Implementing eigenvalue problem solvers for sparse matrices

Siddhant Katyan

April 9, 2019

Abstract

In this project, our goal is to update IterativeSolvers.jl package with iterative methods for the (generalized) eigenvalue problem $Ax = \lambda Bx$. We want to solve eigenvalue problem by posing it as an optimization problem on Manifold, so we will implement a truncated-CG style method for symmetric generalized eigenvalue problems as a solver. The method has excellent global convergence properties and its local rate of convergence is superlinear. It is based on a constrained truncated-CG trust-region strategy to optimize the Rayleigh quotient, in the framework of a recently-proposed trust-region scheme on Riemannian manifolds [1]. The proposed method outperforms Krylov Subscape methods - (Lanczos, Arnoldi) and Jacobi Davidson for the generalized eigenproblem.

Contents

1 Introduction

The aim is to implement a native Julia solver for the eigenvalue problem $Ax = \lambda x$. In particular, the aim is to solve this problem for large and sparse matrices which would replace ARPACK's eigs function. This helps reduce the dependency of Julia language on ARPACK to solve large scale eigenvalue problems.

I will provide benchmarks comparing the performance of the native eigs versus ARPACK's eigs. The task at first is to get the method into a separate package. However, later we can finally merge it to IterativeSolvers.jl.

2 Project Details

Riemannian Trust-Region (RTR) method can be applied to the problem of computing extreme eigenpairs of a matrix pencil, with strong global convergence and local convergence properties. We will implement the Implicit Riemannian Trust-Region (IRTR) method for extreme eigenpair computation, which seeks to overcome the inherent inefficiencies of an explicit trust-region mechanism.

Consider $n \times n$ symmetric matrices A and B, with B positive definite. The generalized eigenvalue problem

$$Ax = \lambda Bx$$

is known to admit n real eigenvalues $\lambda_1 \leq ... \leq \lambda_n$, along with associated B-orthonormal eigenvectors $v_1, ..., v_n$. We seek here to compute the p leftmost eigenvectors of the pencil (A, B). The leftmost eigenspace, $\mathcal{U} = \text{colsp}(v_1, ..., v_p)$ of (A, B) is the column space of any minimizer of the generalized Rayleigh quotient

$$f : R_*^{n\times p} \to R : Y \mapsto \text{ trace } ((Y^T BY)^{-1}(Y^T AY)),$$

where $R_*^{n\times p}$ denotes the set of full-rank $n \times p$ matrices.

Rayleigh quotient depends only on colsp(Y), so that f induces a real-valued function on the set of p-dimensional subspaces of R^n. (This set is known as the Grassmann manifold, which can be endowed with a Riemannian structure [2]. The RTR method iteratively computes the minimizer of f by (approximately) minimzing successive models of f.

Let Y be a full-rank, $n \times p$ matrix. We desire a correction S of Y such that $f(Y + S) < f(Y)$. A difficulty is that corrections of Y that do not modify its column space do not affect the value of the cost function. To address this, we require S to satisfy some complementarity condition with respect to the space $\mathcal{V}_y := \{YM : M \in R^{p\times p}\}$. We impose complementarity via B-orthogonality, namely $S \in \mathcal{H}_Y$ where

$$\mathcal{H}_Y = \{Z \in R^{n\times p} : Y^T BZ = 0\}.$$

Consequently, the task is to minimize the function

$$\hat{f}_Y(S) := \text{trace}\Big(((\text{Y} + \text{S})^T B(Y + S))^{-1}((Y + S)^T A(Y + S))\Big), \quad S \in \mathcal{H}_Y$$

All the below modules will be added as functions in IterativeSolvers.jl and implemented in Julia.

2.1 Modules

- **Module 1: (Preconditioned Truncated CG (IRTR))**

  - *Data*: A, B *symmetric*, B *positive definite*, $\rho' \in (0{,}1)$, *preconditioner* M
  - *Input*: *Iterate* y, $y^T By = 1$
  - *Set* $s_0 = 0$, $r_0 = \nabla \hat{f}_y$, $z_0 = M^{-1} r_0$, $d_0 = -z_0$
  - **for** $j = 0, 1, 2, ...$

```
      Check $\kappa/\theta$ stopping criteria
      if $\|r_j\|_2 \leq \|r_0\|_2 \min\{\kappa, \|r_0\|_2^\theta\}$
          return $s_j$
      Check curvature of current search direction
      if $d_j^T H_y[d_j] \leq 0$
          Compute $\tau$ such that $s = s_j + \tau d_j$ satisfies $\boxed{\rho_y(s) = \rho'}$
          return $s$
      Set $\alpha_j = (z_j^T r_j)/(d_j^T H_y[d_j])$
      Generate next inner iterate
      Set $s_{j+1} = s_j + \alpha_j d_j$
      Check implicit trust-region
      if $\boxed{\rho_y(s_{j+1}) < \rho'}$
          Compute $\tau \leq 0$ such that $s = s_j + \tau d_j$ satisfies $\boxed{\rho_y(s) = \rho'}$
          return s
      Use CG recurrences to update residual and search direction
      Set $r_{j+1} = r_j + \alpha_j H_y[d_j]$
      Set $z_{j+1} = M^{-1} r_{j+1}$
      Set $\beta_{j+1} = (z_{j+1}^T r_{j+1})/(z_j^T r_j)$
      Set $d_{j+1} = -z_{j+1} + \beta_{j+1} d_j$
      Check outer stopping criterion
      $\boxed{\text{Compute } \|\nabla \hat{f}_{y+s_{j+1}}\|_2 \text{ and test}}$
  end for
```

  - The above iteration returns an update vector s_j which is guaranteed to lie inside of the ρ-based trust-region. The result is that the ρ value of the new iterate need not be explicitly computed, the new iterate can be automatically accepted, with an update vector constrained by model fidelity instead of directly chosen trust-region radius based on the performance of the last iterate. An updated *outer iteration* is presented in **Algorithm 2**, which also features an optional *subspace acceleration enhancement as in Jacobi-Davidson iteration method [3].*

- **Module 2: (Implicit Riemannian Trust-Region Algorithm)**
  - *Data: A,B symmetric, B positive definite,* $\rho' \in (0,1)$
  - *Input: Initial subspace* $\mathcal{W}_0$
  - **for** *k=0,1,2,...*

```
        Model-based Minimization
        Generate $y_k$ using a Rayleigh-Ritz procedure on $\mathcal{W}_k$
        Compute $\nabla \hat{f}_{y_k}$ and check $\|\nabla \hat{f}_{y_k}\|_2$
        Compute $s_k$ to approximately minimize $m_{y_k}$ such that $\rho(s_k) \geq \rho'$ (Algorithm 1)
        Generate next subspace
        if performing subspace acceleration
            Compute new acceleration subspace $\mathcal{W}_{k+1}$ from $\mathcal{W}_k$ and $s_k$
        else
            Set $\mathcal{W}_{k+1}$ = colsp($y_k + s_k$)
        end
    end for
```

  - The proposed algorithm is generally faster with a subspace acceleration enhancement. It converges globally even without subspace acceleration.

- **Module 3: A Block Algorithm**
  - To have a block algorithm the solution is to decouple the block Rayleigh quotient into the sum of p separate rank-1 Rayleigh quotients, which can then be addressed individually using the Implicit Riemannian Trust Region strategy [10]. This is done as follows:

    Assume that our iterates satisfy $Y^TAY = \Sigma = \text{diag}(\sigma_1,...,\sigma_p)$, in addition to $Y^TBY = I_p$. In fact, this is a natural consequence of the Rayleigh-Ritz process. Then given given $Y = [y_1...y_p]$, the model m_Y can be rewritten:

$$m_Y(S) = \text{trace}(\Sigma + 2S^T\text{AY}+\text{S}^T(AS\text{ - }BS\Sigma))$$

$$= \sum_{n=1}^{p}(\sigma_i + 2s_i^TAy_i + s_i^T(A - \sigma_iB)s_i) = \sum_{i=1}^{p} m_{y_i}(s_i)$$

  - Note that the update vectors for the decoupled minimizations must have the original orthogonality constraints in place. That is, instead of requiring only that $y_i^TBs_i = 0$, we require that $Y^TBs_i = 0$ for each s_i. This is necessary to guarantee that the next iterate, $Y + S$, has full rank, so that the Rayleigh quotient is defined.

  - As for the truncated conjugate gradient, the p individual IRTR subproblems should be solved simultanoeusly, with the inner iteration stopped as soon as any of the iterations satisfy one of the inner stopping criteria (exceeded trust-region or detected negative curvature). If only a subset of iterations are allowed to continue, then the κ/θ inner stopping criterion may not be feasible.

- **Module 4: Comparison with other Methods**

We will report on preliminary numerical experiments that show the strong potential of our method for computing eigenvalues as compared to other standard methods.

- Lanczos and Arnoldi Method : It exhibits slow convergence and tend to have difficulties finding interior eigenvalues [4]. Moreover, in terms of the number of matrix-vector multiplications (which can be considered as a consistent measure of the computational cost of both algorithms), the proposed method outperform the Lanczos/Arnoldi method (ArnoldiMethod.jl), even for mild accuracy requirements. Even Restarted Lanczos method [11] for generalized eigenproblem has linear convergence (unless ideal preconditioning).

- Jacobi-Davidson Method : The proposed method is better than combination of Jacobi-Davidson and CG method [5], in the sense that it notably differs by the trust-region based inner stopping criterion that avoids a waste of computational effort and yield global convergence properties.

- Power Method : The Power Method implemented in simple.jl, $Bx_{k+1} = Bx_k\tau_k$, where τ_k is a normalizing factor, converges to the principal eigenvector v_n of (A, B) from almost all initial points; but the rate of convergence is only linear [6] and becomes very slow when the eigenvalues of (A,B) are not well separated.

- Inverse Iteration Method: An inverse iteration, $(A - \mu B)x_{k+1} = Bx_k\tau k$, with a shift μ that approximates λ_1, converges linearly to v_1 from almost all initial conditions, implemented in simple.jl, but global convergence is lost in the sense that the iteration converges to the "nearest eigenvector" [7].

- Tracemin: Trace minimization algorithm does not reach superlinear convergence. To improve the speed of convergence of the iteration, Sameh and Tong [8] proposed a dynamic shift technique that appears to be effective in practice but whose workings are not yet rigorously understood.

- LOBPCG : The block version of the proposed algorithm is better than LOBPCG method(lobpcg.jl) in the sense that LOBPCG typically shows no super-linear convergence[13]. Another disadvantage is that when the number of eigenpairs of interest is large we need to form $3m \times 3m$ matrices and solve the corresponding eigenvalue problem of the size $3m$ in the Rayleigh Ritz method on every step [12]. It makes the algorithm less stable as the $3m \times 3m$ eigenvalue problem of the Rayleigh Ritz method is likely to inherit ill-conditioning of the original eigenvalue problem when m is large.

Module 5: Testing and Documentation

I plan on writing the tests and documentation for every commit after it is successfully added to the IterativeSolvers.jl package before moving on to the next task. This would involve:

- Perform test and code coverage analysis if and when possible.
- Speedup checks to make sure that the new code is not completely slowing down Julia's current framework.
- Develop the necessary documentation (design, usage, theory and README).

2.2 Schedule

- **Planned Timings**: Working hours will be flexible but rough timings are:

  10:30 UTC - 17:00 UTC on Weekdays.
  05:30 UTC - 07:30 UTC and 10:30 UTC - 17:00 UTC on Weekends

- **April 24 to May 14: Community Bonding Period**
  - Will spend time in reading the code base of iterative solvers module in Julia to get good understanding of implementation style.
  - The proposed problem solution is available in a standard GenRTR package implemented in Matlab, released by the authors of IRTR algorithm itself. I will spend time in analyzing its implementation structure and how it can be efficiently implemented in Julia.
  - Set up a blog for posting weekly progress along with different ideas, challenges, etc.

- **May 15 to May 22 Implementation of Module 1** (Estimated 1 week)

  x = irtr(fns,params) will return the minimizer (a point on the manifold). The structure **fns** will contain the following function handles necessary to perform the optimization. All these methods will be added to **irtr.jl**

  - **fns.R(x,eta)** : retract tangent vector (η) at T_xM to the manifold.
  - **fns.g(x,eta,zeta)** : Riemannian metric at T_xM
  - **fns.proj(x,eta)** : Project η from nearby T_xM to T_xM (used in iteration to combat round-off)
  - Write a thorough test framework and documentation.

- **May 23 to June 15: Implementation of Module 2** (Estimated 3 weeks)

  (Phase 1 Evaluation deadline)

  - **fns.f(x)** : Compute the objective function at x
  - **fns.fgrad(x)** : Compute the gradient of f at x, returning a tangent vector
  - **fns.fhess(x,eta)** : Apply the Hessian of f, $\mathcal{H}_x : T_xM \to T_xM$
  - **fns.rho(x,eta)** : Compute the ρ-ratio: rho(x,eta)
  - Buffer period. Perform code review to ensure that the design and efficiency are up to scratch. Finish any unfinished work.
  - Review of documentation and tests before the deadline of Phase 1 Evaluation. (On or before 14th June).

- **June 16 to June 30: Implementation of Module 3** (Estimated 2 weeks days)

  Additionally, three other methods may be specified:

  - **fns.rhosearch(x,eta,zeta,rhoprime)** : Returns a value τ such that $\rho(x, \eta + \tau\zeta) \geq \rho^{'}$
  - **fns.prec(x,eta)** : apply an s.p.d. preconditioner for the inner iteration, to eta, a tangent vector in T_xM
  - **fns.dist(x,y)** : returns the distance on the manifold between points x and y
  - **fns.randT(x)** : returns a very small random tangent vector in T_xM
  - Write the necessary documentation.
  - The rest of the time is buffer period.

- **July 1 to August 5: Implementation of Module 4** (Estimated 5 weeks)

  - Implementation of **irtresgev(A,B,p)** : returns the extreme eigenvectors of rank p by calling the Implicit Riemannian Trust-Region with truncated CG solver, **irtr(fns, params)** implemented above.
  - Setting up Benchmarks by adding plots of the distance to the solution versus the number of matrix–vector products by A and B for all above stated standard methods and IRTR method on the same matrix data.
  - Testing the proposed irtresgev() method on matrices arising from different set of problem domains. It can be done extensively in a separate package altogether.
  - The documentation for this module will be written as and when commits happen.

  - Discussion of overall design and estimation of efficiency. Make changes to improve on this if and as required. A Hybrid approach [9] of combining IRTR method with Tracemin method can be tried if time permits.

- **6 August - 12 August: Final Week**
  - Finalize Documentation and testing.
  - Final code review and comments.
  - Integrate all the components and finish off any pending work.

2.3 Deliverables

As guided by Prof.Steven Johnson to the open issue issue in IterativeSolvers.jl. I will try to resolve the one for the generalized eigenproblem by delivering the following:

- *irtr(fns,params)* solver
- *irtresgev(A,B,p)* method
- *test_esgev()* method
- All the above methods will be added to the IterativeSolvers.jl package
- Tests
- Documentation

3 About Me

3.1 Contact Details

- Name: Siddhant Katyan
- Email: siddhant.katyan@research.iiit.ac.in
- Telephone: +91-8707030170
- Timezone: Indian Standard Time (UTC +5:30)

3.2 Profiles

- GitHub username: pyschedelicsid
- Alternative Email: siddhantkatyan@gmail.com

3.3 Current Education Details

- University: International Institute of Information Technology-Hyderabad, India
- Major: MS by Research in Iterative solvers for optimization on Manifolds
- Degree: MS by Research
- Current Year: 1st year
- Expected to graduate: May 2020

3.4 My Background

- I am well acquainted with C, Matlab, and Python programming languages and have done multiple projects based on these. I have primarily worked on iterative solvers including Parallel Algebraic Multigrid methods, where I applied preconditioning techniques for large scale Kernel Methods. As part of my research internship at Microsoft Research India, I worked on low rank approximation in extreme classification problems that also helped me explore Python and its prowess.
- I have been working in the domain of Parallel Scientific Computing for past 2 years. I was Teaching Assistant for the course Parallel Scientific Computing during my undergraduate study at International Institute of Information Technology, Hyderabad.
- I am taking a course on Optimization on Manifolds this semester in which we study theory and algorithms for optimization on smooth manifolds apart from basics of Reimannian Geometry. It broadly covers First-order optimization algorithms on manifolds and Second-order optimization algorithms on manifolds.
- I am comfortable with the math level of a CS graduate.

3.5 Julia Contributions

In order to get familiar with the code base of Julia I went through JacobiDavidson.jl. Though till now I haven't made any commits to the Julia codebase but with the guidance of Prof. Steven Johnson and Harmen Stoppels, I have submitted a couple of ideas, via email, on solving the generalized eigenproblem which is theoretically robust compared to already existing methods implemented in IterativeSolvers.jl.

4 Other Commitments during summer

I am not participating in any other internships during the [redacted]. Towards June-mid, I might not be available for 2-3 days as I will be attending Conference , except that I will be able to devote 45-48 Hours per week and try for more if required.

5 Acknowledgment

Harmen Stoppels, the mentor for this project has been extremely helpful. The discussions with him were very helpful in writing of this proposal, especially addition of the Block version of the proposed algorithm.

References

[1] P-A. Absil, C.G. Baker, K.A. Gallivan : A truncated-CG style method for symmetric generalized eigenvalue problems. Journal of Computational and Applied Mathematics, Volume 189, Issues 1–2, 1 May 2006, Pages 274-285.

[2] Alan Edelman, Tomás A. Arias, and Steven T. Smith : The geometry of algorithms with orthogonality constraints.SIAM J. Matrix Anal. Appl. 20(2)(1998) 303-353.

[3] Y.Notay : Combination of Jacobi–Davidson and conjugate gradients for the partial symmetric eigenproblem. Numer. Linear Algebra Appl. 2002; 9:21–44.

[4] G.H. Golub, Q. Ye : An inverse free preconditioned Krylov Subspace method for symmetric generalized eigenvalue problems, SIAM J. Sci.Comput. 24 (1) (2002) 312–334 (electronic).

[5] G.L.G. Sleijpen, H.A. van der Vorst : A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl. 17 (2)(1996) 401–425.

[6] B.N. Parlett : The symmetric eigenvalue problem, Prentice-Hall, Englewood Cliffs, NJ, 07632, 1980 (republished by SIAM, Philadelphia, 1998).

[7] G.H. Golub, Q. Ye : An exact inverse iteration for generalized eigenvalue problems, BIT 40 (4) (2000) 671–684 (MR 2001j:65063).

[8] A. Sameh, Z. Tong : The trace minimization method for the symmetric generalized eigenvalue problem, J. Comput. Appl. Math. 123 (2000)155–175.

[9] P-A Absil, Christopher G Baker, Kyle A Gallivan, Ahmed Sameh : Adaptive Model Trust Region Methods for Generalized Eigenvalue Problems. ICCS 2005.

[10] C. G. BakerP. -A. AbsilK. A. Gallivan : An Implicit Riemannian Trust-Region Method for the Symmetric Generalized Eigenproblem. Computational Science – ICCS 2006 pp 210-217.

[11] Danny C. Sorensen : Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations. Parallel Numerical Algorithms pp 119-165.

[12] Andrew V. Knyazev : Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. SIAM J. Sci. Comput., 23(2), 517–541.

[13] Andrew Knyazev : Recent implementations, applications, and extensions of the LocallyOptimal Block Preconditioned Conjugate Gradient method (LOBPCG). arXiv:1708.08354.