My research is in the fields of **numerical analysis** and **scientific computing**. I am especially interested in the mathematical modeling of physical systems and the efficient solution of large-scale scientific problems ubiquitous in fields like image reconstruction, computer vision, robotics, dynamic networks, and dynamical systems. I seek to develop mathematically rigorous algorithms to efficiently solve such problems that are defined by increasingly larger data sets arising in a wide range of applications from science and engineering. In addition to my background in numerical computing, I also have extensive experience in applied linear algebra, which is foundational in machine learning algorithms, for instance efficient solution of the (partial) eigendecomposition or the (partial) singular value decomposition (SVD). Matrices, a standard data structure used in numerical methods, in general have a natural translation to graph theory. My coursework throughout my Ph.D. studies provides me with relevant foundation in optimization and control as well as nonlinear inverse problems, prevalent in computer vision and robotics.

My thesis work has focused on improving the convergence of iterative methods used to solve large-scale problems; in particular, solutions of long sequences of very large linear systems and computing approximate invariant subspaces of such systems. I work with data arising in applications from mechanical and civil engineering, geoscience, and computational fluid dynamics. My most recent work has been with systems governed by partial differential equations such as those arising in the numerical simulation of brake squeal [12], topology optimization [3, 14, 17, 23, 29], transient hydraulic tomography [2], and convective thermal flow [15]. I work with established methods, such as preconditioning techniques, Krylov subspace methods, finite elements and differences, and model reduction, to develop and implement novel strategies and innovations to make the solution of large-scale systems tractable. I use insight to the application to exploit the structure, behavior, and characteristics of these different data sets to make informed choices when implementing such strategies. Below, I briefly describe specific research projects I have worked on or am currently working on.

### Solving Systems

I am interested in the efficient solution of sequences of systems that change slowly over time, and in particular those that are very large and/or otherwise difficult to solve. Such sequences arise in a wide range of applications. For instance, we may use discrete gradient descent methods to solve nonlinear systems of equations in optimization and optimal control problems in robotics and computer vision. When applying a line search method such as discrete Newton’s method, we generate a series of solutions of the form \( u^{(k+1)} = u^{(k)} - J(u^{(k)})^{-1}F(u^{(k)}) \), and subsequently a sequence of Jacobian matrices of the form \( J(u^{(0)}), J(u^{(1)}), J(u^{(2)}), \ldots, J(u^{(n)}) \). We can also consider image provenance in media forensics, where we examine the evolution history of a forged image to recover the origin of the media. These modified images, represented using matrices, then constitute a sequence.

Sequences also arise in topology optimization where we are interested in material distribution in large structures and complex materials. In designing the topology, we minimize an objective function subject to certain constraints specific to the structure and its purpose. For instance, when designing buildings or bridges, we want to determine the optimal design to sustain extreme weather conditions or seismic activity. In the numerical solution of these problems, there can be millions of unknowns, and the optimization procedure typically takes hundreds, but possibly up to 1000, optimization steps before reaching the optimal design. Each finite element step requires the solution of a linear system (see Figure 1 for an overview of the algorithm). In addition to being very large, each corresponding system is ill-conditioned, making them even more difficult to solve.

![Figure 1: Overview of the topology optimization algorithm (modified from [29]).](image)

For large, sparse matrices, when solving the linear system \( Ax = b \), we often use iterative methods, for instance **Krylov subspace methods**. Such methods seek to minimize \( \|r_k\|_2 = \min_{x\in\mathbb{R}^n} \|b - Ax\|_2 \), where \( K(A,r) = \{r, Ar, A^2r, A^3r, \ldots, A^{k-1}r\} \) is the Krylov space defined by our matrix \( A \) and the residual \( r \). We can generally expect fast convergence of these methods when the eigenvalues of \( A \) are clustered away from the origin (and we say...
A is well-conditioned), but in practice we cannot control the spectrum of a matrix. In cases where the matrix is ill-conditioned we can construct another matrix we call a preconditioner $P \approx A^{-1}$ such that $PA$ has a clustered spectrum away from the origin, and then our Krylov method applied to $PAx = Pb$ will converge much faster. There are numerous techniques for constructing $P$. We can compute incomplete factorizations of $A$, such as ILU-type preconditioners (e.g., ILUTP [7, 20, 21]), the factorized sparse approximate inverse (AINV) [4], and, for symmetric matrices, incomplete Cholesky. A robust, but generally expensive preconditioner is the algebraic multigrid (AMG) preconditioner [18, 19, 26, 27, 28].

Recycling preconditioners is a popular method for avoiding the potentially high cost of recomputing a new preconditioner too often, by updating and reusing a previously computed preconditioner. I helped to develop a cheap technique to approximately map one matrix in a sequence of systems to another, closely related matrix in this same sequence for which we already have a preconditioner with good convergence properties [9]. We refer to this technique as the Sparse Approximate Map, or SAM update. SAMs are independent from (and therefore agnostic to) the preconditioner type, and allow us to amortize the possibly high cost of a previous preconditioner over many linear solves. Given a sequence of matrices, $\{A_i\}_{i=1}^n$, a preconditioner $P_j$ for $A_i$ and $j < k$, we compute an approximate map $N_k$ such that $A_iN_k \approx A_j$, by solving the least squares problem $N_k = \arg\min_{N \in S} \|A_iN - A_j\|_F$ and define the updated preconditioner as $P_k = N_kP_j$.

Here $S$ is the subspace defined by a chosen sparsity pattern $S$ imposed on our map and different choices for SAM updates can be considered that exploit salient features of the problem. In [9], we analyze fixed subpatterns based on the finite element mesh from which the topology optimization matrices are derived, in particular, patterns that are much sparser than the matrix itself. Here, our most effective map contains an average of about 5.5 nonzeros per column, compared with up to 81 nonzeros in a typical column of the stiffness matrix and takes just over 5 seconds to compute compared with 16.5 minutes to compute the AMG preconditioner for matrices of size $n = 432450$.

When applied to systems arising in graph-based applications, such as dynamic networks, we could consider a graph interpretation approach when choosing a sparsity pattern, for instance one that considers centrality in a network. A general graph-based interpretation of sparsity patterns is given in [11] (and references therein). In particular, for $G(A)$ the (directed) graph representation of the sparsity pattern of a matrix $A$, the sparsity patterns used for the topology optimization matrices can be defined as subgraphs $G'(A) \subseteq G(A)$. More generally, the minimum spanning tree of a graph can be considered a sparsity (sub)pattern. Using powers of $A$ is another standard choice also explored in [9] and can be described as subsets of the transitive closure of $G(A)$.

Efficiently Solving Large-Scale Systems

Applying numerical methods, like the finite element method or finite differences, is often required when solving systems governed by PDEs. In order to achieve accurate solutions, these discretization techniques may result in truly large-scale systems. We can reduce the computational cost of solving them using model reduction, which aims to replace a high-dimensional linear dynamical system by a much lower dimensional dynamical system. Such dynamical systems arise in catastrophe modeling, optimal control, and circuit design, among many others. In my research, I have experience working with interpolatory model reduction and in particular the Iterative Rational Krylov Algorithm (IRKA), which finds optimal interpolation points to construct the reduced model [1, 13]. Solving for these shifts requires many linear solves, and in [9], we show how recycling preconditioners can be used to reduce overall computation time. I am currently working with systems arising in applications using proper orthogonal decomposition (POD). In POD, we are interested in a much smaller decomposition of the physical system that describes its behavior such as a subset of the eigenbasis spanned by the eigenvectors associated with the largest or smallest eigenvalues. For instance, principal component analysis (PCA) is a POD method that computes the eigenbasis of the covariance matrix in order to determine principal components of a data set and is used in machine learning for building predictive models.

In [12], Gräbner et al. use POD to analyze instability in a finite element model simulating brake squeal (see Figures 2a and 2b; both come from [12]). In my current work, I combine techniques to reduce the cost of the linear solves involved, as well as the cost associated with finding the invariant subspace of interest [10]. The behavior of this system can be analyzed by computing the eigenvalues and eigenvectors of the associated quadratic eigenvalue problem (QEP) describing the system: $(\lambda^2M + \lambda D + K)x = 0$, where $M$, $K$, and $D$ are the mass, stiffness, and damping matrices, respectively, from the finite element discretization and industrial models can have over one million degrees of freedom. The eigenvalues that have positive real part are associated with the vibration causing the squeal and are referred to as the unstable eigenvalues. The approximation to the subspace associated with the region of interest (i.e., the region containing the unstable eigenvalues) is then used to project the larger system into a much smaller subspace, but requires the computation of an eigenbasis. To do this, Gräbner et al. use a process referred to as shift-invert Arnoldi, which involves choosing multiple shifts $\tau$ near these unstable eigenvalues, forming several
new (closely related) QEPs. Each is then linearized to obtain the equivalent eigenvalue problem $A_y = \mu_y y$ with

$$A_y = \begin{bmatrix} -K^{-1}D & -K^{-1}M \ I_n & 0_n \end{bmatrix}.$$

The Krylov-Schur algorithm [24, 25] is frequently used to compute approximate invariant subspaces or eigendecompositions, but due to $K^{-1}$ the matrix-vector products in Krylov-Schur using (1) will also implicitly require a linear solve. The system (1) can be very ill-conditioned, and so these linear solves can be very difficult to compute. Coupled with preconditioning, I use Krylov subspace recycling [16] to efficiently solve these matvecs. To further improve convergence, and following the work in [5], I also reuse a previously computed invariant subspace to warm-start Krylov-Schur for the next, closely related system. Finally, since the warm-start space does not form a Krylov space for the current system, we can no longer use the convergence test from (exact) Krylov-Schur. I incorporate a test for convergence based on the relative residual of the equivalent generalized eigenvalue problem.

Perturbation Theory

Knowing the theoretical properties of methods can provide us insight in how well an approach may work, as well as how to choose parameters when using them. For instance, perturbation analysis can provide us with knowledge a priori on the sensitivity of systems to small changes in, or corruption of, the data. This may arise, for instance, in linear problems as loss of precision with finite arithmetic, or in nonlinear problems as blurry images that we want to reconstruct. With generalizability of learning algorithms, we are interested in the performance of a learning model on unseen data. We may consider how the introduction of noise to training data may affect the model’s resilience to outliers, and by how much. I am currently interested in convergence bounds of iterative methods for perturbed matrices of the form $I + K + E$ ($I$ the $n \times n$ identity matrix, rank($K$) = $p < n$, and $\|E\|_2 = \epsilon < 1$) [8]. Matrices of a related form arise in robust principal component analysis (RPCA) [6] as the sum of a low rank and sparse matrix. Sequences of low rank matrices also arise in optimization problems such as image compression or recognition, and video streaming, when extracting low-dimensional information from high-dimensional data.

In the case of the Krylov method GMRES, convergence rates for matrices of the forms $I + K + E$ do not exist. I am interested in how the introduction of noise affects the eigenvalues of $I + K$ within the context of the theoretical framework provided in [22], which analyzes the GMRES convergence of general perturbed matrices, $A + E$. Defining $A = I + K$, and by taking the singular value decomposition of $K$,

$$K = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2 \end{bmatrix} = U_1 \Sigma_1 V_1^T,$$

(2)

we can ultimately show there are at most $2p$ such eigenvalues from two sources: the $p$ angles shared between $U_1$ and $V_2$, and the $p$ eigenvalues of the $p \times p$ matrix $\Sigma_1 V_1^T U_1$. While I am interested in how these (very few) sensitive eigenvalues affect the convergence behavior of the iterative method GMRES, the SVD in (2) that identifies the sources of sensitivity is independent of any solver. So, it may also provide us with insight in the theoretical framework of the generalizability of learning models through the lens of eigenvalue perturbation theory.

Funding

My research done during my thesis work was funded directly or in part by:

1. AFOSR-BRI FA9550-12-1-0442 Co-Design of Hardware/Software for Predicting MAV Aerodynamics
2. NSF-DSM 1025327 QMC Calculations for Deep Earth Materials
3. NSF-DSM 1217156 Innovative Integrative Strategies for Nonlinear Parametric Inversion

Below, are several sources of future potential funding that are aligned with numerical and scientific computing.

1. NSF Computing and Communication Foundations: Algorithmic Foundations
2. NSF Applied Mathematics
3. NSF Computational Mathematics
4. DOE Advanced Scientific Computing Research (Applied Mathematics)
References


[7] A. Carr and E. de Sturler, MATLAB® implementation of Yousef Saad’s ILUT algorithm. Available from https://arxiv.org/abs/1601.05883, 2016. This implementation closely follows the ILUT paper and Sparstok, while exploiting several steps to ensure an efficient implementation in Matlab®.


[25] A. Carr and E. de Sturler, MATLAS® implementation of Yousef Saad’s ILUT algorithm. Available from https://arxiv.org/abs/1601.05883, 2016. This implementation closely follows the ILUT paper and Sparstok, while exploiting several steps to ensure an efficient implementation in Matlab®.


