SpezialForschungsBereich F 32

Karl–Franzens Universität Graz
Technische Universität Graz
Medizinische Universität Graz

A general spectral method for the numerical simulation of one-dimensional interacting fermions

Christian Clason Gregory von Winckel

SFB-Report No. 2011–016 July 2011

A–8010 GRAZ, HEINRICHSTRASSE 36, AUSTRIA

Supported by the
Austrian Science Fund (FWF)

Der Wissenschaftsfonds.

SFB sponsors:

- **Austrian Science Fund** (FWF)
- **University of Graz**
- **Graz University of Technology**
- **Medical University of Graz**
- **Government of Styria**
- **City of Graz**

A GENERAL SPECTRAL METHOD FOR THE NUMERICAL SIMULATION OF ONE-DIMENSIONAL INTERACTING FERMIONS*

Christian Clason† Gregory von Winckel†

July 7, 2011

This work introduces a general framework for the direct numerical simulation of systems of interacting fermions in one spatial dimension. The approach is based on a specially adapted nodal spectral Galerkin method, where the basis functions are constructed to obey the antisymmetry relations of fermionic wave functions. An efficient MATLAB program for the assembly of the stiffness and potential matrices is presented, which exploits the combinatorial structure of the sparsity pattern arising from this discretization to achieve optimal run-time complexity. This program allows the accurate discretization of systems with multiple fermions subject to arbitrary potentials, e. g., for verifying the accuracy of multi-particle approximations such as Hartree–Fock in the few-particle limit. It can be used for eigenvalue computations or numerical solutions of the time-dependent Schrödinger equation.

1 INTRODUCTION

An accurate simulation of interacting many-body fermionic systems is one of the grand challenges of modern computational physics. A first-principles calculation of quantum systems consisting of even a few fermions is difficult, as the application of standard finite difference or

*This work was supported by the Austrian Science Fund (FWF) under grant SFB F32 (SFB "Mathematical Optimization and Applications in Biomedical Sciences").

†Institute for Mathematics and Scientific Computing, Karl-Franzens-Universität Graz, Heinrichstr. 36, 8010 Graz, Austria ({gregory.von-winckel,christian.clason}@uni-graz.at)

finite element methods can lead to prohibitively large matrix approximations of the Hamiltonian. On the other hand, many-body approximation methods may be inaccurate for systems with a low number of particles. This is the case even in a single spatial dimension. One motivation for solving the multi-particle one-dimensional Schrödinger equation in a quantum well numerically is therefore to verify the accuracy of many-body approximations in the few-particle limit. Furthermore, one-dimensional wave functions arise in, e. g., electron transport in semiconductor quantum wires [2, 14] or single-well carbon nanotubes [19].

There is a considerable number of publications using approximation methods such as the Hartree–Fock or multi-configuration time-dependent Hartree–Fock (MCTDHF) method, the Hubbard model, and the Born–Oppenheimer approximation in reduced-dimensional systems. In addition, there currently exists a body of work related to simulation of identical particles in a variety of special cases, where the interaction potential is modeled by a delta function or by certain singular potentials that are aphysical, but which model physical potentials such as Coulomb interaction. In some cases, exactly solvable model problems have been formulated [15]. There are also programs for the two-electron atom with fixed nuclear position [9], and recently, the energy levels of two particles in an infinite potential well with quadratic interaction potential have been computed [1].

However, there are comparatively few works on the direct numerical simulation of interacting fermions [24, 5, 4]. Most are based on sparse grid techniques [12, 25, 11] due to the combinatorial growth in the number of freedom with increasing number of particles. In comparison, spectral methods are very efficient in single-particle discretizations – in particular, exponential convergence can be expected for smooth potentials, – but the resulting matrices are frequently very dense, which is an obstacle in their application to multi-particle discretizations.

We therefore propose a nodal spectral Galerkin scheme where the basis functions are constructed to obey the antisymmetry relations of the fermionic wave function, thereby drastically reducing the degrees of freedom, increasing the sparsity of the stiffness matrix, and diagonalizing the potential matrix. Furthermore, we derive an efficient algorithm for the assembly of these matrices that exploits the combinatorial structure of the sparsity patterns to achieve linear scaling in the number of nonzero entries. We believe the present work is the first general approach that allows solving small systems of one-dimensional interacting fermions with arbitrary potential terms using the full multi-particle wave function. It should be noted that due to its generality, the program will not be optimal for some application models such as those with low regularity; however, it will be useful as a base for adaptations and optimizations for these specific situations.

Since an efficient MATLAB implementation of the proposed approach is by necessity concise and makes heavy use of various features of MATLAB programming – some of which may be of independent interest – we give a detailed derivation and explanation of the program. In particular, we present our procedure in steps, first presenting the abstract algorithm for the assembly, followed by its implementation using the linear algebra tools of MATLAB, and finally the use of vectorization to accelerate the assembly.

The paper is organized as follows. In the next section, we briefly state the multi-particle problem with which we are concerned. Section 3 describes our discretization of the single-

particle (§ 3.1) and multi-particle (§ 3.2) wave function. The efficient assembly of the resulting stiffness matrix is the focus of section 4, where we first describe the abstract algorithm in § 4.1, followed by its efficient implementation in MATLAB in § 4.2. Numerical experiments illustrating the accuracy and performance of the proposed method are presented in section 5.

2 PROBLEM FORMULATION

Let $\psi(\mathbf{x}, t) = \psi(x_1, \dots, x_N, t)$, $x_i \in [-1, 1]$, denote the multi-particle wave function and consider the Hamiltonian

$$H(\mathbf{x}, t)\psi(\mathbf{x}, t) = \left\{ -\frac{1}{2}\Delta + \sum_{j=1}^{N} \left(U(x_j, t) + \sum_{k>j}^{N} V(x_j, x_k) \right) \right\} \psi(\mathbf{x}, t),$$

where $\Delta = \partial_{x_1}^2 + \cdots + \partial_{x_N}^2$ is the N-dimensional Laplacian with appropriate boundary conditions (e. g., homogeneous Dirichlet or periodic). The potential $U(x, t)$ is an externally-applied potential which affects each fermion individually, whereas the symmetric interaction potential $V(x_j, x_k)$ is determined by the relative locations of the j-th and k-th particle. For electrons, this potential is Coulombic, i. e.,

$$V(x_j, x_k) = \frac{q}{|x_j - x_k|},$$

where q is the charge carried by an electron.

The Hamiltonian occurs in the time-dependent Schrödinger equation

$$i\partial_t \psi(\mathbf{x}, t) = H(\mathbf{x}, t)\psi(\mathbf{x}, t) \tag{2.1}$$

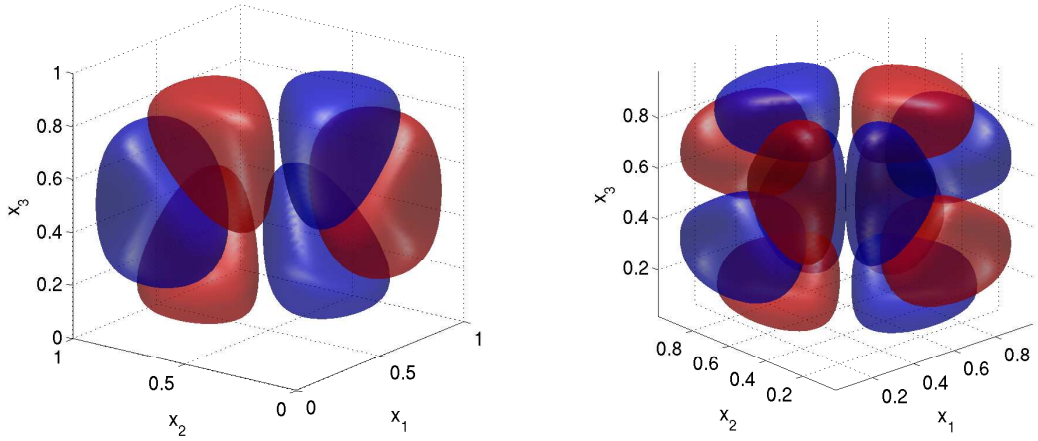
or in the eigenvalue problem

$$H(\mathbf{x})\psi(\mathbf{x}) = \lambda\psi(\mathbf{x}) \tag{2.2}$$

for the computation of stationary eigenstates of (2.1) for a time-independent potential $U(x_j)$. For the sake of exposition, we focus on (2.2) in the following, even though the method is equally applicable for the numerical solution of the time-dependent Schrödinger equation (2.1).

We can identify each composite state of a system of N identical fermions with a tuple $\alpha = (\alpha_1, \dots, \alpha_N)$ describing the combination of states of the single particles. Without interaction, the eigenfunctions $\psi_\alpha(\mathbf{x})$ for multiple particle systems can be constructed from Slater determinants (cf., e. g., [20, sec. 1.4]) of the one-particle eigenfunctions $\psi_j(x)$ [8, § 20.5]:

$$\psi_\alpha(\mathbf{x}) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_{\alpha_1}(x_1) & \psi_{\alpha_2}(x_1) & \cdots & \psi_{\alpha_N}(x_1) \\ \psi_{\alpha_1}(x_2) & \psi_{\alpha_2}(x_2) & \cdots & \psi_{\alpha_N}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{\alpha_1}(x_N) & \psi_{\alpha_2}(x_N) & \cdots & \psi_{\alpha_N}(x_N) \end{vmatrix}. \tag{2.3}$$



(a) $\psi_{1,2,3}(x_1, x_2, x_3)$ (b) $\psi_{1,2,4}(x_1, x_2, x_3)$

Figure 1: Isosurfaces of eigenfunctions. Red and blue surfaces depict level set where ψ is exactly half of its maximum or minimum value respectively.

For example, in two dimension, the eigenfunction have the form

$$\psi_{j,k}(x_1, x_2) = \frac{1}{\sqrt{2}} \left[\psi_j(x_1)\psi_k(x_2) - \psi_k(x_1)\psi_j(x_2)\right], \quad j \neq k.$$

The eigenfunctions for a three-particle system are

$$\begin{aligned}\psi_{i,j,k}(x_1, x_2, x_3) = \frac{1}{\sqrt{6}} \big[&\psi_i(x_1)\psi_2(x_2)\psi_3(x_3) + \psi_j(x_1)\psi_k(x_2)\psi_i(x_3) \\ &+ \psi_k(x_1)\psi_i(x_2)\psi_j(x_3) - \psi_k(x_1)\psi_j(x_2)\psi_i(x_3) \\ &- \psi_i(x_1)\psi_k(x_2)\psi_j(x_3) - \psi_j(x_1)\psi_i(x_2)\psi_k(x_3)\big]\end{aligned}$$

for all $i \neq j \neq k$, which can be visualized by plotting isosurfaces (see Figure 1).

3 DISCRETIZATION

For the numerical simulation, we express (2.2) in weak form and replace the wave function ψ by an approximation from a finite-dimensional subspace. We make use of a tensor product approach, where the multi-particle wave function is written as a product of single-particle wave functions. Due to the combinatorial explosion of the degrees of freedom with growing N, it is crucial to select a convenient discretization for a single particle that yields a sparse basis for the multi-particle case.

3.1 DISCRETIZATION OF SINGLE PARTICLE

To keep the degrees of freedom for a single particle as low as possible, we work with a nodal spectral Galerkin discretization, where the eigenvalue problem (2.2) is projected onto the span of p Lagrange polynomials and the occurring inner products are computed approximately using either Gauss–Lobatto quadrature (for Dirichlet boundary conditions) or Gauss–Fourier quadrature (for periodic boundary condition). This approach is called *Galerkin numerical integration method*, and has been shown to be algebraically equivalent to a pseudospectral method on the same grid [7].

For non-periodic problems, we use the Legendre–Gauss–Lobatto (LGL) nodes

$$\{\xi_0, \dots, \xi_{p+1}\} = \{x \in \Omega : P'_{p+1}(x) = 0\} \cup \{\pm 1\},$$

where $P_k(x)$ is the k-th Legendre polynomial. The corresponding quadrature weights are

$$w_k = \frac{1}{(p+1)(p+2)} \frac{2}{[P_{p+1}(\xi_k)]^2}, \qquad k = 0, \dots, p+1.$$

The Lobatto quadrature is exact for all polynomials up to order $2p+1$.

The Lagrange polynomials corresponding to the LGL nodes are

$$\tilde{\ell}_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{p+1} \frac{x - \xi_k}{\xi_j - \xi_k}, \qquad j = 0, \dots, p+1,$$

which have the important property of being nodal interpolants, i. e.,

$$\tilde{\ell}_j(\xi_k) = \delta_{jk} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}.$$

A key idea in our discretization is to work with the scaled Lagrange polynomials

$$\ell_j(x) = \frac{1}{\sqrt{w_j}} \tilde{\ell}_j(x),$$

for which the mass matrix will become the identity. The derivatives of the Lagrange polynomials are given by [3]

$$\ell'_j(\xi_k) = \begin{cases} \frac{b_j / b_k}{\xi_j - \xi_k} & \text{if } j \neq k, \\ -\sum_{i \neq j} \ell'_j(\xi_i) & \text{if } j = k, \end{cases}$$

using the barycentric weights

$$b_j = \left(\prod_{\substack{k=0 \\ k \neq j}}^{p+1} \xi_j - \xi_k \right)^{-1}, \quad j = 0, \dots, p+1.$$

The single-particle wave function is then approximated nodally using a linear combinarion of the Lagrange polynomials $\{\ell_1, \ldots, \ell_p\}$, thereby excluding the end points and enforcing a homogeneous Dirichlet condition:

$$\psi(x) \approx \sum_{k=1}^{p} \hat{\psi}_k \tilde{\ell}_k(x), \quad \psi(\pm 1) = 0. \tag{3.1}$$

Substituting (3.1) into the weak form of (2.2) (without interaction potential) yields

$$\sum_{k=1}^{p} \left[\langle \ell_j'(x), \ell_k'(x) \rangle + \langle \ell_j(x), U(x) \ell_k(x) \rangle \right] \hat{\psi}_k = \lambda \sum_{k=1}^{p} \langle \ell_j(x), \ell_k(x) \rangle \hat{\psi}_k$$

for all $j \in \{1, \ldots, p\}$. By numerically computing the inner products using Gauss–Lobatto quadrature, we obtain the single-particle mass matrix M, potential matrix U and stiffness matrix K with the entries

$$\begin{aligned} M_{jk} &= \sum_{i=1}^{p} \ell_j(\xi_i) \ell_k(\xi_i) w_i = \delta_{jk}, \\ U_{jk} &= \sum_{i=1}^{p} \ell_j(\xi_i) \ell_k(\xi_i) U(\xi_i) w_i = U(\xi_j) \delta_{jk}, \\ K_{jk} &= \sum_{i=1}^{p} \ell_j'(\xi_i) \ell_k'(\xi_i) w_i. \end{aligned}$$

Note that by the nodal basis property of the ℓ_j, M is the identity matrix, U is a diagonal matrix with the potential evaluated at the LGL nodes on the diagonal, and K is a full matrix. Problem (2.2) is thus reduced to a standard eigenvalue problem for the matrix $K + U$.

For periodic problems, the approach is similar to the above but simplified. The periodic Fourier nodes and quadrature weights are

$$\xi_j = \frac{2\pi j}{p}, \qquad w_j = \sqrt{\frac{2\pi}{p}}.$$

The trigonometric Lagrange polynomials are [23]

$$\ell_j(x) = \frac{\sin(\frac{p}{2}(x - \xi_j))}{p \sin(\frac{1}{2}(x - \xi_j))}, \quad j = 0, \ldots, p-1,$$

which are nodal interpolants at the Fourier nodes. The explicit form of the stiffness matrix elements is then [7, eq. (2.1.52)]

$$K_{jk} = \begin{cases} -\frac{1}{4}(-1)^{j+k} p - \frac{(-1)^{j+k+1}}{2 \sin^2\left(\frac{(j-k)\pi}{p}\right)}, & j \neq k, \\ \frac{(p-1)(p-2)}{12} & j = k. \end{cases}$$

As above, M is the identity matrix and U is a diagonal matrix with elements given by the potential evaluated at the quadrature nodes.

3.2 DISCRETIZATION OF MULTIPLE PARTICLES

Motivated by the Slater construction (2.3) of the eigenfunctions, the multi-particle wave function is discretized using tensor products of N identical single-particle discretizations with p degrees of freedom each. Introducing a tuple $\alpha = (\alpha_1, \ldots, \alpha_N)$, we can write

$$\psi(x_1, \ldots, x_N) \approx \sum_{\alpha \leqslant p} \hat{\psi}_\alpha \varphi_\alpha(x_1, \ldots, x_N) := \sum_{\alpha \leqslant p} \hat{\psi}_\alpha \ell_{\alpha_1}(x_1) \cdots \ell_{\alpha_n}(x_N).$$

where we have used the convention that $\alpha \leqslant p$ holds if $\alpha_j \leqslant p$ for all $1 \leqslant j \leqslant N$. A naive application of this discretization would require the determination of p^N unknown coefficients, which is clearly impractical for reasonable p and $N > 3$. However, the degrees of freedom are greatly reduced by enforcing the anti-symmetry property of the fermionic wave function arising from the Pauli exclusion principle [20, § 1.4.1], i. e.,

$$\psi(\ldots, x_i, \ldots, x_j, \ldots) = -\psi(\ldots, x_j, \ldots, x_i, \ldots) \text{ for all } 1 \leqslant i, j \leqslant N,$$

A particular consequence of this is that the expansion coefficient $\hat{\psi}_\alpha$ must be zero if at least two elements of α are equal. These symmetries can be expressed using Slater determinants to construct the trial functions. Specifically, for $\alpha = (\alpha_1, \ldots, \alpha_N)$ we set

$$\varphi_\alpha(x_1, \ldots, x_N) = \begin{vmatrix} \ell_{\alpha_1}(x_1) & \cdots & \ell_{\alpha_N}(x_1) \\ \vdots & \ddots & \vdots \\ \ell_{\alpha_1}(x_N) & \cdots & \ell_{\alpha_N}(x_N) \end{vmatrix}.$$

The Slater determinant can only be nonzero when the indices of each of the Lagrange polynomials are distinct. Furthermore, two basis functions φ_α and φ_β corresponding to two tuples α and β are identically equal up to a sign change if α is a permutation of β, where the sign change is determined by the signature of this permutation. Therefore it suffices just to consider tuples α that form an ascending non-repeating N-tuple. Let $\mathcal{T}$ denote the set of all ascending non-repeating N-tuples, endowed with the lexicographical order, hereafter denoted by $\succ$. We introduce an enumeration

$$\tau : \mathcal{T} \to \{1, \ldots, N_p\},$$

which assigns to each N-tuple α the number of elements in $\mathcal{T}$ which are less than or equal to α with respect to $\succ$. Restricting the basis functions to the set $\{\varphi_\alpha : \alpha \in \mathcal{T}\}$ reduces their number to

$$N_p = \binom{p}{N} = \frac{p!}{N!(p-N)!},$$

since there are N! possible orderings of each N-tuple. This also corresponds to the fact that the computational domain – the N-dimensional hypercube – can be tessellated into N! uniform N-simplices

Example 3.1. *For* $p = 4$ *and* $N = 3$*, we have* $N_p = 4$ *and*

$$\mathcal{T} = \{(1,2,3), (1,2,4), (1,3,4), (2,3,4)\}.$$

The basis function corresponding to the tuple $\alpha = (1,3,4)$ *with* $\tau(\alpha) = 3$ *is*

$$\begin{aligned}\varphi_\alpha(x_1,x_2,x_3) &= \begin{vmatrix} \ell_1(x_1) & \ell_3(x_1) & \ell_4(x_1) \\ \ell_1(x_2) & \ell_3(x_2) & \ell_4(x_2) \\ \ell_1(x_3) & \ell_3(x_3) & \ell_4(x_3) \end{vmatrix} \\ &= \ell_1(x_1)\ell_3(x_2)\ell_4(x_3) + \ell_3(x_1)\ell_4(x_2)\ell_1(x_3) + \ell_4(x_1)\ell_1(x_2)\ell_3(x_3) \\ &\quad - \ell_4(x_1)\ell_3(x_2)\ell_1(x_3) - \ell_1(x_1)\ell_4(x_2)\ell_3(x_3) - \ell_3(x_1)\ell_1(x_2)\ell_4(x_3).\end{aligned}$$

Substituting $\psi(\mathbf{x}) \approx \sum_{\beta\in\mathcal{T}} \hat{\psi}_\beta \varphi_\beta(\mathbf{x})$ into the weak form of (2.2) yields

$$\sum_{\beta\in\mathcal{T}} \Big[\langle \nabla\varphi_\alpha(\mathbf{x}), \nabla\varphi_\beta(\mathbf{x})\rangle + \sum_{i=1}^{N} \langle \varphi_\alpha(\mathbf{x}), U(x_i)\varphi_\beta(\mathbf{x})\rangle + \sum_{i=1}^{N}\sum_{j=i+1}^{N} \langle \varphi_\alpha(\mathbf{x}), V(x_i,x_j)\varphi_\beta(\mathbf{x})\rangle \Big] \hat{\psi}_\beta = \lambda \sum_{\beta\in\mathcal{T}} \langle \varphi_\alpha(\mathbf{x}), \varphi_\beta(\mathbf{x})\rangle \hat{\psi}_\beta \tag{3.2}$$

for all $\alpha \in \mathcal{T}$. To evaluate the inner products, it is not necessary to construct the Slater determinants explicitly. We first consider the zero-order terms. Since the inner products are computed by a quadrature rule, we can exploit the nodal basis properties of the tensor product Lagrange polynomials. Let $\boldsymbol{\xi}_\gamma = (\xi_{\gamma_1}, \ldots, \xi_{\gamma_N})^T$ denote the tensor product quadrature points and $w_\gamma = \prod_{j=1}^{N} w_{\gamma_j}$ the corresponding quadrature weights. Due to the antisymmetry properties of the basis functions, it is sufficient to carry out the numerical quadrature on one simplex instead of on the full tensor product grid. For convenience, we choose the simplex that is characterized by the condition $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_N$, such that the quadrature nodes are given by the set $\{\boldsymbol{\xi}_\gamma : \gamma \in \mathcal{T}\}$. Then we have by applying Laplace's rule to the first column (noting that $\alpha_i \neq \alpha_j$ for $i \neq j$)

$$\varphi_\alpha(\boldsymbol{\xi}_\gamma) = \begin{vmatrix} \ell_{\alpha_1}(\xi_{\gamma_1}) & \cdots & \ell_{\alpha_N}(\xi_{\gamma_1}) \\ \vdots & \ddots & \vdots \\ \ell_{\alpha_N}(\xi_{\gamma_N}) & \cdots & \ell_{\alpha_N}(\xi_{\gamma_N}) \end{vmatrix} = \begin{cases} (w_\gamma)^{-1/2} & \text{if } \alpha = \gamma, \\ 0 & \text{else.} \end{cases}$$

Now let $j = \tau(\alpha)$ and $k = \tau(\beta)$. Then the corresponding element of the mass matrix $\mathbf{M}$ is given by

$$\mathbf{M}_{jk} = \langle \varphi_\alpha(\mathbf{x}), \varphi_\beta(\mathbf{x})\rangle = \sum_{\gamma\in\mathcal{T}} w_\gamma \varphi_\alpha(\boldsymbol{\xi}_\gamma)\varphi_\beta(\boldsymbol{\xi}_\gamma) = \delta_{jk},$$

i. e., the mass matrix is the identity as claimed. Thus, (3.2) represents a standard eigenvalue problem for the matrix $\mathbf{K} + \mathbf{U} + \mathbf{V}$. (We point out that generically, Galerkin schemes lead to generalized eigenvalue problems.) This property is also useful for the numerical solution

of the time-dependent Schrödinger equation using leap-frog schemes since in this case, no matrix inversions are required. Similarly, the confinement potential matrix $\mathbf{U}$ has the elements

$$\begin{aligned}\mathbf{U}_{jk} &= \sum_{i=1}^{N} \langle \varphi_\alpha(\mathbf{x}), U(x_i)\varphi_\beta(\mathbf{x})\rangle = \sum_{\gamma\in\mathcal{T}} \sum_{i=1}^{N} U(\xi_{\gamma_i}) w_\gamma \varphi_\alpha(\xi_\gamma)\varphi_\beta(\xi_\gamma) \\ &= \left(\sum_{i=1}^{N} U(\xi_{\alpha_i})\right)\delta_{jk},\end{aligned}$$

such that $\mathbf{U}$ is a diagonal matrix with the confinement potential evaluated on the quadrature nodes on the diagonal. The elements of the interaction potential matrix $\mathbf{V}$ are

$$\begin{aligned}\mathbf{V}_{jk} &= \sum_{n=1}^{N}\sum_{m=n+1}^{N} \langle \varphi_\alpha(\mathbf{x}), V(x_n, x_m)\varphi_\beta(\mathbf{x})\rangle \\ &= \sum_{\gamma\in\mathcal{T}} \left(\sum_{n=1}^{N}\sum_{m=n+1}^{N} V(\xi_{\gamma_n}, \xi_{\gamma_m})\right) w_\gamma \varphi_\alpha(\xi_\gamma)\varphi_\beta(\xi_\gamma) \\ &= \left(\sum_{n=1}^{N}\sum_{m=n+1}^{N} V(\xi_{\alpha_n}, \xi_{\alpha_m})\right)\delta_{jk}.\end{aligned}$$

We now turn to the elements of the stiffness matrix,

$$\mathbf{K}_{jk} = \langle \nabla\varphi_\alpha(\mathbf{x}), \nabla\varphi_\beta(\mathbf{x})\rangle = \sum_{n=1}^{N} \langle \partial_{x_n}\varphi_\alpha(\mathbf{x}), \partial_{x_n}\varphi_\beta(\mathbf{x})\rangle. \tag{3.3}$$

Here we cannot use the antisymmetry properties in the same way as before. However, it is still possible to avoid evaluating the Slater determinants, since the Löwdin rules [17, 18] state that the inner product of two Slater determinants

$$A(\mathbf{x}) = \begin{vmatrix} a_1(x_1) & \cdots & a_N(x_1) \\ \vdots & \ddots & \vdots \\ a_1(x_N) & \cdots & a_N(x_N) \end{vmatrix} \quad \text{and} \quad B(\mathbf{x}) = \begin{vmatrix} b_1(x_1) & \cdots & b_N(x_1) \\ \vdots & \ddots & \vdots \\ b_1(x_N) & \cdots & b_N(x_N) \end{vmatrix}$$

can be expressed as

$$\langle A(\mathbf{x}), B(\mathbf{x})\rangle = \begin{vmatrix} \langle a_1, b_1\rangle & \cdots & \langle a_1, b_N\rangle \\ \vdots & \ddots & \vdots \\ \langle a_N, b_1\rangle & \cdots & \langle a_N, b_N\rangle \end{vmatrix}.$$

We can thus write each term in (3.3) in terms of elements of the one-dimensional mass

$(M = I)$ and stiffness (K) matrices

$$
\begin{aligned}
(3.4)\qquad & \langle \partial_{x_n} \varphi_\alpha(\mathbf{x}), \partial_{x_n} \varphi_\beta(\mathbf{x}) \rangle = \\
& \begin{vmatrix}
\langle \ell_{\alpha_1}, \ell_{\beta_1} \rangle & \dots & \langle \ell_{\alpha_1}, \ell_{\beta_{n-1}} \rangle & \langle \ell'_{\alpha_1}, \ell'_{\beta_n} \rangle & \langle \ell_{\alpha_1}, \ell_{\beta_{n+1}} \rangle & \dots & \langle \ell_{\alpha_1}, \ell_{\beta_N} \rangle \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\langle \ell_{\alpha_N}, \ell_{\beta_1} \rangle & \dots & \langle \ell_{\alpha_N}, \ell_{\beta_{n-1}} \rangle & \langle \ell'_{\alpha_N}, \ell'_{\beta_n} \rangle & \langle \ell_{\alpha_N}, \ell_{\beta_{n+1}} \rangle & \dots & \langle \ell_{\alpha_1}, \ell_{\beta_N} \rangle
\end{vmatrix} \\
& = \begin{vmatrix}
\delta_{\alpha_1\beta_1} & \cdots & \delta_{\alpha_1\beta_{n-1}} & K_{\alpha_1\beta_n} & \delta_{\alpha_1\beta_{n+1}} & \cdots & \delta_{\alpha_1\beta_N} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\delta_{\alpha_N\beta_1} & \cdots & \delta_{\alpha_N\beta_{n-1}} & K_{\alpha_N\beta_n} & \delta_{\alpha_N\beta_{n+1}} & \cdots & \delta_{\alpha_N\beta_N}
\end{vmatrix} =: \det(K^n_{\alpha,\beta}).
\end{aligned}
$$

Since α and β contain non-repeating elements, every column of $K^n_{\alpha,\beta}$ apart from the n-th one contains at most one nonzero element (which is equal to 1). This allows computing the entries of the stiffness matrix without evaluation of determinants. Depending on α and β, we can discriminate between three cases:

1. $\alpha = \beta$: In this case, $K^n_{\alpha,\beta}$ is the identity matrix with the n-th column replaced by a selection of the diagonal entries of the one-dimensional stiffness matrix K. Applying Laplace's rule along the diagonal, we obtain for $j = \tau(\alpha)$

$$\mathbf{K}_{jj} = \sum_{n=1}^{N} \langle \partial_{x_n} \varphi_\alpha(\mathbf{x}), \partial_{x_n} \varphi_\alpha(\mathbf{x}) \rangle = \sum_{n=1}^{N} K_{\alpha_n \alpha_n}.$$

2. $\alpha \neq \beta$, and α and β have at most $N-2$ elements in common. In this case, for every $n \in \{1, \dots, N\}$, there is a column of $K^n_{\alpha,\beta}$ which only contains zeros (specifically, the column $m \neq n$ for which $\beta_m \notin \alpha$). Thus, all determinants are zero, and the entry for $j = \tau(\alpha)$ and $k = \tau(\beta)$ is

$$\mathbf{K}_{jk} = 0.$$

3. $\alpha \neq \beta$, and α and β have exactly $N-1$ elements in common. In this case, there is an $n \in \{1, \dots, N\}$ such that $\beta_n \notin \alpha$ and $\beta_i \in \alpha$ for all $i \neq n$. Conversely, there is exactly one $m \in \{1, \dots, N\}$ such that $\alpha_m \notin \beta$. Arguing as in case 2, we thus have that the determinant of $K^i_{\alpha,\beta}$ is zero for all $i \neq n$. Consider now $K^n_{\alpha,\beta}$. Since $\beta_i \in \alpha$ holds for all $i \neq n$, every column except the n-th contains exactly one nonzero element. Therefore, there exists a row – specifically, the m-th row, since $\delta_{\alpha_m \beta_i} = 0$ for all $i \neq n$ – where every element except the one in the n-th row is zero. Applying Laplace's rule to this row yields a minor which is the $(N-1) \times (N-1)$ identity matrix, since the remaining elements in α and β are identical and equally ordered by construction. We thus obtain for $j = \tau(\alpha)$ and $k = \tau(\beta)$

$$\mathbf{K}_{jk} = \sum_{i=1}^{N} \det(K^i_{\alpha,\beta}) = \det(K^n_{\alpha,\beta}) = (-1)^{m+n} K_{\alpha_m \beta_n}.$$

Example 3.2. *Returning to Example 3.1 for* $N = 3$ *and* $p = 4$*, the stiffness matrix is the sum of three* $N \times N$ *matrices. Consider the tuples* $\alpha = (1,3,4)$ *and* $\beta = (2,3,4)$ *with* $\tau(\alpha) = 3$ *and* $\tau(\beta) = 4$*. Then,*

$$\begin{aligned}
\mathbf{K}_{34} &= \langle \partial_{x_1}\varphi_\alpha(\mathbf{x}), \partial_{x_1}\varphi_\beta(\mathbf{x})\rangle + \langle \partial_{x_2}\varphi_\alpha(\mathbf{x}), \partial_{x_2}\varphi_\beta(\mathbf{x})\rangle + \langle \partial_{x_3}\varphi_\alpha(\mathbf{x}), \partial_{x_3}\varphi_\beta(\mathbf{x})\rangle \\
&= \begin{vmatrix} \langle \ell_1', \ell_2'\rangle & \langle \ell_1, \ell_3\rangle & \langle \ell_1, \ell_4\rangle \\ \langle \ell_3', \ell_2'\rangle & \langle \ell_3, \ell_3\rangle & \langle \ell_3, \ell_4\rangle \\ \langle \ell_4', \ell_2'\rangle & \langle \ell_4, \ell_3\rangle & \langle \ell_4, \ell_4\rangle \end{vmatrix} + \begin{vmatrix} \langle \ell_1, \ell_2\rangle & \langle \ell_1', \ell_3'\rangle & \langle \ell_1, \ell_4\rangle \\ \langle \ell_3, \ell_2\rangle & \langle \ell_3', \ell_3'\rangle & \langle \ell_3, \ell_4\rangle \\ \langle \ell_4, \ell_2\rangle & \langle \ell_4', \ell_3'\rangle & \langle \ell_4, \ell_4\rangle \end{vmatrix} + \begin{vmatrix} \langle \ell_1, \ell_2\rangle & \langle \ell_1, \ell_3\rangle & \langle \ell_1', \ell_4'\rangle \\ \langle \ell_3, \ell_2\rangle & \langle \ell_3, \ell_3\rangle & \langle \ell_3', \ell_4'\rangle \\ \langle \ell_4, \ell_2\rangle & \langle \ell_4, \ell_3\rangle & \langle \ell_4', \ell_4'\rangle \end{vmatrix} \\
&= \begin{vmatrix} K_{12} & 0 & 0 \\ K_{32} & 1 & 0 \\ K_{42} & 0 & 1 \end{vmatrix} + \begin{vmatrix} 0 & K_{13} & 0 \\ 0 & K_{33} & 0 \\ 0 & K_{43} & 1 \end{vmatrix} + \begin{vmatrix} 0 & 0 & K_{14} \\ 0 & 1 & K_{34} \\ 0 & 0 & K_{44} \end{vmatrix} = K_{12}.
\end{aligned}$$

This corresponds to case 3 above with $n = m = 1$*,* $\alpha_1 = 1 \notin \beta$ *and* $\beta_1 = 2 \notin \alpha$*.*

4 ASSEMBLY OF THE DISCRETE HAMILTONIAN

In this section, we present our approach for the efficient assembly of the Galerkin matrices. The procedure for the (diagonal) potential matrices and the computation of the diagonal entries of the stiffness matrix is straightforward, so we only need to discuss the computation of the off-diagonal entries $\mathbf{K}_{jk}$, $j \neq k$, of the stiffness matrix. We first introduce and justify the abstract algorithm in section 4.1. In section 4.2, we discuss its implementation in MATLAB.

4.1 ABSTRACT ALGORITHM

Since N_p grows factorially with increasing N and p, a naive evaluation of all $N_p \times N_p$ possible inner products of the form (3.4) is prohibitive. The crucial step in our approach is therefore a procedure to compute only the nonzero entries $\mathbf{K}_{jk}$ together with the corresponding indices j and k. As discussed above, the off-diagonal entries are precisely those corresponding to tuples α and β in $\mathcal{T}$ which have exactly $N-1$ elements in common. For the sake of brevity, we call a $\beta \in \mathcal{T}$ satisfying this condition *connected* to α.[1] Due to the symmetry of the stiffness matrix, it suffices to consider only connected tuples which satisfy $\beta \succ \alpha$; the set of all such tuples will be denoted by $\mathcal{I}_\alpha$.

The set of all connected tuples can in principle be computed by successively removing an element α_m from α and inserting all possible $\beta_n \notin \alpha$ while ensuring that the result remains strictly ascending. The main algorithmic difficulty here is that the differing value need not be in the same position in α and in β (e. g., $(1,2,4)$ and $(2,3,4)$), and that we need to ensure $\beta \succ \alpha$ to avoid redundancy. A more efficient approach is to change the order of operations by first inserting the new element (yielding an $(N+1)$-tuple) and then removing

[1] This notation is motivated by the fact that $\mathcal{T}$ can be identified with the set of all subsets of $\{1,\dots,p\}$ having exactly N elements. The graph obtained from these vertices with edges between all vertices corresponding to subsets having $N-1$ elements in common is known as the *Johnson graph*, cf. [13].

all possible elements such that the result is a connected tuple with $\beta \succ \alpha$. In this way, all $\beta \in \mathcal{I}_\alpha$ can be generated without conditional statements. The procedure for a given $\alpha \in \mathcal{T}$ can be summarized as follows (setting $\alpha_{N+1} = p + 1$):

for $n = 1, \ldots, N$ **do**
  for $b \in \{\alpha_n + 1, \ldots, \alpha_{n+1} - 1\}$ **do**
    $\beta' \leftarrow (\alpha_1, \ldots, \alpha_n, b, \alpha_{n+1}, \ldots, \alpha_N)$
    for $m = 1, \ldots, n$ **do**
      $\beta \leftarrow (\beta'_1, \ldots, \beta'_{m-1}, \beta'_{m+1}, \ldots, \beta'_N)$
      $\mathcal{I}_\alpha \leftarrow \mathcal{I}_\alpha \cup \{\beta\}$
    end for
  end for
end for

Note that the set in the second loop can be empty, and that it is possible to restrict the first loop a priori to only those values of n for which this set is nonempty. Furthermore, knowing β, n and m, we can directly compute $\mathbf{K}_{\tau(\alpha)\tau(\beta)} = (-1)^{m+n} K_{\alpha_m \beta_n}$ without additional effort. In addition, β need not be saved after calculating $\tau(\beta)$, since $\beta_n = b$ by construction.

It remains to argue that this procedure generates every $\beta \in \mathcal{I}_\alpha$ exactly once. Let $\alpha \in \mathcal{T}$ be given and β be generated as above for $n \in \{1, \ldots, N\}$ and $m \in \{1, \ldots, n\}$.

First, since $\alpha_n < b < \alpha_{n+1}$ holds, β' is a strictly ascending $(N+1)$-tuple. Deleting any element then results in a strictly ascending N-tuple. Hence, $\beta \in \mathcal{T}$ holds.

Observe now that by construction, α and β have exactly $N - 1$ elements in common (specifically, α_i for all $i \neq m$), and that $\alpha_m = \beta'_m \notin \beta$ and $\beta_n = b \notin \alpha$ since $m \leqslant n$. Furthermore, either $\beta_m = b > \alpha_m$ (if $m = n$) or $\beta_m = \alpha_{m+1} > \alpha_m$ (else) is satisfied. Since $\beta_i = \alpha_i$ for $i < m$, we have that $\beta \succ \alpha$ and hence $\beta \in \mathcal{I}_\alpha$. Conversely, for every β connected to α, there exist $m, n \in \{1, \ldots, N\}$ with $\alpha_m \notin \beta$ and $\beta_n \notin \alpha$. If $m > n$, then $\beta_i = \alpha_i$ for all $i < n$ and $\beta_{n+1} = \alpha_n$ (otherwise $\alpha_n \neq \alpha_m$ cannot be an element of β, a contradiction to $\beta \in \mathcal{I}_\alpha$). Since α and β are strictly ascending tuples, we have that $\beta_n < \beta_{n+1} = \alpha_n$ holds, which implies $\beta \prec \alpha$, so $\beta \notin \mathcal{I}_\alpha$. We can therefore assume $m \leqslant n$. If $m < n$, we have

$$\alpha_n = \beta_{n-1} < \beta_n < \beta_{n+1} = \alpha_{n+1}$$

and hence $b = \beta_n$ appears in the second loop. On the other hand, $m = n$ implies $\beta_{n-1} = \alpha_{n-1}$ and thus $\alpha_{n-1} < \beta_n \leqslant \alpha_n$ is possible (otherwise we argue as above that $b = \beta_n$ occurs). However, this would imply that $\beta \preceq \alpha$ (since $\beta_i = \alpha_i$ for $i < n$), and thus again $\beta \notin \mathcal{I}_\alpha$. Hence, any $\beta \in \mathcal{I}_\alpha$ will be generated by the above procedure. Finally, for $m \leqslant n$, the connected tuple β is uniquely determined by the pair (m, n) and the value $\beta_n = b$, and therefore each $\beta \in \mathcal{I}_\alpha$ occurs only once in the procedure.

Example 4.1. *Consider the case* $N = 4$ *and* $p = 7$ *and take the tuple* $\alpha = (1, 2, 4, 5)$. *The only possible points of insertion are to the right of index* $n = 2$ *and* $n = 4$. *For* $n = 2$, *we can only insert* $b = 3$ *to obtain a strictly ascending tuple* $\beta' = (1, 2, 3, 4, 5)$. *By successively deleting* β'_1 *and* β'_2, *we obtain* $(2, 3, 4, 5)$ *and* $(1, 3, 4, 5)$. *For* $n = 4$, *we can insert* $b = 6$ *and* $b = 7$ *after* α_4, *so that after deletion of* β'_m, $1 \leqslant m \leqslant n$, *we obtain*

$$(2, 4, 5, 6), (1, 4, 5, 6), (1, 2, 5, 6), (1, 2, 4, 6)$$

and

$$(2,4,5,7),(1,4,5,7),(1,2,5,7),(1,2,4,7),$$

respectively. It is easy to verify that these are indeed all connected tuples which are lexicographically greater than α*.*

The complete procedure to generate the off-diagonal entries of the stiffness matrix is given as Algorithm 1, where we again set $\alpha_{N+1} = p+1$. (Note that $m = n$ is possible, in which case $\beta_{m-1} = \alpha_{m-1}$ and $\beta_m = b$ hold.) This algorithm can be further accelerated by precomputing for each α the set G_α of possible insertion indices n and the range of valid b for such n.

Algorithm 1 Generating off-diagonal entries of stiffness matrix

for $\alpha \in \mathcal{T}$ **do**
  Compute set $G_\alpha = \{n : \alpha_{n+1} > \alpha_n + 1\}$
  for $n \in G_\alpha$ **do**
    for $b \in \{\alpha_n + 1, \dots, \alpha_{n+1} - 1\}$ **do**
      for $m = 1, \dots, n$ **do**
        $\beta \leftarrow (\alpha_1, \dots, \alpha_{m-1}, \alpha_{m+1}, \dots, \alpha_n, b, \alpha_{n+1}, \dots, \alpha_N)$
        $\mathbf{K}_{\tau(\alpha)\tau(\beta)} \leftarrow (-1)^{m+n} K_{\alpha_m b}$
      end for
    end for
  end for
end for

4.2 IMPLEMENTATION

We now discuss specific details of the MATLAB implementation of our approach, which is given in Appendix A.

The one-dimensional Legendre–Gauss–Lobatto nodes and weights are computed from their recursion coefficients [16]. Specifically, the nodes are the ordered eigenvalues of the matrix

$$J = \begin{pmatrix} 0 & b_1 & & & \\ b_1 & 0 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_p & 0 & b_{p+1} \\ & & & b_{p+1} & 0 \end{pmatrix}, \qquad b_k = \begin{cases} \frac{k}{\sqrt{4k^2-1}} & k = 1, \dots, p, \\ \sqrt{\frac{p+1}{2p+1}} & k = p+1. \end{cases}$$

Computing the eigenvalue decomposition $J = VXV^\top$ (with the eigenvectors ordered corresponding to increasing eigenvalue), we obtain the Lobatto nodes as $\xi_j = X_{jj}$ and the corresponding weights as $w_j = (V_{1j})^2$. The stiffness matrix K is computed using an efficient implementation of the barycentric formulation [3]. In the periodic case, K can be computed

using the discrete Fourier transform (DFT) as $K = FDF^{-1}$, where F is the DFT matrix and D is a diagonal matrix with entries

$$D_{jj} = \begin{cases} (j-1)^2 & j < \frac{p}{2} + 1, \\ (p-j)^2 & j \geqslant \frac{p}{2}, \end{cases}$$

which computes the second derivative in the coefficient representation (see function `computeLGL` in the code).

The lexicographically ordered tuples in $\mathcal{T}$ are represented as a matrix $T \in \mathbb{N}^{N_p \times N}$, where each row contains an ascending non-repeating N-tuple, such that the j-th row of T corresponds to $\tau^{-1}(j)$ and $\alpha_i = T_{\tau(\alpha),i}$. This matrix can be generated in MATLAB with the single command `T = nchoosek(1:p,N)`. The confinement and interaction potential matrices $\mathbf{U}$ and $\mathbf{V}$ can then be easily assembled using the `sparse` command and MATLAB's sparse indexing (see line 19–21, noting that `xi(T(i,j)) =` ξ_{α_j} if $i = \tau(\alpha)$). For $\mathbf{V}$, we use again `nchoosek` to generate a matrix containing all pairs of interacting particles (see line 6) and `arrayfun` to parallelize the evaluation of the interaction potential at the quadrature nodes.

Using T, it is also straightforward to compute the matrix representation of $\{G_\alpha : \alpha \in \mathcal{T}\}$, which has the entries

$$G_{jk} = g_j(\tau^{-1}(j)), \qquad g_n(\alpha) = \begin{cases} \alpha_{n+1} - \alpha_n - 1 & \text{for } n = 1 \ldots N-1, \\ p - \alpha_N & \text{for } n = N, \end{cases}$$

by taking the difference of consecutive rows of a suitably augmented matrix T (see lines 10–11).

Given m, n and the inserted value b, the value of the stiffness matrix entry corresponding to the connected tuples α and β can be easily obtained from the one-dimensional stiffness matrix (see line 41). By looping over all rows of T, we consider each $\alpha \in \mathcal{T}$ in ascending lexicographical order, and thus never need to explicitly compute the column index $j = \tau(\alpha)$. It remains to find the column index $k = \tau(\beta)$ of the entry $\mathbf{K}_{jk}$. While it is possible to construct an explicit formula for $\tau(\beta)$, this would be needlessly expensive. Similarly, using MATLAB's `find` and the matrix T would lead to poor scaling of the algorithm. We thus proceed in two steps: First, we assign a unique integer (a *hash*) to each β via the mapping

$$\tau^{\#} : \mathcal{T} \to \{1, \ldots, N_p(p-N)^{N-1}\}, \quad \beta \mapsto \sum_{j=1}^{N} \beta_j (p-N)^{j-1}.$$

This is a linear mapping from $\mathbb{N}^N$ to $\mathbb{N}$ and can thus be represented as a $1 \times N$ matrix h (called `hash` in the code, see line 14). We then construct a lookup table of size $N_T := N_p(p-N)^{N-1}$ which contains for every $i \in \{1, \ldots, N_T\}$ either the value of $\tau(\beta)$ if $i = \tau^{\#}(\beta)$ or zero if i is not in the range of $\tau^{\#}$. Since only N_p elements are nonzero, it can be efficiently represented in MATLAB as a sparse matrix (called `lookup` in the code) and quickly constructed using the `sparse` command (see line 16). For given β, the column index $k = \tau(\beta)$ of the corresponding stiffness matrix entry can then be obtained cheaply by computing `dex` $= \beta h$ (with β interpreted as a row vector in $\mathbb{N}^N$) and setting $k =$ `lookup(dex)` (see lines 38 and 40).

It is possible to accelerate the computation further by making use of MATLAB's vectorization capability. Given α, n and m, we can compute a vector b of possible values to insert after the index n by setting $b = (\alpha_n + 1, \ldots, \alpha_{n-1})^T \in \mathbb{N}^s$. We then consider the block of all possible $\beta \in \mathcal{I}_\alpha$ generated for these values of n and m as a matrix $B \in \mathbb{N}^{s \times N}$ with entries

$$B_{jk} = \begin{cases} \alpha_k & \text{if } k < m \text{ or } k > n, \\ \alpha_{k+1} & \text{if } m \leqslant k < n, \\ b_j & \text{if } k = n. \end{cases}$$

The corresponding hash values $\tau^\#(b_k)$ are then given by the $(s \times N)$ by $(N \times 1)$ matrix-vector product Bh. Since for all $k \neq n$, we have $B_{ik} = B_{jk}$ for all $1 \leqslant i, j \leqslant s$, B is a rank-two matrix and can be written as

$$B = \mathbf{1} B_1 (I - e_n e_n^T) + b e_n^T,$$

where $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{N}^N$, B_1 is the first row of B, I is the identity matrix, and e_n is the n-th canonical unit vector. Hence, the matrix product can be computed as the sum of two terms:

$$Bh = \mathbf{1} B_1 (I - e_n e_n^T) h + b e_n^T h = \mathbf{1} (B_1^{[n]} h^{[n]}) + b h_n,$$

where the superscript $[n]$ denotes deletion of the n-th element. Here, the whole first term and the vector h_n in the second term are independent of m and b and can thus be precomputed (see lines 33 and 38). Additionally, since the elements of B needed for the first term are known elements of α, it is in fact not necessary to construct B. Rather, T can be used for this purpose, using a precomputed set of masks `rInd` which contain the indices $1, \ldots, m-1, m+1, \ldots, N$ for each m (see line 13).

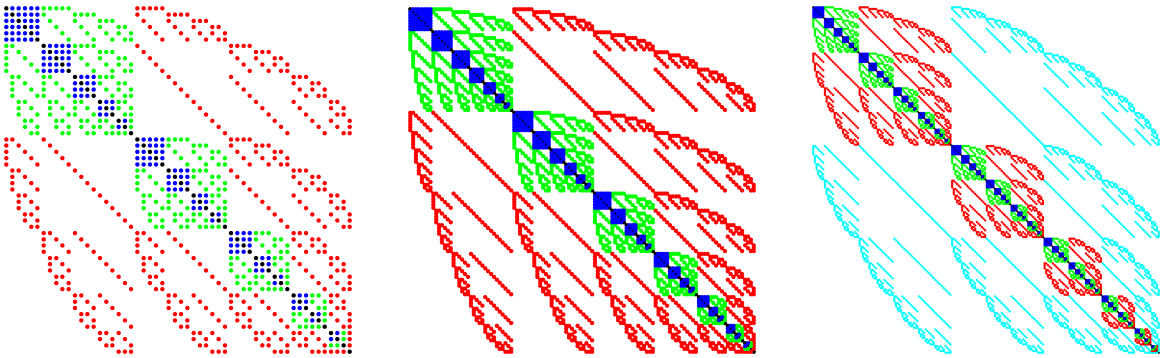
Finally, since n is fixed and m is incremented by one in each loop, the sign of the stiffness matrix entry alternates, starting with $(-1)^{n+1}$. This vector can be precomputed as well (see line 35).

5 NUMERICAL EXPERIMENTS

All computations were performed on an eight-core Intel Xeon X5560 (2.8 GHz) workstation with 24 GBytes of RAM using MATLAB R2011a.

Figure 2 shows the sparsity pattern of the stiffness matrix corresponding to the chosen discretization, where the elements $\mathbf{K}_{jk}$ from each dimension x_i is color coded depending on the removal index m in the tuple $\alpha = \tau^{-1}(j)$ connected to $\beta = \tau^{-1}(k)$. Note the recursive structure of the sparsity pattern, where the pattern for $N = 4$ contains copies of the pattern for $N = 3$ and decreasing p as diagonal blocks.

We next address the accuracy of the spectral discretization by comparing computed eigenvalues of the stiffness matrix (with homogeneous Dirichlet conditions) to the exact eigenvalues.



(a) $N = 3, p = 8$ (b) $N = 3, p = 10$ (c) $N = 4, p = 10$

Figure 2: Sparsity pattern of stiffness matrix for different problem parameters. Shown are the nonzero entries, where the contribution from different dimensions is color coded (blue: x_1, green: x_2, red: x_3, cyan: x_4).

Each eigenvalue for N non-interacting particles corresponding to an eigenfunction ψ_α is constructed from single-particle eigenvalues [8, § 20.5] via

$$\lambda_\alpha = \frac{\pi^2}{4} \sum_{j=1}^{N} \alpha_j^2.$$

Specifically, the first two eigenvalues (as sorted by ascending magnitude) are

$$\lambda_1 = \lambda_{(1,\dots,p)} = \frac{\pi^2}{4} \sum_{i=1}^{N} i^2 = \frac{\pi^2}{24} N(N+1)(2N+1),$$

$$\lambda_2 = \lambda_{(1,\dots,p-1,p+1)} = \frac{\pi^2}{4} \sum_{i=1}^{N-1} i^2 + (N+1)^2 = \frac{\pi^2}{24} (2N+1)(N^2+N+6).$$

These values are compared with the eigenvalues of the stiffness matrix obtained by the command

```
K = assembleFermiMatrix(p,N,@(x) 0,@(x) 0,'dirichlet');
```

which are computed by a Krylov method using MATLAB's built-in `eigs`. Figure 3 shows the difference between the exact and computed first two eigenvalues for $N = 2, 3, 4$ as a function of $p \in \{2N, \dots, 25\}$. Since the multi-particle eigenstates are Slater determinants of different single-particle eigenstates, the resolution of the multi-particle eigenstate is completely determined by the resolution of all component eigenstates. For example, the first eigenstate for $N = 3$ particles corresponds to the tuple $(1, 2, 3)$, while the second eigenstate for $N = 2$ particles corresponds to $(1, 3)$. Hence, the error in both cases is dominated by the resolution of the third single-particle eigenfunction ψ_3. In practice, this can be exploited for determining the value of p which will sufficiently resolve the highest appearing eigenfunction by investigating the single-particle case only.

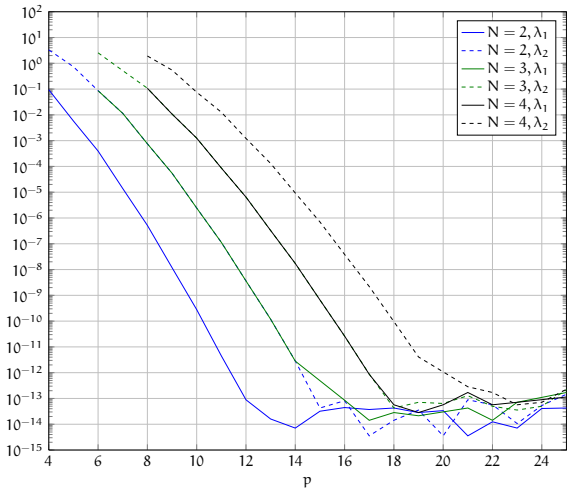


Figure 3: Error in first two eigenvalues of Laplacian (discrete vs. exact) for N particles and p basis functions per particle.

Since to the best of our knowledge, no exact solutions of the multi-particle problem with globally smooth interaction potentials are known, we test the accuracy in the presence of interaction potentials by plotting the difference between the computed eigenvalues and those computed for the finest discretization. The order of convergence can be observed when the difference between the low order and high order approximations is sufficiently large. Figure 4a shows the convergence history for the interaction potential $V(x_j, x_k) = \cos(x_j - x_k)$, no confinement potential and periodic boundary conditions, corresponding to the command

```
[K,U,V] = assembleFermiMatrix(p,N,@(x) 0,@(x) cos(x(1,:)-x(2,:),'periodic');
```

For $N = 2$, this potential models the interaction between two ground state electrons in a helium atom (see, e. g., [10, § 148]). Figure 4b illustrates the convergence for an attractive Gaussian effective potential $V(x_j, x_k) = -\exp(-10(x_j - x_k)^2)$ (cf. [22, 21]), confinement potential $U(x_j) = x_j$ and homogeneous Dirichlet conditions. The matrices can be obtained with the command

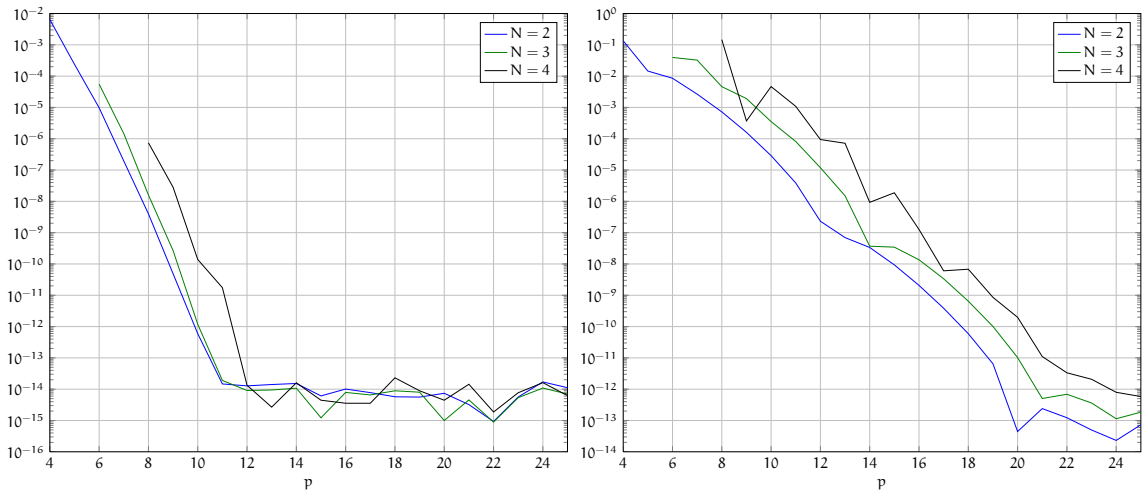
```
[K,U,V] = assembleFermiMatrix(p,N,@(x) x,@(x) ...
    -exp(-10*(x(1,:)-x(2,:))).^2,'dirichlet');
```

When the variable coefficients are smooth functions, spectral methods exhibit exponential convergence, and this can be observed here.

The convergence for the Coulomb potential $V(x_j, x_k) = |x_j - x_k|^{-1}$ with homogeneous Dirichlet conditions and confinement potential $U(x_j) = x_j$ is shown in Figure 5, corresponding to the command

```
[K,U,V] = assembleFermiMatrix(p,N,@(x) x,@(x) 1./abs(x(1,:)-x(2,:)), ...
    'dirichlet');
```

In this case, the variable coefficient is not even bounded, so only algebraic convergence can be expected (cf. [6, Chap. 2]). In this case, the convergence order is quadratic, and thus at least equal to orders that can be achieved, e. g., with second order finite differences.



(a) $V(x_j, x_k) = \cos(x_j - x_k)$ (b) $V(x_j, x_k) = -\exp(-10(x_j - x_k)^2)$

Figure 4: Error in first eigenvalue of Hamiltonian for smooth interaction potentials (discrete vs. finest discretization) for N particles and p basis functions per particle.

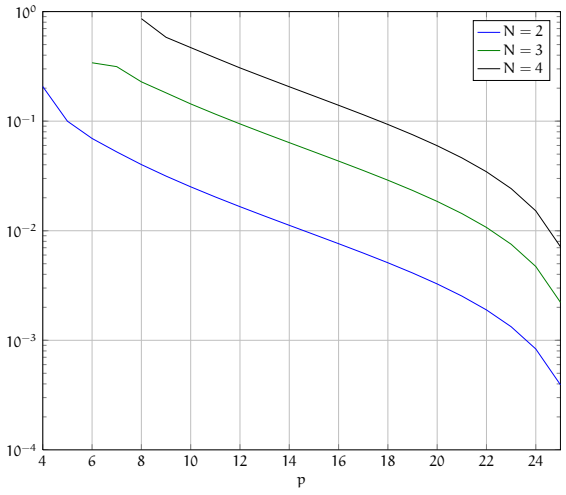


Figure 5: Error in first eigenvalue of Hamiltonian for Coulombic interaction potential (discrete vs. finest discretization) for N particles and p basis functions per particle.

Table 1: Time for assembly of Hamiltonian depending on number of particles N. For each value of N and $p = 3N$, the number N_p of basis functions, the number N_K of nonzero entries in the stiffness matrix and the complete assembly time (in seconds, average of 10) are shown.

N	2	3	4	5	6	7	8
N_P	15	84	495	3003	18564	116280	735471
N_K	135	1596	16335	153153	1355172	11511720	94875759
t	0.007	0.009	0.068	0.586	5.125	41.965	336.840

Finally, the efficiency of our program is indicated by Table 1, which shows the total assembly time of the Coulombic Hamiltonian (averaged over ten runs) as a function of the number of particles. We fix $p = 3N$ to ensure an accuracy of at least 10^{-3} in the eigenvalues of the stiffness matrix for all considered N (cf. Figure 3). Besides the assembly time, we also show the number N_p of degrees of freedom and the number N_K of nonzero entries of the stiffness matrix. As can be seen, the assembly time scales roughly linearly (disregarding cache effects for small N) with the number of nonzero entries, which is the optimal rate to be expected.

6 CONCLUSION

We have presented a general framework for simulating systems of one-dimensional interacting fermions based on a nodal spectral Galerkin method together with an efficient MATLAB code for the assembly of the discretized Hamiltonian. The accuracy of the discretization and the performance of the assembly method was illustrated by computing the energy levels of several systems with different potentials. When a smooth interaction potential is a reasonable model, the method exhibits exponential convergence. The Coulomb potential, however, has a singularity, and the low regularity causes the method to converge only quadratically. A suitable change of variables to desingularize the problem and regain spectral accuracy is the subject of future work. In principle, the proposed approach can be extended to two and three spatial dimensions, even though such problems are not feasible on current desktop machines. Nevertheless, we expect that the spectral discretization will be even more useful in the pursuit of this goal.

REFERENCES

[1] P. AMORE AND F. M. FERNÁNDEZ, *One-dimensional oscillator in a box*, European Journal of Physics, 31 (2010), p. 69.

[2] O. M. AUSLAENDER, H. STEINBERG, A. YACOBY, Y. TSERKOVNYAK, B. I. HALPERIN, K. W. BALDWIN, L. N. PFEIFFER, AND K. W. WEST, *Spin-charge separation and localization in one dimension*, Science, 308 (2005), pp. 88–92.

[3] J.-P. BERRUT AND L. N. TREFETHEN, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517.

[4] G. BEYLKIN, M. J. MOHLEMKAMP, AND F. PÉREZ, *Approximating a wavefunction as an unconstrained sum of Slater determinants*, J. Math. Phys., 49 (2008), pp. 032107, 28.

[5] G. BEYLKIN AND M. J. MOHLENKAMP, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[6] J. P. BOYD, *Chebyshev and Fourier Spectral Methods*, Dover Publications Inc., Mineola, NY, second ed., 2001.

[7] C. G. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods*, Springer, Heidelberg, 2007.

[8] A. Z. Capri, *Nonrelativistic Quantum Mechanics*, World Scientific Publishing Co. Inc., River Edge, NJ, third ed., 2002.

[9] S. Edvardsson, D. Åberg, and P. Uddholm, *A program for accurate solutions of two-electron atoms*, Computer Physics Communications, 165 (2005), pp. 260–270.

[10] S. Flügge, *Practical Quantum Mechanics*, Springer-Verlag, Berlin, 1999.

[11] M. Griebel and J. Hamaekers, *Sparse grids for the Schrödinger equation*, M2AN Math. Model. Numer. Anal., 41 (2007), pp. 215–247.

[12] W. Hackbusch, *The efficient computation of certain determinants arising in the treatment of Schrödinger's equations*, Computing, 67 (2001), pp. 35–56.

[13] A. Hora and N. Obata, *Quantum Probability and Spectral Analysis of Graphs*, Springer, Berlin, 2007.

[14] Y. Jompol, C. J. B. Ford, J. P. Griffiths, I. Farrer, G. A. C. Jones, D. Anderson, D. A. Ritchie, T. W. Silk, and A. J. Schofield, *Probing spin-charge separation in a Tomonaga–Luttinger liquid*, Science, 325 (2009), pp. 597–601.

[15] H. Kohler, *Exact diagonalization of 1d interacting spinless fermions*, Journal of Mathematical Physics, 52 (2011), p. 032107.

[16] D. P. Laurie, *Computation of Gauss-type quadrature formulas*, Journal of Computational and Applied Mathematics, 127 (2001), pp. 201–217.

[17] P.-O. Löwdin, *Quantum theory of many-particle systems. I. Physical interpretations by means of density matrices, natural spin-orbitals, and convergence problems in the method of configurational interaction*, Phys. Rev., 97 (1955), pp. 1474–1489.

[18] R. Pauncz, *The Symmetric Group in Quantum Chemistry*, CRC Press, 1995.

[19] R. Saito, G. Dresselhaus, and M. S. Dresselhaus, *Physical Properties of Carbon Nanotubes*, World Scientific Publishing, 1998.

[20] F. Schwabl, *Advanced Quantum Mechanics*, Springer Verlag, 2008.

[21] P. M. Stevenson, *Gaussian effective potential: Quantum mechanics*, Phys. Rev. D, 30 (1984), pp. 1712–1726.

[22] J. von Stecher, C. H. Greene, and D. Blume, *BEC-BCS crossover of a trapped two-component Fermi gas with unequal masses*, Phys. Rev. A, 76 (2007), p. 053613.

[23] T. F. Xie and S. P. Zhou, *On approximation by trigonometric Lagrange interpolating polynomials*, Bull. Austral. Math. Soc., 40 (1989), pp. 425–428.

[24] X. Yin, Y. Hao, S. Chen, and Y. Zhang, *Ground-state properties of a few-boson system in a one-dimensional hard-wall split potential*, Phys. Rev. A, 78 (2008), p. 013604.

[25] H. Yserentant, *Sparse grid spaces for the numerical solution of the electronic Schrödinger equation*, Numer. Math., 101 (2005), pp. 381–389.

A MATLAB CODE[2]

```
function [K,U,V] = assembleFermiMatrix(p,N,Ufun,Vfun,bc)
%% Precompute
[xi,K1d] = computeLGL(p,bc); % 1D Legendre-Gauss-Lobatto quadrature points
diagK1d = diag(K1d);      % diagonal elements of 1D stiffness matrix

d  = nchoosek(1:N,2);    % all pairs of interacting particles
T  = nchoosek(1:p,N);    % matrix of nD basis elements tuples
Np = size(T,1);          % number of nD basis elements
Nk = NpN/2(p-N);         % number of nonzero off-diagonal entries in K
Ta = [T ones(Np,1)(p+1)]; % precompute: augment T to account for G_N
G  = diff(Ta,1,2)-1;     % precompute: length of gaps in alpha
vN = 1:Np;               % precompute: vector (1,...,Np)
rInd = triu(ones(N,N-1))+ones(N,1)(1:(N-1)); % indices after removal of m
hash = (p-N).^(N-1:-1:0)'; % assign unique id to each basis element
dict = T  hash;          % dictionary for fast lookup of basis elements
lookup = sparse(dict,ones(Np,1),vN,dict(Np),1); % lookup via sparse index

%% Compute potential terms
Vc = arrayfun(@(j)Vfun(xi(T(:,d(j,:)))),1:size(d,1),'UniformOutput',false);
V  = sparse(vN,vN,sum(cat(2,Vc{:}),2),Np,Np); % interaction potential
U  = sparse(vN,vN,sum(Ufun(xi(T)),2),Np,Np); % confinement potential

%% Compute stiffness matrix
row = zeros(Nk,1);            % preallocate vector of row indices of K
col = zeros(Nk,1);            % preallocate vector of column indices
val = zeros(Nk,1);            % preallocate vector of entries
indstart = 0;                 % running index for above vectors

for j = 1:(Np-1)              % loop over all tuples: j = tau(alpha)
    for n = vN(G(j,:)>0)      % insert at all indices n in G_alpha
        bLength = 1:G(j,n);   % number of possible values for b (vector)
```

[2] If your PDF viewer supports file annotations, you can extract the code by clicking here. It can also be downloaded from http://www.uni-graz.at/~clason/codes/assembleFermiMatrix.m.

```
        b = T(j,n) + bLength;  % values to be inserted at beta_n
        hash1 = hash(rInd(n,:)); % common part of hash for beta_i, i != n
        hash2 = hash(n)b;      % beta_n = b
        sgn = (-1).^(n+1:-1:2); % sign of K_jk: (-1)^(m+n)
        for m = 1:n            % remove at all indices m <= n
            ind = indstart + bLength;       % index of entries to set
            dex = T(j,rInd(m,:))hash1 + hash2; % id of beta
            row(ind) = j;                   % j = tau(alpha)
            col(ind) = lookup(dex);         % k = tau(beta)
            val(ind) = sgn(m)K1d(T(j,m),b); % K_jk
            indstart = ind(end);            % increment running index
        end
    end
end

Ko = sparse(row,col,val,Np,Np);             % off-diagonal entries of K
Kd = sparse(vN,vN,sum(diagK1d(T),2),Np,Np); % diagonal entries of K
K  = Kd + Ko + Ko';                         % stiffness matrix
% end function fermion_setup

function [xi,K] = computeLGL(p,bc)
switch lower(bc)
    case 'periodic'
        xi = 2pi(0:p-1)'/p;
        K  = fft(ifft(diag([0:floor(p/2) floor((p-1)/2:-1:1)].^2)).');
    case 'dirichlet'
        p2 = p+2;
        b  = sqrt([(1:p).^2./(4(1:p).^2-1) (p+1)/(2p+1)])';
        [V, X] = eig(diag(b,1) + diag(b,-1));
        [x, ord] = sort(diag(X));
        V  = V(:,ord)';
        w  = 2V(:,1).^2;
        xi = x(2:end-1);
        X  = x ones(1,p2);
        Xdiff = X-X'+eye(p2);
        W = (1./prod(Xdiff,2))ones(1,p2);
        D = W./(W'.Xdiff);
        D(1:(p2+1):(p2^2)) = 1-sum(D);
        D  = -D'diag(1./sqrt(w));
        Di = D(:,2:end-1);
        K  = Di'diag(w)Di;
    otherwise
        error(['Boundary condition "' bc '" is not implemented.'])
end
% end function computeLGL
```