Journal of Computational Physics 246 (2013) 275–288

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Exploring large macromolecular functional motions on clusters of multicore processors

José R. López-Blanco [a], Ruymán Reyes [b], José I. Aliaga [b,*], Rosa M. Badia [c], Pablo Chacón [a], Enrique S. Quintana-Ortí [b]

[a] Instituto de Química Física Rocasolano, Spanish National Research Council (CSIC), 28006 Madrid, Spain
[b] Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I, 12071 Castellón, Spain
[c] Computer Sciences Department, Barcelona Supercomputing Center (BSC-CNS), 08034 Barcelona, Spain

ARTICLE INFO

Article history:
Received 21 December 2012
Received in revised form 12 March 2013
Accepted 15 March 2013
Available online 27 March 2013

Keywords:
Computational biology
Macromolecular machines
Normal modes
Eigenvalue problems
Krylov-subspace method
Clusters of multicore processors

ABSTRACT

Normal modes in internal coordinates (IC) furnish an excellent way to model functional collective motions of macromolecular machines, but exhibit a high computational cost when applied to large-sized macromolecules. In this paper, we significantly extend the applicability of this approach towards much larger systems by effectively solving the computational bottleneck of these methods, the diagonalization step and associated large-scale eigenproblem, on a small cluster of nodes equipped with multicore technology. Our experiments show the superior performance of iterative Krylov-subspace methods for the solution of the dense generalized eigenproblems arising in these biological applications over more traditional direct solvers implemented on top of state-of-the-art libraries. The presented approach expedites the study of the collective conformational changes of large macromolecules opening a new window for exploring the functional motions of such relevant systems.

1. Introduction

The most important components of living cells are proteins and nucleic acids which are formed by long unbranched chains of aminoacids and nucleotides, respectively. They usually assemble together into large macromolecular machines to support the main biological functions. For example, the ribosomal machinery produces new proteins according to the genetic code; the chaperonin proteins assist the folding process of these new-formed proteins; and tubulin and actin filaments support the cellular shape. At the molecular level, the biological activity of these components is frequently tackled by analyzing their dynamics and interactions. However, the direct experimental observation of the functional motions is quite complex and computational studies are often the only way to predict conformational changes. Molecular Dynamics (MD) simulations provide detailed information on the fluctuations and conformational changes of proteins and nucleic acids using available 3D atomic structures. Unfortunately, the large size of the macromolecules and the long time scale of their motion render MD simulations too costly or even prohibitive.

In recent years, the use of coarse-grained models (CG), i.e. reduced representations of biopolymer structures obtained by grouping atoms into simplified entities or pseudo-atoms, has naturally provided significant computational savings by reducing the number of system variables [32]. In this context, CG merged with normal mode analysis (NMA) has arisen as a powerful and popular approach to simulate collective motions of macromolecular complexes at extended time scales [6]. In

* Corresponding author. Tel.: +34 964728297.
E-mail addresses: jrlopez@iqfr.csic.es (J.R. López-Blanco), rreyes@uji.es (R. Reyes), aliaga@uji.es (J.I. Aliaga), rosa.m.badia@bsc.es (R.M. Badia), pablo@chaconlab.org (P. Chacón), quintana@uji.es (E.S. Quintana-Ortí).

0021-9991/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.03.032

biomolecular simulations, NMA plays an important role to analyze structural fluctuations around a well-defined conformation. In particular, CG-NMA has been successfully applied to calculate biologically relevant motions of proteins and nucleic acids [19] even at very low resolutions [11], X-ray refinements [12], flexible fitting of atomic structures into electron microscopy density maps [18], efficient generation of conformational pathways [15], and ligand docking [10,22]. These results and many others validate the use of CG-NMA modeling in describing molecular flexibility and demonstrate how this approach has become, in practice, a powerful alternative to costly atomistic simulations.

We recently extended the applicability of such approaches to larger molecules by formulating the NMA in internal coordinates (ICs), i.e. using the dihedral angles of the macromolecular chain instead of the Cartesian coordinates (CCs) of the atoms [21]. Although ICs reduce the number of degrees of freedom (DOF) in comparison to CCs, the diagonalization step within this procedure remains a major computational bottleneck for large molecules, even when using CG-reduced approximations. Biologically relevant molecular machines such as ribosome complexes, actin filaments, microtubules, viral capsids and many others either are beyond the applicability range or can only be addressed by very aggressive CG approximations (e.g. a single pseudo-atom per molecule). Thus, additional methodological work must be performed to extend the applicability to both larger macromolecular systems and more detailed representations.

In this paper we address the parallel simulation of the motions of large macromolecular complexes via IC-NMA, leveraging clusters of computers equipped with multicore processors to tackle the diagonalization bottleneck. Specifically, we make the following contributions:

- For this biological relevant application, we illustrate the advantages of the Krylov subspace method for the solution of *dense* generalized eigenproblems over more conventional "direct" solvers based on LAPACK. While the superior performance of Krylov subspace methods was hinted in [2] for general problems of moderate size on multicore processors and hardware accelerators, here we extend the analysis to the large-scale eigenproblems arising in biomolecular simulation using clusters of multicore processors.
- We improve the performance of all the message-passing eigensolvers by combining MPI with the SMPSs [29] runtime to obtain an enhanced version of the Cholesky factorization, a crucial and highly parallel operation for all eigensolvers considered in this work (including the direct solver).
- The drastic reduction of computing time obtained with the proposed parallel diagonalization scheme is demonstrated by simulating the collective motions of a representative set of large macromolecular machines. In particular, our experimental study includes a detailed performance evaluation as well as a survey of the essential functional motions of some key biological systems extracted with IC-NMA.

The rest of the paper is structured as follows. In Section 2 we briefly review the IC-NMA method for the simulation of macromolecules, exposing the underlying eigenvalue problem that currently conforms the major computational bottleneck of this approach. In Section 3 we introduce the Krylov-subspace iterative method for the solution of generalized symmetric definite eigenproblems, together with its explicit and implicit variants, and discuss their arithmetic cost and memory requirements. Next, in Section 4, we describe the implementation of these two variants using existing message-passing linear algebra libraries for clusters of computers, as well as an alternative parallelization of the Cholesky factorization that leverages intranode concurrency using the SMPSs runtime. In Section 5 we evaluate the performance of the different methods, including a conventional "direct" solver from LAPACK, which can be considered as the state-of-the-art for the solution of this type of problems. In that section we also include several 3D animations showing selected motions of large macromolecular machines that were obtained using IC-NMA and the fast parallel methods presented here. Finally, we close the paper offering some concluding remarks and a discussion of future work in Section 6.

2. Simulation of macromolecules via IC-NMA

In NMA the relevant collective motions are described by approximating both potential and kinetic energies as quadratic functions of the atomic positions and velocities, respectively. This simplification allows the decomposition of the motion into a series of deformation vectors which encode the potential displacement directions. Such vectors, or modes, are obtained by diagonalizing the second derivative matrices of both potential (Hessian) and kinetic energies. Every mode exhibits a characteristic frequency that is inversely proportional to the deformation energy cost. High frequency modes represent stiff localized displacements, whereas low frequency modes correspond to soft collective conformational changes. The latter deformations have been closely related to functional motions [19,36], and it is also well-known that there is a good correspondence with the essential motions extracted from atomistic MD [24,27,1].

Following the seminal works of Go and others [16,20], we have implemented a complete mathematical NMA framework using dihedral angles as variables. This new multipurpose tool chest, named iMOD [21], exploits the benefits of classical NMA formulations in internal coordinates (ICs) while extending them to cover multiscale modeling, even for huge macromolecular structures. iMOD has two major advantages over current Cartesian approaches: it implicitly preserves model geometry to minimize potential distortions and substantially reduces the number of DOFs to improve efficiency. The software also includes several atomic CG representations but the ICs are always defined by both the canonical backbone dihedral angles, i.e. ϕ and ψ in proteins (see Fig. 1 for details), and the 6 rotational/translational DOFs between chains.

In the rest of this section, we summarize the methodological aspects of the IC-NMA implemented in iMod. The molecular system is modelled as a set of atoms connected by harmonic springs. The potential energy of the system is given by:

$$V = \sum_{i<j} F_{ij}(r_{ij}^t - r_{ij}^0)^2, \tag{1}$$

where F_{ij} is the spring stiffness matrix, r_{ij} is the distance between atoms i and j, and the superindices t and 0 stand for current and equilibrium conformations, respectively. The potential energy expressed as a function of the n ICs is:

$$V = \frac{1}{2}\mathbf{q}\,\mathbf{H}\mathbf{q}^T, \tag{2}$$

where $\mathbf{q}$ is the displacement vector from the equilibrium conformation ($\mathbf{q} = \mathbf{q}^t - \mathbf{q}^0$),

$$\mathbf{H} = (H_{\alpha,\beta}) = \frac{\partial^2 V}{\partial q_\alpha \partial q_\beta} \tag{3}$$

is the Hessian matrix, and the α and β subindices represent any IC to compute the partial derivatives. In a similar manner, the kinetic energy can be expressed as:

$$T = \frac{1}{2}\dot{\mathbf{q}}\,\mathbf{T}\,\dot{\mathbf{q}}^T, \tag{4}$$

where the dot operator (superscript) indicates time differentiation and the so-called kinetic energy matrix

$$\mathbf{T} = (T_{\alpha,\beta}) = \sum_i m_i \frac{\partial \mathbf{r}_i}{\partial q_\alpha} \cdot \frac{\partial \mathbf{r}_i}{\partial q_\beta}. \tag{5}$$

Here, the mass of the ith atom is m_i and $\mathbf{r}_i$ stands for the corresponding Cartesian position vector. In the limit of a small excursion from the equilibrium, the general solutions of the Lagrange equation of motion ($L = T - V$) are:

$$\mathbf{q}_k^t = \mathbf{q}_k^0 + \sum_{k=1}^{n} a_k \mathbf{x}_k \cos(2\pi \upsilon_k t + \delta_k), \tag{6}$$

where a_k and δ_k depend on the initial conditions, and any k-th mode $\mathbf{x}_k$ and its associated frequency υ_k can be obtained by solving the *generalized eigenvalue problem* [23]:

$$\mathbf{HX} = \Lambda \mathbf{TX}, \tag{7}$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with $\lambda_k = (2\pi\upsilon_k)^2$.

In ICs the direct computation of **H** and **T** matrices using Eqs. (2) and (4) requires $\mathcal{O}(n^4)$ and $\mathcal{O}(n^3)$ floating-point arithmetic operations (flops), respectively. Instead, iMod takes advantage of the recursion relationships described in Braun et al. [8] to reduce the cost of computing these matrices to $\mathcal{O}(n^2)$. This speed-up transforms the diagonalization (i.e., the solution of (7) for Λ and $\mathbf{X}$) into the main computational bottleneck of the NMA. Furthermore, a complete eigenspace computation is not required since functional motions are only encoded in the lowest frequency part.

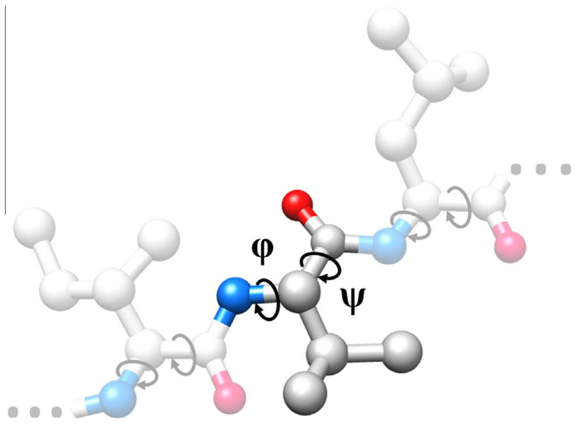


Fig. 1. Dihedral angles. A single aminoacid of the protein chain is highlighted showing the two canonical backbone dihedral angles, ϕ and ψ, considered as variables. Carbon, oxygen and nitrogen atoms are plotted using gray, red and blue colors, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3. Solving generalized symmetric definite eigenproblems

The previous section illustrates that the key numerical problem for the analysis of functional motions of macromolecular complexes via CG-NMA is the solution of a generalized eigenproblem of the form

$$AX = \Lambda BX, \tag{8}$$

where $A, B \in \mathbb{R}^{n\times n}$ correspond, respectively, to the Hessian and kinetic matrices, $\Lambda \in \mathbb{R}^{s\times s}$ is a diagonal matrix with the s sought-after eigenvalues of the matrix pair (A, B), and $X \in \mathbb{R}^{n\times s}$ contains the corresponding unknown eigenvectors (or modes); see (7). In ICs, matrices A, B are both dense and symmetric positive definite (SPD). Furthermore, only the smallest eigenpairs are required ($s \approx 100$) when dealing with macromolecules ($n > 10{,}000$).

The Krylov-subspace method is a competitive approach for the solution of dense and sparse eigenvalue problems when only a few eigenvalues are required [2,28]. The method starts by (implicitly or explicitly) transforming the generalized problem (8) into a standard one in a first stage, to then obtain the sought-after eigenpairs from this simpler form. Specifically, consider the Cholesky factorization of the SPD matrix B given by

$$B = U^T U, \tag{9}$$

where $U \in \mathbb{R}^{n\times n}$ is upper triangular [17]. The generalized eigenproblem in (8) can then be reformulated into its *standard* counterpart

$$CY = \Lambda Y \quad \equiv \quad (U^{-T}AU^{-1})(UX) = \Lambda(UX), \tag{10}$$

where $C \in \mathbb{R}^{n\times n}$ is symmetric (actually, SPD) and $Y \in \mathbb{R}^{n\times s}$. Thus, the standard eigenproblem (10) shares its eigenvalues with those of the original system in (8), while the generalized eigenvectors of the latter can be recovered by solving the upper triangular linear system

$$X := U^{-1}Y. \tag{11}$$

The initial factorization in (9) requires $n^3/3$ flops while recovering the original generalized eigenvectors from (11) adds $n^2 s$ flops. In practice, the triangular factor U overwrites the corresponding entries in the upper triangular part of B and both X and Y use the same matrix, so that these operations do not increase the demand of storage. In case $C := U^{-T}AU^{-1}$ in (10) has to be explicitly built, by exploiting the symmetry of the result, the cost of this operation can be reduced to n^3 flops. However, our experiments reveal a much higher performance if C is obtained by simply solving two triangular linear systems, at a cost of $2n^3$ flops, which in practice compensates the increased number of arithmetic operations. Therefore, in the following discussion and experiments we will assume that, when necessary, C is built by solving two triangular linear systems.

In the subsequent stage, the Krylov-subspace method operates with C or the pair U/A, iteratively approximating the largest eigenvalues of the transformed system. Alternatively, if the smallest eigenvalues are requested, like in a macromolecular NMA, the largest eigenvalues of the inverse problem can be directly approximated. Two different variants of the Krylov subspace method are employed in our work for the solution of the transformed eigenproblem (10):

- **Explicit construction of** C. This variant, hereafter referred to as KE, explicitly builds matrix C. Each step k of the Krylov subspace method then performs a (symmetric) matrix–vector product of the form $z_{k+1} := Cw_k$, with $z_{k=1,2,\ldots}, w_{k=0,1,\ldots} \in \mathbb{R}^n$ and w_0 an initial guess, at a cost of $2n^2$ flops per product.
- **Implicit use of** C. Alternatively, we can keep both A and U so that, at each step of the method, the calculation $z_{k+1} := U^{-T}AU^{-1}w_k$ is performed as a triangular system solve, followed by a matrix–vector product and, finally, a second triangular system solve; i.e., $z_{k+1} := U^{-T}(A(U^{-1}w_k))$. In this variant, referred to as KI, there is no initial cost to pay for the explicit construction of C, but the cost per step for the computation of z_{k+1} is doubled with respect to the previous case, from $2n^2$ to $4n^2$ flops.

In both variants, obtaining w_{k+1} from z_{k+1} requires a few operations of linear cost in n and, in case reorthogonalization is needed during the present step of the Lanczos iteration (a condition that is data-dependent), the QR factorization of a small $s \times s$ matrix ($\frac{4}{3}s^3$ flops). Upon convergence, the Krylov subspace iteration produces a tridiagonal matrix S of dimension $m \times m, m \geqslant 2s$, whose eigenvalues approximate those of C, and a matrix Q of dimension $n \times m$ with the Krylov vectors. Given that $m \ll n$, obtaining the eigenvalues and eigenvectors from S and Q adds a minor theoretical cost to the computations. No appreciable additional storage space is required by any of the previous variants.

In summary, when dealing with dense coefficient matrices, the Krylov subspace method exhibits a computational cost of $\mathcal{O}(n^3)$ flops, basically due to the transformation of the generalized problem into the standard form. Therefore, the solution of large-scale dense eigenproblems appearing in IC-NMA clearly calls for the application of high performance computing techniques on parallel architectures.

4. Libraries and Parallelization of the Eigensolvers

In this section we review the libraries used for the implementation of the generalized eigensolvers. The main components are ScaLAPACK [7], a parallel message-passing library for dense linear algebra computations; the communication application programming interface provided by MPI [30]; and PARPACK [3], a package of parallel subroutines designed to solve large-scale eigenvalue problems based on the Krylov method. The remaining functionality is provided by PBLAS, LAPACK, BLAS, and BLACS; see Fig. 2.

4.1. ScaLAPACK

ScaLAPACK provides a collection of parallel message-passing routines for the solution of a number of dense linear algebra problems, concretely, linear systems of equations, linear-least squares problems, eigenvalue problems, and numerical rank computations, among others. ScaLAPACK builds upon two basic libraries: BLACS is a collection of message-passing routines that use matrices as the basis for communication, which is more natural for linear algebra computations. PBLAS, a parallel version of the BLAS built on top of this library, provides parallel message-passing routines for basic linear algebra operations such as products of matrices and the solution of triangular systems.

ScaLAPACK and PBLAS both employ a 2-D block cyclic layout for the data: matrices are partitioned into square blocks of size $b \times b$ (the data distribution blocking factor), which are then distributed following a block-cyclic 2D layout among a $p \times q$ (logical) grid of processes. The distribution layout determines the communication pattern of the message-passing algorithm, which transfers data via calls to BLACS (usually built on top of MPI). Standard kernels from LAPACK/BLAS, two well-known libraries for dense linear algebra operations, are employed for local vector and matrix operations.

4.2. PARPACK

PARPACK is an extension to ARPACK (Arnoldi Package) that can be used to compute a few eigenvalues/eigenvectors of a symmetric matrix M via an algorithmic variant of the Lanczos process (implicitly restarted Lanczos). The package employs a reverse communication interface so that no reference is done to M from within it. Instead, the user has to provide the result for certain matrix–vector products and, when necessary, triangular linear system solves involving M in both cases.

Listings 1 and 2 provide simplified versions, encoded in C, of the parallel message-passing KE and KI eigensolvers. Inserted comments illustrate the structure of the two variants, the functionality of the invoked ScaLAPACK and PARPACK routines,

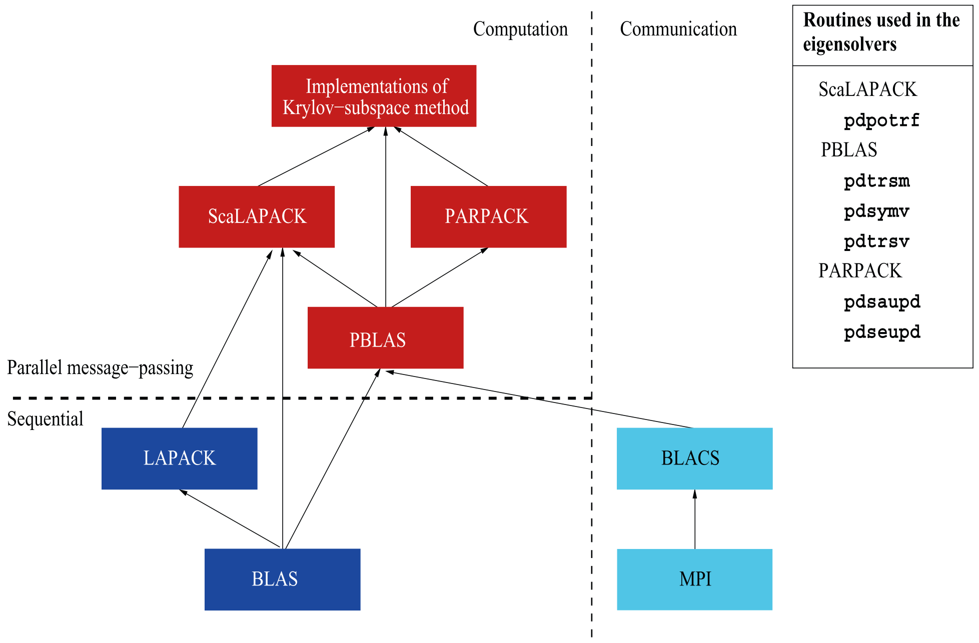


Fig. 2. Libraries (and routines, in the panel to the right) employed in the construction of the parallel message-passing implementations of the Krylov-subspace eigensolvers.

and the reverse communication mechanism. In particular, note how the two PARPACK routines, `pdsaupd` and `pdseupd`, do not access the matrix entries (indeed they do not receive the pointer to the matrix `A` as part of their arguments). Instead, it is the user who is responsible for calculating certain operations involving the matrix via calls to `pdsymv` (for both KE and KI) and `pdtrsv` (only for KI).

Listing 1. Implementation of variant KE using ScaLAPACK + PARPACK (simplified).

```
// KE: Variant with explicit construction of C
// Inputs: Matrix pair (A,B), both of dimension n x n; number of requested eigenvalues s
// Outputs: eigenvalues in vector d, of lenght n, and eigenvectors in X of dimension n x s
  int ido = 0, iter = 0, info = 0;
  double dtol = 0.0, done = 1.0, dzero = 0.0;
  double *w = NULL, *z = NULL;
  // 1. Transform into the associated standard eigenvalue
  // 1.1 B = U^T U; overwrite upper triangular part of B with the Cholesky factor U
  pdpotrf ("Upper", &n, B, ...);
  // 1.2 C := U^-T A U^-1; overwrite A with the solution. Upper triangular factor in B
  pdtrsm ("Right", "Upper", "No transpose", "No-unit", &n, &n, &done, B, ..., A, ... );
  pdtrsm ("Left", "Upper", "Transpose", "No-unit", &n, &n, &done, B, ..., A, ... );
  // 2. Krylov subpace iteration for the variant
  // 2.0 Generate w_0; initial guess returned in WORKD
  pdsaupd (..., &ido, ..., &n, ..., &s, &dtol, ..., &m, Q, ..., IPNTR, WORKD, ..., &info);
  while (ido!= 99) {
    iter++;
    // 2.1 Form z_k+1 := C w_k; C is in A
    w = WORKD + IPNTR[0] - 1;
    z = WORKD + IPNTR[1] - 1;
    pdsymv ("Upper", &n, &done, A, ..., w, ..., &dzero, z, ... );
    // 2.2 z_k+1 -> w_k+1; w and z in WORKD
    pdsaupd (..., &ido, ..., &n, ..., &s, &dtol, ..., &m, Q, ..., WORKD, ..., &info);}
    // 2.3 S,Q -> Lamba,Y; S in WORKD, eigenvalues in d, Ritz vectors in X
    pdseupd (..., &s, ..., d, X, ..., &s, &dtol, ..., &m, Q, ..., IPNTR, WORKD, ..., &info);
// 3. Back-transform
// 3.1 X := U^-1 Y. Overwrite Ritz vectors in X with eigenvectors of the problem
  pdtrsm ("Left", "Upper", "No transpose", "No-unit", &n, &s, &done, B, ..., X, ... );
```

For the small-scale experiments reported in the next section, when the data matrices fit into a single node of the target platform, we have also implemented more efficient eigensolvers that replace the calls to ScaLAPACK/PBLAS/PARPACK by analogous routines from LAPACK/BLAS/ARPACK. These implementations thus avoid the overhead intrinsic to message-passing communication, but their structure, functionality and usage of reverse communication are very similar to those illustrated in the listings.

Table 1
Benchmark of large-scale biological macromolecules. *In this case all heavy atoms were taken into account, in the rest a C_α model was used instead [21]. †Modelled with FilaSitus [35]. ‡Kindly provided and MD optimized by M. Deriu [14].

Acronym	Name	PDB id.	DOFs
GroEL/ES	GroEL/GroES	1sx4	15,870
ER	Eucariotic Ribosome*	2xsy,2xtg	36,488
CCMV	Cowpea chlorotic mottle virus	1cwp	56,454
HBV	Hepatitis B virus	1qgt	66,234
VP	Vault protein	2zuo,2zv4,2zv5	119,486
F-actin	Actin filament	1esv†	141,297
MT	Microtubule 13:3	1tub‡	148,111 (908,954)
HK97	Hong Kong 97 virus Head II	2ft1	149,231 (227,154)
NwV	Nudaurelia capensis omega virus	1ohf	149,506 (264,654)

Listing 2. Implementation of variant KI using ScaLAPACK + PARPACK (simplified).

```
// KI: Variant with implicit use of C
// Inputs: Matrix pair (A,B), both of dimension n x n; number of requested eigenvalues s
// Outputs: eigenvalues in vector d, of lenght n, and eigenvectors in X of dimension n x s
  int ido = 0, iter = 0, info = 0;
  double dtol = 0.0, done = 1.0, dzero = 0.0;
  double *w = NULL, *z = NULL;
  // 1. Transform into the associated standard eigenvalue
  // 1.1 B = U^T U, overwrite upper triangular part of B with the Cholesky factor U
  pdpotrf ("Upper", &n, B, ...);
   // 2. Krylov subpace iteration for the variant
  // 2.0 Generate w_0; initial guess returned in WORKD
  pdsaupd (..., &ido, ..., &n, ..., &s, &dtol, ..., &m, Q, ..., IPNTR, WORKD, ..., &info);
  while (ido != 99) {
    iter++;
    // 2.1 Form z_k+1 := U^-T A U^-1 w_k; C is in A
    w = WORKD + IPNTR[0] - 1;
    z = WORKD + IPNTR[1] - 1;
    pdtrsv ("Upper", "No transpose", "No-unit", &n, B, ..., w, ...);
    pdsymv ("Upper", &n, &done, A, ..., w, ..., &dzero, z, ...);
    pdtrsv ("Upper", "Transpose", "No-unit", &n, B, ..., z, ...);
    // 2.2 z_k+1 -> w_k+1; w and z in WORKD
    pdsaupd (..., &ido, ..., &n, ..., &s, &dtol, ..., &m, Q, ..., WORKD, ..., &info);
}
  // 2.3 S,Q -> Lamba,Y; S in WORKD, eigenvalues in d, Ritz vectors in X
  pdseupd (..., &s, ..., d, X, ..., &s, &dtol, ..., &m, Q, ..., IPNTR, WORKD, ..., &info);
  // 3. Back-transform
  //3.1 X := U^-1 Y. Overwrite Ritz vectors in Y with eigenvectors of the problem
  pdtrsm ("Left", "Upper", "No transpose", "No-unit", &s, &n, &done, B, ..., X, ...);
```

4.3. Optimizing ScaLAPACK Cholesky factorization via SMPSs

ScaLAPACK was designed in the 90s when the majority of distributed-memory supercomputers had a single processor per node. Two straight-forward conventional approaches to adapt the kernels in this library to the current multicore nodes is to place one MPI process per core, or to place a single MPI process per node and exploit the parallelism via a multithreaded implementation of BLAS. Given a message-passing routine from ScaLAPACK, both options provide a transparent parallelization.

As an alternative, in this paper we leverage a message-passing implementation of the Cholesky factorization from ScaLAPACK parallelized via SMPSs [25], a portable and flexible framework to exploit task-level parallelism on shared-memory multiprocessors (including multicore processors). SMPSs is composed of i) a few OpenMP-like pragmas, which allow the programmer to annotate certain functions in the code as tasks; ii) a source-to-source compiler; and iii) a runtime that detects dependencies among tasks and efficiently schedules them for execution in due order, attaining dynamic load balancing.

When applied to parallelize dense linear algebra operations on current multicore processors, SMPSs has demonstrated superior performance over the traditional approach based on LAPACK + multithreaded BLAS [4], much like other tools specifically designed for the dense linear algebra domain [9,26]. For the execution of message-passing dense linear algebra operations on clusters, SMPSs has also shown to yield higher performance to the conventional approaches on a related operation like the symmetric rank-k update [5].

5. Experimental Results

The experiments in this subsection were obtained using IEEE double-precision arithmetic on a 16-node cluster, each node equipped with two Intel Xeon E5520 (hexacore) processors at 2.27 GHz and 24 Gbytes of RAM. The (theoretical) peak performance is 108.96 GFLOPS (1 GFLOPS = 10^9 FLOPS) for a single node (12 cores) and 1,743.40 GFLOPS for the complete platform (192 cores/16 nodes). The memory of the system allowed us to tackle the simulation of macromolecules leading to eigenvalue problems with $n < 42,000$ on a single node, and $n < 150,000$ using the aggregated memory of all nodes.

The interconnect is an InfiniBand QDR fabric, with Mellanox ConnectX-2 adapters linked to a single Mellanox MTS3600 switch. The libraries employed were OpenMPI (GNU-4.1.2.20080704), Mellanox OFED 1.5.3 (InfiniBand drivers and administrative tools), GotoBLAS2 1.13, LAPACK 3.4.0, ScaLAPACK 2.0.0, ARPACK/PARPACK 2.1, and SMPSs 3.5. Unless otherwise stated, in all the test cases we computed the smallest $s = 100$ eigenpairs and set $m = 250$ for the Krylov subspace method underlying PARPACK.

5.1. Benchmarks

The diagonalization strategies have been validated with a set of representative large macromolecules. This benchmark includes filaments, compartmental or chaperonin macromolecules, and several viral capsids; see Table 1. The number of variables of the examples ranges from $n \sim$16,000 to $\sim$150,000. In the largest test cases (MT, HK97 and NwV), a reduction of the total number of DOFs (inside parenthesis in the corresponding column of the table) was required to conform the aggregated memory of the target cluster. To this end, we randomly removed around 85% of the MT dihedral angle variables, and 35% and 45% in the two largest viral cases (HK97 and NwV) as described in [21].

5.2. Small-scale experiments

When dealing with *dense* eigenproblems, the conventional approach is to employ a "direct" solver from LAPACK. For the generalized symmetric definite case, this implies initially transforming the generalized eigenproblem to a standard case (see (10)), followed by a two-stage procedure that first reduces the coefficient matrix C to tridiagonal form via orthogonal transforms, and then obtains the eigenpairs of the tridiagonal matrix via an iterative method such as the QR or MR^3 algorithms [17]. The computation is completed by recovering the eigenvectors of the generalized eigenproblem from those of the transformed case (as in (11)). The costs of the initial transform and final recovery equal those discussed in Section 3, while the two-stage procedure adds $4n^3/3 + 2n^2s$ flops. As we will illustrate next, even for dense eigenvalue problems, this approach is rarely competitive with the Krylov subspace method when only $s \ll n$ eigenvalues are requested; see [2] for a detailed study on multicore processors and GPUs.

In this initial experiment we evaluate three different solvers: LAPACK direct solver `dsyevr` (MR^3 as tridiagonal solver), and variants KE and KI of the Krylov subspace method. The experiments are carried out on 12 cores of a single node of the cluster, for a variety of dimensions (n) of benchmark CCMV (from 1,000 to the largest problem size that fit into one single node, about 40,000). As we employ only one node, these solvers are implemented using LAPACK/ARPACK/BLAS libraries, exploit multithreaded parallelism, and avoid the overhead associated with the use of kernels from the message-passing counterparts of these libraries.

Fig. 3 reports the execution time of the three solvers normalized with respect to those of KI, linked to the multithreaded implementations of LAPACK/BLAS provided by GotoBLAS2. Compared with KI, the figure clearly shows the rapid increase of the execution time incurred by the LAPACK routine as the problem dimension grows, which is due to the fixed additional cost of cubic order in n incurred by the explicit construction of C and its reduction to tridiagonal form performed in the LAPACK solver. The results also show that the advantage of variant KI over variant KE grows with the problem size. This is also caused by the higher fixed cost of cubic order in n for the explicit construction of C required in variant KE. Experiments with other benchmarks, less than 12 cores, and Intel MKL (10.3) for BLAS and LAPACK showed close results, leading to similar

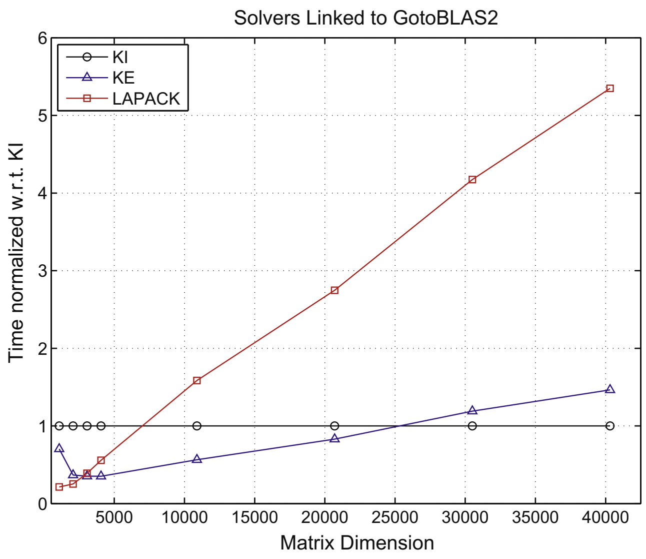


Fig. 3. Performance of the eigensolvers on small-scale instances of benchmark CCMV using 12 cores/single node.

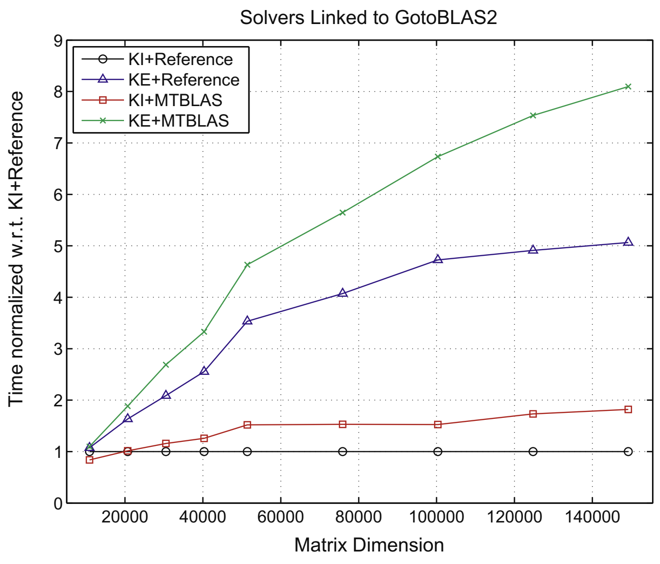
conclusions from this initial study. Therefore, in the following experiments with large-scale macromolecules we will only consider the iterative variants of the Krylov subspace method, discarding the LAPACK approach.

5.3. Large-scale experiments

We next compare the performance of two message-passing implementations of the KE and KI variants of the Krylov subspace method:

- `Reference`. This implementation employs a pure MPI approach with intra-node concurrency exploited by placing one MPI process per core (i.e., 12 MPI processes per node).
- `MTBLAS`. This is a hybrid parallel approach with one MPI process per node and parallelism extracted at the node level with a multithreaded implementation of BLAS.

A best effort was done to utilize the optimal grid configuration, block sizes, etc. for each implementation, variant, and problem size.



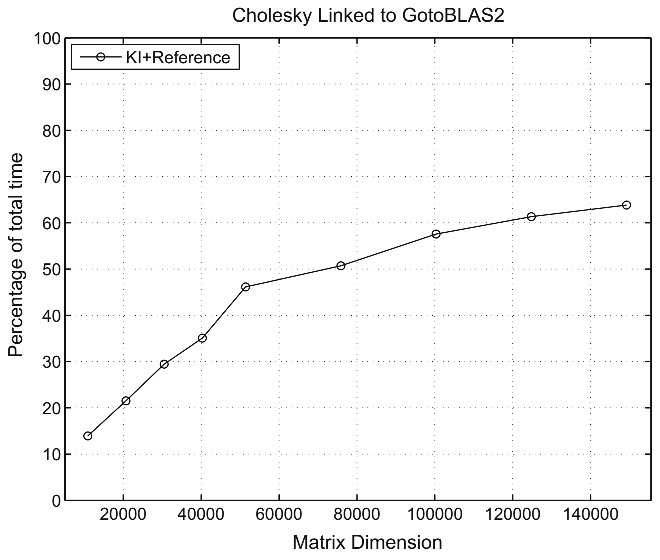


Fig. 4. Performance of the eigensolvers on large-scale instances of benchmarks CCMV (n <50,000) and HK97 ($n \geqslant$50,000) using all 192 cores/16 nodes (top), and percentage of time spent in the Cholesky factorization for the KI + `Reference` eigensolver (bottom).

Table 2
Execution time (in secs.) of the different stages in the KI + `Reference` eigensolver with on the $n = 100{,}325$ instance of benchmark HK97 using all 192 cores/16 nodes.

Operation (routine)	GotoBLAS2
$B = U^T U$ (`pdpotrf`)	274.29
$\bar{z}_{k+1} := U^{-1} w_k$ (`pdtrsv`)	54.96
$\bar{\bar{z}}_{k+1} := C\bar{z}_{k+1}$ (`pdsymv`)	68.35
$z_{k+1} := U^{-T}\bar{\bar{z}}_{k+1}$ (`pdtrsv`)	72.15
$z_{k+1} \rightarrow w_{k+1}$ (`pdsaupd`)	0.43
$H, V \rightarrow \Lambda, Y$ (`pdseupd`)	0.08
$X := U^{-1} Y$ (`pdtrsm`)	3.96
Total	476.38

Table 3
Execution time (in secs.) of the parallel routines for the Cholesky factorization of a SPD matrix using all 192 cores/16 nodes.

Matrix dimension	`Reference`	`MTBLAS`	`MPI/SMPSs`
10,560	1.52	1.60	1.30
50,000	43.83	169.22	32.17
75,000	123.69	199.24	98.24
100,000	266.47	360.60	250.87
125,000	487.74	650.14	492.41

The first experiment with large-scale dimension problems, in Fig. 4 (top), reports the performance of these four implementations (combinations of KE/KI with `Reference`/`MTBLAS`) on all 16 nodes and 192 cores of the cluster using benchmarks CCMV ($n < 50{,}000$) and HK97 ($n \geqslant 50{,}000$). In this case, all results are normalized with respect to the `Reference` implementation of variant KI. The trend in the two plots is coherent with the results reported in the previous section: as n grows, the implicit variant outperforms the explicit one by a larger factor. The second conclusion that can be extracted from this experiment is that, in general, it is better to rely on a parallelization that employs one MPI process per core, especially as the problem size grows. In the next experiment, we therefore concentrate on this combination: variant KI parallelized with the `Reference` approach.

Let's examine now how the execution time of variant KI + `Reference` is distributed for one of the instances of benchmark HK97, say $n = 100{,}325$, in the previous experiment. Table 2 reports the execution time of the stages that compose this solver (see the explanation in Section 3 and Listing 2). In the results for the two invocations of `pdtrsv`, and the single ones to `pdsymv` and `pdsaupd`, we accumulate the total execution time corresponding to all the iterations necessary for convergence of the Krylov subspace method. This experiment reveals that the first four routines substantially dominate the cost of the eigensolvers (more than 98% of the total time), with the Cholesky factorization being the most expensive one. In particular, the factorization occupies around 57% of the execution time with this fraction growing with n up to 64% for the largest problem size, and clearly becoming a performance bottleneck for the solution of large-scale problems; see Fig. 4 (bottom). This justifies the usage of an alternative parallelization of the factorization routine, based on the combination of MPI with SMPSs, that yields higher performance for certain ranges of the problem dimension.

Table 3 shows the impact of this new parallelization of the Cholesky factorization, comparing it with the conventional `Reference` and `MTBLAS` approaches, for SPD matrices of dimension up to 125,000. These results demonstrate the competitive performance of the `MPI/SMPSs` alternative, concretely when the ratio between the problem size and the number of cores is moderate. The acceleration achieved within this range is very appealing and relevant from the application point of view, since in many situations, the IC-NMA is applied multiple times to the same macromolecule in order to simulate a motion trajectory. This calls for the utilization of a large number of cores for the solution of each eigenproblem, even for cases of small to moderate size.

5.4. Application

In this section, we briefly describe and comment the results obtained by applying the proposed parallel diagonalization methods to the benchmark macromolecules (see Table 1). In all cases, the eigensolver KI + `MPI/SMPSs` yields very satisfactory results in terms of both accuracy and efficiency. For the first time, to our knowledge, NMA of several systems up to 150,000 DOFs are presented. Moreover, as reported in Table 4, the use of all 192 cores/16 nodes of the cluster provided enough aggregated memory and computational resources to solve such large eigenproblems in few minutes. For example, less than 9 min were needed to solve a problem with almost 100,000 DOFs, and only ~20 min to handle a system ~50% bigger. By contrast, the diagonalization routine used in the original iMOD (Intel MKL `dspgvx`) took several days to solve a problem with 100,000 DOFs on an Intel Xeon E5–2650. This considerable speed up currently allows us to rapidly predict the motions of

Table 4
Execution time (in secs.) to solve the eigenproblems associated with the large-scale macromolecules using all 192 cores/16 nodes.

Acronym	DOFs	Time
GroEL/ES	15,870	16.21
ER	36,488	57.65
CCMV	56,454	124.64
HBV	66,234	172.55
VP	119,486	723.39
F-actin	141,297	1,027.81
MT	148,111	1,267.07
HK97	149,231	1,306.95
NwV	149,506	1,315.61

macromolecular systems of size and structural detail never explored before. The results are in full agreement with considerably more modest resolution results found in the literature. In the following, we briefly comment the relevant motions characterized by our IC-NMA approach.

Ribosomes are the nano-factories for translating the genetic code into proteins. This protein and nucleic acid macromolecule adopts many conformational states to support its function. For the illustration of its dynamical behavior, an animation of the ratcheted-like motion and some more local rearrangements are included (see Online Resource 1). This well-known motion has been validated experimentally, and even by NMA [11].

Although newly-synthesized proteins naturally fold into their active 3D conformation, some require further assistance. Chaperonins provide the suitable environment and mechanical forces to facilitate folding. GroEL/ES chaperonin is formed by two identical rings with a small cap that encloses the folding chamber. The binding of an unfolded substrate triggers complex dynamical events, such as substrate internalization, entrance blocking, folding and release. NMA is able to describe many of these highly collective motions (see for instance [34]). For example, we have animated some modes that could be involved in substrate internalization and folding (see Online Resource 2).

Cellular shape is mainly maintained by large filaments of proteins. Among them, microtubules and actin filaments (F-actin) also play relevant roles in dynamical processes such as cell division, mobility, intracellular transport or muscle contraction. The modal motions inherent to the filament-like macromolecular structure include bending, stretching and torsion displacements. The relevance of such motions has been already highlighted in [13,14]. In Fig. 5, the first three bending modes for the F-actin are shown. Online Resources 3 and 4 respectively include some of these representative modes for microtubules and F-actin.

The viral life cycle involves several stages including infection, genetic material replication, viral precursor assembly, and maturation into the infective particle. These processes generally involve large concerted rearrangements of coating proteins and nucleic acids. The success of NMA in the exploration of viral collective conformational changes has been widely proven. However, these studies were performed either taking advantage of the icosahedral symmetry of the system [33] or considering each protein chain as a single rigid block [31]. By contrast, our NMA formulation in ICs can work directly at atomic level representation, even without considering explicitly the system symmetry. For example, the collective swelling motions related to maturation of CCMV, HK97 and NwV viruses described in [31] have been characterized with our approaches. In Fig. 6 some representative modal displacements of the CCMV are shown, including those corresponding to the swelling motion (32nd mode). Some of these motions are animated in Online Resources 5 and 6 for CCMV and HBV viruses, respectively. Nevertheless, several non-viral proteins can form virus-like shells which may also exhibit dynamical behavior. For example, in

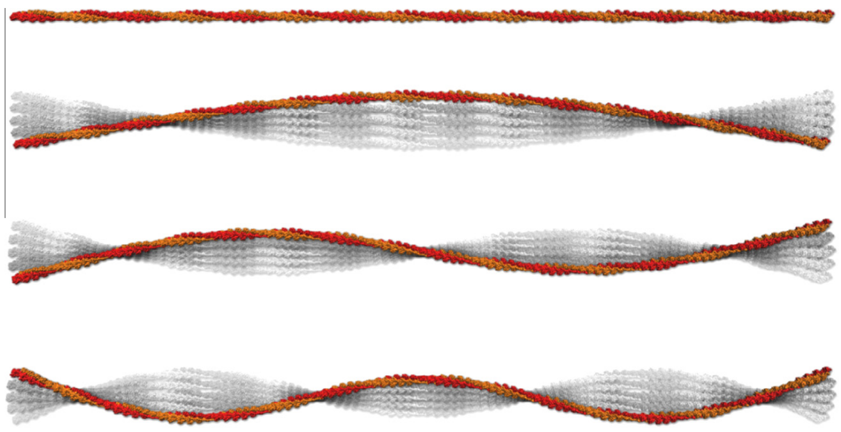


Fig. 5. Illustrative actin filament bending motions. The first three bending modes (1, 3 and 5) computed from the straight experimental structure are shown. See animations in supplementary online resources.

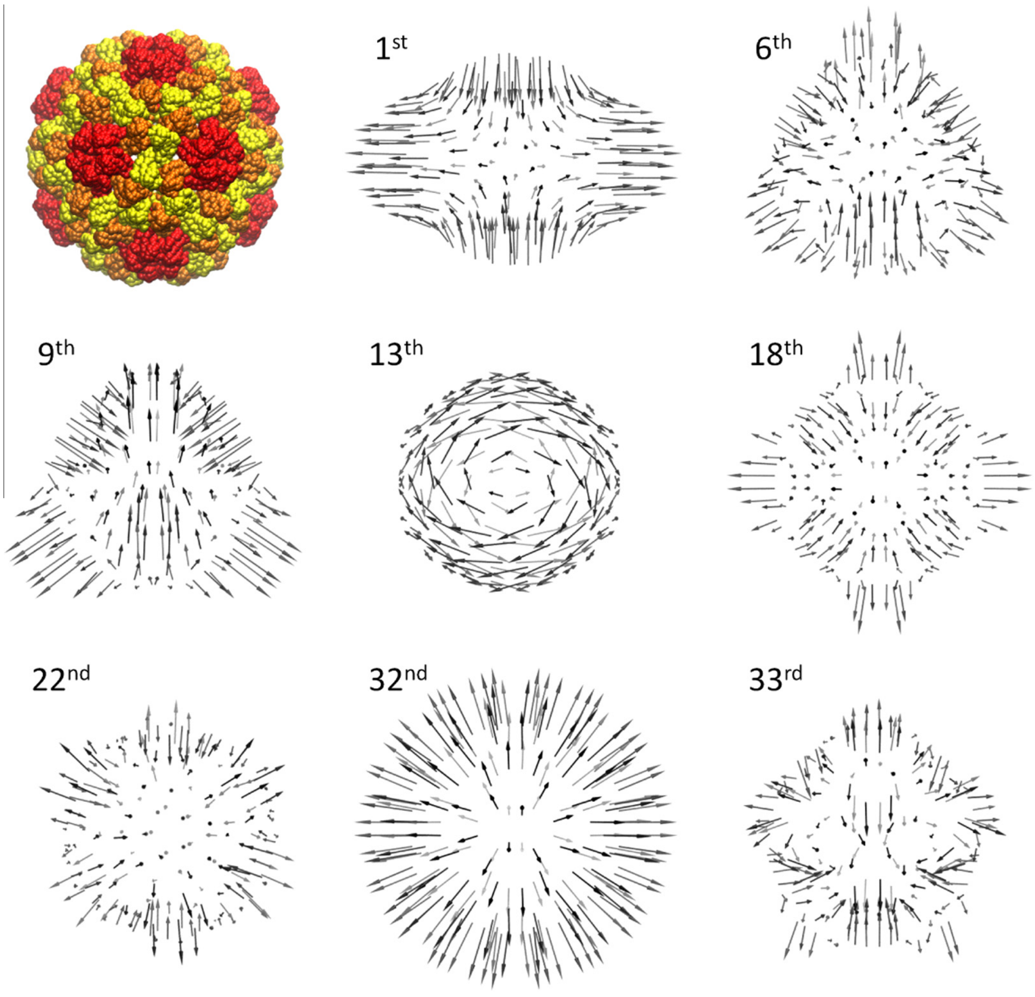


Fig. 6. Illustrative modal displacements of the CCMV virus. The NMA was performed from the experimental atomic structure (colored surface). Each arrow shows the computed displacement of a single viral chain for the indicated mode.

Online Resource 7 it is shown how the compartmental structure acquired by the major vault protein would swing and stretch around its equilibrium conformation.

In summary, by effectively solving large-scale generalized eigenvalue problems on distributed-memory clusters, we greatly extend the applicability of IC-NMA towards much larger systems. The main contribution is a practical demonstration of successful and quick predictions of collective motions of large macromolecules that were very hard to obtain so far. Furthermore, the use of the normal modes in simulation applications such as Langevin dynamics, Montecarlo simulations, conformational transitions or structural refinement is typically based on an iterative process in which the vibrational analysis is repeated multiple times. For example, in the simulation of the structural transition between two known conformations (`iMorph` tool in [21]), the NMA is applied more than 100 times to calculate a feasible smooth trajectory. In this context, the drastic reduction of time is even more relevant. Thus, our new parallel approach provides a fast method to employ in a recursive way the NMA of large macromolecular systems.

6. Concluding remarks

In this paper we have evaluated the application of the Krylov method to model the functional motions of large-scale macromolecular machines. By extending the applicability of IC combined with NMA, we thus expedite the study of the collective conformational changes of relevant biological systems opening a new window for capturing their functional motions at reduced cost.

Our experimental results using a moderate-scale cluster, with 16 nodes and equipped with state-of-the-art multicore technology, show the superior performance of the implicit variant of the iterative Krylov method over the explicit one as well as a direct eigensolver, all of them implemented on top of linear algebra libraries. Thus, the new parallel methods yield the sought-after models for the large-scale NMA of large macromolecules, with almost 150,000 degrees of freedom, in slightly over 20 min, while using a sequential algorithm for the same purpose would have required a platform with over 384 GBytes of memory, and almost 3 days of computing time, assuming similar algorithm performance.

Future work will be focused on the integration of our parallel IC-NMA approach into more advanced simulation applications. The computational savings achieved here will allow, for the first time, to carry out such simulations with large size macromolecules. We think this new avenue to explore the collective motions of the main actors of the biological processes will be crucial for understanding their biological functions.

7. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jcp.2013.03.032.

Acknowledgements

This research was supported by project EU INFRA-2010–1.2.2 "TEXT: Towards EXaflop applicaTions"; project CICYT TIN2011–23283 and FEDER; and Grants CAM-S2010/BMD-2353, HFSP-RGP0039/2008 and MICNN BFU2009–09552.

References

[1] A. Ahmed, S. Villinger, H. Gohlke, Large-scale comparison of protein essential dynamics from molecular dynamics simulations and coarse-grained normal mode analyses, Proteins 78 (16) (2010) 3341. 52.
[2] J. Aliaga, P. Bientinesi, D. Davidović, E. Di Napoli, F.D. Igual, E.S. Quintana-Ortí, Solving dense generalized eigenproblems on multi-threaded architectures, Applied Mathematics and Computation 218 (22) (2012) 11279–11289.
[3] ARPACK project home page. http://www.caam.rice.edu/software/ARPACK/
[4] R.M. Badia, Jose R. Herrero, J. Labarta, J.M. Pérez, E.S. Quintana-Ortí, G. Quintana-Ortí, Parallelizing dense and banded linear algebra libraries using SMPSs, Concurrency and Computation: Practice and Experience 21 (2009) 2438–2456.
[5] R.M. Badia, J. Labarta, V. Marjanovic, A.F. Martín, R. Mayo, E.S. Quintana-Ortí, R. Reyes, Symmetric rank-k update on clusters of multicore processors with SMPSs, in: Proceedings of the International Conference on Parallel Computing – ParCo2011, in press.
[6] I. Bahar, A.J. Rader, Coarse-grained normal mode analysis in structural biology, Current Opinion in Structural Biology 15 (5) (2005) 586. 92.
[7] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, ScaLAPACK Users' Guide, SIAM, 1997.
[8] W. Braun, S. Yoshioki, N. Go, Formulation of static and dynamic conformational energy analysis of biopolymer systems consisting of two or more molecules, Journal of the Physical Society of Japan 53 (9) (1984) 3269–3275.
[9] A. Buttari, J. Langou, J. Kurzak, J. Dongarra, A class of parallel tiled linear algebra algorithms for multicore architectures, Parallel Computing 35 (1) (2009) 38–53.
[10] C.N. Cavasotto, J.A. Kovacs, R.A. Abagyan, Representing receptor flexibility in ligand docking through relevant normal modes, Journal of the American Chemical Society 127 (26) (2005) 9632–9640.
[11] P. Chacon, F. Tama, W. Wriggers, Mega-dalton biomolecular motion captured from electron microscopy reconstructions, Journal of Molecular Biology 326 (2) (2003) 485–492.
[12] M. Delarue, P. Dumas, On the use of low-frequency normal modes to enforce collective movements in refining macromolecular structural models, Proceedings of the National Academy of Sciences USA 101 (18) (2004) 6957–6962.
[13] Marco A. Deriu, Tamara C. Bidone, Francesco Mastrangelo, Giacomo Di Benedetto, Monica Soncini, Franco M. Montevecchi, Umberto Morbiducci, Biomechanics of actin filaments: a computational multi-level study, Journal of Biomechanics 44 (4) (2011) 630–636.
[14] Marco A. Deriu, Monica Soncini, Mario Orsi, Mishal Patel, Jonathan W. Essex, Franco M. Montevecchi, Alberto Redaelli, Anisotropic elastic network modeling of entire microtubules, Biophysical Journal 99 (7) (2010) 2190–2199.
[15] J. Franklin, P. Koehl, S. Doniach, M. Delarue, Minactionpath: maximum likelihood trajectory for large-scale structural transitions in a coarse-grained locally harmonic energy landscape, Nucleic Acids Research 35 (Web Server issue) (2007).
[16] N. Go, T. Noguti, T. Nishikawa, Dynamics of a small globular protein in terms of low-frequency vibrational modes, Proceedings of the National Academy of Sciences USA 80 (12) (1983) 3696–3700.
[17] Gene H. Golub, Charles F. Van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, 1996.
[18] K. Hinsen, E. Beaumont, B. Fournier, J.J. Lacapere, From electron microscopy maps to atomic structures using normal mode-based fitting, Methods in Molecular Biology 654 (2010) 237. 58.
[19] W.G. Krebs, V. Alexandrov, C.A. Wilson, N. Echols, H. Yu, M. Gerstein, Normal mode analysis of macromolecular motions in a database framework: developing mode concentration as a useful classifying statistic, Proteins 48 (4) (2002) 682–695.
[20] M. Levitt, C. Sander, P.S. Stern, Protein normal-mode dynamics: trypsin inhibitor, crambin, ribonuclease and lysozyme, Journal of Moleclar Biology 181 (3) (1985) 423–447.
[21] J.R. Lopez-Blanco, J.I. Garzon, P. Chacon, Imod: multipurpose normal mode analysis in internal coordinates, Bioinformatics 27 (20) (2011) 2843–2850.
[22] A. May, M. Zacharias, Energy minimization in low-frequency normal modes to efficiently allow for global flexibility during systematic protein-protein docking, Proteins 70 (3) (2008) 794–809.
[23] T. Noguti, N. Go, Dynamics of native globular proteins in terms of dihedral angles, Journal of Physical Society of Japan 52 (9) (1983) 3283–3288.
[24] L. Orellana, M. Rueda, C. Ferrer-Costa, J.R. Lopez-Blanco, P. Chacón, M. Orozco, Approaching elastic network models to molecular dynamics flexibility, Joural of Chemical Theory and Computation 6 (9) (2010) 2910–2923.
[25] J.M. Perez, R.M. Badia, J. Labarta, A dependency-aware task-based programming environment for multi-core architectures, in: 2008 IEEE International Conference on Cluster Computing, 2008, pp. 142–151.
[26] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Robert A. van de Geijn, Field G. Van Zee, Ernie Chan, Programming matrix algorithms-by-blocks for thread-level parallelism, ACM Transactions on Mathematical Software 36 (3) (2009) 14:1–14:26.
[27] M. Rueda, P. Chacon, M. Orozco, Thorough validation of protein normal mode analysis: a comparative study with essential dynamics, Structure 15 (5) (2007) 565. 75.
[28] Y. Saad, Numerical Methods for Large Eigenvalue Problems, revised ed., SIAM Press, 2011.
[29] SMP superscalar project home page. http://www.bsc.es/plantillaG.php?catid=385
[30] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra, The Complete Reference. The MIT Press, MPI, 1996.

[31] F. Tama, C.L. Brooks, Diversity and identity of mechanical properties of icosahedral viral capsids studied with elastic network normal mode analysis, Journal of Molecular Biology 345 (2) (2005) 299–314.
[32] V. Tozzini, Minimalist models for proteins: a comparative analysis, Quarterly Reviev of Biophysics 43 (3) (2010) 333–371.
[33] Herman W.T. van Vlijmen, Martin Karplus, Normal mode calculations of icosahedral viruses with full dihedral flexibility by use of molecular symmetry, Journal of Molecular Biology 350 (3) (2005) 528–542.
[34] Bernard R. Brooks, Wenjun Zheng, D. Thirumalai, Allosteric transitions in the chaperonin groel are captured by a dominant normal mode that is most robust to sequence variations, Biophysical Journal 93 (7) (2007) 2289–2299.
[35] Willy Wriggers, Conventions and workflows for using Situs, Acta Crystallographica D 68 (4) (2012) 344–351.
[36] L. Yang, G. Song, R.L. Jernigan, How well can we understand large-scale protein motions using normal modes of elastic network models?, Biophysics Journal 93 (3) (2007) 920–929.