Domain decomposition techniques for contour integration eigenvalue solvers

Vassilis Kalantzis, Yousef Saad
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, USA
Email: {kalantzi,saad}@cs.umn.edu

James Kestyn, Eric Polizzi
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA, USA
Email: {kestyn,polizzi}@ecs.umass.edu

I. Introduction

A common approach for computing all eigenvalue-eigenvector (λ, x) pairs lying inside an interval $[\alpha, \beta]$ of a large and sparse symmetric matrix A is via a Rayleigh-Ritz (projection) process on a well-selected low-dimensional subspace, which, ideally, spans an invariant subspace associated with the desired eigenvalues. In recent years, much interest has been generated in techniques in which this low-dimensional subspace is extracted via an approximation of the spectral projector obtained by numerically integrating the resolvent $(\zeta I - A)^{-1}$, $\zeta \in \mathbb{C}$ on a closed complex contour that encloses the desired eigenvalues.

More specifically, let Γ be a smooth, counter-clockwise oriented curve, that encloses the eigenvalues in $[\alpha, \beta]$. Then, the spectral projector $\mathcal{P}$ associated with the eigenpairs of interest can be approximated as:

$$\mathcal{P} = \frac{1}{2i\pi} \int_\Gamma (\zeta I - A)^{-1} d\zeta \approx \sum_{j=1}^{N_c} \omega_j (\zeta_j I - A)^{-1}, \quad (1)$$

where $\{\zeta_j, \omega_j\}_{1 \le j \le N_c}$ are N_c complex quadrature node-weight pairs of the quadrature rule used (i.e., Gauss-Legendre, midpoint). From a numerical viewpoint, contour integration eigenvalue solvers can be viewed as rational filtering techniques, which convert the solution of the original eigenvalue problem into the solution of a series of complex symmetric linear systems with multiple right-hand sides. A popular contour integration eigensolver is the FEAST method [8], which essentially applies subspace iteration on matrix $\sum_{j=1}^{N_c} \omega_j (\zeta_j I - A)^{-1}$.

The main goal of this poster is to improve the parallel performance of FEAST by using domain decomposition techniques [3], [4], [6]. We show that domain decomposition can lead to faster, and often more scalable, computations within the FEAST framework.

II. Acceleration of FEAST by domain decomposition

Dividing the discretized domain in P (non-overlapping) subdomains and reordering matrix A so that interior variables in each subdomain appear before the interface ones, we get:

$$A = \begin{pmatrix} B_1 & & & E_1 \\ & \ddots & & \vdots \\ & & B_P & E_P \\ E_1^T & \dots & E_P^T & C \end{pmatrix}, \quad C = \begin{pmatrix} C_1 & E_{12} & \dots & E_{1P} \\ E_{21} & C_2 & \dots & E_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ E_{P1} & E_{P2} & \dots & C_P \end{pmatrix}$$

where $E_{ij} \neq 0$ only if subdomains i and j are adjacent. We will compactly write $A = [B, E; E^T, C]$, where $B \in \mathbb{R}^{d \times d}$, $E \in \mathbb{R}^{d \times s}$, $C \in \mathbb{R}^{s \times s}$, and $n = d + s$. With the above reordering, $(\zeta I - A)^{-1}$ can be written through the identity:

$$(A - \zeta I)^{-1} = \begin{pmatrix} (B - \zeta I)^{-1} + F(\zeta) S(\zeta)^{-1} F(\zeta)^T & -F(\zeta) S(\zeta)^{-1} \\ -S(\zeta)^{-1} F(\zeta)^T & S(\zeta)^{-1} \end{pmatrix}$$

where $F(\zeta) = (B - \zeta I)^{-1} E$ and $S(\zeta) = C - \zeta I - E^T (B - \zeta I)^{-1} E$ is the *Schur complement* matrix. If $V \in \mathbb{R} \in \mathbb{R}^{n \times \hat{r}}$ is a properly chosen initial subspace, each FEAST iteration can be applied by computing $Z = \sum_{j=1}^{N_c} \omega_j (\zeta_j I - A)^{-1} V$, and orthonormalizing $Z \in \mathbb{R}^{n \times \hat{r}}$. For each quadrature node $\zeta_j,\ j = 1, \dots, N_c$, and each one of the $\hat{r}$ columns in V, we must solve two linear systems with $B - \zeta_j I$ and one linear system with $S(\zeta_j)$. The factorization of each $B - \zeta_j I,\ j = 1, \dots, N_c$ is decoupled into factorizations of the matrices $B_i - \zeta_j I,\ i = 1, \dots, P$, each one being local to the i^{th} subdomain. On the other hand, the solutions with $S(\zeta_j)$ are distributed.

III. Parallel implementation

We partition the discretized domain in P non-overlapping subdomains with the help of a graph partitioner, e.g., METIS [5], and then we assign each subdomain to a distinct processor group. The different processor groups communicate by means of the Message Passing Interface standard (MPI) [10]. Linear system solutions and Matrix-Vector products with matrices $B - \zeta_j I,\ j = 1, \dots, N_c$ and $E,\ E^T$, respectively, are performed in a completely decoupled manner. To factorize $B - \zeta_j I$ we ported our software to the shared-memory, multithreaded version of the Pardiso library (version 5.0.0) [7]. Thus, the proposed implementation can take advantage of an additional layer of parallelism by exploiting more than one compute threads per MPI process when factorizing and/or solving linear systems with matrices $B_i,\ i = 1, \dots, P$. The advantage of this hybrid approach is that increasing the amount of parallelism does not also increase the size of the

TABLE I
"ITS" DENOTES THE NUMBER OF ITERATIONS IN SUBSPACE ITERATION MADE BY THE CI-M AND DI-DD SCHEMES (THEY ARE IDENTICAL), USING DIFFERENT CHOICES OF N_c AND $\hat{r}$. P DENOTES THE NUMBER OF SUBDOMAINS (ONE MPI PROCESS PER SUBDOMAIN).

		$P = 64$		$P = 128$		$P = 256$	
	Its	CI-M	CI-DD	CI-M	CI-DD	CI-M	CI-DD
$N_c = 2$							
$\hat{r} = 3r/2$	9	3,922.7	2,280.6	2,624.3	1,242.4	1,911.2	859.5
$\hat{r} = 2r$	5	2,863.2	1,764.5	1,877.7	998.5	1,255.5	615.3
$N_c = 4$							
$\hat{r} = 3r/2$	5	4,181.5	2,357.0	2,815.7	1,280.2	1,874.1	877.5
$\hat{r} = 2r$	4	4,330.3	2,571.4	2,869.5	1,462.9	2,023.2	1,036.2
$N_c = 6$							
$\hat{r} = 3r/2$	3	3,710.3	2,068.2	2,504.1	1,122.1	1,790.8	766.5
$\hat{r} = 2r$	3	4,774.8	2,798.5	3,177.7	1,595.2	2,743.6	1,125.1
$N_c = 8$							
$\hat{r} = 3r/2$	3	4,911.6	2,722.2	3,318.7	1,476.1	2,367.7	1,006.5
$\hat{r} = 2r$	2	4,204.7	2,445.2	2,802.1	1,395.4	1,806,6	982.1

interface region. The distributed linear system solutions with $S(\zeta_j),\ j = 1,\dots,N_c$ are performed either by a parallel sparse direct solver, e.g., MUMPS [1], or a Krylov subspace iterative solver, e.g. GMRES [9], combined with a domain decomposition preconditioner.

IV. EXPERIMENTS

The experiments performed at the `Mesabi` Linux cluster at Minnesota Supercomputing Institute. Each node features two sockets, with each socket being equipped with twelve physical cores at 2.5 GHz and 32 GB of system memory. The numerical schemes were implemented in C/C++ and built on top of the PETSc [2] and Intel Math Kernel scientific libraries. The source files were compiled with the Intel MPI compiler `mpiicpc`, using the -O3 optimization level. For the numerical integration we used the Gauss-Legendre quadrature rule.

Our first experiment demonstrates a comparison of our DD-based FEAST, denoted as Contour Integration-DD (CI-DD) against a PETSc-based implementation of the FEAST algorithm, referred to as Contour Integration-MUMPS (CI-M), which utilized MUMPS to factorize and solve the linear systems with matrices $A - \zeta_j I,\ j = 1,\dots,N_c$ (not a domain decomposition approach). The test problem was a 2D discretized Laplacian of size $n = 1500^2$, for which we sought all eigenpairs located inside the interval $[\alpha,\beta] = [(\lambda_{1000} + \lambda_{1001})/2, (\lambda_{1200} + \lambda_{1201})/2]$ to at least eight digits of accuracy each. The CI-DD approach is faster and slightly more scalable than the CI-M scheme.

In our second experiment we compare the CI-DD and CI-MM approaches on a 3D discretized Laplacian of size $n = 150^3$, for which we sought all eigenpairs lying inside the interval $[\alpha,\beta] = [(\lambda_{100} + \lambda_{101})/2, (\lambda_{200} + \lambda_{201})/2]$. For CI-DD, the linear systems with $S(\zeta_j),\ j = 1,\dots,Nc$ were solved by the GMRES iterative solver, preconditioned by an approximation of $S(\zeta_j)$ in which we dropped all entries which magnitude was less or equal than 1e-1. Figure 1 shows a comparison of three different implementations: a) the CI-M scheme, b) the CI-DD scheme with one thread per MPI process, and c) the CI-DD scheme using a fixed number of $P = 32$ MPI process (subdomains) and using the extra compute cores as threads. Because of the increasing cost to apply the preconditioner, the CI-DD (MPI) actually becomes slower as we increase the number of compute cores.

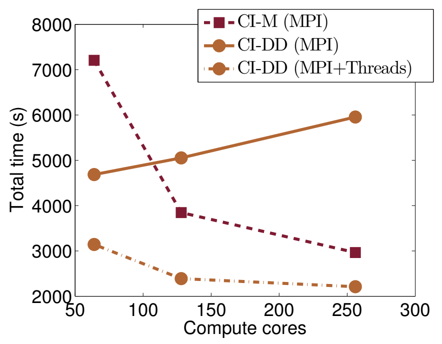

Fig. 1. CI-DD (MPI+Threads) utilizes 32 MPI processes (subdomains), each with "Compute cores" / 32 threads.

V. ACKNOWLEDGMENT

This work was supported jointly by NSF under award CCF-1505970 and by the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences under award number DE-SC0008877.

REFERENCES

[1] P. R. AMESTOY, I. S. DUFF, J.-Y. L'EXCELLENT, AND J. KOSTER, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM Journal on Matrix Analysis and Applications, 23 (2001), pp. 15–41.

[2] S. BALAY, S. ABHYANKAR, M. F. ADAMS, J. BROWN, P. BRUNE, K. BUSCHELMAN, L. DALCIN, V. EIJKHOUT, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, K. RUPP, B. F. SMITH, S. ZAMPINI, AND H. ZHANG, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015.

[3] V. KALANTZIS, J. KESTYN, E. POLIZZI, AND Y. SAAD, *Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems, submitted*, 2016.

[4] V. KALANTZIS, R. LI, AND Y. SAAD, *Spectral schur complement techniques for symmetric eigenvalue problems*, Electronic Transactions on Numerical Analysis, 45 (2016), pp. 305–329.

[5] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing, 20 (1998), pp. 359–392.

[6] J. KESTYN, V. KALANTZIS, E. POLIZZI, AND Y. SAAD, *Pfeast: A high performance sparse eigenvalue solver using distributed-memory linear solvers*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016.

[7] A. KUZMIN, M. LUISIER, AND O. SCHENK, *Fast methods for computing selected elements of the green's function in massively parallel nanoelectronic device simulations*, in Proceedings of the 19th International Conference on Parallel Processing, Euro-Par'13, Berlin, Heidelberg, 2013, Springer-Verlag, pp. 533–544.

[8] E. POLIZZI, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), p. 115112.

[9] Y. SAAD AND M. H. SCHULTZ, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[10] M. SNIR, S. OTTO, S. HUSS-LEDERMAN, D. WALKER, AND J. DONGARRA, *MPI-The Complete Reference, Volume 1: The MPI Core*, MIT Press, Cambridge, MA, USA, 2nd. (revised) ed., 1998.