

Scalable anisotropic vibrations of megascale macromolecules

Algorithms

Jordy Homing Lam

December 27, 2023

This is a supporting document submitted to support our article "Scalable anisotropic vibrations of megascale macromolecules". It contains the pseudocodes for the following algorithms:

Algorithm 1 3-D Reverse Cuthill McKee (3DRCM)

Algorithm 2 Modified Gram-Schmidt Vector (MGSV)

Algorithm 3 Iterative Classical Gram-Schmidt (ICGS)

Algorithm 4 p-step Lanczos Factorization (PLF)

Algorithm 5 Implicitly Restarted Lanczos Method (IRLM)

Algorithm 6 Thick Restart Lanczos Method (TRLM)

Algorithm 7 Jacobi-Davidson Method (JDM)

Algorithm 8 Chebyshev-Davidson Method (CDM)

Algorithm 9 Chebyshev Filtered Matrix Vector Product (ChebAv)

1 3-D Reverse Cuthill McKee

A Reverse Cuthill McKee algorithm supplemented with a k-D tree data structure where $k=3$. Analysis of a k-D tree can be found in the work of ‘Bentley, J. L. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 509–517 (1975)’. An empty 1-d array of length l is notated as 0^l . Time complexity marked in comment.

Algorithm 1 3-D Reverse Cuthill McKee

```

1: procedure 3DRCM( $X \in \mathbb{R}^{N \times 3}$ ,  $R_C \in \mathbb{R}^1$ )
2:                                      $\triangleright$  (a) Prepare a 3-D tree  $O(N \log N)$ 
3:    $3dTree \leftarrow KdTree(X)$ 
4:                                      $\triangleright$  (b) Find peripheral node  $O(N \times 3N^{2/3})$ 
5:   Initialize  $D = 0^N$                                       $\triangleright$  Degrees.
6:   for  $i = 0, \dots, N - 1$  do
7:      $|\mathcal{N}(X_i)| = 3dTree(R_C, X_i)$ 
8:      $D[i] \leftarrow |\mathcal{N}(X_i)|$ 
9:    $Inds = Argsort(D)$  ;  $IndsRev = Argsort(Inds)$                                       $\triangleright$  Quicksort.
10:   $m = \max(D)$ 
11:                                      $\triangleright$  (c) Breadth-first loop  $O(3N^{5/3} + 2cm^2N)$ 
12:  Initialize  $L = 0^N$  ;  $\tilde{D} = 0^m$ ;  $n = 0$ 
13:  for  $z = 0, \dots, N - 1$  do
14:    if  $Inds[z] == -1$  then
15:      continue
16:     $L[n] \leftarrow Inds[z]$ ;  $n += 1$ 
17:     $Inds[IndsRev[Inds[z]]] \leftarrow -1$                                       $\triangleright$  Indicate visited.
18:     $Level_s = n - 1$ ;  $Level_t = n$ 
19:    while  $Level_s < Level_t$  do
20:      for  $\tilde{i} = Level_s, \dots, Level_t - 1$  do
21:         $i \leftarrow L[\tilde{i}]$  ;  $n_{old} = n$ 
22:         $\mathcal{N}(X_i) = 3dTree(R_C, X_i)$                                       $\triangleright$  Neighbors in  $R_C$ .
23:        for  $j = 0, \dots, |\mathcal{N}(X_i)| - 1$  do
24:           $\tilde{j} = \mathcal{N}(X_i)[j]$ 
25:          if  $Inds[IndsRev[\tilde{j}]] == -1$  then
26:            continue
27:           $Inds[IndsRev[\tilde{j}]] \leftarrow -1$ 
28:           $L[n] \leftarrow j$  ;  $n += 1$ 
29:         $l_s = 0$ 
30:        for  $k = n_{old}, \dots, n - 1$  do                                      $\triangleright$  Insertion Sort.
31:           $\tilde{D}[l_s] \leftarrow D[L[k]]$  ;  $l_s += 1$ 
32:        for  $k = 1, \dots, l_s - 1$  do
33:           $\tilde{d} \leftarrow \tilde{D}[k]$ 
34:           $\tilde{l} \leftarrow L[n_{old} + k]$ 
35:           $l_t \leftarrow k$ 
36:          while  $l_t > 0$  and  $\tilde{d} < \tilde{D}[l_t - 1]$  do
37:             $\tilde{D}[l_t] \leftarrow \tilde{D}[l_t - 1]$ 
38:             $L[n_{old} + l_t] \leftarrow L[n_{old} + l_t - 1]$ ;  $l_t -= 1$ 
39:           $\tilde{D}[l_t] \leftarrow \tilde{d}$ 
40:           $L[n_{old} + l_t] \leftarrow \tilde{l}$ 
41:         $Level_s \leftarrow Level_t$  ;  $Level_t \leftarrow n$ 
42:                                      $\triangleright$  (d) Reverse
43:  return  $L[:: -1]$ 

```

2 Modified Gram-Schmidt Vector

A Modified Gram-Schmidt (MGS) algorithm to orthogonalize a vector u against the basis $V = [v_0, \dots, v_m]$.

Algorithm 2 Modified Gram-Schmidt Vector

```

1: procedure MGSV( $u \in \mathbb{R}^n; V \in \mathbb{R}^{m \times n}$ )
2:   for  $i = 0, \dots, m-1$  do
3:      $u \leftarrow u - (u^\top v_i) v_i$ 
4:   return  $u$ 

```

3 Iterative Classical Gram-Schmidt

An Iterative Classical Gram-Schmidt (ICGS) algorithm to orthogonalize a vector u against the basis $V = [v_0, \dots, v_m]$. A break is triggered when the deflation is revoked.

Algorithm 3 Iterative Classical Gram-Schmidt

```

1: procedure ICGS( $u \in \mathbb{R}^n; V \in \mathbb{R}^{m \times n}$ )
2:    $r_0 = \|u\|_2$ 
3:   for  $i_{iter} = 1, 2, 3$  do
4:      $u \leftarrow u - VV^\top u$ 
5:      $r_1 = \|u\|_2$ 
6:     if  $r_1 > r_0/2$  then
7:       Break
8:      $r_0 \leftarrow r_1$ 
9:   if  $r_1 \leq r_0/2$  then
10:    Warning! Loss of orthogonality
11:  return  $u$ 

```

4 p-step Lanczos Factorization

A p-step Lanczos Factorization (PLF) to produce p basis between index j_s and j_t . Algorithm 4 PLF is where a polynomial filter can be applied, for example, in Algorithm 8 CF with a first-kind Chebyshev polynomial. Step 9 and 10 are optional; a proof of the spectrum bound can be found in ‘Zhou et al, Bounding the spectrum of large Hermitian matrices, Linear Algebra and its Applications 435(3), 2011, p.480-493, <https://doi.org/10.1016/j.laa.2010.06.034>’. In our case, the lower bound is well known.

Algorithm 4 p-step Lanczos Factorization

```

1: procedure PLF( $A \in \mathbb{R}^{n \times n}, V \in \mathbb{R}^{(m+1) \times n}, \alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^m, j_s = 0, j_t = m$ )
2:   for  $j = j_s, \dots, j_t - 1$  do
3:      $r = Av_j$  ▷  $r = p(A, v_j, a, b)$  if filter in use.
4:      $\alpha_j \leftarrow v_j^\top r$ 
5:      $r \leftarrow r - \alpha_j v_j$ 
6:      $r \leftarrow MGSV(r, V[:j]); r \leftarrow MGSV(r, V[:j])$  ▷ Full Reorthogonalization (FRO).
7:      $\beta_j \leftarrow \|r\|_2$ 
8:      $v_{j+1} = r/\beta_j$ 
9:      $T[j, j] = \alpha; T[j, j-1] = T[j-1, j] = \beta$ 
10:    Solve  $TQ = QD$  ▷ Spectrum upper bound as  $D_{max} + \beta_j \|e_j^\top Q_j\|_\infty$ 
11:  return  $V, \alpha, \beta$ 

```

5 Implicitly Restarted Lanczos Method

An Implicitly Restarted Lanczos Method (IRLM) to compute k smallest eigenpairs. This algorithm calls Algorithm 2 MGSV and Algorithm 4 PLF. At max, 15000 restarts were allowed.

Algorithm 5 Implicitly Restarted Lanczos Method

```

1: procedure IRLM( $A \in \mathbb{R}^{n \times n}, V \in \mathbb{R}^{(m+1) \times n}, \alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^m, \epsilon = 1e-8$ )
2:   Initialize  $v_0 \leftarrow v_{rand} / \|v_{rand}\|_2$ 
3:    $j_s = 0$ 
4:   for  $i_{iter} = 1, \dots, 15000$  do
5:                                      $\triangleright$  (a) Initial p-steps Lanczos Factorization
6:       PLF( $A, V, \alpha, \beta, j_s = j_s, j_t = m$ )
7:                                      $\triangleright$  (b) Implicit Shift
8:        $S, W = eig(\alpha, \beta[1:m-1])$ 
9:        $\theta = Sort(Diag(S))[1:m-k]$ 
10:       $\tilde{\beta} = \beta_{k+p-1}$ 
11:       $Q = I$ 
12:      for  $i = 0 \dots p-1$  do
13:           $T = Tridiag(\alpha, \beta[1:m-1])$ 
14:           $\tilde{Q}\tilde{R} = T - \theta_i I$ 
15:           $T \leftarrow \tilde{Q}^\top T \tilde{Q}$ 
16:           $\alpha, \beta \leftarrow T$ 
17:           $Q \leftarrow Q \tilde{Q}$ 
18:       $\beta \leftarrow [\beta, \tilde{\beta}]$ 
19:                                      $\triangleright$  (c) Implicit Restart
20:       $\sigma = Q[k-1, m-1]$ 
21:       $V[:, k] \leftarrow Q[:, k] V[:, m, :]$ 
22:       $v_k \leftarrow \beta_{k-1} Q[:, k] V[:, m, :] + \sigma \beta_{m-1} v_m$ 
23:       $\beta_{k-1} \leftarrow \|v_k\|_2$ 
24:       $v_k \leftarrow v_k / \beta_{k-1}$ 
25:       $v_k \leftarrow MGSV(v_k, V[:, k]); v_k \leftarrow MGSV(v_k, V[:, k])$ 
26:       $v_k \leftarrow v_k / \|v_k\|_2$ 
27:       $j_s \leftarrow k$ 
28:                                      $\triangleright$  Reset the PLF
29:                                      $\triangleright$  (d) Estimate Convergence
30:      if  $|\beta_{k-1}| < \epsilon$  then
31:          Break
32:       $S, W = eig(Tridiag(\alpha[1:k], \beta[1:k-1]))$ 
33:      return  $S[:, k], W^\top V[:, k]$ 

```

6 Thick Restart Lanczos Method

A Thick Restart Lanczos Method to compute the k smallest eigenpairs. This algorithm calls Algorithm 2 MGSV and Algorithm 4 PLF. Algorithm 4 PLF is where a filter can be applied through the matrix-vector multiplication, for example, in Algorithm 9, a filter is constructed using first-kind Chebyshev polynomial basis. At max, 15000 restarts were allowed.

Algorithm 6 Thick Restart Lanczos Method

```

1: procedure TRLM( $A \in \mathbb{R}^{n \times n}, V \in \mathbb{R}^{(m+1) \times n}, \alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^m, \epsilon = 1e-12$  )
2:   Initialize  $v_0 \leftarrow v_{rand}/\|v_{rand}\|_2$ 
3:    $j_s = 0; b = 0; S = 0; r = v_0$ 
4:   for  $i_{iter} = 1, \dots, 15000$  do
5:                                      $\triangleright$  (a) Reinitialize
6:      $\beta[:k] \leftarrow 0$ 
7:      $\alpha[:k] \leftarrow \text{Diag}(S)$ 
8:      $r \leftarrow \text{MGSV}(r, V[:k])$ 
9:      $r \leftarrow r/\|r\|_2$ 
10:     $v_k \leftarrow r$ 
11:                                      $\triangleright$  (b) p-step Lanczos Factorization, filter inside.
12:    PLF( $A, V, \alpha, \beta, j_s = j_s, j_t = m$ )
13:                                      $\triangleright$  (c) Thick Restart
14:     $T = \text{Tridiag}(\alpha, \beta[:m-1]); T[k, :k] \leftarrow b; T[:k, k] \leftarrow b$ 
15:     $S, W = \text{eig}(T)$ 
16:     $W \leftarrow W[:m, \text{Argort}(\text{Diag}(S))[:k]]$ 
17:     $S \leftarrow \text{Diag}(S)[\text{Argort}(\text{Diag}(S))[:k]]$ 
18:     $V[:k, :] \leftarrow W^\top V$ 
19:     $b = \beta_{m-1} W[m-1, :k]$ 
20:     $j_s \leftarrow k$ 
21:                                      $\triangleright$  Reset the PLF
22:                                      $\triangleright$  (d) Check Convergence
23:    if  $\|b\|_2 < \epsilon$  then
24:      Break
25: return  $S, V$ 

```

7 Jacobi-Davidson Method

A Jacobi-Davidson Method to compute k smallest eigenpairs. This algorithm calls Algorithm 2 MGSV and Algorithm 3 ICGS. The correction z is approximated with Generalized Minimal Residual Iteration (GMRES) routine provided in CuPy; unless otherwise stated, the iteration is stopped if residual error in GMRES reach $1e-6$ or the number of iterations exceeds 20. Note that this can be replaced by a Minimal Residual Iteration (MIREs) routine, which processes symmetric matrices. j_s counts towards the desired number of converged Ritz pair k . Some preconditioners (e.g. ILU(k)) were considered in solving the correction equation, but their size and density become prohibitive as n increases; trivial preconditioner e.g. $\text{diag}(A)^{-1}$ does not improve performance.

Algorithm 7 Jacobi-Davidson Method

```

1: procedure JDM( $A \in \mathbb{R}^{n \times n}, \epsilon = 1e-12, k = 64$  )
2:   Initialize  $v_0 = v_{rand}/\|v_{rand}\|_2$ 
3:    $G = v_0^\top A v_0$ 
4:    $V = [v_0, ]$ 
5:    $Q = [0^n, ]; \Lambda = []$ 
6:    $j_s = 0$ 
7:
8:   for  $i_{iter} = 1, \dots, 15000$  do
9:      $S, W = \text{eig}(G)$ 
10:    while True do
11:                                     ▷ (a) Get Residual
12:       $u = VW[:, 0]; \theta = S[0, 0]$ 
13:       $r = Au - \theta u$ 
14:       $\sigma = \theta$                                      ▷ Propose shift.
15:       $\tilde{Q} = [Q, u]$ 
16:
17:      if  $(\|r\|_2 > \epsilon) \cup ((\dim(S)[0] \leq 1) \cap (j_s \neq k - 1))$  then                                     ▷ Non-convergence.
18:        Break.
19:                                     ▷ (b) Update Projections
20:       $\Lambda \leftarrow [\Lambda, \theta]$ 
21:       $Q \leftarrow \tilde{Q}$ 
22:       $V \leftarrow VW[:, 1:j]; S \leftarrow S[1:j, 1:j]$ 
23:       $G \leftarrow S; W \leftarrow I$ 
24:       $j_s += 1$ 
25:      if  $j_s == k$  then
26:        Return  $\Lambda, Q$                                      ▷ All converged.
27:
28:      if  $\dim(S)[0] == 2k$  then                                     ▷ (c) Restart if workspace is full
29:         $V \leftarrow VW[:, 0:k]; S \leftarrow S[0:k, 0:k]$ 
30:         $G \leftarrow S; W \leftarrow I$ 
31:
32:                                     ▷ (d) Correction equation with GMRES.
33:       $(I - uu^\top)(A - \theta I)(I - uu^\top)z = -r$ 
34:       $z \leftarrow \text{MGSV}(\tilde{Q}, z)$ 
35:       $z \leftarrow \text{ICGS}(V, z); z \leftarrow \text{ICGS}(V, z)$ 
36:       $z \leftarrow z/\|z\|_2$ 
37:                                     ▷ (e) Update Projections
38:       $\tilde{z} = Az$ 
39:       $V \leftarrow [V, z]$ 
40:       $G \leftarrow [G, V^\top \tilde{z}; \tilde{z}^\top V, z^\top \tilde{z}]$ 
41:                                     ▷ (f) Guard not all converged.
42:    return  $\Lambda, Q$ 

```

8 Chebyshev-Davidson Method

A Chebyshev-Davidson Method to compute k smallest eigenpairs. This algorithm calls Algorithm 2 MGSV and Algorithm 3 ICGS. λ_{min} is the lower bound of spectrum. λ_{max} is upper bound of the spectrum obtained from Algorithm 4 PLF. θ_s is the 'squeezing' moving lower bound.

Algorithm 8 Chebyshev-Davidson Method

```

1: procedure CDM( $A \in \mathbb{R}^{n \times n}, \lambda_{min}, \lambda_{max}, \epsilon = 1e - 12, k = 64,$ )
2:   Initialize  $v_0 = v_{rand}/\|v_{rand}\|_2$ 
3:    $G = v_0^\top A v_0$ 
4:    $V = [v_0, ]$ 
5:    $Q = [A v_0, ]; \Lambda = []$ 
6:    $j_s = 0; j = 1$ 
7:    $u = V[:, j_s]$ 
8:    $\theta_s = (\lambda_{max} + G[0, 0])/2$ 
9:
10:  for  $i_{iter} = 1, \dots, 15000$  do
11:     $z = p(A, u, \theta_s, \lambda_{max})$  ▷ (a) Low-pass Chebyshev filter
12:     $z \leftarrow MGSV(\tilde{Q}, z)$ 
13:     $z \leftarrow ICGS(V, z)$ 
14:     $z \leftarrow z/\|z\|_2$ 
15:
16:     $\tilde{z} = Az$  ▷ (b) Update Projections
17:     $V[:, j] \leftarrow z$ 
18:     $Q[:, j] \leftarrow \tilde{z}$ 
19:     $G[j, j_s : j] \leftarrow Q[:, j]^\top V[:, j_s : j + 1]$ 
20:     $G[j_s : j, j] \leftarrow G[j, j_s : j]^\top$ 
21:     $S, W = eig(G)$ 
22:
23:     $j_r = j + 1$  ▷ (c) Restart if workspace is full
24:    if  $j + 1 \geq 2k$  then
25:       $j_r = \max(j_s + 1, k + 5, \min(k + j_s, 2k - 5))$ 
26:       $V[:, j_s : j_r] \leftarrow V[:, j_s : j + 1]W[:, 0 : j_r - j_s + 1]$ 
27:       $Q[:, j_s : j_r] \leftarrow Q[:, j_s : j + 1]W[:, 0 : j_r - j_s + 1]$ 
28:       $G[j_s : j_r, j_s : j_r] \leftarrow S[0 : j_r - j_s + 1, 0 : j_r - j_s + 1]; W \leftarrow I$ 
29:
30:       $\theta = S[0]$  ▷ (d) Get Residual
31:       $r = Q[:, j_s] - \theta V[:, j_s]$ 
32:       $u = V[:, j_s]$  ▷ Next  $u$ 
33:      if  $\|r\|_2 < \epsilon$  then
34:         $\Lambda \leftarrow [\Lambda, \theta]; j_s += 1$ 
35:        if  $j_s \geq k$  then
36:          Return  $\Lambda, Q$  ▷ All converged.
37:           $u = V[:, j_s - 1]$  ▷ Next  $u$ 
38:         $j = j_r$ 
39:
40:     $\theta_s = \max(S_{median}, \lambda_{min})$  ▷ (e) Update moving lower bound
41:
42:  return  $\Lambda, Q$  ▷ (f) Guard not all converged.

```

9 Chebyshev Filtered Matrix Vector Product

Matrix vector product filtered on the basis of first-kind Chebyshev polynomial. $p(t) = \sum_{j=0}^{j=M} \kappa_j T_j(t)$. Note that (1) κ_j is a set of user-defined coefficients e.g. the scaled coefficients in equation 28 of the main text. (2) In Algorithm 8 CDM, only the M-th degree polynomial is considered.

Algorithm 9 Chebyshev Filtered Matrix Vector Product

```
1: procedure CHEBAV( $A \in \mathbb{R}^{n \times n}, v \in \mathbb{R}^{n \times 1}, a, b$ )
2:    $e =: (b - a)/2; c =: (b + a)/2$ 
3:    $y = \kappa_0 v$ 
4:    $v_+ = \kappa_1 (Av - cv)/e$ 
5:    $y += \kappa_1 v_+$ 
6:    $v_- = v; v \leftarrow v_+;$ 
7:   for  $j = 2, \dots, M$  do
8:      $v_+ = 2/e(Av - cv) - v_-$ 
9:      $y += \kappa_j v_+$ 
10:     $v_- \leftarrow v; v \leftarrow v_+$ 
11:  return  $y$ 
```
