GSoC 2016 Application Ayush Pandey- Support for complex numbers within Convex.jl

March 25, 2016

1 About Me

1.1 Username and Contact Information

Name : Ayush Pandey
University : Indian Institute of Technology (IIT), Kharagpur
Email : ayush.pandey@iitkgp.ac.in
IRC Handle : KrishnaKanhaiya at freenode.net
Github Username : Ayush-iitkgp

1.2 Personal Background

Hello, I am Ayush Pandey, a 4th year undergraduate student pursuing an Integrated Master of Science(MS) degree in Mathematics and Computing Sciences with micro-specialization in Optimization Theory and Applications at IIT Kharagpur, India. I am proficient in C, C++, Java and Python. I am moderately proficient in Julia.

1.3 Previous Experience

My first encounter with the field of optimization and operation research was during my 3rd year in the college when I took a course in Operation Research. As a part of the course, we were asked to code the Linear Programming methods (source code). I enjoyed the course a lot and since then have been reading optimization. I was also chosen to be a part of the IIT Kharagpur's contingent in 4th InterIIT TechMeet where IIT Kharagpur won gold medal in Portfolio Optimization event in which the participants had to present the solution for the Cardinality Constrained Efficient Frontier which we solved using 3 heuristic methods namely Genetic Algorithm, Tabu Search and Simulated Annealing (source code).

1.4 Relevant Courses

- Operation Research (Theory and Lab)
- Non-Linear Programming (ongoing)
- Convex Optimization (ongoing)
- Linear Algebra, Programming and Data Structures, Object Oriented System Design

1.5 Answers of listed questions

`1. What do you want to have completed by the end of the program?`

By the end of the program, I want to add support for complex-domain optimization problem in Convex.jl which presently supports only real-domain optimization problems.

`2. Who's interested in the work, and how will it benefit them?`

Many problems in applied sciences are posed as optimization problems over the complex field such as Phase retrieval from sparse signals, designing an FIR filter given a desired frequency response. The present

approach is to manually convert the complex-domain problems to real-domain problems (example) and pass to solvers. The correct approach to such problem is to make existing packages like Convex.jl deal with complex-domain problems directly without converting them to real-domain problem.

`3. What are the potential hurdles you might encounter, and how can you resolve them?`

I already satisfy the requirements for the project.

`4. How will you prioritise different aspects of the project like features, API usability, documentation and robustness?`

Please refer to the "Timeline" section where I have described in details about my plans to tackle different aspects of the project.

`5. Does your project have any milestones that you can target throughout the period?`

Yes, at the the end of week 8, the support for the complex-domain problems would be ready for testing for the Julia community.

`6. Are there any stretch goals you can make if the main project goes smoothly?`

Yes, I do plan to publish a research paper under the guidance of my mentors by the end of the GSoc period.

`7. What other time commitments, such as summer courses, other jobs, planned vacations, etc., will you have over the summer?`

I expect to work full time on the project that is 40 or more hours a week. I have no other commitments in the summer as of now.

1.6 Contribution to Open-Source Projects

- (**Merged**) `Added solution Unconstrained Markowitz Efficient Frontier example.`#128
- (**Open**) `Added sdp examples.`#129
- (**Merged**) `Corrected Pass to solver:  stuff matrices objective function.`#9

1.7 Experience with Julia

I have been using Julia for last one and half month. In terms of functionality, I like Julia because of its **multiple dispatch** feature as it lets me overload operators with a lot of ease than other programming languages.

But the most astonishing feature of Julia is that it is empowering. In other high-level languages, the users can not be developers because developing new packages in those language require the users to know the intricacies of low-level language whereas in Julia, users can develop packages for their needs in Julia itself without compromising with the speed.

2 The Project

2.1 The Problem and Motivations

The aim of the project is to add support for solving complex domain optimization problems in Convex.jl (a Julia package for Disciplined Convex Programming).

Many problems in applied sciences are posed as optimization problems over the complex field such as phase retrieval from sparse signals, or designing an FIR filter given a desired frequency response. The present approach is to manually convert the complex-domain problems to real-domain problems (example) and pass to solvers. This process can be time consuming and non-intuitive sometimes. The correct approach to such problem is to make existing packages deal with complex-domain problems. Thus, during this summer, I aim to implement the support for complex-domain optimization problems in Convex.jl.

3 The Plan

I propose to implement required functionality in Convex.jl so that it could accept and solve the complex-domain optimization problems without having the users to explicitly convert the complex problems to real-domain optimization problems. I plan to implement a 6 step procedure to support complex-domain optimization problems within Convex.jl:

1. Add support for complex variables and complex SDPs in Convex.jl
2. Provide the users of the package with the option to express complex SDP in user-friendly language
3. Implement the infrastructure to convert the complex SDP to real SDP internally
4. Solve the real SDP using existing backend solvers via the MathProgBase interface
5. Convert the solution of the real SDP into corresponding complex SDP
6. Output the complex domain solution to the user

I also aim to publish the work done during this period under the guidance of my mentors.

3.1 Mathematical Formulation

First of all, we need to understand the mathematical formulation of complex-domain optimization problems before we get into the implementation details.

3.1.1 Definitions

Hermitian Matrix - A matrix $X \in \mathbf{C}^{n \times n}$ is hermitian if $X = X^*$ where X^* is the conjugate transpose of the X.

Complex Positive Semidefinite Matrix - A matrix $X \in \mathbf{C}^{n \times n}$ is positive semidefinite, we write as $X \succeq 0$ if for all column vectors $\alpha \in \mathbf{C}^n$, we have

$\alpha^T X \alpha \geq 0$

Notation - The inner product of two complex matrices X and Y is represented as $< X, Y >$.

Note - The inner product of two hermitian matrices $< X, Y > = Tr(Y^* X) = Tr(XY)$ and is always real.

3.2 Complex Semidefinite Program

In a complex semidefinite programming, we maximize or minimize for an objective matrix C which is Hermitian matrix subject to linear constraints on X and the constraint that X is complex positive semidefinite.

The canonical form of the Complex Semidefinite Programming Problem is:

minimize

$< C, X >$

subject to

X is Hermitian matrix

$X \succeq 0$

$< A_i, X > \quad \leq b_i, \qquad$ i = 1,2,.... m

where $A_1, A_2, ..., A_m \in \mathbf{C}^{n \times n}$ and $C \in \mathbf{C}^{n \times n}$ are known Hermitian matrices, $b_1, ...b_m \in \mathbf{R}$ are known numbers and $X \in \mathbf{C}^{n \times n}$ is the variable Hermitian matrix.

4 Execution

The problem of complex optimization can be tackled using the bijective transformation τ from the $C^{n\times n}$ space to the $R^{2n\times 2n}$ space.

There is an reduction from complex semidefinite programs to semidefinite programs involving real matrices as a complex matrix $X \in C^{n\times n}$ defines a real matrix

$$\tau(X) = \begin{bmatrix} Real(X) & -Imaginary(X) \\ Imaginary(X) & Real(X) \end{bmatrix}$$

where $Real(X) \in R^{n\times n}$ and $Imaginary(X) \in R^{n\times n}$ are the real and the imaginary parts of X.

Note 1- If complex matrix $X \in C^{n\times n}$ is Hermitian then its transformation $\tau(X) \in R^{2n\times n}$ is positive semidefinite.

Note 2- For two hermitian matrices X, Y

$< \tau(X), \tau(Y) >= Tr(\begin{bmatrix} Real(Y) & -Imaginary(Y) \\ Imaginary(Y) & Real(Y) \end{bmatrix}\begin{bmatrix} Real(X) & -Imaginary(Y) \\ Imaginary(X) & Real(X) \end{bmatrix}) = 2 < X, Y >$

After the transformation τ, the properties of being Hermitian and complex positive semidefinite translate into symmetric and real positive semidefinite.

Thus we have the corresponding real SDP:

minimize

$< \tau(C), Y >$

subject to

Y is symmetric matrix of order $2n$

$Y \succeq 0$

$< \tau(A_i), Y > \quad \leq 2b_i \qquad$ i = 1,2,.... m

$$Y = \begin{bmatrix} D & -E \\ E & D \end{bmatrix}$$

where D is a symmetric matrix of order n and E is a skew-symmetric matrix of same order. Here Y is the variable positive semidefinite matrix of order $2n$ and $\tau(A_i)$ i=1,2....n are the transformation of the hermitian matrices A_i i=1,2....n appearing in canonical form of the complex semidefinite programming problem.

Note- $\tau(Y) = X$, thus if Y is feasible solution for the real SDP, then $\tau^{-1}(Y) = X$ is the feasible solution for the complex SDP and the **value objective function of complex SDP is half that of real SDP**.

Hence solving the complex SDP is equivalent to solving the corresponding real SDP and transforming the value to complex domain.

4.1 1. Support complex variable and complex SDP in Convex.jl

On the lines of the existing variable system in Convex.jl, I would implement 3 new variable types for supporting complex scalar, vector and hermitian positive semidefinite matrix. This would require me to append new variable type to variable.jl which would be a subtype of AbstractExpr.

```
In [ ]: # Add a new complex variable

        z = ComplexVariable()

        ComplexVariable of
        :head: complex
        :size: (1, 1)
```

```
:sign: NotDefined
:vexity: Convex.AffineVexity()

c = ComplexVariable(n)        # n is the number of elements in complex vector
C = HermitianSemidefinite(n)    # n is the order the complex positive semi-definite matrix
```

4.2 2. Provide the users of the package with the option to express complex SDP in user-friendly language

Since the objective function in complex SDP involves **inner product** operation, I would overload the **dot** operator so that it accepts a complex semidefinite matrix variable and an hermitian matrix. I would also need to extend the present API for expressing real SDPs to complex SDPs such that methods it becomes easy for users to express SDPs in Convex.jl.

```
In [ ]: C = [1 -im;im 1] # Initialized a Hermitian matrix
        n = 2
        Z = HermitianSemiDefinite(n)    #Created a complex positive semidefinite matrix
        objective = dot(Z,C)

        A = [2 -im;im 1]
        b = 5
        c1 = dot(Z,A) <= b
        c2 = (Z in :HermitianSDP)

        p = minimize(objective,c1,c2)
```

4.3 3. Implement the infrastructure to convert the complex SDP to real SDP internally

In order to achieve this step, it would be required to make our Convex.jl package capable of distinguishing between real SDPs and complex SDPs. The current implementation exposes an API **"minimize/maximize"** to the user and creates an instance of an user-defined type **Problem**. I plan to make changes in the minimize/maximize API making it capable of distinguishing between real and complex SDPs. So, when the SDP is real, the package would use the existing the implementation and create an instance of "Problem" as usual but when the SDP is complex, first the package would convert the SDP into real SDP using the transformation τ described in the "Execution" section and then create an instance of "Problem". The end result of this step would be an equivalent real SDP of the given complex SDP.

```
In [ ]: # Redefining minimize/maximize API to distinguish between Real and Complex SDPs

        function (objective::AbstractExpr, constraints::Constraint...)
            if objective.children[2].head == :complex # In the if condition, we use the unique represen
                convert_to_real(:minimize, objective, collect(constraints))
            else
                Problem(:minimize, objective, collect(constraints))
            end
        end

        #defining a new function which would take an complex SDP and convert to real SDP using the tran
        function convert_to_real(head::Symbol, objective::AbstractExpr, constraints::Array=Constraint[]
            # These steps would convert the problem to real SDP
            # After the conversion to real SDP, we would create an instance of type Problem
            Problem(:minimize, objective, collect(constraints))
        end
```

4.4 4. Solve the real SDP using existing backend solvers via the MathProgBase interface

Now, when we have an equivalent real SDP of the problem, I plan to use the existing implementation in solution.jl which uses methods from MathProgBase package to convert the problem to conic form(since the converted SDP is already in the conic form, we may not need the conversion to conic form), load the problem into solver and optimize it. The **"solve!"** would be used to solve the SDP and a new instance of user-defined type **"Solution"** is created which has value of the primal and dual variables, status(Optimal, Unbounded), optimum value of the objective function stored in it in different fields. Now, when we have the solution of the real SDP, the next step is to transform the solution of the real SDP to its corresponding complex SDP. Also, the default setting in Convex.jl prints the output from solver(which we disable by passing in verbose=0), I would also need to change the setting so that the output from the solver is not displayed by default as we would not want the solution of the real SDP to be outputted to the user by default.

In [ ]:
```
# Given below is the function in the existing source code which I plan to use to solve the tran
function solve!(problem::Problem;
                warmstart=false, check_vexity=true)

#Checking the vexity of the Problem
  if check_vexity
    vex = vexity(problem)
  end

# Convert the problem to the conic form
# Since we have formulated the complex SDP in such a way that the existing problem is already i
# The step below can be skipped which would lead to less computation time.
  c, A, b, cones, var_to_ranges, vartypes, conic_constraints = conic_problem(problem)

# This method call "loadproblem" method from MathProgrBase which in turn loads the problem to t
  m = problem.model
  load_problem!(m, c, A, b, cones, vartypes)
  if warmstart
    set_warmstart!(m, problem, length(c), var_to_ranges)
  end

# The package uses MathProgbase interface to call the existing methods for optimization
  MathProgBase.optimize!(m)

# The method below assigns the values to different fields in "Solution" type and displays it to
  populate_solution!(m, problem, var_to_ranges, conic_constraints)
  if !(problem.status==:Optimal)
    warn("Problem status $(problem.status); solution may be inaccurate.")
  end
end
```

4.5 5. Convert the solution of the real SDP into corresponding complex SDP

This step is exactly reverse of step 3. In this step, we use the transformation τ^{-1} to obtain the solution of the complex SDP from the solution of the real SDP which is obtained in step 4. Lets us assume that the primal value of the variables of the real SDP is stored in temporary variable Y which is a square matrix of order $2n$. As from the transformation τ^{-1}, it can be easily verified that first n columns of first n rows contain the real part of the complex matrix X and the next n columns contain negative the imaginary part of the matrix X. Also, the optimum value of the complex SDP is half the optimum value of the complex SDP. Thus, I would need to implement a new function which would take the primal values and the optimum values of the real SDP and convert to the corresponding complex SDP. This would be done by creating a function which

would take the user-defined type **"Solution"** (this contains the solution for the real SDP) as an argument and convert it to complex-domain solution. I would also be required to make our solver intelligent so that it could distinguish between real and complex-domain SDPs. This would be done by creating a new field **"domain"** in the user-defined type **"Solution"** which would take values as either **real or complex**. If the the field **domain** contains value complex then the transformation τ^{-1} would be called else no need.

```
In [ ]: # Redefining the user-defined type Solution
        type Solution{T<:Number}
          domain                                  # field domain would either take the value real or compl
          primal::Array{T, 1}
          dual::Array{T, 1}
          status::Symbol
          optval::T
          has_dual::Bool
        end

        #defining a new function which would take the user-defined type "Solution", check if the soluti
        # SDP. If yes, it would use inverse transformation to get the complex-domain solution.
        function real_to_complex(s::Solution)
            if
                s.domain =="real"
                return
            else
                # use the inverse transformation to obtain complex-domain solution. The complex-domain
                end
        end
```

4.6 6. Output the complex domain solution to the user

Now, when we have the complex-domain solution of the complex SDP. I would be using the existing machinery in Convex.jl to output the optimum value of the objective function and the value of the primal variables.

```
In [ ]: # solve the complex SDP
        solve!(p)
        p.optval # Displays the optimum values of the complex SDP

        # example of the atom which accepts complex variables
        x = Variable() #or  x = ComplexVariable()
        e1 = square(x)
        x.value = 4im
        evaluate(e1)

        1x1 Array{Complex{Float64},2}:
        -16.0+0.0im

        # example of the atom which does not accept complex variables
        e2 = max(x,0)
        evaluate(e2)

        Warning:evaluate method can not be called on the expression with complex values, please provide
```

5 Timeline (tentative)

5.0.1 Community Bonding period (22nd April - 22nd May)

My summer vacation will start from 30th of April. During this period, I would want to get myself more familiarized with the source code of Convex.jl. I have in particular 2 issue in mind which I would like to send the pull request to namely:

- `Issue multiplying expressions with matrices.`#122

I believe solving the above issue would help me strengthen my understanding of source code and make myself comfortable with contributing to the package.

- `Missing kron for convex programming variables.`#57

This would be the first step towards supporting complex numbers in Convex.jl, **kron** atom is used in Chebychev design of an FIR filter given a desired frequency response. Thus implementing this feature would make Convex.jl useful in the applications related to filter design

5.0.2 Week 1

Goal: Implement support for complex variable and complex SDP in Convex.jl

I plan to implement the 6 step procedure described in the "Execution" section on weekly basis. We would only be able to merge the code into the existing package after all the 6 steps have been implemented. I would start by making changes in variable.jl file. The end outcome of this week would be that the users would be able to declare a variable as complex-domain variable.

5.0.3 Week 2

Goal: Overload dot operator and extend minimize/maximize API

In this week, I would overload the multiply dot operator to support complex multiplication and extend the present API for expressing real SDPs to complex SDPs such that it becomes easy for users to express complex SDPs in Convex.jl.

5.0.4 Week 3 and 4

Goal: Implement new functions for transformation τ

These 2 weeks would be required to implement step 3 by making changes in problem.jl file. I would also write new routines to convert complex SDPs to real SPDs.

5.0.5 Week 5

Goal: Integration of existing solvers via MathProgBase interface

During this week, I would work to integrate the existing solvers like SCS, Mosek via MathProgBase interface to solve the transformed real SDP, I would also need to change the default setting in Convex.jl so that the output from the solver is not outputted to the user.

5.0.6 Week 6 and 7

Goal: Convert the solution of the real SDP into corresponding complex SDP

During these 2 weeks, I implement a new function which would take the primal values and the optimum values of the real SDP and convert to the corresponding complex SDP using the transformation τ^{-1}.

5.0.7 Week 8

Goal: Display the complex domain solution to the user

During this week, I would be writing codes to use the existing machinery in Convex.jl to output the optimum value of the objective function and the value of the primal variables to the users once the optimization is complete.

At the end of this week, the support for the complex-domain problems in Convex.jl would be ready for testing for the Julia community.

5.0.8 Week 9 and 10

Goal: Documentation, Notebooks, Bug fixes

By this time I will make sure that none of the above implementations has introduced any bug and are complete by documentation as well as testing. I will extend this period to my Future Work as writing example notebooks and preparing for a major release of the package.

5.0.9 Week 11 and 12

Goal: Writing presolve routine

During these two weeks, I plan to understand the existing SDP solvers like SCS and Mosek in detail and try to figure what could be better ways than the existing ones so that we could make our presolve routine better. I would also be required to read and understand the existing presolve routines for Linear Programming Problems so that I could think on similar lines.

5.0.10 End-Term evaluation

Goal: Working towards the publication

Buffer period for any lagging work. I also aim to work towards writing a research paper during this period under the guidance of my mentors.

6 References

[1]. Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming. http://www.sciencedirect.com/science/article/pii/S0022000003001454

[2]. Invariant semidefinite programs. http://arxiv.org/pdf/1007.2905v2.pdf

[3]. Convex Optimization in Julia. http://arxiv.org/pdf/1410.4821.pdf

[4]. #103 Support for complex variables.

[5]. #191 Add complex variables.