Response to the reviewers

We resubmit our paper—with a long delay for which we apologize—with several modifications.

- Neither of the referees commented on what we feel are the most interesting aspects of our work, namely: (1) IPT is both a 'direct' and an 'iterative' eigensolver, in that it computes any number of eigenvalues with the same basic iterative procedure; (2) IPT is elementary (a dozen lines of code) and self-contained: it does not rely on Hessenberg reduction, does not use the Rayleigh-Ritz procedure, etc. We have rewritten the abstract and introduction in order to emphasize these points more.
- The editor and referees asked for less geometry and more numerical linear algebra. We have cut (or pushed to the appendix) several sections, including those dealing with cusps etc. We hope the paper is more easily readable for this community now.
- We found that another software package, PRIMME, is both friendlier to the user and faster than SLEPc on our single-node. We now compare IPT (in a new Julia implementation, faster than the original Python) to PRIMME.
- The section on iterative refinement was cut: we feel it did not add much (and neither of the referees commented on it). We also streamlined the quantum chemistry example.

In the following we address the referees' concerns point by point.
Thank you again for your constructive, clarifying comments.

Reviewer 1

Reviewer Point P 1.1 — The technical sections 2 and 3 are still quite difficult to follow without experience using charts, projective varieties, cusps etc. This make me wonder if the matrix-oriented readership of SIMAX will fully appreciate the work. (e.g., I'm not sure why the whole idea couldn't be formulated in affine coordinates alone)

Reply: We have cut much of these sections and hope this is no longer a concern.

Reviewer Point P 1.2 — This should be put in perspective. One would never solve the linear systems in JD / GD exactly (i.e. by direct methods). Those are handled by a few steps of an iterative (preconditioned Krylov subspace) solver, so only some matrix vector products and maybe preconditioner applications are needed. JD with exact solves is equivalent to Rayleigh quotient iteration.

Reply: This is true, the approximate solve steps only involve a few more matvecs. They nevertheless increase runtime with respect to IPT, as the new figures show.

Reviewer Point P 1.3 — Were just standard settings from Slepc used? A short explanation on that would improve the comparison.

Reply: As noted we now use PRIMME rather than SLEPc for reference iterative solvers. (We found PRIMME to be faster, and easier to extract convergence histories.) As explained in sec. 6, we use PRIMME algorithms (LOBPCG, GD, JDQMR, etc) with D^{-1} as preconditioner. Other than this, parameters have their default values.

Reviewer 2

Reviewer Point P 2.1 —
The statement that this iteration (which iterates all the eigenvectors at the same time) produces a full-rank matrix is highly non-trivial and constitutes the main theoretical result of the paper (given in the Appendix). For some reason, it is formulated as a Lemma.

Reply: Thank you. We have renamed this result a "theorem".

Reviewer Point P 2.2 — The section 4.4. "Cusps and exceptional points" does not give any specific results (I may be missing something), thus it is better to provide explicit statements of what has been done in this section and what insights are obtained.

Reply: We have moved this discussion to the appendix under "Explicit examples".

Reviewer Point P 2.3 — However, the class of matrices for which the proposed method is guaranteed to work is not so big, so examples and tests on matrices that are not "good" perturbations of the the diagonal should be added. What happens if the condition is violated?

Reply: We have added a figure (Fig. 6) to address this question. In short, as off-diagonal elements grow, the number of iterations required for convergence also grows—until the iteration diverges. Fig. 2 also addresses this question from a different perspective, namely by comparison with Rayleigh-Schrödinger theory. Are these two figures sufficient?

Reviewer Point P 2.4 — It is not clear, how the proposed method is used to find the lowest lying eigenspace.

Reply: The method does not single out extremal eigenvalues the way standard iterative solvers do. To get the lowest eigenvalue, we simply consider the column i of M such that $M_{ii} = \min \mathrm{diag}(M)$. We have added a sentence in sec. 6.1 to this effect.

Reviewer Point P 2.5 — The details of hyperparameters of other methods are also not included (i.e. block size, etc.)

Reply: We now use PRIMME rather than SLEPc with D^{-1} as preconditioner and default settings otherwise. A sentence in sec. 6.1 states this explicitly.

Reviewer Point P 2.6 — It would be interesting to see the convergence curve for the IPT method with respect to the number of iterations.

Reply: Please see the left panels of Fig. 3 and 4.

Reviewer Point P 2.7 — Jacobi-Davidson method is typically more efficient that Davidson alone

Reply: Not necessarily when M is near-diagonal, as Fig. 3 and 4 show. In quantum chemistry applications, Davidson is used more commonly than JD. In any case, we now compare IPT to both GD, JD, and some combination of them ("PRIMME-DYNAMIC"). **Reviewer Point P 2.8** — With

exact solve, Jacobi-Davidson is a Riemannian Newton method, i.e. it enjoys quadratic convergence. If the matrix is close to the diagonal, I would expect linear convergence with very small factor. Is it confirmed by experiments (this can be verified by looking at the convergence curve, not only the final iteration).

Reply: RQI is now included in performance comparisons. As anticipated by the referee, RQI requires very few iterations for a near-diagonal matrix, but since each iteration is more expansive, RQI is slower than GD or JD.

Reviewer 3

Reviewer Point P 3.1 — I understand that the authors have invested a lot of time into developing these ideas and writing the paper. The second version of the paper is better than the first version, but I believe that these methods will have very little impact. I suggest publishing the paper in a lower quality journal.

Reply: Any specific recommendations? We would love to have a list of low-quality journals for our future use.

Reviewer Point P 3.2 — In their response to the referees the authors state that it could also be used to compute a block of eigenvalues. If that can be implemented and the method is competitive then the paper would probably be publishable in SIAM Journal on Matrix Analysis and Applications.

Reply: Indeed it can, please see Fig. 5.

Reviewer Point P 3.3 — I continue to believe that the niche in which this method is useful is tiny. In their response the authors mention their section 6.1. However, transforming M into the Z0 basis is itself costly. I do not think this is useful.

Reply: We have removed this section altogether.

Reviewer Point P 3.4 — From Figure 6, I infer that for symmetric dense matrices GD is better unless epsilon is extremely small

Reply: IPT is comparable to GD for dense symmetric matrices when one eigenpair is requested, but much faster when 5 or more are requested due to its parallelism (Fig. 5). For instance, for a dense symmetric matrix with $\epsilon = .1$, IPT computes $k = 10$ eigenpairs in 50ms (the time required for 18 matvecs); PRIMME's GD by contrast needs 400ms (the time for 80 matvecs).

Reviewer Point P 3.5 — Table 1 shows that "IPT is equally efficient as the Davidson algorithm". This confirms what I wrote in my first report "I see no reason to believe that this method is in some sense better". LOBPCG and GD might be even better than the Davidson used to make table 1.

Reply: Some reasons why our method is at least in some sense better include: its higher parallelism, the fact that one or all eigenvalues can be computed using the exact same method (i.e. IPT is both 'direct' and 'iterative'), its non-reliance on Rayleigh-Ritz. And, most important in our view, its elementary character. There is nothing fancy in IPT—just a straightforward fixed-point iteration.