We thank the refereee for their time and work in reviewing our manuscript. Here below we address point by point all the issues raised in their reports.

> *This paper presents the current status in an exact diagonalisation impurity solver EDIpack. It provides a detailed exposition of the multiple interfaces as well as a number of tutorial-style examples that provide details that will be appreciated to new users of the package. I believe that providing robust and well-tested computer codes, as well as high quality documentation, are a big service to the community. I wholeheartedly recommend this work for publication.*

We sincerely thank the referee for their positive and encouraging comments. We are very pleased to hear that the exposition, interfaces, and tutorial examples were found to be helpful and of value to new users. We are grateful for your recommendation for publication and your support for open and well-maintained scientific software.

> *On p. 8, function {\tt ed_set_Hloc} is introduced, but without much further information. Maybe the reader could be referred to some later section where its use is illustrated.*

We acknowledge the referee's suggestion. The function `ed_set_Hloc`, introduced in Section 3.1.1, is employed to define the impurity Hamiltonian. At this stage of the manuscript, its purpose is solely described: setting the local non-interacting component of the impurity Hamiltonian, without further elaboration.

To enhance clarity and facilitate reader comprehension of its practical application, we have revised the text at this location. Specifically, we have included a forward reference to Section 5, where `ed_set_Hloc` is explicitly utilized in various application examples. These examples encompass the Hubbard model on the Bethe lattice (Section 5.1), the interacting Bernevig-Hughes-Zhang model (Section 5.5), and the Kane-Mele-Hubbard model (Section 5.6). These instances aim to elucidate both the structure of the input arguments and the context in which this function is commonly employed.

> *On p. 9, the discussion about the phonon cutoff sounds potentially a bit misleading. The cutoff is surely problem dependent, and there are (generalized) cases where the shift A_m cannot be considered as a free parameter.*

We thank the referee for this observation. Indeed, the cutoff on the number of phonon excitations is inherently problem dependent and should be carefully adjusted in light of the specific electron-phonon coupling regime, the phonon frequency, and the interplay with electronic correlations. Furthermore, we agree that the shift A_m, while often treated as a

tunable parameter in simplified models, is not always free to choose. In more general contexts, such as in first-principles-derived Hamiltonians or systems constrained by symmetry or external potentials, A_m may be fixed by physical considerations or derived from the underlying microscopic model.

To reflect this more nuanced detail, we have revised the paragraph on page 9 to acknowledge that the shift A_m is not universally a free parameter. The new text reads:

"In many cases, one can reduce the number of excited phonons in the ground state, and consequently the required cutoff, by appropriately choosing the shift A_m. However, we stress that this is not always possible: in some generalized models or realistic systems derived from microscopic considerations, A_m may be fixed and not a freely tunable parameter. In such cases, the phonon cutoff must be increased accordingly to ensure numerical convergence."

This modification ensures that our discussion does not overgeneralise while aligning with the range of physical contexts in which EDIpack may be applied. We thank the referee for helping us clarify this important point.

> *I find the discussion at the end of page 11 somewhat unclear, especially the terminology. Is this a discussion of consecutive indexing vs. occupation number representation?*

We thank the referee for pointing out this issue. Indeed, the original text aimed to explain how electronic Fock states within a given symmetry sector are indexed and related to their underlying occupation number (bitstring) representation. However, we acknowledge that the terminology and phrasing in the original submission may have been ambiguous.

To clarify this point, we have revised the final paragraph of page 11 in the current version of the manuscript. The updated text now reads:

"From a computational perspective, constructing a symmetry sector amounts to defining an injective map $\mathcal{M} : \mathcal{S}_{\vec{Q}} \to \mathcal{F}_e$ that relates the consecutive indexing of the electronic states $|i\rangle$ within a given symmetry sector $\mathcal{S}_{\vec{Q}}$ to the Fock states $|I_e\rangle$ in the electronic Fock space in their number representation. This map is typically implemented as a rank-1 integer array, whose size corresponds to the dimension of the sector $D^{\mathcal{S}}$."

This new formulation replaces the previous wording and makes explicit the distinction between the consecutive indexing of the sector states $|i\rangle$ and the Fock states in the integer-based occupation number representation $|I_e\rangle$. In doing so, we clarify that the mapping $\mathcal{M}$ is what connects the two encodings of the same physical state. We believe the revised text should eliminate the ambiguity the referee noted.

> *The motivation of including the code on p. 12 is unclear. What is the intention here?*

We thank the referee for raising this point. The code snippet on page 12 was originally included to illustrate the concrete implementation of the mapping from Fock states to symmetry sectors for different values of `ed_mode`. Our goal was to make explicit the internal logic used to construct the quantum number-conserving subspaces, bridging the gap between formal description and practical realization. Moreover, this code snippet is useful to understand the structure of the `sparse_map` used to implement fast-algorithm for the evaluation of the reduced impurity density matrix in section 3.5.2.

In response to the referee's comment, we have slightly revised the manuscript to clarify this intent. Specifically, we have added the following introductory sentence to the relevant section:

"The following code excerpts illustrate how the sector construction algorithms differ depending on the value of ed_mode, and serve as a concrete reference for understanding how quantum number constraints are enforced in practice."

> *Top of p. 14: What is global share, what are istart, ishift, iend? I suppose this is Fortran-specific.*

We thank the referee for their close reading and for raising this question concerning the variables `istart`, `iend`, and `ishift`.

We recognise that the earlier reference to the global share was misleading and ultimately unnecessary concept that we have chosen to eliminate for clarity.

The issue at hand concerns the parallel construction of a matrix that is distributed across multiple threads or MPI processes. In such a setting, each thread is responsible for computing and storing only a subset—or "chunk" `Q`—of the full matrix, typically defined as a range of contiguous rows. To maintain consistency with the global indexing of the matrix, while allowing each thread to work independently and with locally indexed arrays, a mapping must be established between global and local indices.

The variables `istart` and `iend` define the global index range of matrix rows assigned to a given thread or process. That is, a given thread will handle only the rows from `istart` to `iend` (inclusive), corresponding to its local share of the full matrix. The variable `ishift` is then used to translate global row indices into local ones, effectively setting `ishift` = `mpiRank` $*$ `Q`. This allows local arrays to be aligned correctly with the globally defined matrix structure.

While the code is written in Fortran, this construction is not specific to the language itself. Rather, it is a general approach for handling parallel matrix distribution in any programming environment where the global matrix is divided into row blocks across multiple processing units. To avoid potential confusion and to make the implementation choices more transparent to the reader in the revised we now write:

"*In a parallel setting, each thread is assigned a contiguous block of matrix rows* $Q = D_S/N_{cpu}$*. We denote the locally stored rows on a given thread by* $q = 1, \dots, Q$*, and relate them to their corresponding global indices* $i = 1, \dots, D_S$ *via the variables \texttt{istart}, \texttt{iend}, and \texttt{ishift}. Here, \texttt{istart} and \texttt{iend} define the global index range* $[q_1, q_Q]$ *assigned to the thread, corresponding to* $q_1 =$ `istart` *and* $q_Q =$ `iend`*. The variable \texttt{ishift}* $=$ `istart` $- 1$ *allows for mapping between the local index* q *and the global index* i *through the simple relation* $i = q +$ `ishift`*. This mapping ensures consistent global indexing while enabling efficient local computation.*"

> *p. 16: GFmatrix is said to be a critical component for high-speed execution. Maybe it could be describe in more detail? In what way is it efficient? What does it mean it is multi-layed? In passing, it would be nice if the capitalization would be uniform, e.g. GFmatrix vs. gfmatrix (I understand that Fortran compiler does not care, but the human reader perhaps does).*

We thank the referee for pointing out the need for a clearer and more detailed description of the `gfmatrix` type, which indeed plays a central role in the performance of the code. In response, we have revised the corresponding section of the manuscript (page 16) to provide a more precise explanation of its structure and efficiency features.

The term multi-layered refers to the hierarchical organization of the `gfmatrix` object, which is essential for storing and accessing the analytic structure of correlation functions, i.e. the weights and the poles, in a flexible and scalable way. Specifically, the data are organized across three conceptual layers: (1) the many-body eigenstates of the Hamiltonian spectrum, (2) the amplitude channels associated to each such state, and (3) the set of poles and weights corresponding to physical excitations from this state. This structure is necessary to correctly represent the Lehmann decomposition of Green's functions and related objects, while still allowing dynamic access and manipulation during calculations.

The efficiency of `gfmatrix` stems from two main features. First, correlation functions do not need to be precomputed and stored on fixed frequency or time grids; instead, their values are computed on-the-fly at arbitrary complex arguments, allowing precise and flexible evaluation as required by different solvers or output formats. Second, memory usage is optimized by avoiding redundant storage. The internal design separates spectral data, and evaluation logic, allowing minimal overhead during the object construction and easing post-processing analysis. We also recognize that the use of the term "high-performance" could be misleading and rephrases the paragraph at page 16 in a more precise way as:

"*The module* `ED_GFMATRIX` *implements the class* `gfmatrix`*, which provides an efficient and flexible representation of DCFs. It is structured as a hierarchical, or {\it multi-layered}, data container that organizes spectral weights and poles across three levels: (i) the initial eigenstate* $|n\rangle$ *of the Hamiltonian spectrum, (ii) the amplitude channel associated with this state, corresponding to the applied operator, and (iii) the set of excitations from* $|n\rangle$*, each*

characterized by a pole and its corresponding weight.
This structure allows the `gfmatrix` *class to simultaneously handle multiple initial states and operators, enabling on-the-fly evaluation of DCFs at arbitrary complex frequencies* $z \in \mathbb{C}$*. By avoiding precomputed grids and redundant storage this approach minimizes memory footprint and computational overhead. As a result,* `gfmatrix` *is a core component of EDIpack, essential for efficient memory management, consistent storage of spectral data, and high-performance post-processing.*"

We also thank the referee for pointing out inconsistencies in capitalization. We have now ensured that the identifier is uniformly written as **gfmatrix** throughout the manuscript.

> *A trick for computing off-diagonal functions is presented on p. 21. Doesn't this require switching to complex-valued floating points even in cases where the Hamiltonian is purely real?*

We thank the referee for this observation. It is indeed correct that the general strategy presented for computing off-diagonal DCFs, as described on page 21, involves constructing linear combinations of operators, which in the most general case would require working with complex-valued floating point arithmetic—even if the Hamiltonian itself is real.

However, in the special case where the Hamiltonian is real and symmetric, the resulting Green's functions are symmetric under exchange of orbital indices, i.e., $G_{\alpha\beta}(z) = G_{\beta\alpha}(z)$. This property can be exploited to compute off-diagonal elements using only real-valued linear combinations of operators. In such cases, all required DCFs can be obtained exploiting memory advantages of real-valued computations.

In the revised manuscript we clarify this point by writing in the **normal** paragraph of the section 3.8:

"*The sector Hamiltonian matrix in this mode is assumed to be real symmetric, which simplifies the evaluation of the off-diagonal terms by exploiting the following symmetry under orbital exchange* $G_{\alpha\beta\sigma\sigma}(z) = G_{\beta\alpha\sigma\sigma}(z)$*. In this case, the off-diagonal Green's function components can be obtained using just real-valued auxiliary operators such as* $\mathcal{O} = c_{\alpha\sigma} + c_{\beta\sigma}$*, along wit the identity* $G_{\alpha\beta\sigma\sigma} = \frac{1}{2}(C_{\mathcal{O}} - G_{\alpha\alpha\sigma\sigma} - G_{\beta\beta\sigma\sigma})$*, where* $C_{\mathcal{O}}$ *denotes the DCF associated to* $\mathcal{O}$*. This approach, avoiding the need for complex arithmetic, preserves the efficiency of real-valued computation.*"

> *In Eq. (18), is Z the same on both sides of approximation sign?*

We are indebted with the referee for pointing out this important subtlety, which in fact extends to both sections 3.8 and 3.9. In the original version of the manuscript, the partition function Z appearing in different expressions (including the Eq.(18) indicated the referee), may have misleadingly suggested that it was being evaluated exactly, even though the trace had been truncated.

In the revised manuscript we have corrected this issue. We now use the symbol Z to indicate the exact partition function and reserve the symbol $\tilde{Z}$ to explicitly indicated the partition function approximated using only the finite set of low-energy eigenstates retained in the trace, i.e., those contributing significantly due to their large Boltzmann weights. This makes the approximation consistent on both sides of the involved equations, namely Eqs. (13), (14) and (18), properly reflecting the finite-state summation used in practice.

> *Are the code listings on p. 25 and p. 26 of sufficient interest to readers?*

We acknowledge the referee observation. We believe that the code listings on pages 25 and 26, i.e. the implementation of the fast-algorithm for the evaluation of the impurity RDM, provide valuable insight into the practical aspects of the implementation discussed in the text, particularly with respect to the subtle issues related to state decomposition and the treatment of fermionic sign conventions where required.

These listings are intended to complement the discussion by offering concrete examples that clarify how the formal logic is realized in code. Given the potential for confusion when dealing with basis state reconstruction and operator ordering in fermionic systems, we believe that these examples will be of interest and practical benefit to users aiming to extend or interface with the library.

> *p. 28: "as the nonsu2 and superc diagonalizations entail nontrivial subtleties in optimizing the off-diagonal components of X". What are these subtleties?*

We acknowledge that our brief comment reads a bit obscure while this point requires a more detailed, yet brief, discussion.

The bath optimization is a crucial step in the DMFT framework. As we discuss in the manuscript, to preserve the general application of EDIpack we introduced several control parameters for the conjugate-gradient fit. A key one is the definition of the norm, or distance, between matrix functions appearing in Eq. 30.

While for function matrices with a particularly symmetric structure the choice of the metric is inessential, it might dramatically affect the quality of the optimization in more general cases. The subtleties in optimizing the off-diagonal components in either `nonsu2` and `superc` diagonalizations arise from the complex interplay between these components' physical characteristics and the picked optimization metric. In either case, the coexistence of components having a very different physical interpretation, e.g., anomalous terms, and/or spanning very different energy scales represents a serious challenge to the optimization of the bath. This requires careful tuning of the control parameters to ensure that the optimization does not disproportionately favor certain matrix elements over others, thereby maintaining the overall fidelity of the bath representation. In this sense, introducing other norms which ensure a better balance among the components is a development line to improve EDIpack usage.

In the revised manuscript we amended to this unclear statement writing:
"We plan to expand the available options for the cg_norm parameter in future EDIpack updates, since the nonsu2 and superc diagonalization modes involve subtle challenges in optimizing the off-diagonal components of X, due to the coexistence of matrix elements with different physical meanings and energy scales, which require carefully balanced optimization metrics to ensure an accurate bath representation."

> *As a general coding comment: would it be possible to remove use of global variables?*

We thank the referee for this general but important observation. In principle, it is indeed possible to eliminate the use of global variables by explicitly passing all data structures through subroutine arguments. However, in the context of Fortran, and similarly in other compiled languages, global variables (typically implemented via module-level variables) do not incur any performance or memory management penalties.

Their use in EDIpack is deliberate and serves to maintain clarity and modularity in the code, particularly for widely shared objects such as configuration parameters, system-wide operators, or state indexing data. Removing these global definitions would require passing a large number of arguments through deep call chains, which we believe would significantly increase code complexity, reduce readability, and introduce greater potential errors, without offering any computational or architectural advantages.

However, we recognise the importance of careful design in managing global memory, and we have ensured that such variables are encapsulated within dedicated modules with well defined scopes. Should it prove beneficial for future code maintainability or extension by third-party contributors, we are ready to reviewing and refining this aspect of the implementation.

> *p. 40: I find the discussion of parallelism in Julia wrapper unnecessary. Such information does not necessarily age well. This belongs to a readme file.*

We thank the referee for this suggestion. We agree that the detailed discussion of parallelism integration within the Julia wrapper is better suited for specialized documentation such as a README file or the on-line user manual rather than the main manuscript. Accordingly, we have removed the entire paragraph from the revised version to keep the presentation focused and ensure the manuscript's longevity and clarity. Relevant implementation details regarding MPI usage in Julia will be provided in the package documentation to better serve users requiring this information.

> *p. 41: The code listing mentions Wband and de, but I don't see where this is coming from.*

We thank the referee for carefully reviewing this sectoin and pointing this out. Indeed, the variable `Wband` should not appear in the code listing, as it corresponds to `D`, which is

correctly defined earlier in the script. Additionally, the variable `de` represents the energy grid spacing, computed as $de = 2D/Le$, where Le is the number of energy points discretizing the Bethe lattice density of states. We have corrected the code listing accordingly to eliminate Wband and clarify the definition of de.

> *p. 42: "provides access to well-tested functions". This is unclear. Functions doing what?*

We thank the referee for pointing out the ambiguity in this sentence. Our intention was to clarify that, although the bath optimization problem is conceptually distinct from the core impurity diagonalization task for which EDIpack was originally developed, nonetheless the package provides access to a well-tested implementation of the conjugate-gradient fitting procedure for the bath optimization. We have revised the sentence accordingly in the manuscript to make this connection and the role of these functions more explicit writing:

"Given the critical importance of this optimization, EDIpack provides access to a well-tested implementation of the conjugate gradient method for performing the bath optimization, ensuring stability and reproducibility of the results, even though this task remains conceptually distinct from the diagonalization of the impurity problem, which remains the core focus of the package."

> *p. 47: generate_kgrid is confusingly complex. There must be a simpler way to accomplish this.*

We agree with the referee that the implementation of $\texttt{generate}_k\texttt{grid}$ could be simplified for clarity. The purpose of this function is merely to construct a uniform 2D grid in the Brillouin zone using the reciprocal lattice vectors, which can indeed be achieved with more concise or intuitive code.

More generally, we note that this step is not essential to the core functionalities provided by EDIpack, as it pertains only to the sampling of the Brillouin zone. For more involved tight-binding constructions, users may wish to rely on specialized libraries such as PythTB, which are designed for this purpose. Our goal here was only to provide a minimal working example for illustration.

> *p. 57: Is the footnote necessary? Will it be of interest to the expected readers of this paper?*

We appreciate the referee's suggestion. Upon reconsideration, we agree that the footnote was not essential for the intended readership and have removed it from the revised version of the manuscript.

> *Parenthesis missing in the equation at the bottom of p. 12.*
>
> *Bottom of p. 13: "are store" -> "are stored"*

> *p. 41: "an comprehensive" -> "a", "fo manipulating" -> for*
>
> *p. 46: This example two main goals: missing "has"?*

We thank the referee for carefully pointing out these typographical errors. All the mentioned issues have been corrected in the revised version of the manuscript.