Applications of Convex.jl in Optimization Involving Complex Numbers

June 22, 2017

1 Applications of Convex.jl in Optimization Involving Complex Numbers

1.1 Ayush Pandey | JuliaCon 2017

https://github.com/Ayush-iitkgp/JuliaCon2017Presentation

2 About Me

- **BS and MS in Mathematics and Computing Sciences (July'12-June'17) at Indian Institute of Technology Kharagpur**
- **Google Summer of Code 2016 and 2017 student under the Julia Language**
- **GitHub:** Ayush-iitkgp
- **Website:** https://ayush-iitkgp.github.io

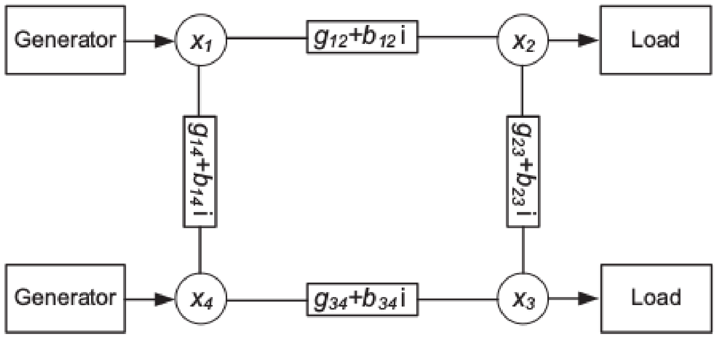
3 Outline

- Power Flow Optimization
- Fidelity in Quantum Information

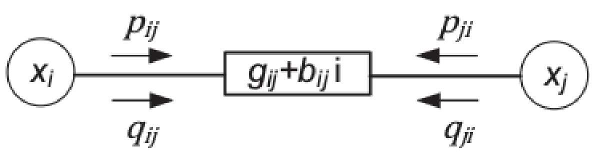
3.0.1 Optimal Power Flow - Introduction

- Power flow study is an analysis of a connected electrical power system's capability to adequately supply the connected load.

- **Unknowns:** - Voltages angle and magnitude information for each bus
- **Knowns:** - Load(such as appliances and lights.) and generator real power and voltage condition.



An example of a power network



Each transmission line has four flows

3.0.2 Optimal Power Flow - Mathematical Formulation

Constraints

- p_{ij}: Active power entering the line from node i
- q_{ij}: Reactive power entering the line from node i
- Let x_i denote the complex voltage for node i of the network.

We have the following power balance equations which are **non-linear** in unknown x_i and x_j.

Objective Depends upon the business needs such as:

- Minimize power losses in an electrical network
- Minimize cost of generation

$$
\begin{aligned}
p_{ij}(x) &= \mathrm{Re}\left\{x_i(x_i - x_j)^*(g_{ij} - b_{ij}\mathrm{i})\right\}, \\
p_{ji}(x) &= \mathrm{Re}\left\{x_j(x_j - x_i)^*(g_{ij} - b_{ij}\mathrm{i})\right\}, \\
q_{ij}(x) &= \mathrm{Im}\left\{x_i(x_i - x_j)^*(g_{ij} - b_{ij}\mathrm{i})\right\}, \\
q_{ji}(x) &= \mathrm{Im}\left\{x_j(x_j - x_i)^*(g_{ij} - b_{ij}\mathrm{i})\right\}.
\end{aligned}
$$

Power balance equations

3.0.3 Optimal Power Flow - SDP Relaxation

- The original optimal power flow problem is non-convex in nature.
- Thanks to the **lifting technique** which converts the above optimization problem to a SemiDefinite Programming Problem.
- The relaxed SDP problem finds the **near global solution** of the original non-convex problem.

3.0.4 Example

- The data is taken from the IEEE 14 Bus test case which represents a portion of the American Electric Power System (in the Midwestern US) as of February, 1962.

In [37]:
```
using Convex  # Read the input data
using FactCheck
using MAT   #Pkg.add("MAT")
TOL = 1e-2;
input = matopen("Data.mat")
varnames = names(input)
Data = read(input, "inj","Y");
n=size(Data[2],1); # Create some intermediate variables
Y=Data[2];
inj=Data[1];
```

In [38]:
```
W = ComplexVariable(n,n); # W is the matrix of pairwise products of the voltages

objective = real(sum(diag(W))); # The objective is to minimize cost of generation

c1 = Constraint[]; # The constraints are power balance equations
for i=2:n
    push!(c1,sum(W[i,:].*(Y[i,:]'))==inj[i]);
end
c2 = W in :SDP;
c3 = real(W[1,1])==1.06^2;
push!(c1, c2);
push!(c1, c3);
```

In [39]:
```
p = maximize(objective,c1); # Create the problem
solve!(p); # Solve the problem
p.optval #15.125857662600703
evaluate(objective) #15.1258578588357
```

```
----------------------------------------------------------------------------
        SCS v1.2.6 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012-2016
----------------------------------------------------------------------------
Lin-sys: sparse-direct, nnz in A = 1344
eps = 1.00e-04, alpha = 1.80, max_iters = 20000, normalize = 1, scale = 5.00
Variables n = 393, constraints m = 812
Cones:        primal zero / dual free vars: 406
```

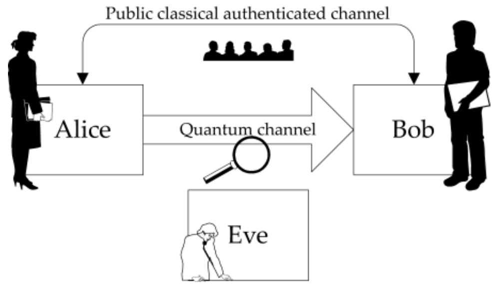
```
        sd vars: 406, sd blks: 1
Setup time: 6.53e-04s
----------------------------------------------------------------------------
 Iter | pri res | dua res | rel gap | pri obj | dua obj | kap/tau | time (s)
----------------------------------------------------------------------------
     0|      inf       inf      -nan      -inf       inf       inf  3.16e-04
   100| 3.92e-02  8.28e-02  6.88e-05 -1.43e+01 -1.43e+01  1.23e-16  1.01e-01
   200| 7.31e-03  1.97e-02  1.66e-05 -1.48e+01 -1.48e+01  0.00e+00  1.61e-01
   300| 2.71e-03  5.88e-03  5.13e-06 -1.50e+01 -1.50e+01  1.27e-16  2.24e-01
   400| 8.98e-04  1.92e-03  1.68e-06 -1.51e+01 -1.51e+01  1.27e-16  2.73e-01
   500| 2.88e-04  6.27e-04  5.46e-07 -1.51e+01 -1.51e+01  1.27e-16  3.30e-01
   600| 9.25e-05  2.02e-04  1.75e-07 -1.51e+01 -1.51e+01  1.27e-16  3.98e-01
   680| 3.73e-05  8.15e-05  7.03e-08 -1.51e+01 -1.51e+01  1.27e-16  4.39e-01
----------------------------------------------------------------------------
Status: Solved
Timing: Solve time: 4.39e-01s
        Lin-sys: nnz in L factor: 2906, avg solve time: 1.97e-05s
        Cones: avg projection time: 6.13e-04s
----------------------------------------------------------------------------
Error metrics:
dist(s, K) = 2.1500e-09, dist(y, K*) = 2.0328e-09, s'y/|s||y| = 3.5844e-13
|Ax + s - b|_2 / (1 + |b|_2) = 3.7310e-05
|A'y + c|_2 / (1 + |c|_2) = 8.1457e-05
|c'x + b'y| / (1 + |c'x| + |b'y|) = 7.0266e-08
----------------------------------------------------------------------------
c'x = -15.1259, -b'y = -15.1259
============================================================================
```

In [40]:
```
output = matopen("Res.mat"); # Verify the results
names(output);
outputData = read(output, "Wres");
Wres = outputData;
real_diff = real(W.value) - real(Wres);
imag_diff = imag(W.value) - imag(Wres);
@fact real_diff => roughly(zeros(n,n), TOL);
@fact imag_diff => roughly(zeros(n,n), TOL)
```

Out[40]:
```
[1m[32mSuccess[0m :: (line:441) :: fact was true
  Expression: imag_diff --> roughly(zeros(n,n),TOL)
    Expected: [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0; 0.0 0.0 0.0 0.0
    Occurred: [7.21891e-9 1.09908e-5 2.03142e-5 2.4322e-5 2.20655e-5 4.88765e-5 4.56099
```

3.0.5 Fidelity in Quantum Information Theory - Introduction

- This example is inspired from a lecture of John Watrous in the course on Theory of Quantum Information.
- Fidelity is a measure of the **closeness** of two quantum states.



Quantum Cryptography

- The ability to distinguish between the quantum states is equivalent to the ability to distinguish between the classical probability distributions.
- If fidelity between two states is 1, they are the same quantum state.
- **Application**
- The Fidelity between two Hermitian semidefinite matrices P and Q is defined as:

$$F(P,Q) = ||P^{1/2}Q^{1/2}||_{tr} = \max \; |trace(P^{1/2}UQ^{1/2})|$$

where the trace norm $||.||_{tr}$ is the sum of the singular values, and the maximization goes over the set of all unitary matrices U.

3.0.6 Fidelity in Quantum Information Theory - Mathematical Formulation

Fidelity can be expressed as the optimal value of the following complex-valued SDP:

$$\textbf{maximize}\frac{1}{2}trace(Z+Z^*)$$

$$\text{subject to } \begin{bmatrix} P & Z \\ Z^* & Q \end{bmatrix} \succeq 0$$

$$\text{where } Z \in \mathbf{C}^{n\times n}$$

3.0.7 Example

```
In [46]: n = 20 # Create the data
         P = randn(n,n) + im*randn(n,n);
         P = P*P';
         Q = randn(n,n) + im*randn(n,n);
         Q = Q*Q';

         Z = ComplexVariable(n,n); # Declare convex variable

         objective = 0.5*real(trace(Z+Z'));   # Specify the problem
```

```
        constraint = [P Z;Z' Q] [U+2AB0] 0;
        problem = maximize(objective,constraint);

        solve!(problem) # Solve the problem
```

```
----------------------------------------------------------------------------
        SCS v1.2.6 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012-2016
----------------------------------------------------------------------------
Lin-sys: sparse-direct, nnz in A = 1621
eps = 1.00e-04, alpha = 1.80, max_iters = 20000, normalize = 1, scale = 5.00
Variables n = 801, constraints m = 6401
Cones:        primal zero / dual free vars: 3161
        sd vars: 3240, sd blks: 1
Setup time: 1.91e-03s
----------------------------------------------------------------------------
 Iter | pri res | dua res | rel gap | pri obj | dua obj | kap/tau | time (s)
----------------------------------------------------------------------------
     0|      inf       inf       -nan       -inf       -inf       inf  7.54e-03
   100| 1.01e-03  2.31e-01  2.50e-02 -5.66e+02 -5.39e+02  0.00e+00  2.30e-01
   200| 2.22e-04  5.11e-02  3.54e-03 -5.75e+02 -5.71e+02  0.00e+00  4.48e-01
   300| 7.54e-05  1.64e-02  7.96e-04 -5.75e+02 -5.74e+02  0.00e+00  6.73e-01
   400| 3.46e-05  6.67e-03  2.29e-04 -5.75e+02 -5.75e+02  0.00e+00  9.00e-01
   500| 1.79e-05  2.98e-03  7.65e-05 -5.75e+02 -5.75e+02  0.00e+00  1.11e+00
   600| 9.20e-06  1.39e-03  2.84e-05 -5.75e+02 -5.75e+02  0.00e+00  1.30e+00
   700| 4.56e-06  6.63e-04  1.16e-05 -5.75e+02 -5.75e+02  0.00e+00  1.51e+00
   800| 2.18e-06  3.20e-04  5.17e-06 -5.75e+02 -5.75e+02  0.00e+00  1.73e+00
   900| 1.03e-06  1.55e-04  2.42e-06 -5.75e+02 -5.75e+02  0.00e+00  1.93e+00
   980| 5.63e-07  8.75e-05  1.35e-06 -5.75e+02 -5.75e+02  0.00e+00  2.14e+00
----------------------------------------------------------------------------
Status: Solved
Timing: Solve time: 2.14e+00s
        Lin-sys: nnz in L factor: 8823, avg solve time: 6.83e-05s
        Cones: avg projection time: 2.04e-03s
----------------------------------------------------------------------------
Error metrics:
dist(s, K) = 3.1242e-09, dist(y, K*) = 1.1761e-09, s'y/|s||y| = -1.5744e-12
|Ax + s - b|_2 / (1 + |b|_2) = 5.6250e-07
|A'y + c|_2 / (1 + |c|_2) = 8.7453e-05
|c'x + b'y| / (1 + |c'x| + |b'y|) = 1.3527e-06
----------------------------------------------------------------------------
c'x = -575.3507, -b'y = -575.3492
============================================================================
```

```
In [47]: # Verify that computer fidelity is equal to actual fidelity
         computed_fidelity = evaluate(objective)
         P1,P2 = eig(P);
```

```
sqP = P2 * diagm([p1^0.5 for p1 in P1]) * P2'
Q1,Q2 = eig(Q)
sqQ = Q2 * diagm([q1^0.5 for q1 in Q1]) * Q2'
actual_fidelity = sum(svd(sqP * sqQ)[2])
diff = computed_fidelity - actual_fidelity
@fact diff => roughly(0, TOL)
```

```
Out[47]: [1m[32mSuccess[0m :: (line:441) :: fact was true
           Expression: diff --> roughly(0,TOL)
             Expected: 0
             Occurred: -0.0010774993736504257
```

4 Thank You!