

1 **Title:**

2 **Scalable anisotropic vibrations of megascale macromolecules**

3 **Author List**

4 Jordy Homing Lam^{1,5,6}, Aiichiro Nakano*^{1,3,4}, Vsevolod Katritch*^{1,2,5,6}

5

6 **Affiliations**

7 ¹ Department of Quantitative and Computational Biology, University of Southern California, Los
8 Angeles, CA, USA.

9 ² Department of Chemistry, University of Southern California, Los Angeles, CA, USA.

10 ³ Department of Physics and Astronomy, University of Southern California, Los Angeles, CA, USA.

11 ⁴ Department of Computer Science, University of Southern California, Los Angeles, CA, USA.

12 ⁵ Bridge Institute and Michelson Center for Convergent Biosciences, University of Southern California,
13 Los Angeles, CA, USA.

14 ⁶ Center for New Technologies in Drug Discovery and Development, University of Southern California,
15 Los Angeles, CA, USA.

16

17 *To whom correspondence shall be addressed: anakano@usc.edu, katritch@usc.edu

18 **Abstract**

19 The Normal Mode Analysis (NMA) is a standard approach to elucidate the anisotropic vibrations of
20 macromolecules at their folded states, where low-frequency collective motions can reveal rearrangements

21 of domains and changes in the exposed surface of macromolecules. Recent advances in structural biology
22 have enabled the resolution of megascale macromolecules with millions of atoms. However, the calculation
23 of their vibrational modes remains elusive due to the prohibitive cost associated with constructing and
24 diagonalizing the underlying eigenproblem and the current approaches to NMA are not readily adaptable
25 for efficient parallel computing on graphic processing unit (GPU). Here, we present eigenproblem
26 construction and diagonalization approach that implements level-structure bandwidth-reducing algorithms
27 to transform the sparse computation in NMA to a globally-sparse-yet-locally-dense computation, allowing
28 batched tensor products to be most efficiently executed on GPU. We mapped, optimized, and compared
29 several low-complexity Krylov-subspace eigensolvers, supplemented by techniques such as Chebyshev
30 filtering, sum decomposition, external explicit deflation and shift-and-inverse, to allow fast GPU-resident
31 calculations. The method allows accurate calculation of the first 1000 vibrational modes of some largest
32 structures in PDB (> 2.4 million atoms) at least 250 times faster than existing methods.

33 **Introduction**

34 The Normal Mode Analysis (NMA) is a standard approach to derive motions from static snapshots
35 of a macromolecular structure. As a computational probe to shape-changing motions, the analysis has found
36 wide applicability in the refinement and fitting of macromolecular structures^{1,2} and in the simulations of
37 functional motions when coupled with enhanced sampling techniques^{3,4}. Many of these predicted modes of
38 motion align consistently with available experimental data in kinases⁵, ion channels⁶ and transporters⁷. Such
39 successful applications have led to an increasing appreciation of the interdependence among biological
40 structure, dynamics, and function⁸. In NMA, the input macromolecular structure is assumed to be in a
41 conformational minimum of its potential energy. Spatial arrangements of atoms (or atom groups like
42 residues) and forces among them are then assimilated into the Hessian matrix and the equation of motion
43 is analyzed under the harmonic approximation. The outcome of NMA are eigenvectors, representing sets
44 of optimal displacements for each atom in the system, ranked by their ascending eigenvalues that reflect

45 increasing strain along those directions. Early attempts of NMA on macromolecules^{9–12} were derived from
46 all-atom potentials and full dense Hessian matrices. This approach to NMA is still the golden standard in
47 analyzing the shear and hinge motions of proteins with sizes typically less than 5000 atoms.¹³ However,
48 applying NMA in this form to larger macromolecules is difficult as their energy minimizations are prone
49 to overstepping, rendering unphysical modes.^{14,15} In a pioneering work by Tirion¹⁵, this requirement was
50 relaxed by replacing the detailed all-atom potentials with an elastic network of pairwise Hookean potential,
51 connecting atoms within a neighborhood boundary. The NMA analysis then proceeds by assuming that the
52 provided structure was minimized under experimental conditions. As such, crystal coordinates, which lack
53 hydrogens and/or sidechains, can be analyzed directly in absence of a forcefield. This practical form of
54 NMA is now commonly known as the elastic network model (ENM). Despite its simplicity, as demonstrated
55 convincingly by Hinsen^{16,17} and Sanejouand¹⁸, the low-frequency ENM modes were shown to agree with
56 the fluctuations observed in X-ray crystallography as well as with NMAs performed under standard all-
57 atom potentials, as long as only low-frequency modes are concerned. The ENM also found major
58 applications in Cryogenic Electron Microscopy (Cryo-EM) and Cryogenic Electron Tomography (Cryo-
59 ET) for the flexible refinement of the composite density maps^{19–21} and the flexible fitting of atomic
60 models^{22–24} when EM volume or atomic templates are available. These applications of NMA are especially
61 compelling for large macromolecular complexes, when other methods are computationally expensive.
62 However, computing the NMA, even in this practical ENM form, remains challenging for large
63 macromolecular systems of more than 1,000,000 particles. Namely, the storage, the construction and the
64 diagonalization of the Hessian matrix all create imminent difficulties in terms of time and memory
65 complexities as the size of the matrix increases quadratically with the number of atoms N . While ideally
66 the memory complexity of NMA only increases linearly, the linear factors due to packing density of atoms
67 as well as the cubic time complexity to diagonalize the Hessian matrix present a major bottleneck in
68 calculations (See Methods). As such, many innovative approaches, including Anisotropic Network Model
69 (ANM)^{15,25}, Rotation-Translation Block (RTB) method²⁶ and Block Normal Mode (BNM)²⁷, were
70 undertaken to eliminate the degrees of freedom (DOF) to be differentiated in these larger systems, hence

71 reducing N for 8-10 times. Agreement of these methods with experimental data^{18,25} are satisfactory, though
72 the displacement information of the discounted atoms were lost and the appropriate level of granularity,
73 especially in presence of elongated (e.g., lipids) or planar (e.g., aromatics) chemical moieties, are hard to
74 be determined. To apply NMA without excessive coarse-graining, the development of faster numerical
75 recipes is necessary^{28,29}. More recently, the choices on diagonalization algorithms were examined on a
76 hyperthreaded machine in the work of Koehl³⁰. This ultimately allowed the first 100 ENM modes of a ZIKV
77 virus (PDBID: 5IZ7), with around 800 thousand atoms and around 300 million nonzero entries in its
78 Hessian, to be calculated within an hour when the Jacobi-Davidson Method (JDM)^{31,32} were filtered with
79 an 80-degree low-pass Chebyshev polynomial of the Hessian.³⁰ However, the current state-of-the-art is
80 still far from handling megascale systems with more than a million atoms and billions of non-zero entries
81 in the Hessian.

82 In this work, we designed several synergistic algorithms to fully engage dense graphic processing
83 unit (GPU) kernels in both construction and diagonalization of the Hessian in atomic NMAs. Specifically,
84 we developed a level-structure algorithm to minimize the bandwidth of the Hessian, prior to its construction,
85 and thereby converting its sparse computation into a globally-sparse-yet-locally-dense computation, which
86 enables the batched execution of tensor products on GPUs. We also mapped, optimized and compared
87 several low-complexity Krylov-subspace eigensolvers, supplemented by techniques such as Chebyshev
88 filtering³³⁻³⁵, sum decomposition, external explicit deflation³⁶ and shift-and-inverse, to allow fast
89 calculations on a GPU device. The level-structure algorithm produces an isomorph of the original elastic
90 network, for which its unpermuted eigenpairs can be recovered in linear complexity using the bijection
91 mapping. The implementation of this INCHING (“Isomorphic Nma Calculations Harnessing 1 Necessary
92 Gpu”) algorithm presented here was benchmarked on macromolecules from the Protein Data Bank³⁷ with
93 sizes up to 2.4 million atoms. Compared to other existing methods, 250-370 times speedup in throughput
94 was achieved while maintaining residual error under 10^{-12} . The utility of the INCHING method was also
95 demonstrated through examples, including the largest experimentally resolved atomistic structure of a
96 mature HIV-1 capsid³⁸ (PDBID:3J3Q) with 2.4 million atoms and 1.6 billion non-zero entries in its Hessian.

97 Using our INCHING program, we were able to resolve its first 64 atomic normal modes within 44 minutes
98 of wall-clock time on a single NVIDIA® A100 Tensor Core GPU and its first 1000 atomic normal modes
99 within 63 hours. Fast, accurate, and highly scalable GPU-optimized implementation of NMA approach will
100 find practical applications in conformational analysis of large and dynamic macromolecular complexes,
101 and facilitate their refinement from cryo-EM/cryo-ET data.

102 **Results**

103 **Overview of the INCHING Algorithm**

104 The NMA is a study of a high-dimensional potential energy surface under harmonic approximation. In this
105 work, the INCHING algorithm is applied to the elastic network model (ENM), a practical form of NMA
106 introduced by Tirion¹⁵. As described in Methods, NMA is based on solving a standard eigenproblem, $HQ =$
107 ΩQ , concerning a Hessian matrix $H \in \mathbb{R}^{3N \times 3N}$, where $\Omega \in \mathbb{R}^{3N \times 3N}$ and $Q \in \mathbb{R}^{3N \times 3N}$ are the eigenvalue
108 matrix and the eigenvector matrix, respectively. For large collective conformational changes, only the
109 eigenpairs with the smallest non-zero eigenvalues (the lowest-frequency modes) are of interest. However,
110 challenges in the construction and diagonalization of the Hessian grow with the system size, and are
111 notoriously resistant to parallelization. Consequently, current approaches to NMA are not readily adaptable
112 for GPU computing, which imposes even stricter memory limits and algorithmic requirements to fully
113 leverage on-chip parallelism. As illustrated in Fig 1a, the INCHING algorithm was implemented in three
114 synergistic stages – permutation, construction, and diagonalization – to resolve the challenges of calculating
115 NMA on GPU. In the following sections, we will describe each stage in detail.

116 **Permutations to Achieve Bandwidth Reduction**

117 GPUs are hardware engineered to efficiently access contiguous memory locations, thus inherently favoring
118 dense computations. However, in practice, the experience of memory access is dictated by the sparsity

119 pattern of the data presented. Since the inception of NMA, it has been recognized that the Hessian is
120 globally-sparse with prevailing zeros, primarily due to the long-range cut-offs applied to molecular
121 mechanics potentials.¹¹ These globally-sparse Hessians are also locally-sparse, requiring very large
122 bandwidths with non-uniform adjacencies (Fig. 1b c.f. Fig 1c.) due to sequentially distal segments of
123 polymer(s) interacting in tertiary and/or quaternary structure(s)^{39,40} (Extended Data Figure 1), and the
124 situation is further complicated by chemicals (e.g., cofactors, ions, water, lipids) with no inherent order
125 being integrated into the polymer structure (Extended Data Figure 2). All these factors contribute to
126 inefficient on-chip parallelism when dense row-sweeps were performed in batches.

127 The primary objective of the INCHING algorithm is to strategically permute columns and rows of
128 the Hessian, prior to all its computations, such that a globally-sparse-yet-locally-dense computation can be
129 achieved in subsequent stages. By permuting the atom ordering, we can always generate a graph isomorph
130 of the original elastic network, while allowing retrieval of its unpermuted eigenpairs in linear complexity
131 if a bijection mapping is provided. However, finding a permutation that exactly minimizes the bandwidth
132 of a matrix is NP-complete⁴¹. To approximate this reduction, we have designed a level-structure algorithm
133 called 3DRCM, which extends on the well-established Reverse Cuthill-McKee (RCM)^{42,43} algorithm, to
134 handle 3-D coordinate data. Traditional RCM and variants⁴⁴⁻⁴⁶ operate under the assumption that the input
135 —a matrix—is realized and stored once-and-for-all. This framework is suitable for testing new offline
136 analyses on the preserved matrix. However, a major problem overlooked is that it may not even be practical
137 to realize the matrix efficiently, when its parallel computation spans a large bandwidth. In 3DRCM, a 3-D-
138 tree data structure⁴⁷ is taken as input instead of a matrix to eliminate this circular reference. This
139 modification enables efficient dynamic neighborhood lookups in the permutation process and avoids
140 premature realization of the Hessian matrix. In Extended Data Figure 3a, we showed that for
141 macromolecules with over 100 thousand atoms, the mean bandwidth for the Hessian in default PDB
142 sequence ordering can vary between 10% and 90% of N . However, by permuting the Hessian with 3DRCM
143 before its construction, the mean bandwidth of the Hessian is consistently reduced and remains below 10%
144 of N , hence confirming the local density of the matrix. In Method and Extended Data Figure 3b, we also

145 showed that the time complexity of 3DRCM is practically dominated by a linear term proportional to the
146 packing density of the macromolecule, thereby ensuring the cost-effectiveness of the algorithm.

147 GPU-resident Construction of the Hessian with Batched Tensor Products

148 The proposed 3DRCM permutation enables spatial neighbors from each batch of atoms to be retrieved as
149 locally-dense indexing slices. In our Method, we further showed that, by extending on the tensorial
150 representation suggested by Koehl³⁰, the Hessian can be constructed in batches through vectorized
151 operations, such as tensor products and broadcasts, to leverage efficient on-chip parallelism in GPU. This
152 replaces the need for explicit index loading and external storage of the distance matrix to enable fast
153 computations. Besides, in our implementation, only the lower triangle of the Hessian was incrementally
154 stored, and later accessed, in Compressed Sparse Row (CSR) format, this halves the memory requirement.

155 In [Extended Data Figure 4](#), we showed that even for a system comprising 2.4 million atoms ([PDBID: 3J3Q](#)),
156 the computation of the 3DRCM permutation ordering and the subsequent construction of the Hessian on
157 GPU took only 5.2 minutes and 6.9 minutes in wall-clock time, respectively. Our method shows significant
158 speedup over ProDy2.4⁴⁸, an open-source implementation intended for coarse-grained protein
159 representation in ANM tasks. For the largest case that ProDy2.4 handled ([PDBID: 6NCL](#)), containing 305
160 thousand atoms, ProDy2.4 took 7.8 hours to construct the Hessian, whereas our INCHING approach
161 delivered the matrix in just 93 seconds, showing a 302-fold acceleration.

162 GPU-resident Diagonalization of the Hessian Matrix

163 In NMA calculations, a significant portion of computational resources is often dedicated to diagonalizing
164 the Hessian, where the throughput is dependent on the performance of the eigensolver. One popular method
165 in solving large symmetric eigenproblem is the Implicitly Restarted Lanczos Method (IRLM)⁴⁹, which is
166 available in the ARPACK package⁵⁰, a backend incorporated into ProDy2.4. In ARPACK, the matrix-
167 vector multiplications are accelerated by threaded BLAS subroutines. However, despite the utilization of a

168 high-end AMD processor with 64 threads, ProDy2.4 routines (ProDy-ARPACK-FullSparse) fail to
169 converge within 48 hours when systems exceed 305 thousand atoms (**PDBID: 6HIV**, 311 thousand atoms),
170 indicating that we have reached the limit of hardware improvement. Note that ProDy2.4 with the default
171 LAPACK⁵¹ backend (ProDy-LAPACK-FullDense) working on a full dense Hessian is faster than all other
172 programs when there are less than 500 atoms, but we were not able to proceed once exceeding 29 thousand
173 atoms due to rapid rise in memory consumption. We also implemented a version of INCHING (INCHING-
174 ARPACK-FullSparse) that uses our fast Hessian construction routine on GPU followed by a one-time
175 device-to-host transfer to carry out diagonalization with ARPACK, but we were not able to proceed beyond
176 1.2 million atoms in 48 hours, reflecting the bottleneck in calculation is fundamentally the diagonalization
177 process.

178 GPU-accelerated approaches for solving large sparse symmetric eigenproblems continues to be a
179 vibrant area of research.⁵²⁻⁵⁷ To explore this next-generation technology, we implemented, optimized and
180 compared several diagonalization methods on a single NVIDIA® GPU device, including the Implicitly
181 Restarted Lanczos Method (IRLM)^{49,50}, the Thick Restart Lanczos Method (TRLM)⁵⁸, the Jacobi Davidson
182 Method (JDM)^{31,32} and some of their Chebyshev-filtered versions e.g. the Chebyshev-filtered Thick Restart
183 Lanczos Method (CTRLM)³⁴ and Chebyshev-Davidson Method (CDM)³⁵. In general, these methods all
184 share the objective of computing eigenpairs within an interval (e.g., those with the smallest eigenvalues),
185 but they differ in their approaches to update the Krylov subspace that approximates the eigenpairs. (See
186 Methods for detail). The IRLM and TRLM are variants of the Hermitian Lanczos Method (HLM), but they
187 differ in how they utilize the solutions of a much smaller tridiagonal eigenproblem to reinitialize the Krylov
188 subspace at restarts. On the other hand, the JDM directly corrects the Krylov subspace by solving for an
189 approximate fit to eliminate residuals and subsequently also works over solutions of a smaller projected
190 eigenproblem. The CTRLM and CDM uses filters (low- or band-pass) constructed from Chebyshev
191 polynomials to magnify wanted interval in the spectrum. The choice among these methods is not obvious,
192 though their speed all depends on the cost of matrix-vector multiplication. In our INCHING protocols
193 (INCHING-TRLM-HalfSparse, INCHING-IRLM-HalfSparse, INCHING-JDM-HalfSparse, INCHING-

194 CTRLM-HalfSparse, INCHING-CDM-HalfSparse), these multiplications are accelerated by the SpMV
195 CSR kernel in cuSPARSE⁵⁹. The GPU computation was instructed through CuPy⁶⁰, a python API to
196 NVIDIA®’s CUDA⁶¹, cuBLAS⁶², cuSPARSE⁶³ and custom kernel programming. To accommodate the
197 lower memory capacity of GPU, the Hessian is accessed as a sum decomposition of its lower triangle,
198 halving the memory requirement. Alternatively, the calculations can also be done in the full sparse matrix
199 with a further 40% speedup, when GPU memory is not exhausted. (See Extended Data Figure 5) In
200 Extended Data Figure 6, we further showed that an explicit external deflation³⁶ of the first 6 rigid modes
201 with zero eigenvalues (rotations and translations) can help to improve runtime of the remaining 58 non-
202 zero eigenpairs in sub-megascaling regime while sharing very similar memory footprint. In addition, by
203 incorporating low- and band-pass Chebyshev filters^{34,57}, we also lifted the memory limit regarding the
204 number of modes to be resolved. (See INCHING-CTRLM-HalfSparse in Figure 2 and Extended Data
205 Figure 8)

206 **Benchmarks**

207 Benchmarks on correctness, memory and speed for calculating the first 64 modes are illustrated in
208 Fig. 2. All reported runtimes are wall-clock time, and the tests were conducted on a computer with a 64-
209 threads AMD EPYC™ 7513 processor and a single NVIDIA® A100 Tensor Core GPU, unless specifically
210 noted. The accuracy is measured by the 2-norm of the residual error. Across 116 benchmark cases
211 encompassing macromolecules ranging from around a thousand atoms to around 2.4 million atoms,
212 including PDBID:3J3Q, the largest experimentally resolved atomic structure in PDB at <10Å range, our
213 INCHING protocols were able to afford accuracy at 10^{-12} level and achieved a peak memory consumption
214 consistently lower than the ProDy2.4 routines. Importantly, with memory requirement halved by accessing
215 only the lower triangle of the Hessian with a sum decomposition, the throughputs of our INCHING
216 protocols, including construction and diagonalization of the Hessian, are still 146-251 times faster than
217 ProDy-ARPACK-FullSparse and are 265-1290 times faster than ProDy-LAPACK-FullDense, depending
218 on our choice of diagonalization algorithm. Remarkably, for a 305-thousand-atoms system (PDBID:

219 6NCL), all of our INCHING protocols took at most 5 minutes to converge with the fastest convergence
220 being achieved by INCHING-TRLM-HalfSparse at 2.9 minutes. The same task took ProDy-ARPACK-
221 FullSparse 12.4 hours to converge. INCHING-TRLM-HalfSparse, INCHING-JDM-HalfSparse and their
222 Chebyshev-filtered versions (INCHING-CTRLM-HalfSparse, INCHING-CDM-HalfSparse) also
223 converged for the HIV-1 capsid structure (PDBID:3J3Q, 2.4 million atoms), with the fastest convergence
224 achieved by INCHING-CTRLM-HalfSparse within 44 minutes. In Extended Data Figure 8a-c, we show
225 that, at the same accuracy level as JDM (10^{-12}), moderate speed-up in sub-megascle regime (mean at
226 1.21) can be achieved by CDM with an 80-degree polynomial, though the speed-up diminished to a slow-
227 down in the megascle regime (mean at 0.91). This is in contrast to applying an optimized low-pass filter
228 to TRLM, where we showed that the Chebyshev-filtered TRLM (CTRLM) can steadily deliver a speed-up
229 over TRLM in the megascle regime (mean at 7.44). In Extended Data Figure 8d-e, we further showed that
230 linear or better scaling in runtime has been achieved in terms of the radii $R_C = 6,8,14\text{\AA}$ to be considered,
231 except for several cases at a lower radius $R_C = 6\text{\AA}$, likely due to poorer conditioning of the matrix. In
232 Extended Data Figure 9, by incorporating a band-pass Chebyshev filter developed in recent works^{34,57} into
233 our TRLM implementation (INCHING-CTRLM-HalfSparse), we also achieved linear time scaling in the
234 number modes to be solved, while keeping memory usage constant. This ultimately allows 1000 modes of
235 all the benchmarks to be solved, in batches of 64 eigenvectors, without compromising memory or run time
236 or atomic details. The method was also tested on a RTX4090 NVIDIA® GPU in batches of 28 eigenvectors
237 in Extended Data Figure 10b,c.

238 Anisotropic vibrations of some megascle systems

239 To illustrate the usage of our software, we have applied INCHING to some of the largest atomic objects
240 available, both natural and artificial. In Fig 3, we illustrate the first non-zero mode of the mature HIV-1
241 capsid structure (PDBID:3J3Q), the largest experimentally resolved atomic object to date at $<10\text{\AA}$
242 resolution range, containing 2.4 million atoms and 1.6 billion non-zero entries in its Hessian. The cone-
243 shaped capsid is an assembly of 186 hexamers and 12 pentamers, and it was suggested that these pentamers

located at its hemispherical ends induce stable closure of the capsid by allowing sharp bite angles at the surface.⁶⁴ With our INCHING-JDM-HalfSparse protocol, we were able to resolve its first 64 normal modes at residual error 10^{-12} within 1.3 hours on an NVIDIA® A100 Tensor Core GPU, where the first non-zero mode corresponds to the oscillation of its hexameric surface roughly anchored at the pentamers supporting the theory of quasiequivalence.⁶⁵ In Fig 4, the normal modes of a dilated human nuclear pore complex (NPC)⁶⁶ resolved at 50 Å were shown (EMDB: EMD14321, PDBID: 7R5J). The structure contains 4.8 million atoms and 3.5 billion non-zero entries in its Hessian. We were able to obtain the first 30 non-zero modes of this NPC structure within 12.5 hours at accuracy of 10^{-12} . In this calculation, an explicit external deflation³⁶ was applied to remove the first six rigid modes. We observe that while the first 14 non-zero modes mostly concern motions in the cytoplasmic and nuclear ring, the constriction of the inner ring can be observed at the fifteenth non-zero mode, reflecting the key functionally relevant conformational change. This calculation is at the edge of 80 GB of memory limit for the hardware used, though more powerful GPU systems on the market could handle even larger macromolecule superstructures. In Fig 5, we applied the same methodology on the largest artificial DNA origami nano-structure, a DNA airplane made of 33kbps at its relaxed state⁶⁷ containing around 1.8 million atoms, including hydrogens, with a Hessian containing 1.8 billion non-zero entries. The structure has an apparent bilateral symmetry, where the joints leading to the wings are not chemically symmetric.⁶⁷ Interestingly, while its first non-zero mode presents a symmetric flop in its wings, the second non-zero mode demonstrates a complicated non-symmetric twisting motion involving its wings and stabilizers. In Extended Data Figure 10a, we also solved the first 64 modes of a 5-million pseudo-atoms coarse-grained representation of a Faustovirus capsid⁶⁸ (PDBID: 5J7V 26 million atoms resolved at 15.5 Å), illustrating the potential to incorporate coarse-graining strategies in handling systems that cannot fit into memory.

266

267 **Discussion**

268 In recent years, there has been a significant shift in the focus of NMA methodologies, with an emphasis on
269 accommodating larger atomic systems. The aim of an NMA is to gain insight into how some macroscopic
270 motions involving communicating domains, can arise from microscopic interactions at the atomic level. In
271 this respect, the mode shapes of NMA, which orchestrate collective motions concerning distal parts of the
272 macromolecule, can often provide hints to understanding the biological structure at hand. Existing methods
273 to analyze NMA can readily handle medium-sized structures with a few thousand atoms, but beyond this
274 application of NMA is limited by computing resources and is not easily scalable by parallelization. This
275 fundamentally restricts the molecular plasticity analysis in areas such as Cryo-ET and Cryo-EM, where
276 macromolecular structures studied are often composed of millions of atoms and rather flexible. In these
277 applications, fast and accurate mega-scale NMA would greatly facilitate modeling conformational
278 dynamics in the refinement of the composite density maps^{19–21} and the flexible fitting of atomic models^{22–}
279 ²⁴ when EM volume or atomic-resolution template structures are available. Two major challenges in NMA
280 scaling to mega-atom macromolecular complexes are the construction of the Hessian matrix and solution
281 of the large-scale eigenproblem entailed. In this work, we have developed and implemented several
282 advanced numerical recipes optimized for GPU computing, including a bandwidth-reduction algorithm and
283 several alternative eigensolvers, to handle these challenges. This allows us to resolve normal modes, under
284 the practical form of elastic network model, without coarse-graining the provided atomic structure for
285 systems with several million atoms. In many cases, coarse-graining of biomolecular residues is expert-
286 driven, and the appropriate level of granularity can be hard to determine when the chemical moiety is
287 elongated (e.g., lipids) or is planar in shape (e.g., aromatics), especially in heterogeneous complexes
288 comprised of protein, nucleic acid, lipid membranes, and sugars. The coarse-grained particle motions must
289 be broadcasted into an all-atom construct if an atomic-scale understanding of the system is desired. In this
290 respect, the implementation of an all-atom NMA, faithful to the formulation of Tirion¹⁵, is a direct response
291 to this shortcoming as all the atoms are now included in the system without neglecting each of their degrees

of freedom (DOF). Several recent approaches to reduce DOF of atomic systems are interesting and indispensable to further scale up though. One of the first and the most popular approaches is the Anisotropic Network Model (ANM), which consider a subset of n_b atoms from the all-atom system, usually only the phosphorus P in nucleic acids and/or the backbone carbon C α of the proteins^{15,25}, thus effectively reducing N for 8-10 times. Agreement of ANM with experimental data^{18,25} is satisfactory, though the displacement information of the discounted atoms is lost. This issue was alleviated by the rotation-translation block (RTB) method²⁶, where a projection operator was developed to coarse-grain the atomic system into a system of n_b rigid blocks, each with its own translational and rotational DOF, hence effectively reducing the $3N \times 3N$ Hessian matrix to a $6n_b \times 6n_b$ RTB matrix. Further refinement along this line is the Block Normal Mode (BNM)²⁷ method, which surrogates the initial realization of the peak-memory-consuming $3N \times 3N$ Hessian matrix; this was done by exploiting the block structure of the projection operator and by constructing only part of the sparse full atomic Hessian on-the-fly. Very recently, it was also shown that an extrapolation of RTB modes can effectively predict nonlinear motions at large amplitudes.⁶⁹ Depending on the granularity of the system, more than tenfold reduction in the size of the Hessian matrix can be achieved. However, the cost of diagonalization can still be very prohibitive.

In this respect, parallel computing on GPU, as we implemented in this work, is an effective way to amortize the cost at a fundamental level, without compromising accuracy or atomistic detail. We also expect that advances in GPU hardware⁷⁰ in memory and processing rate, integrated with techniques in solving large-scale eigenproblems, in particular recent trends in spectrum slicing^{34,52–57,71}, would eventually allow even faster NMA calculations on systems exceeding 20 million atoms, such as structure of a Faustovirus⁶⁸ available in low resolution (PDBID: 5J7V, 15.5 Å). Nonetheless, given the applicability of NMA in biological systems, we believe our framework will be useful in the exploration of megascale structural dynamics. The growing complexity of the megacomplexes resolved by Cryo-EM and/or cryo-ET now calls for methods that can rapidly capture conformational and functional plasticity, potentially as an intrinsic part

316 of the refinement pipeline. Such methods will play a crucial role in advancing our understanding of these
317 macromolecular machines.

318

319 Methods

320 Normal Mode Analysis of an Elastic Network Model

321

322 The Normal Mode Analysis (NMA) is a classic approach to derive motions from static structures at local
323 minima of a potential energy surface. The potential energy V is dependent on the conformation $X \in \mathbb{R}^{3N}$
324 for a structure with N atoms at time t . Without loss of generality, we begin with a particular conformation
325 $X^{(0)} \in \mathbb{R}^{3N}$. For small displacements, we may then tolerate a second-order Taylor expansion,

$$326 V(X|X^{(0)}) = V(X^{(0)}|X^{(0)}) + \nabla V(X^{(0)}|X^{(0)})(X - X^{(0)}) + \frac{1}{2}(X - X^{(0)})^T \nabla^2 V(X^{(0)}|X^{(0)})(X - X^{(0)}) \\ 327 + \dots$$

328 (Equation 1)

329 By choosing the energy level $V(X^{(0)}|X^{(0)}) = 0$ and assuming local minimum $\nabla V(X^{(0)}|X^{(0)})^T = 0$, we
330 are left with the quadratic form,

$$331 V_{NMA} = \frac{1}{2} \Delta X^T H \Delta X$$

332 (Equation 2)

333 , where $H =: \nabla^2 V(X^{(0)}|X^{(0)})^T \in \mathbb{R}^{3N \times 3N}$ is the Hessian matrix and $\Delta X =: (X - X^{(0)}) \in \mathbb{R}^{3N}$ is the
334 deviation from the minimum. We will defer the layout of H to the next section when the form of potential
335 energy V is defined and proceed to discuss the outcome of an NMA. Substituting V_{NMA} into the equations
336 of motion gives a partial differential system

$$337 M \frac{d^2 \Delta X}{dt^2} = -H \Delta X$$

338 (Equation 3)

339 And, the general solution $\Delta X = Qe^{-i\omega t}$ gives a generalized eigenproblem

340
$$HQ = MQ\Omega$$

341 (Equation 4)

342 Where the eigenvector matrix $Q \in \mathbb{R}^{3N \times 3N}$ can be viewed as an eigentensor $Q \in \mathbb{R}^{3N \times N \times 3}$. The physical
343 meaning of this eigentensor Q is that each of its atom slice (the second index) gives a displacement vector,
344 hence $3N$ linearized mode shapes, i.e., the collective motions, of the static structure $X^{(0)}$ can be obtained.

345 In NMA, the first 6 modes will always have zero eigenvalues corresponding to rigid translational and
346 rotational displacements. The eigentensor Q is also the best displacement under orthonormality constraint

347
$$\min_Q V_{NMA} \text{ s.t. } Q^T Q = I$$
. Eigenfrequency $\sqrt{\omega_i}$ is the square root of the eigenvalue ω_i .

348 A variety of V exists. In the standard NMA, potential energies from a detailed all-atom forcefield
349 can be used. However, this approach suffers from the tedious, and sometimes virtually unachievable,
350 requirement of energy minimization.¹⁰ This requirement was surrogated in the work of Tirion¹⁵, where the
351 potential energy of an atomic system was considered as an elastic network model (ENM) connecting atoms
352 i and j

353
$$V(X) =: \sum_{ij} \frac{1}{2} k_{ij} (r_{ij} - r_{ij}^{(0)})^2$$

354 (Equation 5)

355 Where r_{ij} is the variable Euclidean distance between atom pairs; $r_{ij}^{(0)}$ is the equilibrium Euclidean distance
356 from $X^{(0)}$. Importantly, a Heaviside function k_{ij} , parameterized on $r_{ij}^{(0)}$, was used to eliminate the long-
357 range interactions beyond the threshold R_C . This can also be considered as applying an adjacency matrix
358 $k_{ij} \in K$ to the network.

359
$$k_{ij}(r_{ij}^{(0)}) = \begin{cases} 1, & r_{ij}^{(0)} \leq R_C \\ 0, & r_{ij}^{(0)} > R_C \end{cases}$$

360

361 (Equation 6)

362 As in the formulation of Tirion, a unified mass $M = I$ were taken, hence reducing the generalized
363 eigenproblem to a standard eigenproblem.

364
$$HQ = Q\Omega$$

365 (Equation 7)

366 **Permuting the Hessian to Produce Graph Isomorphs**

367 In our later exposition, we will permute the Hessian matrix such that the resultant matrix is locally dense.

368 Observe that a similarity transform of H with permutation matrix $P \in \mathbb{R}^{3N \times 3N}$ preserves the eigenvalues
369 Ω as

370
$$(PHP^T)(PQ) = (PQ)\Omega$$

371 (Equation 8)

372 and the eigenvector $P^T PQ$ of the original problem can be recovered by simply applying inverse of the
373 unitary permutation matrix, which is its transpose. Importantly, if we restrict the permutation to atom
374 ordering, i.e., permuting every 3 consecutive indices as an immutable group in the $3N$ indices, then the
375 resultant Hessian is a representation of a graph isomorph of the original elastic network, where the
376 bijection is provided by correspondence between the original atom ordering and the permuted atom
377 ordering. This property means that the similarity transform can be computed by simply initiating with a
378 permuted coordinate and similarly Q can be recovered by permuting the atom slice of PQ using the same
379 bijection with linear cost.

380 **Constructing the Hessian Using Batched Tensor Products and Broadcasts**

381 We begin with the following un-vectorized notation, where the superscript (0) is dropped for readability.

382
$$X^{(0)} =: [X_1^{(0)}, \dots, X_n^{(0)}]; Vec(X^{(0)}) =: [x_1, y_1, z_1, \dots, x_n, y_n, z_n]$$

383 (Equation 9)

384

385 Applying differentiations, the off-diagonal force constant block is specified by

386 $H_{ij} = -\frac{k_{ij}}{\left(r_{ij}^{(0)}\right)^2} \left(X_i^{(0)} - X_j^{(0)}\right) \left(X_i^{(0)} - X_j^{(0)}\right)^T$

387 $= -\frac{k_{ij}}{\left(r_{ij}^{(0)}\right)^2} \begin{bmatrix} (x_i - x_j)^2 & (x_i - x_j)(y_i - y_j) & (x_i - x_j)(z_i - z_j) \\ (y_i - y_j)(x_i - x_j) & (y_i - y_j)^2 & (y_i - y_j)(z_i - z_j) \\ (z_i - z_j)(x_i - x_j) & (z_i - z_j)(y_i - y_j) & (z_i - z_j)^2 \end{bmatrix}$

388 $H_{ii} = -\sum_j H_{ij}$

389 (Equation 10)

390

391 For any matrix A with n rows, we can define for a row in it,

392 $\beta_i =: \left| i - \min_j (j \mid a_{ij} \neq 0) \right|$

393 (Equation 11)

394 Then the bandwidth of the matrix is $\max_i(\beta_i)$ and the profile of the matrix is $\sum_i \beta_i$, the mean bandwidth is

395 $\frac{\sum_i \beta_i}{n}$. Clearly, due to the cutoff R_C , the layout of the Hessian tensor $H \in \mathbb{R}^{N \times 3 \times N \times 3}$, viewing from the atom

396 slices, will have the same bandwidth as the adjacency matrix K of the macromolecule. The tensor is globally

397 sparse in most cases. This observation also applies to standard NMA as long range cut-offs were used for

398 van der Waals, electrostatic, and hydrogen-bonding interactions.¹² In general, it is not advisable to construct

399 a dense Hessian matrix that contains a large number of zero entries due to the quadratic storage consumption

400 and the resultant increase in fill-ins. To construct a sparse Hessian matrix, we find the tensorial layout by

401 Koehl³⁰ a good starting notation. In which, he defined the following $N \times 3$ matrix, which can also be

402 vectorized as $3N$ elements.

403 $U_{ij}(X^{(0)}) =: \left(0, \dots, 0, \frac{X_i^{(0)} - X_j^{(0)}}{r_{ij}^{(0)}}, 0, \dots, 0, \frac{X_j^{(0)} - X_i^{(0)}}{r_{ij}^{(0)}}, 0, \dots, 0 \right)$

404 (Equation 12)

405

406 The distance $r_{ij}^{(0)}$ can be precomputed and filtered by $k_{ij}(r_{ij}^{(0)})$ for pairs within R_C . Then, the off-
 407 diagonal block of the entire Hessian tensor is given as a sum of Kronecker products.

408
$$H_{ij} = \sum_i \sum_j k_{ij} U_{ij} \otimes U_{ij}^T$$

409 (Equation 13)

410

411 More importantly, when Hessian-vector multiplication Hw is accessed in eigensolvers, the following can
 412 be done separately for each (i, j) pairs.

413
$$Hw = \sum_{ij} k_{ij} (U_{ij} \otimes U_{ij}^T) w = \sum_{ij} k_{ij} (U_{ij} w) U_{ij}$$

414 (Equation 14)

415

416 Given explicit indexing for locations of the non-zeros k_{ij} , the above Hessian-vector product can be
 417 effectively parallelized on a hyperthreaded computer.

418 In our implementation, the tensorial representation above was refined to fully exploit on-chip
 419 parallelism on GPU and to avoid explicit index loading which tends to thread divergence on GPU. To
 420 facilitate discussion, we define a difference matrix $G_{ij} \in \mathbb{R}^{N \times 3}$

421
$$G_j(X^{(0)}) =: (0, \dots, 0, X_{b_3}^{(0)} - X_j^{(0)}, \dots, X_i^{(0)} - X_j^{(0)}, \dots, X_{b_4}^{(0)} - X_j^{(0)}, 0, \dots, 0)$$

422 (Equation 15)

423

424 Supposed, we know $|b_3 - b_4|$ is the bandwidth of an off-diagonal batch of H , then, for two contiguous
 425 atom indices $(b_1, b_2]$ and $(b_3, b_4]$, the batch $G \in \mathbb{R}^{3|b_1-b_2| \times 3|b_3-b_4|}$ can be given by the four index corners
 426 (b_1, b_2, b_3, b_4) .

427
$$G_{(b_1, b_2, b_3, b_4)} = \sum_{i \in (b_1, b_2]} \sum_{j \in (b_3, b_4]} G_i \otimes G_j^T$$

428 (Equation 16)

429 In practice, $(b_3, b_4]$ could include intervals that lack neighboring atoms for the range $(b_1, b_2]$, but the
 430 indexing slice $(b_3, b_4]$ can be further refined into multiple contiguous indexing slices to eliminate
 431 unnecessary calculations. Nevertheless, filtering for non-zero elements in the batch can be done by
 432 observing [Equation 10](#) in that the squared interatomic distance is given by the trace of each block
 433 $\left(r_{ij}^{(0)}\right)^2 = \text{tr}(G_{ij})$, then we can consider the following block-wise operations, readily parallelizable on
 434 GPU by in-place tensorial broadcasts.

$$435 \quad Y_{ij} = \text{tr}(G_{ij})$$

$$436 \quad K_{ij} = \begin{cases} 1, & Y_{ij} \leq R_C^2 \\ 0, & Y_{ij} > R_C^2 \end{cases}$$

$$437 \quad Y_{ij} = \begin{cases} Y_{ij}^{-1}, & \forall i \neq j \\ 0, & \forall i = j \end{cases}$$

$$438 \quad H_{(b_1, b_2, b_3, b_4)} = \left((K_{(b_1, b_2, b_3, b_4)} \odot Y_{(b_1, b_2, b_3, b_4)}) \otimes 1_{3 \times 3} \right) \odot G$$

439 [\(Equation 17\)](#)

440

441 The \odot is the Hadamard product operator; $1_{3 \times 3}$ is the all-ones matrix presenting the broadcast. This is
 442 followed by row-sum in the diagonal $H_{ii} = -\sum_{j \in (b_3, b_4]} H_{ij}$ accordingly. Only the lower triangle is
 443 incrementally stored in the Column-Sparse-Row (CSR) format for subsequent calculation, hence
 444 consuming $O(9p)$ memory for the data content; p is the number of non-zero pairwise interaction for $i \geq$
 445 j . The dense batch $G_{(b_1, b_2, b_3, b_4)}$ presented above will require $O(9|b_1 - b_2||b_3 - b_4|)$ erasable memory,
 446 but we will show later that the local bandwidth $|b_3 - b_4|$ can be reduced.

447 **Bandwidth and Layout of the Hessian**

448 Obviously, the cost of the parallel computation will depend on the local bandwidth $|b_3 - b_4|$ of the batch
 449 $(b_1, b_2]$. The Hessian obtained with the default ordering of atoms can result in very large bandwidths with
 450 non-uniform patterns meaning interactions among sequentially distal parts within the tertiary structure or
 451 individual chains within a quaternary structure. A hypothetical minimal example is a macromolecule with

452 3 peptide chains α , β and γ , where only α - γ and β - γ interactions were found, but not α - β ; in this case, the
453 order α - β - γ will have a much larger bandwidth than α - γ - β . Many reasons can attribute to this observation.
454 An example found in the benchmark set is a chimeric Sesbania mosaic virus coat protein⁷²(PDBID:
455 4Y5Z) composed of 12 sets of pentamers arranged on the vertices of an icosahedron, where each protein
456 chain in the pentamer interacts with 7 other chains within 8 Å. (See Extended Data Figure 1) A systematic
457 order to build up this icosahedron, as done by the authors, is to place a pentamer on each of the four
458 corners of the three orthogonal golden rectangles, done one rectangle after another starting at the shorter
459 sides of the golden rectangles. (See Extended Data Figure 1a) In this case, pentamers on the shorter sides
460 interact within 8 Å among each other, but pentamers on the long sides of the same rectangle do not. This
461 creates exactly the situation where the aforementioned α - β - γ order arises. Indeed, the vertices on a
462 icosahedron can never be clustered satisfactorily and there are specialized algorithm to calculate their
463 vibrational dynamics⁷³. Besides, due to the inconsistent presence of water molecules, the exact symmetry
464 is destroyed in the crystal structure, which is commonly encountered. Nevertheless, the degeneracies in
465 modes due to the apparent symmetry of this macromolecule is captured in our program. (See Extended
466 Data Figure 1d) There are also cases where no meaningful sequential ordering is possible when intact
467 chemical structures (e.g. cofactors, ions, water, lipids) are intercalated between sequentially distal parts of
468 the polymer. A practical example in the benchmark set is a PsbM-deletion mutant of photosystem II⁷⁴
469 (PDBID: 5H2F), where elongated lipids and cofactors are integral part of the macromolecule complex
470 surrounded by several protein chains. In this case, it is not obvious as for how to rearrange the atoms to
471 reduce the bandwidth, but we can show that the Hessian can always be permuted to reduce bandwidth by
472 the algorithm described in the next section. (See Extended Data Figure 2)

473 **Bandwidth Reduction of the Hessian Matrix**

474 Finding a permutation to minimize the bandwidth is NP-complete⁴¹, but a reduction of bandwidth
475 and its overall profile can be approximated by algorithms such as the Reverse Cuthill McKee (RCM)
476 algorithm. The RCM is a greedy approximation with a breadth-first level structure. The input of a

477 standard RCM algorithm is an adjacency graph, and its outputs is a permuted node ordering. In RCM,
478 starting from a peripheral node with the lowest degree of connection, adjacent unvisited nodes are
479 collected as a level structure and re-ordered by their degree of connection. Provided that an adjacency
480 matrix K is precomputed, the time complexity of the RCM is bounded by $O(4|E| + 2cm|E| + N)$, where
481 $m =: \max_i \sum_j K_{ij}$ is the maximum degree among all nodes and $2|E|$ is the number of edges in K . A detail
482 proof is in reference⁴³. The term $4|E|$ is referring to $2|E|$ operations to determine the degree of each node
483 plus another $2|E|$ operations to sweep through the adjacency matrix to locate the neighbors. The term
484 $2cm|E|$ refers to the insertion sorting of the degree in retrieved neighbors. The last term N is due to
485 reversal of order. The input of RCM is an adjacency matrix K . For 3-D coordinates, this may be obtained
486 by a cell-linked list data structure in $O(27Nn_c)$ where n_c is the average number of particles per cell in
487 volume R_c^3 or simply an all-to-all calculation as done in ProDy2.4 for small-sized systems. Therefore, a
488 standard RCM algorithm operating on 3-D coordinate data will require a total time complexity of
489 $O(27Nn_c + 4|E| + 2cm|E| + N)$ and ideally a $O(mN)$ memory to operate due to the adjacency matrix.
490 To surrogate the storage of the adjacency matrix, we supplemented the RCM with a k-D tree data
491 structure⁴⁷, which takes the coordinate data $X^{(0)}$ directly as input. The algorithm is labelled as 3DRCM to
492 avoid confusion with the standard RCM. A pseudocode of our 3DRCM implementation is provided in the
493 Supplementary Information as **Algorithm 1**. A 3-D tree is a balanced space-partitioning binary tree, which
494 can be constructed and stored by finding and storing hyperplanes that split the number of points in halves.
495 As such, the construction of a 3-D tree takes $O(N \log N)$ time and $O(3N)$ space. On a 3-D tree, each range
496 search for adjacent neighbors within threshold distance takes $O(3N^{2/3})$ ⁷⁵. At this stage, the total time
497 complexity of 3DRCM appears to be $O(2 * 3N^{5/3} + 2cm|E| + N)$, where the first term $2 * 3N^{5/3}$ is due to
498 the collection of degree in each node and the search for neighbor, but we will show that in both 3DRCM
499 and RCM the term $O(2cm|E|)$ is dominating. A tighter bound on $O(2cm|E|)$ in the context of NMA can
500 be obtained as follows. For a stable macromolecule without atomic clashes, the shortest interatomic
501 distance is due to covalent bonds at around 1 Å. The Kepler's conjecture proven by Hales⁷⁶ states that the

502 maximum volume ratio occupied by equidistant sphere packing is $\frac{\pi}{3\sqrt{2}} < 0.7405$. Hence, the maximum
503 number of atoms allowable without clashes within radius R_C is $\rho =: R_C^3 \frac{\pi}{3\sqrt{2}} \geq m$. This packing density ρ
504 presents an upper bound to the number m of non-zero entries in a row on the adjacency matrix. This
505 bounds the total number of edges as

506
$$\rho N \geq mN > 2|E|$$

507 (Equation 18)

508
509 For an 8 Å radius, there can only be less than 380 atoms in absence of clash. Therefore, the time complexity
510 of 3DRCM and RCM in the context of NMA are bounded by $O(6N^{\frac{5}{3}} + c\rho^2N + N)$ and $O(27N\rho + 2\rho N +$
511 $c\rho^2N + N)$ respectively. Taking $c = 1$, the 3DRCM is dominated by the pseudolinear term $O(\rho^2N)$ unless
512 N is beyond 3.7 million atoms. From experience, for a system containing 2.4 million atoms, computing the
513 permutation ordering with 3DRCM and the calculation of the Hessian took only 5.2 minutes and 6.9
514 minutes in wall-clock time respectively. See Extended Data Figure 4. Hence, for most practical purpose,
515 the trade-offs in time complexity in 3DRCM compared to RCM is negligible.

516
517 **Low-complexity Krylov-Subspace Eigensolvers**
518 In this work we have implemented several iterative methods^{31,32,49,58} to solve large eigenvalue problems on
519 the GPU. Specifically, in NMA, we are mostly interested in the smallest non-zero eigenpairs. The Hessian
520 matrix of concern is a sparse positive semidefinite real symmetric matrix without weak diagonal dominance.
521 To facilitate communication, we adopt more generic notations here. The matrix of concern is notated as
522 $A \in \mathbb{R}^{n \times n}$; the exact and the approximate eigenpairs are $(\Lambda \in \mathbb{R}^{m \times m}, U \in \mathbb{R}^{n \times m})$ and $(\tilde{\Lambda} \in \mathbb{R}^{m \times m}, \tilde{U} \in$
523 $\mathbb{R}^{n \times m})$ respectively; n and m denotes the dimension of the basis set and the number of basis respectively.
524 The standard eigenproblem is thus $AU = U\Lambda$ and we assumed orthonormality among the exact

525 eigenvectors. The approximate $\tilde{\lambda} \in \tilde{\Lambda}$ can be obtained from the Rayleigh quotient $\tilde{\lambda} = \tilde{u}^T A \tilde{u}$ and the
 526 residual $r =: A\tilde{u} - \tilde{\lambda}\tilde{u}$ can always be evaluated by its 2-norm, the residual error $\|r\|_2$.

527 In the following paragraphs, we will progressively introduce two branches of iterative methods
 528 implemented, namely the Hermitian Lanczos Methods followed by the Jacobi-Davidson Method. Only
 529 rationales and key equations were presented. A common central idea is to find \tilde{u} by refining an initial guess
 530 $v_0 \in \mathbb{R}^n$ to build up an orthonormal basis $V \in \mathbb{R}^{n \times m}$ in the Krylov subspace

$$531 \quad \mathcal{K}_k(A, v_0) =: \text{span}\{v_0, A^1 v_0, A^2 v_0, \dots, A^{k-1} v_0\}$$

532 (Equation 19)

533

534 The residual is minimized when the Galerkin condition

$$535 \quad (Ve_j)^T (A\tilde{u} - \tilde{\lambda}\tilde{u}) = 0 \quad \forall j \in (0, \dots, m-1)$$

536 (Equation 20)

537

538 is satisfied, where $\tilde{u} =: Vy$. The $y \in \mathbb{R}^{m \times 1}$ is an unknown component to combine V , but if $V \in \mathcal{K}_k$ has its
 539 orthonormality maintained satisfactorily, for example by incorporating the Modified Gram Schmidt
 540 algorithm (MGS), then y is the eigenvector of a symmetric tridiagonal eigenproblem⁷⁷ of a much smaller
 541 size $m \times m$

$$542 \quad (V^T A V)y = \tilde{\lambda}y,$$

543 (Equation 21)

544

545 and $\tilde{u} \in \mathbb{R}^n$ can be recovered by definition. This orthogonal projection technique, an example of Rayleigh-
 546 Ritz process, is common to all three methods implemented.

547 **Technical remarks.** Given the recurring application of certain techniques in the implemented methods, an
 548 assortment of technical remarks is presented here before progressing further. The MGS can be applied more
 549 than once to prevent loss of orthogonality due to float point roundoffs and the procedure is called a full

550 reorthogonalization (FRO). A pseudocode of orthogonalizing a vector against a basis with MGS, and
 551 similarly with an Iterative Classical Gram-Schmidt (ICGS) algorithm is provided in the Supplementary
 552 Information as [Algorithm 2](#) and [Algorithm 3](#). The low-complexity $O(n^3)$ cost of the matrix-vector
 553 multiplications in producing \mathcal{K}_k can be amortized by parallelisms in GPU. When only the lower triangle
 554 L of A is available, the multiplication with an arbitrary vector x can be done as a sum decomposition $Lx +$
 555 $L^T x - Dx$, where $D =: \text{diag}(A)$. In our implementation, a custom kernel that utilizes the SpMV algorithm
 556 in CuSPARSE is written to accommodate the availability of the lower triangle, halving the memory
 557 requirement to access the Hessian. In all cases, the convergence rate of the i -th eigenpair is dictated by the
 558 ratio of adjacent exact eigenvalues $\left| \frac{\lambda_i}{\lambda_{i+1}} \right| \leq 1$, the smaller the better. Note that A was shifted upward as $A +$
 559 I to avoid poor condition number. Several strategies to improve the convergence rate for the smallest
 560 eigenpairs includes (1) the Shift-and-Inverse technique, where the Ritz pairs of $(A - \sigma I)^{-1}$ closest to σ
 561 was sought instead and the inverse is incorporated into the matrix-vector product by solving for $v^{(k+1)}$ in
 562 $(A - \sigma I)v^{(k+1)} = v^{(k)}$, (2) the implicitly shifted QR algorithm by Francis⁷⁸, where multiple shifts were
 563 applied to the subspace iteration problem typically concerning a small eigenproblem (See part b of
 564 [Algorithm 5](#)) and (3) filter diagonalization^{33,34,57} that magnifies convergence rate for regions of a spectrum
 565 (See [Algorithm 4,8](#) and [9](#) and later sections). Finally, the six exact eigenvectors U_6 corresponding to the
 566 rigid modes can be removed from \mathcal{K}_k by an explicit external deflation³⁶; the dense deflated matrix $A -$
 567 $\sigma_H U_6 I_6 U_6^T$, with eigenvalues corresponding to U_6 raised to σ_H , is not stored but incorporated into the
 568 matrix-vector multiplications; the sparse U_6 is stored in CSR format. This is implemented in all three
 569 methods and can be optionally called.
 570 **Hermitian Lanczos Method (HLM).** The $V \in \mathcal{K}_k$ described earlier can be obtained iteratively as v_{k+1} by
 571 projecting away components of previously obtained $v_i \forall i \in 0 \dots k$ from the current matrix-vector product
 572 Av_k . In the passing, the smaller eigenproblem $T =: V^T AV$ is also produced. Importantly, for a symmetric
 573 matrix A and $V \in \mathcal{K}_k$, the T is symmetric tridiagonal. Therefore, the recursion to obtain v_{k+1} simplifies to
 574 three terms

575 $v_{k+1} = Av_k - \beta_k v_{k-1} - \alpha_k v_k$

576 (Equation 22)

577

578 Where $\alpha_k =: v_k^T A v_k$ and $\beta_k =: v_{k-1}^T A v_k$ are the main diagonal and the first subdiagonal of T . This is the
 579 essence of the HLM proposed by Lanczos⁷⁹. Collecting the recursion and a rearrangement reveals the
 580 Lanczos factorization stopping at the k -th step.

581 $AV_k = V_k T_k + \beta_k v_{k+1} e_k^T$

582 (Equation 23)

583

584 A pseudocode of a p-step Lanczos Factorization is provided in the Supplementary Information as [Algorithm](#)
 585 [4](#). Note that the approximate eigenvectors obtained from the HLM will converge to those with extremal
 586 eigenvalues, i.e., both the largest and the smallest eigenpairs, but only the smallest eigenpairs are wanted
 587 in our problem setting. Further, while the basis V_k builds up in each iterative step, eventually we will
 588 experience overflow in storage, likely before convergence. To address these concerns, two practical variants
 589 of the HLM with different restarting techniques were proposed.

- **Implicitly Restarted Lanczos Method (IRLM).** The IRLM algorithm was proposed by Lehoucq
 591 and Sorensen.⁴⁹ In IRLM, the Lanczos factorization is supplemented with an implicitly shifted QR
 592 algorithm by Francis⁷⁸. The purpose is to shift the smaller eigensystem $T^{(j-1)}$ at each restart round
 593 j by the current approximation of the $m - k$ unwanted eigenvalues such that the convergence rate
 594 of the k wanted smallest eigenvalues is improved. The only information we need to determine these
 595 shifts is the sorted approximated eigenvalues and the number of wanted eigenvalues we desired.
 596 As a result, after obtaining each QR factorization $Q^{(j)} R^{(j)} = T^{(j-1)} - \tilde{\lambda}_j I$, the Lanczos
 597 factorization (Equation 23) is modified as

598 $A\hat{V}_k = \hat{V}_k \hat{T}_k + v_{k+1} \hat{b}$

599 (Equation 24)

607

600 where $\hat{b} = Q^{(j)T} e_k$ and $\hat{T} = R^{(j)} Q^{(j)} + \tilde{\lambda}_k I$ and the first k orthonormal basis is updated as $\hat{V}_k = V_k Q^{(j)}$. This is equivalent to performing $m - k$ simple polynomial filterings.⁸⁰ In our
 601 implementation, to promote parallelism, the $m - k$ QR factorizations was performed before a
 602 cumulative update of the matrices; this is followed by a full reorthogonalization of the updated
 603 basis V_k . Note that in IRLM the residual error $\|r\|_2$ is bounded by the β_{k+1} , a result due to Paige⁸¹
 604 such that $\|r\|_2$ needs not be computed. A pseudocode of our IRLM implementation is provided in
 605 the Supplementary Information as [Algorithm 5](#).

- **Thick-Restart Lanczos Method (TRLM).** The TRLM algorithm was proposed by Wu and Simon⁵⁸. In TRLM, the smaller eigenproblem T is again solved with its eigenpairs sorted, but rather than performing implicit shift, the T is projected onto its wanted eigenvectors Y_k as $\hat{T} = Y_k^T T Y_k$ and the basis updated to $\hat{V} = V Y_k$ accordingly. As a result, the Lanczos factorization
 611 (Equation 23) is modified as

$$613 \quad A\hat{V}_k = \hat{V}_k \hat{T}_k + \beta_k v_{k+1} e_k^T Y$$

614 (Equation 25)

615 When we want to find the next v_{k+1} by orthogonalizing $A v_k$ against the previous \hat{V}_k , these
 616 transforms allow us to compute it with $\beta_{k-1}(Y^T e_{k-1})$ known from the restart

$$617 \quad \hat{v}_{k+1} = A\hat{v}_k - \hat{\alpha}\hat{v}_k - \hat{V}_{k-1}\beta_{k-1}(Y^T e_{k-1})$$

618 Full reorthogonalization was done in the basis \hat{V}_k . A pseudocode of our TRLM implementation is
 619 provided in the Supplementary Information as [Algorithm 6](#).

620

621 **Jacobi-Davidson Method (JDM).** The JDM algorithm was proposed by Fokkema, Sleijpen and Van der
 622 Vorst³². Similar to the HLM, the JDM also considers approximations in the Krylov subspace, but rather
 623 than refining the approximations with a Lanczos factorization, it attempts to solve the following equation

$$624 \quad A(\tilde{u} + z) = (\tilde{\lambda} + \eta)(\tilde{u} + z)$$

625 (Equation 26)

626 where an approximate solution pair (η, z) to correct the approximate eigenpair $(\tilde{\lambda}, \tilde{u})$ can be obtained by
627 incorporating the Galerkin condition $(\tilde{u} + z)^T z = 0$ into projectors resulting in the following correction
628 equation

$$629 \quad (I - \tilde{u}\tilde{u}^T)(A - \tilde{\lambda}I)(I - \tilde{u}\tilde{u}^T)z = -r$$

630 (Equation 27)

631 The correction equation does not need to be solved exactly such that A in this equation can be replaced by
632 a preconditioner of A or by simply finding a solution of z to certain extent of precision. For the NMA
633 eigenproblems, we cannot find a satisfactory preconditioner that is not dense such that GPU memory is not
634 overflowed. Therefore, in our implementation, the latter strategy was adopted by restricting the number of
635 steps or precision in the solution in the generalized minimal residual iteration (GMRES), which is also a
636 Shift-and-Inverse example. An implementation of the GMRES taking a sparse matrix input was modified
637 from the CuPy package. A pseudocode of our JDM implementation is provided in the Supplementary
638 Information as Algorithm 7.

639 **Filter Diagonalization**

640 The Lanczos and Davidson methods can be adjusted by filters to isolate certain eigenvalues. The motivation
641 of this Filter Diagonalization (FD) approach is to transform the spectrum such that eigenvalues from wanted
642 intervals become dominant^{33,82}. (See Extended Data Figure 10b for an illustration.) For example, to obtain
643 the lower spectrum, the Chebyshev-Davidson Method³⁵ (CDM) removed the Jacobi correction in JDM
644 (which approximates a moving rational filter) and applied a moving low-pass, fixed-degree, Chebyshev
645 polynomial of the first kind to the input matrix. (Algorithm 8) FD can also be applied to solve for the interior
646 rather than the extremal eigenpairs.^{34,57} This can be useful when we are extracting a large amount of
647 eigenpairs from the lower end of the spectrum, as then the interior slices of the spectrum can be obtained
648 without storing and orthogonalizing against the filtered subspace from the very lowest end. To achieve this,
649 a band-pass filter, which is an M -degree Chebyshev expansion of a Dirac-delta-like function damped by a

650 Jackson kernel^{33,83}, was developed in the EVSL library^{34,57} and incorporated into our INCHING-CTRLM
 651 protocol ([Algorithm 6](#), shared with INCHING-TRLM protocol). The band-pass filter $p(t)$ were obtained
 652 following the three-term recurrence of T_j , the j -th degree Chebyshev polynomial of the first kind.

653
$$T_{j+2}(t) = 2t T_{j+1}(t) - T_j(t); T_0(t) = 1; T_1(t) = t;$$

654
$$p(t) = \sum_{j=0}^{j=M} g_j^{(M)} \mu_j T_j(t) / \sum_{j=0}^{j=M} g_j^{(M)} \mu_j T_j(\cos(\theta))$$

655 [\(Equation 28\)](#)

656 The coefficients μ_j in the expansion and the damping kernel $g_j^{(M)}$ were precalculated with the following
 657 equations, where δ_{j0} is the Kronecker delta and $\tilde{\pi} := \pi/(M+2)$ and θ is the adjusted center of the
 658 arccosine of the corresponding transformed wanted interval $[\alpha_s, \beta_s]$ within $[-1, 1]$. (See reference^{34,57} for
 659 further details.)

660
$$g_j^{(M)} =: \frac{\sin((j+1)\tilde{\pi})}{(M+2)\sin(\tilde{\pi})} + \left(1 - \frac{j+1}{M+2}\right) \cos(j\tilde{\pi})$$

661
$$\mu_j = -\frac{1}{2}\delta_{j0} + \cos(j\theta)$$

662 [\(Equation 29\)](#)

663 The filter was applied during matrix-vector multiplication ([Algorithm 9](#)), where the spectrum of A were
 664 scaled and shifted to the range of cosine $[-1, 1]$ using spectral bounds from the Lanczos process (i.e.
 665 [Algorithm 4](#)) following reference³⁵.

666 Implementation notes

667 In many parts of the algorithms presented above, matrix-vector multiplication, whether dense or sparse, is
 668 often the calculation bottleneck. In our implementations, this shortcoming is amortized by capitalizing on
 669 the high degree of parallelism in GPU and existing techniques built around NVIDIA® GPUs, including
 670 CUDA⁶¹ (v. 11.6.2), cuBLAS⁶², and cuSPARSE⁶³. To facilitate installation, code readability and version

control, the CuPy package (v. 11.5) was used as an application programming interface to access these technologies. In all the algorithms implemented, only the lower triangle of the Hessian matrix was required as input in CSR format. Our INCHING-JDM and INCHING-IRLM implementations were written with reference to the thesis of Geus⁸⁴ and the ARPACK package⁵⁰ respectively. Our INCHING-TRLM implementation is modified from the CuPy library’s default, where memory footprint was optimized by exploiting sum decomposition of the lower triangle; further speedup was achieved by reducing the number of transposes done to the basis set. Where a smaller projected eigenproblem has to be solved, the calculation was done on CPU by calling ‘numpy.linalg.eigh’ from the NumPy (v. 1.23.5) package⁸⁵. For methods with Chebyshev filter, the implementations in the EVSL library⁵⁷ and the Chebyshev-Davidson method³⁵ were referenced. When a band-pass Chebyshev filter is invoked, the converged eigenvectors are sorted again before off-loading for storage. For benchmarks with ProDy, the ProDy (v.2.4) package⁴⁸ was installed with the NumPy (v. 1.23.5) package⁸⁵ and the SciPy (v.1.8.0) package⁸⁶. Dense eigenproblems in ProDy are solved using a LAPACK backend called through NumPy. Sparse eigenproblems in ProDy are solved using a ARPACK backend called through SciPy ‘scipy.sparse.linalg.eigsh’. Benchmarks on correctness, memory and speed were conducted on a single piece of A100 NVIDIA® GPU with tensor core activated and 80GB GPU memory capacity and the AMD EPYC™ 7513 processor with 64 threads. For systems with more than 2.1 billion non-zero entries, 64-bit integers were used for indexing in the CSR format, otherwise 32-bit integers were used by default. Methods were also tested on a RTX4090 NVIDIA® GPU with 24GB memory and 64-bit indexing used throughout.

690 **Hyperparameters in calculations**

691 In all cases, the following hyperparameters were used unless otherwise stated. We followed the work of
692 Koehl³⁰ taking the neighborhood cutoff threshold $R_C = 8\text{\AA}$ for atomic systems. After performing 3DRCM,
693 the indexing slice for each batch of atom was analyzed for presence of gaps, i.e., regions where no neighbor
694 of the batch is present. The indexing slice was then split into multiple contiguous indexing slices by
695 removing gaps that extend for more than 100 atoms. For INCHING-IRLM, the tolerance of error bound is

696 set to 10^{-10} . For both INCHING-TRLM and INCHING-JDM, the tolerance of residual error is 10^{-12} . The
697 correction equation in INCHING-JDM was solved using the GMRES algorithm, where only 20 steps of
698 minimization at max were allowed. The default maximum allowed number of restarts for INCHING-IRLM,
699 INCHING-TRLM and INCHING-JDM is 15000 steps, but for $R_C = 6\text{\AA}$ benchmarks, at max 30000 steps
700 were allowed to accommodate difficult convergence in some cases. The maximum allowed basis in the
701 Krylov subspace is three times the desired number of output eigenpairs, i.e. which is $64 \times 3 = 192$ vectors
702 in the benchmark. The 1000 modes presented in [Extended Data Figure 9](#) (200 modes in [Extended Data](#)
703 [Figure 10](#)) were calculated using INCHING-CTRLM in batches of 64 modes with 128 basis (28 modes
704 with 56 basis) for all structures, though larger basis were affordable; external explicit deflation of the free
705 modes were applied when low-pass filter is used. To scan through the lower end of the spectrum using
706 INCHING-CTRLM, we begin by applying the low-pass filter to obtain the largest of the smallest m
707 eigenvalues α_s and set $\alpha_s - 10^{-10}$ as the left bound of the next interior wanted interval $[\alpha_s, \beta_s]$; the
708 process continues with the band-pass filter until a desired number of eigenpairs are obtained. Note that a
709 binary search was performed to locate β_s such that $[\alpha_s, \beta_s]$ contains $< m$ eigenvalues for the band-pass
710 case and $< 5m$ eigenvalues for the low-pass case. The polynomial degrees were also maximized to
711 maintain the smallest transformed eigenvalues in $[\alpha_s, \beta_s]$ as 0.7 for both filters such that convergence rate
712 remained constant. (See [Extended Data Figure 10b](#)). All programs were stopped if convergence was not
713 achieved within 48 hours, wall-clock time. All INCHING programs were stopped if the maximum number
714 of restarts were exceeded. For all benchmarks with $R_C = 14\text{\AA}$, 64-bit integers indexing were used for
715 consistency.

716 Benchmark coordinates

717 While ideally all the structures on the Protein Data Bank can be considered, we randomly selected structures
718 at an increasing interval. For systems with less than 100 thousand atoms, structures were downloaded in
719 the PDB format; the interval of increase is around 1 thousand atoms. For systems with more than 100
720 thousand atoms, structures were downloaded as mmCIF format; the interval of increase is around 10

721 thousand atoms. Note that while the iterative methods all work very well, some structures can contain
722 components not connected within 8 Å threshold, resulting in more than 6 rigid modes. This happens in a
723 small portion of PDB structures when crystallographic water(s) or protein chain(s) with no neighbor within
724 8 Å were scrupulously put into the crystallographic map. These structures were removed from consideration
725 to avoid confusion. Overall, 85 structures with less than 100 thousand atoms and 31 structures with more
726 than 100 thousand atoms were tested. This benchmark set contains 116 structures in total with number of
727 atoms ranging from around 1 thousand atoms to 2.4 million atoms.

728 **Megascle atomic structures**

729 We applied our method to some of the largest atomic objects available. The mature HIV-1 capsid structure
730 (PDBID:[3J3Q](#)) containing around 2.4 million atoms was obtained from the Protein Data Bank³⁷ without
731 any modification. The human Nuclear Pore Complex structure ([EMDB: EMD14321](#), PDBID: [7R5J](#)) was
732 downloaded as a bioassembly from the Protein Data Bank³⁷ and chains LA,MA,NA,OA with clashing
733 linkers were removed; the final structure contains 4.8 million atoms. The atomic structure of the 26 million
734 atoms Faustovirus capsid ([PDBID:5J7V](#)) was downloaded from the Protein Data Bank; C α were extracted
735 from the structure resulting in a 5 million pseudo-atom coarse-grained representation; NMA with $R_C = 14\text{\AA}$
736 was performed with INCHING-CTRLM-HalfSparse with external explicit deflation of the 6 free modes;
737 the first 64 non-zero modes were obtained. The atomic structure of the DNA airplane⁶⁷ was obtained from
738 Nanobase.org⁸⁷ and the atomic coordinates were reconstructed using TacoxDNA⁸⁸; the final structure
739 includes hydrogens and contains 1.8 million atoms. The NMA of HIV-1 capsid structure were calculated
740 using INCHING-JDM-HalfSparse and the same hyperparameters of numerical methods outlined above.
741 For the DNA airplane and the NPC complex, the NMA was calculated using INCHING-JDM-HalfSparse
742 with external explicit deflation of the 6 free modes, the number of modes to resolve, the maximum allowed
743 basis in the Krylov subspace, the maximum number of allowed steps in GMRES and the maximum number
744 of restarts allowed were revised to the first 30 non-zero normal modes, 120 vectors, 150 GMRES steps and
745 550000 restarts respectively, otherwise all other hyperparameters are the same. The megascle structures
746 were displayed using PyMOL (v2.4.1).

747 **A logistic kernel to visualize atomic motion**

748 In all the iterative methods, the resultant eigenvector matrix is orthonormalized, which means as size of
749 the system increases a decreasing magnitude of atomic displacement vector will be experienced. In our
750 visualization module, to avoid this scaling problem, a logistic kernel is applied to fine-tune the atomic
751 magnitude x_i of the eigentensor. First, to avoid eccentricity in the atomic magnitudes, they are clipped
752 between (0.025, 0.975) quantile of the magnitudes. Second, a logistic kernel is applied on the clipped
753 magnitude x_i for the i-th atom

754
$$y_i = G(x_i|\vartheta) = \frac{1}{(1 + \vartheta e^{-x_i})} < 1$$

755 **(Equation 30)**

756 ,where $\vartheta = 0.05$. To remove the effect of scaling on the normalized eigenvector as the number of atom
757 increase, the output magnitudes are further centered at $\tilde{y}_i =: \frac{y_i - y^-}{y^+ - y^-} \leq 1$, where (y^-, y^+) are the minimum
758 and maximum of $y_i \forall i$.

759 **Data Availability**

760 All atomic coordinates of macromolecules in the benchmark dataset were accessed from the Protein Data
761 Bank. The benchmark dataset, including the atomic coordinates and supplementary data files on the
762 calculated eigenvalues, speed, accuracy from elastic network model can be downloaded from
763 [10.5281/zenodo.8061217](https://zenodo.10.5281/zenodo.8061217).

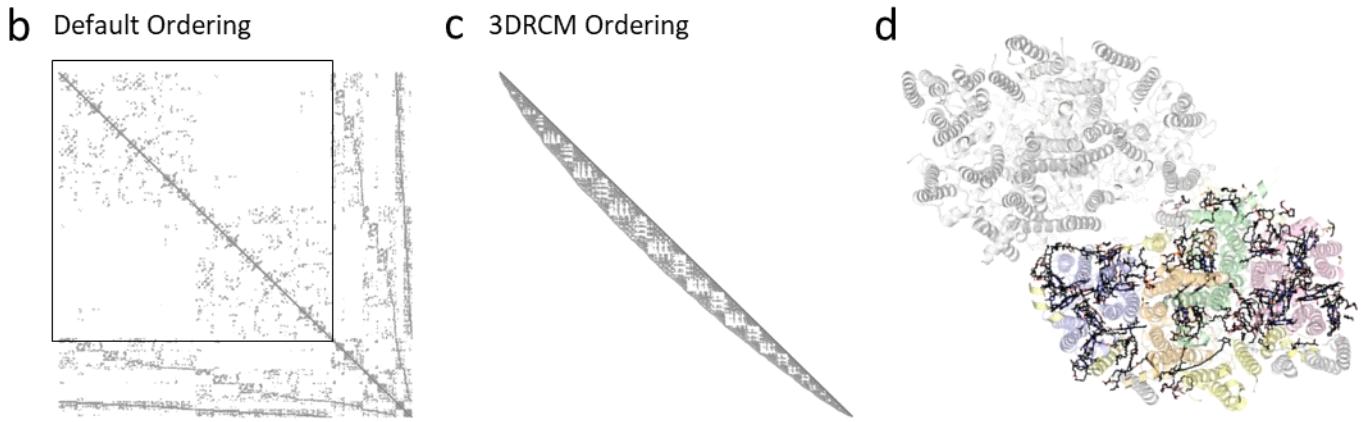
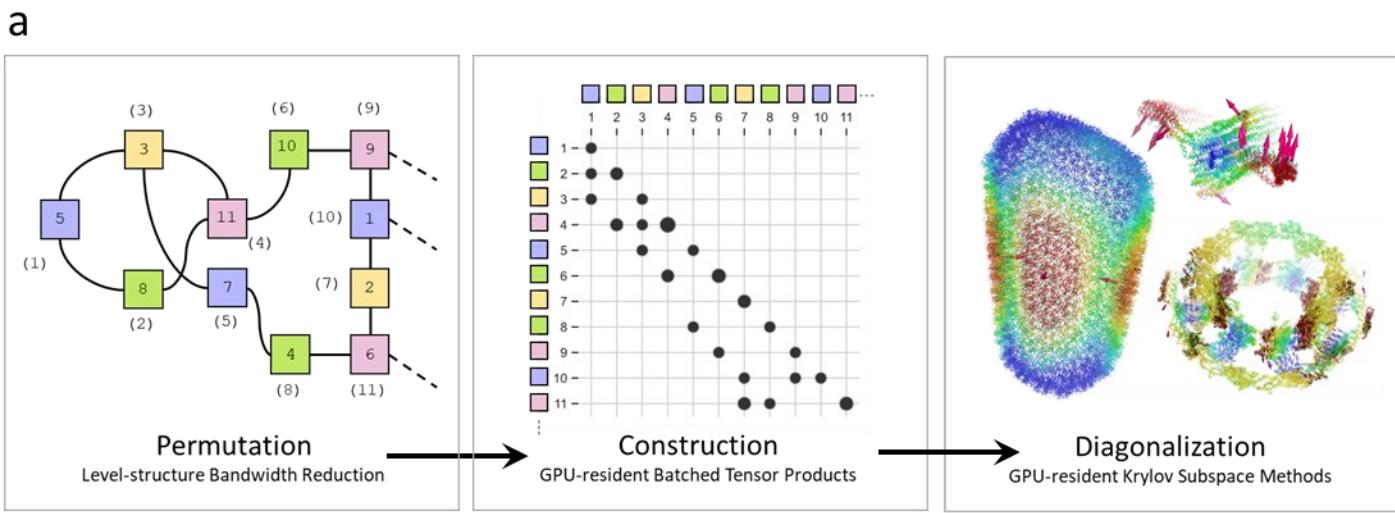
764 **Code Availability**

765 The INCHING code is available at <https://github.com/jhmlam/Inching>. Jupyter notebooks and Python
766 scripts for the experiments and analyses presented in the paper are available. Frozen versions of the
767 software and associated code for analysis are also available in the [10.5281/zenodo.10443729](https://zenodo.10.5281/zenodo.10443729).

768 **Acknowledgement**

769 This study was supported by internal funding from USC Dornsife college to V.K., and an NSF grant,
770 OAC-2118061 to A.N. The authors acknowledge the Center for Advanced Research Computing (CARC)
771 at the University of Southern California for providing computing resources that have contributed to the
772 research results reported within this publication. URL: <https://carc.usc.edu>

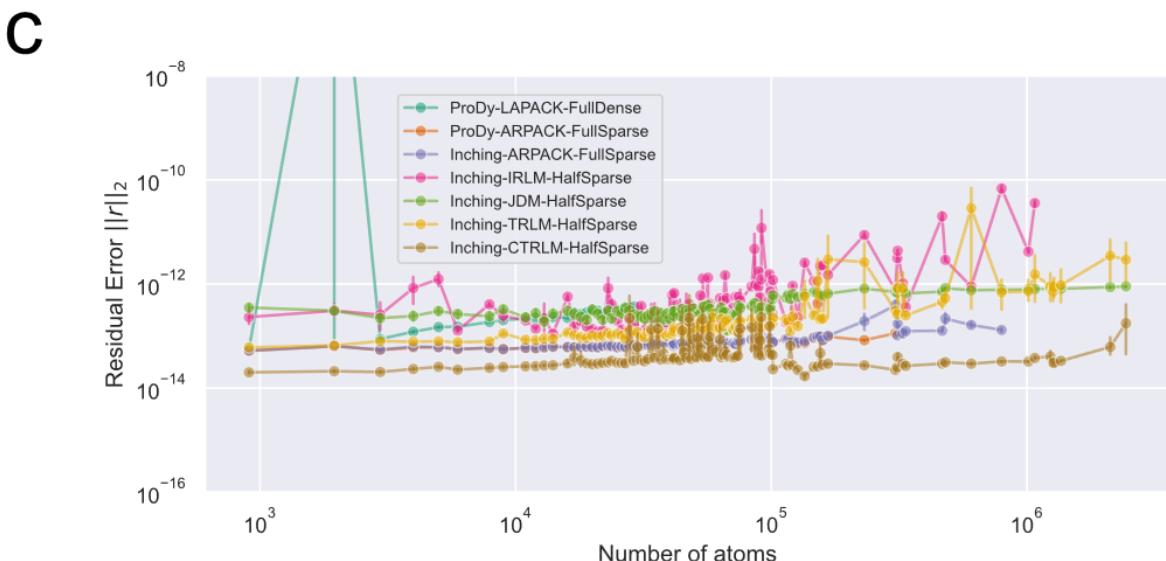
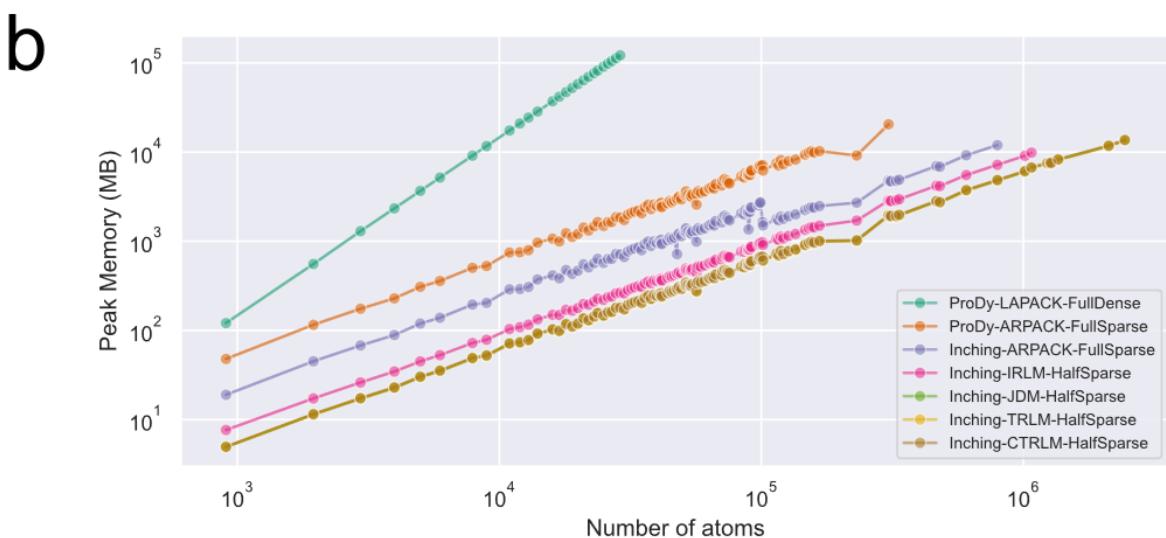
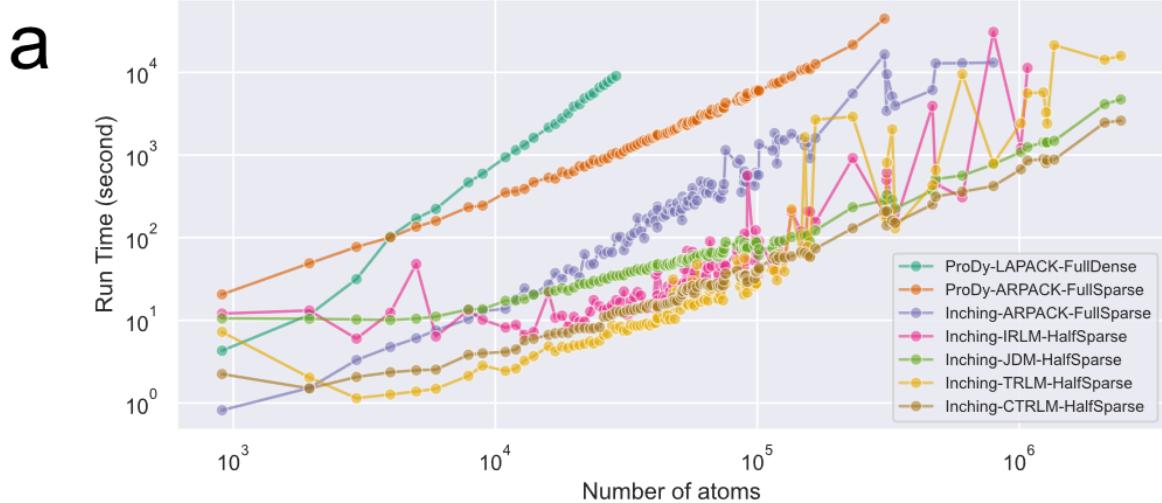
773



774

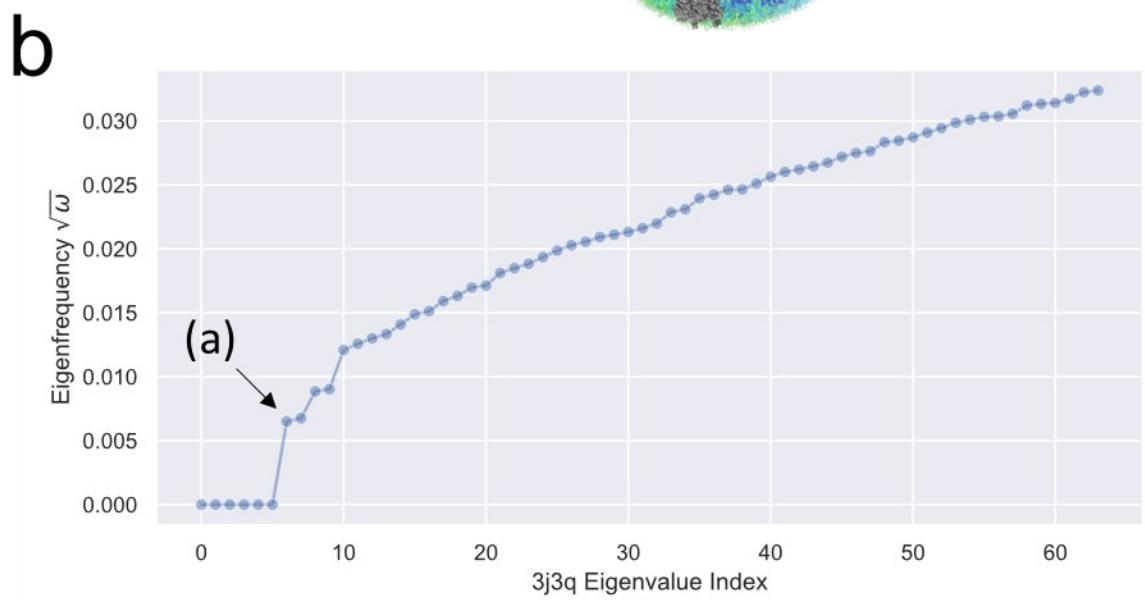
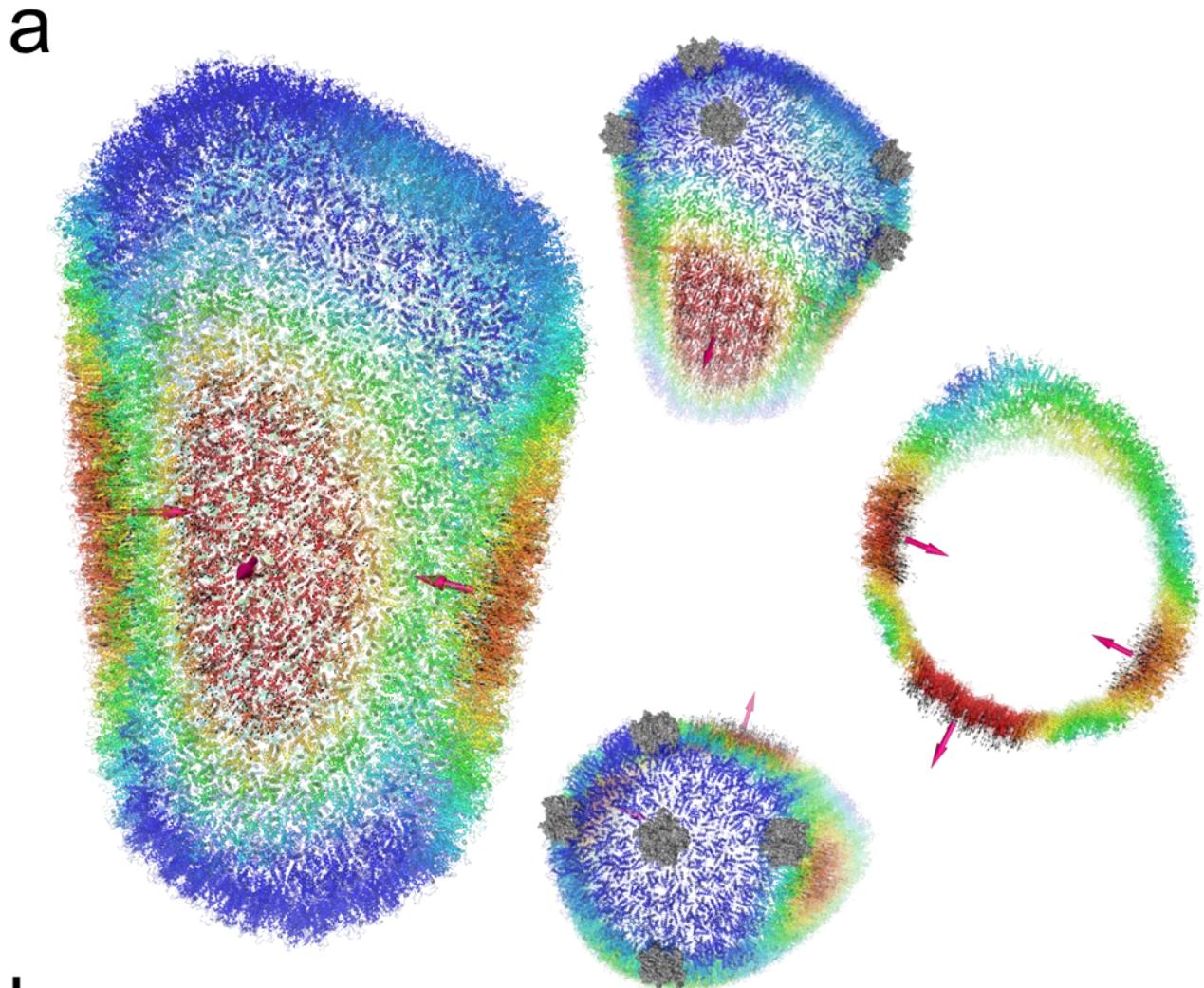
775 **Figure 1. Overview of the INCHING algorithm** (a) Our INCHING algorithm consists of three stages,
 776 namely, permutation, construction, and diagonalization. In the permutation stage, 3DRCM takes atom
 777 coordinates as the input and produce a bandwidth-reduced indexing as output. The initial atom ordering is
 778 denoted within square-shaped nodes, while the 3DRCM-permuted atom ordering is represented using
 779 parentheses. In the construction stage, the lower triangle of a Hessian with globally-sparse-yet-locally-
 780 dense pattern is constructed incrementally on GPU with tensor products and broadcasts. In the
 781 diagonalization stage, the eigenproblem is solved on GPU to produce depictable mode shapes. (b)
 782 Sparsity pattern of the Hessian with default atom ordering for the PsbM-deletion mutant of photosystem

783 II. The top left square highlighted in correspond to the proteins in the macromolecular complex, which is
784 also comprises cofactors, lipids and water molecules. (c) Sparsity pattern of the Hessian lower triangle
785 with 3DRCM permuted atom ordering for the same complex (d) Overall spatial distribution of the
786 chemicals intercalated in multiple protein chains in default sequential order (green, blue, pink, orange,
787 yellow). Only one of the dimeric halves of the structure is colored, the rest of the macromolecule is shown
788 as a grey surface; note that the complex is asymmetric due to slight difference in lipids and cofactors
789 intercalated. The Hessian matrix in default ordering has a much longer bandwidth everywhere than that
790 with the 3DRCM ordering.



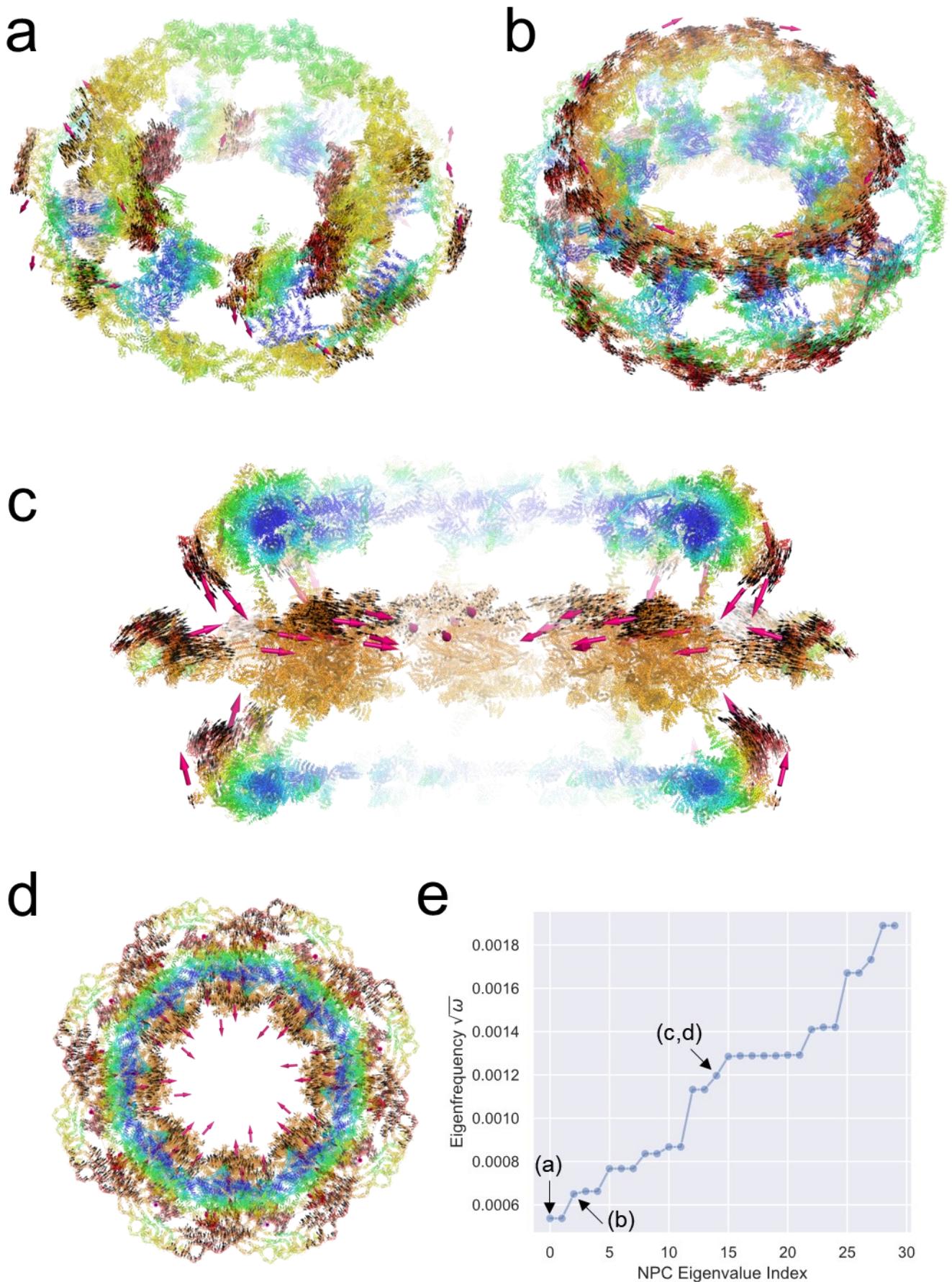
792 **Figure 2. Benchmark on throughput, memory consumption and correctness for each**
793 **macromolecular structure in the benchmark dataset.** For all methods, the first 64 eigenpairs were
794 calculated. Methods tested includes 2 ProDy methods (i.e., “ProDy-LAPACK-FullDense” and “ProDy-
795 ARPACk-FullSparse”) and 5 INCHING methods (i.e., “INCHING-TRLM-HalfSparse”, “INCHING-
796 CTRLM-HalfSparse”, “INCHING-JDM-HalfSparse”, “INCHING-IRLM-HalfSparse”, “INCHING-
797 ARPACk-FullSparse”). Text in the name describe the eigensolver (e.g., ARPACK, LAPACK, JDM,
798 TRLM, CTRLM, IRLM), the access of the Hessian matrix (e.g., “Full” means that the whole Hessian
799 matrix is accessed and stored; “Half” means that only the lower triangle of the Hessian matrix is accessed
800 and stored.), and the storage format of the Hessian matrix (e.g., “Dense” means a 2-D array is stored in
801 double precision; “Sparse” means a CSR format is stored in double precision). “TRLM”, “CTRLM”,
802 “JDM”, “IRLM” are our implementation of the Thick Restart Lanczos Method (TRLM), Chebyshev-
803 filtered Thick Restart Lanczos Method (CTRLM), Jacobi-Davidson Method (JDM) and Implicitly
804 Restarted Lanczos Method (IRLM). (a) Benchmark on overall run time including Hessian realization and
805 subsequent diagonalization. The run time is wall-clock time to complete all the calculation. (b)
806 Benchmark on peak memory consumption. Note that for “ProDy-LAPACK-FullDense”, we were not able
807 to proceed once there are more than 29 thousand atoms due to memory overflow. Note that for “ProDy-
808 ARPACk-FullSparse”, we were not able to proceed once there are more than 305 thousand atoms as it
809 takes more than 48 hours to converge. Also note that “INCHING-TRLM-HalfSparse”, “INCHING-
810 CTRLM-HalfSparse” and “INCHING-JDM-HalfSparse” share very similar peak memory requirement.
811 (c) Benchmark on residual error. The error bar presented is the 95% confidence interval calculated from
812 the residual error of all the eigenvalues of a macromolecular structure in the benchmark dataset. Note that
813 for “ProDy-LAPACK-FullDense”, the second macromolecular structure (PDBID: 1A8L) were not able to
814 converge within 10^{-10} likely due to severe fill-ins. All programs were stopped if run time exceeds 48
815 hours. All INCHING programs were stopped if number of restarts exceeded 15000 rounds.

816



818 **Figure 3. Vibrations of the mature HIV-1 capsid structure.** (a) The first non-zero mode of the capsid.
819 The black arrows are the displacement field of 1000 atoms randomly chosen from those in the top 90%
820 quantile of vibration magnitude in the eigenvector. The arrow in magenta indicates an average direction
821 for local clusters of the displacement field. Color scale in the cartoon from blue to red indicates increasing
822 magnitude of vibration. Note that a logistic kernel is applied to the eigenvector to control the magnitude
823 in visualization. (See Methods). The top-right inset is a clipped view of the capsid. The bottom-right inset
824 indicates the locations of the pentamers (Grey opaque surface). (b) The first 64 eigenfrequency of the
825 macromolecule. The black arrow indicates the first non-zero eigenmode displayed in (a).

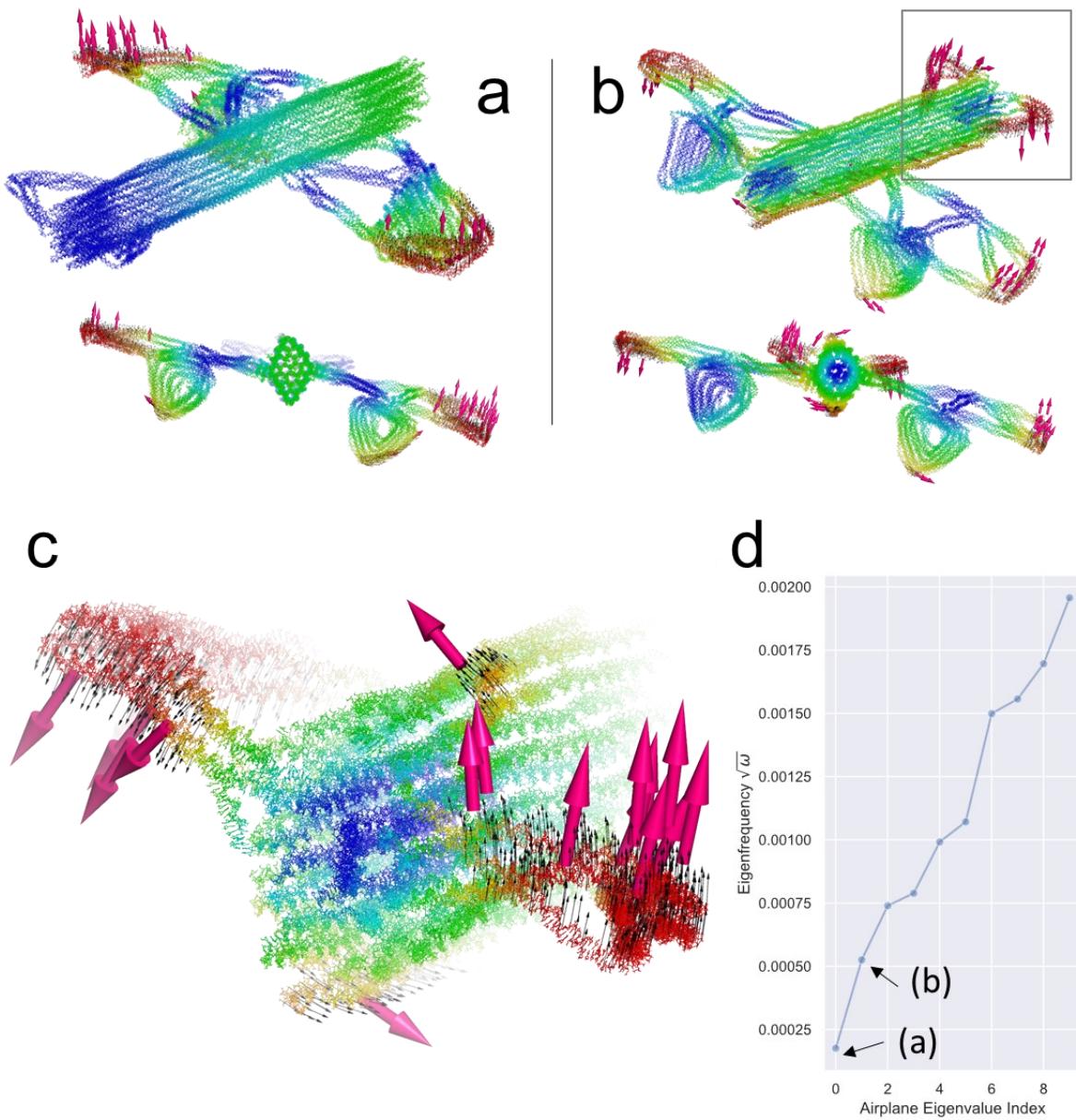
826



828

829 **Figure 4. Vibrations of the dilated human Nuclear Pore Complex.** Overview of motions in several
830 representative non-zero modes. (a) The first non-zero normal mode (indexed as Mode 0) (b) the third
831 non-zero normal mode (indexed as Mode 2) (c) Clipped view of the fifteenth non-zero normal mode
832 (indexed as Mode 14) (d) Bird-eye view of the fifteenth non-zero normal mode (indexed as Mode 14)
833 The arrows in magenta indicate an average directions for the local clusters of the displacement field,
834 while the tiny black arrows show more detailed displacement fields of 10000 atoms randomly chosen
835 from those in the top 80% quantile of vibration magnitude in the eigenvector. Color scale in the cartoon
836 from blue to red indicates increasing magnitude of vibration. Note that a logistic kernel is applied to the
837 eigenvector to control the magnitude in visualization. (See Methods). (e) The first 30 non-zero
838 eigenfrequencies of the macromolecule. The black arrows indicate the first non-zero eigenmode displayed
839 in (a), (b), (c) and (d).

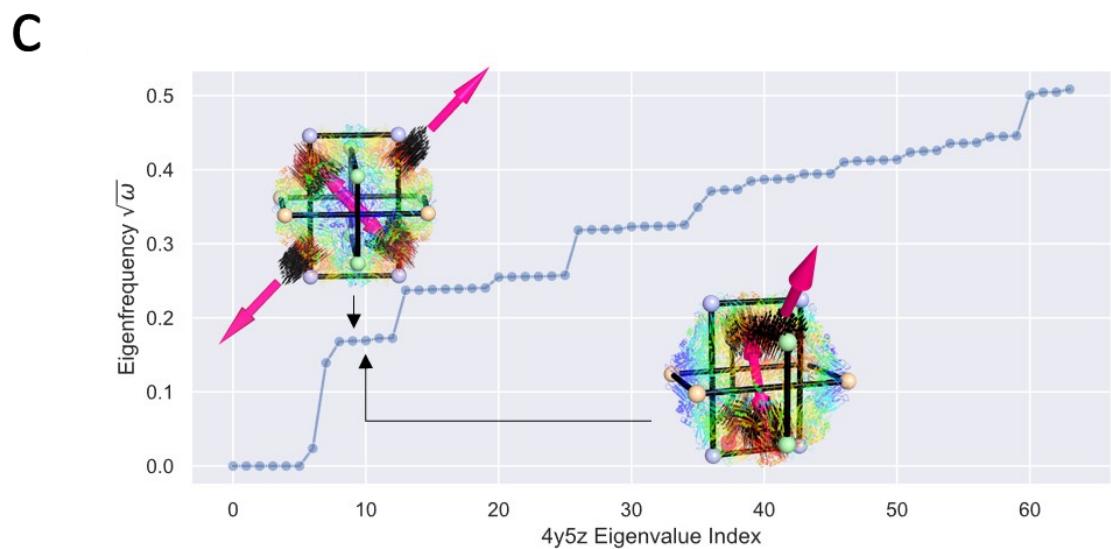
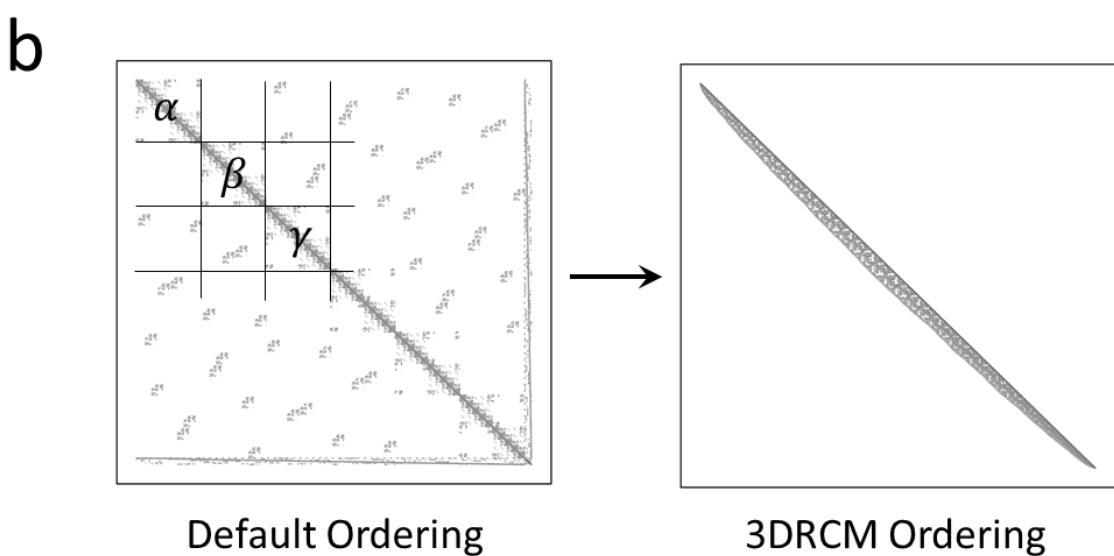
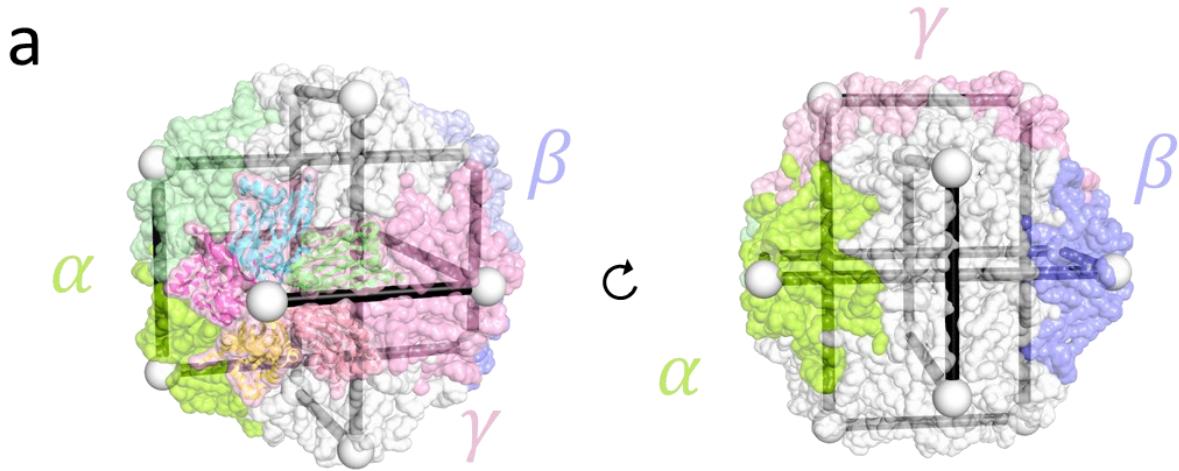
840



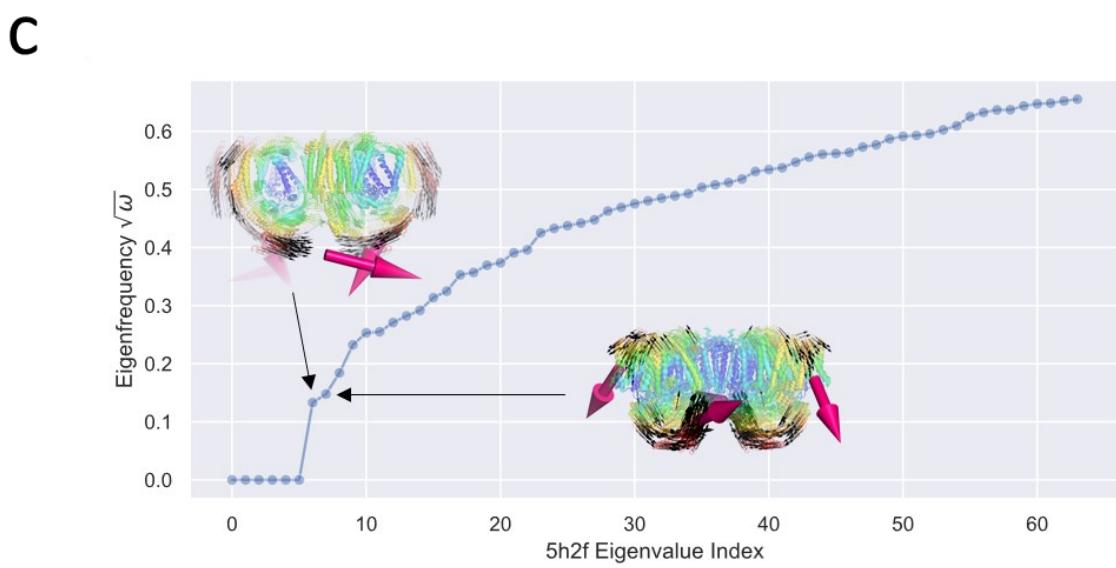
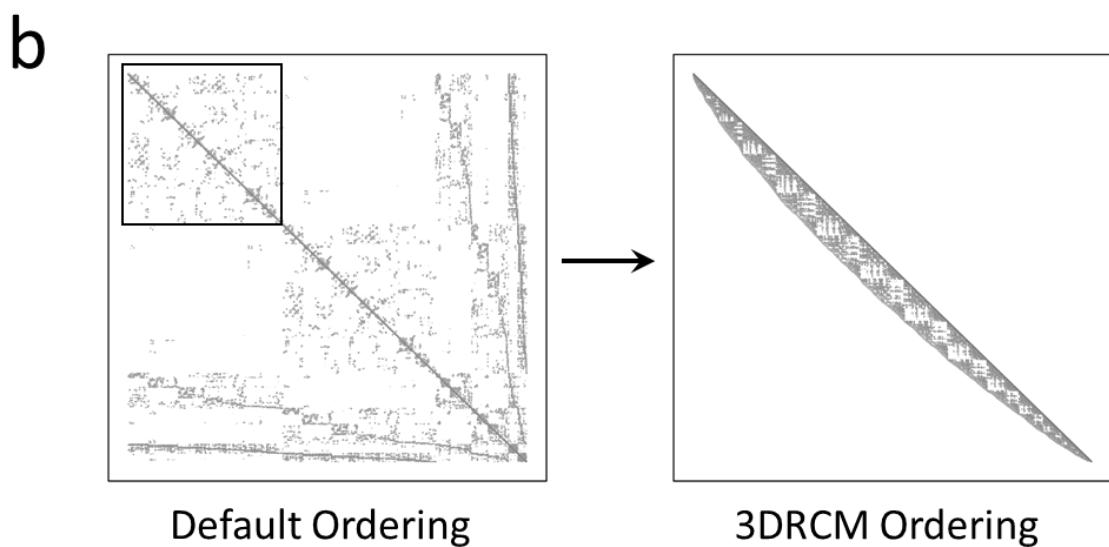
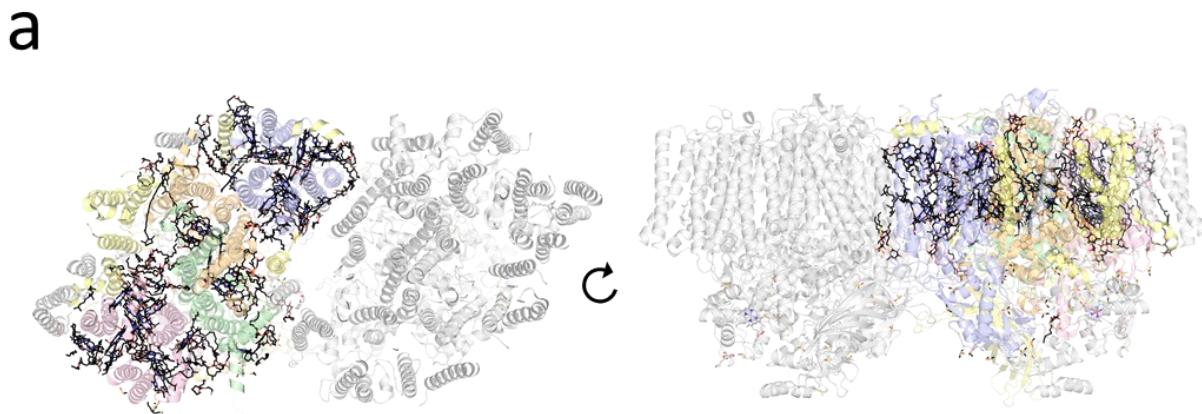
841

842 **Figure 5. Vibrations of a DNA origami airplane.** Overview of motions in the first two non-zero modes
 843 of the airplane. (a) The first non-zero eigenmode. (b) The second non-zero eigenmode; the square border
 844 indicates the stabilizer, which is also shown in (c). Color scale in the cartoon from blue to red indicates
 845 increasing magnitude of vibration. Note that a logistic kernel is applied to the eigenvector to control the
 846 magnitude in visualization. (See Methods). The bottom inset show the airplane along its apparent bilateral
 847 symmetry axis. (c) A zoom into the motion of the stabilizer in the second non-zero mode. The black
 848 arrows are the displacement field of 1000 atoms randomly chosen from those in the top 90% quantile of

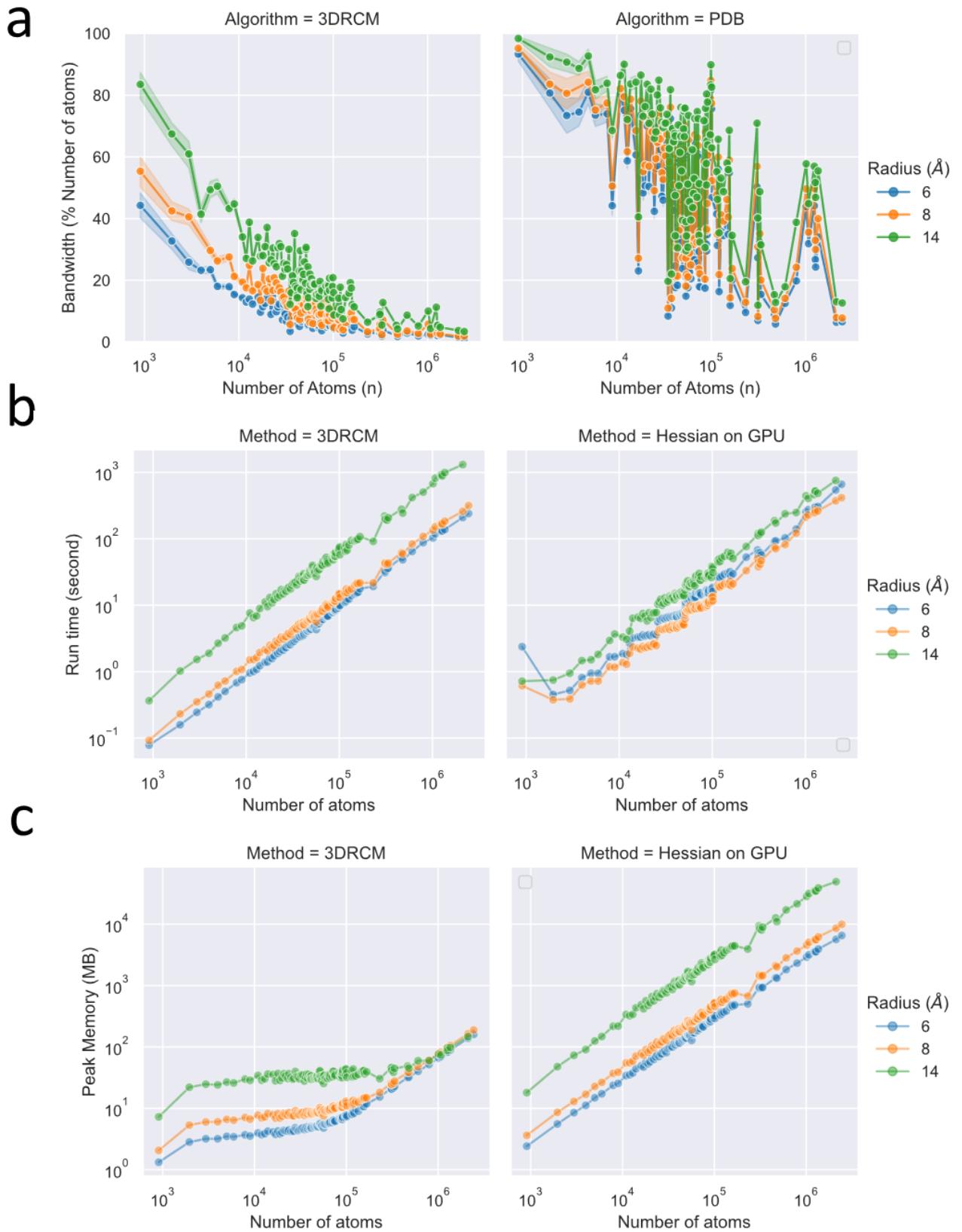
849 vibration magnitude in the eigenvector. The arrow in magenta indicates an average direction for local
850 clusters of the displacement field, which also correspond to (a). (d) The first 10 non-zero eigenfrequencies
851 of the macromolecule. Note that explicit external deflation were applied to remove the rigid modes.



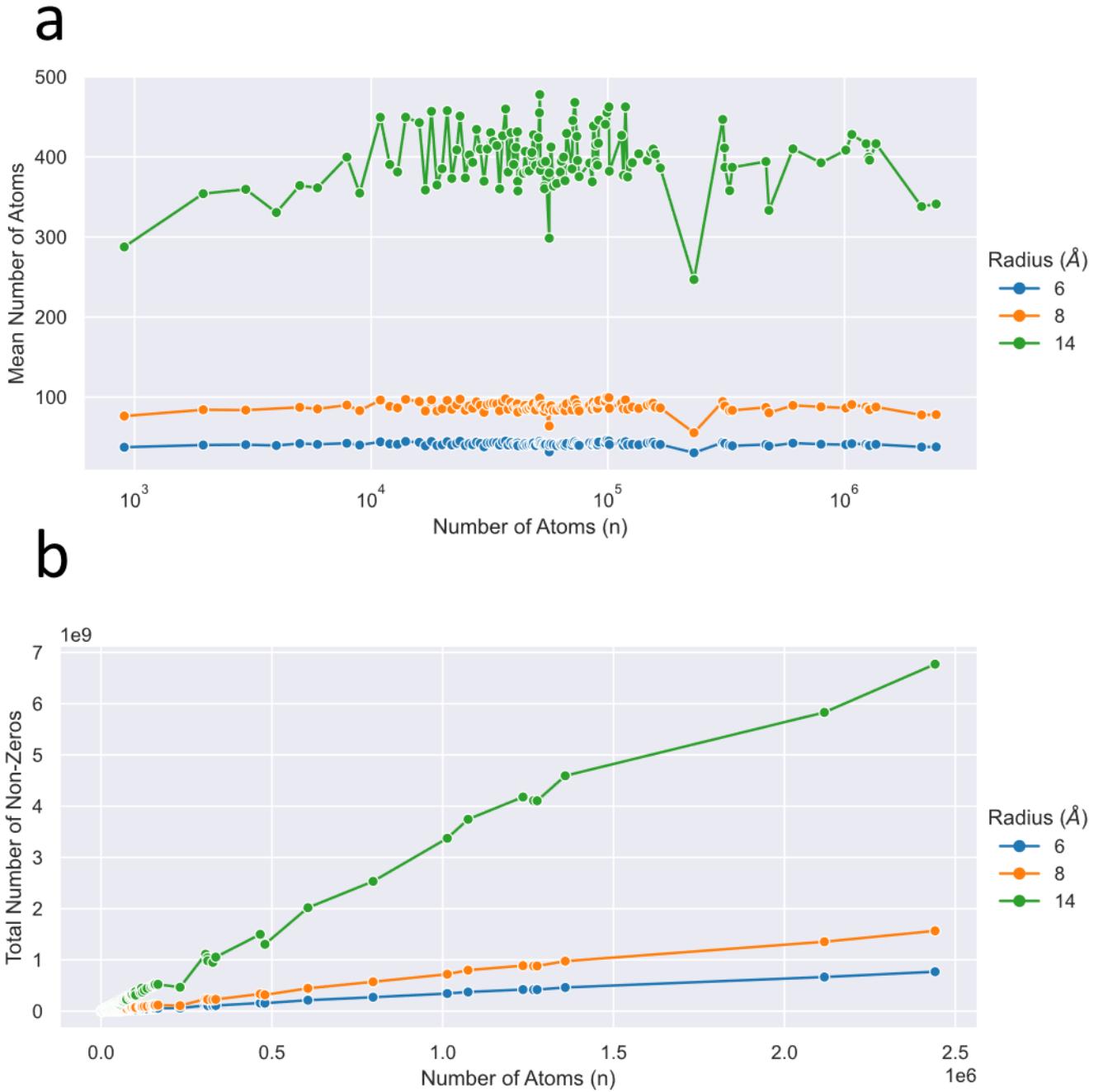
853 **Extended Data Figure 1. Layout of the Hessian Matrix with and without permutation of atom**
854 **ordering by 3DRCM for a case study on the chimeric Sesbania mosaic virus coat protein (PDBID:**
855 **4Y5Z).** The protein structure is a homo-60-mer made of 12 sets of pentamers arranged as an icosahedron
856 with apparent symmetry. (a) Overall spatial distribution of the pentamers in default sequential order.
857 Segment α , β , and γ were colored in green, blue and pink respectively. Segment α consist of chain
858 A,B,C,D,E,F,G,H,I and J. Segment β consist of chain K,L,M,N,O,P,Q,R,S and T. Segment γ consist of
859 chain U, V, W, X, Y, Z, a, c, e and g. The three orthogonal golden rectangles were shown in black with
860 vertices in white sphere. The rest of the macromolecule is shown as a grey surface. A pentamer is
861 displayed as cartoons in segment γ . (b) The Hessian matrix in default ordering (left) and the lower
862 triangle of the Hessian matrix in 3DRCM ordering (right). The Hessian matrix in default ordering has a
863 much longer bandwidth everywhere than the Hessian matrix in 3DRCM ordering; the last few rows and
864 columns in the default ordering refers to water and ions. In the default ordering, it is also clear that
865 segment α do not interact with segment β , but segment γ interacts with both α and β . (c) The first 64
866 eigenfrequency of the macromolecule. Note that the macromolecule is asymmetric due to the inconsistent
867 presence of waters. Inset shows the mode shapes from two apparently degenerate modes. The black
868 arrows are the displacement field of 1000 atoms randomly chosen from those in the top 90% quantile of
869 vibration magnitude in the eigenvector. The arrow in magenta indicates an average direction for local
870 clusters of the displacement field. The vertices belonging to the same golden rectangle is displayed in the
871 same color.



873 **Extended Data Figure 2. Layout of the Hessian Matrix with and without permutation of atom**
874 **ordering by 3DRCM for a case study on the PsbM-deletion mutant of photosystem II (PDBID:**
875 **5H2F).** The macromolecular complex is made of cofactors, lipids, water and protein chains. (a) Overall
876 spatial distribution of the chemicals intercalated in multiple protein chains in default sequential order
877 (green, blue, pink, orange, yellow). Chemicals were shown in black. Only the right half of the structure is
878 colored, the rest of the macromolecule is shown as a grey surface. (b) The Hessian matrix in default
879 ordering (left) and the lower triangle of the Hessian matrix in 3DRCM of the Hessian matrix in 3DRCM
880 ordering (right). The Hessian matrix in default ordering has a much longer bandwidth everywhere than
881 the 3DRCM ordering. The top left square highlighted in the Hessian matrix of the default ordering
882 correspond to that of the right half proteins. Note that the macromolecule is asymmetric due to slight
883 difference in chemicals intercalated. (c) The first 64 eigenfrequencies of the macromolecule. Inset shows
884 the stacked mode shapes from each of the first two modes. The black arrows are the displacement field of
885 1000 atoms randomly chosen from those in the top 90% quantile of vibration magnitude in the
886 eigenvector. The arrow in magenta indicates an average direction for local clusters of the displacement
887 field.

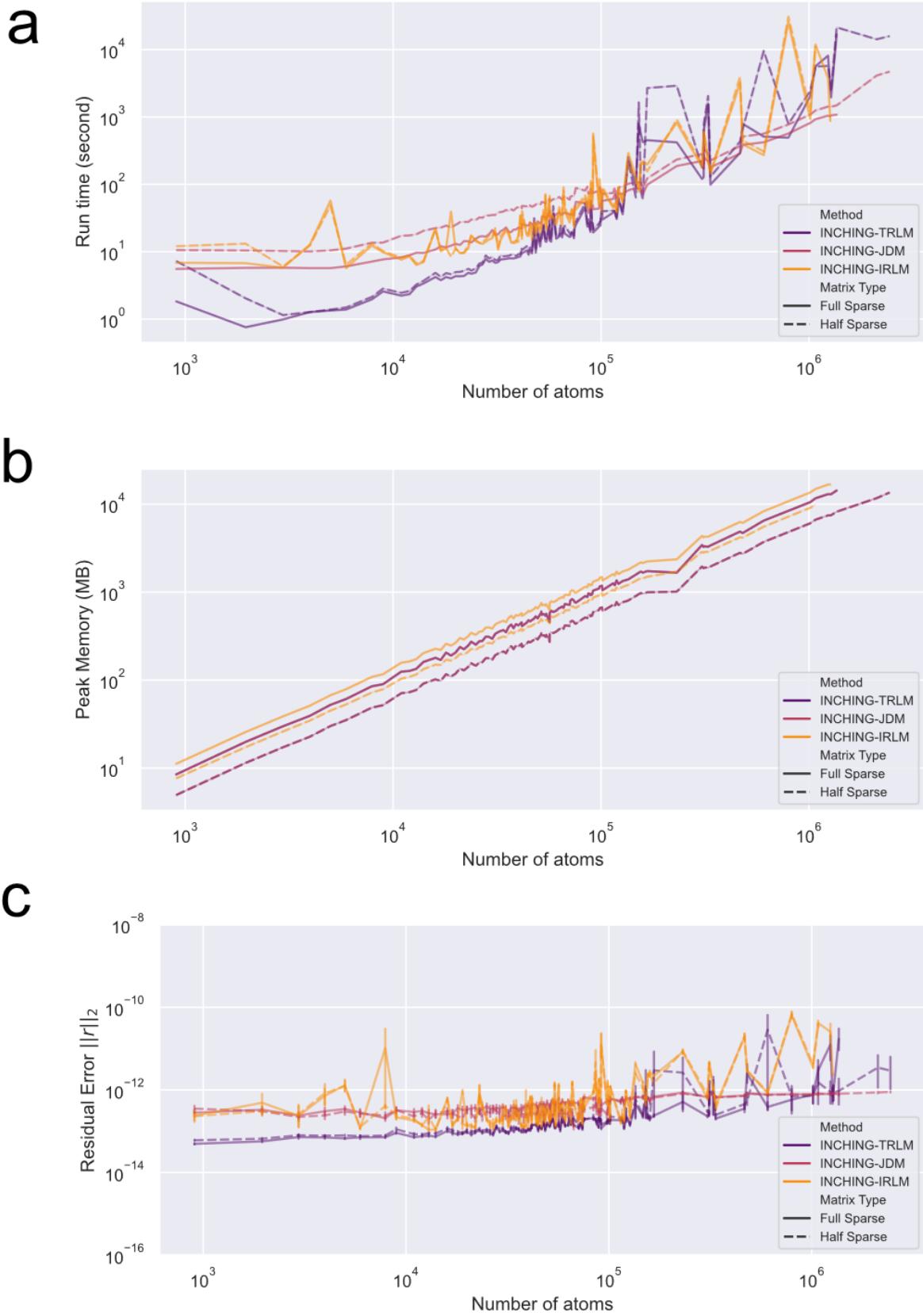


889 **Extended Data Figure 3. Benchmarks on permutation and construction of the Hessian on GPU.**
890 Color code refers to radius $R_C = 6 \text{ \AA}$ (blue), 8 \AA (orange) or 14 \AA (green). (a) Mean bandwidth of a 100-
891 atoms batch of Hessian in default PDB ordering and 3DRCM ordering. To facilitate communication, the
892 mean bandwidth reported here is the mean bandwidth in terms of atom slice (i.e., $N \times N$) of the
893 ($N \times N \times 3 \times 3$) Hessian tensor. The mean bandwidth is presented as the percentage of the total number of
894 atoms N . The Hessian realized with 3DRCM permuted atom ordering is consistently less than 10% of N
895 once there are more than 100 thousand atoms, whereas in the default PDB ordering it can fluctuate
896 between 10% and 90% of N . The shade is the 95% confidence interval. (b) Run time of the algorithms.
897 The reported time is wall-clock time. (c) Peak Memory of the algorithms. Note that for 3DRCM, the
898 memory reported is in random access memory, whereas for the Hessian on GPU, the memory reported is
899 the GPU memory. For $R_C = 14 \text{ \AA}$, 64-bit indexing were used throughout instead of 32-bit indexing to
900 accommodate the larger amount of non-zero entries.



901

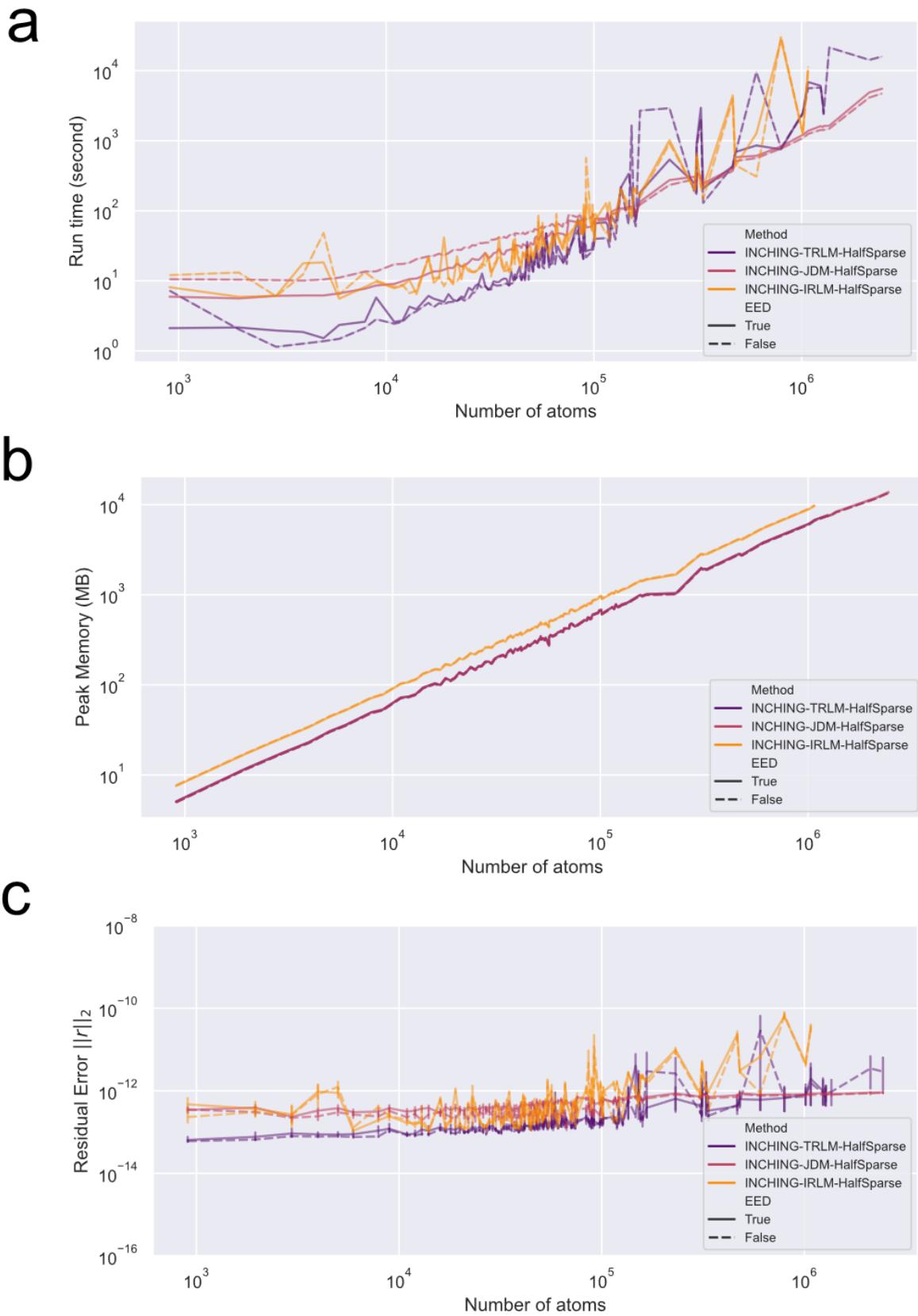
902 **Extended Data Figure 4. Information on the benchmark dataset.** The benchmark dataset has 116
 903 macromolecules with size ranging from around 1000 atoms to around 2.4 million atoms. Color code
 904 refers to radius $R_C = 6 \text{ \AA}$, 8 \AA or 14 \AA . (a) Mean number of atoms within a sphere
 905 of radius R_C . (b) Total number of non-zero entries in the full Hessian matrix.



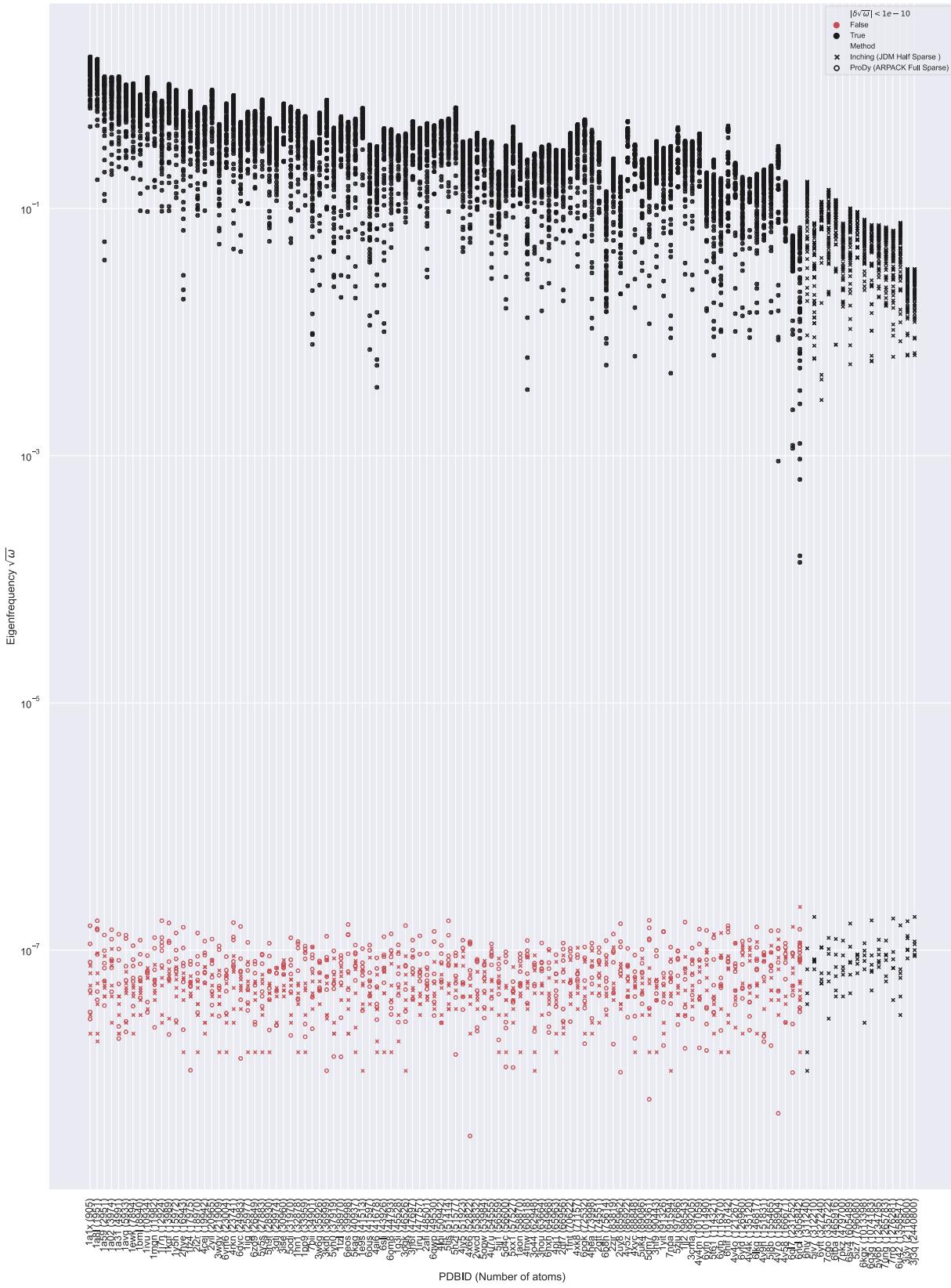
907 **Extended Data Figure 5. Comparison of the effect of accessing and storing the full Hessian matrix**
908 **rather than the lower triangle on throughput, memory consumption and correctness in the**
909 **benchmark dataset.** For all methods, the first 64 eigenpairs were calculated with radius $R_C = 8 \text{ \AA}$.
910 Methods tested includes 3 INCHING methods (i.e. INCHING-TRLM, INCHING-JDM, INCHING-
911 IRLM) with only the lower triangle of the Hessian stored and accessed in dash line and with the full
912 Hessian matrix stored and accessed in solid line. The methods in dash line with only the lower triangle of
913 the Hessian stored were also displayed in Figure 2 of main text. In all methods, the storage format of the
914 Hessian matrix is always “Sparse” meaning a CSR format is stored in double precision. INCHING-
915 TRLM, INCHING-JDM, INCHING-IRLM are our implementation of the Thick Restart Lanczos Method
916 (TRLM), Jacobi-Davidson Method (JDM) and Implicitly Restarted Lanczos Method (IRLM). (a)
917 Benchmark on overall run time including Hessian realization and subsequent diagonalization. The run
918 time is wall-clock time to complete all the calculation. (b) Benchmark on peak memory consumption.
919 Also note that INCHING-TRLM-HalfSparse and INCHING-JDM-HalfSparse share very similar peak
920 memory requirement. (c) Benchmark on residual error. The error bar presented is the 95% confidence
921 interval calculated from the residual error of all the eigenvalues of a macromolecular structure in the
922 benchmark dataset. All programs were stopped if run time exceeds 48 hours. All INCHING programs
923 were stopped if number of restarts exceeded 15000 rounds.

924

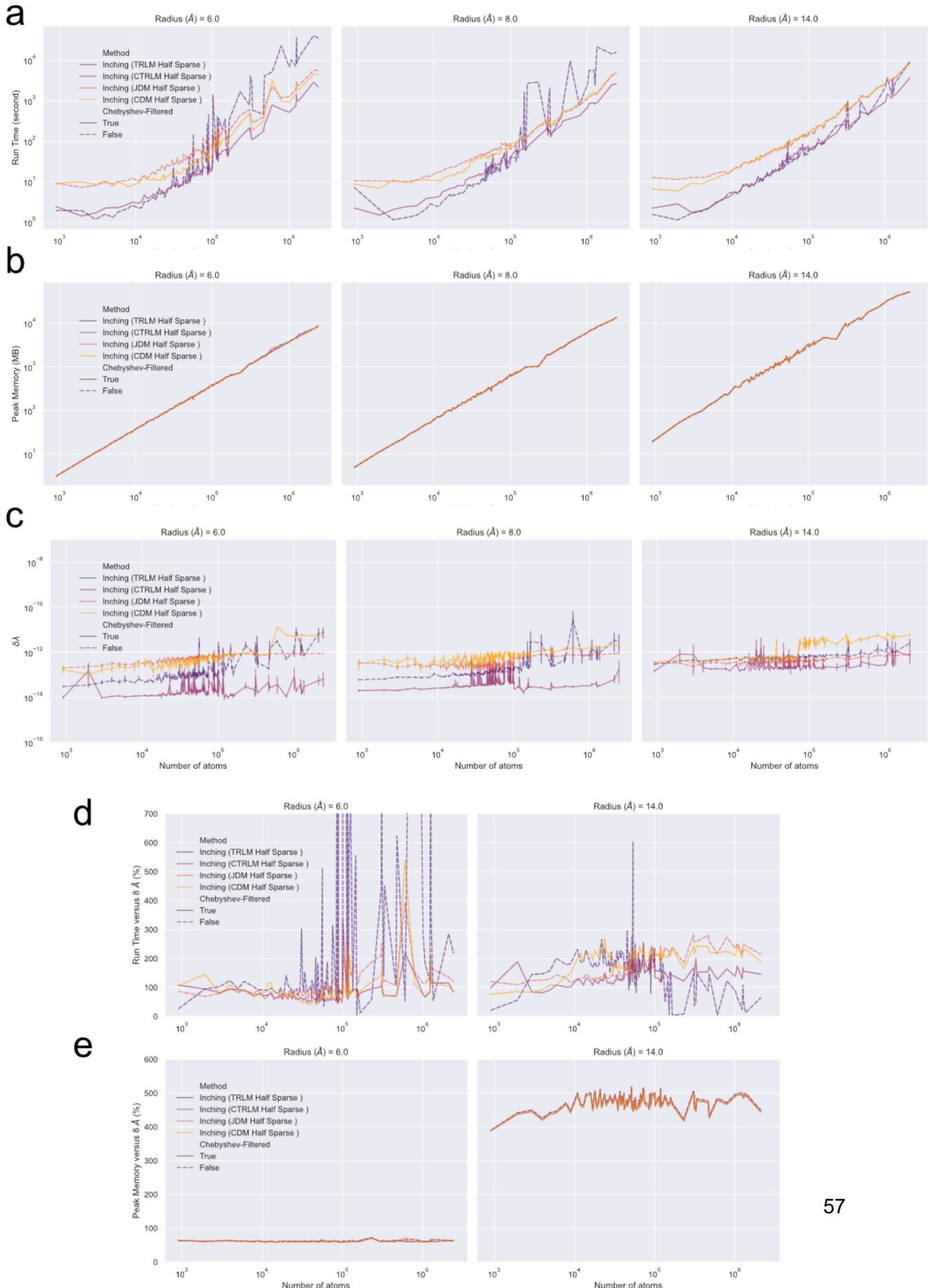
925



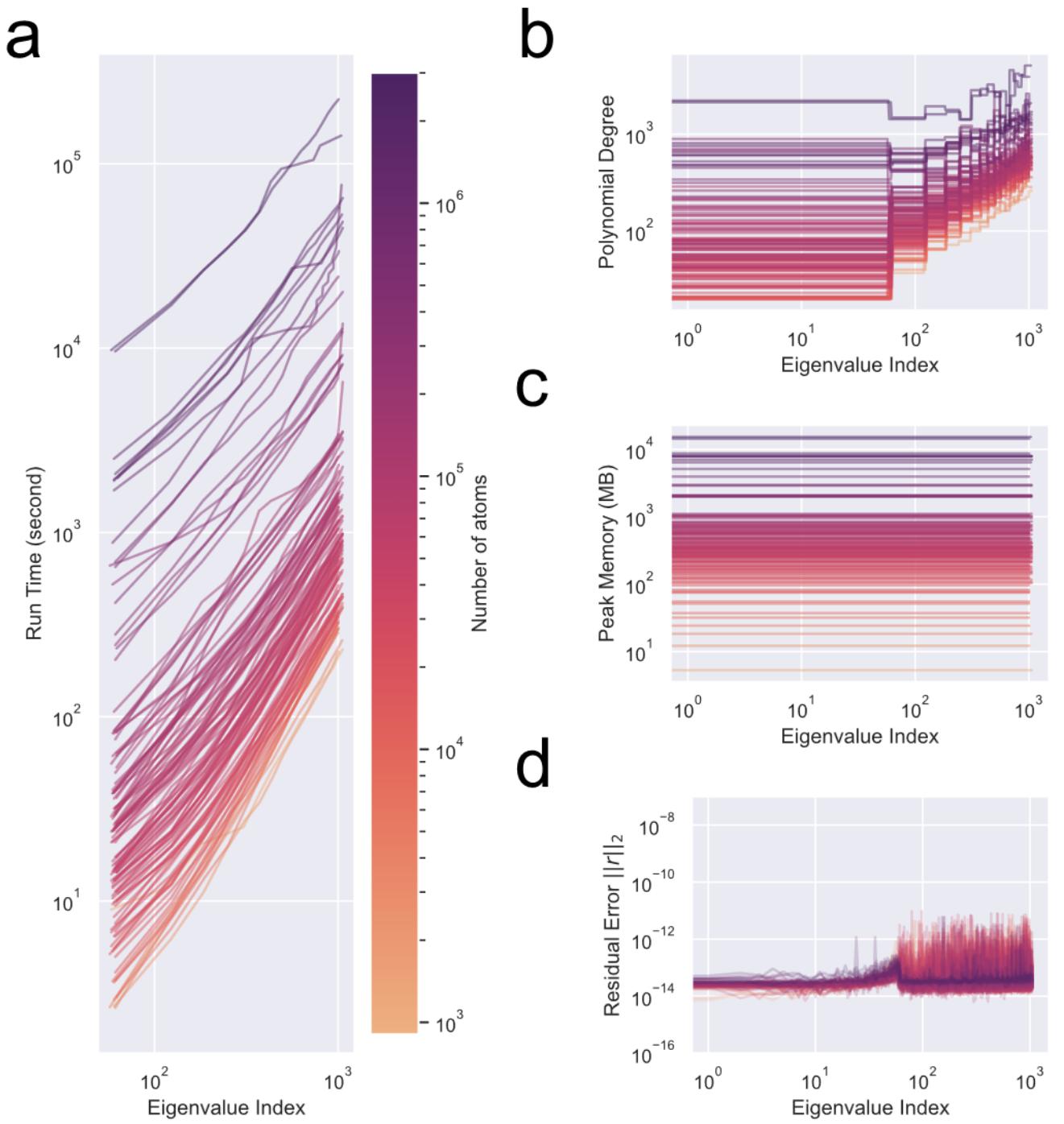
927 **Extended Data Figure 6. Comparison of the effect of explicit external deflation (EED) on**
928 **throughput, memory consumption and correctness in the benchmark dataset.** For all methods, the
929 first 64 eigenpairs were calculated with radius $R_C = 8 \text{ \AA}$. Methods tested includes 3 INCHING methods
930 (i.e. INCHING-TRLM-HalfSparse, INCHING-JDM-HalfSparse, INCHING-IRLM-HalfSparse without
931 EED in dash line and with EED in solid line. The methods in dash line (i.e. INCHING-TRLM-
932 HalfSparse, INCHING-JDM-HalfSparse, INCHING-IRLM-HalfSparse without EED were also displayed
933 in Figure 2 of main text. In all methods, “Half” means only the lower triangle of the Hessian matrix is
934 accessed and stored and the storage format of the Hessian matrix is always “Sparse” meaning a CSR
935 format is stored in double precision. INCHING-TRLM-HalfSparse, INCHING-JDM-HalfSparse,
936 INCHING-IRLM-HalfSparse are our implementation of the Thick Restart Lanczos Method (TRLM),
937 Jacobi-Davidson Method (JDM) and Implicitly Restarted Lanczos Method (IRLM). (a) Benchmark on
938 overall run time including Hessian realization and subsequent diagonalization. The run time is wall-clock
939 time to complete all the calculation. (b) Benchmark on peak memory consumption. Also note that
940 INCHING-TRLM-HalfSparse and INCHING-JDM-HalfSparse share very similar peak memory
941 requirement. (c) Benchmark on residual error. The error bar presented is the 95% confidence interval
942 calculated from the residual error of all the eigenvalues of a macromolecular structure in the benchmark
943 dataset. All programs were stopped if run time exceeds 48 hours. All INCHING programs were stopped if
944 number of restarts exceeded 15000 rounds.



946 **Extended Data Figure 7. Comparing Eigenfrequencies calculated.** Eigenfrequencies computed with
947 INCHING-JDM-HalfSparse in cross marker, compared to those from ProDy with ARPACK backend, i.e.
948 ProDy-ARPACK-FullSparse in circle marker. The ARPACK package is a golden standard
949 implementation of the Implicitly Restarted Lanczos Method (IRLM). Whenever the eigenvalue of the
950 same index calculated by INCHING-JDM-HalfSparse differs from ProDy-ARPACK-FullSparse with
951 absolute value larger than 10^{-10} , the difference is highlighted in red. For cases where ProDy-ARPACK
952 FullSparse did not converge within 48 hours, there is no datapoint. In general, eigenfrequencies less than
953 10^{-6} are considered rigid modes with eigenvalue zero. In all cases considered, there are only 6 rigid
954 modes. For calculation of eigenfrequencies, absolute value was applied to all eigenvalues to avoid
955 spurious imaginary eigenfrequencies for rigid modes with small negative numerical eigenvalue, e.g.,
956 -10^{-14} . For all methods, the first 64 eigenpairs were calculated with radius $R_C = 8 \text{ \AA}$.



958 **Extended Data Figure 8. Scaling in cutoff radii and effect of Chebyshev filtering on throughput,**
959 **memory consumption and correctness in the benchmark dataset.** For all methods, the first 64
960 eigenpairs were calculated. Methods tested includes 4 INCHING methods (i.e. INCHING-TRLM-
961 HalfSparse, INCHING-JDM-HalfSparse, INCHING-CTRLM-HalfSparse and INCHING-CDM-
962 HalfSparse. Methods with Chebyshev low-pass filtering in solid line, otherwise dashline. INCHING-
963 TRLM-HalfSparse, INCHING-JDM-HalfSparse and INCHING-CTRLM-HalfSparse were also displayed
964 in Figure 2 of main text. In all methods, “Half” means only the lower triangle of the Hessian matrix is
965 accessed and stored and the storage format of the Hessian matrix is always “Sparse” meaning a CSR
966 format is stored in double precision. INCHING-TRLM-HalfSparse, INCHING-JDM-HalfSparse,
967 INCHING-CTRLM-HalfSparse and INCHING-CDM-HalfSparse are our implementation of the Thick
968 Restart Lanczos Method (TRLM), Jacobi-Davidson Method (JDM), Chebyshev-filtered Thick Restart
969 Lanczos Method (CTRLM) and Chebyshev-Davidson Method (CDM). Three cutoff radii were considered
970 including $R_C = 6 \text{ \AA}, 8 \text{ \AA}$ and 14 \AA . (a) Benchmark on overall run time including Hessian realization and
971 subsequent diagonalization. The run time is wall-clock time to complete all the calculation. (b)
972 Benchmark on peak memory consumption. (c) Benchmark on residual error. The error bar presented is
973 the 95% confidence interval calculated from the residual error of all the eigenvalues of a macromolecular
974 structure in the benchmark dataset. Comparison of 6 \AA and 14 \AA calculations in the percentage of run
975 time (d) and peak memory (e) versus standard 8 \AA calculations. Note that comparison of scaling in radii
976 in memory is dependent on the number and indexing of non-zero entries in Hessian, where for all 14 \AA
977 calculations, 64-bit indexing were used instead of 32-bit indexing to accommodate the growth in non-zero
978 entries in Hessian beyond capability of 32-bit indexing.



979

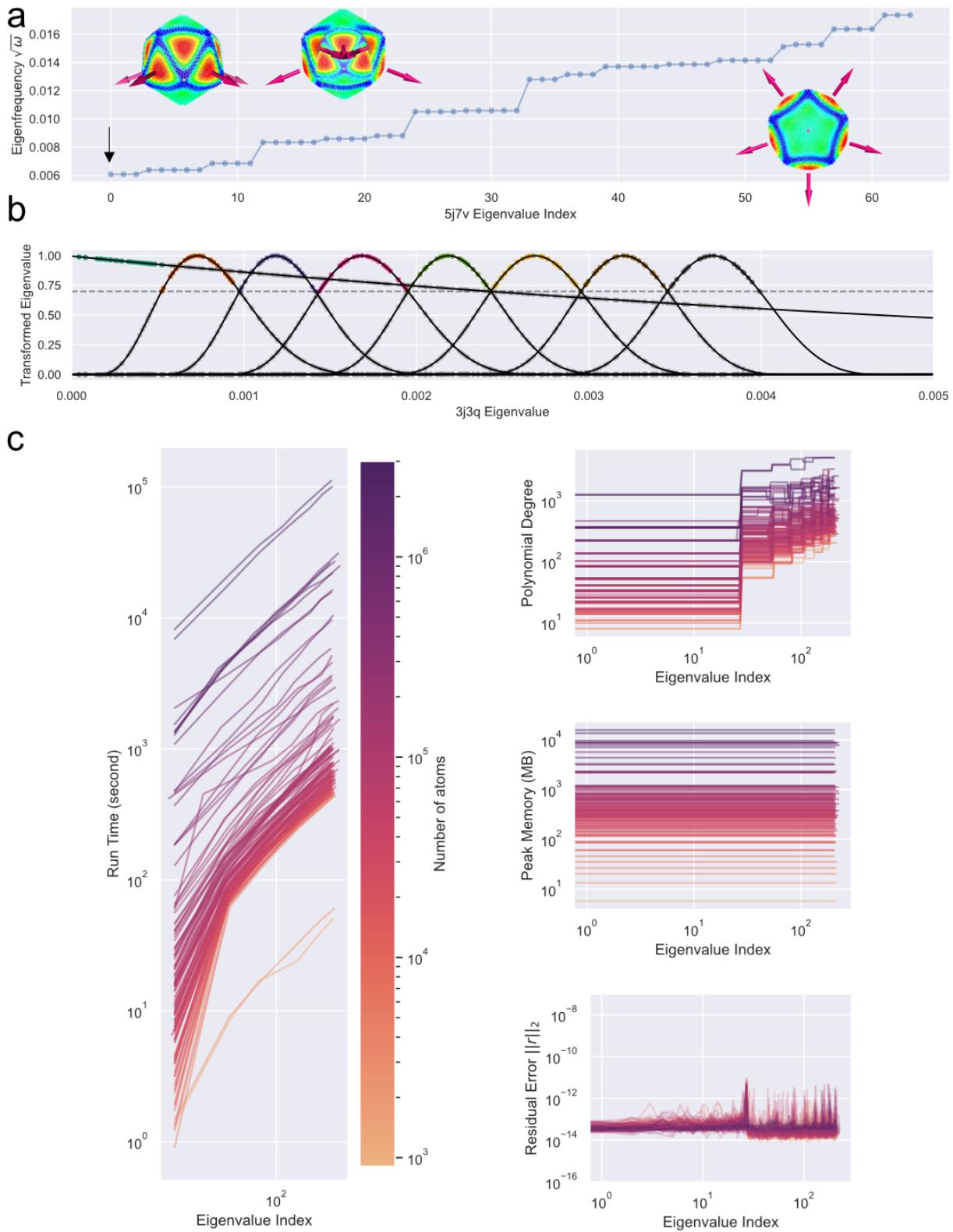
980 **Extended Data Figure 9. Scaling in number of eigenmodes in Chebyshev-filtered Thick Restart**

981 **Lanczos Method.** Up to 1000 or slightly more eigenpairs were calculated using our INCHING-CTRLM-

982 HalfSparse implementation of the Chebyshev-filtered Thick Restart Lanczos Method with radius of

983 interaction $R_C = 8\text{\AA}$. The coloring of the lines, as shown in the color bar, refers to the number of atoms in
984 the system. (a) Run time to complete until the indexed eigenvalue (b) Optimized polynomial degrees
985 involved in calculating the indexed eigenvalue. (c) Peak memory consumed when calculating the
986 spectrum slice containing the indexed eigenvalue. (d) Residual error in each of the eigenvalues.

987



989 **Extended Data Figure 10. Results in applying Chebyshev-filtered Thick Restart Lanczos Method.**

990 (a) The first 64 eigenfrequency of the Faustovirus capsid (PDBID: 5J7V) coarse-grained as 5 million
991 pseudoatoms. Inset shows the mode shapes of the first non-rigid mode. The left- and right-most insets are
992 sideview and bird-eye view respectively; the middle inset is the sideview sectioned to show pattern inside.
993 The arrow in magenta indicates an average direction for local clusters of the displacement field. (b)
994 Mapping of the lowest 224 eigenvalues in batches of 28 eigenvectors when low- and band-pass
995 Chebyshev filters were applied to the HIV capsid system (PDBID: 3J3Q) without coarse-graining. The
996 calculation was done on a GPU RTX4090. Colors on the points indicate eigenvalues resolved in different
997 batches of 64 eigenvalues. (c) Corresponding scaling in number of eigenmodes on GPU RTX4090 in
998 batches of 28 eigenvectors. Up to 200 or slightly more eigenpairs were calculated using INCHING-
999 CTRL-M-HalfSparse with cutoff radius $R_C = 8\text{\AA}$. The coloring of the lines, as shown in the color bar,
1000 refers to the number of atoms in the system. Run time to complete until the indexed eigenvalue, optimized
1001 polynomial degrees involved in calculating the indexed eigenvalue, peak memory consumed when
1002 calculating the spectrum slice containing the indexed eigenvalue and residual error in each of the
1003 eigenvalues were also shown.

1004

1005 **Reference**

- 1006 1. Delarue, M. & Dumas, P. On the use of low-frequency normal modes to enforce collective
1007 movements in refining macromolecular structural models. *Proceedings of the National
1008 Academy of Sciences* **101**, 6957–6962 (2004).
- 1009 2. Tama, F., Miyashita, O. & Brooks III, C. L. Normal mode based flexible fitting of high-
1010 resolution structure into low-resolution experimental data from cryo-EM. *Journal of
1011 Structural Biology* **147**, 315–326 (2004).

- 1012 3. Gur, M., Madura, J. D. & Bahar, I. Global Transitions of Proteins Explored by a Multiscale
1013 Hybrid Methodology: Application to Adenylate Kinase. *Biophysical Journal* **105**, 1643–1652
1014 (2013).
- 1015 4. Franklin, J., Koehl, P., Doniach, S. & Delarue, M. MinActionPath: maximum likelihood
1016 trajectory for large-scale structural transitions in a coarse-grained locally harmonic energy
1017 landscape. *Nucleic Acids Research* **35**, W477–W482 (2007).
- 1018 5. Bakan, A. & Bahar, I. The intrinsic dynamics of enzymes plays a dominant role in
1019 determining the structural changes induced upon inhibitor binding. *Proceedings of the
1020 National Academy of Sciences* **106**, 14349–14354 (2009).
- 1021 6. Shrivastava, I. H. & Bahar, I. Common Mechanism of Pore Opening Shared by Five
1022 Different Potassium Channels. *Biophysical Journal* **90**, 3929–3940 (2006).
- 1023 7. Lezon, T. R. & Bahar, I. Constraints Imposed by the Membrane Selectively Guide the
1024 Alternating Access Dynamics of the Glutamate Transporter GltPh. *Biophysical Journal* **102**,
1025 1331–1340 (2012).
- 1026 8. Bahar, I., Lezon, T. R., Bakan, A. & Shrivastava, I. H. Normal Mode Analysis of
1027 Biomolecular Structures: Functional Mechanisms of Membrane Proteins. *Chem. Rev.* **110**,
1028 1463–1497 (2010).
- 1029 9. Levitt, M. Conformation analysis of proteins. (University of Cambridge, 1972).
- 1030 10. Levitt, M., Sander, C. & Stern, P. S. The normal modes of a protein: Native bovine
1031 pancreatic trypsin inhibitor. *International Journal of Quantum Chemistry* **24**, 181–199
1032 (1983).
- 1033 11. Brooks, B. & Karplus, M. Normal modes for specific motions of macromolecules: application
1034 to the hinge-bending mode of lysozyme. *Proceedings of the National Academy of Sciences*
1035 **82**, 4995–4999 (1985).
- 1036 12. Brooks, B. & Karplus, M. Harmonic dynamics of proteins: normal modes and fluctuations in
1037 bovine pancreatic trypsin inhibitor. *Proc. Natl. Acad. Sci. U.S.A.* **80**, 6571–6575 (1983).

- 1038 13. Ma, J. Usefulness and Limitations of Normal Mode Analysis in Modeling Dynamics of
1039 Biomolecular Complexes. *Structure* **13**, 373–380 (2005).
- 1040 14. Hayward, S. & Groot, B. L. de. Normal Modes and Essential Dynamics. in *Molecular*
1041 *Modeling of Proteins* 89–106 (Humana Press, 2008). doi:10.1007/978-1-59745-177-2_5.
- 1042 15. Tirion, M. M. Large Amplitude Elastic Motions in Proteins from a Single-Parameter, Atomic
1043 Analysis. *Phys. Rev. Lett.* **77**, 1905–1908 (1996).
- 1044 16. Hinsen, K., Thomas, A. & Field, M. J. Analysis of domain motions in large proteins. *Proteins:*
1045 *Structure, Function, and Bioinformatics* **34**, 369–382 (1999).
- 1046 17. Hinsen, K. Analysis of domain motions by approximate normal mode calculations. *Proteins:*
1047 *Structure, Function, and Bioinformatics* **33**, 417–429 (1998).
- 1048 18. Tama, F. & Sanejouand, Y.-H. Conformational change of proteins arising from normal mode
1049 calculations. *Protein Engineering, Design and Selection* **14**, 1–6 (2001).
- 1050 19. Brink, J. *et al.* Experimental Verification of Conformational Variation of Human Fatty Acid
1051 Synthase as Predicted by Normal Mode Analysis. *Structure* **12**, 185–191 (2004).
- 1052 20. Schilbach, S. *et al.* Structures of transcription pre-initiation complex with TFIIH and
1053 Mediator. *Nature* **551**, 204–209 (2017).
- 1054 21. Jin, Q. *et al.* Iterative Elastic 3D-to-2D Alignment Method Using Normal Modes for Studying
1055 Structural Dynamics of Large Macromolecular Complexes. *Structure* **22**, 496–506 (2014).
- 1056 22. Krieger, J. M., Sorzano, C. O. S., Carazo, J. M. & Bahar, I. Protein dynamics developments
1057 for the large scale and cryoEM: case study of ProDy 2.0. *Acta Cryst D* **78**, 399–409 (2022).
- 1058 23. Zhang, Y. *et al.* State-dependent sequential allostery exhibited by chaperonin TRiC/CCT
1059 revealed by network analysis of Cryo-EM maps. *Progress in Biophysics and Molecular*
1060 *Biology* **160**, 104–120 (2021).
- 1061 24. Vuillemot, R., Miyashita, O., Tama, F., Rouiller, I. & Jonic, S. NMMD: Efficient Cryo-EM
1062 Flexible Fitting Based on Simultaneous Normal Mode and Molecular Dynamics atomic
1063 displacements. *Journal of Molecular Biology* **434**, 167483 (2022).

- 1064 25. Atilgan, A. R. *et al.* Anisotropy of Fluctuation Dynamics of Proteins with an Elastic Network
1065 Model. *Biophysical Journal* **80**, 505–515 (2001).
- 1066 26. Tama, F., Gadea, F. X., Marques, O. & Sanejouand, Y.-H. Building-block approach for
1067 determining low-frequency normal modes of macromolecules. *Proteins: Structure, Function,*
1068 *and Bioinformatics* **41**, 1–7 (2000).
- 1069 27. Li, G. & Cui, Q. A Coarse-Grained Normal Mode Approach for Macromolecules: An Efficient
1070 Implementation and Application to Ca²⁺-ATPase. *Biophysical Journal* **83**, 2457–2474
1071 (2002).
- 1072 28. Perahia, D. & Mouawad, L. Computation of low-frequency normal modes in
1073 macromolecules: Improvements to the method of diagonalization in a mixed basis and
1074 application to hemoglobin. *Computers & Chemistry* **19**, 241–246 (1995).
- 1075 29. Marques, O. & Sanejouand, Y.-H. Hinge-bending motion in citrate synthase arising from
1076 normal mode calculations. *Proteins: Structure, Function, and Bioinformatics* **23**, 557–560
1077 (1995).
- 1078 30. Koehl, P. Large Eigenvalue Problems in Coarse-Grained Dynamic Analyses of
1079 Supramolecular Systems. *J. Chem. Theory Comput.* **14**, 3903–3919 (2018).
- 1080 31. Sleijpen, G. L. G. & Van der Vorst, H. A. A Jacobi–Davidson Iteration Method for Linear
1081 Eigenvalue Problems. *SIAM Rev.* **42**, 267–293 (2000).
- 1082 32. Sleijpen, G. L. G., Booten, A. G. L., Fokkema, D. R. & van der Vorst, H. A. Jacobi-davidson
1083 type methods for generalized eigenproblems and polynomial eigenproblems. *Bit Numer
1084 Math* **36**, 595–633 (1996).
- 1085 33. Wall, M. R. & Neuhauser, D. Extraction, through filter-diagonalization, of general quantum
1086 eigenvalues or classical normal mode frequencies from a small number of residues or a
1087 short-time segment of a signal. I. Theory and application to a quantum-dynamics model.
1088 *The Journal of Chemical Physics* **102**, 8011–8022 (1995).

- 1089 34. Li, R., Xi, Y., Vecharynski, E., Yang, C. & Saad, Y. A Thick-Restart Lanczos Algorithm with
1090 Polynomial Filtering for Hermitian Eigenvalue Problems. *SIAM J. Sci. Comput.* **38**, A2512–
1091 A2534 (2016).
- 1092 35. Zhou, Y. & Saad, Y. A Chebyshev–Davidson Algorithm for Large Symmetric Eigenproblems.
1093 *SIAM J. Matrix Anal. Appl.* **29**, 954–971 (2007).
- 1094 36. Parlett, B. N. 5. Deflation. in *The Symmetric Eigenvalue Problem* 87–92 (Society for
1095 Industrial and Applied Mathematics, 1998). doi:10.1137/1.9781611971163.ch5.
- 1096 37. Berman, H. M. *et al.* The Protein Data Bank. *Nucleic Acids Research* **28**, 235–242 (2000).
- 1097 38. Zhao, G. *et al.* Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom
1098 molecular dynamics. *Nature* **497**, 643–646 (2013).
- 1099 39. Townshend, R. J. L. *et al.* Geometric deep learning of RNA structure. *Science* **373**, 1047–
1100 1051 (2021).
- 1101 40. Baek, M. *et al.* Accurate prediction of protein structures and interactions using a three-track
1102 neural network. *Science* **373**, 871–876 (2021).
- 1103 41. Papadimitriou, Ch. H. The NP-Completeness of the bandwidth minimization problem.
1104 *Computing* **16**, 263–270 (1976).
- 1105 42. Cuthill, E. & McKee, J. Reducing the bandwidth of sparse symmetric matrices. in
1106 *Proceedings of the 1969 24th national conference* 157–172 (Association for Computing
1107 Machinery, 1969). doi:10.1145/800195.805928.
- 1108 43. George, A. & Liu, J. W. Computer Solution of Large Sparse Positive Definite Systems (Alan
1109 George and Joseph W. Liu. *SIAM Rev.* **26**, 289–291 (1984).
- 1110 44. Azad, A., Jacquelin, M., Buluç, A. & Ng, E. G. The Reverse Cuthill-McKee Algorithm in
1111 Distributed-Memory. in *2017 IEEE International Parallel and Distributed Processing*
1112 *Symposium (IPDPS)* 22–31 (2017). doi:10.1109/IPDPS.2017.85.
- 1113 45. Mlakar, D., Winter, M., Parger, M. & Steinberger, M. Speculative Parallel Reverse Cuthill-
1114 McKee Reordering on Multi- and Many-core Architectures. in *2021 IEEE International*

- 1115 *Parallel and Distributed Processing Symposium (IPDPS)* 703–713 (2021).
1116 doi:10.1109/IPDPS49936.2021.00080.
- 1117 46. Karantasis, K. I., Lenhardt, A., Nguyen, D., Garzarán, M. J. & Pingali, K. Parallelization of
1118 Reordering Algorithms for Bandwidth and Wavefront Reduction. in *SC '14: Proceedings of*
1119 *the International Conference for High Performance Computing, Networking, Storage and*
1120 *Analysis* 921–932 (2014). doi:10.1109/SC.2014.80.
- 1121 47. Bentley, J. L. Multidimensional binary search trees used for associative searching.
1122 *Commun. ACM* **18**, 509–517 (1975).
- 1123 48. Bakan, A., Meireles, L. M. & Bahar, I. ProDy: Protein Dynamics Inferred from Theory and
1124 Experiments. *Bioinformatics* **27**, 1575–1577 (2011).
- 1125 49. Lehoucq, R. B. & Sorensen, D. C. Deflation Techniques for an Implicitly Restarted Arnoldi
1126 Iteration. *SIAM J. Matrix Anal. Appl.* **17**, 789–821 (1996).
- 1127 50. Lehoucq, R., Sorensen, D. & Yang, C. ARPACK Users' Guide: Solution of Large-Scale
1128 Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. *SIAM* (1998).
- 1129 51. Anderson, E. *et al.* *LAPACK Users' Guide*. (Society for Industrial and Applied Mathematics,
1130 1999). doi:10.1137/1.9780898719604.
- 1131 52. Yu, V. W. *et al.* GPU-acceleration of the ELPA2 distributed eigensolver for dense symmetric
1132 and hermitian eigenproblems. *Computer Physics Communications* **262**, 107808 (2021).
- 1133 53. Yu, V. W. *et al.* ELSI — An open infrastructure for electronic structure solvers. *Computer*
1134 *Physics Communications* **256**, 107459 (2020).
- 1135 54. Wu, X., Davidović, D., Achilles, S. & Di Napoli, E. ChASE: a distributed hybrid CPU-GPU
1136 eigensolver for large-scale hermitian eigenvalue problems. in *Proceedings of the Platform*
1137 *for Advanced Scientific Computing Conference* 1–12 (Association for Computing Machinery,
1138 2022). doi:10.1145/3539781.3539792.

- 1139 55. Sgherzi, F., Parravicini, A. & Santambrogio, M. D. A Mixed Precision, Multi-GPU Design for
1140 Large-scale Top-K Sparse Eigenproblems. in *2022 IEEE International Symposium on*
1141 *Circuits and Systems (ISCAS)* 1259–1263 (2022). doi:10.1109/ISCAS48785.2022.9937893.
- 1142 56. Anzt, H. *et al.* Optimizing Krylov Subspace Solvers on Graphics Processing Units. in *2014*
1143 *IEEE International Parallel & Distributed Processing Symposium Workshops* 941–949
1144 (2014). doi:10.1109/IPDPSW.2014.107.
- 1145 57. Li, R., Xi, Y., Erlandson, L. & Saad, Y. The Eigenvalues Slicing Library (EVSL): Algorithms,
1146 Implementation, and Software. *SIAM J. Sci. Comput.* **41**, C393–C415 (2019).
- 1147 58. Wu, K. & Simon, H. Thick-Restart Lanczos Method for Large Symmetric Eigenvalue
1148 Problems. *SIAM J. Matrix Anal. Appl.* **22**, 602–616 (2000).
- 1149 59. Bell, N. & Garland, M. Efficient Sparse Matrix-Vector Multiplication on CUDA | Research.
1150 *NVIDIA Technical Report NVR-2008-004* https://research.nvidia.com/publication/2008-12_efficient-sparse-matrix-vector-multiplication-cuda (2008).
- 1152 60. Okuta, R., Unno, Y., Nishino, D., Hido, S. & Loomis, C. CuPy: A NumPy-Compatible Library
1153 for NVIDIA GPU Calculations. *Proceedings of Workshop on Machine Learning Systems*
1154 (*LearningSys*) in *The Thirty-first Annual Conference on Neural Information Processing*
1155 *Systems (NIPS)* (2017).
- 1156 61. Nickolls, J., Buck, I., Garland, M. & Skadron, K. Scalable Parallel Programming with CUDA:
1157 Is CUDA the parallel programming model that application developers have been waiting for?
1158 *Queue* **6**, 40–53 (2008).
- 1159 62. cuBLAS. <https://docs.nvidia.com/cuda/cublas/>.
- 1160 63. cuSPARSE. <https://docs.nvidia.com/cuda/cusparse/>.
- 1161 64. Pornillos, O., Ganser-Pornillos, B. K. & Yeager, M. Atomic-level modelling of the HIV capsid.
1162 *Nature* **469**, 424–427 (2011).
- 1163 65. Caspar, D. L. D. & Klug, A. Physical Principles in the Construction of Regular Viruses. *Cold*
1164 *Spring Harb Symp Quant Biol* **27**, 1–24 (1962).

- 1165 66. Mosalaganti, S. *et al.* AI-based structure prediction empowers integrative structural analysis
1166 of human nuclear pores. *Science* **376**, eabm9506 (2022).
- 1167 67. Huang, C.-M., Kucinic, A., Johnson, J. A., Su, H.-J. & Castro, C. E. Integrated computer-
1168 aided engineering and design for DNA assemblies. *Nat. Mater.* **20**, 1264–1271 (2021).
- 1169 68. Klose, T. *et al.* Structure of faustovirus, a large dsDNA virus. *Proceedings of the National
1170 Academy of Sciences* **113**, 6206–6211 (2016).
- 1171 69. Hoffmann, A. & Grudinin, S. NOLB: Nonlinear Rigid Block Normal-Mode Analysis Method. *J.
1172 Chem. Theory Comput.* **13**, 2123–2134 (2017).
- 1173 70. Pandey, M. *et al.* The transformational role of GPU computing and deep learning in drug
1174 discovery. *Nat Mach Intell* **4**, 211–221 (2022).
- 1175 71. Ikegami, T., Sakurai, T. & Nagashima, U. A filter diagonalization for generalized eigenvalue
1176 problems based on the Sakurai–Sugiura projection method. *Journal of Computational and
1177 Applied Mathematics* **233**, 1927–1936 (2010).
- 1178 72. Gulati, A. *et al.* Structural studies on chimeric Sesbania mosaic virus coat protein: Revisiting
1179 SeMV assembly. *Virology* **489**, 34–43 (2016).
- 1180 73. Yang, Z., Bahar, I. & Widom, M. Vibrational Dynamics of Icosahedrally Symmetric
1181 Biomolecular Assemblies Compared with Predictions Based on Continuum Elasticity.
1182 *Biophysical Journal* **96**, 4438–4448 (2009).
- 1183 74. Uto, S. *et al.* Mutual relationships between structural and functional changes in a PsbM-
1184 deletion mutant of photosystem II. *Faraday Discuss.* **198**, 107–120 (2017).
- 1185 75. Lee, D. T. & Wong, C. K. Worst-case analysis for region and partial region searches in
1186 multidimensional binary search trees and balanced quad trees. *Acta Informatica* **9**, 23–29
1187 (1977).
- 1188 76. Hales, T. *et al.* A FORMAL PROOF OF THE KEPLER CONJECTURE. *Forum of
1189 Mathematics, Pi* **5**, e2 (2017).

- 1190 77. Cuppen, J. J. M. A divide and conquer method for the symmetric tridiagonal eigenproblem.
- 1191 *Numer. Math.* **36**, 177–195 (1980).
- 1192 78. Francis, J. G. F. The QR Transformation A Unitary Analogue to the LR Transformation—
- 1193 Part 1. *The Computer Journal* **4**, 265–271 (1961).
- 1194 79. Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear
- 1195 differential and integral operators. *Journal of Research of the National Bureau of Standards*
- 1196 **45**, 255–282 (1950).
- 1197 80. Sorensen, D. C. Implicit Application of Polynomial Filters in a k-Step Arnoldi Method. *SIAM*
- 1198 *J. Matrix Anal. Appl.* **13**, 357–385 (1992).
- 1199 81. Paige, C. C., Parlett, B. N. & van der Vorst, H. A. Approximate solutions and eigenvalue
- 1200 bounds from Krylov subspaces. *Numerical Linear Algebra with Applications* **2**, 115–133
- 1201 (1995).
- 1202 82. Saad, Y. 7. Filtering and Restarting Techniques. in *Numerical Methods for Large Eigenvalue*
- 1203 *Problems* 163–191 (Society for Industrial and Applied Mathematics, 2011).
- 1204 doi:10.1137/1.9781611970739.ch7.
- 1205 83. Weiße, A., Wellein, G., Alvermann, A. & Fehske, H. The kernel polynomial method. *Rev.*
- 1206 *Mod. Phys.* **78**, 275–306 (2006).
- 1207 84. Geus, R. The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue
- 1208 problems with application to the design of accelerator cavities. (ETH Zurich, 2002).
- 1209 doi:10.3929/ethz-a-004469464.
- 1210 85. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357–362 (2020).
- 1211 86. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat*
- 1212 *Methods* **17**, 261–272 (2020).
- 1213 87. Poppleton, E., Mallya, A., Dey, S., Joseph, J. & Šulc, P. Nanobase.org: a repository for DNA
- 1214 and RNA nanostructures. *Nucleic Acids Research* **50**, D246–D252 (2022).

1215 88. Suma, A. *et al.* TacoxDNA: A user-friendly web server for simulations of complex DNA
1216 structures, from single strands to origami. *Journal of Computational Chemistry* **40**, 2586–
1217 2595 (2019).

1218