Pymablock: An algorithm and a package for quasi-degenerate perturbation theory

Isidora Araya Day[1,2⋆], Sebastian Miles[1], Hugo K. Kerstens[2], Daniel Varjas[3,4] and Anton R. Akhmerov[2†]

1 QuTech, Delft University of Technology, 2600 GA Delft, The Netherlands
2 Kavli Institute of Nanoscience, Delft University of Technology, 2600 GA Delft, The Netherlands
3 Max Planck Institute for the Physics of Complex Systems, Nöthnitzer Strasse 38, 01187 Dresden, Germany
4 Institute for Theoretical Solid State Physics, IFW Dresden and Würzburg-Dresden Cluster of Excellence ct.qmat, Helmholtzstr. 20, 01069 Dresden, Germany

⋆ iarayaday@gmail.com , † pymablock@antonakhmerov.org

Abstract

A common technique in the study of complex quantum-mechanical systems is to reduce the number of degrees of freedom in the Hamiltonian by using quasi-degenerate perturbation theory. While the Schrieffer–Wolff transformation achieves this and constructs an effective Hamiltonian, its scaling is suboptimal, it is limited to two subspaces, and implementing it efficiently is both challenging and error-prone. We introduce an algorithm for constructing an equivalent effective Hamiltonian as well as a Python package, Pymablock, that implements it. Our algorithm combines an optimal asymptotic scaling and the ability to handle any number of subspaces with a range of other improvements. The package supports numerical and analytical calculations of any order and it is designed to be interoperable with any other packages for specifying the Hamiltonian. We demonstrate how the package handles constructing a k.p model, analyses a superconducting qubit, and computes the low-energy spectrum of a large tight-binding model. We also compare its performance with reference calculations and demonstrate its efficiency.

Copyright I. Araya Day *et al*.
This work is licensed under the Creative Commons Attribution 4.0 International License.
Published by the SciPost Foundation.

Received 2024-06-12
Accepted 2025-01-21
Published 2025-02-12
doi:10.21468/SciPostPhysCodeb.50

Check for updates

This publication is part of a bundle: Please cite both the article and the release you used.

DOI	Type
doi:10.21468/SciPostPhysCodeb.50	Article
doi:10.21468/SciPostPhysCodeb.50-r2.1	Codebase release

Contents

1 Introduction

Effective models enable the study of complex quantum systems by reducing the dimensionality of the Hilbert space. Their construction separates the low and high-energy subspaces by block-diagonalizing a perturbed Hamiltonian

$$\mathcal{H} = \begin{pmatrix} H_0^{AA} & 0 \\ 0 & H_0^{BB} \end{pmatrix} + \mathcal{H}', \tag{1}$$

where H_0^{AA} and H_0^{BB} are separated by an energy gap, and $\mathcal{H}'$ is a series in a perturbative parameter. This procedure requires finding a series of the basis transformation $\mathcal{U}$ that is unitary and that also cancels the off-diagonal block of the transformed Hamiltonian order by order, as shown in Fig. 1. The low-energy effective Hamiltonian $\tilde{\mathcal{H}}^{AA}$ is then a series in the perturbative parameter, whose eigenvalues and eigenvectors are approximate solutions of the complete Hamiltonian. As a consequence, the effective model is sufficient to describe the low-energy properties of the original system while also being simpler and easier to handle.

A common approach to constructing an effective Hamiltonian is the Schrieffer–Wolff transformation [1, 2], also known as Löwdin partitioning [3], or quasi-degenerate perturbation theory. This method parameterizes the unitary transformation $\mathcal{U} = e^{-\mathcal{S}}$ and finds the series $\mathcal{S}$ that decouples the A and B subspaces of $\tilde{\mathcal{H}} = e^{\mathcal{S}}\mathcal{H}e^{-\mathcal{S}}$. This idea enabled advances in multiple fields of quantum physics. As an example, all the k.p models are a result of treating crystalline momentum as a perturbation that only weakly mixes atomic orbitals separated in

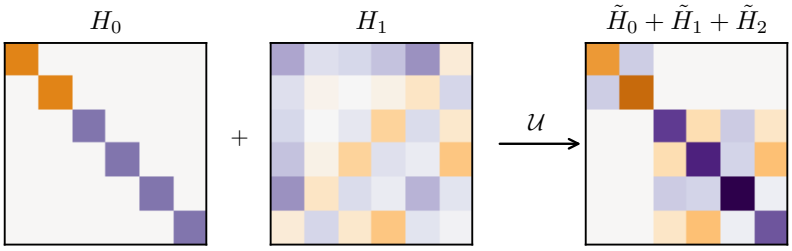

Figure 1: Block-diagonalization of a Hamiltonian with a first order perturbation.

energy [4–7]. More broadly, this method serves as a go-to tool in the study of superconducting circuits and quantum dots, where couplings between circuit elements and drives are treated as perturbations to reproduce the dynamics of the system [8, 9]. Applied to time-dependent Hamiltonians, the Schrieffer–Wolff transformation is an essential tool for the design of quantum gates [10, 11].

Constructing effective Hamiltonians is, however, both algorithmically complex and computationally expensive. This is a consequence of the recursive equations that define the unitary transformation, which require an exponentially growing number of matrix products in each order. In particular, already a 4-th order perturbative expansion that is necessary for many applications may require hundreds of terms. While the computational complexity is only a nuisance when analysing model systems, it becomes a bottleneck whenever the Hilbert space is high-dimensional. Several other approaches improve the performance of the Schrieffer–Wolff algorithm by either using different parametrizations of the unitary transformation [3, 12–15], adjusting the problem setting to density matrix perturbation theory [16, 17], or a finding a similarity transform instead of a unitary [18]. An alternative formulation of the perturbative diagonalization uses Wegner's flow equation [19, 20] to construct a continuous unitary transformation (CUT) that depends on a fictitious flow parameter, which at infinity eliminates the undesired terms from the Hamiltonian [21, 22]. CUT is common in the study of many-body systems [23], and it relies on solving a set of differential equations to obtain the effective Hamiltonian. A more recent line of research even applies the ideas of Schrieffer–Wolff transformation to quantum algorithms for the study of many-body systems [24, 25]. Despite these advances, neither of the approaches combines an optimal scaling with the ability to construct effective Hamiltonians.

Another limitation of the Schrieffer–Wolff transformation is that it only decouples two subspaces at a time. While a straightforward generalization of the Schrieffer–Wolff transformation to multiple subspaces is to decouple one block at a time, this approach is suboptimal and depends on the order in which the blocks are decoupled. The literature on multi-block diagonalization is scarce and considers two approaches: the least action or the block-diagonality of the generator [26]. The former constructs a unitary transformation that is as close as possible to the identity, and the latter constructs a block off-diagonal unitary similar to the Schrieffer–Wolff generator. These approaches are useful to design gates for superconducting qubits [27] and to characterize nonlocal interactions in multi-qubit systems [28], both of which require the decoupling of qubit subspaces from different sets of higher energy states. Reference [26], however, showed that the two generalizations of the Schrieffer–Wolff transformation yield different effective Hamiltonians when applied to more than two subspaces. While the perturbative CUT method naturally decouples multiple subspaces [29], in general solving the differential equations inherent to the method may become a computational bottleneck. To our knowledge, there is no general algorithm that constructs effective Hamiltonians for multiple subspaces directly from the least action principle, and how to do so is an open question.

We introduce an algorithm to construct effective models with optimal scaling, thus making it possible to find high order corrections for systems with millions of degrees of freedom. This algorithm exploits the efficiency of recursive evaluations of series satisfying polynomial constraints and obtains the same effective Hamiltonian as the Schrieffer–Wolff transformation in the case of two subspaces. Our algorithm, however, deals with any number of subspaces, providing a generalization of the Schrieffer–Wolff transformation for multi-block diagonalization and selective decoupling between any two states. We make the algorithm available via the open source package Pymablock[1] (PYthon MAtrix BLOCK-diagonalization), a versatile tool for the study of numerical and symbolic models.

2 Constructing an effective model

We illustrate the construction of effective models by considering several representative examples. The simplest application of effective models is the reduction of finite symbolic Hamiltonians, which appear in the derivation of low-energy dispersions of materials. Starting from a tight-binding model, one performs Taylor expansions of the Hamiltonian near a k-point, and then eliminates several high-energy states [4, 6]. In the study of superconducting qubits, for example, the Hamiltonian contains several bosonic operators, so its Hilbert space is infinite-dimensional, and the coupling between bosons makes the Hamiltonian impossible to diagonalize. The effective qubit model describes the analytical dependence of qubit frequencies and couplings on the circuit parameters [8, 30–34]. This allows to design circuits that realize a desired qubit Hamiltonian, as well as ways to understand and predict qubit dynamics, for which computational tools are being actively developed [35–37]. Finally, mesoscopic quantum devices are described by a single particle tight-binding model with short range hoppings. This produces a numerical Hamiltonian that is both big and sparse, which allows to compute a few of its states but not the full spectrum [38]. Because only the low-energy states contribute to observable properties, deriving how they couple enables a more efficient simulation of the system's behavior.

Pymablock treats all the problems, including the ones above, using a unified approach that only requires three steps:

- Define a Hamiltonian.
- Call `pymablock.block_diagonalize` .
- Request the desired order of the effective Hamiltonian.

The following code snippet shows how to use Pymablock to compute the fourth order correction to an effective Hamiltonian $\tilde{\mathcal{H}}$:

```
# Define perturbation theory
H_tilde, *_ = block_diagonalize([H_0, H_1], subspace_eigenvectors=[vecs_A,
 ↪  vecs_B])

# Request 4th order correction to the effective Hamiltonian
H_AA_4 = H_tilde[0, 0, 4]
```

The function `block_diagonalize` interprets the Hamiltonian $H_0 + H_1$ as a series with two terms, zeroth and first order and calls the block diagonalization routine. The subspaces to decouple are spanned by the eigenvectors `vecs_A` and `vecs_B` of H_0. This is the main function

[1] The documentation and tutorials are available in https://pymablock.readthedocs.io/.

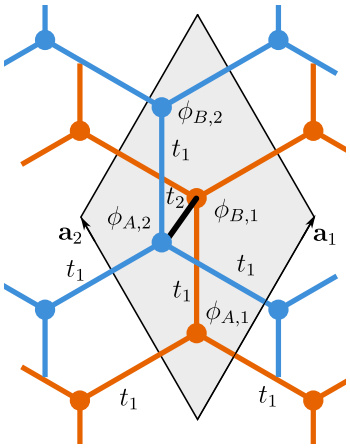

Figure 2: Crystal structure and hoppings of AB-stacked bilayer graphene.

of Pymablock, and it is the only one that the user ever needs to call. Its first output is a multivariate series whose terms are different blocks and orders of the transformed Hamiltonian. Calling `block_diagonalize` only defines the computational problem, whereas querying the elements of `H_tilde` does the actual calculation of the desired order. This interface treats arbitrary formats of Hamiltonians and system descriptions on the same footing and supports both numerical and symbolic computations.

2.1 k.p model of bilayer graphene

To illustrate how to use Pymablock with analytic models, we consider two layers of graphene stacked on top of each other, as shown in Fig. 2. Our goal is to find the low-energy model near the **K** point [6]. To do this, we first construct the tight-binding model Hamiltonian of bilayer graphene. The main features of the model are its 4-atom unit cell spanned by vectors $\mathbf{a}_1 = (1/2, \sqrt{3}/2)$ and $\mathbf{a}_2 = (-1/2, \sqrt{3}/2)$, and with wave functions $\phi_{A,1}, \phi_{B,1}, \phi_{A,2}, \phi_{B,2}$, where A and B indices are the two sublattices, and $1, 2$ are the layers. The model has hoppings t_1 and t_2 within and between the layers, respectively, as shown in Fig. 2. We also include a layer-dependent onsite potential $\pm m$.

We define the Bloch Hamiltonian using the Sympy package for symbolic Python [39].

```
t_1, t_2, m = sympy.symbols("t_1 t_2 m", real=True)
alpha = sympy.symbols(r"\alpha")

H = Matrix([
    [m, t_1 * alpha, 0, 0],
    [t_1 * alpha.conjugate(), m, t_2, 0],
    [0, t_2, -m, t_1 * alpha],
    [0, 0, t_1 * alpha.conjugate(), -m]]
)
```

$$H = \begin{pmatrix} m & t_1\alpha & 0 & 0 \\ t_1\alpha^* & m & t_2 & 0 \\ 0 & t_2 & -m & t_1\alpha \\ 0 & 0 & t_1\alpha^* & -m \end{pmatrix},$$

where $\alpha(\mathbf{k}) = 1 + e^{i\mathbf{k}\cdot\mathbf{a}_1} + e^{\mathbf{k}\cdot\mathbf{a}_2}$, with k the wave vector. We consider $\mathbf{K} = (4\pi/3, 0)$ the reference point point in $\mathbf{k}$-space: $\mathbf{k} = (4\pi/3 + k_x, k_y)$ because $\alpha(\mathbf{K}) = 0$, making k_x and k_y small perturbations. Additionally, we consider $m \ll t_2$ a perturbative parameter.

To call `block_diagonalize`, we need to define the subspaces for the block diagonalization, so we compute the eigenvectors of the unperturbed Hamiltonian at the **K** point, $H(\alpha(\mathbf{K})=m=0)$. Then, we substitute $\alpha(\mathbf{k})$ into the Hamiltonian, and call the block diagonalization routine using that k_x, k_y, and m are perturbative parameters via the `symbols` argument.

```
vecs = H.subs({alpha: 0, m: 0}).diagonalize(normalize=True)[0]

H_tilde, U, U_adjoint = block_diagonalize(
    H.subs({alpha: alpha_k}),
    symbols=(k_x, k_y, m),
    subspace_eigenvectors=[vecs[:, :2], vecs[:, 2:]] # AA, BB
)
```

The order of the variables in the perturbative series will be that of `symbols`. For example, requesting the term $\propto k_x^i k_y^j m^l$ from the effective Hamiltonian is done by calling `H_tilde[0, 0, i, j, l]`, where the first two indices are the block indices (AA). The series of the unitary transformation U and $U^\dagger$ are also defined, and we may use them to transform other operators. We collect corrections up to third order in momentum to compute the standard quadratic dispersion of bilayer graphene and trigonal warping. We query these terms from `H_tilde` and those proportional to mass to obtain the effective Hamiltonian (shown as produced by the code):[2]

$$\tilde{H}_{\text{eff}} = \begin{bmatrix} m & \frac{3t_1^2}{4t_2}(-k_x^2 - 2ik_xk_y + k_y^2) \\ \frac{3t_1^2}{4t_2}(-k_x^2 + 2ik_xk_y + k_y^2) & -m \end{bmatrix} + \begin{bmatrix} \frac{3mt_1^2}{2t_2^2}(-k_x^2 - k_y^2) & \frac{\sqrt{3}t_1^2}{8t_2}(k_x^3 - 5ik_x^2k_y + 9k_xk_y^2 + 3ik_y^3) \\ \frac{\sqrt{3}t_1^2}{8t_2}(k_x^3 + 5ik_x^2k_y + 9k_xk_y^2 - 3ik_y^3) & \frac{3mt_1^2}{2t_2^2}(k_x^2 + k_y^2) \end{bmatrix}.$$

The first term is the standard quadratic dispersion of gapped bilayer graphene. The second term contains trigonal warping and the coupling between the gap and momentum. All the terms take less than two seconds in a personal computer to compute.

2.2 Dispersive shift of a transmon qubit coupled to a resonator

The need for analytical effective Hamiltonians often arises in circuit quantum electrodynamics (cQED) problems, which we illustrate by studying a transmon qubit coupled to a resonator [8]. Specifically, we choose the standard problem of finding the frequency shift of the resonator due to its coupling to the qubit, a phenomenon used to measure the qubit's state [30]. The Hamiltonian of the system is given by

$$\mathcal{H} = -\omega_t\left(a_t^\dagger a_t - \frac{1}{2}\right) + \frac{\alpha}{2}a_t^\dagger a_t^\dagger a_t a_t + \omega_r\left(a_r^\dagger a_r + \frac{1}{2}\right) - g(a_t^\dagger - a_t)(a_r^\dagger - a_r), \tag{2}$$

where a_t and a_r are bosonic annihilation operators of the transmon and resonator, respectively, and ω_t and ω_r are their frequencies. The transmon has an anharmonicity α, so that its energy levels are not equally spaced. In presence of both the coupling g between the transmon and the resonator and the anharmonicity, this Hamiltonian admits no analytical solution. We therefore treat g as a perturbative parameter.

To deal with the infinite dimensional Hilbert space, we observe that the perturbation only changes the occupation numbers of the transmon and the resonator by ± 1. Therefore computing n-th order corrections to the n_0-th state allows to disregard states with any occupation

[2] The full code is available at https://pymablock.readthedocs.io/en/latest/tutorial/bilayer_graphene.html.

numbers larger than $n_0 + n/2$. We want to compute the second order correction to the levels with occupation numbers of either the transmon or the resonator being 0 and 1. We accordingly truncate the Hilbert space to the lowest 3 levels of the transmon and the resonator. The resulting Hamiltonian is a 9×9 matrix that we construct using Sympy [39].

Finally, to compute the energy corrections of the lowest levels, we call `block_diagonalize` for each state separately, replicating a regular perturbation theory calculation for single wavefunctions. To do this, we observe that H_0 is diagonal, and use `subspace_indices` to assign the elements of its eigenbasis to the 4 subspaces of interest and the rest. This corresponds to a multi-block diagonalization problem with 5 blocks. For example, to find the qubit-dependent frequency shift of the resonator, χ, we start by computing the second order correction to $|0_t 0_r\rangle$:

```
indices = [0, 1, 2, 3, 4, 4, 4, 4, 4] # 00 is the first state in the basis
H_tilde, *_ = block_diagonalize(H, subspace_indices=indices, symbols=[g])
H_tilde[0, 0, 2][0, 0] # 2nd order correction to 00
```

$$E_{00}^{(2)} = \frac{g^2}{-\omega_r + \omega_t}. \tag{3}$$

Repeating this process for the states $|1_t 0_r\rangle$, $|0_t 1_r\rangle$, and $|1_t 1_r\rangle$ requires requesting the terms `H_tilde[1, 1, 2][0, 0]`, `H_tilde[2, 2, 2][0, 0]`, and `H_tilde[3, 3, 2][0, 0]`, and yields the desired resonator frequency shift:

$$\begin{aligned}\chi &= (E_{11}^{(2)} - E_{10}^{(2)}) - (E_{01}^{(2)} - E_{00}^{(2)}) \\ &= -\frac{2g^2}{\alpha + \omega_r - \omega_t} + \frac{2g^2}{-\alpha + \omega_r + \omega_t} - \frac{2g^2}{\omega_r + \omega_t} + \frac{2g^2}{\omega_r - \omega_t} \\ &= -\frac{4\alpha g^2 \left(\alpha\omega_t - \omega_r^2 - \omega_t^2\right)}{(\omega_r - \omega_t)(\omega_r + \omega_t)(-\alpha + \omega_r + \omega_t)(\alpha + \omega_r - \omega_t)}.\end{aligned} \tag{4}$$

In this example, we have not used the rotating wave approximation, including the frequently omitted counter-rotating terms $\sim a_r a_t$ to illustrate the extensibility of Pymablock. Computing higher order corrections to the qubit frequency only requires increasing the size of the truncated Hilbert space and requesting `H_tilde[0, 0, n]` to any order n.

2.3 Induced gap in a double quantum dot

Large systems pose an additional challenge due to the cubic scaling of linear algebra routines with matrix size. To overcome this, Pymablock is equipped with an implicit method, which utilizes the sparsity of the input and avoids the construction of the full transformed Hamiltonian. We illustrate the efficiency of this method by applying it to a system of two quantum dots coupled to a superconductor between them, shown in Fig. 3, and described by the Bogoliubov-de Gennes Hamiltonian:

$$H_{BdG} = \begin{cases} (\mathbf{k}^2/2m - \mu_{sc})\sigma_z + \Delta\sigma_x, & \text{for } L/3 \le x \le 2L/3, \\ (\mathbf{k}^2/2m - \mu_n)\sigma_z, & \text{otherwise,} \end{cases} \tag{5}$$

where the Pauli matrices σ_z and σ_x act in the electron-hole space, $\mathbf{k}$ is the 2D wave vector, m is the effective mass, and Δ the superconducting gap.

We use the Kwant package [40] to build the Hamiltonian of the system,[3] which we define over a square lattice of $L \times W = 200 \times 40$ sites. On top of this, we consider two perturbations:

[3] The full code is available at https://pymablock.readthedocs.io/en/latest/tutorial/induced_gap.html.

the barrier strength between the quantum dots and the superconductor, t_b, and an asymmetry of the dots' potentials, $\delta\mu$.

The system is large: it is a sparse array of size 63042×63042, with 333680 non-zero elements, so even storing all the eigenvectors would take 60 GB of memory. The perturbations are also sparse, with 632, and 126084 non-zero elements for the barrier strength and the potential asymmetry, respectively. The sparsity structure of the Hamiltonian and the perturbations is shown in the left panel of Fig. 3, where we use a smaller system of $L \times W = 8 \times 2$ for visualization. Therefore, we use sparse diagonalization [41] and compute only four eigenvectors of the unperturbed Hamiltonian closest to zero energy, which are the Andreev states of the quantum dots.

```
vals, vecs = scipy.sparse.linalg.eigsh(h_0, k=4, sigma=0)
vecs, _ = scipy.linalg.qr(vecs, mode="economic")  # orthogonalize the vectors
```

We now call the block diagonalization routine and provide the computed eigenvectors.

```
H_tilde, *_ = block_diagonalize([h_0, barrier, dmu], subspace_eigenvectors=[vecs])
```

Because we only provide the low-energy subspace, Pymablock uses the implicit method. Calling `block_diagonalize` is now the most time-consuming step because it requires precomputing several decompositions of the full Hamiltonian. It is, however, manageable and it only produces a constant overhead of less than three seconds.

To compute the spectrum, we collect the lowest three orders in each parameter in an appropriately sized tensor.

```
h_tilde = np.array(np.ma.filled(H_tilde[0, 0, :3, :3], fill_value).tolist())
```

This takes two more seconds to run, and we can now compute the low-energy spectrum after rescaling the perturbative corrections by the magnitude of each perturbation.

```
def effective_energies(h_tilde, barrier, dmu):
    barrier_powers = barrier ** np.arange(3).reshape(-1, 1, 1, 1)
    dmu_powers = dmu ** np.arange(3).reshape(1, -1, 1, 1)
    return scipy.linalg.eigvalsh(
        np.sum(h_tilde * barrier_powers * dmu_powers, axis=(0, 1))
    )
```

Finally, we plot the spectrum of the 2 Andreev states in Fig. 3. As expected, the crossing at $E = 0$ due to the dot asymmetry is lifted when the dots are coupled to the superconductor. In addition, we observe how the proximity gap of the dots increases with the coupling strength.

Computing the spectrum of the system for 3 points in parameter space would require the same time as the total runtime of Pymablock in this example. This demonstrates the speed of the implicit method and the efficiency of Pymablock's algorithm.

2.4 Selective diagonalization

Lastly, we demonstrate the generality of Pymablock's algorithm by applying it to decouple arbitrary states in a generic Hamiltonian. This is an alternative to separating a Hamiltonian into blocks, and it requires that the states to decouple are different in energy. To illustrate this, we consider a 16×16 Hamiltonian $\mathcal{H} = H_0 + H_1$ with H_0 a diagonal matrix and H_1 a random Hermitian perturbation. Our goal is to construct an effective Hamiltonian whose only matrix elements are those in a binary mask, which, without loss of generality, we choose to be a smiley face.

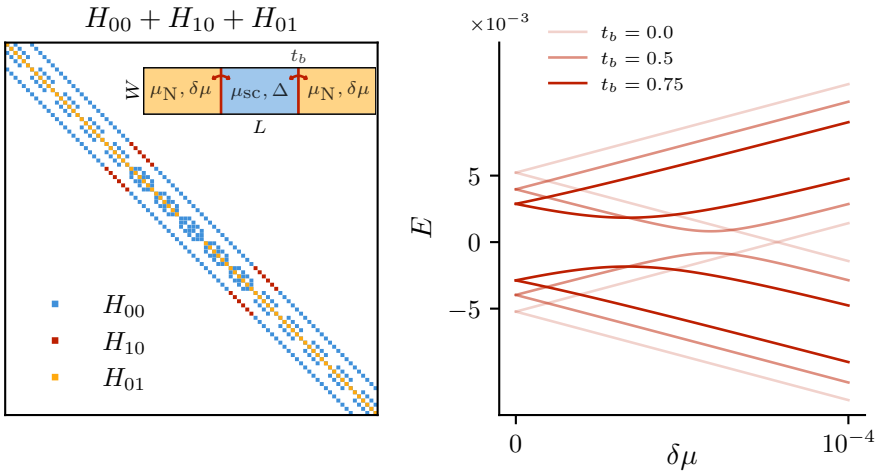

Figure 3: Hamiltonian (left) and Andreev levels (right) of two quantum dots coupled to a superconductor (inset). The barrier t_b between the dots and the superconductor, H_{10}, and the asymmetry $\delta\mu$ between the dots' potential, H_{01}, are perturbations.

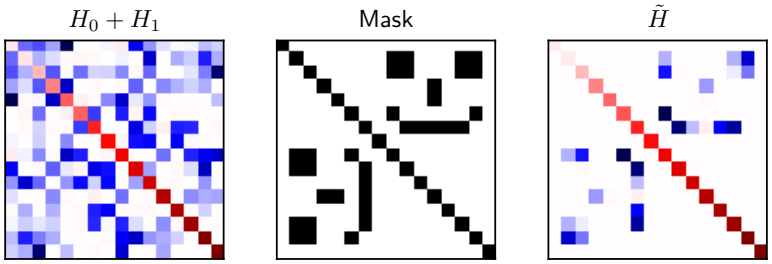

Figure 4: Selective diagonalization of a Hamiltonian with a random perturbation.

We apply the mask to the Hamiltonian by providing it as the `fully_diagonalize` argument to `block_diagonalize`.[4]

```
H_tilde, *_ = block_diagonalize([H_0, H_1], fully_diagonalize={0: mask})
```

The argument `fully_diagonalize` is a dictionary where the keys label the blocks of the Hamiltonian, and the values are the masks that select the terms to keep in that block. We only used one block in this example: the entire Hamiltonian. Finally, the effective Hamiltonian only contains the terms in the mask, as shown in Fig. 4.

3 Perturbative block-diagonalization algorithm

3.1 Problem statement

Pymablock finds a series of the unitary transformation $\mathcal{U}$ (we use calligraphic letters to denote series) that eliminates the off-diagonal components of the Hamiltonian

$$\mathcal{H} = H_0 + \mathcal{H}', \tag{6}$$

[4] The full code is available at https://pymablock.readthedocs.io/en/latest/tutorial/getting_started.html#selective-diagonalization.

with $\mathcal{H}' = \mathcal{H}'_S + \mathcal{H}'_R$ containing an arbitrary number and orders of perturbations with block-diagonal and block-offdiagonal components, respectively. Here and later we use the subscript S to denote the selected part and R to denote remaining components of a series, with the goal of the perturbation theory to obtain a Hamiltonian with only the selected part. In other words, we aim to find a unitary transformation $\mathcal{U}$ that cancels the remaining part of the Hamiltonian. In different settings, selected and remaining parts may mean different things. In quasi-degenerate perturbation theory, the Hilbert space is subdivided into A and B subspaces, which makes H_0 a block-diagonal matrix

$$H_0 = \begin{pmatrix} {H_0}^{AA} & 0 \\ 0 & {H_0}^{BB} \end{pmatrix}, \tag{7}$$

and the goal of the perturbation theory is to eliminate the offdiagonal AB and BA blocks of $\mathcal{H}$. In this case the selected part is the block-diagonal part, and the remaining part is the block-offdiagonal part. Differently, in the context of Rayleigh-Schrödinger perturbation theory, H_0 is a diagonal matrix so that the selected part is the diagonal, and the remaining part of an operator are all its matrix elements that are not on the diagonal.

To consider the problem in the most general setting, we only require the selected and remaining parts of an operator to satisfy the following constraints:

1. The selected and remaining parts of an operator add to identity: $\mathcal{A} = \mathcal{A}_S + \mathcal{A}_R$.
2. Taking either part of an operator is idempotent: $(\mathcal{A}_S)_S = \mathcal{A}_S$.
3. Taking either part commutes with Hermitian conjugation: $(\mathcal{A}_S)^\dagger = (\mathcal{A}^\dagger)_S$.
4. The remaining part of any operator has no matrix elements within eigensubspaces of H_0. This is required to ensure that the perturbation theory is well-defined.

The separation of an operator into selected and remaining parts is a generalization of taking block-diagonal and block-offdiagonal parts. In particular, the separation allows to choose any subset of the offdiagonal matrix elements as remaining, as long as none of the matrix elements belong to an eigensubspace of H_0. That none of the matrix elements belong to a same eigensubspace of H_0 becomes evident in the textbook quasi-degenerate perturbation theory, where the corrections to energies and wavefunctions contain differences between energy of the states from different subspaces. The main difference between our generalization and the standard separation into block-diagonal and block-offdiagonal is that the product of a selected part and remaining part of two operators may have a non-zero selected part: $(\mathcal{A}_S\mathcal{B}_R)_S \neq 0$, while $(\mathcal{A}^{AA}\mathcal{B}^{AB})^{AA} = 0$. The generality of the selected and remaining parts allows to consider all perturbation theory methods with the same algorithm, including multi-block diagonalization, selective diagonalization, and the Schrieffer–Wolff transformation. Several expressions simplify if the selected part corresponds to a block-diagonal operator and simplify further if there are only two subspaces. We keep track of these simplifications.

All the series we consider may be multivariate, and they represent sums of the form

$$\mathcal{A} = \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty}\cdots\sum_{n_k=0}^{\infty} \lambda_1^{n_1}\lambda_2^{n_2}\cdots\lambda_k^{n_k} A_{n_1,n_2,\ldots,n_k}, \tag{8}$$

where λ_i are the perturbation parameters and $A_{n_1,n_2,\ldots,n_k}$ are linear operators. The problem statement, therefore, is finding $\mathcal{U}$ and $\tilde{\mathcal{H}}$ such that

$$\tilde{\mathcal{H}} = \mathcal{U}^\dagger\mathcal{H}\mathcal{U}, \quad \tilde{\mathcal{H}}_R = 0, \quad \mathcal{U}^\dagger\mathcal{U} = 1, \tag{9}$$

which is schematically shown in Fig. 1 for the case of two subspaces, where the selected parts are AA and BB, and the remaining parts are AB and BA. Series multiply according to the Cauchy product:

$$\mathcal{C} = \mathcal{A}\mathcal{B} \Leftrightarrow C_{\mathbf{n}} = \sum_{\mathbf{m}+\mathbf{p}=\mathbf{n}} A_{\mathbf{m}} B_{\mathbf{p}} \,.$$

The Cauchy product is the most expensive operation in perturbation theory, because it involves a large number of multiplications between potentially large matrices. For example, evaluating $\mathbf{n}$-th order of $\mathcal{C}$ requires $\sim \prod_i n_i \equiv N$ multiplications of the series elements.[5] A direct computation of all the possible index combinations in a product between three series $\mathcal{A}\mathcal{B}\mathcal{C}$ would have a higher cost $\sim N^2$, however, if we use associativity of the product and compute this as $(\mathcal{A}\mathcal{B})\mathcal{C}$, then the scaling of the cost stays $\sim N$.

There are many ways to solve the problem (9) that give identical expressions for $\mathcal{U}$ and $\tilde{\mathcal{H}}$. We are searching for a procedure that satisfies two additional constraints:

- It has the same complexity scaling as a Cauchy product, and therefore $\sim N$ multiplications per additional order.
- It does not require multiplications by H_0.
- It requires only one Cauchy product by $\mathcal{H}_S$, the selected part of $\mathcal{H}$.

The first requirement is that the algorithm scaling is optimal: the desired expression at least contains a Cauchy product of $\mathcal{U}$ and $\mathcal{H}$. Therefore the complexity scaling of the complete algorithm may not become lower than the complexity of a Cauchy product and we aim to reach this lower bound. The second requirement is because in perturbation theory, n-th order corrections to $\tilde{\mathcal{H}}$ carry n energy denominators $1/(E_i - E_j)$, where E_i and E_j are the eigenvalues of H_0 belonging to different subspaces. Therefore, any additional multiplications by H_0 must cancel with additional energy denominators. Multiplying by H_0 is therefore unnecessary work, and it gives longer intermediate expressions. The third requirement we impose by considering a case in which $\mathcal{H}_R = 0$, where $\mathcal{H}_S$ must at least enter $\tilde{\mathcal{H}}$ as an added term, without any products. Moreover, because $\mathcal{U}$ depends on the entire Hamiltonian, there must be at least one Cauchy product by $\mathcal{H}'_S$. The goal of our algorithm is thus to be efficient and to produce compact results that do not require further simplifications.

3.2 Existing solutions

A common approach to constructing effective Hamiltonians in the 2×2 block case is to use the Schrieffer–Wolff transformation [1]:

$$\begin{aligned} \tilde{\mathcal{H}} &= e^{\mathcal{S}} \mathcal{H} e^{-\mathcal{S}} \,, \\ e^{\mathcal{S}} &= 1 + \mathcal{S} + \frac{1}{2!}\mathcal{S}\mathcal{S} + \frac{1}{3!}\mathcal{S}\mathcal{S}\mathcal{S} + \cdots \,, \end{aligned} \tag{10}$$

where $\mathcal{S} = \sum_n S_n$ is an antihermitian polynomial series in the perturbative parameter, making $e^{\mathcal{S}}$ a unitary transformation. Requiring that $\tilde{\mathcal{H}}^{AB} = 0$ gives a recursive equation for S_n, whose terms are nested commutators between the series of $\mathcal{S}$ and $\mathcal{H}$. Similarly, the transformed Hamiltonian is given by a series of nested commutators

$$\tilde{\mathcal{H}} = \sum_{j=0}^{\infty} \frac{1}{j!} \left[\mathcal{H}, \sum_{n=0}^{\infty} S_n \right]^{(j)} \,, \tag{11}$$

[5] If both $\mathcal{A}$ and $\mathcal{B}$ are known in advance, fast Fourier transform-based algorithms can reduce this cost to $\sim N \log N$. In our problem, however, the series are constructed recursively and therefore this optimization is impossible.

where the superscript (j) denotes the j-th nested commutator $[A,B]^{(j)} = [[A,B]^{(j-1)},B]$, with $[A,B]^{(0)} = A$ and $[A,B]^{(1)} = AB-BA$. Regardless of the specific implementation, this expression does not meet either of our two requirements:

- The direct computation of the series elements requires $\sim \exp N$ multiplications, and even an optimized one has a $\sim N^2$ scaling.
- Evaluating Eq. (11) contains multiplications by H_0.

Additionally, while in the 2×2 block case the Schrieffer–Wolff transformation produces a minimal unitary transformation, i.e. as close to identity as possible, this is not the case in the multi-block case [26]. The generalization of this approach to multiple subspaces is an open question [26].

Alternative parametrizations of the unitary transformation $\mathcal{U}$ require solving unitarity and block diagonalization conditions too, but give rise to a different recursive procedure for the series elements. For example, using hyperbolic functions

$$\mathcal{U} = \cosh \mathcal{G} + \sinh \mathcal{G}\,, \quad \mathcal{G} = \sum_{i=0}^{\infty} G_i\,, \tag{12}$$

leads to different recursive expressions for G_i [13], but does not change the algorithm's complexity. On the other hand, using a polynomial series directly

$$\mathcal{U} = \sum_{i=0}^{\infty} U_i\,, \tag{13}$$

gives rise to another recursive equation for U_i [3, 12, 14, 15]. Still, this choice results in an expression for $\tilde{\mathcal{H}}$ whose terms include products by H_0, and therefore requires additional simplifications.

Another approach uses Wegner's flow equation [19, 20] to construct a continuous unitary transformation (CUT) that depends smoothly on a fictitious parameter l, $\mathcal{U}(l)$. The goal is to define a generator $\eta(l)$ such that $\mathcal{H}(l) = \mathcal{U}^\dagger(l)\mathcal{H}(0)\mathcal{U}(l)$ flows towards the desired effective Hamiltonian:

$$\frac{d\mathcal{H}(l)}{dl} = \big[\eta(l), \mathcal{H}(l)\big], \tag{14}$$

where $\mathcal{U}(l)$, $\mathcal{H}(l)$, and $\eta(l)$ are once again series in the perturbative parameters. At $l = \infty$, the transformed Hamiltonian does not contain the undesired terms, $\mathcal{H}(\infty) = \tilde{\mathcal{H}}$. Finding the unitary amounts to solving a set of differential equations

$$\frac{d\mathcal{U}(l)}{dl} = \eta(l)\mathcal{U}(l)\,. \tag{15}$$

Together with the Eq. (14) and an appropriate choice of η, this gives a set of coupled differential equations, that become linear if solved order by order. The convergence and stability of flow equations depends on the parameterization of the flow generator η, and multiple strategies for this choice are known [23, 42]. The CUT method is common in the study of many-body systems, where one needs to either decompose the Hamiltonian into sets of quasiparticle creation and annihilation operators, or choose a different operator basis together with a set of commutation rules. Despite the numerical complication of solving differential equations, CUT extends beyond the perturbative regime [20, 22, 23].

The following three algorithms satisfy both of our requirements while solving a related problem. First, density matrix perturbation theory [16, 17, 43] constructs the density matrix ρ of a perturbed system as a power series with respect to a perturbative parameter:

$$\rho = \sum_{i=0}^{\infty} \rho_i\,. \tag{16}$$

The elements of the series are found by solving two recursive conditions, $\rho^2 = \rho$ and $[\mathcal{H}, \rho] = 0$, which avoid multiplications by H_0 and require a single Cauchy product each. This approach, however, deals with the entire Hilbert space, rather than the low-energy subspace, and does not provide an effective Hamiltonian. Second, the perturbative similarity transform by C. Bloch [2,18] constructs the effective Hamiltonian in a non-orthogonal basis, which preserves the Hamiltonian spectrum while breaking its hermiticity. Third, the recursive Schrieffer–Wolff algorithm [37] applies the Schrieffer–Wolff transformation to the output of lower-order iterations, and calculates the effective Hamiltonian at a fixed perturbation strength, rather than as a series. Finally, none of these linear scaling algorithms above handles more than two subspaces. We thus identify the following open question: can we construct an effective Hamiltonian with a linear scaling algorithm that produces compact expressions?

3.3 Pymablock's algorithm

The first idea that Pymablock exploits is the recursive evaluation of the operator series, which we illustrate by considering the unitarity condition. Let us separate the transformation $\mathcal{U}$ into an identity and $\mathcal{U}' = \mathcal{W} + \mathcal{V}$:

$$\mathcal{U} = 1 + \mathcal{U}' = 1 + \mathcal{W} + \mathcal{V}, \quad \mathcal{W}^\dagger = \mathcal{W}, \quad \mathcal{V}^\dagger = -\mathcal{V}. \tag{17}$$

We use the unitarity condition $\mathcal{U}^\dagger\mathcal{U} = 1$ by substituting $\mathcal{U}'$ into it:

$$1 = (1 + \mathcal{U}'^\dagger)(1 + \mathcal{U}') = 1 + \mathcal{U}'^\dagger + \mathcal{U}' + \mathcal{U}'^\dagger\mathcal{U}'. \tag{18}$$

This immediately yields

$$\mathcal{W} = \frac{1}{2}(\mathcal{U}'^\dagger + \mathcal{U}') = -\frac{1}{2}\mathcal{U}'^\dagger\mathcal{U}'. \tag{19}$$

Because $\mathcal{U}'$ has no 0-th order term, $(\mathcal{U}'^\dagger\mathcal{U}')_{\mathbf{n}}$ does not depend on the $\mathbf{n}$-th order of $\mathcal{U}'$ nor $\mathcal{W}$, and therefore Eq. (19) allows to compute $\mathcal{W}$ using the already available lower orders of $\mathcal{U}'$. Alternatively, using Eq. (17) we could define $\mathcal{W}$ as a Taylor series in $\mathcal{V}$:

$$\mathcal{W} = \sqrt{1 + \mathcal{V}^2} - 1 \equiv f(\mathcal{V}) \equiv \sum_n a_n \mathcal{V}^{2n}.$$

A direct computation of all possible products of terms in this expression requires $\sim \exp N$ multiplications. A more efficient approach for evaluating this expression introduces each term in the sum as a new series $\mathcal{A}^{n+1} = \mathcal{A}\mathcal{A}^n$ and reuses the previously computed results. This optimization brings the exponential cost down to $\sim N^2$. However, we see that the Taylor expansion approach is both more complicated and more computationally expensive than the recurrent definition in Eq. (19). Therefore, we use Eq. (19) to efficiently compute $\mathcal{W}$. More generally, a Cauchy product $\mathcal{A}\mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ have no 0-th order terms depends on $\mathcal{A}_1, \ldots, \mathcal{A}_{n-1}$ and $\mathcal{B}_1, \ldots, \mathcal{B}_{n-1}$. This makes it possible to use $\mathcal{A}\mathcal{B}$ in a recurrence relation, a property that we exploit throughout the algorithm.

To compute $\mathcal{U}'$ we also need to find $\mathcal{V}$, which is defined by the requirement $\tilde{\mathcal{H}}_R = 0$. Additionally, we constrain $\mathcal{V}$ to have no selected part: $\mathcal{V}_S = 0$, a choice we make to minimize the norm of $\mathcal{U}'$, and satisfy the least action principle [44]. That $\mathcal{V}_S = 0$ minimizes the norm of $\mathcal{U}'$ follows from the following statements:

1. The norm of a series is minimal, when each of the subsequent terms is chosen to be minimal order by order.

2. The Hermitian part of $\mathcal{U}'$, $W_{\mathbf{n}}$, is determined by the unitarity condition (19) at each order from lower orders of $\mathcal{U}'$.

3. The norm of $W_{\mathbf{n}}+V_{\mathbf{n}}$ is minimal, when the norm of $V_{\mathbf{n}}$ is minimal because of Hermiticity properties of $\mathcal{W}$ and $\mathcal{V}$.

4. Finally, because $\mathcal{V}_R$ is fixed by the requirement $\tilde{\mathcal{H}}_R = 0$, $\mathcal{V}_S = 0$ provides the minimal norm of $\mathcal{U}'$.

In the 2×2 block case, this choice makes $\mathcal{W}$ block-diagonal and ensures that the resulting unitary transformation is equivalent to the Schrieffer–Wolff transformation (see section 3.4). In general, however, $\mathcal{W}_R \neq 0$.

The remaining condition for finding a recurrent relation for $\mathcal{U}'$ is that the transformed Hamiltonian

$$\tilde{\mathcal{H}} = \mathcal{U}^\dagger \mathcal{H}\mathcal{U} = \mathcal{H}_S + \mathcal{U}'^\dagger \mathcal{H}_S + \mathcal{H}_S\mathcal{U}' + \mathcal{U}'^\dagger \mathcal{H}_S\mathcal{U}' + \mathcal{U}^\dagger \mathcal{H}'_R\mathcal{U}\,, \tag{20}$$

has only the selected part $\tilde{\mathcal{H}}_R = 0$, a condition that determines $\mathcal{V}$. Here we used $\mathcal{U} = 1+\mathcal{U}'$ and $\mathcal{H} = \mathcal{H}_S + \mathcal{H}'_R$, since H_0 is has no remaining part by definition. Because we want to avoid products by $\mathcal{H}_S$, we need to get rid of the terms that contain it by replacing them with an alternative expression. Our strategy is to define an auxiliary operator $\mathcal{X}$ that we can compute without ever multiplying by $\mathcal{H}_S$. Like $\mathcal{U}'$, $\mathcal{X}$ needs to be defined via a recurrence relation, which we determine later. Because Eq. (20) contains $\mathcal{H}_S$ multiplied by $\mathcal{U}'$ from the left and from the right, eliminating $\mathcal{H}_S$ requires moving it to the same side. To achieve this, we choose $\mathcal{X} = \mathcal{Y} + \mathcal{Z}$ to be the commutator between $\mathcal{U}'$ and $\mathcal{H}_S$:

$$\mathcal{X} \equiv [\mathcal{U}', \mathcal{H}_S] = \mathcal{Y} + \mathcal{Z}\,, \quad \mathcal{Y} \equiv [\mathcal{V}, \mathcal{H}_S] = \mathcal{Y}^\dagger\,, \quad \mathcal{Z} \equiv [\mathcal{W}, \mathcal{H}_S] = -\mathcal{Z}^\dagger\,. \tag{21}$$

If the selected part $\mathcal{A}_S$ corresponds to a block-diagonal operator, $\mathcal{Y}$ is block off-diagonal. Additionally, in the 2×2 block case $\mathcal{Z}$ is block-diagonal. We use $\mathcal{H}_S\mathcal{U}' = \mathcal{U}'\mathcal{H}_S - \mathcal{X}$ to move $\mathcal{H}_S$ through to the right and find

$$\begin{aligned}
\tilde{\mathcal{H}} &= \mathcal{H}_S + \mathcal{U}'^\dagger \mathcal{H}_S + (\mathcal{H}_S\mathcal{U}') + \mathcal{U}'^\dagger \mathcal{H}_S\mathcal{U}' + \mathcal{U}^\dagger(\mathcal{H}'_R\mathcal{U}) \\
&= \mathcal{H}_S + \mathcal{U}'^\dagger \mathcal{H}_S + \mathcal{U}'\mathcal{H}_S - \mathcal{X} + \mathcal{U}'^\dagger(\mathcal{U}'\mathcal{H}_S - \mathcal{X}) + \mathcal{U}^\dagger \mathcal{H}'_R\mathcal{U} \\
&= \mathcal{H}_S + (\mathcal{U}'^\dagger + \mathcal{U}' + \mathcal{U}'^\dagger\mathcal{U}')\mathcal{H}_S - \mathcal{X} - \mathcal{U}'^\dagger\mathcal{X} + \mathcal{U}^\dagger \mathcal{H}'_R\mathcal{U} \\
&= \mathcal{H}_S - \mathcal{X} - \mathcal{U}'^\dagger\mathcal{X} + \mathcal{U}^\dagger \mathcal{H}'_R\mathcal{U}\,,
\end{aligned} \tag{22}$$

where the terms multiplied by $\mathcal{H}_S$ cancel according to Eq. (18). The transformed Hamiltonian does not contain multiplications by $\mathcal{H}_S$ anymore, but it does depend on $\mathcal{X}$, an auxiliary operator whose recurrent definition we do not know yet. To find it, we first focus on its anti-Hermitian part, $\mathcal{Z}$. Since recurrence relations are expressions whose right-hand side contains Cauchy products between series, we need to find a way to make a product appear. We do so by using the unitarity condition $\mathcal{U}'^\dagger + \mathcal{U}' = -\mathcal{U}'^\dagger\mathcal{U}'$ to obtain the recursive definition of $\mathcal{Z}$:

$$\begin{aligned}
\mathcal{Z} &= \frac{1}{2}(\mathcal{X} - \mathcal{X}^\dagger) \\
&= \frac{1}{2}\Big[(\mathcal{U}' + \mathcal{U}'^\dagger)\mathcal{H}_S - \mathcal{H}_S(\mathcal{U}' + \mathcal{U}'^\dagger)\Big] \\
&= \frac{1}{2}\Big[-\mathcal{U}'^\dagger(\mathcal{U}'\mathcal{H}_S - \mathcal{H}_S\mathcal{U}') + (\mathcal{U}'\mathcal{H}_S - \mathcal{H}_S\mathcal{U}')^\dagger\mathcal{U}'\Big] \\
&= \frac{1}{2}(-\mathcal{U}'^\dagger\mathcal{X} + \mathcal{X}^\dagger\mathcal{U}')\,.
\end{aligned} \tag{23}$$

Similar to computing $W_{\mathbf{n}}$, computing $Z_{\mathbf{n}}$ requires lower-orders of $\mathcal{X}$ and $\mathcal{U}'$. Then, we compute the Hermitian part of $\mathcal{X}$ by requiring that $\tilde{\mathcal{H}}_R = 0$ in the Eq. (22) and find

$$\mathcal{Y}_R = (\mathcal{U}^\dagger \mathcal{H}'_R\mathcal{U} - \mathcal{U}'^\dagger\mathcal{X} - \mathcal{Z})_R\,. \tag{24}$$

Once again, despite $\mathcal{X}$ enters the right hand side, because all the terms lack 0^{th} order, this defines a recursive relation $\mathcal{Y}$. To fix $\mathcal{Y}_S$, we use its definition (21), which gives

$$[\mathcal{V}, H_0] = \mathcal{Y} - [\mathcal{V}, \mathcal{H}'_S], \tag{25}$$

which is a continuous-time Lyapunov equation for $\mathcal{V}$. In order for this equation to be satisfiable, the selected part of the right hand side must vanish, since the left hand side has no selected part. Therefore we find:

$$\mathcal{Y}_S = [\mathcal{V}, \mathcal{H}'_S]_S\,, \tag{26}$$

and it vanishes if the selected part corresponds to a block-diagonal matrix.

The final part is straightforward. Finding $\mathcal{V}$ from $\mathcal{Y}$ amounts to solving a Sylvester's equation, Eq. (26), which we only need to solve once for every new order. This is the only step in the algorithm that requires a direct multiplication by $\mathcal{H}'_S$. In the eigenbasis of H_0, the solution of Sylvester's equation is $V_{\mathbf{n},ij} = (\mathcal{Y}_R - [\mathcal{V}, \mathcal{H}'_S]_R)_{\mathbf{n},ij}/(E_i - E_j)$, where E_i are the eigenvalues of H_0. However, even if the eigenbasis of H_0 is not available, there are efficient numerical algorithms to solve Sylvester's equation (see Sec. 4.2). An alternative is to decompose the Hamiltonian into its eigenoperator basis. This approach avoids specifying the eigenbasis of H_0, and therefore it is better suited for second-quantized Hamiltonians [45, 46].

We now have the complete algorithm:

1. Define series $\mathcal{U}'$ and $\mathcal{X}$ and make use of their block structure and Hermiticity.
2. To define the hermitian part of $\mathcal{U}'$, use $\mathcal{W} = -\mathcal{U}'^{\dagger}\mathcal{U}'/2$.
3. To find the antihermitian part of $\mathcal{U}'$, solve Sylvester's equation $[\mathcal{V}, H_0] = (\mathcal{Y} - [\mathcal{V}, \mathcal{H}'_S])_R$. This requires $\mathcal{X}$.
4. To find the antihermitian part of $\mathcal{X}$, define $\mathcal{Z} = (-\mathcal{U}'^{\dagger}\mathcal{X} + \mathcal{X}^{\dagger}\mathcal{U}')/2$.
5. For the Hermitian part of $\mathcal{X}$, use $\mathcal{Y} = (-\mathcal{U}'^{\dagger}\mathcal{X} + \mathcal{U}^{\dagger}\mathcal{H}'\mathcal{U})_R + [\mathcal{V}, \mathcal{H}'_S]_S$.
6. Compute the effective Hamiltonian as $\tilde{\mathcal{H}} \equiv \tilde{\mathcal{H}}_S = \mathcal{H}_S - \mathcal{X} - \mathcal{U}'^{\dagger}\mathcal{X} + \mathcal{U}^{\dagger}\mathcal{H}'_R\mathcal{U}$.

3.4 Equivalence to Schrieffer–Wolff transformation

Pymablock's algorithm applied to 2×2 block-diagonalization and the Schrieffer–Wolff transformation both find a unitary transformation $\mathcal{U}$ such that $\tilde{\mathcal{H}}_R = \tilde{\mathcal{H}}^{AB} = 0$. They are therefore equivalent up to a gauge choice in each subspace, A and B. We establish the equivalence between the two by demonstrating that this gauge choice is the same for both algorithms. The Schrieffer–Wolff transformation uses $\mathcal{U} = \exp \mathcal{S}$, where $\mathcal{S} = -\mathcal{S}^{\dagger}$ and $\mathcal{S}^{AA} = \mathcal{S}^{BB} = 0$, this restriction makes the result unique [2]. On the other hand, our algorithm produces the unique block-diagonalizing transformation with a block structure $\mathcal{U}^{AA} = \mathcal{U}^{AA\dagger}$, $\mathcal{U}^{BB} = \mathcal{U}^{BB\dagger}$ and $\mathcal{U}^{AB} = -\mathcal{U}^{\dagger}_{BA}$. The uniqueness is a consequence of the construction of the algorithm, where calculating every order gives a unique solution satisfying these conditions. To see that the two solutions are identical, we expand $\exp \mathcal{S}$ into Taylor series. In the resulting series every term containing a product of an even number of terms of $\mathcal{S}$ is a Hermitian, block-diagonal matrix, while every term containing a product of an odd number of terms of $\mathcal{S}$ is an anti-Hermitian block off-diagonal matrix. Therefore $\exp \mathcal{S}$ has the same structure as $\mathcal{U}$ above. Because both series are fixed by the hermiticity constraints on their block structure, we conclude that $\exp \mathcal{S}$ from conventional Schrieffer–Wolff transformation is identical to $\mathcal{U}$ found by our algorithm.

3.5 Extra optimization: common subexpression elimination

While the algorithm of Sec. 3.3 satisfies our requirements, we improve it further by reusing products that are needed in several places, such that the total number of matrix multiplications is reduced. Firstly, we rewrite the expressions for $\mathcal{Z}$ in Eq. (23) and $\tilde{\mathcal{H}}$ in Eq. (22) by utilizing the Hermitian conjugate of $\mathcal{U}'^\dagger\mathcal{X}$ without recomputing it:

$$\mathcal{Z} = \frac{1}{2}\left[(-\mathcal{U}'^\dagger\mathcal{X}) - \text{h.c.}\right], \quad \tilde{\mathcal{H}} = \mathcal{H}_S + \mathcal{U}^\dagger\mathcal{H}'_R\mathcal{U} - (\mathcal{U}'^\dagger\mathcal{X} + \text{h.c.})/2 - \mathcal{Y}_S,$$

where h.c. is the Hermitian conjugate, and $\mathcal{Z}$ drops out from $\tilde{\mathcal{H}}$ because it is antihermitian. Additionally, we reuse the repeated $\mathcal{A} \equiv \mathcal{H}'_R\mathcal{U}'$ in

$$\mathcal{U}^\dagger\mathcal{H}'_R\mathcal{U} = \mathcal{H}'_R + \mathcal{A} + \mathcal{A}^\dagger + \mathcal{U}'^\dagger\mathcal{A}. \tag{27}$$

Next, we observe that some products from the $\mathcal{U}^\dagger\mathcal{H}_R\mathcal{U}$ term appear both in $\mathcal{X}$ in Eq. (24) and in $\tilde{\mathcal{H}}$ (22). To avoid recomputing these products, we introduce $\mathcal{B} = \mathcal{X} - \mathcal{H}'_R - \mathcal{A}$ and define the recursive algorithm using $\mathcal{B}$ instead of $\mathcal{X}$. With this definition, we compute the remaining part of $\mathcal{B}$ as:

$$\begin{aligned}
\mathcal{B}_R &= \left[\mathcal{Y} + \mathcal{Z} - \mathcal{H}'_R - \mathcal{A}\right]_R \\
&= \left[\mathcal{A}^\dagger + \mathcal{U}'^\dagger\mathcal{A} - \mathcal{U}'^\dagger\mathcal{X}\right]_R \\
&= \left[\mathcal{U}'^\dagger\mathcal{H}'_R + \mathcal{U}'^\dagger\mathcal{A} - \mathcal{U}'^\dagger\mathcal{X}\right]_R \\
&= -(\mathcal{U}'^\dagger\mathcal{B})_R,
\end{aligned} \tag{28}$$

where we also used Eq. (24) and the definition of $\mathcal{A}$. The selected part of $\mathcal{B}$, on the other hand, is given by

$$\begin{aligned}
\mathcal{B}_S &= \left[\mathcal{X} - \mathcal{H}'_R - \mathcal{A}\right]_S \\
&= \left[\frac{1}{2}\left[(-\mathcal{U}'^\dagger\mathcal{X}) - \text{h.c.}\right] + \mathcal{Y} - \mathcal{A}\right]_S \\
&= \left[\frac{1}{2}\left[(-\mathcal{U}'^\dagger[\mathcal{X} - \mathcal{H}'_R - \mathcal{A}]) - \text{h.c.}\right] + \mathcal{Y} - \frac{1}{2}\left[\mathcal{A}^\dagger + \mathcal{A}\right] + \frac{1}{2}\left[(-\mathcal{U}'^\dagger\mathcal{A}) - \text{h.c.}\right]\right]_S \\
&= \left[\frac{1}{2}\left[(-\mathcal{U}'^\dagger\mathcal{B}) - \text{h.c.}\right] + \left[\mathcal{V}\mathcal{H}'_S + \text{h.c}\right] - \frac{1}{2}\left[\mathcal{A}^\dagger + \text{h.c.}\right]\right]_S,
\end{aligned} \tag{29}$$

where we used Eq. (23) and that $\mathcal{U}'^\dagger\mathcal{A}$ is Hermitian. Using $\mathcal{B}$ changes the relation for $\mathcal{V}$ in Eq. (26) to

$$[\mathcal{V}, H_0] = \left(\mathcal{B} - \mathcal{H}' - \mathcal{A} - [\mathcal{V}, \mathcal{H}'_S]\right)_R. \tag{30}$$

Finally, we combine Eq. (22), Eq. (27), Eq. (29) and Eq. (28) to obtain the final expression for the effective Hamiltonian:

$$\tilde{\mathcal{H}}_S = \mathcal{H}_S + \frac{1}{2}\left[\mathcal{A} - \mathcal{U}'^\dagger\mathcal{B} + 2\mathcal{V}\mathcal{H}'_S + \text{h.c.}\right]_S. \tag{31}$$

Together with the series $\mathcal{U}'$ in Eqs. (19,30), $\mathcal{A} = \mathcal{H}'_R\mathcal{U}'$, and $\mathcal{B}$ in Eqs. (29,28), this equation defines the optimized algorithm.

4 Implementation

4.1 The data structure for block operator series

The optimized algorithm from the previous section requires constructing 14 operator series, whose elements are computed using a collection of recurrence relations. This warrants defining a specialized data structure suitable for this task that represents a multidimensional series

of operators. Because the recurrent relations are block-wise, the data structure needs to keep track of separate blocks. In order to support varied use cases, the actual representation of the operators needs to be flexible: the block may be dense arrays, sparse matrices, symbolic expressions, or more generally any object that defines addition and multiplication. Finally, the series needs to be queryable by order and block, so that it supports a block-wise multivariate Cauchy product—the main operation in the algorithm.

The most straightforward way to implement a perturbation theory calculation is to write a function that has the desired order as an argument, computes the series up to that order, and returns the result. This makes it hard to reuse already computed terms for a new computation, and becomes complicated to implement in the multidimensional case when different orders in different perturbations are needed. We find that a recursive approach addresses these issues: within this paradigm, each series needs to define how its entries depend on lower-order terms.

To address these requirements, we define a `BlockSeries` Python class and use it to represent the series of $\mathcal{U}$, $\mathcal{H}$, and $\tilde{\mathcal{H}}$, as well as the intermediate series used to define the algorithm. The objects of this class are equipped with a function to compute their elements and it stores the already computed results in a dictionary. Storing the results for reuse is necessary to optimize the evaluation of higher order terms and it allows to request additional orders without restarting the computation. For example, the definition of the `BlockSeries` for $\tilde{\mathcal{H}}$ has the following form:

```
H_tilde = BlockSeries(
    shape=(2, 2),  # 2x2 block matrix
    n_infinite=n,  # number of perturbative parameters
    eval=compute_H_tilde,  # function to compute the elements
    name="H_tilde",
    dimension_names=("lambda", ...),  # parameter names
)
```

Here `compute_H_tilde` is a function implementing Eq. (31) by querying other series objects. Calling `H_tilde[0, 0, 2]`, the second order perturbation $\sim \lambda^2$ of the *AA* block, then does the following:

1. Evaluates `compute_H_tilde(0, 0, 2)` if it is not already computed.
2. Stores the evaluation result in a dictionary.
3. Returns the result.

To conveniently access multiple orders at once, we implement NumPy array indexing so that `H_tilde[0, 0, :3]` returns a NumPy masked array array with the orders $\sim \lambda^0$, $\sim \lambda^1$, and $\sim \lambda^2$ of the *AA* block. The masking allows to support a common use case where some orders of a series are zero, so that they are omitted from the computations. We expect that the `BlockSeries` data structure is suitable to represent a broad class of perturbative calculations, and we plan to extend it to support more advanced features in the future.

We utilize `BlockSeries` to implement multiple other optimizations. For example, we exploit Hermiticity when computing the Cauchy product of $U'^{\dagger}U'$ in Eq. (19), by only evaluating half of the matrix products, and then complex conjugate the result to obtain the rest. Similarly, for Hermitian and anti-Hermitian series, like the off-diagonal blocks of $\mathcal{U}'$, we only compute the *AB* blocks, and use the conjugate transpose to obtain the *BA* blocks. This approach should also allow us to implement efficient handling of symmetry-constrained Hamiltonians, where some blocks either vanish or are equal to other blocks due to a symmetry. Moreover, using `BlockSeries` with custom objects yields additional information about the algorithm and accommodates its further development. Specifically, we have used a custom object with a counter

to measure the algorithm complexity (see also Sec. 5) and to determine which results are only used once so that they can be immediately discarded from storage.

4.2 The implicit method for large sparse Hamiltonians

A distinguishing feature of Pymablock is its ability to handle large sparse Hamiltonians, that are too costly to diagonalize, as illustrated in Sec. 2.3. Specifically, we consider the situations when the size N_E of the subspace of interest—explicit subspace—is small compared to the entire Hilbert space, so that obtaining the basis Ψ_E of the explicit subspace is feasible using sparse diagonalization. The projector on this subspace $P_E = \Psi_E^\dagger \Psi_E$ is then a low-rank matrix, a property that we exploit to avoid constructing the matrix representation of operators in the other, implicit, subspace.

The key tool to solve this problem is the projector approach introduced in Ref. [47], which introduces an equivalent extended Hamiltonian using the projector $P_I = 1 - P_A$ onto the implicit subspace:

$$\bar{\mathcal{H}} = \begin{pmatrix} \Psi_E^\dagger \mathcal{H} \Psi_E & \Psi_E^\dagger \mathcal{H} P_I \\ P_I \mathcal{H} \Psi_E & P_I \mathcal{H} P_I \end{pmatrix}. \tag{32}$$

In other words, the explicit subspace is written in the basis of Ψ_E, while the basis of the implicit subspace is the same as the original complete basis of $\mathcal{H}$ to preserve its sparsity. The extended Hamiltonian projects out the E-degrees of freedom from the implicit subspace to avoid duplicate solutions in $\bar{\mathcal{H}}$, which introduces N_E eigenvectors with zero eigenvalues. Introducing $\bar{\mathcal{H}}$ allows to multiply by operators of a form $P_I H_{\mathbf{n}} P_I$ efficiently by using the low-rank structure of P_E. In the code we represent the operators of the implicit subspace as `LinearOperator` objects from the SciPy package [41], enabled by the ability of the `BlockSeries` to store arbitrary objects. Storing the remaining blocks of $\bar{\mathcal{H}}$ as dense matrices—efficient because these are small and dense—finishes the implementation of the Hamiltonian.

To solve the Sylvester's equation we write it for every row of $V_{\mathbf{n}}^{EI}$ separately:

$$V_{\mathbf{n},ij}^{EI}(E_i - H_0) = Y_{\mathbf{n},j}^{EI}. \tag{33}$$

This equation has a solution despite $E_i - H_0$ not being invertible because $Y_{\mathbf{n}}^{EI} P_A = 0$. We solve this equation using the MUMPS sparse solver [48,49], which prepares an efficient sparse LU-decomposition of $E_i - H_0$, or the KPM approximation of the Green's function [50]. Both methods work on sparse Hamiltonians with millions of degrees of freedom.

4.3 Code generation

An efficient computation of a perturbative block-diagonalization requires a significant amount of repeated optimizations. These include keeping track of the Hermiticity of involved series, applying the simplifications due to block-diagonalization and the presence of only two blocks, or deletion of series terms that are only used once. To separate the conceptual definition of the algorithm from these optimizations, we designed the code generation system that accepts a high-level description of the algorithm written in a domain-specific language and outputs the optimized Python code using the Python parser and the manipulation of the Python abstract syntax tree. For example, the definition of the series $\mathcal{B}$ from Eqs. (29,28) is written as:

```
with "B":
    start = 0
    if diagonal:
        ("U'† @ B" - "U'† @ B".adj + "H'_offdiag @ U'" + "H'_offdiag @ U'".adj) /
        ↪ -2
```

```
    if diagonal:
        zero if commuting_blocks[index[0]] else "V @ H'_diag" + "V @ H'_diag".adj
    if offdiagonal:
        -"U'† @ B"
```

The corresponding compiled function for evaluating the terms of $\mathcal{B}$ begins with

```
def series_eval(*index):
    which = linear_operator_series if use_linear_operator[index[:2]] else series
    result = zero
    if index[0] == index[1]:
        result = _zero_sum(
            result,
            diag(
                _safe_divide(
                    _zero_sum(
                        which["U'† @ B"][index], -Dagger(which["U'† @ B"][index]),
                        which["H'_offdiag @ U'"][index],
                        Dagger(which["H'_offdiag @ U'"][index]),
                    ), -2,
                ), index,
            ),
        )
    ...
```

Here we only show the beginning of the generated function to illustrate the correspondence between the high-level description and the generated code.

The code generation system has accommodated multiple rewrites of the algorithm during the development. We anticipate that it will enable treating different types of perturbative computations or other related algorithms, such as the derivative removal by adiabatic gate (DRAG) algorithm [51, 52]. Contrary to the perturbation theory setting, DRAG requires that the time-dependent Hamiltonian is block-diagonal in the rotating frame, and it achieves this goal by adding a series of corrections to the original Hamiltonian. Its overall setting, however, is similar to time-dependent perturbation theory in that it amounts to solving a system of recurrent algebraic equations. Our preliminary research already demonstrates that our code generation framework allows for a generalization of our work to the time-dependent perturbation theory, and we are confident that it applies to the DRAG algorithm as well.

5 Benchmark

To the best of our knowledge, there are no other packages implementing arbitrary order quasi-degenerate perturbation theory. Literature references provide explicit expressions for the 2×2 effective Hamiltonian up to fourth order, together with the procedure for obtaining higher order expressions [5]. Because the full reference expressions are lengthy,[6] we do not provide them, but for example at 4-th order the effective Hamiltonian is a sum of several expressions of the form:

$$\sum_{m''m'''l} \frac{H'_{mm''}H'_{m''m'''}H'_{m'''l}H'_{lm'}}{(E_{m''}-E_l)(E_{m'''}-E_l)(E_m-E_l)}, \tag{34}$$

where the m-indices label states from the A-subspace and l-indices label the states from the B-subspace. More generally, at n-th order each term is a product of n matrix elements of the

[6] The full expression takes almost a page of text.

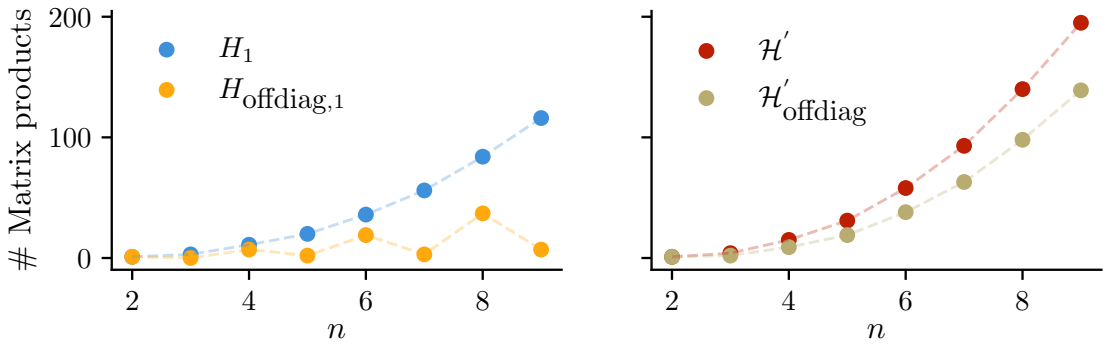

Figure 5: Matrix products required to compute $\tilde{H}_n^{AA}$ for a dense and block off-diagonal first-order perturbation (left) and a dense and block off-diagonal perturbative series with terms of all orders present (right).

Hamiltonian and $n-1$ energy denominators. Directly carrying out the summation over all the states requires $\mathcal{O}(N_A^2 N_B^{n-1})$ operations, where N_A and N_B are the number of states in the two subspaces. In other words, the direct computation scales worse than a matrix product with the problem size. Formulating Eq. (34) as $n-1$ matrix products combined with $n-1$ solutions of Sylvester's equation, brings this complexity down to $\mathcal{O}((n-1)\times N_A N_B^2)$. This optimization, together with the hermiticity of the sum, allows us to evaluate the reference expressions for the effective Hamiltonian for 2-nd, 3-rd, and 4-th order using 1, 4, and 27 matrix products, respectively. Pymablock's algorithm yields the following expressions for the first four orders of the effective Hamiltonian:[7]

$$
\begin{aligned}
Y_{1,AB} &= H_{1,AB}\,,\\
\tilde{H}_{2,AA} &= H_{1,AB}V_1^\dagger/2 + \text{h.c.}\,,\\
Y_{2,AB} &= V_1 H_{1,BB} - H_{1,AA}^\dagger V_1\,,\\
\tilde{H}_{3,AA} &= H_{1,AB}V_2^\dagger + \text{h.c.}\,,\\
Y_{3,AB} &= -\frac{V_1 V_1^\dagger H_{1,BA}^\dagger}{2} + V_2 H_{1,BB} - \frac{\left(H_{1,AB}V_1^\dagger + V_1 H_{1,AB}^\dagger\right)V_1}{2} - H_{1,AA}^\dagger V_2\,,\\
\tilde{H}_{4,AA} &= \frac{H_{1,AB}V_3^\dagger}{2} + \frac{V_1 V_1^\dagger\left(H_{1,AB}V_1^\dagger + V_1 H_{1,AB}^\dagger\right)}{8} + \text{h.c.}\,,
\end{aligned}
\tag{35}
$$

where V_n are the solutions of Sylvester's equation with $Y_{n,AB}$ as the right-hand side. These expressions utilize 1, 3, and 11, matrix products to obtain the same orders of the effective Hamiltonian. The advantage of the Pymablock algorithm becomes even more pronounced at higher orders or with multiple perturbative parameters due to the exponential growth of the number of terms in the reference expressions. While finding the optimized implementation from the reference expressions is possible for the 3-rd order, we expect it to be extremely challenging for the 4-th order, and essentially impossible to do manually for higher orders. Moreover, because the `BlockSeries` class tracks absent terms, in practice the number of matrix products depends on the sparsity of the block structure of the perturbation, as shown in Fig. 5.

The efficiency of Pymablock becomes especially apparent when applied to sparse numerical problems, similar to Sec. 2.3. We demonstrate the performance of the implicit method by using it to compute the low-energy spectrum of a large tight-binding model, and compar-

[7] The output is generated by the algorithm, with manual modifications only done for formatting.

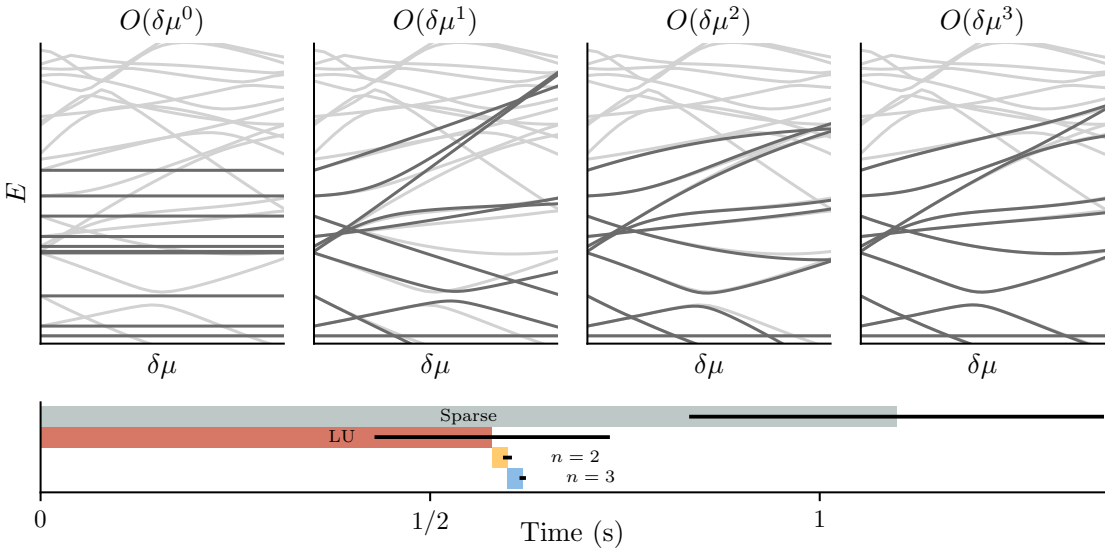

Figure 6: Top panels: band structure of the perturbative effective Hamiltonian (black) of a tight-binding model compared to exact sparse diagonalization (gray). Bottom panel: a comparison of the Pymablock's time cost with sparse diagonalization. Most of the time is spent in the LU decomposition of the Hamiltonian (red). The entire cost of the implicit method is lower than a single sparse diagonalization (gray). The operations of negligible cost are not shown. The bars length corresponds to the average time cost over 40 runs, and the error bars show the standard deviation.

ing Pymablock's time cost to that of sparse diagonalization. We define a 2D square lattice of 52×52 sites with nearest-neighbor hopping and a random onsite potential $\mu(\mathbf{r})$. The perturbation $\delta\mu(\mathbf{r})$ interpolates between two different disorder realizations. For the sake of an illustration, we choose the system's parameters such that the dispersion of the lowest few levels with $\delta\mu$ features avoided crossings and an overall nonlinear shape, whose details are not relevant. Similar to Sec. 2.3, constructing the effective Hamiltonian involves three steps. First, we compute the 10 lowest states of the unperturbed Hamiltonian using sparse diagonalization. Second, `block_diagonalize` computes a sparse LU decomposition of the Hamiltonian at each of the 10 eigenenergies. Third, we compute corrections $\tilde{H}_1$, $\tilde{H}_2$, and $\tilde{H}_3$ to the effective Hamiltonian, each being a 10×10 matrix. Each of these steps is a one-time cost, see Fig. 6. Finally, to compare the perturbative calculation to sparse diagonalization, we construct the effective Hamiltonian $\tilde{H} = H_0 + \delta\mu\tilde{H}_1 + \delta\mu^2\tilde{H}_2 + \delta\mu^3\tilde{H}_3$ and diagonalize it to obtain the low-energy spectrum for a range of $\delta\mu$. This has a negligible cost compared to constructing the series. The comparison is shown in Fig. 6. We observe that while the second order results are already very close to the exact spectrum, the third order corrections fully reproduce the sparse diagonalization. At the same time, the entire cost of computing the perturbative band structure for a range of $\delta\mu$ is lower than computing a single additional sparse diagonalization.

6 Conclusion

We developed an algorithm for constructing an effective Hamiltonian that combines advantages of different perturbative expansions. The main building block of our approach is a set of recurrence relations that define several series that depend on each other and combine into

the effective Hamiltonian. Our algorithm constructs the same effective Hamiltonians as the Schrieffer–Wolff transformation [1] in the case of 2 subspaces, while keeping the linear scaling per extra order similar to the density matrix perturbation theory [16,17] or the non-orthogonal perturbation theory [18]. Its expressions minimize the number of matrix multiplications per order, making it appealing both for symbolic and numerical computations. Pymablock's algorithm performs multi-block diagonalization and selective diagonalization with a single optimized algorithm.

We provide a Python implementation of the algorithm in the Pymablock package [53]. The package is thoroughly tested (95% test coverage as of version 2.1), becoming a reliable tool for constructing effective Hamiltonians that combine multiple perturbations to high orders. The core of the Pymablock interface is the `BlockSeries` class that handles arbitrary objects as long as they support algebraic operations. This enables Pymablock's construction of effective models for large tight-binding models using its implicit method as well as for second quantized Hamiltonians. As of version 2.1, applying Pymablock to second quantized Hamiltonians requires the user to provide a custom solver of the Lyapunov equation, which we plan to streamline in future versions. It also allows Pymablock to solve both symbolic and numerical problems in diverse physical settings, and potentially to incorporate it into existing packages, such as scqubits [35], QuTiP [54,55], or dft2kp [56].

Beyond the Schrieffer–Wolff transformation, the Pymablock package provides a foundation for defining other perturbative expansions. We anticipate extending it to time-dependent problems, where the different regimes of the time-dependent drive modify the recurrence relations that need to be solved [10,57]. Applying the same framework to problems with weak position dependence would allow to construct a nonlinear response theory of quantum materials. These two extensions are active areas of research [7,46,51,52,58,59]. Finally, we expect that in the many-particle context the same framework supports implementing different flavors of diagrammatic expansions.

Acknowledgments

We thank Valla Fatemi and Antonio Manesco for feedback on the manuscript. We also thank David P. DiVincenzo for motivating and helpful discussions regarding the multi-block diagonalization algorithm.

Data availability The code used to produce the reported results is available on Zenodo [53].

Author contributions A. R. A. had the initial idea and oversaw the project. All authors developed the algorithm. I. A. D., S. M., H. K. K, and A. R. A. wrote the package. I. A. D. and A. R. A. wrote the paper.

Funding information This research was supported by the Netherlands Organization for Scientific Research (NWO/OCW) as part of the Frontiers of Nanoscience program, a NWO VIDI grant 016.Vidi.189.180, and OCENW.GROOT.2019.004. D.V. acknowledges funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy through the Würzburg-Dresden Cluster of Excellence on Complexity and Topology in Quantum Matter – ct.qmat (EXC 2147, project-ids 390858490 and 392019).

References

[1] J. R. Schrieffer and P. A. Wolff, *Relation between the Anderson and Kondo Hamiltonians*, Phys. Rev. **149**, 491 (1966), doi:10.1103/PhysRev.149.491.

[2] S. Bravyi, D. P. DiVincenzo and D. Loss, *Schrieffer-Wolff transformation for quantum many-body systems*, Ann. Phys. **326**, 2793 (2011), doi:10.1016/j.aop.2011.06.004.

[3] P.-O. Löwdin, *Studies in perturbation theory. IV. Solution of eigenvalue problem by projection operator formalism*, J. Math. Phys. **3**, 969 (1962), doi:10.1063/1.1724312.

[4] J. M. Luttinger and W. Kohn, *Motion of electrons and holes in perturbed periodic fields*, Phys. Rev. **97**, 869 (1955), doi:10.1103/PhysRev.97.869.

[5] R. Winkler, *Spin-orbit coupling effects in two-dimensional electron and hole systems*, Springer, Berlin, Heidelberg, Germany, ISBN 9783540011873 (2003), doi:10.1007/b13586.

[6] E. McCann and M. Koshino, *The electronic properties of bilayer graphene*, Rep. Prog. Phys. **76**, 056503 (2013), doi:10.1088/0034-4885/76/5/056503.

[7] B. A. Bernevig, Z.-D. Song, N. Regnault and B. Lian, *Twisted bilayer graphene. I. Matrix elements, approximations, perturbation theory, and a $k \cdot p$ two-band model*, Phys. Rev. B **103**, 205411 (2021), doi:10.1103/PhysRevB.103.205411.

[8] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson and W. D. Oliver, *A quantum engineer's guide to superconducting qubits*, Appl. Phys. Rev. **6**, 021318 (2019), doi:10.1063/1.5089550.

[9] J. Romhányi, G. Burkard and A. Pályi, *Subharmonic transitions and Bloch-Siegert shift in electrically driven spin resonance*, Phys. Rev. B **92**, 054422 (2015), doi:10.1103/PhysRevB.92.054422.

[10] M. Malekakhlagh, E. Magesan and D. C. McKay, *First-principles analysis of cross-resonance gate operation*, Phys. Rev. A **102**, 042605 (2020), doi:10.1103/physreva.102.042605.

[11] A. Petrescu, C. Le Calonnec, C. Leroux, A. Di Paolo, P. Mundada, S. Sussman, A. Vrajitoarea, A. A. Houck and A. Blais, *Accurate methods for the analysis of strong-drive effects in parametric gates*, Phys. Rev. Appl. **19**, 044003 (2023), doi:10.1103/physrevapplied.19.044003.

[12] J. H. Van Vleck, *On σ-type doubling and electron spin in the spectra of diatomic molecules*, Phys. Rev. **33**, 467 (1929), doi:10.1103/PhysRev.33.467.

[13] I. Shavitt and L. T. Redmon, *Quasidegenerate perturbation theories. A canonical Van Vleck formalism and its relationship to other approaches*, J. Chem. Phys. **73**, 5711 (1980), doi:10.1063/1.440050.

[14] D. J. Klein, *Degenerate perturbation theory*, J. Chem. Phys. **61**, 786 (1974), doi:10.1063/1.1682018.

[15] K. Suzuki and R. Okamoto, *Degenerate perturbation theory in quantum mechanics*, Prog. Theor. Phys. **70**, 439 (1983), doi:10.1143/PTP.70.439.

[16] R. McWeeny, *Perturbation theory for the Fock-Dirac density matrix*, Phys. Rev. **126**, 1028 (1962), doi:10.1103/PhysRev.126.1028.

[17] L. A. Truflandier, R. M. Dianzinga and D. R. Bowler, *Notes on density matrix perturbation theory*, J. Chem. Phys. **153**, 164105 (2020), doi:10.1063/5.0022244.

[18] C. Bloch, *Sur la théorie des perturbations des états liés*, Nucl. Phys. **6**, 329 (1958), doi:10.1016/0029-5582(58)90116-0.

[19] F. Wegner, *Flow-equations for Hamiltonians*, Ann. Phys. **506**, 77 (1994), doi:10.1002/andp.19945060203.

[20] S. Kehrein, *The flow equation approach to many-particle systems*, Springer, Berlin, Heidelberg, Germany, ISBN 9783540340676 (2006), doi:10.1007/3-540-34068-8.

[21] C. Knetter and G. S. Uhrig, *Perturbation theory by flow equations: Dimerized and frustrated $S = 1/2$ chain*, Eur. Phys. J. B - Condens. Matter Complex Syst. **13**, 209 (2000), doi:10.1007/s100510050026.

[22] J. Oitmaa, C. Hamer and W. Zheng, *Series expansion methods for strongly interacting lattice models*, Cambridge University Press, Cambridge, UK, ISBN 9780521842426 (2006), doi:10.1017/CBO9780511584398.

[23] H. Krull, N. A. Drescher and G. S. Uhrig, *Enhanced perturbative continuous unitary transformations*, Phys. Rev. B **86**, 125113 (2012), doi:10.1103/PhysRevB.86.125113.

[24] J. Wurtz, P. W. Claeys and A. Polkovnikov, *Variational Schrieffer-Wolff transformations for quantum many-body dynamics*, Phys. Rev. B **101**, 014302 (2020), doi:10.1103/PhysRevB.101.014302.

[25] Z. Zhang, Y. Yang, X. Xu and Y. Li, *Quantum algorithms for Schrieffer-Wolff transformation*, Phys. Rev. Res. **4**, 043023 (2022), doi:10.1103/PhysRevResearch.4.043023.

[26] I. N. H. Mankodi and D. P. DiVincenzo, *Perturbative power series for block diagonalisation of Hermitian matrices*, (arXiv preprint) doi:10.48550/arXiv.2408.14637.

[27] E. Magesan and J. M. Gambetta, *Effective Hamiltonian models of the cross-resonance gate*, Phys. Rev. A **101**, 052308 (2020), doi:10.1103/PhysRevA.101.052308.

[28] X. Xu, M., C. Vignes, M. H. Ansari and J. Martinis, *Lattice Hamiltonians and stray interactions within quantum processors*, Phys. Rev. Appl. **22**, 064030 (2024), doi:10.1103/PhysRevApplied.22.064030.

[29] C. Knetter, K. P. Schmidt and G. S. Uhrig, *The structure of operators in effective particle-conserving models*, J. Phys. A: Math. Gen. **36**, 7889 (2003), doi:10.1088/0305-4470/36/29/302.

[30] A. Blais, R.-S. Huang, A. Wallraff, S. M. Girvin and R. J. Schoelkopf, *Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation*, Phys. Rev. A **69**, 062320 (2004), doi:10.1103/PhysRevA.69.062320.

[31] G. Zhu, D. G. Ferguson, V. E. Manucharyan and J. Koch, *Circuit QED with fluxonium qubits: Theory of the dispersive regime*, Phys. Rev. B **87**, 024510 (2013), doi:10.1103/PhysRevB.87.024510.

[32] X. Li et al., *Tunable coupler for realizing a controlled-phase gate with dynamically decoupled regime in a superconducting circuit*, Phys. Rev. Appl. **14**, 024070 (2020), doi:10.1103/PhysRevApplied.14.024070.

[33] A. Blais, A. L. Grimsmo, S. M. Girvin and A. Wallraff, *Circuit quantum electrodynamics*, Rev. Mod. Phys. **93**, 025005 (2021), doi:10.1103/RevModPhys.93.025005.

[34] E. A. Sete, A. Q. Chen, R. Manenti, S. Kulshreshtha and S. Poletto, *Floating tunable coupler for scalable quantum computing architectures*, Phys. Rev. Appl. **15**, 064063 (2021), doi:10.1103/PhysRevApplied.15.064063.

[35] P. Groszkowski and J. Koch, *Scqubits: A Python package for superconducting qubits*, Quantum **5**, 583 (2021), doi:10.22331/q-2021-11-17-583.

[36] S. P. Chitta, T. Zhao, Z. Huang, I. Mondragon-Shem and J. Koch, *Computer-aided quantization and numerical analysis of superconducting circuits*, New J. Phys. **24**, 103020 (2022), doi:10.1088/1367-2630/ac94f2.

[37] B. Li, T. Calarco and F. Motzoi, *Nonperturbative analytical diagonalization of Hamiltonians with application to circuit QED*, PRX Quantum **3**, 030313 (2022), doi:10.1103/PRXQuantum.3.030313.

[38] A. Melo, T. Tanev and A. R. Akhmerov, *Greedy optimization of the geometry of Majorana Josephson junctions*, SciPost Phys. **14**, 047 (2023), doi:10.21468/SciPostPhys.14.3.047.

[39] A. Meurer et al., *SymPy: Symbolic computing in Python*, PeerJ Comput. Sci. **3**, e103 (2017), doi:10.7717/peerj-cs.103.

[40] C. W. Groth, M. Wimmer, A. R. Akhmerov and X. Waintal, *Kwant: A software package for quantum transport*, New J. Phys. **16**, 063065 (2014), doi:10.1088/1367-2630/16/6/063065.

[41] P. Virtanen et al., *SciPy 1.0: Fundamental algorithms for scientific computing in Python*, Nat. Methods **17**, 261 (2020), doi:10.1038/s41592-019-0686-2.

[42] S. Savitz and G. Refael, *Stable unitary integrators for the numerical implementation of continuous unitary transformations*, Phys. Rev. B **96**, 115129 (2017), doi:10.1103/PhysRevB.96.115129.

[43] R. McWeeny, *Self-consistent perturbation thèory*, Chem. Phys. Lett. **1**, 567 (1968), doi:10.1016/0009-2614(68)85047-X.

[44] L. S. Cederbaum, J. Schirmer and H.-D. Meyer, *Block diagonalisation of Hermitian matrices*, J. Phys. A: Math. Gen. **22**, 2427 (1989), doi:10.1088/0305-4470/22/13/035.

[45] G. T. Landi, *Eigenoperator approach to Schrieffer-Wolff perturbation theory and dispersive interactions*, (arXiv preprint) doi:10.48550/arXiv.2409.10656.

[46] L. Reascos, G. F. Diotallevi and M. Benito, *Universal solution to the Schrieffer-Wolff transformation generator*, (arXiv preprint) doi:10.48550/arXiv.2411.11535.

[47] M. Irfan, S. R. Kuppuswamy, D. Varjas, P. M. Perez-Piskunow, R. Skolasinski, M. Wimmer and A. R. Akhmerov, *Hybrid kernel polynomial method*, (arXiv preprint) doi:10.48550/arXiv.1909.09649.

[48] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl. **23**, 15 (2001), doi:10.1137/S0895479899358194.

[49] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent and S. Pralet, *Hybrid scheduling for the parallel solution of linear systems*, Parallel Comput. **32**, 136 (2006), doi:10.1016/j.parco.2005.07.004.

[50] A. Weiße, G. Wellein, A. Alvermann and H. Fehske, *The kernel polynomial method*, Rev. Mod. Phys. **78**, 275 (2006), doi:10.1103/RevModPhys.78.275.

[51] F. Motzoi, J. M. Gambetta, P. Rebentrost and F. K. Wilhelm, *Simple pulses for elimination of leakage in weakly nonlinear qubits*, Phys. Rev. Lett. **103**, 110501 (2009), doi:10.1103/PhysRevLett.103.110501.

[52] L. S. Theis, F. Motzoi, S. Machnes and F. K. Wilhelm, *Counteracting systems of diabaticities using DRAG controls: The status after 10 years*, Europhys. Lett. **123**, 60001 (2018), doi:10.1209/0295-5075/123/60001.

[53] I. Araya Day, S. Miles, H. K. Kerstens, D. Varjas and A. R. Akhmerov, *Pymablock*, Zenodo (2024), doi:10.5281/zenodo.14188554.

[54] J. R. Johansson, P. D. Nation and F. Nori, *QuTiP: An open-source Python framework for the dynamics of open quantum systems*, Comput. Phys. Commun. **183**, 1760 (2012), doi:10.1016/j.cpc.2012.02.021.

[55] J. R. Johansson, P. D. Nation and F. Nori, *QuTiP 2: A Python framework for the dynamics of open quantum systems*, Comput. Phys. Commun. **184**, 1234 (2013), doi:10.1016/j.cpc.2012.11.019.

[56] J. V. V. Cassiano, A. de Lelis Araújo, P. E. Faria Junior and G. J. Ferreira, *DFT2kp: Effective kp models from ab-initio data*, SciPost Phys. Codebases 25 (2024), doi:10.21468/SciPostPhysCodeb.25.

J. V. V. Cassiano, A. de Lelis Araújo, P. E. Faria Junior and G. J. Ferreira, *Codebase release 0.0 for DFT2kp*, SciPost Phys. Codebases 25-r0.0 (2024), doi:10.21468/SciPostPhysCodeb.25-r0.0.

[57] M. Rodriguez-Vega, M. Lentz and B. Seradjeh, *Floquet perturbation theory: Formalism and application to low-frequency limit*, New J. Phys. **20**, 093022 (2018), doi:10.1088/1367-2630/aade37.

[58] J. Venkatraman, X. Xiao, R. G. Cortiñas, A. Eickbusch and M. H. Devoret, *Static effective Hamiltonian of a rapidly driven nonlinear system*, Phys. Rev. Lett. **129**, 100601 (2022), doi:10.1103/PhysRevLett.129.100601.

[59] Y. Xu and L. Guo, *Perturbative framework for engineering arbitrary Floquet Hamiltonian*, (arXiv preprint) doi:10.48550/arXiv.2410.10467.