Sparse Symmetrizers of Diagonalizable Real-λ Matrices

Apoorva Panidapu

Under the direction of

Mo Chen
Graduate Student
Massachusetts Institute of Technology

Research Science Institute
August 2, 2022

Abstract

Every diagonalizable $m \times m$ matrix A with real eigenvalues can be symmetrized by some change of basis S; that is, there exists an invertible S such that $H = SAS^{-1}$ is symmetric. In fact, there are infinitely many such S, but we want to specifically find sparse S, such as a diagonal or tridiagonal matrix. To this end, we formulate and code a convex optimization problem that enforces matrix sparsity by minimizing the L^1 norm. We successfully recover sparse outputs for diagonal and diagonally dominant matrices A, but fail to find sparse S for tridiagonal and sparse random matrices.

Summary

In physics and engineering computational problems, we can encode models of the real world as arrays that we call matrices. Then, through simple multiplications, we can symmetrize such matrices to better understand the objects they model. For example, in wave equations, finding the right symmetrizer yields the law of conservation of energy. In this paper, we write an algorithm that automatically finds symmetrizers that improve our understanding of physical problems and can potentially increase computational efficiency when working with real-world models.

1 Introduction

Sparse matrices, arrays in which most of the elements are zero, have recently become a subject of interest in fields where computational efficiency is greatly valued, such as compressed sensing and signal processing. For large matrices that arise in these sorts of physics and engineering computational problems, there is typically some sort of natural sparse symmetrizing change of basis that can reveal a right inner product or weighting to better understand the problem. For example, finding the right inner product in wave equations yields the law of conservation of energy [1]. However, when working with these large matrices, it can be difficult to find the necessary matrix to reveal the inner product. To solve this issue, we can instead frame this question as an optimization problem to automatically produce this change of basis.

More specifically, for every diagonalizable $m \times m$ matrix A with real eigenvalues, there exists an invertible S such that $H = SAS^{-1}$ is symmetric, meaning $H = H^T$. In fact, there are infinitely many such S, but we prefer sparser S, such as a diagonal or tridiagonal matrix. In this paper, we will study if such a sparse S can be generated by solving some tractable optimization problem. Moreover, we would like to phrase it in terms of a convex optimization problem. This is because convexity allows us to find a global optimizer efficiently and lends itself to several powerful theoretical tools, such as duality [2].

Through numerical experiments in the computing language Julia, we determine that our convex optimization formulation is able to automatically output sparse matrices for diagonal and diagonally dominant matrices A. However, the program fails to output a satisfactorily sparse matrix for tridiagonal and sparse random matrices A. We speculate that this is due to these matrices being less sparse.

The remainder of the paper is structured as follows: In Section 2, we introduce the necessary background to understand the problem statement. We then proceed onto the convex reformulation of the problem in Section 3 and provide some illustrative examples in Section 4. In Section 5, we briefly elaborate on the relevance of p-norms to the notion of sparsity in optimization. Finally, in Section 6, we discuss the results of this research and potential future work, and posit its practical applications in Section 7. To see the code implemented for the convex optimization problem, refer to Appendix A.

2 Preliminary Information

To better understand the problem statement, we provide the following necessary definitions which commonly arise in physical computation problems.

Definition 2.1. A square $m \times m$ matrix H is said to be *symmetric* if $a_{i,j} = a_{j,i}$ for all i, j. In other words, H is symmetric if and only if $H = H^T$.

Definition 2.2. A square $m \times m$ matrix A is said to be *diagonal* if all the entries outside the main diagonal are zero, i.e, if $a_{i,j} = 0$ for $i \neq j$.

For example, the identity matrix is a diagonal matrix.

Definition 2.3. A square $m \times m$ matrix A is said to be *diagonalizable* if there exists a diagonal matrix D and an invertible matrix C that satisfies

$$C^{-1}AC = D.$$

In other words, $A = CDC^{-1}$ is similar to a diagonal matrix D.

Definition 2.4. Let A be a square $m \times m$ matrix and $p \geq 1$ be a real number. Then, the L^p-norm of A, denoted $||A||_p$, is defined to be

$$||A||_p = \left(\sum_{i=1}^{m} \sum_{j=1}^{m} |a_{i,j}|^p \right)^{1/p}.$$

Definition 2.5. A square $m \times m$ real symmetric matrix W is said to be *positive-definite* if $z^T W z > 0$ for every non-zero vector $z \in \mathbb{R}^m$.

There is no universal rigorous definition for sparsity, so to satisfy the intentions of our problem, we define it as follows.

Definition 2.6. A matrix is *sparse* if over 70% of its elements are zero.

This condition of sparsity allows computation with large matrices to be significantly more efficient because the number of operations performed decreases; the sparser a matrix is, the faster the output time in corresponding computational problems.

Now that we have defined the relevant terms, we can proceed to the problem analysis and reformulation.

3 Positive-Definite Reformulations

Recall that for diagonalizable $m \times m$ matrix A with real egenvalues, there exists an invertible S such that $H = SAS^{-1}$ is symmetric. Then, by definition,

$$H = SAS^{-1} = H^T = (S^{-1})^T A^T S^T.$$

After a simple manipulation involving cancellation by inverses, we obtain the equation

$$S^T SA = A^T S^T S.$$

Define the symmetric positive-definite matrix $W = S^T S$ to write the equation

$$S^T SA = WA = A^T S^T S = A^T W.$$

Then, we can instead ask the question of whether we can find a sparse positive-definite W such that $WA = A^T W$. If so, we can work backwards to find sparse S, if necessary, using methods like the sparse Cholesky factorization [3].

This reformulation is beneficial because $WA = A^T W$ is a linear relationship in W, which means it is easier to analyze with a wider variety of methods. Moreover, W defines a weighted inner product $\langle x, y\rangle = x^T W y$ which reveals the symmetric property of A without changing the basis.

We now frame this question as a convex optimization problem as follows:

$$\min_{W\in\mathbb{R}^{m\times m}} ||W||_1, \tag{1}$$

$$\text{subject to } WA = A^T W, \tag{2}$$

$$W = W^T, \tag{3}$$

$$W \succeq I. \tag{4}$$

Note that, for simplicity, we restrict A and W to be real matrices. We minimize a regularizer $||W||_1 = \sum_{i,j} |W_{i,j}|$ to push the optimization towards sparse solutions W, as is well-known from the field of compressed sensing. The last condition corresponds to W being positive-definite, which ensures that we obtain nontrivial solutions.

However, to code this algorithm efficiently, we further manipulate the written formulation. In Julia, we are able to declare a matrix to be positive-definite instead of having an additional constraint. To this end, we let matrix $V = W - I$ so the inequality constraint $W \succeq I$ becomes $V \succeq 0$ for $V = W - I$. Using the substitution $W = V + I$ we can rewrite equality constraint (2) as

$$(V + I)A = A^T(V + I)$$

and constraint (3) as $(V + I) = (V + I)^T$, which simplifies to $V = V^T$. Now, we proceed according to standard procedure in such optimization problems and declare a new general variable T to rewrite (5) as follows:

$$\min_{T\in\mathbb{R}^{m\times m}} \text{sum}(T), \tag{5}$$

$$\text{subject to } (V + I)A = A^T(V + I),$$

$$V = V^T,$$

$$T. \geq V + I,$$

$$T. \geq -(V + I),$$

$$V \succeq 0,$$

where $. \geq$ denotes an element-wise comparison, meaning all corresponding elements satisfy the written inequality.

4 Computed Example for Antisymmetric Matrices

To illustrate the optimization algorithm, we discuss the following example taken from Johnson [4].

Let there be two coupled masses on springs, as shown in Figure 1.

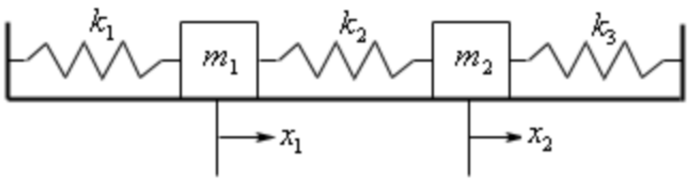

Figure 1: Two coupled masses on springs

Using Newton's law for the positions $x_1(t)$ and $x_2(t)$ and by letting $v = dx/dt$, we can translate this situation into the following system of first-order ordinary differential equations which simplifies to:

$$\frac{dx_1}{dt} = v_1 \frac{dx_2}{dt} = v_2 \frac{dv_1}{dt} = -\frac{k_3}{m} x_2 + \frac{k_2}{m}(x_1 - x_2).$$

Here, the spring with spring constant k_2 exerts a force with magnitude $k_2(x_1 - x_2)$ on the two masses.

We now write this system in matrix form as

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{(k_1+k_2)}{m} & \frac{k_2}{m} & 0 & 0 \\ \frac{k_2}{m} & -\frac{(k_3+k_2)}{m} & 0 & 0 \end{bmatrix}.$$

For simplicity, we set $m_1 = m_2 = m$ and $k_1/m = k_2/m = k_3/m = 0.01$ to obtain matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.02 & 0.01 & 0 & 0 \\ 0.01 & -0.02 & 0 & 0 \end{bmatrix}.$$

We slightly modify the convex optimization program provided in Appendix A to account for the antisymmetric nature of A, and obtain the output matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 66.67 & 33.33 \\ 0 & 0 & 33.33 & 66.67 \end{bmatrix}.$$

This matrix illustrates a basis under which the two modes of the springs in the problem are orthogonal, which allows one to build complicated types of motion from these modes.

5 L^p-norms

We are optimizing for sparsity in this problem, which can be thought of as minimizing the "L^0"-norm of a matrix which counts the number of nonzero entries. In our formulation, we have $p = 1$ as a convex condition. However, the formulation fails to recover sparse symmetrizing matrices for tridiagonal and sparse random matrices. We speculate that this is because the L^1 norm doesn't enforce sparsity enough. Instead, we consider two alternative routes to output sparse matrices in these cases: minimizing the p-norm for $0 < p < 1$ or weighting the main diagonal of the matrix by some small value such that it makes nonzero entries off the diagonal more costly, thereby pushing nonzero terms onto the diagonal in order to satisfactorily minimize the objective function.

6 Results and Future Work

We use the numerical computing language Julia to solve our proposed convex optimization problem with an SDP solver [5] for generated examples. Thus far, we have verified that we can find sparse W for diagonal and diagonally dominant matrices. However, tridiagonal and sparse random matrices didn't produce a sparse enough W. As discussed in the previous Section 5, an interesting direction to pursue is seeing whether minimizing the L^p-norm of W for $0 < p < 1$ (as opposed to the L^1-norm) would produce sparser W. We speculate that this approach would succeed based on previous literature in optimization, but it comes at the cost of the convexity property, taking a toll on computational efficiency. We haven't yet verified this claim due to software limitations, as there is currently no existing solver in Julia that can accommodate for the variety of constraint types in our formulation.

However, we can easily adapt the algorithm to succeed in producing sparse matrices for antisymmetric matrices, as seen in Section 4. Moreover, we have only been concerned with finding S such that $SAS^{-1} = H$ for a real symmetric matrix H, but we can write similar formulations to find S such that $SAS^{-1} = G$, where G is a complex-valued Hermitian matrix where $G = \overline{G^T}$.

The potential real-world applications of this research are discussed in the next section.

7 Practical Takeaways

As mentioned in Section 1, sparse matrices have major implications in improving efficiency and understanding in physics and engineering computational problems. Thus, this research can be applied to uncover certain properties for physical problems that allow scientists to better understand the intuition and structure of their problem. Moreover, this research further adds to the literature of leveraging numerical algebra to speed up computation without losing a significant amount of information. Finally, tangential applications to this research include higher accuracy with lower cost in compressed sensing and more complex recommendation systems since sparsity would allow these systems to handle greater amounts of information.

8 Acknowledgments

I'd like to thank Mo Chen for mentoring this research and Professor Steven Johnson for supervising and offering valuable guidance. I'm furthermore grateful to Dr. John Rickert, Dr. Tanya Khovanova, Dr. Jessie Oehrlein, Professor David Jerison, and Professor Ankur Moitra for their helpful comments and suggestions. I'd also like to thank RSI, CEE, MIT, and the generous sponsors, Mr. Alexander Clark, Mr. Alexander Wu, Dana Caulder, Ph.D., Mr. David Schairer, Eric Rains, Ph.D., Mr. and Mrs. Tony Eltringham, Mr. Sam and Mrs. Cindy Leung, and Admiral William A. Owens (USN, Ret.) for the opportunity to work on this problem. Finally, I want to thank my family and my friends at the Research Science Institute for their unceasing support and company.

References

[1] S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 2018.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] T. A. Davis. *Direct Methods for Sparse Linear Systems*, pages 37–68. Society for Industrial and Applied Mathematics, 2006.

[4] S. Johnson, Lecture notes in 18.06: Linear algebra at mit. Available at , `https://github.com/mitmath/1806/blob/master/notes/ODEs.ipynb`.

[5] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[6] G. Calafiore and L. Ghaoui. *Optimization Models*. Cambridge University Press, 2014.

A Code in Julia

The following is our optimization code in the case of a random diagonal matrix S. The program prints out the values of W, V, $V+I$, and $W^TA - A^TW$ to verify that the problem constraints are satisfied.

```
using JuMP, SCS, LinearAlgebra, HiGHS, Distributions, BenchmarkTools
n = 10
H = Hermitian(rand(n,n))
d = rand(n)
S = diagm(d)
A = S^(-1)*H*S
W = S'*S
model = Model(SCS.Optimizer)
    set_silent(model)
    @variable(model, T[1:n, 1:n])
    @variable(model, V[1:n, 1:n], PSD)
    @objective(model, Min, sum(T))
    @constraint(model, c1, V .== V')
    @constraint(model, c2, T .>= (V + I))
    @constraint(model, c3, T .>= -(V + I))
    @constraint(model, c4, (V + I)*A .== A'*(V + I))
    JuMP.optimize!(model)
    T_sol = JuMP.value.(T)
    if termination_status(model) == JuMP.MOI.OPTIMAL
        @show termination_status(model)
        @show primal_status(model)
    @show objective_value(model)
    @btime value(V)
    @show display(value(V) + I)
    @show display(W)
    @show display(W*A - A'*W)
end
```