libOMM: Recent Development and Future Directions

Haizhao Yang
Department of Mathematics, Duke University

Joint work with Jianfeng Lu
Department of Mathematics, physics, and chemistry, Duke University

August, 2017

Introduction

Optimization model

- The eigenspace associated to the first n smallest eigenvalues is given by the trace minimization
$$E = \min_{X \in \mathbb{R}^{N \times n}} E_c(X) = \min_{X \in \mathbb{R}^{N \times n},\, X^* X = I_n} \operatorname{tr}(X^* H X),$$
where I_n is an $n \times n$ identity matrix
- The orthonormality constraint $X^* X = I_n$ is expensive.
- In zero-temperature systems, target quantity:
$$XX^* \in \mathbb{R}^{N \times N}.$$

Introduction

Optimization model

- Instead, we search for the eigenspace by an unconstrained minimization

$$E = \min_{X \in \mathbb{R}^{N\times n}} E_{\mathrm{omm}}(X) = \min_{X \in \mathbb{R}^{N\times n}} \operatorname{tr}\big((2I_n - X^*X)(X^*(H-\eta I_n)X)\big),$$

  where η is a proper shift.
- This is the orbital minimization method (OMM) originally proposed for a linear scaling density matrix method using sparse BLAS. [Mauri, Galli, Car,,Phys. Rev. B, 1993; Ordejon, Drabold, Grumbach, Martin, Phys. Rev. B, 1993]

Introduction

Optimization model

- Instead, we search for the eigenspace by an unconstrained minimization

$$E = \min_{X \in \mathbb{R}^{N \times n}} E_{\mathrm{omm}}(X) = \min_{X \in \mathbb{R}^{N \times n}} \mathrm{tr}\big((2I_n - X^* X)(X^*(H - \eta I_n)X)\big),$$

  where η is a proper shift.
- This is the orbital minimization method (OMM) originally proposed for a linear scaling density matrix method using sparse BLAS. [Mauri, Galli, Car,,Phys. Rev. B, 1993; Ordejon, Drabold, Grumbach, Martin, Phys. Rev. B, 1993]
- All local minima of the OMM are global minima. [Lu and Thicke, JCP, 2017]

Introduction

Optimization model

- Instead, we search for the eigenspace by an unconstrained minimization

$$E = \min_{X \in \mathbb{R}^{N \times n}} E_{\mathrm{omm}}(X) = \min_{X \in \mathbb{R}^{N \times n}} \mathrm{tr}\big((2I_n - X^* X)(X^*(H - \eta I_n)X)\big),$$

  where η is a proper shift.
- This is the orbital minimization method (OMM) originally proposed for a linear scaling density matrix method using sparse BLAS. [Mauri, Galli, Car,,Phys. Rev. B, 1993; Ordejon, Drabold, Grumbach, Martin, Phys. Rev. B, 1993]
- All local minima of the OMM are global minima. [Lu and Thicke, JCP, 2017]
- Only matrix-matrix multiplication and addition are needed.

Introduction

Optimization model

- Instead, we search for the eigenspace by an unconstrained minimization

$$E = \min_{X \in \mathbb{R}^{N\times n}} E_{\text{omm}}(X) = \min_{X \in \mathbb{R}^{N\times n}} \operatorname{tr}\big((2I_n - X^*X)(X^*(H - \eta I_n)X)\big),$$

  where η is a proper shift.
- This is the orbital minimization method (OMM) originally proposed for a linear scaling density matrix method using sparse BLAS. [Mauri, Galli, Car,,Phys. Rev. B, 1993; Ordejon, Drabold, Grumbach, Martin, Phys. Rev. B, 1993]
- All local minima of the OMM are global minima. [Lu and Thicke, JCP, 2017]
- Only matrix-matrix multiplication and addition are needed.
- Good alternative to direct diagonalization
  - Sparse Hamiltonian or Hamiltonian with planewave discretization.
  - In the case of good initial guess.

Barzilai-Borwein method

Goal: $\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x})$, where f is a smooth function

Let $\boldsymbol{g}^{(k)} = \nabla f(\boldsymbol{x}^{(k)})$ and $\boldsymbol{F}^{(k)} = \nabla^2 f(\boldsymbol{x}^{(k)})$.

- **gradient method**: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{g}^{(k)}$
  - choice of α_k: fixed, exact line search, *or* backtracking line search
  - **pros**: simple
  - **cons**: no use of 2nd order information, relatively slow progress
- **Newton's method**: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - (\boldsymbol{F}^{(k)})^{-1}\boldsymbol{g}^{(k)}$
  - **pros**: 2nd-order information, 1-step for quadratic function, fast convergence near solution
  - **cons**: forming and computing $(\boldsymbol{F}^{(k)})^{-1}$ is expensive, need modifications if $\boldsymbol{F}^{(k)} \not\succ 0$
- **BB method**: choose α_k so that $\alpha_k \boldsymbol{g}^{(k)}$ "approximates" $(\boldsymbol{F}^{(k)})^{-1}\boldsymbol{g}^{(k)}$

$$\alpha_k(x^{(k)} - x^{(k-1)}) \approx g^{(k)} - g^{(k-1)}$$

instead of

$$F_k(x^{(k)} - x^{(k-1)}) = g^{(k)} - g^{(k-1)}$$

CG v.s. BB

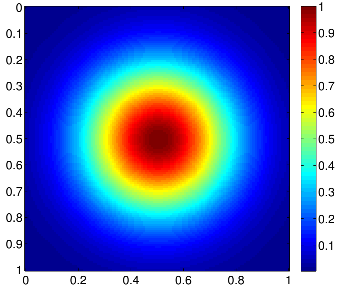
Problem set up

A Hamiltonian matrix H in two dimensions

$$\left(-\frac{\Delta}{2} + V(\mathbf{r})\right) \phi_j(\mathbf{r}) = \epsilon_j \phi_j(\mathbf{r}), \qquad \mathbf{r} \in \ell\mathbb{T}^2 := [0, \ell)^2,$$

with a periodic boundary condition, where $V(\mathbf{r})$ is the potential field, ϵ_j is the orbital energy of the corresponding Kohn-Sham orbital, $\phi_j(\mathbf{r})$.

CG v.s. BB



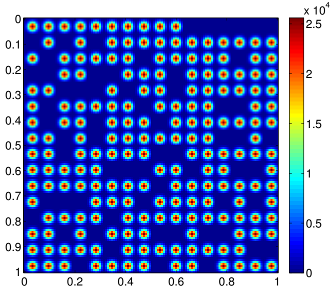


Figure: Left: a Gaussian well. Right: the potential energy operator $V(\mathbf{x})$.

	N	iter	R_{iter}	time	R_{time}	err
PCG	256	4.4e+02	1.8e+00	2.479e-01	1.8e+00	1.0e-05
ABB	256	3.9e+02	1.6e+00	3.690e-01	2.6e+00	1.1e-05
BBSD	256	2.4e+02	1.0e+00	1.407e-01	1.0e+00	1.1e-05
PCG	1024	1.5e+03	1.6e+00	1.347e+01	1.6e+00	1.0e-05
ABB	1024	1.4e+03	1.4e+00	1.193e+01	1.4e+00	1.1e-05
BBSD	1024	9.5e+02	1.0e+00	8.268e+00	1.0e+00	1.1e-05
PCG	4096	2.5e+03	1.8e+00	5.163e+01	1.8e+00	1.1e-05
ABB	4096	1.5e+03	1.1e+00	3.116e+01	1.1e+00	1.1e-05
BBSD	4096	1.4e+03	1.0e+00	2.911e+01	1.0e+00	1.1e-05
PCG	16384	2.8e+03	1.6e+00	1.620e+03	1.7e+00	1.1e-05
ABB	16384	2.5e+03	1.4e+00	1.391e+03	1.5e+00	1.1e-05
BBSD	16384	1.7e+03	1.0e+00	9.442e+02	1.0e+00	1.1e-05

Table: Numerical results for planewave discretization (preconditioned).

	N	iter	R_{iter}	time	R_{time}	err
CG	256	937	2.96	0.2	3.84	1.4e-05
ABB	256	268	0.85	0.0	0.83	1.4e-05
BBSD	256	316	1.00	0.1	1.00	1.3e-05
CG	1024	3386	3.33	3.6	3.73	1.4e-05
ABB	1024	525	0.52	0.5	0.55	1.4e-05
BBSD	1024	1016	1.00	1.0	1.00	1.4e-05
CG	4096	4112	3.56	48.0	3.61	1.5e-05
ABB	4096	516	0.45	6.1	0.46	1.5e-05
BBSD	4096	1154	1.00	13.3	1.00	1.5e-05

Table: Numerical results for finite difference discretization (no preconditioner).

Preconditioning OMM

OMM:

$$E = \min_{X \in \mathbb{R}^{N \times n}} E_{\text{omm}}(X) = \min_{X \in \mathbb{R}^{N \times n}} \operatorname{tr}\big((2I_n - X^* X)(X^*(H - \eta I_n)X)\big),$$

where η is a proper shift.

Lemma

The condition number of the OMM without preconditioner is approximately at least

$$\max \left\{ \frac{\lambda_N - \lambda_1}{\lambda_{n+1} - \lambda_n}, \frac{\lambda_N - \lambda_1}{4(\eta - \lambda_n)}, \frac{4(\eta - \lambda_1)}{\lambda_{n+1} - \lambda_n} \right\},$$

where $\lambda_1 < \lambda_2 < \cdots < \lambda_N$ are eigenvalues of H.

Preconditioning OMM

- **Inverse shifted Laplacian** (adopted in libOMM) is a conventional preconditioner:
$$\mathcal{P} = P \otimes I_n, \quad \text{where,} \quad P = (S - \tau^{-1}\Delta)^{-1},$$
where τ is a parameter setting the scale, and S is the overlapping matrix.
- For planewave discretization, the **TPA** preconditioner is more efficient empirically:
$$P_{kk'} = \delta_{kk'} \frac{27 + 18s + 12s^2 + 8s^3}{27 + 18s + 12s^2 + 8s^3 + 16s^4} \tag{1}$$
with $s = |k|^2/\tau$ and τ a scaling parameter.
- A similar asymptotic behavior **in common**.

Preconditioning OMM

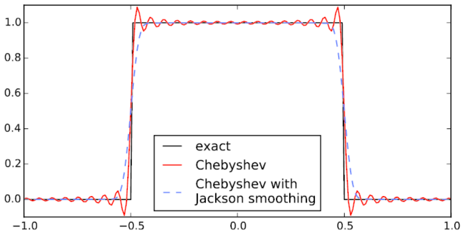
Better idea for preconditioning OMM

- Approximate spectral projector $\mathcal{P}$ corresponding to the n low-lying eigenspace.
- $\mathcal{P}(X)$ can approximate the target subspace.
- Restrict search in the direction $\mathcal{P}(\nabla f(X))$.

Preconditioning OMM

Question: How to construct the approximate spectral projector $\mathcal{P}$?

- Chebyshev polynomials $p(x)$;
- Rational functions $\frac{p(x)}{q(x)}$.



Chebyshev polynomial approximation

Preconditioning OMM

Question: How to construct the approximate spectral projector $\mathcal{P}$?

- Chebyshev polynomials $p(x)$;
- Rational functions $\frac{p(x)}{q(x)}$.

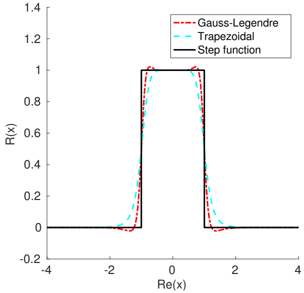
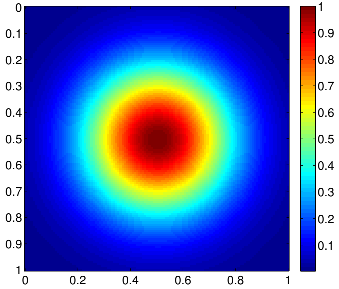


Figure: Rational functions by discretizing the contour integral.

Numerical results



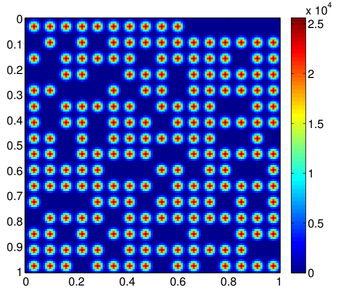


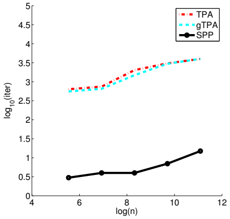
Figure: Left: a Gaussian well. Right: the potential energy operator $V(\mathbf{x})$.

Numerical results

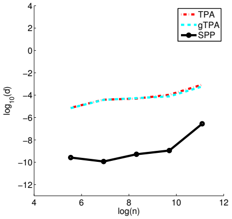
Main parameters

- Number of grid points on the contour: 30;
- Iterative solver: a relative tolerance 10^{-5} and a maximum iteration number 75;
- OMM: convergence tolerance 10^{-13} and the maximum iteration number 4000;

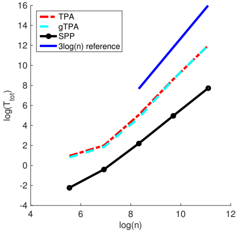
Numerical results



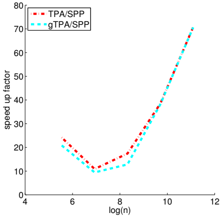
(a) number of iterations



(b) log(error)



(c) runtime scaling



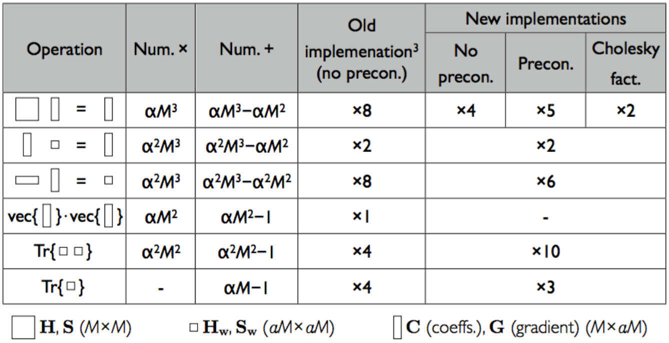
(d) speed-up factor

OMM

$$\min_{X\in\mathbb{R}^{N\times n}} \operatorname{tr}\big((2I_n - X^*X)(X^*(H-\eta I_n)X)\big)$$

requires only matrix-matrix multiplication and addition.

- Reduce the number of iterations;
- Improve the speed per iteration, especially the matrix-matrix multiplication.



Operation	Num. ×	Num. +	Old implementation[3] (no precon.)	New implementations		
				No precon.	Precon.	Cholesky fact.
□ ▯ = ▯	αM^3	$\alpha M^3-\alpha M^2$	×8	×4	×5	×2
▯ ▫ = ▯	$\alpha^2 M^3$	$\alpha^2 M^3-\alpha M^2$	×2	×2		
▭ ▯ = ▫	$\alpha^2 M^3$	$\alpha^2 M^3-\alpha^2 M^2$	×8	×6		
vec{▯}·vec{▯}	αM^2	αM^2-1	×1	-		
Tr{▫ ▫}	$\alpha^2 M^2$	$\alpha^2 M^2-1$	×4	×10		
Tr{▫}	-	$\alpha M-1$	×4	×3		

□ **H**, **S** ($M\times M$)   ▫ $\mathbf{H}_w$, $\mathbf{S}_w$ ($aM\times aM$)   ▯ **C** (coeffs.), **G** (gradient) ($M\times aM$)

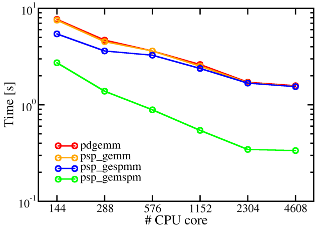
Table: Linear algebra operations used in an OMM line search. Listed are the type of operation, the number of multiplications and additions it requires, and the number of times it is used in the various implementations. M is the number of basis orbitals, and α the proportion of occupied-to-unoccupied states ($aM = N/2$)

PSP: parallel and sparse BLAS

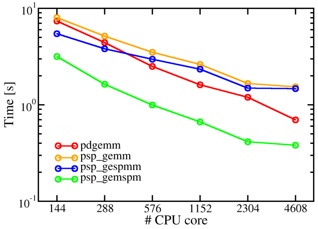
- The pspBLAS is an extensible distributed-memory parallel library offering a basic set of linear algebra primitives.
- It achieves scalability and load balance via different distribution strategies: 1D, 2D block (cyclic) distribution[1].
- Rountines for sparse data types includes (sparse) matrix (sparse) vector multiplication, (sparse) matrix (sparse) matrix multiplication, etc.
- Supports several sparse format, e.g. COO, CSC, and CSR[2]
- Similar user habits with Scalapack

[1] 1.5D and 3D under development.
[2] CSR will be implemented in C++.

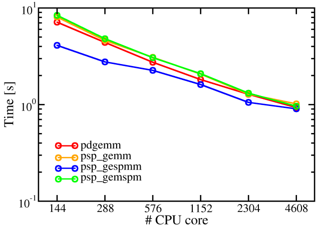
Comparison of Scalapack and PSP



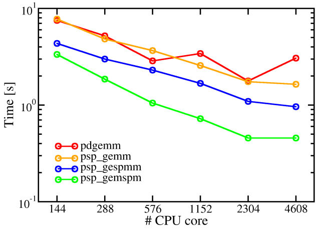
(n,n)



(n,t)



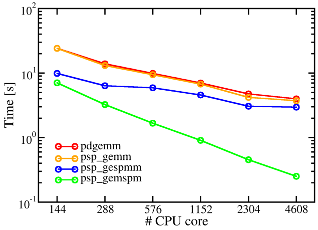
(t,n)



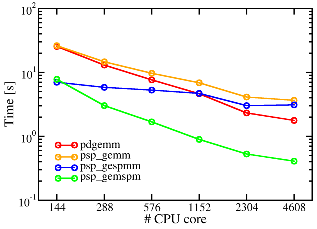
(t,t)

Figure: Matrix size: 2000 by 2000. 95% zero entries.

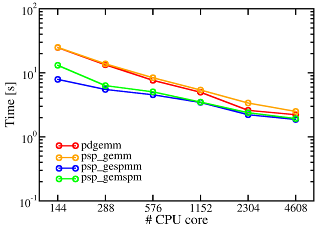
Comparison of Scalapack and PSP



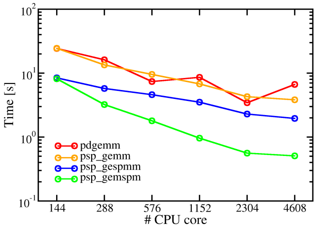
(n,n)



(n,t)



(t,n)

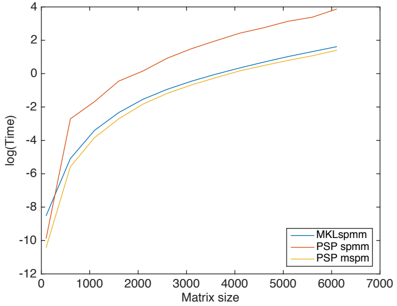


(t,t)

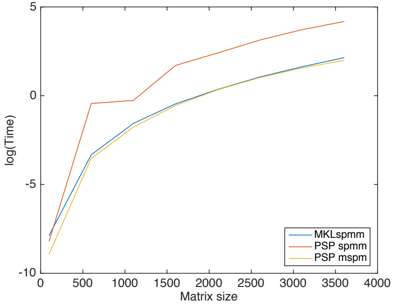
Figure: Matrix size: 3000 by 3000. 99% zero entries.

Comparison of sequential sparse BLAS

Optimization at the cache level



90% zeros

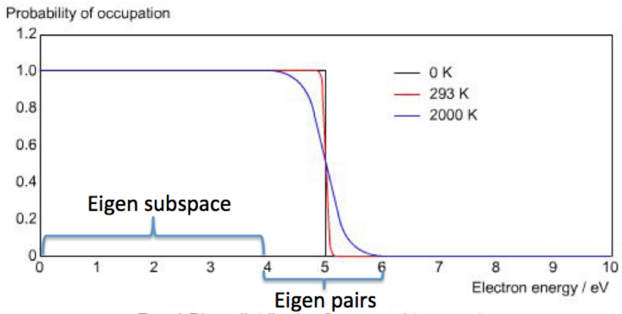


99% zeros

Figure: Comparison of Intel MKL and PSP, sparse matrix times dense matrix, (n,n).

OMM for finite temperature

$$f(E) = \frac{1}{e^{(E-E_F)/kT} + 1}$$



Fermi-Dirac distribution for several temperatures

Interior eigen solver

Main subproblem

Given a hermitian matrix $A \in \mathbb{R}^{N \times N}$ and a spectrum range (a, b), find the eigenpairs in (a, b).

Possible solution

- Construct a spectral projector of A, denoted as $f(A)$, i.e.

  $$f(A) = PP^*,$$

  where P consists of all the eigenvectors corresponding to the eigenvalues in (a, b).
- $f(A)G$ gives the column space of P, where $G \in \mathbb{R}^{N \times n}$ is a random matrix:

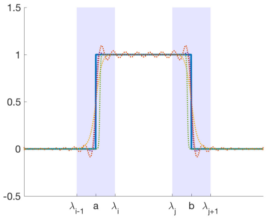
  $$f(A)G = P(P^* G).$$

Interior eigen solver

Approximate spectral projectors:

- Chebyshev polynomial $f(x)$, $f(A)G = \sum_{n=0}^{r} a_n A^n G$;
- Rational function $f(x) = \frac{\sum_{n=0}^{r} a_n x^n}{\sum_{n=0}^{s} b_n x^n}$,

$$f(A)G = (\sum_{n=0}^{r} a_n A^n)(\sum_{n=0}^{s} b_n A^n)^{-1} G.$$

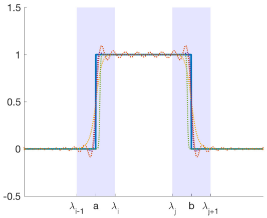


Interior eigen solver

Approximate spectral projectors:

- Chebyshev polynomial $f(x)$, $f(A)G = \sum_{n=0}^{r} a_n A^n G$;
- Rational function $f(x) = \frac{\sum_{n=0}^{r} a_n x^n}{\sum_{n=0}^{s} b_n x^n}$,

$$f(A)G = (\sum_{n=0}^{r} a_n A^n)(\sum_{n=0}^{s} b_n A^n)^{-1} G.$$



Our contribution (Li, **Y.**, preprint, 2017)

- Optimal approximation to the rectangular function for the fixed order (r, s).
- Fast and stable algorithm for computing $f(A)G$ for high orders.

Numerical results

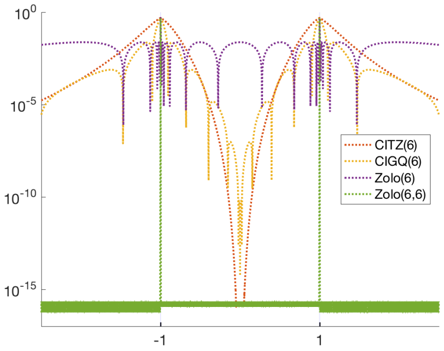


Figure: Step function approximation error.

Thank you!