Overview of Anasazi and its newest eigensolver, TraceMin [1]

Alicia Klinvex
aklinvex@purdue.edu

Purdue University
Department of Computer Science

Sandia National Laboratories[2]

October 29, 2014

[1] Approved for unlimited release. SAND # 2014-16137PE.

[2] Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

What is Anasazi?

- A package for solving large sparse generalized eigenvalue problems

$$Ax = \lambda Bx$$

- Computes a subset of eigenpairs, either the largest or smallest
- Has impressive scalability
- Supports very large problems
- Effective on many different architectures
- Relies on operators rather than matrices

How does Anasazi relate to the rest of Trilinos?

- Uses Teuchos (RCP, LAPACK wrappers)
- Can use the operators and multivectors of Epetra/Tpetra
- Can use the preconditioners of Ifpack/Ifpack2
- Can use the linear solvers of Amesos, Belos, and AztecOO
- Can (potentially) be used by Zoltan to reorder a graph

Which eigensolvers are available in Anasazi?

Solver	Gen.	Prec.	Complex Herm.	Non-Herm.	Interior
BKS	yes	yes*	yes	yes	yes
Block-Davidson	yes	yes	yes	no	yes
Generalized-Davidson	yes	yes	no	yes	yes
LOBPCG	yes	yes	yes	no	no
RTR	yes	yes	yes	no	no
TraceMin family	yes	yes	no*	no*	yes

- BKS - Block Krylov Schur
- LOBPCG - Locally Optimal Block Preconditioned Conjugate Gradient
- RTR - Riemannian Trust-Region method

What's new in Anasazi?

- We now have an Anasazi user manual "Installing the Anasazi Eigensolver Package with Application to Some Graph Eigenvalue Problems," which can be found on Rich Lehoucq's page
- We have expanded our examples to include Tpetra examples, matrix-free examples, and computation of the Fiedler vector
- We have a new class of solvers referred to as the "TraceMin family" which includes...
  - TraceMin (1982)
  - Jacobi-Davidson (1996)
  - TraceMin-Davidson (2000)

An important note about the TraceMin family of solvers...

These solvers have been marked "experimental." Please report any unexpected behavior you may encounter.

What kinds of problems can TraceMin solve?

$$Ax = \lambda Bx$$

- A is real, symmetric
- B is symmetric positive semidefinite (SPD)
- TraceMin will converge to the smallest eigenpairs

How does TraceMin work?

- Based on the following observation derived from the Courant-Fischer theorem

$$\min_{Y^T B Y = I_p} trace\left(Y^T A Y\right) = \sum_{i=1}^{p} \lambda_i$$

- We may solve the following constrained minimization problem to generate the next iterate from the current one

$$\min_{Y_k^T B \Delta_k = 0} trace\left(\left(Y_k - \Delta_k\right)^T A \left(Y_k - \Delta_k\right)\right)$$

What is Y?

The multivector Y which minimizes that trace is the set of eigenvectors corresponding to the smallest eigenvalues

How do we solve that constrained minimization problem?

- If A is SPD, then solving this problem:

$$\min_{Y_k^T B \Delta_k = 0} trace\left((Y_k - \Delta_k)^T A (Y_k - \Delta_k) \right)$$

gives rise to the following saddle point problem

$$\begin{bmatrix} A & BY \\ Y^T B & 0 \end{bmatrix} \begin{bmatrix} \Delta \\ L \end{bmatrix} = \begin{bmatrix} AY \\ 0 \end{bmatrix}$$

What makes TraceMin so special?

- The equations

$$\begin{bmatrix} A & BY \\ Y^T B & 0 \end{bmatrix} \begin{bmatrix} \Delta \\ L \end{bmatrix} = \begin{bmatrix} AY \\ 0 \end{bmatrix}$$

  need not be solved to a low relative residual
- We can use a direct or iterative method to solve this problem and it won't affect TraceMin's global convergence

What do we mean by global convergence?

- Convergence to the *desired* eigenpairs is not dependent on initial input
- Algorithms with *local convergence* may converge to other eigenpairs instead

How do we solve the saddle point problem iteratively?

- Use a projected Krylov method to solve

$$(PAP)\,\Delta = PAY \text{ with } P = I - BY\left(Y^T B^2 Y\right)^{-1} Y^T B$$

  which guarantees that Δ will be B-orthogonal to Y
- Using the approximate Schur-complement

$$S = -Y^T BA^{-1}BY \text{ so } \Delta = A^{-1}BYS^{-1}$$

- Use a block preconditioner such as

$$M = \begin{bmatrix} \hat{A} & 0 \\ 0 & Y^T B\hat{A}^{-1}BY \end{bmatrix}$$

  on the entire saddle point problem

TraceMin algorithm

1: Choose an initial V
2: **repeat**
3: Form a section from V

$$Y^T AY = \Sigma \text{ and } Y^T BY = I$$

Note that Y approximates the eigenvectors
Σ is the diagonal matrix of approximate eigenvalues
4: Compute the residual $R = AY - BY\Sigma$
5: Solve the saddle point problems

$$\begin{bmatrix} A & BY \\ Y^T B & 0 \end{bmatrix} \begin{bmatrix} \Delta \\ L \end{bmatrix} = \begin{bmatrix} AY \\ 0 \end{bmatrix}$$

6: Update $V = Y - \Delta$
7: **until** converged

How fast is TraceMin?

Bounds

Global linear convergence, bounded by

$$\frac{\lambda_i}{\lambda_{s+1}}$$

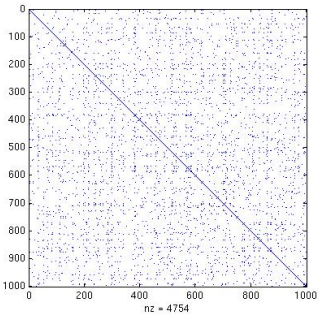
- Bad if eigenvalues are poorly separated and far from the origin
- Origin shifts can lead to faster convergence if the shifts approximate the eigenvalues
- Instead of solving $Ax_i = \lambda_i Bx_i$ with convergence rate λ_i/λ_{s+1}, solve $(A - \sigma_i B)\, x_i = (\lambda_i - \sigma_i)\, Bx_i$ with convergence rate $(\lambda_i - \sigma_i) \,/\, (\lambda_{s+1} - \sigma_i)$

Subspace dimension

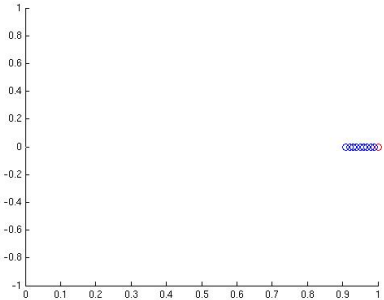
We generally choose $s = 2p$, where p is the desired number of eigenpairs

How important are origin shifts?

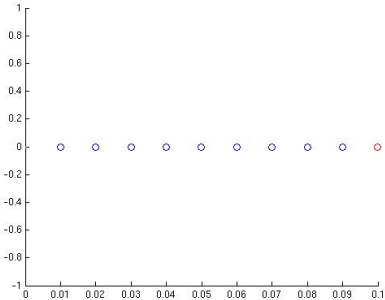
- A synthetic test matrix with 1000 rows and 4754 nonzeros
- Eigenvalues equally spaced in the interval [0.91,10.9]
- Seeking: 4 smallest eigenpairs
- Subspace: 9 vectors
- Tolerance: 10^{-6}
- We will try TraceMin with and without a shift of 0.9



How important are origin shifts?

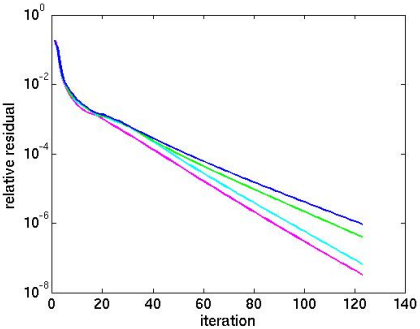


A bad eigenvalue distribution (without shift)

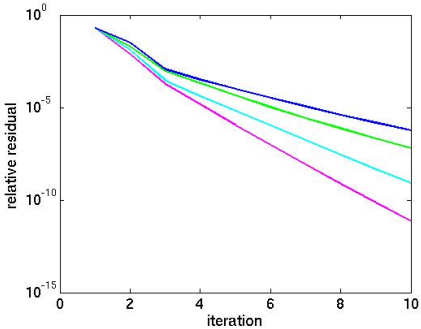


A better eigenvalue distribution (with shift)

How important are origin shifts?



Without shift
120 iterations



With shift
10 iterations

- We could have used multiple shifts instead of one for even better performance.

How do we choose the origin shifts?

- After a few iterations, we have a reasonable approximation of the desired eigenvalues
- Jacobi-Davidson uses the Ritz values, which are overestimates
- TraceMin and TraceMin-Davidson adjust the Ritz values by the residuals in an effort to keep the global convergence
- Our solver managers allow you to specify how the shifts are chosen and when it's appropriate to shift

Difference between the three TraceMin solvers

Solver	Subspace Dimension	Choice of Shift
TraceMin (1982)	constant	conservative
Jacobi-Davidson (1996)	expanding	aggressive
TraceMin-Davidson (2000)	expanding	conservative

How do we handle expanding subspaces

- Instead of $V = Y - \Delta$ at each iteration, we have $V = [V\ \Delta]$
- Upon restart, keep the most important part of the subspace, discard the rest.

How do our trace-minimization solvers compare to SLEPc's Jacobi-Davidson?

- Their Jacobi-Davidson implementation is very similar to our TraceMin-Davidson.
  They only shift after the residuals become very small.
- Because SLEPc is built on top of PETSc, it performs operations with single vectors rather than blocks.
- Our operations use blocks whenever possible.
- They do not have multiple options for solving the saddle point problem

Why do we want to use blocks?

This is a parallel program, and it can be better to perform one large communication rather than several small ones.

"Hidden" spectral transformations

- TraceMin obtains the smallest eigenvalues and corresponding eigenvectors of the symmetric generalized eigenvalue problem it is given.
- If we want to find other eigenpairs, such as the largest, we must perform a spectral transformation

An important observation

- If (λ_1, x_1) is the smallest eigenpair of $Bx = \lambda Ax$, then $\left(\frac{1}{\lambda_1}, x_1\right)$ is the largest eigenpair of $Ax = \lambda Bx$.
- Instead of giving TraceMin the problem $Ax = \lambda Bx$, we will give it $Bx = \sigma Ax$
- If this is a standard eigenvalue problem ($B = I$), we NEVER need to solve a linear system.

"Hidden" spectral transformations

How to request the smallest eigenpairs using Anasazi's block Krylov-Schur (using a spectral transformation)

- Create a linear solver
- Wrap the linear solver in an operator object
- Create a status test object and give it to the BKS solver manager
- Request the *largest* eigenpairs using the parameter which
- Call the BKS solve routine
- Invert the Ritz values that were returned

How to request the largest eigenpairs using Anasazi's TraceMin (using a spectral transformation)

- Request the largest eigenpairs using the parameter which
- Call the TraceMin solve routine

Our goal is to make the spectral transformation invisible to the user by hiding it in the solver manager

Future plans - obtaining a large number of eigenpairs

- In some applications, it is important to obtain all eigenvalues in a given interval, along with their associated eigenvectors
- We may want to compute 1000 eigenpairs or more
- We can recursively split up the global interval of interest into many subintervals (like adaptive quadrature)
- This process is called *multisectioning*

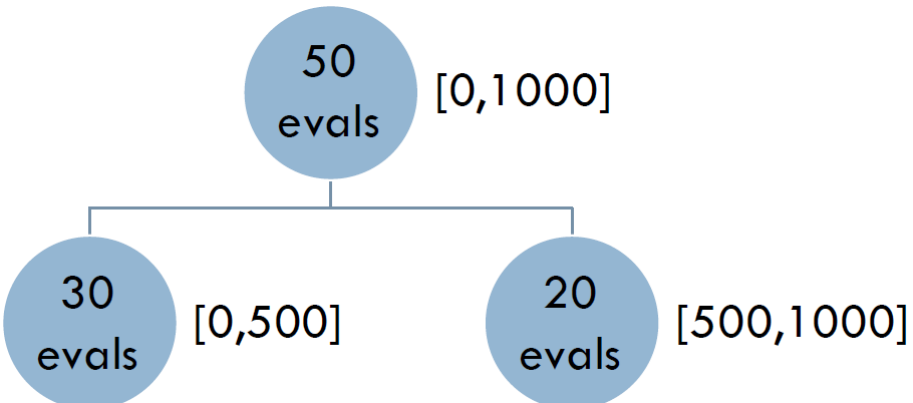
How do we know how many eigenvalues are in an interval?

- Compute the inertia using a sparse factorization package $(A - \sigma B) = LDL^T$
- Number of positive elements in D is the number of positive eigenvalues of $Ax = \lambda Bx$

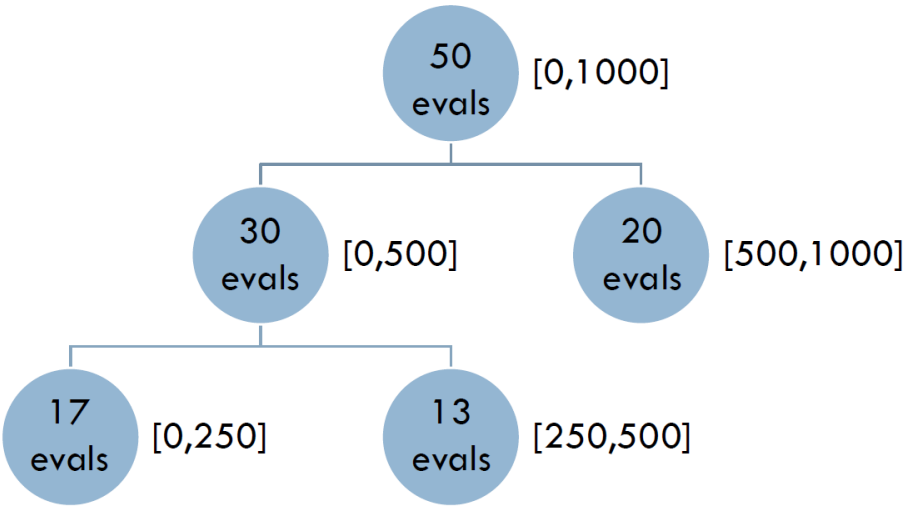
A multisectioning demonstration

[0,1000]

A multisectioning demonstration



A multisectioning demonstration



How does this compare to existing packages?

- We have an existing implementation of TraceMin-multisectioning outside of Trilinos
  - Small groups of MPI processes work on the subintervals independently
  - Dynamic load balancing
- FEAST (Eric Polizzi's eigensolver based on contour integration) is capable of multisectioning, but it requires the user to subdivide the large global interval themselves
- SLEPc's Krylov-Schur implementation is capable of multisectioning, but all processes work on one subinterval at a time...together

What's wrong with having all the processes work together on one subinterval?

SLEPC's Krylov-Schur calls a direct solver

Acknowledgements

- This work was supported in part by the Army Research Office, ARO grant number 7W911NF-11-1-0401.
- Ahmed Sameh and I are very grateful to Intel Corporation for allowing us the use of their computing resources.
- I would like to thank the Trilinos team at Sandia National Labs for all their help with the Trilinos package and the invitation to TUG.