GSoC Application

Fast Extremal Eigenvalue Iterative Solver with Preconditioners

Mohamed Tarek Mohamed

March 26, 2018

Contents

1 Abstract

In this project, I will implement the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm for finding extremal generalized eigenvalues and their corresponding eigenvectors. The package will be interfaced in IterativeSolvers.jl. Additionally, parallel algebraic multigrid preconditioners will be implemented extending AMG.jl. Common preconditioners will then be grouped into a package and further interfaced in IterativeSolvers.jl. Testing wil be done using buckling problems defined with the help of JuAFEM.jl. The methods developed in this project will be benchmarked against Base.eigs and JacobiDavidson.jl, and results will be published on Github. Finally, the packages developed will be documented and ported to Julia 1.0 when it is released.

2 About Me

Name Mohamed Tarek Mohamed (Github: mohamed82008)

Location Canberra, ACT, Australia

Timezone UTC+10 (April-September) and UTC+11 (October-March)

University University of New South Wales, Canberra

Degree PhD in Computer Science

Research Area Topology optimization

Email mohamed82008@gmail.com

GSoC Organization NumFOCUS

Programming Language Julia

GSoC Mentor Harmen Stoppels (Github: haampie)

I am a 23 year old, first year PhD student in Computer Science, originally from Egypt. My field of research proudly sits in the intersection of mechanical engineering, computational mathematics and optimization theory. I have a Master degree in Industrial and Systems Engineering, and a Bachelor degree in Mechanical Engineering, and I am a passionate computational mathematics reader. I have academic programming experience in Java and Python but my favourite language of all is the Julia programming language which I will be using in this project, should I get accepted. I have experience in open source development using Julia as can be seen from my Github account [1]. I also got permission from my PhD supervisor to commit the time necessary for GSoC during weekdays since this project overlaps with my research interest. Additionally, I will also be working on this project during weekends.

3 Introduction

Extremal eigenvalue solvers are used in a number of practical applications. In structural mechanics for example, the smallest eigenvalue of a generalized eigenvalue problem can give a prediction of how much more load the structure can take before reaching an unstable buckling state. For steady-state vibrating systems, the minimum eigenvalue gives a prediction of the load frequency at which the structure becomes unstable possibly leading to collapses like the Tacoma Narrows bridge incident.

A number of algorithms exist for finding the minimum or maximum eigenvalue λ and its corresponding eigenvector $\boldsymbol{x}$ satisfying $\boldsymbol{Ax} = \lambda \boldsymbol{Bx}$ for any given matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are both Hermitian and $\boldsymbol{B}$ is not singular, then the eigenvector corresponding to the minimum (maximum) eigenvalue is the minimizer (maximizer) of the Rayleigh quotient defined as $\frac{\boldsymbol{x}'\boldsymbol{Ax}}{\boldsymbol{x}'\boldsymbol{Bx}}$ and the eigenvalue itself is the minimum (maximum) value. One scalable algorithm which builds on this idea, to find the extremal eigenvalues and eigenvectors of sparse matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, is the algorithm proposed in [2], called the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm. The LOBPCG algorithm is capable of finding the k

[1] https://github.com/mohamed82008

minimum or maximum eigenvalues and their eigenvectors and is shown to have a good performance in the paper.

4 Summary of Proposed Work

In this project, under the mentorship of Harmen Stoppels who is one of the contributors to IterativeSolvers.jl [2], I would like to implement the LOBPCG algorithm in Julia, and compare its performance against existing sparse matrix eigenvalue functions and packages such as: the Base.eigs function which uses ARPACK's Lanczos and Arnoldi methods, as well as the Jacobi-Davidson algorithm implemented in JacobiDavidson.jl [3]. A proof of concept for a single (non-block) extremal eigenvalue problem, without preconditioning, was written by me in LOBPCG.jl [4].

After the package is completed and well optimized, it will then be interfaced in IterativeSolvers.jl to provide an easy and fast way to find the k extremal eigenvalues and eigenvectors for sparse matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ from IterativeSolvers.jl. IterativeSolvers.jl hosts a number of iterative solvers for solving systems of equations involving huge sparse matrices, but it is still lacking in eigenvalue algorithms.

Another goal of this project is to develop a package that defines a number of common preconditioners which are used to accelerate iterative solvers for solving systems of equations or eigenvalue problems. Of particular interest are the incomplete Cholesky and the algebraic multigrid (AMG) preconditioners which will be defined, wrapped and/or extended. Some variants of the AMG preconditioner already exist in AMG.jl [5]. Moreover, RestrictProlong.jl [6] allows an efficient definition of the "restrict" and "prolong" operators of the AMG preconditioner. However, AMG.jl lacks a number of important features. More specifically, in this part of the project, the Cleary-Luby-Jones-Plassmann (CLJP) coarsening algorithm will be implemented. The CLJP coarsening is known to be more amenable to distributed-memory parallelism as discussed in [1]. More features from the same paper will also be added if time permits. In particular, a parallel AMG package will be attempted

[2] https://github.com/JuliaMath/IterativeSolvers.jl
[3] https://github.com/haampie/JacobiDavidson.jl
[4] https://github.com/mohamed82008/LOBPCG.jl
[5] https://github.com/ranjanan/AMG.jl
[6] https://github.com/timholy/RestrictProlong.jl

including parallel Ruge-Stuben (RS), CLJP and hybrid algorithms from [1].

Testing of the developed tools will be performed using buckling problems from structural mechanics defined using JuAFEM.jl [7]. I have experience in using JuAFEM.jl and I have thoroughly read and modified it for supporting arbitrary precision in my forked version [8]. The final test matrices and benchmarks will then be reported on a Github repository.

Finally, since the release of Julia v1.0 is likely to overlap the GSoC project's period, the packages developed will be developed in a way compatible with both Julia v0.6 and v1.0. Compat.jl [9], top-level code specialization and/or completely different Github branches of the packages will be used, as seen fit on a case-by-case basis.

5 Deliverables and Timeline

Deliverable	**Duration**
1. Efficient LOBPCG.jl with documentation and basic tests	2 weeks
2. Interface for LOBPCG.jl and JacobiDavidson.jl in IterativeSolvers.jl	1 week
3. Testsuite for sparse eigenvalue problems and benchmark results	1 week
	4 weeks
4. Preconditioners.jl providing diagonal, incomplete Cholesky and AMG preconditioners	2 weeks
5. Interface and document Preconditioners.jl in IterativeSolvers.jl	1 week
6. Benchmark preconditioners with LOBPCG	1 week
	4 weeks
7. Julia v1.0 compatible versions of the above deliverables	1 week
8. Develop and test parallel CLJP coarsening AMG	2 weeks
9. Develop and test parallel RS coarsening AMG	2 weeks
	5 weeks

[7] https://github.com/KristofferC/JuAFEM.jl
[8] https://github.com/mohamed82008/JuAFEM.jl/tree/old
[9] https://github.com/JuliaLang/Compat.jl

6 Bibliography

[1] Van Emden Henson and Ulrike Meier Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[2] Andrew V. Knyazev. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.