New Directions in Numerical Linear Algebra and High Performance Computing:
Celebrating the 70th Birthday of Jack Dongarra, Manchester, July 7-8, 2021 (virtual)

Many eigenpair computation via Hotelling deflation

Zhaojun Bai
University of California, Davis

Introduction

1. Problem statement ("many eigenpair computation"):

   Let (λ_i, v_i) be the eigenpairs of a $n \times n$ symmetric matrix A with the eigenvalue ordering $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$. Compute the partial decomposition:

   $$AV_{n_e} = V_{n_e} \Lambda_{n_e},$$

   where $\Lambda_{n_e} = diag(\lambda_1, \ldots, \lambda_{n_e})$,
   Interested in the cases where n is "huge" and n_e is "large"

2. Emerging applications:
   - electronic structure calculations of Lithium-ion electrolyte,
   - graphene,
   - dyanmics analysis of viral capsids of supramolecular systems such as Zika and West Nile viruses (structural biology)
   - ...

Introduction

3. Existing approaches – *an incomplete list*:
   - Full eigenvalue decomposition:
     - LAPACK, ScaLAPACK, PLASMA, MAGMA
     - ELPA (Eigenvalue Solves for Petaflop Applications), https://elpa.mpcdf.mpg.de/
     - EigenExa, https://www.r-ccs.riken.jp/labs/lpnctrt/projects/eigenexa/
     - ...
     - QDWHeig [Sukkari, Ltaief, Keyes at KAUST]
     - SLATE [Gates *et al*, U. of Tennessee]

     Stable, but expensive, $O(n^2)$ storage and $O(n^3)$ flops

   - "Spectrum slicing:"
     - SLEPc, https://slepc.upv.es/
     - EVSL, http://www.cs.umn.edu/$\sim$saad/software
     - FEAST, http://www.ecs.umass.edu/$\sim$polizzi/feast/
     - z-Pares, http://zpares.cs.tsukuba.ac.jp/
     - ...
     - SISLICE [Williams-Young and Yang, LBNL]

     Scalable, but issues with duplicate/missing eigenvalues between slices, ...

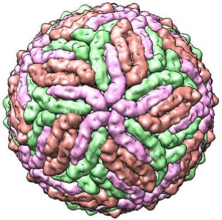
Introduction

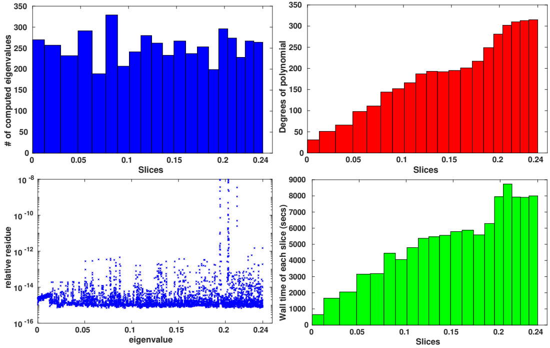
4. Using Lanczos or any other subspace projection methods for many eigenpairs are challenging – numerically and computationally:

   - needs large subspace (memory) M, e.g., $m = 2n_e$
   - require (internal) locking (deflation) to avoid danger of converging again to the same eigenvalues ($O(nm^2)$ flops)

Introduction

5. Many eigenpair computation is challenging even when vast computational resources are available.
   Case demo: **EVSL** (http://www.cs.umn.edu/∼saad/software)





Left: Dengue virus model (`www.rcsb.org/structure/4cct`). $n = 307,260$
Middle: DOS sliced $5,076$ eigenvalues into 20 subintervals (top) and the relative residual norms of eigenpairs (bottom). $\|\widehat{V}^T\widehat{V} - I\|_F = O(10^{-6})$.
Right: Degrees of filter polynomials (top) CPU timing (bot.) for each slices

Introduction

6. This talk is about an on-going project on

   Lanczos + Hotelling deflation + Communication-avoiding

   for computing many eigenpairs.

Rest of the talk

I. Hotelling deflation = Explicit External Deflation (EED)

II. Backward stability of EED

III. Communication-avoiding algorithm for MPK (CA-MPK)

IV. Eigensolver sTRLED (= TRLan + EED + CA-MPK)

V. Concluding remarks

Joint work with

- Jack J. Dongarra, Univ of Tennessee
- Chao-Ping Lin, UC Davis
- Ding Lu, Univ of Kentucky
- Ichitaro Yamazaki, Sandia National Labs.

I. Explicit External Deflation (EED)

1. Hotelling deflation (EED = Explicit External Deflation)

Let $AV = V\Lambda$ be the eigen-decomposition of A, partition

$$V = [V_k \; V_{n-k}] \quad \textit{and} \quad \Lambda = \begin{bmatrix} \Lambda_k & \\ & \Lambda_{n-k} \end{bmatrix},$$

and define

$$\widehat{A} = A + V_k \Sigma_k V_k^T,$$

where $\Sigma_k = \mathit{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$ are shifts. Then
(a) The eigenvalues of $\widehat{A}$ are

$$\lambda_i(\widehat{A}) = \begin{cases} \lambda_i + \sigma_i & \textit{for } 1 \le i \le k \\ \lambda_i & \textit{for } k+1 \le i \le n \end{cases}$$

(b) A and $\widehat{A}$ have the same eigenvectors

Therefore, one can use proper shifts σ_i to move "computed" eigenvalues away, and then compute the next batch of "favorite" eigenvalues for an eigensolver.

I. Explicit External Deflation

2. Governing equations of the exact EED process: for $j = 1, 2, \ldots,$

$$A_j = A_{j-1} + \sigma_j v_j v_j^T = A + V_j \Sigma_j V_j^T \quad \text{and } A_0 = A.$$

$$A_j v_{j+1} = \lambda_{j+1} v_{j+1}$$

$$A_j V_j = V_j (\Lambda_j + \Sigma_j)$$

with initial $A v_1 = \lambda_1 v_1$, where $\Sigma_j = \text{diag}(\sigma_1, \ldots, \sigma_j)$.

I. Explicit External Deflation

3. Benefits of EED:

- Easily incooperated into existing eigensolvers, such as TRLan and ARPACK.
- Small projection subspace dimensions (core memory requirements) even for many eigenpairs.
- No need to explicitly (re)-orthogonalize the projection subspace to the computed eigenvectors.
- Accelerated convergence with warm start for A_j.
- Straightforward extension to generalized symmetric eigenproblem $Av = \lambda Bv$:

$$(A + \sigma B V_k V_k^T B)x = \lambda Bx$$

- Readily exploit structures of A and B, for example, see EED for the linear response eigenvalue problem [Bai-Li-Lin'17].
- Extensible to other eigenvalue-type problems, such as SVD, sparse PCA.
- Nonlinear eigenvector problem (NEPv) analogy: level shifting

I. Explicit External Deflation

4. **Two key issues of EED:**

   (a) Numerical linear algebra issue:

   Numerical stability with approximate eigenvectors $\widehat{V}_k$*:*

   $$\widehat{A}_j = A + \widehat{V}_j \Sigma_j \widehat{V}_j^T$$

   (b) High performance computing issue:

   Cost of matrix powers kernel (MPK) for generating Krylov subspace:

   $$\left[p_0(\widehat{A}_j)v_0,\ p_1(\widehat{A}_j)v_0,\ \ldots,\ p_s(\widehat{A}_j)v_0 \right]$$

   where $\{p_k(\cdot)\}$ *are recursively defined polynomials.*

II. Backward stability of EED

1. Mixed messages from previous work:
   [Wilkinson'65], [Parlett'82], [Saad'89], [Jang/Lee'06], ...

2. Governing equations of *inexact* EED:

$$
\begin{aligned}
\widehat{A}_j &= \widehat{A}_{j-1} + \sigma_j \widehat{v}_j \widehat{v}_j^T = A + \widehat{V}_j \Sigma_j \widehat{V}_j^T \\
\widehat{A}_j \widehat{v}_{j+1} &= \widehat{\lambda}_{j+1} \widehat{v}_{j+1} + \eta_{j+1} \\
\widehat{A}_j \widehat{V}_j &= \widehat{V}_j (\widehat{\Lambda}_j + \Sigma_j) + \widehat{V}_j \Sigma_j \Phi_j + E_j
\end{aligned}
$$

where $\widehat{A}_0 = A$,

$$
\begin{aligned}
\|\eta_{j+1}\| &\leq tol \cdot \|A\|, \\
\Phi_j &= \mathrm{utri}(\widehat{V}_j^T \widehat{V}_j - I_j), \\
E_j &= [\eta_1, \eta_2, \ldots, \eta_j].
\end{aligned}
$$

and tol is prescribed relative residual tolerance.

II. Backward stability of EED

3. Metrics of backward stability for computed eigenpairs $(\widehat{\Lambda}_{j+1}, \widehat{V}_{j+1})$ with relative residual tolerance tol:

   - The loss of orthogonality

   $$\omega_{j+1} = \|\widehat{V}_{j+1}^T \widehat{V}_{j+1} - I\|_F = O(tol)$$

   - The symmetric backward error norm

   $$\delta_{j+1} = \min_{\Delta \in \mathcal{H}} \|\Delta\|_F = O(tol \cdot \|A\|),$$

   where

   $$\mathcal{H} = \left\{ \Delta \mid (A + \Delta)Q_{j+1} = Q_{j+1}\widehat{\Lambda}_{j+1},\, \Delta = \Delta^T,\, Q_{j+1} = \text{orth}(\widehat{V}_{j+1}) \right\}$$

II. Backward stability of EED

4. Two key quantities associated with the shifts
   - Spectral gap
     $$\gamma_j \equiv \min_{\lambda \in \mathcal{I}_{j+1}, \theta \in \mathcal{J}_j} |\lambda - \theta|,$$
     where $\mathcal{I}_{j+1} = \{\widehat{\lambda}_1, \ldots, \widehat{\lambda}_j, \widehat{\lambda}_{j+1}\}$ is the set of computed eigenvalues, and $\mathcal{J}_j = \{\widehat{\lambda}_1 + \sigma_1, \ldots, \widehat{\lambda}_j + \sigma_j\}$ is the set of computed eigenvalues with shifts.
   - Shift-gap ratio
     $$\tau_j \equiv \frac{1}{\gamma_j} \cdot \max_{1 \le i \le j} |\sigma_i|.$$
5. Theorem.
   Under mild assumptions, if
   $$\gamma_j^{-1}\|A\| = O(1) \quad \textit{and} \quad \tau_j = O(1), \qquad (1)$$
   then
   $$\omega_{j+1} = O(tol) \quad \textit{and} \quad \delta_{j+1} = O(tol \cdot \|A\|).$$
6. *Rule of Thumb:*
   dynamical choice of shifts σ_j to satisfy the conditions (1).

II. Backward stability of EED

7. Numerics of TRLED = TRLan + EED:

- Test matrices:

matrix	n	$[\lambda_{\min}, \lambda_{\max}]$	$[\lambda_{\text{low}}, \lambda_{\text{upper}}]$	n_e
Laplacian	$40,000$	$[0, 7.9995]$	$[0, 0.07]$	205
worms20	$20,055$	$[0, 6.0450]$	$[0, 0.05]$	289
SiO	$33,401$	$[-1.6745, 84.3139]$	$[-1.7, 2.0]$	182
Si34H36	$97,569$	$[-1.1586, 42.9396]$	$[-1.2, 0.4]$	310
Ge87H76	$112,985$	$[-1.214, 32.764]$	$[-1.3, -0.0053]$	318
Ge99H100	$112,985$	$[-1.226, 32.703]$	$[-1.3, -0.0096]$	372

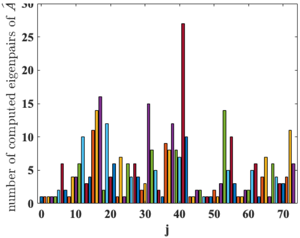
II. Backward stability of EED

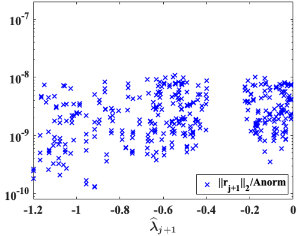
7. Numerics of TRLED = TRLan + EED, *cont'd*

- Results:

matrix	$\widehat{n}_e$	$j_{\max}$	$\omega_{\widehat{n}_e}$	$\|R_{\widehat{n}_e}\|_{\mathrm{F}}/\mathtt{Anorm}$	CPU time (sec.)	
					TRLED	TRLan
Laplacian	205	60	$1.93 \cdot 10^{-8}$	$6.33 \cdot 10^{-8}$	66.5	86.0
worms20	289	86	$2.63 \cdot 10^{-8}$	$7.24 \cdot 10^{-8}$	57.3	74.8
SiO	182	41	$2.33 \cdot 10^{-8}$	$4.71 \cdot 10^{-8}$	42.4	47.1
Si34H36	310	72	$3.41 \cdot 10^{-8}$	$7.50 \cdot 10^{-8}$	309.9	310.4
Ge87H76	318	66	$4.08 \cdot 10^{-8}$	$8.50 \cdot 10^{-8}$	388.7	421.0
Ge99H100	372	74	$3.65 \cdot 10^{-8}$	$7.63 \cdot 10^{-8}$	501.1	533.4

- EED profile of Ge99H100:





II. Backward stability of EED

7. Numerics of TRLED = TRLan + EED, *cont'd*

   - Observations:

     (1) All desired eigenvalues are successfully computed: $n_e = \widehat{n}_e$

     (2) all computed eigenpairs are backward stable:

     $$\omega_{n_e} = O(tol) \quad \text{and} \quad \delta_{n_e} \approx \|R_{n_e}\| = O(tol \cdot \|A\|).$$

     (3) EED does not slow down, in fact, slightly faster, partially due to smaller "internal" memory usage and warm start.

8. With proper choice of shifts, Hotelling deflation (EED) would not compromise numerical stability of an eigensolver.

[Lin, Lu and Bai, *arXiv:2105.01298*, May 2021]

III. Communication-avoiding algorithm for computing MPK

1. Sparse-plus-low-rank matrix powers kernel (MPK)

$$[p_0(B)v, p_1(B)v, \ldots, p_s(B)v]$$

   with

$$B = A + \sigma U_k U_k^T = \text{sparse} + \text{low rank}$$

   and $AU_k = U_k \Lambda_k$, $U_k^T U_k = I_k$ and $p_j(\cdot)$ are polynomials defined *recursively*.

2. For simplicity, consider polynomials $\{p_j(\cdot)\}$ in the monomial basis and compute the MPK:

$$\begin{aligned}[p_0(B)v, p_1(B)v, \ldots, p_s(B)v] &= \left[x, Bx, B^2x, \ldots, B^s x\right] \\ &\equiv \left[x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(s)}\right]\end{aligned}$$

3. Standard MPK algorithm for computing $x^{(1)}, x^{(2)}, \ldots$

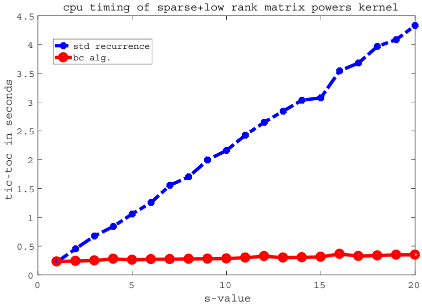
   1: $x^{(0)} = x$;
   2: **for** $j = 1 : s$ **do**
   3: $\quad x^{(j)} = Bx^{(j-1)} = Ax^{(j-1)} + \sigma(U_k(\underline{U_k^T x^{(j-1)}}))$
   4: **end for**

III. Communication-avoiding algorithm for computing MPK

4. Performance example

- $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
- MPK $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
- Timing of the standard algorithm in MATLAB on a desktop – blue curve.

$n = 250\,000$ and $k = 1\,000$



- **Q: what's the red curve?**

III. Communication-avoiding algorithm for computing MPK

5. An algorithm has two costs:

$$\text{arithmetic (flops)} + \text{movement of data (communication)}$$

6. Communication is the bottleneck on modern architectures.

7. Q: How to exploit the sparse-plus-low-rank matrix structure to reduce communication cost?

   Ans.: *use a specialized communication-avoiding algorithm developed by*
   - *Leiserson-Rao-Toledo'97* (“out-of-core”) *and*
   - *Knight-Carson-Demmel'13* (“exploiting data sparsity”)

III. Communication-avoiding algorithm for computing MPK

8. **Communication-Avoiding (CA) algorithm** for the MPK

$x^{(0)} = x$

$b_0 = U_k^T x^{(0)}$

$W_k^{(j)} = \Lambda_k^j + \sigma \sum_{i=1}^{j} \Lambda_k^{i-1}(\Lambda_k + \sigma)^{j-i}$ *for* $j = 1 : s-1$,

$b_j = W_k^{(j)} b_0$ *for* $j = 1 : s-1$

$[c_0, c_1, \ldots, c_{s-1}] = U_k[b_0, b_1, \ldots, b_{s-1}]$

for $j = 1 : s$ **do**

  $x^{(j)} = Ax^{(j-1)} + \sigma c_{j-1}$

end for

9. Benefits of CA-MPK algorithm:

- Reduced flops

$$\text{flops}_{\text{std}} = nnzA \cdot s + nks + nks$$

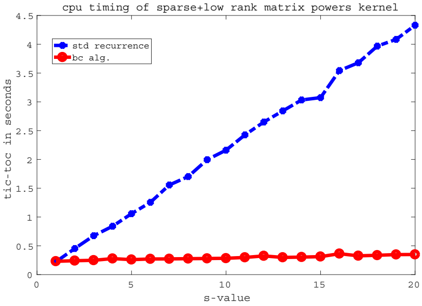
$$\text{flops}_{\text{ca}} = nnzA \cdot s + nks + nk + O(k^2 s)$$

- Reduced movement of data: U_k is only accessed twice.

III. Communication-avoiding algorithm for computing MPK

10. Performance example, *cont'd*:
    - $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
    - MPK $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
    - Timing in MATLAB
        - Standard algorithm – blue curve
        - CA algorithm – red curve

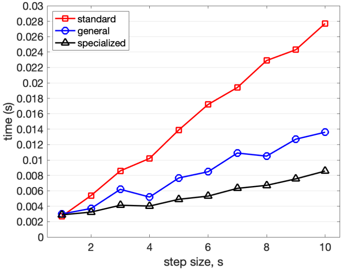
$n = 250000$ and $k = 1000$



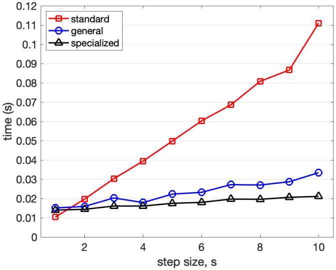
III. Communication-avoiding algorithm for computing MPK

10. Performance example, *cont'd*:

- $B = A + \sigma U_k U_k^T$, where A is 2D Laplacian and U_k are eigenvectors
- MPK $V_s = [p_0(B)v_0, p_1(B)v_0, \ldots, p_s(B)v_0]$
- Timing in one node (32 cores) of Cori (NERSC)



$k = 100$



$k = 200$

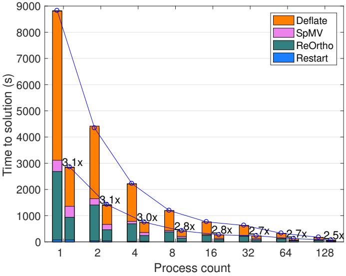
11. Rounding error analysis of CA-MPK

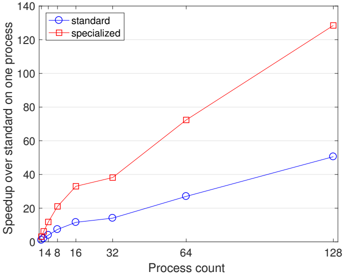
IV. Lanczos algorithm with EED and MPK

1. Lanczos algorithm
   - Lanczos process
     $$AQ_m = Q_m T_m + \beta_m q_{m+1} e_m^T$$
     where
     $$\text{span}\{Q_j\} = \text{span}\{q, Aq, \ldots, A^{m-1}q\} \quad \text{(Krylov subspace)}$$
   - Rayleigh-Ritz approximation
     $$T_m x_i = \theta_i x_i$$
     $$(\lambda_i, v_i) \approx (\theta_i, Q_m x_i)$$
2. Lacnzos algorithm is efficient for computing a few exterior eigenvalues (and eigenvectors).
3. Two main kernels
   - Matrix-Vector multiply (SpMV) for generating Krylov subspace: Aq
   - Re-orthogonalization for maintaining orthonormal basis vectors: Q_j
4. Two variants of Lanczos method:
   - Thick-restart Lanczos (TRLan) – control m size
   - s-step Lanczos (s-Lanczos) – reduce communication cost by using MPK

5. sTRLED = s-step-TRLan + EED + CA-MPK
6. Preliminary results on strong-parallel scaling of sTRLED on multi-processor for `Si87H76`:

$n = 240,369$, $n_e = 700$
computing 100 eigenvalues at a time with $m = 200$ and $s = 5$





[Bai, Dongarra, Lu, Yamazaki, IPDPS19]

V. Concluding remarks

1. Many eigenpairs computation in emerging applications, a challenging problem even when vast computational resources are available.

2. Two techniques discussed in this talk:

   - Explicit external deflation (EED) for reliably moving away computed eigenpairs,

   - a communication-avoiding matrix powers kernel (CA-MPK) for fast sparse-plus-low-rank MPK

3. *The capability of being able to efficiently compute large number of eigenvalues will not just be appealing, but also mandatory for the next generation of eigensolvers.*