# New developments of LOBPCG for large-scale nonlinear eigenvalue problems

Fei Xue

**University of Louisiana at Lafayette**
**Department of Mathematics**
Supported by NSF-1115520

TeXAMP 2013
October 26, 2013
Rice University, Houston, Texas

# Introduction

## Generalized algebraic eigenvalue problem

Find the eigenpair $(\lambda, v)$ of $Av = \lambda Bv$,
where $\lambda$ is the **smallest** value, and $A, B \in \mathbb{C}^{n\times n}$ are large and sparse Hermitian positive definite (HPD) matrices.

## Inverse power method

Start with $x_0$ with $\|x_0\|_2 = 1$

For $k = 0, 1, \ldots,$ until convergence

$x_{k+1} = A^{-1}Bx_k$;

$x_{k+1} = x_{k+1}/\|x_{k+1}\|_2$;

End For

$$\rho_m = \frac{\langle x_m, Ax_m\rangle}{\langle x_m, Bx_m\rangle} \text{ (the Rayleigh quotient of } x_m)$$

# Introduction (Cont'd)

## Inverse power method (modified but equivalent)

Start with $x_0$ with $\|x_0\|_2 = 1$

For $k = 0, 1, \ldots,$ until convergence

$x_{k+1} = x_k - A^{-1}(Ax_k - \rho_k Bx_k)$; (i.e., $x_{k+1} = \rho_k A^{-1} Bx_k$)

$x_{k+1} = x_{k+1}/\|x_{k+1}\|_2$;

End For

## Comments

- $-A^{-1}(Ax_k - \rho_k Bx_k)$ is a **correction** of $x_k$
- For $A$ large and sparse, it is expensive or impractical to compute $A^{-1}v$ by solving $Ax = v$
- Instead, construct a **preconditioner** $M \approx A$ such that computing $M^{-1}v$ is much less expensive

# Introduction (Cont'd)

## Preconditioned steepest descent (PSD)

Start with $x_0$ with $\|x_0\|_2 = 1$

For $k = 0, 1, \ldots,$ until convergence

$x_{k+1} = x_k + \alpha_k M^{-1}(Ax_k - \rho_k Bx_k)$; where $\alpha_k$ is chosen such that $\rho_{k+1} = \frac{\langle x_{k+1}, Ax_{k+1}\rangle}{\langle x_{k+1}, Bx_{k+1}\rangle}$ is minimal for all $\alpha_k \in \mathbb{C}$

$x_{k+1} = x_{k+1}/\|x_{k+1}\|_2$;

End For

## Comments

- For $Av = \lambda Bv$ with HPD $B$, the Courant-Fischer min-max theorem (variational theorem) applies, namely, $\lambda_k = \max\{\min\{\rho(x) : x \in S, \dim(S) = n - k + 1\}\}$.
- PSD = application of the steepest descent method for **minimization** of the Rayleigh quotient $\rho$

# Introduction (Cont'd)

## SD vs. CG for unconstrained minimization

- It is well known that SD converges much slower than CG.
- CG constructs a three-term recurrence involving $x_k$, $p_k$ (the latest search direction) and $g_k$ (the current gradient). $p_{k+1}$ is some linear combination of $p_k$ and $g_k$.
- $g_k$ is the "residual vector" of the system of equations
  - For SPD linear systems, $f(x_k) = \frac{1}{2}x_k^H A x_k - b^H x_k$, and $g_k = \nabla f(x_k) = Ax_k - b$.
  - For Hermitian eigenproblems, $f(x_k) = \frac{\langle x_k, Ax_k\rangle}{\langle x_k, Bx_k\rangle}$ and $g_k = \nabla f(x_k) = \frac{2}{x_k^H B x_k}(Ax_k - \rho_k Bx_k)$.
- The use of preconditioner $M$ and search direction $p_k$ are critical to accelerate convergence.

# How does CG minimize the Rayleigh quotient?

## PCG-like methods for eigenvalue problems

- Use PCG-like methods to compute the smallest (left-most) eigenvalue $\lambda_1$ ($\leq \lambda_2 \ldots \leq \lambda_n$)
- Locally optimal PCG (LOPCG) projects $(A, B)$ onto $\mathrm{span}\{x_k, g_k, p_k\}$ and solves the $3 \times 3$ eigenproblem for the minimal Ritz value.
- Alternatively, PCG forms $g_{k+1}$ as a linear combination of $p_k$ and $g_k$, then projects $(A, B)$ onto $\mathrm{span}\{x_k, g_{k+1}\}$ and solves $2 \times 2$ eigenproblem for the the minimal Ritz value.
- The minimal Ritz values $=$ the minimization of $\rho_{k+1}$ for $x_{k+1} = x_k + \alpha_k g_k + \beta_k p_k$ over all $\alpha_k, \beta_k \in \mathbb{C}$ (LOPCG) or for $x_{k+1} = x_k + \gamma_k p_{k+1}$ over all $\gamma_k \in \mathbb{C}$ (PCG).

# Block variants

## LOBPCG and BPCG

- Use the block variants of LOPCG or PCG to compute the $m$ smallest eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$.
- LOBPCG: let $X_k \in \mathbb{C}^{n\times m}$, $Q_k = (X_k^H A X_k)(X_k^H B X_k)^{-1}$; project $(A, B)$ onto $\mathrm{span}\{X_k, M^{-1}(AX_k - BX_kQ_k), P_k\}$, and find the $m$ smallest Ritz values
- BPCG: form a linear combination of $M^{-1}(AX_k - BX_kQ_k)$ and $P_k$ as $P_{k+1}$; project $(A, B)$ onto $\mathrm{span}\{X_k, P_{k+1}\}$, and find the $m$ smallest Ritz values.
- LO(B)PCG needs fewer iterations than (B)PCG to converge; (B)PCG requires less arithmetic and storage cost per iteration

# Hermitian nonlinear eigenvalue problems

## Problem description

- $T(\lambda)v = 0$, where $T : \mathbb{R} \to \mathbb{C}^{n\times n}$ depends continuously and nonlinearly (in general) on the real variable.
- $Av = \lambda Bv \iff T(\lambda)v = 0$ where $T(\lambda) = \lambda B - A$.
- Assume that $a < b$ are such that $T(a) > 0$ and $T(b) < 0$; assume in addition that $\lambda_i(\mu)$, the $i$-th eigenvalue of $T(\mu)$, has exactly one zero on $(a, b)$ for all $1 \leq i \leq n$.
- Let the Rayleigh functional $\rho(x) : \mathbb{C}^n \to \mathbb{R}$ be such that $x^H T(\rho(x))x = 0$. With the above assumption, for $\forall x \in \mathbb{C}^n$, there exists exactly one $\rho(x) \in (a, b)$.
- The min-max principle also holds in this case; $\lambda_k = \max\{\min\{\rho(x) : x \in S, \dim(S) = n - k + 1\}\} \in (a, b)$

# Hermitian nonlinear eigenvalue problems

## Problem description

- Thanks to the min-max principle, LOBPCG and BPCG can be applied to find the smallest $m$ eigenvalues on $(a, b)$
- Let $X_k e_j$ be the $j$-th column of $X_k$, and $\rho(X_k e_j)$ be the corresponding Rayleigh functional value
  Let $U = \left[X_k \;\; M^{-1}T(\mathrm{diag}(\rho(X_k e_1), \ldots, \rho(X_k e_m)))X_k \;\; P_k\right]$.
  LOBPCG projects $T(\cdot)$ onto $U$ and solves the $3m \times 3m$ eigenproblem for the $m$ smallest Ritz values and Ritz vectors $W_k$. Update $X_{k+1} = UW_k$, $P_{k+1} = X_{k+1} - X_k$
- BPCG constructs $P_{k+1}$ as a linear combination of $M^{-1}T(\mathrm{diag}(\rho(X_k e_1), \ldots, \rho(X_k e_m)))X_k$ and $P_k$, projects $T(\cdot)$ onto $U = [X_k \;\; P_{k+1}]$ and solves the $2m \times 2m$ eigenproblem.

# Hermitian nonlinear eigenvalue problems

## Memory cost and convergence rate

- LOBPCG and BPCG require a minimum storage for $4m$ and $3m$ vectors; expensive for large $m$ (e.g., $m \approx 100$ or above)
- Use LOPCG or PCG + deflation of converged eigenvectors instead, which require only a storage of $m + \mathcal{O}(1)$ vectors
- With the same preconditioner, LOPCG or PCG with deflation converges much slower than the block variants for large $m$

## Indefinite preconditioner

- To accelerate the convergence of LOPCG and PCG with deflation, use a **variable and indefinite** preconditioner
- For example, use incomplete LDL decomposition of $T(\sigma)$ where $\sigma$ is near the desired eigenvalue being computed; update the preconditioned when necessary.

# Numerical experiments

## Problem 1: An artificial problem

- $T(\lambda)v = \left(e^{\lambda/\sqrt{\pi}}A + \sin(\lambda/4)B - 12C\right)v = 0$, where $A = \texttt{delsq}(\texttt{numgrid}(\texttt{128},'\texttt{S}'))$, $B = I_n$,

$$C = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & 2 & -1 \\ & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n\times n}.\ n = 15876.$$

- Lowest eigenvalue $\lambda_1 = -3.0918$, highest $\lambda_n = 5.3588$.

# Numerical results

## Problem 1: An artificial problem

- Compute the highest 30 eigenvalues to a residual norm $10^{-10}$
- Incomplete LDL preconditioner with drop tolerance $10^{-3}$
- Update preconditioner once 10 more eigenpairs have converged

| Method | Preconditioned MVPs | CPU time | Memory cost |
|---|---|---|---|
| PCG+Deflation | 564 | 262.6s | $30 + \mathcal{O}(p)$ |
| LOPCG+Deflation | 535 | 377.5s | $30 + \mathcal{O}(p)$ |
| BPCG | 372 | 157.0s | $90 + \mathcal{O}(p)$ |
| LOBPCG | 313 | 164.2s | $120 + \mathcal{O}(p)$ |

Table: Performance of four PCG-like methods for Problem 1

# Numerical experiments

## Problem 2: Vibration of a string

- A rational eigenvalue problem arising in the FE discretization of a boundary problem describing the vibration of a string with mass $m$ attached by an elastic spring of stiffness $k$.
- $R(\lambda)v = \left(A - \lambda B + \frac{\lambda}{\lambda - \sigma} C\right) v = 0$, where

$$A = \frac{1}{h}\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ \ddots & 2 & -1 & \\ & & -1 & -1 \end{bmatrix}, \quad B = \frac{6}{h}\begin{bmatrix} 4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & 4 & 1 \\ & & 1 & 2 \end{bmatrix},$$

$C = k e_n e_n^T \in \mathbb{R}^{n \times n}$, $n = 10000$, $\sigma = k/m$, $h = 1/n$.

- Lowest eigenvalue $\lambda_1 = 4.4820$, highest $\lambda_n = 1.2000 \times 10^9$.

# Numerical results

## Problem 2: vibration of a string

- Compute the lowest 50 eigenvalues to a residual norm $10^{-10}$
- The matrices are tridiagonal; LDL preconditioner can be used
- Update preconditioner once 10 more eigenpairs have converged

| Method | Preconditioned MVPs | CPU time | Memory cost |
|---|---|---|---|
| PCG+Deflation | 702 | 376.5s | $50 + \mathcal{O}(p)$ |
| LOPCG+Deflation | 626 | 337.0s | $50 + \mathcal{O}(p)$ |
| BPCG | 353 | 211.9s | $150 + \mathcal{O}(p)$ |
| LOBPCG | 282 | 173.7s | $200 + \mathcal{O}(p)$ |

Table: Performance of four PCG-like methods for Problem 2

# Conclusions

## Brief summary and future work

- We studied several variants of preconditioned conjugate gradient methods to solve nonlinear Hermitian eigenvalue problems for extreme eigenvalues.
- Each variant has its strength and weakness (memory vs. CPU time cost); overall performance is problem-dependent
- Orthogonalization dominates the computation; not suitable for a large number of eigenvalues
- Efficient methods based on **local orthogonalization** for a large number of interior eigenvalues under development; results very promising ($n \approx$ 1M, 1000 eigenvalues).