical properties of this integrator. While the classical REXI method [21] is originally tailored for real-valued problems, this approach is also capable of dealing with complex-valued solutions in an efficient way. In comparison, fewer summands are necessary to achieve the same accuracy, leading to an improved ratio of accuracy per parallel task. We have shown along the lines of two challenging, real-world examples the impact of the parallel-in-time integrator, in particular with respect to a standard spatial parallelization technique.

For this work we have exclusively focused on the time-dependent, single-particle Schrödinger equation. The parallel-in-time method used and extended here was motivated by this equation, but its application is not limited to this particular problem. The approach can be extended to the many-particle Schrödinger equation and, using Newton’s method or a suitable implicit-explicit splitting strategy like spectral deferred corrections [57], general nonlinear Schrödinger equations can be addressed. However, there are features of the spatial discretization scheme, which actually limit the potential speedup gained by the REXI approach itself.

When assessing the potential of a parallel method, it is important to compute the speedup with respect to the fastest serial method available. In the case we considered in this paper, the REXI method was also the fastest serial method. This, however, is in general not the case.

Let us discuss some different situations in which we compare the REXI and the Chebyshev method, to highlight the factors that need to be taken into account when determining the speedup the REXI method can provide. We do not consider non-exponential methods like Crank-Nicolson, since there the time-step size is prohibitively small. For the sake of simplicity, we restrict ourselves to comparing the dominant costs of both methods. The dominant cost of the REXI method is the solution of n_R linear systems, while the dominant cost of the Chebyshev method is the computation of n_C

matrix vector products involving the matrix M (3). In the case that we considered in Section 4, $n_R = 16$ and $n_C = 26$. In general, solving a linear system is much more expensive than computing a matrix-vector product. The reason why in our case the serial REXI method is faster than the Chebyshev method is that $M = -iB^{-1}A$, i.e., applying M to a vector involves solving a linear system as well. If we use a discretization in which $B = I$, e.g., a finite difference discretization this argument no longer holds.

Consider the case where $B = I$. If the time it takes to solve one linear system is longer than it takes to compute n_C matrix-vector products, the REXI method provides no speedup over the Chebyshev method independent of the number of processors that are used. Note that for spectral methods with suitable domain geometries, the costs for solving a linear system and applying a matrix to a vector are very similar. Thus, for those discretizations REXI can provide speedup, too, and the original papers did indeed focus on those methods. In the case that we considered, $n_C = 26$. When solving the linear system not in spectral space but with a linear solver like GMRES, the statement essentially means that each system must be solved using fewer than 26 iterations, which is a severe limit on the number of iterations.

Let us now assume that we are in a situation where $B = I$ and solving one of the linear systems in (5) is actually faster than n_C matrix-vector products. If we use the GMRES method to solve the linear systems, we can use a method like the shifted GMRES method [58], which is able to solve a set of shifted linear systems at about the same cost as it takes to solve one system. While this leads to a very efficient method, it leaves no room for any speedup due to time-parallelization. If we now assume that we need to precondition the GMRES iteration and each shift required a different preconditioner, we can no longer apply the shifted GMRES method. Hence, in this situation, it is again possible to obtain a speedup using time-parallelization.

Thus, in the case where solving a linear system involving the matrix B is expensive enough, using the time-parallelization of the REXI method provides a speedup over solving sequentially. If solving these linear systems is cheap, it is not clear that the REXI method yields a speedup with respect to a certain sequential method. In the case of a finite element discretization, we are, however, in the situation, where solving a linear system is expensive enough to make the use of the time-parallel REXI method beneficial.

Acknowledgements

The authors would like to thank Martin Gander and Martin Schreiber for their valuable input, in particular during the 8th PinT Workshop at the Center for Interdisciplinary Research in Bielefeld, Germany. The authors furthermore thankfully acknowledge the provision of computing time on the JURECA cluster at Jülich Supercomputing Centre.

References

- [1] E. Schrödinger, Ann. Phys. 384 (1926) 361–376. doi:[10.1002/andp.19263840404](https://doi.org/10.1002/andp.19263840404).

- [2] J. Mazur, R. J. Rubin, J. Chem. Phys. 31 (1959) 1395–1412. doi:[10.1063/1.1730605](#).
- [3] E. A. McCullough, R. E. Wyatt, J. Chem. Phys. 51 (1969) 1253–1254. doi:[10.1063/1.1672133](#).
- [4] E. A. McCullough, R. E. Wyatt, J. Chem. Phys. 54 (1971) 3578–3591. doi:[10.1063/1.1675384](#).
- [5] Y. R. L. Park, C. T. Tahk, D. J. Wilson, J. Chem. Phys. 53 (1970) 786–791. doi:[10.1063/1.1674059](#).
- [6] G. G. Balint-Kurti, Theor. Chem. Acc. 127 (2010) 1–17. doi:[10.1007/s00214-010-0760-4](#).
- [7] X. Li, J. Chem. Phys. 150 (2019) 114111. doi:[10.1063/1.5079326](#).
- [8] C. A. Ullrich, Time-Dependent Density-Functional Theory. Concepts and Applications, Oxford University Press, 2016.
- [9] A. Messiah, Quantum Mechanics, Dover Publications, 1999.
- [10] R. Shankar, Principles of Quantum Mechanics, 2nd ed., Springer, 2014.
- [11] D. J. Griffiths, Introduction to Quantum Mechanics, 2nd ed., Cambridge University Press, 2017. doi:[10.1017/9781316841136](#).
- [12] K. Burrage, Adv. Comput. Math. 7 (1997) 1–31. doi:[10.1023/A:1018997130884](#).
- [13] R. Schöbel, R. Speck, PFASST-ER: Combining the Parallel Full Approximation Scheme in Space and Time with parallelization across the method, 2019. [arXiv:1912.00702](#).
- [14] A. T. Clarke, C. J. Davies, D. Ruprecht, S. M. Tobias, J. Comput. Phys. X 7 (2020) 100057. doi:[10.1016/j.jcpx.2020.100057](#).
- [15] S. Friedhoff, J. Hahne, S. Schöps, Proc. Appl. Math. Mech. 19 (2019) e201900262. doi:[10.1002/pamm.201900262](#).
- [16] D. Samaddar, D. Coster, X. Bonnin, L. Berry, W. Elwasif, D. Batchelor, Comput. Phys. Commun. 235 (2019) 246–257. doi:[10.1016/j.cpc.2018.08.007](#).
- [17] J. B. Schroder, R. D. Falgout, C. S. Woodward, P. Top, M. Lecouvez, in: 2018 IEEE Power & Energy Society General Meeting (PESGM), IEEE, pp. 1–5.
- [18] W. C. Agboh, D. Ruprecht, M. R. Dogar, Combining coarse and fine physics for manipulation using parallel-in-time integration, 2019. [arXiv:1903.08470](#).
- [19] L. N. Trefethen, J. A. C. Weideman, SIAM Rev. 56 (2014) 385–458. doi:[10.1137/130932132](#).

- [20] N. Hale, N. Higham, L. Trefethen, SIAM J. Numer. Anal. 46 (2008) 2505–2523. doi:[10.1137/070700607](https://doi.org/10.1137/070700607).
- [21] T. S. Haut, T. Babb, P. G. Martinsson, B. A. Wingate, IMA J. Numer. Anal. 36 (2016) 688–716. doi:[10.1093/imanum/drv021](https://doi.org/10.1093/imanum/drv021).
- [22] M. Schreiber, N. Schaeffer, R. Loft, Parallel Comput. (2019). doi:[10.1016/j.parco.2019.01.005](https://doi.org/10.1016/j.parco.2019.01.005).
- [23] M. Schreiber, P. S. Peixoto, T. Haut, B. Wingate, Int. J. High Perform. C. 32 (2018) 913–933. doi:[10.1177/1094342016687625](https://doi.org/10.1177/1094342016687625).
- [24] C. Johnson, Numerical Solution of Partial Differential Equations by the Finite Element Method, Dover Publications, 2009.
- [25] S. C. Brenner, L. R. Scott, The Mathematical Theory of Finite Element Methods, Springer, 2000.
- [26] D. Braess, Finite Elements. Theory, Fast Solvers, and Applications in Elasticity Theory, 3rd ed., Cambridge University Press, 2007.
- [27] H. Arai, I. Kanesaka, Y. Kagawa, Bull. Chem. Soc. Jpn. 49 (1976) 1785–1787. doi:[10.1246/bcsj.49.1785](https://doi.org/10.1246/bcsj.49.1785).
- [28] I. Kanesaka, H. Arai, K. Kawai, Bull. Chem. Soc. Jpn. 51 (1978) 28–32. doi:[10.1246/bcsj.51.28](https://doi.org/10.1246/bcsj.51.28).
- [29] W. Ritz, J. Reine Angew. Math. 135 (1909) 1–61. doi:[10.1515/crll.1909.135.1](https://doi.org/10.1515/crll.1909.135.1).
- [30] C. Leforestier, R. Bisseling, C. Cerjan, M. Feit, R. Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O. Roncero, R. Kosloff, J. Comput. Phys. 94 (1991) 59 – 80. doi:[10.1016/0021-9991\(91\)90137-A](https://doi.org/10.1016/0021-9991(91)90137-A).
- [31] R. Bellman, Introduction to Matrix Analysis, 2nd ed., SIAM, 1997.
- [32] J. Liesen, V. Mehrmann, Linear Algebra, Springer Undergraduate Mathematics Series, Springer, 2015. doi:[10.1007/978-3-319-24346-7](https://doi.org/10.1007/978-3-319-24346-7).
- [33] C. Moler, C. Van Loan, SIAM Rev. 20 (1978) 801–836. doi:[10.1137/1020098](https://doi.org/10.1137/1020098).
- [34] C. Moler, C. Van Loan, SIAM Rev. 45 (2003) 3–49 (electronic). doi:[10.1137/S00361445024180](https://doi.org/10.1137/S00361445024180).
- [35] N. J. Higham, Functions of Matrices. Theory and Computation, SIAM, 2008.
- [36] S. Ellacott, SIAM J. Numer. Anal. 20 (1983) 989–1000. doi:[10.1137/0720069](https://doi.org/10.1137/0720069).
- [37] L. N. Trefethen, Numer. Math. 37 (1981) 297–320. doi:[10.1007/BF01398258](https://doi.org/10.1007/BF01398258).
- [38] R. E. Greene, K.-T. Kim, Complex Anal. Synerg. 3 (2017) 1. doi:[10.1186/s40627-016-0009-7](https://doi.org/10.1186/s40627-016-0009-7).

- [39] J. H. Curtiss, Amer. Math. Monthly 78 (1971) 577–596.
- [40] G. W. Stewart, SIAM J. Matrix Anal. Appl. 23 (2002) 601–614. doi:[10.1137/S0895479800371529](https://doi.org/10.1137/S0895479800371529).
- [41] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, number 37 in Text in Applied Mathematics, Springer, 2000.
- [42] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, Springer Series in Computational Mathematics, 2nd ed., Springer, 2002. doi:[10.1007/978-3-642-05221-7](https://doi.org/10.1007/978-3-642-05221-7).
- [43] H. Tal-Ezer, R. Kosloff, J. Chem. Phys. 81 (1984) 3967–3971. doi:[10.1063/1.448136](https://doi.org/10.1063/1.448136).
- [44] B. S. Kirk, J. W. Peterson, R. H. Stogner, G. F. Carey, Eng. Comput. 22 (2006) 237–254. doi:[10.1007/s00366-006-0049-3](https://doi.org/10.1007/s00366-006-0049-3).
- [45] PETSc, PETSc website, ??? URL: <https://www.mcs.anl.gov/petsc>.
- [46] Jülich Supercomputing Centre, J. Large-Scale Res. Facilities 4 (2018) A132. doi:[10.17815/jlsrf-4-121-1](https://doi.org/10.17815/jlsrf-4-121-1).
- [47] D. R. Hartree, Math. Proc. Cambridge Philos. Soc. 24 (1928) 89–110. doi:[10.1017/S0305004100011919](https://doi.org/10.1017/S0305004100011919).
- [48] I. Mills, T. Cvitaš, K. Homann, N. Kallay, K. Kuchitsu (Eds.), Quantities, Units and Symbols in Physical Chemistry, 2nd ed., Blackwell Science, 1993.
- [49] T. J. Rivlin, An Introduction to the Approximation of Functions, Dover Publications, 1981.
- [50] M. J. D. Powell, Approximation theory and methods, Cambridge University Press, 1981.
- [51] L. N. Trefethen, Approximation Theory and Approximation Practice, SIAM, 2012.
- [52] C. W. Clenshaw, Math. Comp. 9 (1955) pp.118–120. doi:[10.1090/S0025-5718-1955-0071856-0](https://doi.org/10.1090/S0025-5718-1955-0071856-0).
- [53] A. Askar, A. S. Cakmak, J. Chem. Phys. 68 (1978) 2794–2798. doi:[10.1063/1.436072](https://doi.org/10.1063/1.436072).
- [54] X. S. Li, J. W. Demmel, ACM Trans. Math. Software 29 (2003) 110–140. doi:[10.1145/779359.779361](https://doi.org/10.1145/779359.779361).
- [55] A. Tonomura, J. Endo, T. Matsuda, T. Kawasaki, H. Ezawa, Am. J. Phys 57 (1989) 117–120. doi:[10.1119/1.16104](https://doi.org/10.1119/1.16104).
- [56] Y. Saad, M. H. Schultz, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869. doi:[10.1137/0907058](https://doi.org/10.1137/0907058).

- [57] M. L. Minion, Commun. Math. Sci. 1 (2003) 471–500.
- [58] A. Frommer, U. Glässner, SIAM J. Sci. Comput. 19 (1998) 15–26. doi:[10.1137/S1064827596304563](https://doi.org/10.1137/S1064827596304563).