EXPLOITING COMPRESSION IN SOLVING DISCRETIZED LINEAR SYSTEMS

BY

ERIN ELIZABETH CARRIER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor Michael T. Heath, Chair
Professor Luke Olson
Professor Paul Fischer
Professor Per Christian Hansen, Technical University of Denmark

Abstract

Solving systems of linear algebraic equations is crucial for many computational problems in science and engineering. Numerous techniques are available for solving such linear systems, including direct methods such as Gaussian elimination and iterative methods such as GMRES. This thesis proposes a method for exploiting compression while computing the solution to a given discretized system of linear algebraic equations and investigates both its overall effectiveness in practice and which factors determine its effectiveness. The method is based on computing an approximate solution in a reduced space, and thus we seek a basis in which the solution has a compressed representation and can consequently be computed more efficiently. We address three primary issues: (1) how to compute an approximate solution to the given discretized linear system using a given basis, (2) how to choose a basis that yields significant compression, and (3) how to detect when the basis is of sufficient dimension to provide a satisfactory approximation. While all three aspects have antecedents in previous ideas and methods, we combine, adapt, and extend them in a manner we believe to be novel for the purpose of solving discretized linear systems. We demonstrate that the resulting method can be competitive with, and sometimes outperform, current standard methods and is effective for efficiently solving linear systems resulting from the discretization of major classes of continuous problems.

Acknowledgments

This dissertation would not have been possible without the support of many people. First, I would like to thank my advisor Professor Michael Heath for guiding me through this project and for his thoughtful contributions and advice throughout the research and writing process. I also would like to thank my committee members, Professors Luke Olson, Paul Fischer, and Per Christian Hansen, for their insightful feedback on my research.

I am grateful to all of the faculty and students in the Scientific Computing group for creating a welcoming and friendly environment. To the scientific computing faculty, thank you for wanting the best for all of the students and helping them to succeed. To all of the scientific computing students, thank you for making the office a great and entertaining place to work.

I also thank the numerous other teachers and professors I have had throughout my entire education. In particular, I would like to thank Greg Wolffe at Grand Valley State University for his willingness to serve as an advisor and a research mentor during my undergraduate degree.

To all of my friends, I am eternally grateful for your always being there as a friendly sounding board and for the many celebrations and dinners together. Thanks for the support through the stressful times and for creating everlasting memories. Without you all, graduate school would not have been the same.

Finally, I would like to thank my family for their support throughout my seemingly endless years of education. Most importantly, thank you to Nate Bowman, for being incredibly supportive throughout it all.

Table of Contents

Chapter 1: Compression Method

1.1 MOTIVATION

In this thesis, we explore a simple and general approach to computing an approximate solution to a discretized system of linear algebraic equations based on the idea that the most convenient basis for the discretization may not yield an efficient representation of the solution. To motivate such a method, we present an illustrative example. Consider solving a differential equation, discretized using a finite difference method. While discretizing using finite differences yields a *sparse* linear system, for most problems one would expect that the solution vector to such a linear system would be dense but have an underlying pattern, as it represents samples of the smooth solution function. This leads to the natural question: might there be a compressed representation of the solution vector if it is represented in a different discrete space, where a representation of a solution is said to be *compressed* if it depends on far fewer parameters than the dimension of the linear system? If so, how can we exploit the compressed representation to compute the solution vector more efficiently, and how can we choose such a discrete space? We will see that the answer to the first question is often "yes," and the answers to the second question are the topic of this thesis.

1.2 OVERVIEW OF THESIS

In the remainder of this chapter, we present a method that exploits such compression when solving systems of linear equations and discuss the details of the method as well as its relationship to other existing ideas and methods. In Chapter 2, we discuss forming and choosing a discrete basis for the compression space. Chapters 3 and 4 present numerical results to test specific hypotheses and the overall effectiveness of the method on 1D and 2D test problems, respectively. Chapter 5 presents a termination criterion to determine when the dimension of the discrete basis suffices to produce a satisfactory approximate solution. A preliminary version of a portion of this work was presented in [1].

1.3 BASIC COMPRESSION METHOD

We begin by formalizing the concept of a compressed representation of a solution. Let $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ be an $m \times n$ linear system, where the $m \times n$ matrix $\boldsymbol{A}$ and m-vector $\boldsymbol{b}$ are given, and the unknown solution n-vector $\boldsymbol{x}$ is to be determined. A solution $\boldsymbol{x}$ has a compressed

representation in a basis $\boldsymbol{X}$ if there exists a k-vector $\boldsymbol{z}$ for which $\boldsymbol{x} \approx \boldsymbol{X}\boldsymbol{z}$, where $\boldsymbol{X}$ is an $n \times k$ basis matrix and $k \ll n$. Although we refer to $\boldsymbol{X}$ as a *basis*, it is not necessarily assumed to be of full rank.

We now formulate an algorithm for exploiting such compression when solving a system of linear algebraic equations. Simply stated, for a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, we sample the column space of $\boldsymbol{A}$ and then project $\boldsymbol{b}$ onto the subspace spanned by the samples, thereby approximating the right-hand-side vector $\boldsymbol{b}$ by a linear combination of the columns of $\boldsymbol{A}\boldsymbol{X}$. This basic method is stated more formally in Algorithm 1.1.

Algorithm 1.1 Basic Compression Method

Given $m \times n$ matrix $\boldsymbol{A}$ and m-vector $\boldsymbol{b}$;
Choose integer k, $k \leq n$;
Choose $n \times k$ matrix $\boldsymbol{X}$;
$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$;
Solve $m \times k$ least squares problem $\boldsymbol{Y}\boldsymbol{z} \cong \boldsymbol{b}$ for $\boldsymbol{z}$;
$\boldsymbol{x} = \boldsymbol{X}\boldsymbol{z}$;

The resulting vector $\boldsymbol{x}$ is an approximate solution because

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{z} = \boldsymbol{Y}\boldsymbol{z} \cong \boldsymbol{b}.$$

As stated in Section 1.1, the most convenient basis for the discretization may not yield a compressed representation of the solution. To distinguish it from the *discretization basis*, we refer to $\boldsymbol{X}$ more specifically as the *compression basis*. The choice of $\boldsymbol{X}$ is crucial to both the accuracy of the approximation and the efficiency of the method. We address in Chapter 2 how to choose $\boldsymbol{X}$, including how the discretization basis affects this choice.

Algorithm 1.1 is formally applicable whether or not $\boldsymbol{A}$ is square or has full rank; in any case, it computes an appropriate approximate solution. If $\boldsymbol{A}$ has full column rank, it approximates the conventional solution (this holds for both square nonsingular and overdetermined full column rank cases). If $\boldsymbol{A}$ lacks full column rank (including the underdetermined case), it will compute a regularized solution. In our investigation, we focus primarily on the square case $m = n$, but the lessons learned are applicable to the other cases as well.

The possibility of a rank deficient $\boldsymbol{Y}$ in Algorithm 1.1, due to rank deficiency of $\boldsymbol{X}$ or of $\boldsymbol{A}$, can be handled easily. If $\boldsymbol{Y}$ is rank deficient, simply compute a basic solution using the QR factorization already computed (see e.g., [2, p. 136]). This is an appropriate remedy as we seek a compressed representation of the solution.

Although Algorithm 1.1 is formally applicable to any linear system, we will focus on

discretized linear systems in order to leverage the properties of the underlying continuous solution function and the particular discretization in choosing an effective compression basis.

1.4 RELATED WORK

If one assumes that $\boldsymbol{A}$ is square and nonsingular and $\boldsymbol{X}$ is of full rank, then Algorithm 1.1 becomes a special case of the *Prototype Projection Method* [3, Algorithm 5.1] by taking $\boldsymbol{V} = \boldsymbol{X}$, $\boldsymbol{W} = \boldsymbol{A}\boldsymbol{X}$, and starting with $\boldsymbol{x} = \boldsymbol{0}$. Our Algorithm 1.1 is substantially more general, however, in that we make *none* of those assumptions. Furthermore, we will consider a number of choices for $\boldsymbol{X}$ that are substantially different from those considered in [3]. The basis matrix $\boldsymbol{X}$ can also be interpreted as a *right preconditioner* [3, Equation (9.2)], but we do not require that $\boldsymbol{X}$ be square and nonsingular, nor do we solve the resulting modified system by a standard iterative method.

Another perspective comes from the numerical solution of ill-posed problems, such as integral equations of the first kind, for which an otherwise uselessly noisy solution can be regularized by restricting the computed solution to a designated subspace. An example is *subspace restricted SVD* [4, 5], in which the truncated SVD (TSVD) method is applied to a projected version of the system matrix $\boldsymbol{A}$, with the projection based on a user-chosen subspace that is specified in advance, which means that its dimension k must be fixed in advance. Our Algorithm 1.1 is simpler and more efficient than SVD-based methods, and most importantly, it lends itself readily to an incremental implementation (as we show in Section 1.6), thereby enabling k to be determined on the fly based on a residual tolerance or other measure of error.

Yet another antecedent from the perspective of ill-posed problems is [6, p. 115], where essentially the same idea as Algorithm 1.1 is suggested as a potential type of regularization for ill-conditioned systems, but it is illustrated using only one choice of basis, namely the discrete cosine transform (DCT), and it is not developed any further.

Our approach also differs from *augmented* or *enriched* Krylov subspace methods (see [7, 8, 9]) in that we dispense with the Krylov subspace entirely and rely solely on the subspace spanned by $\boldsymbol{X}$, which may bear no direct relation to the system matrix $\boldsymbol{A}$. Such methods are used for the numerical solution of a variety of problems, including ill-posed problems.

1.5 RELATED THEORY

While more flexible then Saad's Prototype Projection Method [3, Algorithm 5.1], Algorithm 1.1 is fundamentally an oblique projection method onto $\boldsymbol{X}$ orthogonal to $\boldsymbol{AX}$. As such, we summarize the applicable projection method theory. First, from [3, Proposition 5.3] and taking $\boldsymbol{x}_0 = \boldsymbol{0}$, we know that a vector $\hat{\boldsymbol{x}}$ is the solution if and only if $\|\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2 = \min_{\hat{\boldsymbol{x}} \in \mathcal{K}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2$, where $\mathcal{K}$ is the space spanned by the columns of $\boldsymbol{X}$. Additionally, as a projection method, per [3, Section 5.2.3], $\|\hat{\boldsymbol{x}} - \boldsymbol{x}_*\|_2 \geq \|(I - P_{\mathcal{K}})\boldsymbol{x}_*\|_2$, and thus, the relative error is bounded below by the sine of the acute angle between the true solution $\boldsymbol{x}_*$ and the subspace $\mathcal{K}$. Although this bound is valid for the wide range of bases we investigate in Chapter 2, it is not useful for choosing $\boldsymbol{X}$ or k, as it is a lower, not an upper bound on the error.

1.6 INCREMENTAL FORMULATION

In principle, the number k of compression basis vectors controls the tradeoff between cost and accuracy. Additional accuracy can usually be attained by increasing k, which incurs higher cost. However, one may have no foundation for choosing k a priori. This leads to Algorithm 1.2, a more practical version of the method, which proceeds to build up the compression basis $\boldsymbol{X}$ incrementally, stopping at the k for which a residual tolerance is met. Technically, Algorithm 1.2 is an incremental implemenation of a direct method, *not* an iterative method, as typically for the bases discussed in Chapter 2 the successive vectors $\boldsymbol{x}_k$ are known in advance, and the previous vectors are not required to generate the next vector.

Algorithm 1.2 is stated with a residual tolerance for convenience; for some problems, however, a residual tolerance alone may not be sufficient. A more robust stopping criterion is discussed in Chapter 5. Henceforth, our investigation focuses on this incremental form of the algorithm.

In an incremental version of the method, such as Algorithm 1.2, rank deficiency can be detected by monitoring r_{kk}, the diagonal entry of $\boldsymbol{R}$. If the relative magnitude of r_{kk} falls below some tolerance, simply skip to the next basis vector, if any. Thus, the final $\boldsymbol{Y}$ need never be rank deficient.

Algorithm 1.2 Incremental Compression Method

Given $m \times n$ matrix $\boldsymbol{A}$, m-vector $\boldsymbol{b}$, tolerance tol;
$k = 1$;
Choose n-vector $\boldsymbol{x}_k$; $\boldsymbol{X}_k = [\boldsymbol{x}_k]$;
$\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_k$; $\boldsymbol{Y}_k = [\boldsymbol{y}_k]$;
Solve $m \times 1$ least squares problem $\boldsymbol{Y}_k\boldsymbol{z} \cong \boldsymbol{b}$ for $\boldsymbol{z}$ by QR factorization;
while $(\|\boldsymbol{b} - \boldsymbol{Y}_k\boldsymbol{z}\|_2 > tol)$
    $k = k + 1$;
    Choose n-vector $\boldsymbol{x}_k$; $\boldsymbol{X}_k = [\boldsymbol{X}_{k-1}, \boldsymbol{x}_k]$;
    $\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_k$; $\boldsymbol{Y}_k = [\boldsymbol{Y}_{k-1}, \boldsymbol{y}_k]$;
    Update and extend QR factorization of $\boldsymbol{Y}_{k-1}$ to incorporate $\boldsymbol{y}_k$;
    Solve $m \times k$ least squares problem $\boldsymbol{Y}_k\boldsymbol{z} \cong \boldsymbol{b}$ using updated QR;
$\boldsymbol{x} = \boldsymbol{X}_k\boldsymbol{z}$;

1.7 COMPUTATIONAL COST

To characterize the overall cost for a given $\boldsymbol{X}$, we consider the major components of the method and both their cost per iteration and their cost for k iterations. For consistency with traditional well-known operation counts, we will count only the number of multiplications and assume a similar number of additions, and we exclude lower order terms.

For Algorithm 1.1, k is fixed and the major components of the algorithm are:

- Computing $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$,
- Solving least squares problem $\boldsymbol{Y}\boldsymbol{z} \cong \boldsymbol{b}$,
- Computing resulting solution as $\boldsymbol{x} = \boldsymbol{X}\boldsymbol{z}$.

We assume $\boldsymbol{A}$ contains nnz nonzero entries and the least squares problem is solved by computing the QR factorization using Householder transformations and then solving $\boldsymbol{R}\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{b}$. We assume $\boldsymbol{Q}$ is not explicitly formed, but instead $\boldsymbol{Q}^T\boldsymbol{b}$ is computed by applying the Householder transformations to $\boldsymbol{b}$. As this contributes only lower order terms, the cost of doing so is neglected. The computational costs of the various components of Algorithm 1.1 are summarized in Table 1.1.

For Algorithm 1.2, the individual components of the method are the same; however, as it is an incremental algorithm, we must consider the cost per step and whether any components are now duplicated each step. The main changes with the incremental algorithm are that the QR factorization is now performed incrementally and updated and extended with each additional basis vector, and once the QR factorization is updated, the least squares problem must be solved for each intermediate value of k. Table 1.2 summarizes the approximate

Component	Cost
Compute $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$	$\mathcal{O}(k * nnz)$
Compute $\boldsymbol{Q}\boldsymbol{R}$ factorization to form $\boldsymbol{R}\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{b}$	$k^2m - \frac{1}{3}k^3$
Solve $k \times k$ upper triangular system $\boldsymbol{R}\boldsymbol{z} = \boldsymbol{Q}^T\boldsymbol{b}$ for $\boldsymbol{z}$	$\frac{1}{2}k^2$
Compute $\boldsymbol{x} = \boldsymbol{X}\boldsymbol{z}$	nk

Table 1.1: Computational cost breakdown for Algorithm 1.1.

costs for each component of Algorithm 1.2, both per step k and the full cost per component to reach the final k. To distinguish between the value of k at the current step and the final value of k in the expressions of computational cost, for this comparison only, we use $\hat{k}$ to denote the intermediate values and denote the final value as k.

Component	Cost for Step $\hat{k}$	Total Cost to Reach k
Compute $\boldsymbol{y}_{\hat{k}} = \boldsymbol{A}_{\hat{k}}\boldsymbol{x}_{\hat{k}}$	$\mathcal{O}(nnz)$	$\mathcal{O}(k * nnz)$
Update and extend QR factorization of $\boldsymbol{Y}_{\hat{k}}$	$\hat{k}((m - \hat{k} + 1) + m)$	$k^2m - \frac{1}{3}k^3$
Solve upper triangular system $\boldsymbol{R}_{\hat{k}}\boldsymbol{z}_{\hat{k}} = \boldsymbol{Q}_{\hat{k}}^T\boldsymbol{b}$ for $\boldsymbol{z}_{\hat{k}}$	$\frac{1}{2}\hat{k}^2$	$\frac{1}{6}k^3$
Compute $\boldsymbol{x}_k = \boldsymbol{X}_k\boldsymbol{z}_k$	-	nk

Table 1.2: Computational cost breakdown for Algorithm 1.2.

If $\boldsymbol{X}$ is dense, then the resulting $\boldsymbol{Y}$ will also be dense. But if $\boldsymbol{X}$ is sparse, then $\boldsymbol{Y}$ may be sparse as well, in which case sparsity could potentially be exploited when computing the QR factorization to lower the cost further. In any case, the cost per iteration of Algorithm 1.2 is comparable with that of GMRES [10], plus the cost of generating the basis vectors $\boldsymbol{x}_k$, which varies from trivial to more significant, depending on the chosen basis.

Chapter 2: Compression Bases

A crucial aspect of the compression-based method is the choice of the basis matrix $\boldsymbol{X}$. Among the many possible compression bases, the particular choice will clearly affect the overall performance of the method. Roughly speaking, we want to choose a basis for which Algorithm 1.2 will capture the main features of the solution as quickly as possible. This chapter discusses how to approach choosing a basis, the issues that arise when considering a given basis for use in the compression-based method, and describes each of the individual families of bases we considered.

2.1 CHOOSING A BASIS

In seeking a compressed representation of the solution to the linear system, the method solves for $\boldsymbol{z}$, a vector of coefficients of discrete basis vectors (the columns of $\boldsymbol{X}$) that approximates the solution to the discretized linear system. The components of the solution $\boldsymbol{x}$ to the linear system may be samples of the true solution function (for simple discretizations such as the finite difference method for differential equations or the Nystrom method for integral equations) or coefficients of basis functions (e.g., for collocation or Galerkin discretizations). See Appendices B.1 and C.1 for details of the various discretization methods.

One would expect that using more complex discretizations, where the components of $\boldsymbol{x}$, the solution vector to the discretized linear system, are coefficients in a functional expansion, may affect which bases perform well. To consider this issue in more detail, we first adopt the following terminology. A function defined on a continuous (discrete) domain is called *global* (*dense*) if it is nonzero or nonnegligible on a relatively large portion of its domain. Conversely, it is called *local* (*sparse*) if it is nonzero or nonnegligible on only a small portion of its domain. A basis is said to be *modal* (*nodal*) if it consists of global (local) basis functions. Additionally, some bases, notably hierarchical bases, are combinations of these.

Using this terminology, consider Table 2.1, which can help guide the choice of compression basis, depending on the properties of the underlying continuous solution and the basis for the discretization. In general, the table can be interpreted as follows: to approximate a global function (column 1) using a nodal basis (column 2) yields a dense coefficient vector (column 3), whereas using a modal basis (column 2) may yield a sparse coefficient vector (column 3), depending on the convergence rate of the functional expansion. For a local function, on the other hand, using a nodal basis (column 2) may yield a sparse coefficient vector (column 3), whereas a modal basis (column 2) yields a dense coefficient vector (column 3). Table 2.1

is also applicable for approximating discrete functions, with global and local interpreted as dense and sparse, respectively.

The motivation for this table is drawn from approximation theory. To illustrate, consider interpolating the known true solution function for `greengard-ex1` (see Section B.2) with Chebyshev polynomials (interpolating a global function using a modal basis). Interpolating (with Chebyshev extrema as the interpolation points) gives the coefficients plotted in Figure 2.1, where we see that most of the coefficients are essentially zero, and the nonzero coefficients decay rapidly. This example illustrates how rapid convergence of the functional expansion yields a sparse coefficient vector.

function to be approximated	basis type	coefficient vector
global	nodal	dense
global	modal	sparse
local	nodal	sparse
local	modal	dense

Table 2.1: Guide to choosing a basis.

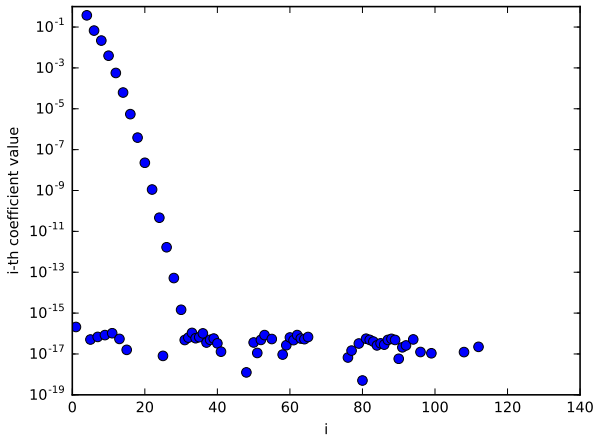


Figure 2.1: Coefficient values when interpolating true solution function for `greengard_ex1` with Chebyshev polynomials.

To solve a problem using the compression-based method, we must consider both the basis used for the discretization and the compression basis used by the solver, so we make two passes through Table 2.1 to help guide the choice of basis. For the first pass, we consider the basis for the discretization, and the table then indicates what to expect for the solution vector to the discretized linear system. Simple discretizations, with no explicit discretization basis, are implicitly local. On the second pass, we consider approximating a discrete function,

namely the solution vector $\boldsymbol{x}$ to the discretized linear system. For the compression-based method to be advantageous, we need to choose a basis in the second pass that yields a sparse coefficient vector. For instance, to approximate a global solution function discretized using the Galerkin method with a nodal basis, we expect a dense solution vector $\boldsymbol{x}$. So, to approximate this dense solution vector (back to column 1), in order to have a chance for a sparse $\boldsymbol{z}$ (column 3), we need a modal basis (column 2). Interestingly, this indicates that for a global solution function (by far the most common case), the compression basis and the discretization basis should be of opposite types to benefit most from the compression-based method, effectively decoupling the two bases.

2.2 ISSUES

In this section, we describe issues that arise when considering a basis for use in the compression method. These issues are applicable in a general sense to both 1D and higher-dimensional problems and are therefore discussed primarily in general terms. Any details that apply only to higher dimensions are discussed in the Section 2.3, after discussing the details of how to extend 1D bases to higher dimensions.

2.2.1 Ordering of Basis Vectors

For the compression-based method to realize the full benefit of a compressed representation of the solution, the relevant discrete basis vectors (those with non-negligible coefficients) must come early in the incremental process, and thus the *ordering* of the basis vectors is crucial to the effectiveness of the method. Fortunately, many of the bases derived from approximation theory and signal processing have a natural ordering that works well for most applications. For example, consistent with their convergence theory, polynomial bases are naturally ordered from lower to higher degrees, and sinusoidal bases are naturally ordered from lower to higher frequencies. In practice, the terms of lower degree or frequency usually carry the main signal, while the terms of higher degree or frequency tend to represent unwanted noise. For this reason, the compression-based method can have a regularizing effect, especially for poorly conditioned problems. Similarly, for hierarchical or multiresolution bases, ordering from coarser to finer levels is the natural ordering. We expect the same natural orderings motivated by approximation theory and signal processing to generally be best for the compression-based method, a hypothesis we investigate in Section 3.5. We will see that for some highly localized bases, however, the best ordering is less clear and may depend on the specific problem.

2.2.2 Evaluation Points

As discussed previously, many compression bases are motivated by continuous functions considered in approximation theory. For use in the compression-based method, however, continuous basis functions must be converted into discrete basis vectors by evaluating each continuous function at a discrete set of points. In particular, for 1D problems with continuous basis functions $\phi_j(t)$, $j = 1, \ldots, k$, the entries of the corresponding discrete $n \times k$ basis matrix $\boldsymbol{X}$ are given by

$$x_{i,j} = \phi_j(t_i), \tag{2.1}$$

where $t_i, i = 1, \ldots, n.$ are the *evaluation* points.

Using equally-spaced evaluation points t_i in Equation 2.1 may be the most natural and convenient choice. However, any distinct evaluation points could be chosen. From approximation theory, we know that accuracy and conditioning are often improved by using zeros of orthogonal polynomials, whose distribution tends to be clustered toward the endpoints of the interval domain. Therefore, in addition to equally-spaced evaluation points, we explore the use of evaluation points based on Chebyshev and Legendre polynomials.

For bases that are inherently discrete, such as those motivated by digital signal processing (DSP), there is no explicit need to choose evaluation points. However, many of these bases are derived from continuous basis functions evaluated at discrete points, which may or may not be optminally chosen for use as a compression basis. Thus, in addition to the conventional evaluation points for such bases, we also consider alternative choices.

As discussed previously, we focus on solving linear systems resulting from the discretization of continuous problems. Therefore, each entry of the solution vector $\boldsymbol{x}$ corresponds to a specific degree of freedom, which may correspond to a specific coordinate or to a specific discretization basis function (depending on the discretization). By virtue of the ordering assumed during the discretization and formation of $\boldsymbol{A}$, these degrees of freedom have a specific ordering. As the effectiveness of the compression method relies on detecting a pattern in the solution vector $\boldsymbol{x}$, it is important for the ordering of the evaluation points to be compatible with the ordering of the degrees of freedom of the discretization points. For 1D problems, when the degrees of freedom correspond to spatial coordinates, there is a common natural ordering for both the coordinates of the degrees of freedom and the evaluation points. Therefore, incompatibility is generally not a concern in 1D. We discuss compatibility for higher dimensions in Section 2.2.2.

We explore the effect of the specific discretization on the choice of evaluation points and investigate the impact of the choice of evaluation points on the effectiveness of the compression-based method in Section 3.4 and Section 3.5, respectively.

2.2.3 Orthogonality

For both theoretical and numerical reasons, it may be advantageous for the basis matrix $\boldsymbol{X}$ to be orthogonal. Many of the discrete bases we consider are derived from orthogonal functions (via either specified transforms or evaluation of continuous basis functions) and are therefore orthogonal or nearly orthogonal. (Note that sampling continuous, orthogonal functions does not guarantee that the resulting discrete basis vectors are orthogonal.) If desired, any basis can be orthogonalized on the fly as in Algorithm 2.3 (at substantial additional expense). It is crucial to emphasize, however, that the method does not require an orthogonal (or orthogonalized) $\boldsymbol{X}$. The method can even accommodate rank deficient $\boldsymbol{X}$ by simply discarding redundant samples (as revealed by the QR factorization).

Algorithm 2.3 Orthogonal Compression Method

Given $m \times n$ matrix $\boldsymbol{A}$, m-vector $\boldsymbol{b}$, tolerance tol;
$k = 1$;
Choose n-vector $\boldsymbol{x}_k$; $\boldsymbol{X}_k = [\boldsymbol{x}_k]$;
$\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_k$; $\boldsymbol{Y}_k = [\boldsymbol{y}_k]$;
Solve $m \times 1$ least squares problem $\boldsymbol{Y}_k\boldsymbol{z} \cong \boldsymbol{b}$ for $\boldsymbol{z}$ by QR factorization;
while $(\|\boldsymbol{b} - \boldsymbol{Y}_k\boldsymbol{z}\|_2 > tol)$
    Choose n-vector $\boldsymbol{s}$;
    Solve $n \times k$ least squares problem $\boldsymbol{X}_k\boldsymbol{t} \cong \boldsymbol{s}$ using QR factorization;
    $k = k + 1$;
    $\boldsymbol{x}_k = \boldsymbol{s} - \boldsymbol{X}_{k-1}\boldsymbol{t}$; $\boldsymbol{X}_k = [\boldsymbol{X}_{k-1}, \boldsymbol{x}_k]$;
    $\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_k$; $\boldsymbol{Y}_k = [\boldsymbol{Y}_{k-1}, \boldsymbol{y}_k]$;
    Update and extend QR factorization of $\boldsymbol{Y}_{k-1}$ to incorporate $\boldsymbol{y}_k$;
    Solve $n \times k$ least squares problem $\boldsymbol{Y}_k\boldsymbol{z} \cong \boldsymbol{b}$ using updated QR;
$\boldsymbol{x} = \boldsymbol{X}_k\boldsymbol{z}$;

2.3 2D EXTENSIONS OF BASES

Most of the bases to be discussed in Section 2.4, are derived from 1D continuous functions or 1D discrete transforms and therefore lend themselves naturally to 1D problems. For systems of linear equations resulting from 2D problems, the 1D bases can be extended to 2D by forming tensor products of the 1D basis functions [11, p. 118]. Specifically, given a set of 1D basis functions $\phi_j(t)$, $j = 1, \ldots, n$, we define a set of 2D basis functions as

$$\psi_{i,j}(s,t) = \phi_i(s)\phi_j(t), \quad i,j = 1, \ldots, n. \tag{2.2}$$

2.3.1 Ordering of Basis Vectors in 2D

As with 1D basis functions, the ordering of the basis vectors for 2D basis functions is crucial to the effectiveness of the method. In 1D, most of the bases have natural orderings, often representing low-to-high frequency. However, as the 2D basis functions have multiple indices, they have no default linear ordering. The doubly-indexed basis functions can be viewed as a 2D grid of basis function such as the following for $n = 4$.

$$\begin{bmatrix} \psi_{0,0} & \psi_{0,1} & \psi_{0,2} & \psi_{0,3} \\ \psi_{1,0} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{2,0} & \psi_{2,1} & \psi_{2,2} & \psi_{2,3} \\ \psi_{3,0} & \psi_{3,1} & \psi_{3,2} & \psi_{3,3} \end{bmatrix}$$

While many linear orderings over a set of doubly-indexed basis functions can be defined, an ordering that captures the overall low-to-high frequency trend that defines the natural ordering for the sets of 1D basis functions is desirable. We next define a few possible linear orderings of 2D basis functions, and we investigate the impact of the choice of ordering on the effectiveness of the method for 2D test problems in Section 4.3.

Diagonal Ordering

One such traversal of the 2D grid of doubly-indexed basis functions traverses in order of the diagonals starting in the upper left and ending in the lower right. We call this the *diagonal* ordering, and it is defined by the following rules:

1. Order the functions by the sum of the two indices, from smallest to largest.
2. Within a subset having the same sum of indices, order the functions by the maximum of the two indices, smallest to largest.
3. Within a subset having the same sum of indices and maximum index, order the functions by the first index, smallest to largest.

For $n = 4$, a graphical depiction of the diagonal ordering is shown in Figure 2.2.
This results in the overall ordering for $n = 4$

$$\psi_{0,0}, \psi_{0,1}, \psi_{1,0}, \psi_{1,1}, \psi_{0,2}, \psi_{2,0}, \psi_{1,2}, \psi_{2,1}, \psi_{0,3}, \psi_{3,0}, \psi_{2,2}, \psi_{1,3}, \psi_{3,1}, \psi_{2,3}, \psi_{3,2}, \psi_{3,3}.$$

$$\begin{bmatrix} \psi_{0,0} & \psi_{0,1} & \psi_{0,2} & \psi_{0,3} \\ \psi_{1,0} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{2,0} & \psi_{2,1} & \psi_{2,2} & \psi_{2,3} \\ \psi_{3,0} & \psi_{3,1} & \psi_{3,2} & \psi_{3,3} \end{bmatrix} \approx \begin{bmatrix} 0 & 1 & 4 & 8 \\ 2 & 3 & 6 & 11 \\ 5 & 7 & 10 & 13 \\ 9 & 12 & 14 & 15 \end{bmatrix}$$

Figure 2.2: Graphical depiction of diagonal ordering for 2D basis functions.

Chevron Ordering

Another ordering, which we call the *chevron* ordering, can be obtained by traversing through chevron patterns, starting in the upper left and ending in the lower right entries of the matrix. This ordering corresponds to switching rules (1) and (2) in Section 2.3.1, resulting in the following set of rules:

1. Order the functions by the maximum of the two indices, smallest to largest.
2. Within a subset having the same maximum index, order the functions by the sum of the two indices, smallest to largest.
3. Within a subset having the same sum of indices and maximum index, order the functions by the first index, smallest to largest.

This ordering is depicted by the diagram in Figure 2.3 for $n = 4$, which illustrates the chevron pattern.

$$\begin{bmatrix} \psi_{0,0} & \psi_{0,1} & \psi_{0,2} & \psi_{0,3} \\ \psi_{1,0} & \psi_{1,1} & \psi_{1,2} & \psi_{1,3} \\ \psi_{2,0} & \psi_{2,1} & \psi_{2,2} & \psi_{2,3} \\ \psi_{3,0} & \psi_{3,1} & \psi_{3,2} & \psi_{3,3} \end{bmatrix} \approx \begin{bmatrix} 0 & 1 & 4 & 9 \\ 2 & 3 & 6 & 11 \\ 5 & 7 & 8 & 13 \\ 10 & 12 & 14 & 15 \end{bmatrix}$$

Figure 2.3: Graphical depiction of chevron ordering for 2D basis functions.

This results in the overall ordering for $n = 4$

$$\psi_{0,0}, \psi_{0,1}, \psi_{1,0}, \psi_{1,1}, \psi_{0,2}, \psi_{2,0}, \psi_{1,2}, \psi_{2,1}, \psi_{2,2}, \psi_{0,3}, \psi_{3,0}, \psi_{1,3}, \psi_{3,1}, \psi_{2,3}, \psi_{3,2}, \psi_{3,3}.$$

Circular Ordering

The diagonal and chevron patterns represent ordering by the 1-norm and ∞-norm of the index pairs, respectively. An alternative ordering, based on the idea of circular truncation of basis functions for spectral methods [11, p. 119], yields circular arc patterns by instead

ordering by the 2-norm of the index pairs. We call this the *circular* ordering, and it is defined by the following rules:

1. Order the functions by the 2-norm of the index pair (i.e., $\sqrt{i^2 + j^2}$), smallest to largest.
2. Within a subset having the same 2-norm of the index pairs, order the functions by the first index, smallest to largest.

This results in the overall ordering for $n = 5$ (shown for $n = 5$ in order to illustrate the difference in the pattern from the chevron ordering)

$$\psi_{0,0}, \psi_{0,1}, \psi_{1,0}, \psi_{1,1}, \psi_{0,2}, \psi_{2,0}, \psi_{1,2}, \psi_{2,1}, \psi_{2,2}, \psi_{0,3}, \psi_{3,0}, \psi_{1,3}, \psi_{3,1},$$
$$\psi_{2,3}, \psi_{3,2}, \psi_{0,4}, \psi_{4,0}, \psi_{1,4}, \psi_{4,1}, \psi_{3,3}, \psi_{2,4}, \psi_{4,2}, \psi_{3,4}, \psi_{4,3}, \psi_{4,4}.$$

Alternative Orderings

For basis functions ordered by increasing degree or frequency, the chevron and diagonal orderings generally favor lower degree over higher degree, while also favoring more balanced blends over more skewed blends. However, one could simply traverse the 2D grid of basis functions in a row-major or column-major order, which corresponds to favoring a lower degree in one dimension over the other. We refer to these as *row-major* and *column-major*; however, in general, we would not expect these to be competitive with the diagonal or chevron orderings that more comprehensively favor lower degrees.

2.3.2 Evaluation Points for 2D Bases

As discussed in Section 2.2.2, to obtain discrete basis vectors from continuous basis functions, the continuous basis functions are evaluated at discrete points, and the evaluation points should be compatible with the discretization points. For 2D problems, the evaluation points now lie in a 2D domain. For standard evaluation points, such as equally-spaced points, Chebyshev extrema, etc., these evaluation points form a grid in the 2D domain. Evaluating the 2D basis functions described in Section 2.3 at the grid of evaluation points results in 2D basis vectors.

For use in the compression-based method for solving linear systems, however, the basis vectors must be 1D. For standard evaluation points forming a 2D grid, it may seem natural simply to flatten the grid of evaluation points, say in row- or column-major order. While this may be natural for some discretizations that take a similar approach with ordering

the degrees of freedom, for 2D problems with more complex discretizations the degrees of freedom for the discretized problem may not be ordered in a systematic spatial ordering based on their corresponding coordinates. Furthermore, for some discretizations, such as 2D finite elements with an irregular mesh, the spatial coordinates corresponding to the degrees of freedom may be irregularly placed throughout the domain.

Because of this, for 2D problems where the degrees of freedom correspond to spatial coordinates, there is no natural linear ordering of evaluation points that ensures compatibility with the ordering of the degrees of freedom. To ensure compatibility, some knowledge of the discretization is therefore necessary. In particular, utilizing the spatial coordinates corresponding to the degrees of freedom as evaluation points, and in the same order, will ensure compatibility. An example illustrating the importance of compatibility of evaluation points for 2D problems is illustrated in Section 4.4.

2.4 BASES CONSIDERED

We consider bases from a variety of sources. Many of the bases considered are motivated by *approximation theory*, which is concerned with the development of accurate and efficient approximations to continuous functions, (see, e.g., [12, 13, 14]). For a linear system whose solution represents an underlying continuous function, approximation theory provides a plentiful source of potential bases. Another source of potential bases is digital signal processing, which seeks efficient representations of discrete signals, often for purposes of compression or filtering.

In the subsequent sections, we describe each of the individual families of bases considered. Results with many of these families of bases will be presented in Chapters 3 and 4. For brevity, we do not include here all of the detailed equations and methods for generating the bases; full details can be found in the references cited. This is primarily an overview of the different bases we consider for the purposes of understanding the performance of the compression-based method for solving linear systems.

Most of the bases discussed are illustrated in Appendix A (in some cases only one of several versions is illustrated). When the original bases are directly defined as continuous basis functions, they are illustrated as continuous functions. Bases that are directly defined as discrete basis vectors are illustrated as discrete using stem plots.

2.4.1 Random Sampling

As stated in Chapter 1, the compression-based method samples the column space of $\boldsymbol{A}$ and projects $\boldsymbol{b}$ onto the subspace spanned by the samples. Randomized methods based on random sampling have been shown to be useful in various other areas of numerical linear algebra, such as the randomized singular value decomposition (SVD) [15]. However, as our compression-based method relies on detecting a pattern in the solution to find a compressed representation, we do not expect that random sampling will be efficient. We test this hypothesis by considering random bases drawn from various distributions in Section 3.5.9.

2.4.2 Polynomials

Various polynomial basis functions are assessed as potential bases for our method. For orthogonal polynomials, we consider both the Legendre polynomials and the Chebyshev polynomials. In addition to these common orthogonal polynomials, we also consider the Lagrange polynomial basis functions, also known as the cardinal polynomials, and the Bernstein polynomial basis functions. For the Bernstein polynomial basis functions, we specifically consider the n Bernstein polynomials of degree $n-1$. Note that for the Lagrange basis functions a set of points, t_i, $i = 1, \ldots, n$, must be chosen to define the basis functions (in addition to choosing any evaluation points) and the i-th basis function is 1 at t_i and 0 at t_j, where $j \neq i$. To avoid ill-conditioning when evaluating the basis functions (which becomes a problem even with fairly small n), the Lagrange basis is defined using the Chebyshev extrema. These four sets of polynomials are illustrated for $n = 8$ in Appendix A.

2.4.3 Piecewise Polynomials

Another family of bases considered is piecewise polynomial bases. We consider both the standard nodal piecewise linear basis function (i.e., "hat functions"), which are commonly used in finite element methods, and hierarchical piecewise linear basis functions, which have a variety of applications including sparse grids [16, 17] and multigrid methods [18]. In the hierarchical piecewise linear basis, the basis vectors are defined using nested grids. Each basis function is still piecewise linear, but the grid is refined at each successive level, causing the individual basis functions to be nonzero over less of the domain as the level increases. Illustrations of both of these bases for $n = 9$ are shown in Appendix A. The hierarchical basis is shown with the basis vectors separated by level.

2.4.4 Sinusoids

A variety of sinusoidal bases are considered. Specifically, we consider bases related to the discrete cosine transfrom (DCT), the discrete sine transform (DST), and the discrete Hartley transform (DHT).

The DCT, originally presented in [19] with applications to pattern recognition and filtering, has been used in many areas of science and engineering, such as JPEG image compression. The basis functions for the DCT are actually a class of discrete Chebyshev polynomials [19]. Additional details regarding the DCT can be found in [20, 21, 22]. The DST is another popular transform due to its use in transform coding [23].

There are eight different versions of both the DCT and the DST. The eight versions represent slight differences in the treatments of the boundaries (specifically whether they are treated as odd or even and around which point – endpoint or midpoint). The formulas for all eight versions of both the DST and DCT can be found in [24]. One version of the DCT and one version of the DST are illustrated in Appendix A. In addition to the eight canonical DCT (DST) versions, we also consider simply cosines (sines) of increasing frequency evaluated at the various evaluation points discussed in Section 2.2.2.

The DHT, originally presented in [25], uses both sines and cosines, and is closely related to the DFT (discrete Fourier transform), with the primary difference that it takes real inputs to real outputs. As we are attempting to approximate a real solution vector $\boldsymbol{x}$, this is a more appropriate basis to consider than the DFT. Specific formulas for the DHT can be found in [20, 22]. The DHT is also illustrated in Appendix A.

2.4.5 Square Waves

Square wave bases are made up of square pulses, typically with two allowed states (e.g., $-1, 1$ or $0, 1$), and have many applications such as signal processing, image processing, and compression. The primary square wave bases we consider are Walsh, Haar, and Slant transform basis functions.

The Walsh functions form a complete orthogonal set over the unit interval and are the basis functions for the Walsh-Hadamard transform, which has found numerous applications such as signal processing, pattern recognition, filtering, compression, etc. [21]. The Walsh basis functions have three commonly used orderings: sequency, Hadamard, and Paley. The Walsh functions take one of two values: -1 or 1. Thus, the Walsh functions are nonzero over the whole domain. Additional details regarding generating the Walsh functions can be found in [21, 26].

The Haar functions also form a complete orthogonal set of functions and are the basis functions for the Haar transform. Unlike the Walsh functions, the Haar functions are nonzero over only a small portion of the domain. Additional details on generating the Haar basis functions can be found in [21]. Haar functions (and the Haar transform matrix) are commonly found in both normalized and unnormalized forms. However, for the compression-based method, normalization of a basis vector by a constant has no impact on the effectiveness, as it merely changes the value of the corresponding coefficient in $\boldsymbol{z}$. Thus, we consider only the unnormalized Haar basis vectors.

The Slant transform is slightly different than Walsh and Haar in that, while still being made up of square waves, it contains some stair-like waveforms. The Slant transform has been applied to image processing with the idea that it can more efficiently represent linear changes in the image brightness [27]. Details on constructing the Slant transform basis matrix can be found in [21, 27].

Illustrations of each of these sets of basis functions can be found in Appendix A. Additionally, various orderings for the Walsh functions are also illustrated.

2.4.6 Wavelets

The Haar transform discussed above is one of the simplest wavelet transforms. We also consider bases derived from the wavelet transform with other wavelets, such as those in the Daubechies or Symlets families (see the "Wavelet Browser" in [28] for details on specific wavelets). We specifically consider bases derived from the multilevel wavelet decomposition in PyWavelets [28]. Additionally, we look at other bases derived from wavelets. The Haar transform and the Walsh-Hadamard transform are closely related, with the Walsh-Hadamard transform being considered a wavelet packet coming from the full tree decomposition [29, p. xiv]. In a similar manner, we consider additional bases obtained from the full tree of the wavelet packet decomposition in [28] with other wavelets. Illustrations of basis functions obtained from the multilevel wavelet decomposition and the full tree wavelet packet decomposition for a single wavelet can be found in Appendix A.

2.4.7 Gaussians

Gaussian functions have a variety of applications in science, engineering, mathematics, and statistics. We consider both nodal and hierarchical bases derived from Gaussian functions. Gaussian functions are specifically considered because the amount of overlap between them can be controlled by adjusting a parameter c. The overlap parameter c is inversely

proportional to the width of the Gaussian; additional details on the relationship can be found in Appendix A. Increasing c decreases the width of the Gaussian, thereby decreasing the overlap, whereas decreasing c increases the width of the Guassian and the amount of overlap. Illustrations of nodal and hierarchical Gaussian bases with various amounts of overlap can be found in Appendix A.

2.4.8 Sinc Functions

Sinc functions are commonly used in signal processing and interpolation of signals from samples, so we consider them as a possible compression basis. As with Gaussians, we consider both nodal and hierarchical bases composed of sinc functions. Illustrations of the nodal and hierarchical sinc functions can be found in Appendix A.

Chapter 3: 1D Numerical Results

In this chapter we present numerical results for the compression-based method on a variety of 1D test problems. We first discuss the test problems and the experimental setup, and then present a range of results to test and verify specific hypotheses and illustrate the overall effectiveness of the method.

3.1 1D TEST PROBLEM DETAILS

We are primarily concerned with linear systems resulting from discretizing continuous problems. In this chapter, we focus our investigation on solving discretized integral equations and differential equations in one dimension (results for 2D test problems are presented in Chapter 4). Specifically, for integral equations, we focus on linear systems resulting from the discretization of Fredholm integral equations of the first kind in one dimension using Nystrom (quadrature), collocation, and Galerkin discretizations. For differential equations, we focus on linear systems resulting from the discretization of second-order, linear, constant coefficient boundary value problems (BVPs) in one dimension using finite difference, collocation, and Galerkin discretizations.

Due to the nature of the compression-based method, many linear system test problems in the literature are unsuitable for our purposes. A substantial fraction of the linear system test problems in the literature artificially generate the right-hand-side vector $\boldsymbol{b}$ by multiplying the matrix resulting from the discretization with either the constant vector $\boldsymbol{e}$ of all ones or a random vector. In the first case, any compression basis with a constant vector as the first basis vector will find the exact solution after one step, making the problem trivially easy. In the second case (a random solution vector), it is the opposite – a problem that is uncharacteristically hard. In this case, there is by definition no pattern to the solution vector, so we would not expect it to have a compressed representation. Because of these concerns, we employ test problems found in the literature that are fully specified without such arbitrary choices. In addition, we also use a few test problems, denoted with names `designed_*`, that are designed by us to have specific solutions for use in targeted testing of specific hypotheses.

An overview of the various discretizations and detailed formulas for each test problem can be found in Appendices B and C for 1D integral equation and boundary value problems, respectively.

3.2 EXPERIMENTAL SETUP

All results presented are obtained using Python with IEEE double precision arithmetic. Functions and data structures from NumPy [30, 31] and SciPy [32] are utilized wherever appropriate. Unless otherwise stated, all boundary value problems have Dirichlet boundary conditions. For each test, Algorithm 1.2 is run until a relative residual tolerance of 10^{-15} has been reached or until $k = n$. This is done to illustrate the full behavior of the method, although in practice one would stop as soon as a satisfactory solution has been attained (see Chapter 5).

For convenience of implementation, the bases and problems are translated to the domain $[0, 1]$ in some instances. Additionally, for some tests involving non-simple discretizations, the discretization basis functions must be homogeneous. In such cases, the Chebyshev discretization basis is actually a modified Chebyshev basis, where the basis functions are modified to satisfy homogeneous boundary conditions (see Section A.1.1 for details).

Throughout these results, when we discretize via finite differences we use equally-spaced mesh points, and for composite trapezoid quadrature we use equally-spaced nodes. For Clenshaw-Curtis quadrature we use Chebyshev extrema as the nodes wherever possible, and use Chebyshev roots as the nodes only when necessary (for problems with open domains).

Most results are illustrated with plots of the relative error and relative residual. The residual is computed as the discrete 2-norm of the residual vector from the solution to the linear system. For simple discretizations, where the entries of the solution vector are samples of the continuous solution, the error vector is computed using the solution vector from the compression-based method and the vector obtained by sampling the true solution function at the discretization points. For other discretizations (where the components of the solution to the linear system are coefficients of basis functions), the true solution function and the approximate solution function from the coefficients obtained via the compression-based method are both sampled at 200 evenly spaced points and an error vector is computed. These 200 evenly spaced points are chosen because they are unlikely to be biased towards any given set of discretization points (since they will not agree with any discretization points or quadrature points for our tests). For either type of discretization, the error is then computed using the discrete 2-norm of the error vector. Because we compare the computed approximate solution with the known true solution of the original continuous problem, the error reported includes discretization error as well as the error in solving the discretized linear system.

Many of these results compare the performance of the compression-based method with GMRES and TSVD. Although it is possible for the compression-based method to reproduce GMRES and TSVD by an appropriate choice of basis, for purposes of comparing their

effectiveness with the compression-based method, we use standard native implementations of these methods. This is done to be as fair and unbiased as possible when comparing the results. We specifically compare with GMRES and TSVD as they are similarly general methods (i.e., they do not rely on any special properties of the linear system such as symmetry of $\boldsymbol{A}$.)

3.3 EXAMPLE SOLUTIONS

Before delving into specific results, we first show a couple of examples to illustrate how and why the compression-based method can work well. Figure 3.1 shows an example comparing the sequence of solutions obtained by Algorithm 1.2 as k increases, with Chebyshev polynomials evaluated at equally-spaced points as the compression basis, and the true solution for the integral equation problem `gravity_c1`, discretized using the Nystrom method with composite trapezoid quadrature. Although this problem is discretized with $n = 128$, we see that an excellent solution is already obtained at $k = 6$. Note that we are not simply approximating the continuous true solution directly using Chebyshev polynomials, but only indirectly by solving the discretized linear system using the compression-based method of Algorithm 1.2. Similarly, Figure 3.2 shows the sequence of solutions obtained by Algorithm 1.2, with Chebyshev polynomials evaluated at equally-spaced points as the compression basis, compared with the true solution for the BVP `greengard-ex1`, discretized using finite differences with $n = 128$. Results are presented only for odd k as only those basis functions have an effect since the solution is even. We see that with $k = 11$, an excellent solution is obtained. These examples illustrate the ability of the compression-based method to obtain useful solutions with relatively small k; they are not meant to be taken as representative of the behavior for all of the test problems considered.

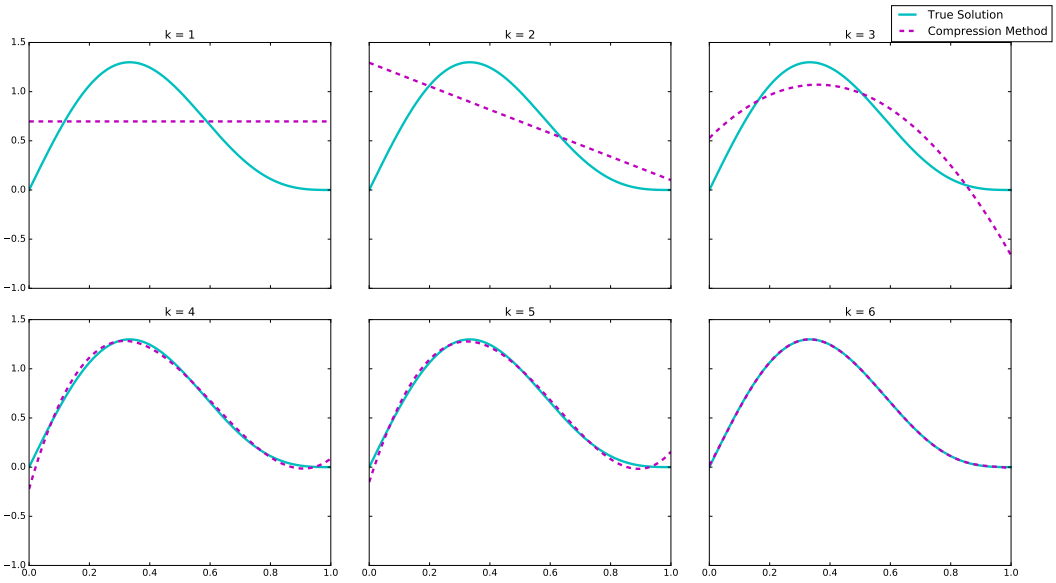


Figure 3.1: Plots of solution obtained by compression-based method for $k = 1, \ldots, 6$ for `gravity_c1` discretized using Nystrom method using composite trapezoid quadrature with $n = 128$ (as discussed in Section 3.3). Chebyshev polynomials evaluated at equally-spaced points are used as the compression basis.

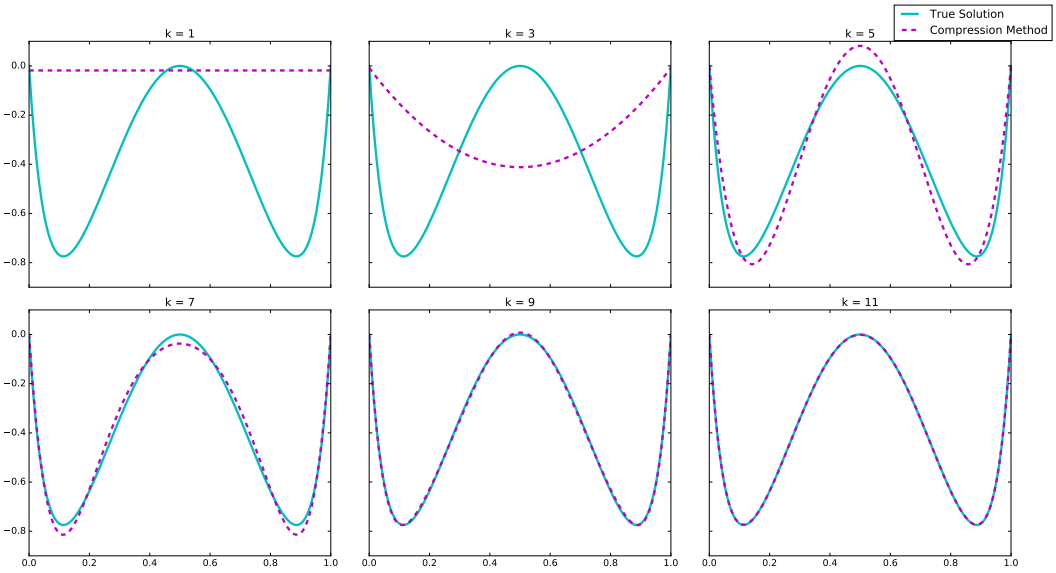


Figure 3.2: Plots of solution obtained by compression-based method for $k = 1, 3, 5, \ldots, 11$ for `greengard-ex1` discretized using finite differences with $n = 128$ (as discussed in Section 3.3). Chebyshev polynomials evaluated at equally-spaced points are used as the compression basis.

3.4 TABLE VERIFICATION

To verify the various paths for the two passes through Table 2.1, we use the integral equation problem `designed_sine_ie` and the boundary value problem `designed_sine_bvp`, which both have global solutions. While problems with global solutions are most common, in

order to verify the table fully, we also use one problem with a local solution, `greengard-ex2`. For each test, we consider both collocation and Galerkin discretizations. Whenever applicable, both discretizations are tested with two sets of bases, piecewise linear (hierarchical and nodal) and polynomials (Chebyshev and Lagrange), for the modal and nodal discretization and compression bases.

Figure 3.3 and Figure 3.4, illustrate the full results (both sets of bases for both discretizations) for `designed_sine_ie` and `designed_sine_bvp`, respectively. The left and right columns of subfigures in Figure 3.3 and Figure 3.4 correspond to discretizing with nodal and modal discretization bases, respectively. Considering the left column of subfigures in Figure 3.3 and Figure 3.4, Table 2.1 suggests that for a problem with a global solution discretized with a nodal basis, a modal compression basis is necessary to have a chance of a sparse coefficient vector, and the results for the various discretizations and nodal discretization bases confirm that the modal compression bases (hierarchical piecewise linear or Chebyshev) perform best. Similarly, Table 2.1 suggests that for a problem with a global solution discretized with a modal basis, a nodal compression basis is required to have a chance of a sparse coefficient vector, and the results in the right column of subfigures in Figure 3.3 and Figure 3.4 confirm that the nodal compression bases (nodal piecewise linear or Lagrange) indeed perform best.

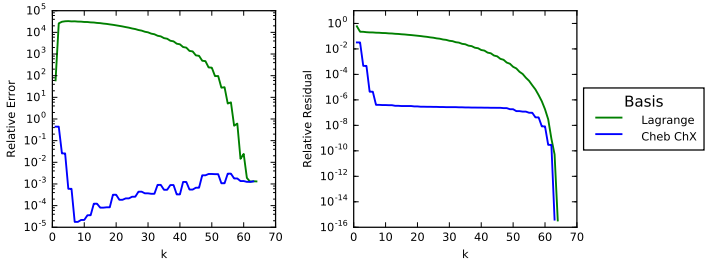
Similarly, for the BVP `greengard-ex2`, Figure 3.5 illustrates the full results (polynomial bases for the collocation discretization and both sets of bases for Galerkin discretizations), with the left and right columns of subfigures corresponding to discretizing with nodal and modal discretization bases, respectively. Considering Figure 3.5a, Figure 3.5c, and Figure 3.5e, we see that for a local solution problem discretized with a local basis, a local compression basis performs best, which agrees with Table 2.1. In particular, we also see the importance of ordering. This is a boundary layer problem (see Figure C.4), whose solution is nonzero only near the left and right boundaries. The Lagrange and nodal piecewise linear compression bases taken in outside-in (OI) order perform well, whereas the same nodal compression bases taken in the usual left-to-right ordering do not achieve a good solution until $k = n$.

Considering Figure 3.5b, Figure 3.5d, and Figure 3.5f, Table 2.1 suggests that for a problem with a local solution discretized with a modal basis, a modal compression basis is required to have a chance of a sparse coefficient vector. This is one test that does not support the table as the nodal compression bases, composed of the Lagrange polynomials or nodal piecewise linear polynomials, performs better than the corresponding modal compression basis. However, the table is merely a guide and to perform well one would need an appropriate global basis. For Figure 3.5b, Figure 3.5d, and Figure 3.5f, the solutions to the discretized linear systems are

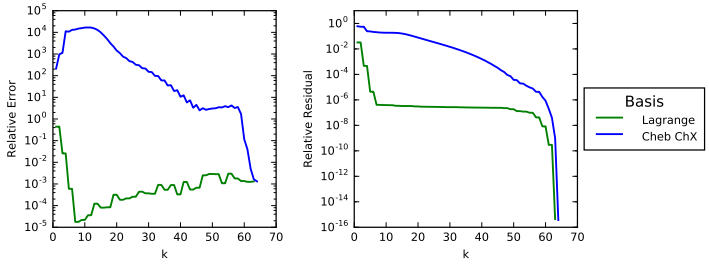
vectors of coefficients of the modal basis functions. Although these vectors are likely not to be overly sparse, to perform well we would need to have a discrete compression basis for which the vectors of coefficients to the modal basis functions have compressed representations. It is not unreasonable that these coefficients may not be approximated well by the standard global bases.

It is important to remark that, although we present results both for problems with global and local solutions in order to test Table 2.1 fully, in practice global solutions are far more common, and for problems with global solutions, Table 2.1 has proved consistently valid in our tests.

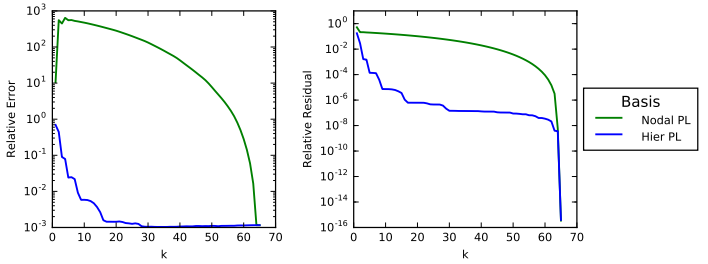
It is crucial to recognize that the right column of subfigures in Figure 3.5 is simply considered for completeness, and a global discretization is an unconventional, and likely impractical, choice for a problem with a local solution. Additionally, although the local solution problem considered is a BVP, these results should not be taken as differentiating the validity of the table between these two classes of problems. The vast majority of our tests confirm Table 2.1 for both integral equations and boundary value problems.



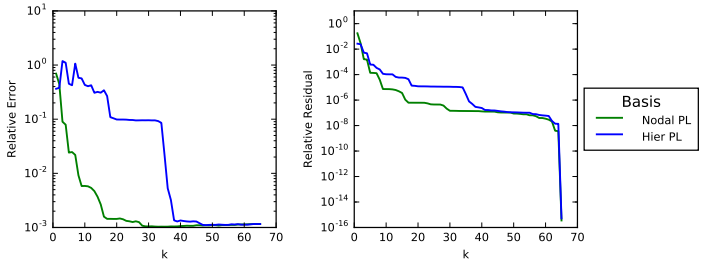
(a) Discretized with collocation using Lagrange basis.



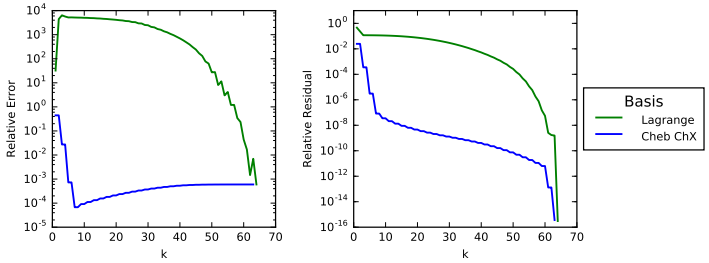
(b) Discretized with collocation using Chebyshev basis.



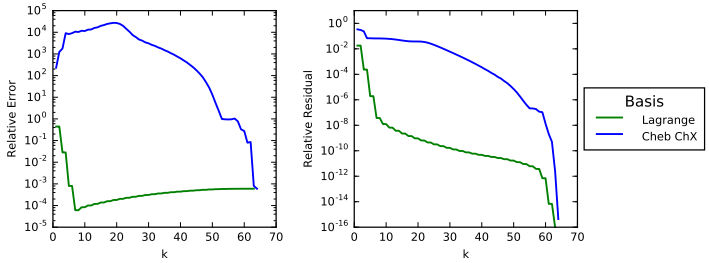
(c) Discretized with collocation using nodal PL basis.



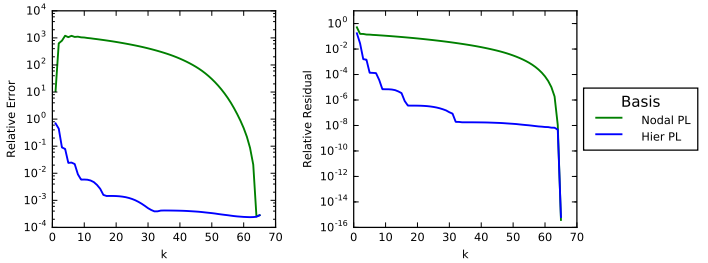
(d) Discretized with collocation using hierarchical PL basis.



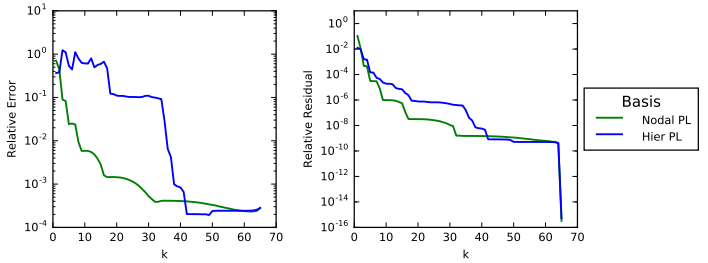
(e) Discretized with Galerkin using Lagrange basis.



(f) Discretized with Galerkin using Chebyshev basis.

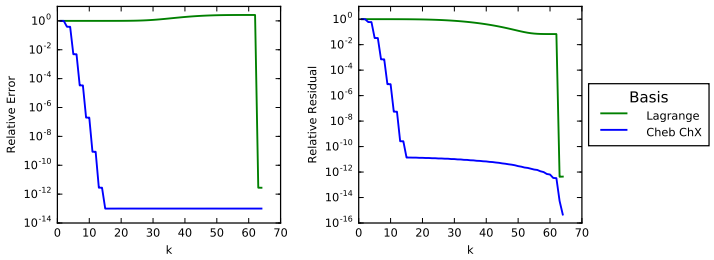


(g) Discretized with Galerkin using nodal PL basis.

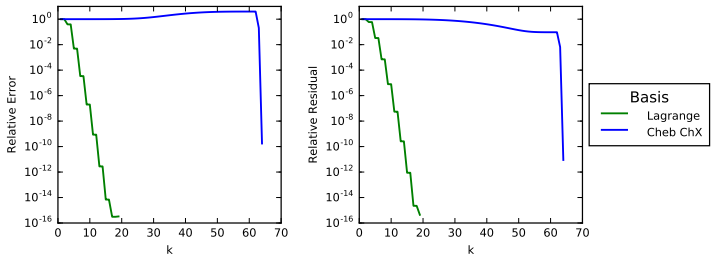


(h) Discretized with Galerkin using hierarchical PL basis.

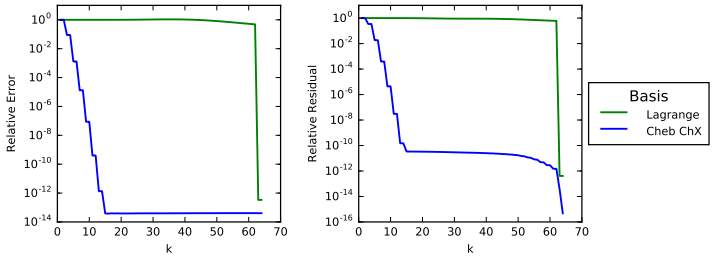
Figure 3.3: Global vs. local basis results for `designed_sine_ie` as discussed in Section 3.4.



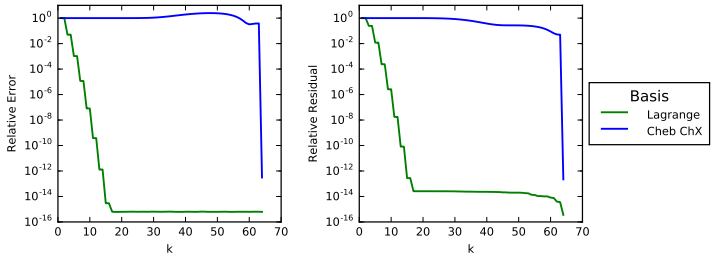
(a) Discretized with collocation using Lagrange basis.



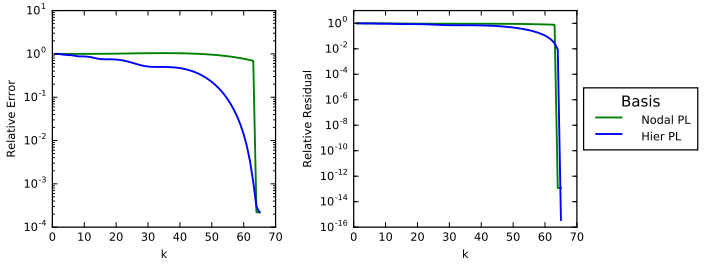
(b) Discretized with collocation using Chebyshev basis.



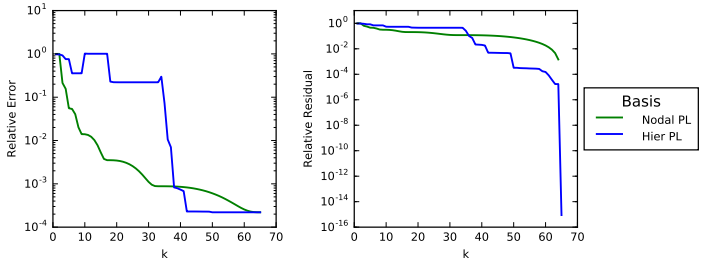
(c) Discretized with Galerkin using Lagrange basis.



(d) Discretized with Galerkin using Chebyshev basis.

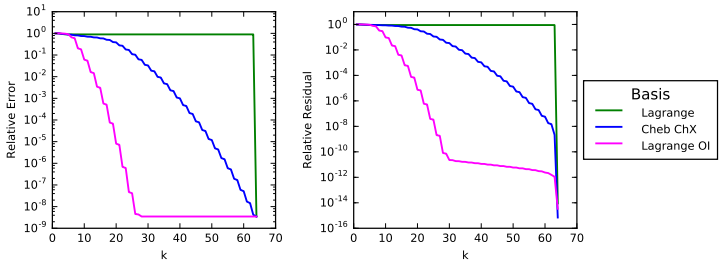


(e) Discretized with Galerkin using nodal PL basis.

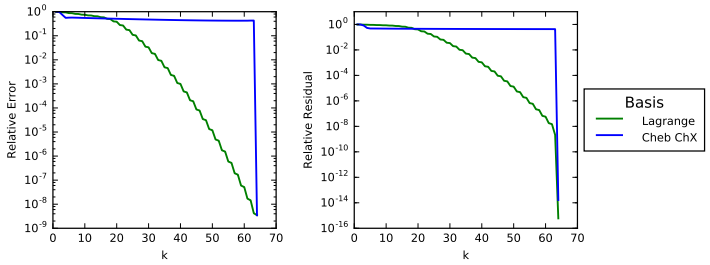


(f) Discretized with Galerkin using hierarchical PL basis.

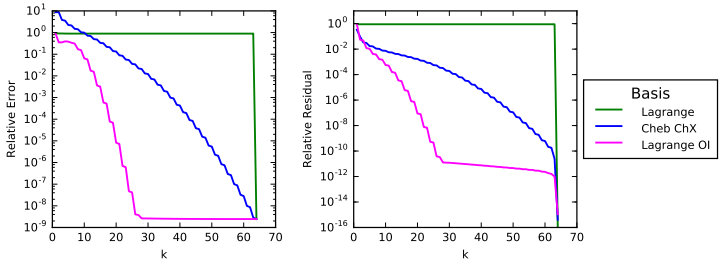
Figure 3.4: Global vs. local basis results for `designed_sine_bvp` as discussed in Section 3.4.



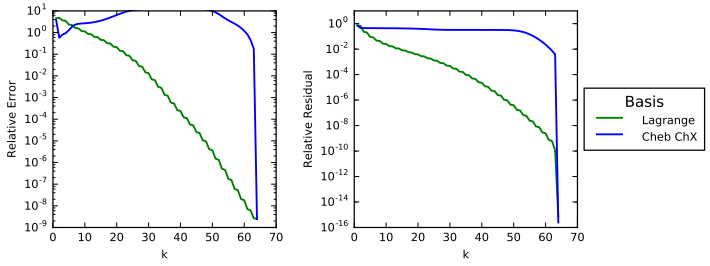
(a) Discretized with collocation using Lagrange basis.



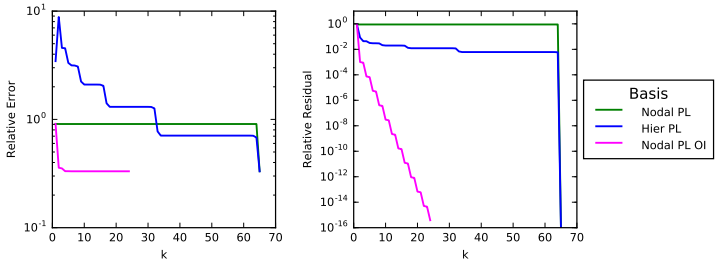
(b) Discretized with collocation using Chebyshev basis.



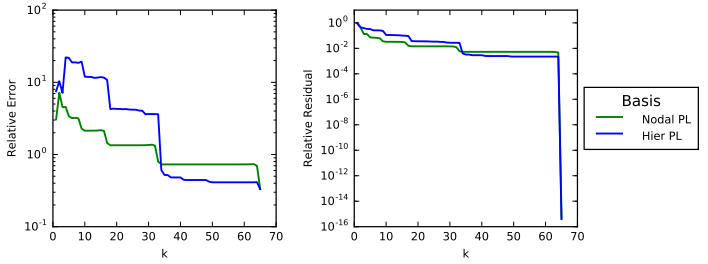
(c) Discretized with Galerkin using Lagrange basis.



(d) Discretized with Galerkin using Chebyshev basis.



(e) Discretized with Galerkin using nodal PL basis.



(f) Discretized with Galerkin using hierarchical PL basis.

Figure 3.5: Global vs. local basis results for `greengard-ex2` as discussed in Section 3.4.

3.5 PRELIMINARY TESTING

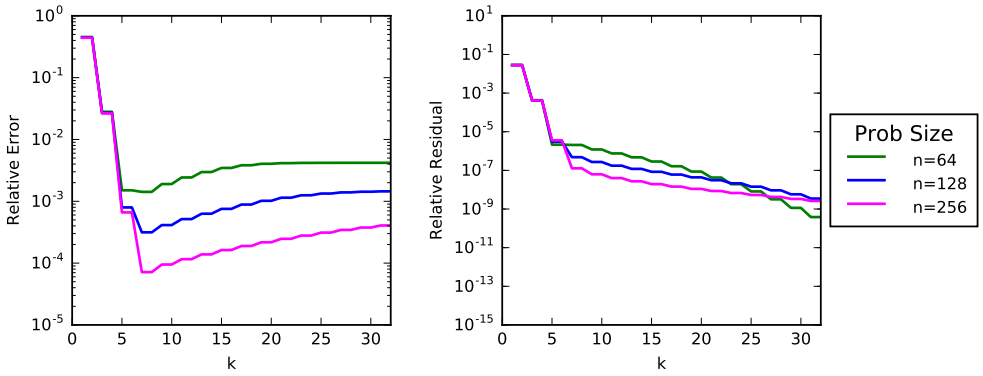
In this section, we illustrate results for targeted experiments designed to test specific hypotheses. The primary purpose of these tests is to determine the best options within the compression-based method before proceeding to more comprehensive testing in which we compare its effectiveness with that of other methods. We perform a variety of tests for each family of compression basis functions on a single boundary value problem and a single integral equation problem, `designed_sine_bvp` and `deisgned_sine_ie`, respectively, both discretized with simple discretizations. These problems are specifically designed by us to have the same true solution: a simple sine that closely resembles a low degree polynomial. For many of the tests we illustrate results with plots; however, for brevity, some results are simply discussed.

3.5.1 Scaling

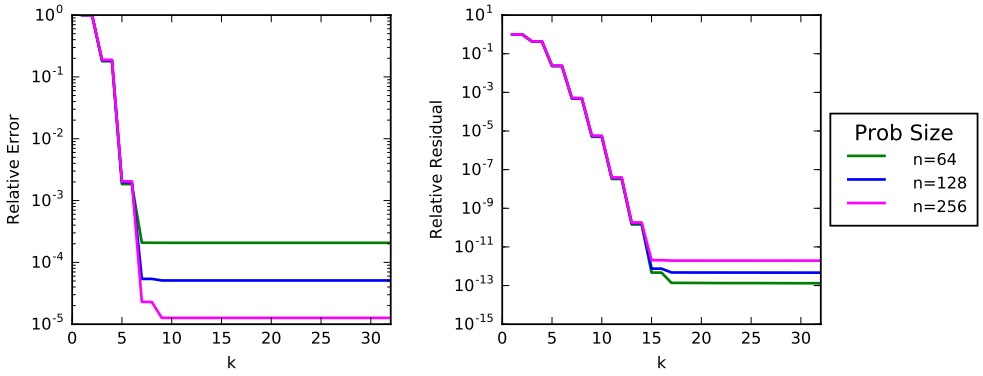
We first test how effectiveness varies with the size of the problem. Specifically, as we increase the problem size, how does the value of k required for a good approximate solution vary? This question is investigated in Figure 3.6 for `designed_sine_ie` and `designed_sine_bvp` where the problems are discretized with varying numbers of points, $n = 64, 128, 256$. As n increases, the minimum achievable error decreases because the discretization error decreases, but the value of k required to achieve the minimum error remains essentially constant. While not unexpected, this is impressive as it indicates that the dimension of the compression subspace required by the compression-based method grows little, if any, with n. While presented here only for the Chebyshev polynomial compression basis, we see a similar lack of dependence of k on n for other problems and compression bases.

3.5.2 Evaluation Points

We raised the issue of the choice of evaluation points in Section 2.2.2. To investigate this issue, we consider one compression basis (Chebyshev polynomials) evaluated at equally-spaced points (Eq), Chebyshev extrema (ChX), Chebyshev roots (ChR), and the Gauss-Legendre (Leg) points. The results are shown for `designed_sine_ie` and `designed_sine_bvp` in Figure 3.7. Specifically, Figure 3.7a and Figure 3.7b show the results for `designed_sine_ie` discretized with composite trapezoid quadrature and Clenshaw-Curtis quadrature, respectively. We see that best performance is obtained when the evaluation points match the nodes for the quadrature rule used to discretize the problem. Similarly, in Figure 3.7c, where the



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



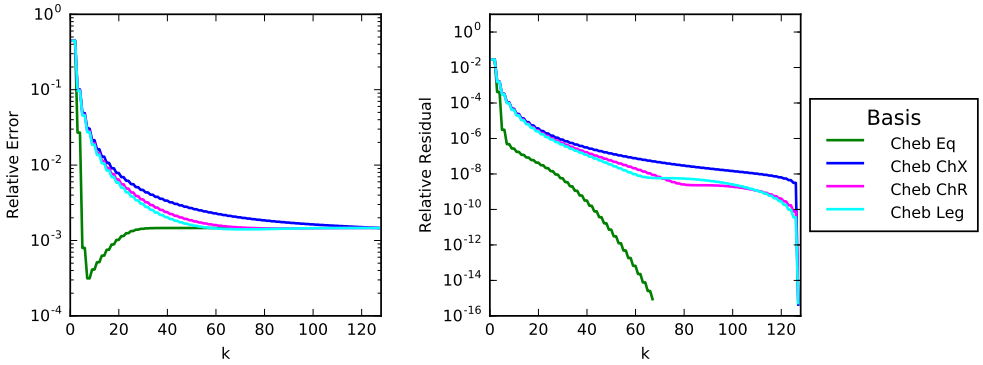
(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.6: Scaling n comparison with Chebyshev polynomials evaluated at equally-spaced points as compression basis: $n = 64, 128, 256$ (as discussed in Section 3.5.1).

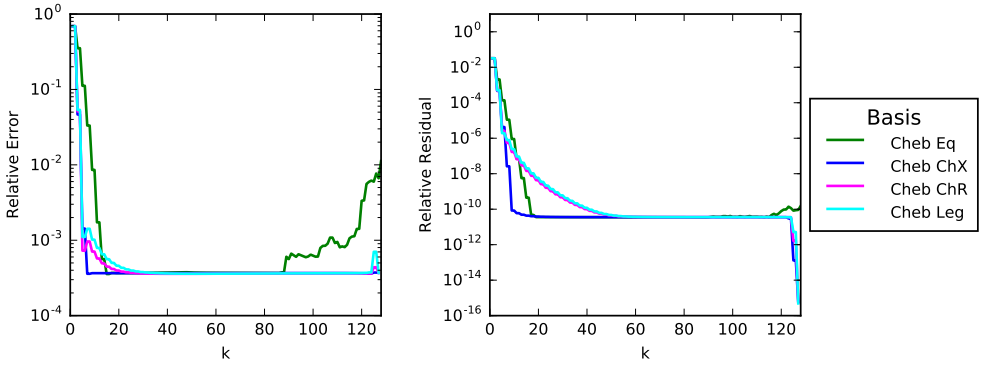
problem is discretized with finite differences (equally-spaced mesh points), we see that Cheb Eq is the most effective by far. While presented here only for Chebyshev polynomials as the compression basis, similar effects are seen for other compression bases as well. Overall, our tests show that a compression basis is most effective when the evaluation points match the discretization points.

3.5.3 Polynomials

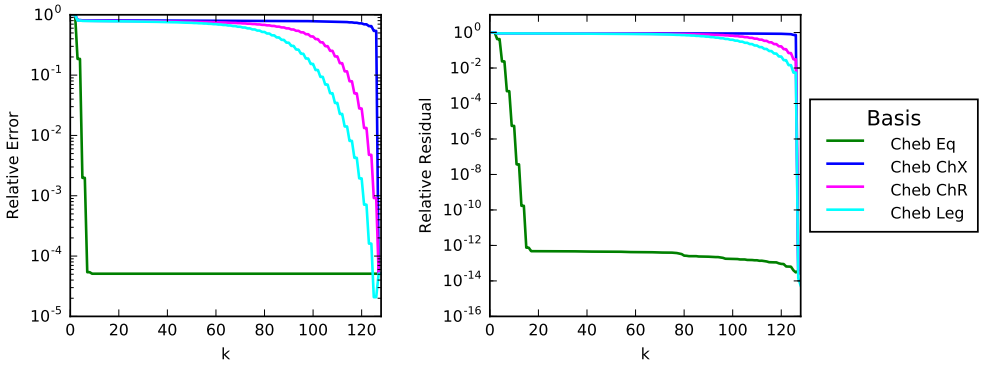
In Section 2.4 a variety of polynomial bases were discussed, leading to the natural question of which polynomial bases are most effective as a compression basis. In Figure 3.8a and Figure 3.8b we compare Chebyshev, Legendre, Lagrange, and Bernstein polynomial compression bases with the polynomials evaluated at equally-spaced points (with the exception of Lagrange which are evaluated at the points used to define them) on `designed_sine_ie` and `designed_sine_bvp`, respectively. We see that Chebyshev and Legendre compression bases perform well, achieving a reasonably small error at about $k = 8$ with $n = 128$.



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



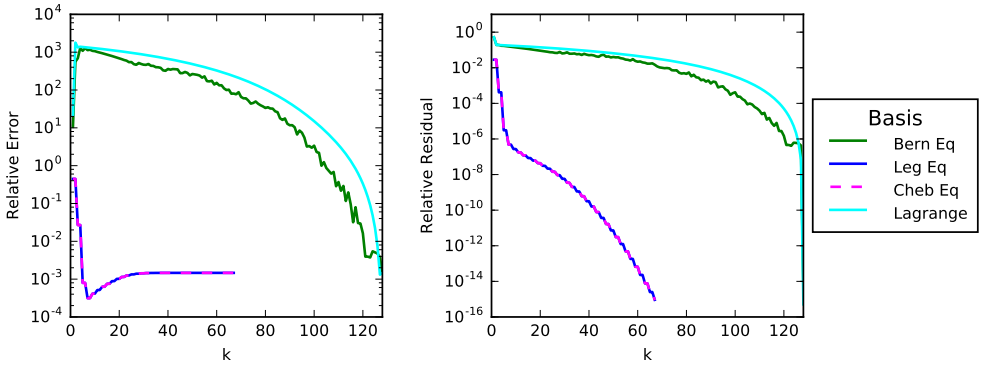
(b) Relative error and relative residual for `designed_sine_ie` discretized using Clenshaw-Curtis quadrature.



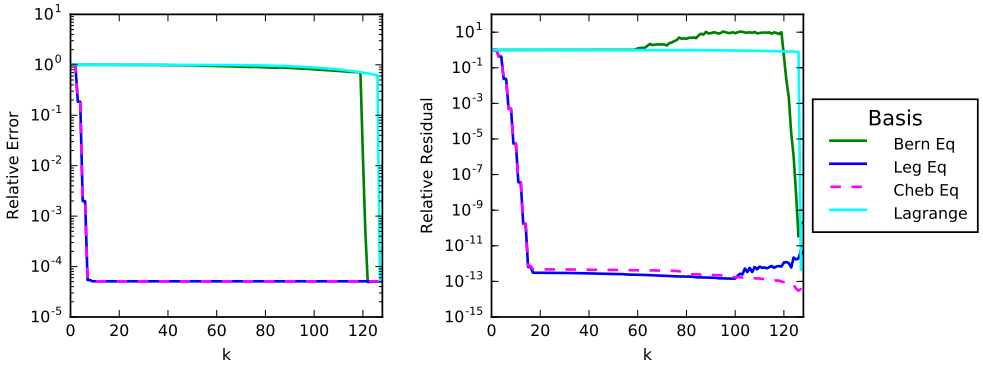
(c) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.7: Comparison of evaluation points for Chebyshev polynomial compression basis as discussed in Section 3.5.2.

Bernstein and Lagrange perform poorly, however, with the error decreasing slowly (in the case of `desinged_sine_ie`) or hardly decreasing at all until $k \approx n$ (in the case of `designed_sine_bvp`). Since the performance of Chebyshev and Legendre is almost indistinguishable, we will henceforth use Chebyshev polynomials as the polynomial compression basis of choice for simple discretizations.



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

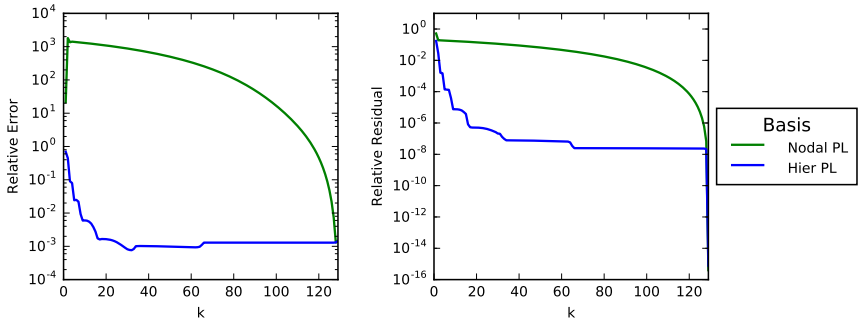
Figure 3.8: Comparison of polynomial compression bases as discussed in Section 3.5.3.

It is unsurprising that some of the polynomial compression bases perform well, as these problems have true solutions that closely resemble a low degree polynomial. We also investigate whether there is any difference in effectiveness if the solution actually is a polynomial, as opposed to just closely resembling a polynomial, by testing the same four polynomial compression bases on `designed_poly_ie` and `designed_poly_bvp`. The results are unsurprising (and excluded here for brevity): both Chebyshev and Legendre compression bases perform well, achieving a small error at $k = 5$, as the solution is a 4th degree polynomial. This is not significantly different from the results for the designed sine problems, which is expected as the solutions closely resemble one another.

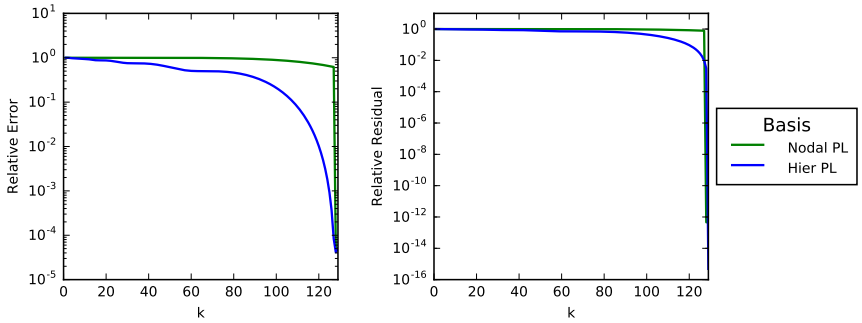
In Section 2.2 we discussed the importance of the ordering of the basis vectors for the effectiveness of the compression-based method. Based on approximation theory we expect that polynomial compression bases will be more effective when taken in the natural low-to-high degree ordering. We test this hypothesis with the Chebyshev compression basis on `designed_sine_ie` and `designed_sine_bvp`, considering the basis vectors taken in both low-to-high and high-to-low ordering. The results are not surprising (and excluded here for brevity), with low-to-high degree ordering performing vastly better. Other bases with similar theory motivating certain orderings showed similar results.

3.5.4 Piecewise Polynomials

We also use `designed_sine_ie` and `designed_sine_bvp` to test the effectiveness of the piecewise polynomial compression bases. In Figure 3.9, we compare the performance of nodal and hierarchical piecewise linear compression bases with equally-spaced evaluation points. Examining these results, we see that in both cases the hierarchical compression basis performs better than the nodal basis, which is expected as `designed_sine_ie` and `designed_sine_bvp` have global solutions and thus, according to Table 2.1, when discretized with finite differences (an implicitly nodal discretization), a modal compression basis would be required to have a chance to do well.



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



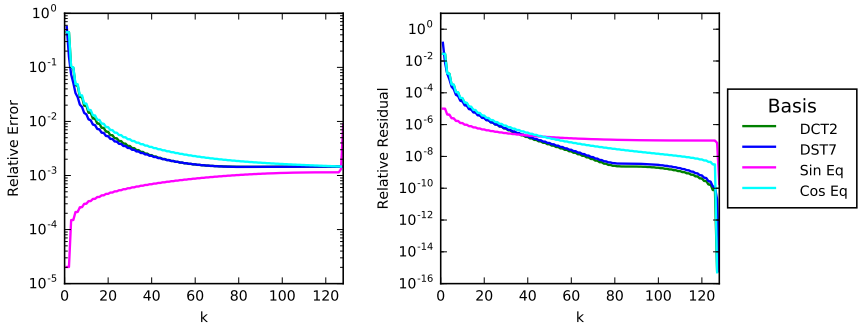
(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.9: Comparison of piecewise polynomial compression bases as discussed in Section 3.5.4.

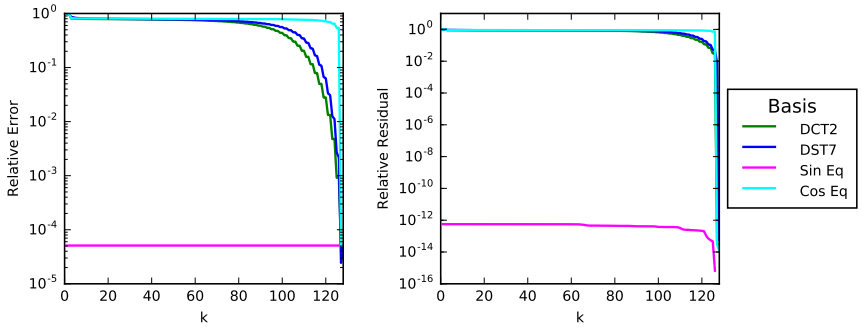
3.5.5 Sinusoids

Due to the sheer number of versions of the DCT and DST, we do not show full test results here, though the results showed that none of the standard DCT or DST perform particularly well or poorly as compression bases. Consequently, we illustrate a comparison among selected sinusoids. From the canonical DCT and DST we include DCT2 and DST7, which tend to perform better than the other canonical DCT and DST, respectively. Additionally, we

compare these to alternative compression bases of sines and cosines simply evaluated at the discretization points. This comparison is shown in Figure 3.10. Examining the results, we see that, for both problems, sines evaluated at evenly spaced points perform well as a compression basis. However, this is to be expected as the true solution function for these problems is a sine function. The other bases, on the other hand, perform moderately well for `designed_sine_ie`, but they are not particularly effective for `designed_sine_bvp`. We also examined the use of the DHT as a compression basis, but it was not effective, and the results are omitted here for brevity.



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.10: Comparison of selected sinusoidal compression bases as discussed in Section 3.5.5.

There are connections between the canonical versions of the DCT and DST and boundary conditions. Namely, the canonical versions of the DCT and DST are the eigenvectors of second difference matrices with various boundary conditions [33]. We have investigated whether this has an impact on which versions of the DCT or DST perform best, i.e. if the boundary conditions associated with the best performing version match the boundary conditions for the problem. We found no clear correlation between the best performing version and the boundary conditions in the matrix. Overall, DCT2 performs best, but the DCT2 basis vectors are the eigenvectors of the second difference matrix with both left and right Neumann boundary conditions enforced at the midpoint, while we consider only

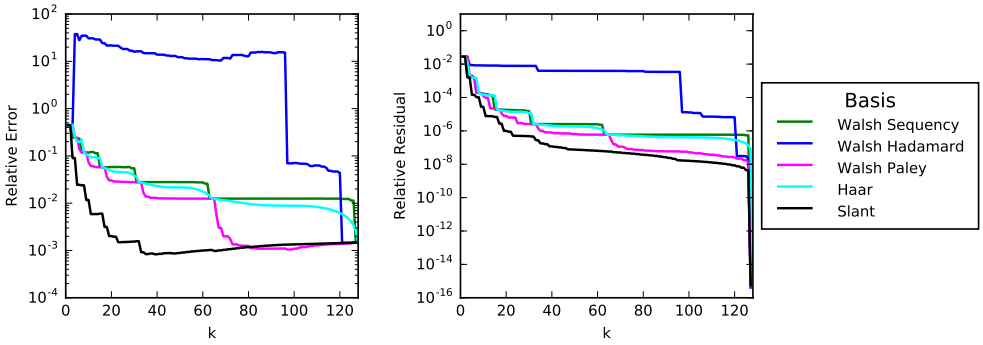
boundary conditions enforced at the endpoints. This lack of correlation is unsurprising, however, and can be attributed to a combination of a variety of factors that have a stronger impact on performance. In [19], it is noted that the DCT2 basis vectors are actually sampled Chebyshev polynomials. In fact, the basis vectors for DCT2 are the same as the basis vectors for the Discrete Chebyshev Transform [34]. As we illustrate in Section 3.5.3, the choice of evaluation points is crucial to performance. For the compression-based method, we can incorporate knowledge of the solution to perform better by simply choosing sines, cosines, or Chebyshev polynomials for the compression basis and evaluating them at points corresponding to the discretization points.

3.5.6 Square Waves

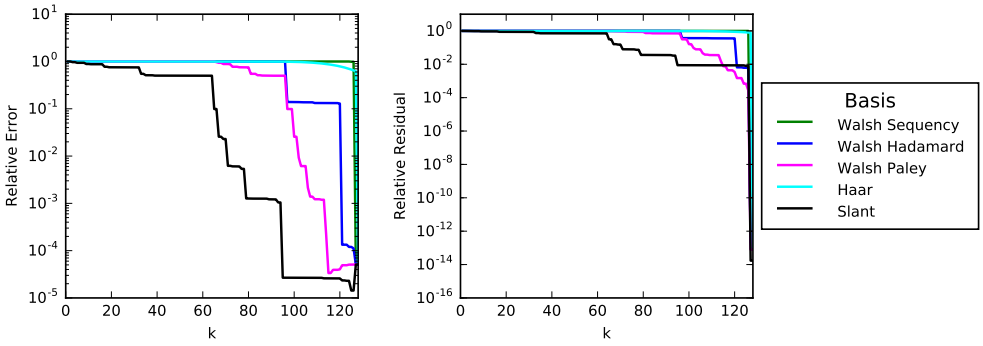
A variety of square wave bases are tested to determine their effectiveness as compression bases. Specifically, we consider the Haar basis, Slant basis, and each of the three standard orderings for the Walsh basis. The results of this comparison are shown in Figure 3.11 for `designed_sine_ie` and `designed_sine_bvp`. Among the orderings for Walsh, Walsh Paley performs best, but it is still ineffective overall. Interestingly, the Walsh functions yield noticeable drops in the error when k is a power of two. Various other orderings for Walsh have been investigated attempting to capture these drops in error sooner, but have not been fruitful. Overall, Slant is the most effective of the square wave bases.

3.5.7 Wavelets

Similar to square wave bases, we also consider bases derived from wavelets other than the Haar wavelet as discussed in Section 2.4.1. A variety of wavelet bases are tested to determine their effectiveness as compression bases. Figure 3.12 illustrates results for a comparison of a few selected wavelet bases derived from the multi-level wavelet decomposition "Wav" and the full-tree wavelet packet decomposition "Wav Pckt" with a few wavelet families for `designed_sine_ie` and `designed_sine_bvp`. In Figure 3.12a, we see that for `designed_sine_ie`, the wavelet bases derived from the Symlets 4 (sym4) wavelet multi-level and full-tree wavelet decompositions perform best of the bases considered. However, in Figure 3.12b, we see that for `designed_sine_bvp`, none of the wavelet bases perform well. Overall, the wavelet bases are not generally effective as compression bases for these types of problems and are not considered further.



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.

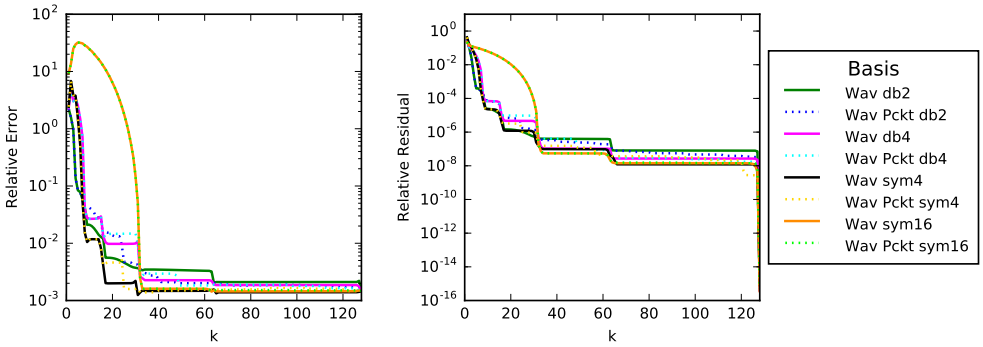


(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

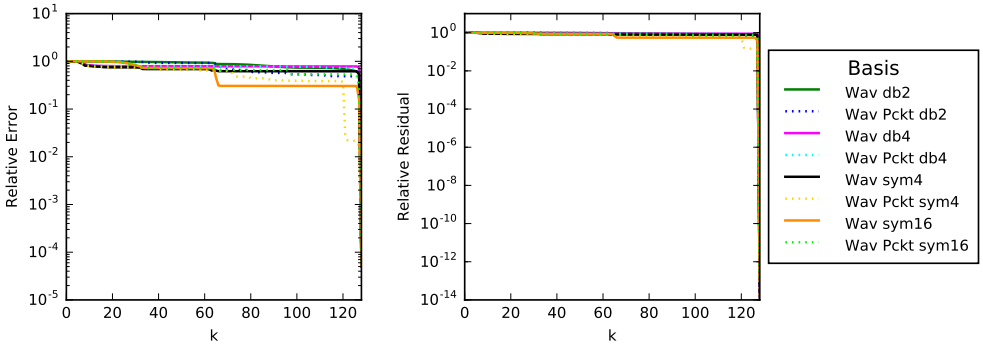
Figure 3.11: Comparison of various square wave compression bases and their applicable orderings as discussed in Section 3.5.6.

3.5.8 Gaussians

To investigate the effect of overlap between compression basis functions, we consider Gaussian basis functions with adjustable overlap between them. We investigate their effectiveness as a compression basis as well as the optimal amount of overlap. Figure 3.13 illustrates a performance comparison between nodal and hierarchical Gaussian bases, with various degrees of overlap for each, on `designed_sine_ie` and `designed_sine_bvp`. In general, we see that hierarchical Gaussian bases are more effective than nodal Gaussian bases, which is expected according to Table 2.1. For both hierarchical and nodal Gaussian bases, as the overlap increases the basis becomes more global and more effective. However, there is a tradeoff between overlap and ill-conditioning: as overlap increases the basis functions become more global, but the basis matrix becomes more ill-conditioned. Therefore, we see there is an optimal amount of overlap beyond which effectiveness decreases. Examining the error plots in Figure 3.13, this tradeoff is not necessarily clear, but looking at the residual plots as well, we see that for hierarchical Gaussian bases, the optimal overlap is approximately $c = 0.125$, and for nodal Gaussian bases, the optimal overlap is approximately $c = 0.01$. With the



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

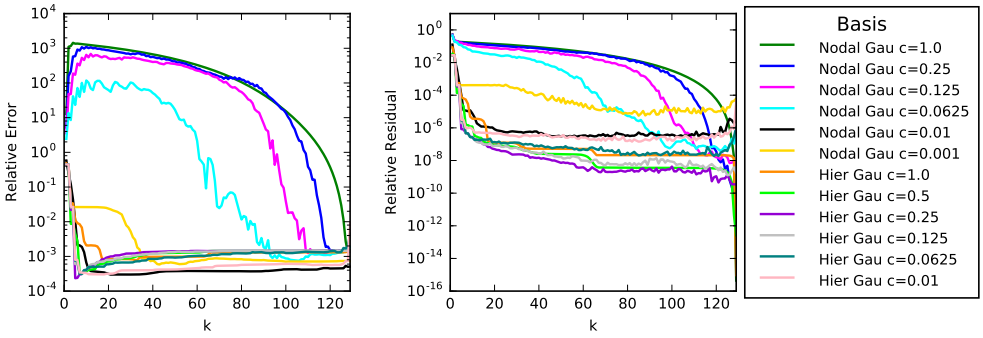
Figure 3.12: Comparison of various wavelet compression bases and their applicable orderings as discussed in Section 3.5.7.

optimal amount of overlap, the nodal Gaussian basis is essentially global as well and is more competitive with the hierarchical Gaussian bases, but still less effective. To confirm that the behavior we see is not specific to Gaussians, we also performed a similar test with quartic polynomials and saw the same overlap trend.

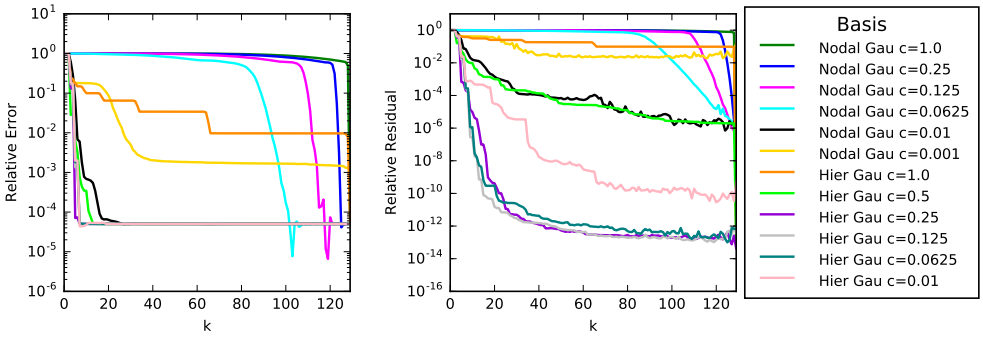
Somewhat similar to Gaussians, we also considered nodal and hierarchical compression bases composed of sinc functions. Illustrations of the results are omitted for brevity, but overall nodal and hierarchical sinc bases are not effective as compression bases. Although commonly used in signal processing, considering the shape of the sinc functions (with damped oscillations), the poor performance is not unexpected. The numerous oscillations are more likely to contribute to noise than capture the main pattern of the solution with relatively few basis vectors.

3.5.9 Random Sampling

Results for randomly generated bases ("random bases") are illustrated in Figure 3.14. We consider random bases drawn from both uniform and normal distributions and modified



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



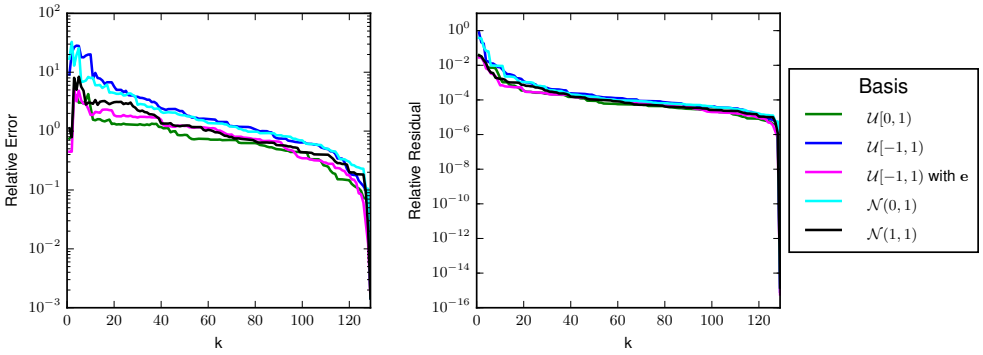
(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.13: Comparison of nodal and hierarchical Gaussian compression bases with various amounts of overlap as discussed in Section 3.5.8.

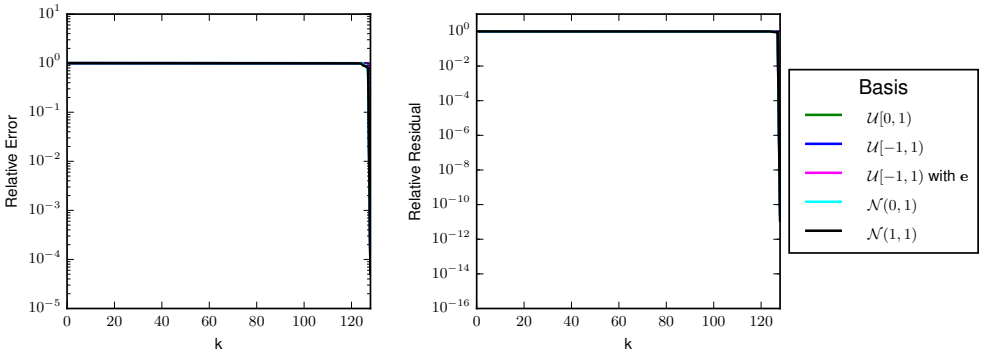
random bases (where the first vector is taken to be the constant vector of all ones). Due to the randomness, results with a random basis vary significantly between runs, making it difficult to glean any information from a single run. Because of this variability, the results presented are averaged over 100 runs to ensure that the overall trend is shown. From these results, we see that compression bases generated by random sampling are ineffective, with the error decreasing very slowly (or not at all) until the final basis vectors. This is expected, however. As we hypothesized in Section 2.4.1, random basis vectors are unable to capture a pattern in the solution with few basis vectors, and they are therefore not suitable for use in the compression-based method

3.5.10 Preliminary Conclusions

These individual tests on `designed_sine_ie` and `designed_sine_bvp` discretized with simple discretizations allow us to draw a number of preliminary conclusions regarding the effectiveness of the various options for the compression-based method. For these global solution problems discretized with simple discretizations, as indicated by Table 2.1, global bases



(a) Relative error and relative residual for `designed_sine_ie` discretized with composite trapezoid quadrature.



(b) Relative error and relative residual for `designed_sine_bvp` discretized using finite differences.

Figure 3.14: Comparison of various random sampling compression bases as discussed in Section 3.5.9.

such as Chebyshev polynomials and hierarchical Gaussians with significant overlap show strong potential as compression bases, whereas local bases such as Lagrange polynomials and nodal piecewise linear appear to be inefficient as compression bases. We also conclude that there are a variety of crucial factors to consider regarding a compression basis, including the ordering of the basis vectors, where the natural order is typically best, when applicable, and the choice of evaluation points, where matching the discretization points is typically best, when applicable. Additionally, while sinusoidal bases can be effective, the canonical discrete transform versions are not always the best choice. We also conclude that some bases, including Bernstein polynomials, DHT, sinc functions, and random sampling are simply not competitive as compression bases and will not be considered further.

3.6 COMPREHENSIVE RESULTS

To illustrate the overall effectiveness of the compression-based method, we present results of the best performing compression bases (among the many bases listed in Section 2.4) on the test problems from the literature described in Section 3.1 with details in Appendices B

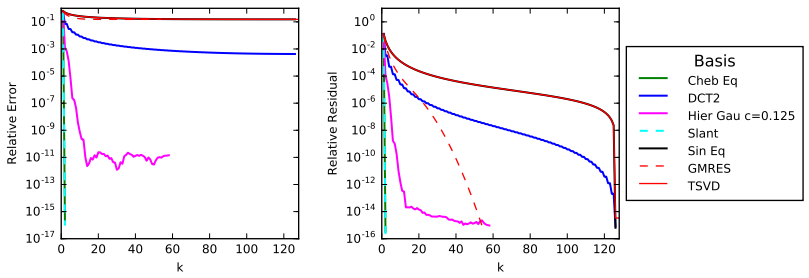
and C. For each test, results will also be shown for GMRES and TSVD. Whenever possible (when $\boldsymbol{A}$ is square and full rank), the error and residual for the solution from a direct solver are also shown (specifically NumPy's [30] built-in direct solver). As this is a fixed error and residual, results for the direct solver are indicated by a horizontal straight line on each plot. We first present results for implicit (simple) and explicit nodal discretizations and then for modal discretizations.

3.6.1 Nodal Discretizations

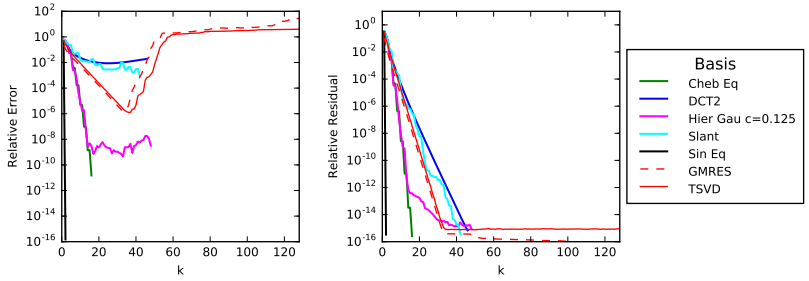
We begin by presenting results for simple discretizations, which are implicitly nodal, of problems with global solutions. For a nodal discretization of a problem with a global solution, Table 2.1 indicates that, in order to have a chance to do well, a modal compression basis is required. Thus, we illustrate the best performing modal compression bases. Wherever appropriate, the evaluation points for the compression basis are chosen to match the discretization points (e.g., ChebEq is chosen when the discretization points are equally spaced).

Results are illustrated for a variety of integral equation and boundary value test problems with global solutions in Figure 3.15. The integral equations are discretized using composite trapezoid quadrature with $n = 128$. The boundary value problems are discretized using finite differences, primarily with $n = 128$. For `greengard-ex3`, due to the highly oscillatory nature of the solution, it is necessary to have more points to resolve this behavior, so we take $n = 512$, and the results are illustrated in Figure 3.15e.

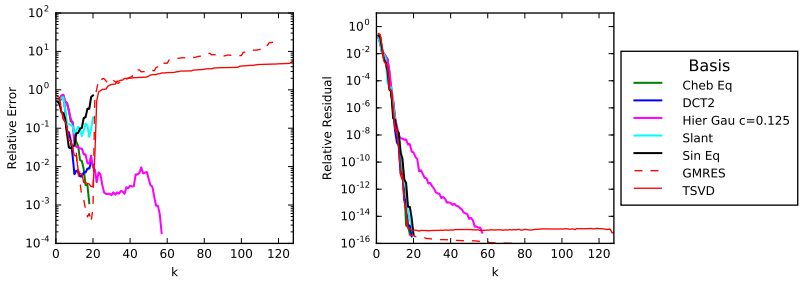
From these results, we see that multiple bases perform well for many of the problems, and it is possible for the best performing basis to achieve a lower error with smaller k than GMRES or TSVD. For `gravity_c1`, the best bases achieve a lower error overall than conventional methods. We also see how the nature of the solution can affect performance. For example, we see in Figure 3.15b that sampled sines perform extremely well for `gravity_c1`, a problem for which the true solution is a combination of sines. However, sines do not universally perform well, illustrating how additional information about the solution can help guide the basis choice. Overall, while other bases may occasionally perform as well or better (depending on the nature of the solution), Chebyshev polynomial bases and hierarchical Gaussian bases with significant overlap tend to perform best.



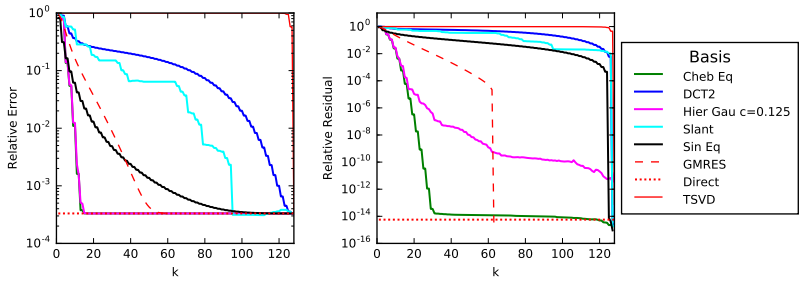
(a) Relative error and relative residual for overall test for `deriv2_c1` discretized by Nystrom method using composite trapezoid quadrature with $n = 128$.



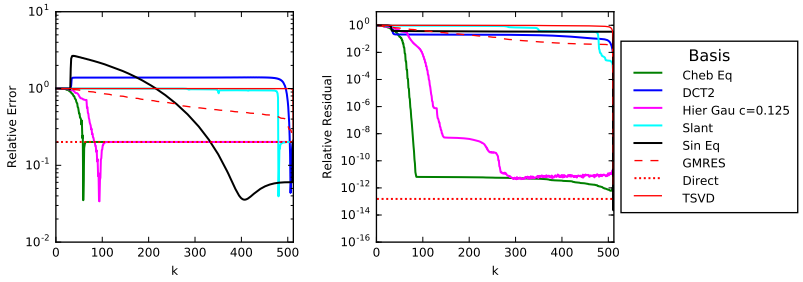
(b) Relative error and relative residual for overall test for `gravity_c1` discretized by Nystrom method using composite trapezoid quadrature with $n = 128$.



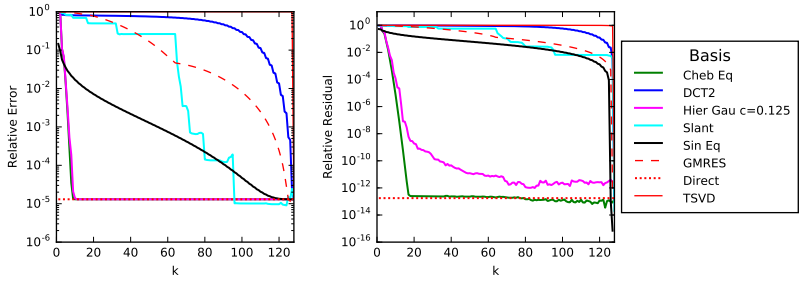
(c) Relative error and relative residual for overall test for `shaw` discretized by Nystrom method using composite trapezoid quadrature with $n = 128$.



(d) Relative error and relative residual for overall test for `greengard-ex1` discretized using finite differences with $n = 128$.



(e) Relative error and relative residual for overall test for `greengard-ex3` discretized using finite differences with $n = 512$.

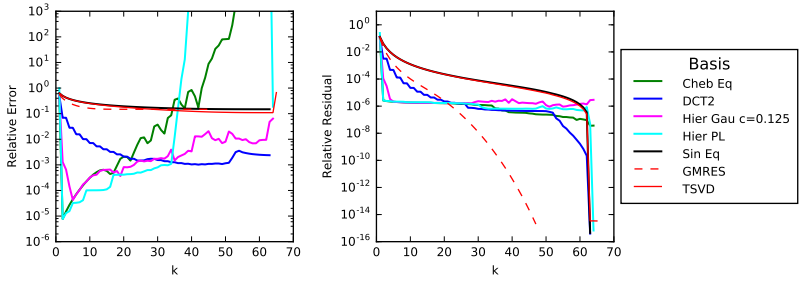


(f) Relative error and relative residual for overall test for `fornberg` discretized using finite differences with $n = 128$.

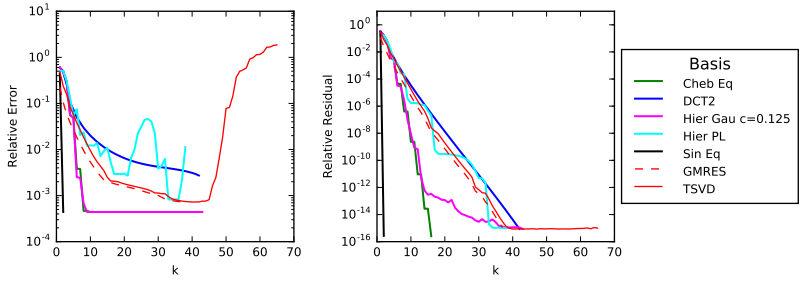
Figure 3.15: Overall tests comparing best performing modal compression bases for integral equation problems and boundary value problems with global solutions discretized using simple discretizations (implicitly nodal) as discussed in Section 3.6.1.

In addition to results for simple discretizations, which are implicitly nodal, we also present results for explicitly nodal discretizations of problems with global solutions. From the preliminary testing in Section 3.5, it is important to match the evaluation points to the discretization points. This is also an important consideration here as the nodal bases have certain points that are used to define them. We define the Lagrange polynomials using the Chebyshev extrema (for conditioning purposes), and we define the nodal piecewise linear basis using equally-spaced points. Because of the impact of the matching of points on performance, slight modifications have been made in the choice of compression bases when using an explicitly local discretization to use those that match. For example, when using Galerkin discretized with Lagrange polynomials (defined using the Chebyshev extrema), we use the Chebyshev polynomials sampled at the Chebyshev extrema instead of the Chebyshev polynomials sampled at equally-spaced points.

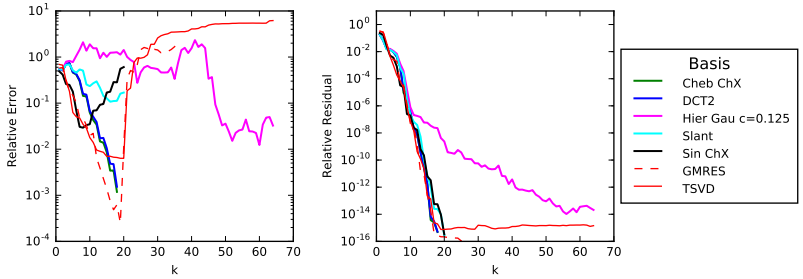
Due to the inordinate number of combinations of discretization methods and discretization bases, we show only a few representative results (one for each problem), which are illustrated in Figure 3.16. Overall, these results are similar to the comprehensive testing results for simple discretizations in Figure 3.15. In most cases, one or more compression bases perform well. In general, Chebyshev polynomials (evaluated at the points matching those used to define the discretization basis) and hierarchical Gaussian bases (with the optimal overlap value determined in Figure 3.13) tend to perform best. Occasionally, other bases perform well, but this is typically due to special instances as we have seen previously. For example, in Figure 3.16a hierarchical piecewise linear performs well because the true solution for `deriv2_c1` is a linear polynomial.



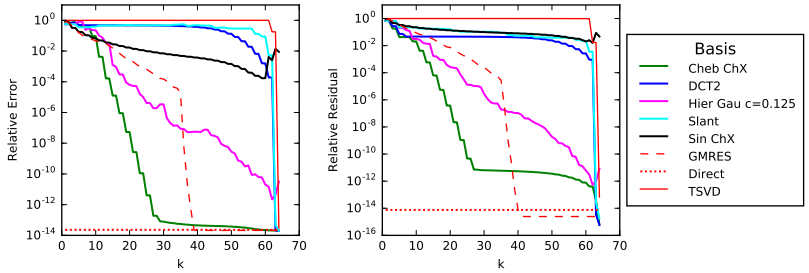
(a) Results for `deriv2_c1` discretized using collocation with nodal piecewise linear basis with uniform collocation points.



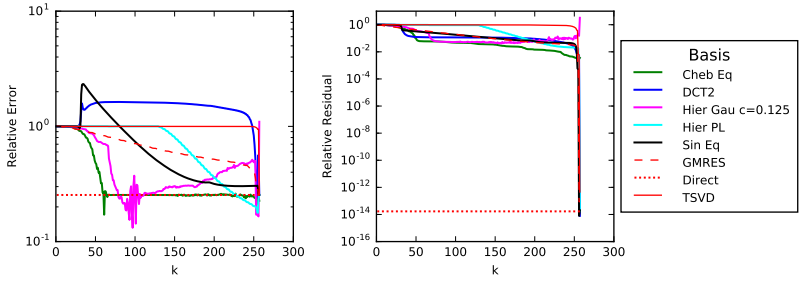
(b) Results for `gravity_c1` discretized using Galerkin with nodal piecewise linear basis.



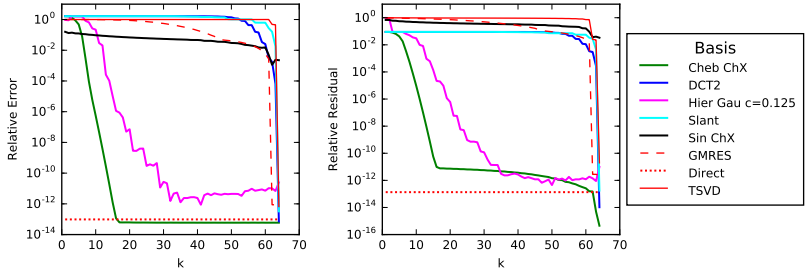
(c) Results for `shaw` discretized using collocation with Lagrange basis with Chebyshev extrema collocation points.



(d) Results for `greengard-ex1` discretized using Galerkin with Lagrange basis.



(e) Results for `greengard-ex3` discretized using Galerkin with nodal piecewise linear basis.



(f) Results for `fornberg` discretized using collocation with Lagrange basis with Chebyshev extrema collocation points.

Figure 3.16: Selected nodal discretization with modal compression bases results for problems with global solutions as discussed in Section 3.6.1.

Local Solutions

While the results in Figures 3.15 and 3.16 focused on problems with global solutions, we also present results for the single local solution problem, `greengard-ex2`, discretized using finite differences. As indicated by Table 2.1 and illustrated in Section 3.4, the best performing bases for `greengard-ex2` with nodal discretizations are nodal compression bases taken in outside-in order as it is a boundary layer problem. We compare the performance for both the "Lagrange OI" and the "Nodal PL OI" compression bases with traditional methods in Figure 3.17. We see that the performance for each nodal compression basis is the same, as both evaluate to the identity matrix when sampled at the points used to define the basis functions. We also see that the method is reasonably competitive with GMRES for this problem.

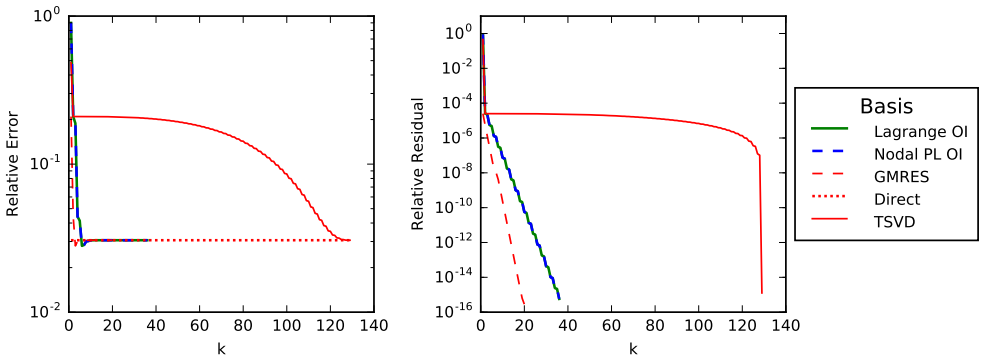


Figure 3.17: Relative error and relative residual for overall test for `greengard-ex2`.

3.6.2 Modal Discretizations

We now present results for the test problems with modal discretizations. As discussed in Section 3.4, modal discretizations of problems with local solutions are an unconventional and likely impractical choice, and, as illustrated in Figure 3.5, the modal compression bases considered are not effective at capturing the pattern in the solution vector from such a discretization. Thus, we do not present additional results for `greengard-ex2` with modal discretizations beyond those presented in Figure 3.5.

According to Table 2.1 for a modal discretization of a problem with a global solution, we expect that a nodal compression basis is required to have a chance of performing well. As the remaining test problems have global solutions, we present results only for nodal compression bases. As with explicit nodal discretizations, due to the inordinate number of combinations of discretization methods and discretization bases, we show only a few representative results

(one for each problem). In Figure 3.18, we compare all of the available local bases (nodal Gaussian, Lagrange, piecewise linear) as well as the identity matrix for `deriv2_c1`. Each of the local bases is evaluated at its natural points (i.e., the points used to define them). Interestingly, the results for all of these local bases are identical. This is unsurprising, however, as each local basis matrix reduces to the identity matrix when evaluated at its natural points. Therefore, in Figure 3.19 we present results for the remaining test problems with only the identity matrix as the local compression basis.

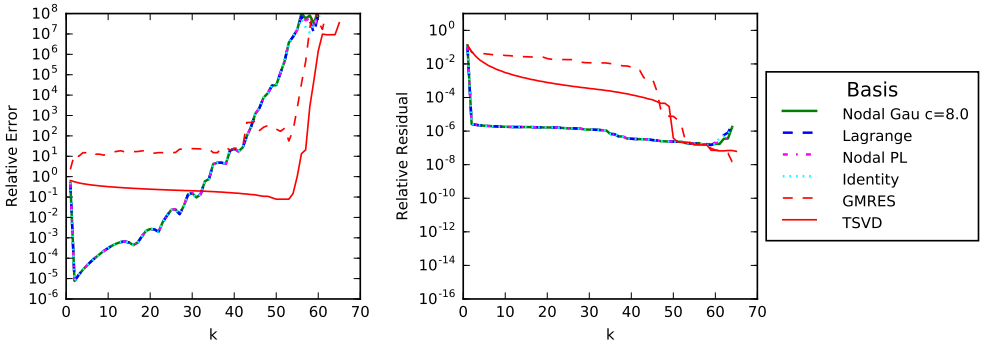
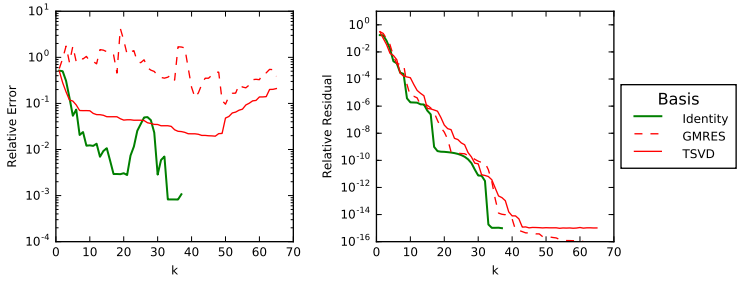
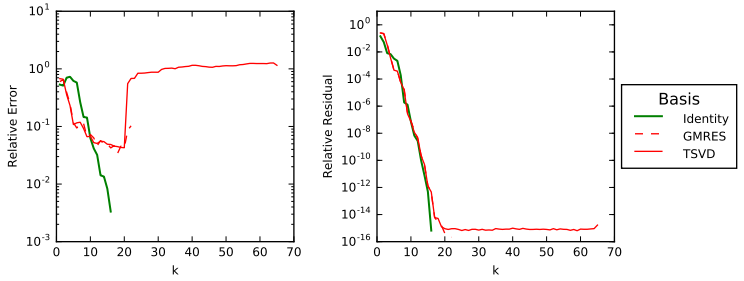


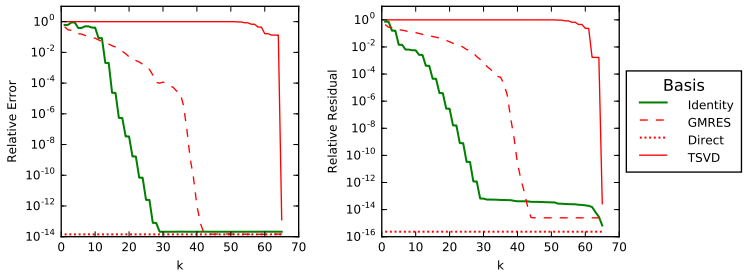
Figure 3.18: Full nodal compression basis comparison for `deriv2_c1` discretized using collocation with Chebyshev polynomials with equally-spaced collocation points as discussed in Section 3.6.2.



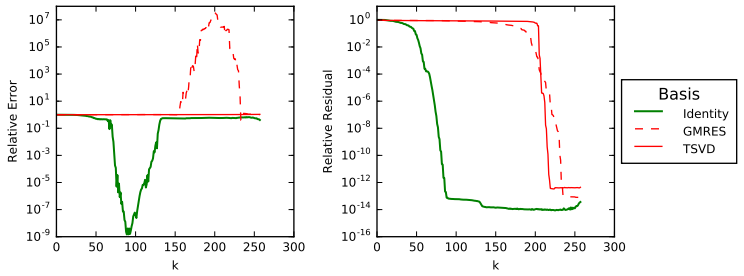
(a) Results for `gravity_c1` discretized using collocation with hierarchical piecewsise linear basis with equally-spaced collocation points.



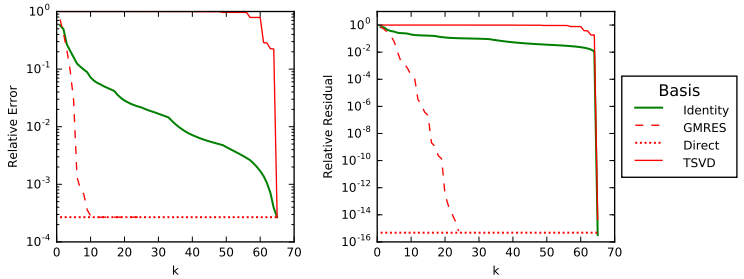
(b) Results for `shaw` discretized using Galerkin with Chebyshev basis.



(c) Results for `greengard_ex1` discretized using Galerkin with Chebyshev basis.



(d) Results for `greengard_ex3` discretized using collocation with Chebyshev basis.



(e) Results for `fornberg` discretized using Galerkin with hierarchical piecewise linear basis.

Figure 3.19: Selected global discretizations with nodal compression bases results for problems with global solutions as discussed in Section 3.6.2.

In Figures 3.18 and 3.19, we see that for numerous problems a reasonably low error is obtained with relatively small k. For `deriv2_c1`, in Figure 3.18, we see that the error decreases sharply at first, obtaining an excellent solution almost immediately. This is unsurprising as the problem is discretized with collocation using the Chebyshev basis, and the true solution is simply a linear function, so only the first couple of coefficients are necessary. With the identity matrix as the compression basis taken in left-to-right order, these coefficients are determined almost immediately. In Figure 3.18, we also see a regularizing effect, where, after a certain value of k, the error begins to increase rapidly as increasing the dimension of the compression subspace with additional basis vectors is simply adding unwanted noise. Overall, with the identity matrix as the compression basis matrix, our compression-based method reduces to an incremental QR factorization that is truncated and therefore has the potential to perform well for problems having effectively low rank.

3.7 TESTING SUMMARY

Overall, these results illustrate that the compression-based method has the potential to perform well on linear systems resulting from the discretization of 1D continuous problems. For most 1D problems, one or more compression bases perform well, and in numerous cases the best performing basis achieves a lower error with smaller k than GMRES or TSVD. For some problems, we see a regularizing effect, with the minimum error achieved by the compression-based method lower than the minimum error with conventional methods, making the compression-based method an interesting option for linear systems resulting from ill-posed problems, which typically require regularization.

While the method has the potential to perform well, the basis must be chosen carefully, and the properties of the solution affect both the choice of basis and the ordering of the basis vectors. For instance, properties such as whether the solution is low or high frequency, odd or even, etc. can affect the performance of certain bases and orderings. Whether the solution function is global or local and what type of basis is used for the discretization can help guide the choice of basis for the solver, as in Table 2.1.

Chapter 4: 2D Numerical Results

In this chapter we present numerical results for the compression-based method applied to 2D test problems. As with 1D numerical results, we first provide an overview of the types of test problems considered, discuss the experimental setup, and then present results to test specific hypotheses and illustrate the overall effectiveness of the method.

4.1 2D TEST PROBLEM DETAILS

For the 1D results in Chapter 3, we consider linear systems resulting from discretizing continuous problems, and we explicitly perform the discretization of the problems in order to investigate comprehensively the impact of the discretization on the effectiveness of the method. For the 2D numerical results, however, we consider a wider range of test problems with various levels of knowledge of the details of the discretization. For some problems, such as those discretized with simple discretizations and used for preliminary testing, detailed knowledge of the discretization is available and utilized. Other test problems are more complex and are used directly in their discretized form provided by the source packages, without detailed knowledge of the discretization. In essence, these discretized problems are treated as "quasi-black-box" problems, and we do not control details of the discretization other than the size of the resulting linear system.

The 2D test problems considered, regardless of the level of knowledge of the discretization utilized, all fundamentally result from the discretization of an underlying continuous problem. While we use the terminology "quasi-black box" to refer to the test problems for which we do not utilize detailed knowledge of the discretization, we do not consider truly general problems that are not derived from underlying continuous problems, as there is no expectation that the compression-based method would be an effective choice for such problems. For the compression-based method to be effective, there must be a compressed representation of the solution and a suitable basis must be identified for computing it. Therefore, unless an effective basis is available, the compression-based method is not well suited to truly general linear systems.

General information for the 2D test problems can be found in Appendix D. As we no longer focus solely on the continuous form of the problem, for many of the problems we do not give detailed equations, but instead provide details on the source package of the problem, any necessary details for generating the problem, and an illustration of the solution. For those problems not taken from the original source in a fully discretized form, full formulas for the underlying continuous problems are included in Appendix D.

4.2 EXPERIMENTAL SETUP

All results presented are obtained using Python 2.7 with IEEE double precision arithmetic. Some test problems are generated using additional software. Specifically, the test problems `ir-*` are generated by the IR Tools MATLAB package [35], and the resulting discretized linear systems are written out to full machine precision and read in by the Python code, which runs the compression-based method. Similarly, all 2D problems discretized with finite elements are discretized using FEniCS [36, 37], and the resulting linear system is explicitly assembled using the code in Listing D.1.

Functions and data structures from NumPy [30, 31] and SciPy [32] are utilized wherever appropriate. For each test, the compression-based method is run until a relative residual tolerance of 10^{-15} has been reached or until $k = n$. This is done to illustrate the full behavior of the method, although in practice one would stop as soon as a satisfactory solution has been attained, the topic of Chapter 5.

As discussed in Section 2.3.2, in order to obtain 1D basis vectors, which are required for the compression-based method, a 1D ordering of the evaluation points must be determined, which is separate from which linear ordering is used to traverse the grid of doubly-indexed basis functions as discussed in Section 2.3. The notation "Eq", etc. is used to denote a 1D ordering of the evaluation points obtained by flattening a 2D grid of evaluation points in row-major order. However, as discussed in Section 2.3.2, it is crucial that the evaluation points are compatible with the discretization. Accordingly, we introduce the notation "Custom" to denote other evaluation points or other linear orderings of the evaluation points, such as the ordered coordinates corresponding to the ordered degrees of freedom for the discretized problem.

Most results are illustrated with plots of the relative error and relative residual. As in Chapter 3, the residual is computed as the discrete 2-norm of the residual vector from the solution to the linear system. As some 2D test problems are taken directly in discretized form from the original source, we do not always have the details of the discretization and the underlying continuous problem. Therefore, we adjust how we compute the error for 2D test problems, and for many 2D problems the error is not computed in comparison to the true solution function, which was the approach for 1D numerical tests. Specifically, for problems explicitly discretized by us using finite differences or the Nystrom method, the underlying continuous problem is known and the true solution vector is obtained by sampling the true solution function at the discretization points. For problems discretized using finite elements, the true solution vector is taken as the vector computed using the standard built-in direct solver for the discretized linear systems. For problems from the

IR Tools package [35] (i.e., `ir-*`), the true solution vector is provided by the package that generates the problem. In all cases, the error is computed as the discrete 2-norm of the error vector between the respective "true solution" vector and the approximate solution obtained by the compression-based method.

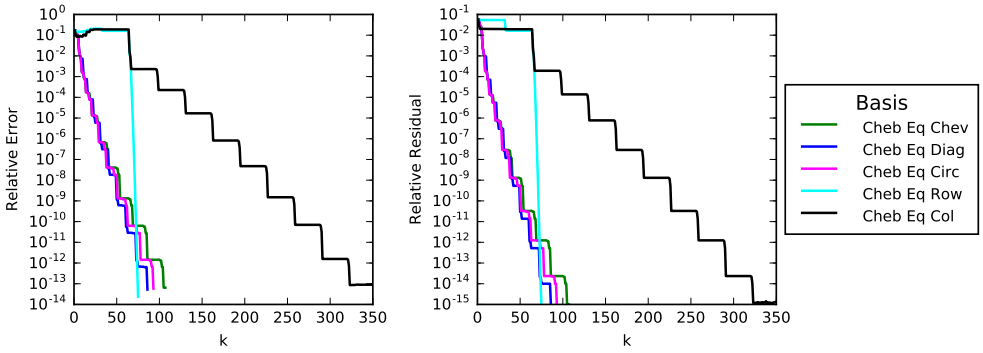
As in Chapter 3, many of these results compare the performance of the compression-based method with standard native implementations of GMRES and TSVD (instead of reproducing them with our method), and we specifically compare with GMRES and TSVD as they are similarly general methods (i.e., they do not rely on any special properties of the linear system such as symmetry of $\boldsymbol{A}$.)

4.3 PRELIMINARY TESTING

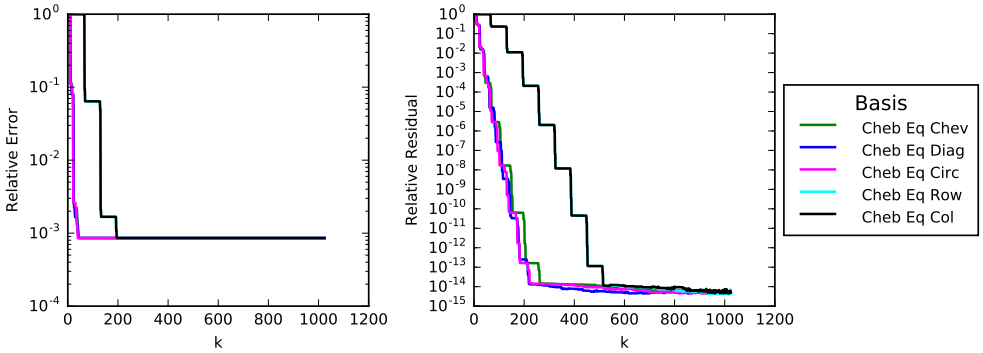
In this section, we illustrate results for any additional targeted experiments to test specific hypotheses beyond those hypotheses tested in Section 3.5. As the majority of the hypotheses tested in Section 3.5 are applicable in 2D as well, experiments for those tests are not repeated here. Any additional targeted tests are performed on the 2D boundary value problem, `2d-designed-sine-bvp` and 2D integral equation problem, `su-ex3`, which are discretized using simple discretizations and for which the continuous formulations of the problems are available.

As discussed in Section 2.3, unlike 1D bases, which often have a natural ordering, there is no single, natural, linear, low-to-high frequency or degree ordering for the 2D doubly-indexed basis functions. To investigate this issue, we test the various orderings for a single compression basis. Specifically, we consider the 2D basis resulting from Chebyshev polynomials taken in diagonal (Diag), chevron (Chev), circular (Circ), row-major (Row), and column-major (Col) orders (as described in Section 2.3). As both of these problems are discretized using equally-spaced points, only equally-spaced evaluation points are considered. The results are shown in Figures 4.1a and 4.1b for `su-ex3` and `2d-designed-sine-bvp`, respectively. We see that, in general, diagonal, chevron, and circular orderings are most competitive. As the circular ordering does not generally perform better and is slightly more costly to compute, we henceforth consider only diagonal and chevron orderings. While Figure 4.1a illustrates a case where the row-major ordering appears to perform best, overall results show that row-major and column-major orderings are not generally competitive with diagonal and chevron orderings and typically result in slower decay of the error.

An alternative to using 2D basis functions is to map the entries of the unknown vector to be solved for in a locality-preserving way from 2D to 1D so they can be approximated by 1D basis functions. One such possibility is by mapping the unknowns according to a space-filling



(a) Order comparison test results for `su-ex3` with $n = 32$ (linear system of size $n^2 \times n^2$).



(b) Order comparison test results for `2d-designed-sine-bvp` with $n = 32$ (linear system of size $n^2 \times n^2$).

Figure 4.1: Comparison of various orderings of 2D basis functions.

curve [38, 39], such as the Hilbert curve first described by Hilbert in 1891. We investigated the use of space-filling curves such as the Hilbert curve for 2D test problems; however, despite their locality preserving property, the curve obtained by reordering according to the mapping is not smooth, which does not lend itself to being well-approximated based on the 1D results. Accordingly, space-filling curves are not an effective technique for approaching 2D problems using the compression-based method, and, hence, results are omitted.

4.4 COMPREHENSIVE RESULTS

To illustrate the overall effectiveness of the method, we present results of the best performing compression bases, as determined by the 1D testing, on the 2D test problems discussed in Section 4.1 with details in Appendix D.

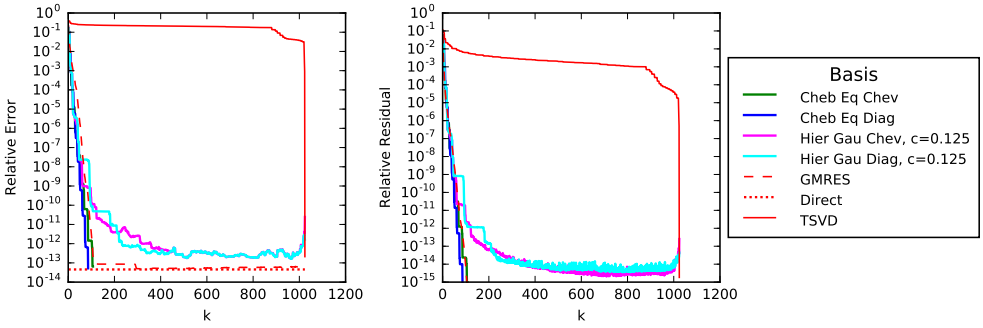
For each comprehensive test, results are also shown for GMRES and TSVD. When $\boldsymbol{A}$ is square and full rank, the error and residual for the solution from the direct solver in NumPy [30, 31] (or SciPy [32] if the matrix is stored in sparse format) are also shown, except for problems for which the true solution is obtained via a direct solver.

In Figure 4.2, the results of comprehensive tests are illustrated for `su-ex3` and `texas2d-ex1` discretized using simple discretizations. These are similar to the problems used for the targeted testing in Section 4.3; however, we compare the best performing bases against GMRES and TSVD. For these problems, as the discretization details are known, equally-spaced evaluation points are used for the basis functions in order to match the discretization points for both problems. For `su-ex3`, the results illustrated in Figure 4.2a show that the compression-based method with Chebyshev bases performs similar to and even slightly better than GMRES and significantly better than TSVD. It is also effective, though slightly less so with hierarchical Gaussian compression bases. In Figure 4.2b, with a Chebyshev basis, a low error is obtained at extremely small k relative to the problem size, achieving a minimum error at about $k = 15$, a far smaller value of k than GMRES. Together the results in Figure 4.2 clearly illustrate that, as in 1D, the compression-based method can be competitive with, and potentially significantly outperform, other methods for problems with smooth solutions discretized with simple discretizations.

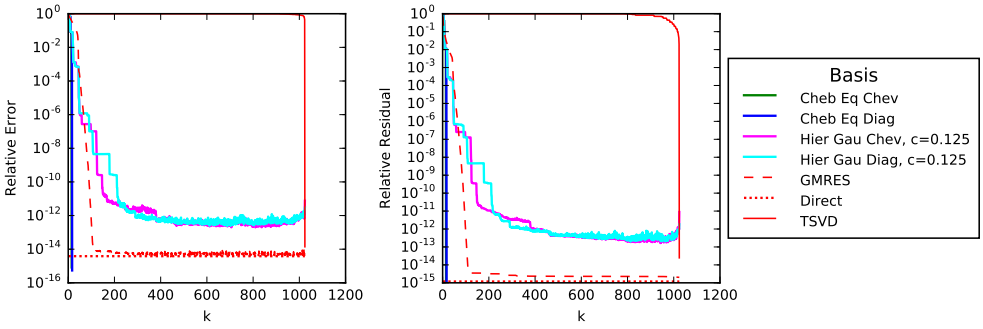
Figure 4.3 illustrates results for `fenics-simple`, a simple boundary value problem discretized with finite elements on a uniform square mesh. This is a problem for which the solution of the discretized linear system should reproduce the true solution function exactly, as the solution is quadratic and piecewise linear elements can reproduce a quadratic exactly on a uniform mesh [40]. As the mesh is a uniform finite element mesh on the unit square, equally-spaced evaluation points may seem to be a reasonable choice. However, as discussed in Section 2.3.2, we must ensure compatibility between the discretization and the evaluation points in the ordering of the spatial coordinates.

For illustrative purposes only, Figure 4.3a presents the results when using equally-spaced, row-wise ordered evaluation points. In contrast, Figure 4.3b illustrates results for the same problem using a Chebyshev compression basis with the evaluation points ordered to match the ordering of the discretization points according to the ordering of the degrees of freedom. This problem highlights the importance of ensuring that the evaluation points match the discretizations points, both in value and in ordering. With the evaluation points reordered to match the ordering of the degrees of freedom, the method achieves a low error at an extremely low value of k. In particular, Figure 4.3c illustrates the same results as Figure 4.3b, but truncated to illustrate with more detail the behavior for small values of k. From Figure 4.3c, it is clear that a good approximation with low error is obtained at $k = 6$, far less than the k required by GMRES to achieve an equivalent error.

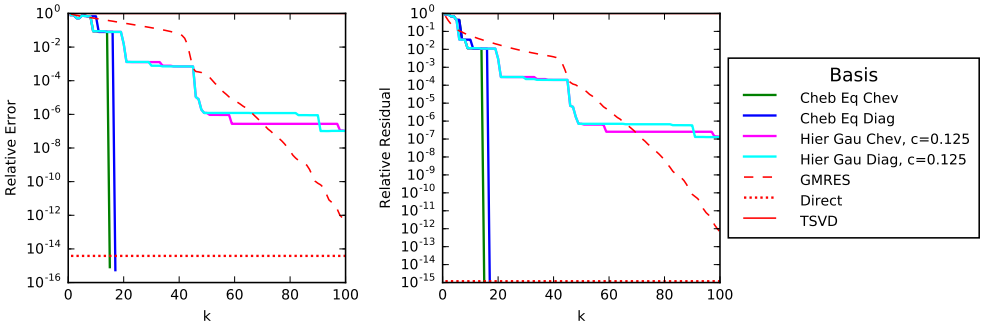
Figure 4.4 illustrates the overall performance of the compression-based method for solving a slightly more complex boundary value problem discretized with finite elements. This is the same test problem considered in Figure 4.2b, but discretized on a uniform finite element



(a) Relative error and relative residual for `su-ex3` discretized using Nystrom method with $n = 32$ (linear system of size $n^2 \times n^2$).



(b) Relative error and relative residual for `texas2d-ex1` test problem discretized using finite differences with $n = 32$ (linear system of size $n^2 \times n^2$).

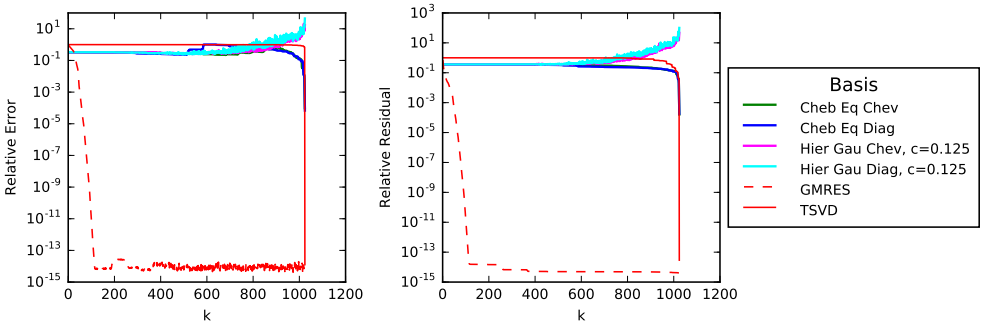


(c) View of Figure 4.2b zoomed in to show detail for initial values of k.

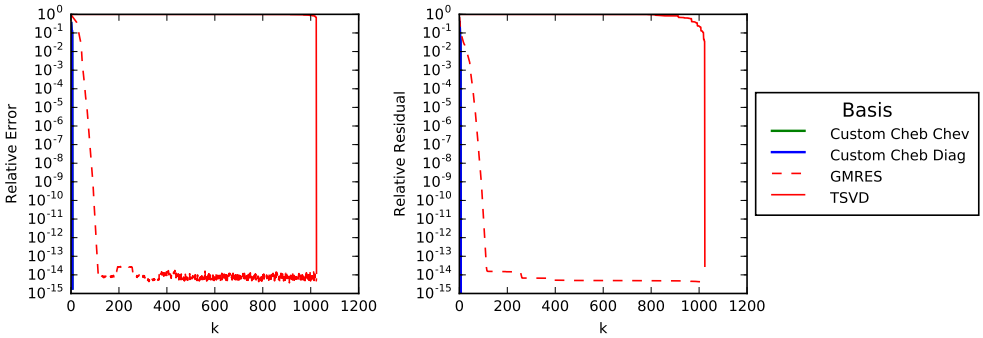
Figure 4.2: Comprehensive test illustrating relative error and relative residual for `su-ex3` and `texas2d-ex1` discretized with simple discretizations.

mesh and on an irregular finite element mesh over the unit square. As Figure 4.3 illustrates clearly the necessity to incorporate detailed knowledge of the both discretization points and ordering for problems discretized using finite elements, only results of the Chebyshev basis evaluated at the coordinates corresponding to the ordered degrees of freedom are illustrated in Figure 4.4.

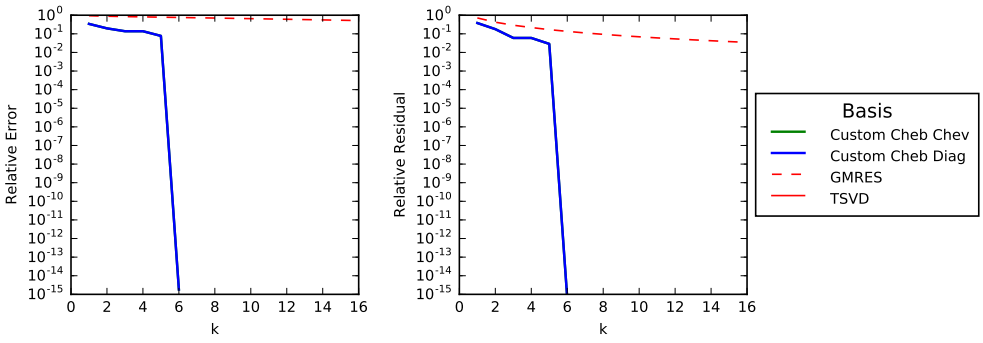
Figure 4.4a illustrates results with the uniform finite element mesh. While the initial decay of the error is rapid, the rate of decay of the error decreases after relatively few k, although the error does continue to decrease at a slower rate. Unlike Figure 4.3, where due



(a) Results with default (row-major flattened) ordering of compression basis evaluation points.



(b) Results when using discretization coordinates in ordering corresponding to degrees of freedom for basis evaluation points.



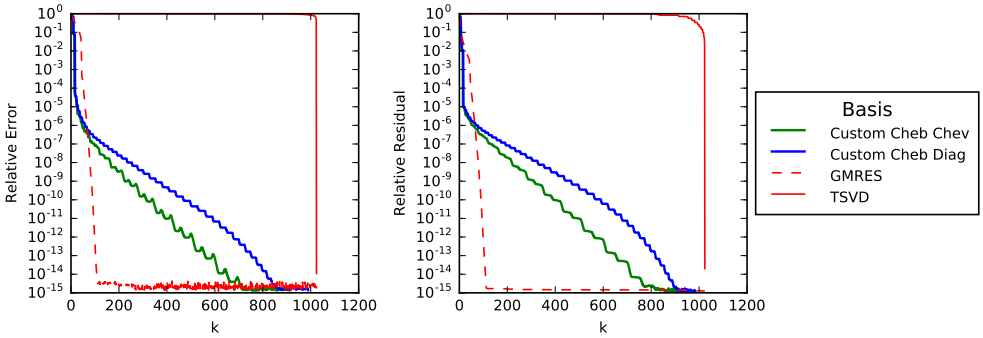
(c) View of Figure 4.3b zoomed in to show detail for initial values of k.

Figure 4.3: Comprehensive tests illustrating relative error and relative residual for `fenics-simple` test problem with $n = 32$ (linear system of size $n^2 \times n^2$) with both row-wise equally-spaced and custom basis evaluation points.

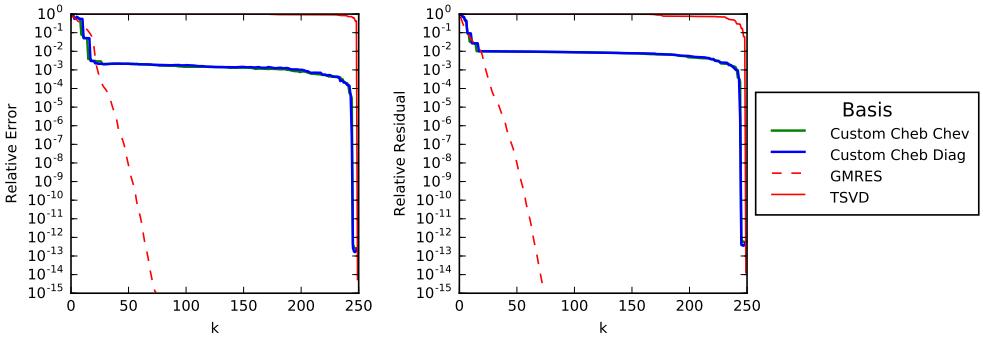
to the nature of the continuous problem and the choice of discretization the components of the solution to the linear system are exact nodal values, the approximate solution to the problem for which results are illustrated in Figure 4.4 includes discretization error as well as error in solving the linear system. In Figure 4.4b, results with the irregular finite element mesh are presented. The error initially decays rapidly, then plateaus at approximately 10^{-3} until $k \approx n$, at which point the error drops significantly. In both cases the compression-based method does not appear to achieve the minimum error until k is quite large. However, it is crucial to recall that, unlike in 1D, the error for these problems is not computed by comparing to the true solution of the continuous BVP, but by instead comparing to the solution of the linear system, and thus it does not reflect any discretization error. Based on various estimates, the discretization error for the uniform mesh is approximately 10^{-3} or 10^{-4} (depending on the specific error estimate), with the discretization error for the irregular mesh being a similar order of magnitude, but slightly higher. Thus, despite superficial appearances in Figure 4.4, GMRES is not outperforming the compression-based method as, due to the manner in which the error is computed, accuracy at an error level of 10^{-15} (or any value less than discretization error) is effectively meaningless, as it does not imply a more accurate solution to the continuous BVP. In fact, in Figure 4.4 we see that the compression-based method exhibits a initial rapid decay of the error. In particular, in Figure 4.4a the error for the compression-based method drops far faster than for GMRES and an error within the range of discretization error is obtained with far smaller k than for GMRES, indicating that the compression-based method is actually significantly outperforming GMRES.

In Figure 4.5, we illustrate results with the best performing 2D bases and 2D orderings for `ir-inverse-interpolation`. For this problem, we do not utilize details about the discretization, but instead we simply evaluate the basis functions at equally-spaced points. From these results, we see that both hierarchical Gaussians and Chebyshev polynomials with the diagonal and chevron orderings perform quite well, achieving a low error at approximately $k = 25$ for a linear system of size 1024. We also see that all of the bases and ordering combinations achieve a lower error overall than conventional methods, which make little to no progress to the solution. This illustrates a 2D problem for which the performance of the compression-based method is quite impressive.

Figures 4.6a and 4.6b illustrate results with the compression-based method for the image deblurring problems `ir-image-deblurring-smooth` and `ir-image-deblurring-hst`, respectively. As with Figure 4.5, we do not utilize detailed knowledge of the problem such as the type of blurring or discretization details. However, as we are working with images, equally-spaced evaluation points are a logical choice as pixels are inherently equally spaced. In Figure 4.6a, we see that the compression-based method performs quite well with the



(a) Discretized using finite elements with a uniform mesh with 1024 total degrees of freedom.



(b) Discretized using finite elements with an irregular mesh with 249 total degrees of freedom.

Figure 4.4: Comprehensive tests illustrating relative error and relative residual for `texas2d-ex1` discretized with finite elements.

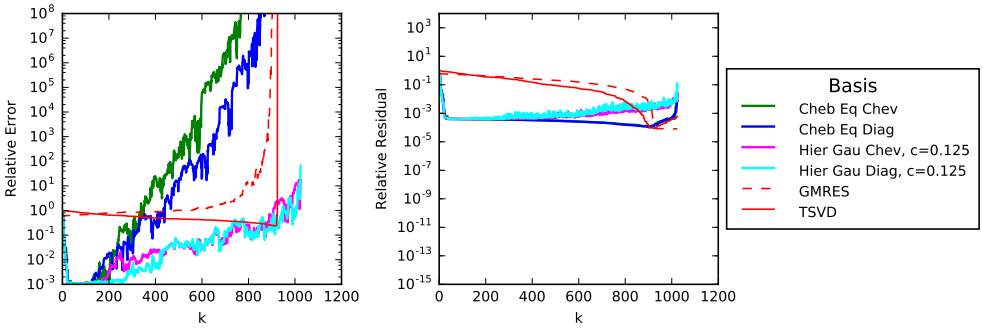
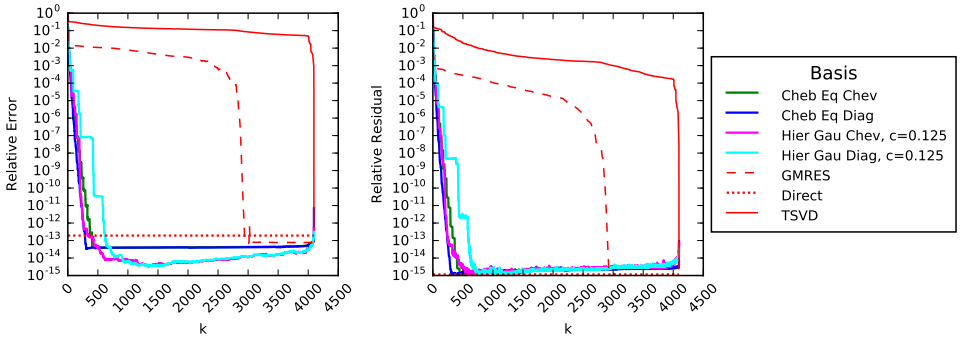


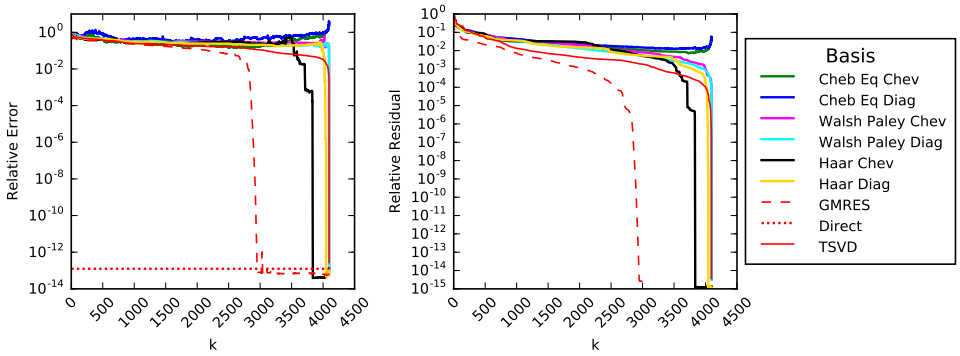
Figure 4.5: Relative error and relative residual for `ir-inverse-interpolation` with $n = 32$ (linear system of size $n^2 \times n^2$).

Chebyshev basis, achieving a minimum error at $k \ll n$, with hierarchical Gaussians also performing reasonably well. Additionally, all of the compression bases illustrated achieve a good solution with significantly smaller k than required by TVSD and GMRES. As the true solution (unblurred) image shown in Figure D.6b is a smooth function, the strong performance is expected. For Figure 4.6b, the true solution (unblurred) image is a picture of the Hubble space telescope, which we would not expect to be well approximated by smooth functions. Therefore, Figure 4.6b includes bases that do not regularly perform well, but

are motivated by domains other than approximation of smooth functions. Even with these bases, we see that none of the bases perform well; however, based on the 1D testing, this is not unexpected as the jumps in the discrete bases need to occur at the locations of the edges in the image, which often does not occur until sufficiently oscillatory basis vectors are included.



(a) Relative error and relative residual for `ir-image-deblurring-smooth` with $n = 64$ (linear system of size $n^2 \times n^2$).



(b) Relative error and relative residual for `ir-image-deblurring-hst` with $n = 64$ (linear system of size $n^2 \times n^2$).

Figure 4.6: Comprehensive tests for `ir-image-deblurring-*` test problems.

In Figure 4.7, we illustrate results with the best performing 2D bases and 2D orderings for `ir-inverse-diffusion`. As with the problems `ir-inverse-interpolation` and `ir-image-deblurring-*`, this problem is not considered in its continuous form and we do not utilize in-depth discretization details. From these results, we see that none of the combinations of basis and ordering perform particularly well for this problem in comparison to TSVD or GMRES. For some values of k, however, they do achieve a lower error than the error in the solution from a direct solver.

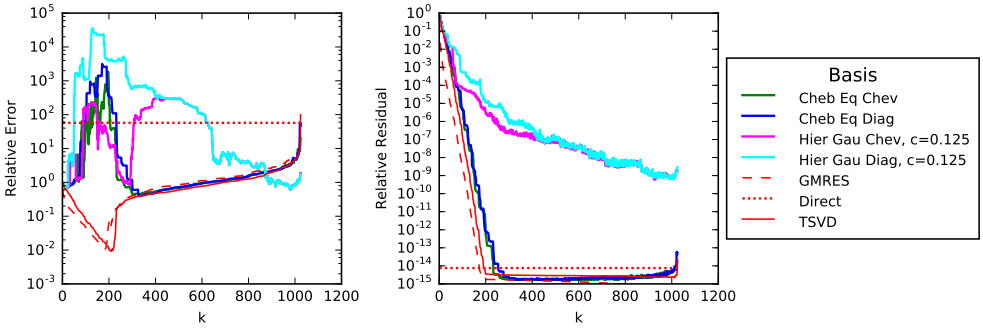


Figure 4.7: Relative error and relative residual for `ir-inverse-diffusion` with $n = 32$ (linear system of size $n^2 \times n^2$).

4.5 TESTING SUMMARY

Overall, the 2D test results illustrate that the compression-based method has the potential to perform well on 2D problems. As with 1D test problems, for best performance the evaluation points for the compression basis should match the discretization points. For 2D problems with reasonably smooth solutions discretized using simple discretizations where the discretization points are known and the evaluation points are chosen to match the discretization points, the method performs well. However, for other 2D test problems, the ability to match the evaluation and discretization points and the pattern of the solution are more challenging in 2D, particularly because more complex discretizations are often used in 2D. If the discretization points are known, custom discrete bases obtained by evaluating the traditional basis functions at the discretization points in the order they are used for the discretized linear system can alleviate some issues that arise with more varied orderings of the degrees of freedom for 2D problems, such as those resulting from finite element discretizations. Ultimately, the performance of the compression-based method in 2D is tied closely to the ability to match the points used in the discretization with evaluation points for the basis functions and the ability of the basis functions to approximate the pattern of the solution with relatively few basis functions, which are the same considerations that are crucial for its effectiveness for 1D test problems.

Chapter 5: Stopping Criterion

The results presented thus far have been run until $k = n$ or until a relative residual tolerance of 10^{-15} has been met, in order to illustrate the full behavior of the error and the residual. In practice, however, we will not achieve significant compression unless we can stop with $k \ll n$. The initial formulation of the incremental method in Algorithm 1.2 utilizes the standard stopping criterion for iterative methods: to continue until a specified residual tolerance is met. This is not an ideal stopping criterion for Algorithm 1.2, however, because it may force running well beyond the point of diminishing returns, or worse, to the point that the error *increases*, as in Figure 5.1, thus degrading the solution. In order for Algorithm 1.2 to be a viable method in practice, we need an automated stopping criterion that maximizes compression and is robust with regards to this potential degradation of the solution with increasing k.

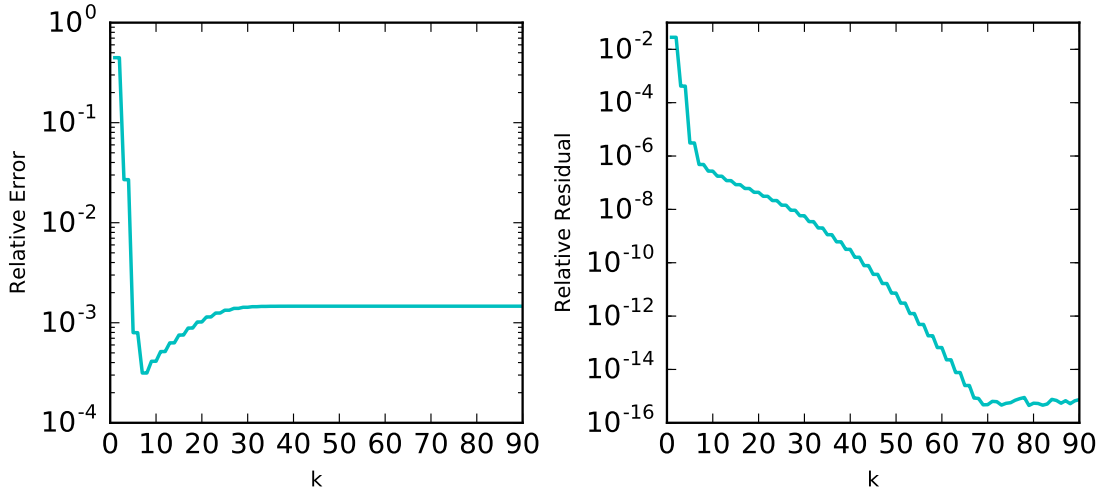


Figure 5.1: Example results (truncated at $k = 90$) for `designed_sine_ie` with $n = 128$ discretized using composite trapezoid quadrature with Chebyshev polynomials for the compression basis illustrating potential degradation of the solution as k increases.

While the results presented previously illustrated both the residual and the error, in principle the true solution function is not generally known, as otherwise there would be no need to solve the system. Consequently, any potential stopping criterion cannot make use of the error, but must rely on other information available without knowing the true solution.

In this chapter, we first discuss techniques for related methods and their performance when used in conjunction with the compression-based method. We then provide a brief overview of the quantities available to be monitored, present a heuristic stopping criterion based on one of these quantities, and numerically confirm its effectiveness on a variety of test problems with varying noise levels (where applicable).

5.1 CURRENT RELATED TECHNIQUES

Stopping criteria for conventional iterative methods are disccussed in [41, 42, 43, 44, 45, 46, 47]; however, they are either not applicable to our method or do not significantly enhance the standard relative residual tolerance in conjunction with our method.

A variety of techniques have been developed for determining the regularization parameter when using traditional regularization methods for solving ill-posed problems (e.g. TSVD, Tikhonov regularization). For ill-posed problems such as integral equations, in effect, Algorithm 1.2 acts as a regularization method, with k serving as a discrete regularization parameter. Therefore, we summarize these approaches briefly and discuss their performances when used in conjunction with Algorithm 1.2. More detailed descriptions of these techniques can be found in [6, chapter 5] and [48, chapter 7].

5.1.1 Summary of Techniques

The discrepancy principle, typically attributed to [49], chooses the regularization parameter such that the 2-norm of the residual equals the 2-norm of the error in the right-hand side, $\|e\|_2$. It requires knowing either $\|e\|_2$ or the noise distribution

The L-curve, proposed in [50, 51], chooses the regularization parameter to correspond to the L-shaped corner in a log-log plot of the solution norm versus the residual norm. Finding the corner of the L-curve requires data for regularization parameter values on both sides of the corner and typically involves maximizing the curvature (and potentially some form of smoothing).

Generalized cross validation (GCV) [52], attempts to choose the regularization parameter as the value for which the regularized solution will best predict omitted data values by minimizing the GCV function.

The normalized cumulative periodogram (NCP) analysis technique, [53, 54, 55], utilizes the DFT and power spectrum of the residual vector to choose the regularization parameter for which the residual vector most resembles white noise.

5.1.2 Discussion of Performance

We have evaluated each of these traditional regularization parameter techniques for use in combination with Algorithm 1.2. While these methods may perform well for some test problems and noise levels, no single criterion performed well across all test problems and noise levels. For instance, occasionally the residual tolerance specified by the discrepancy

principle was too low to be achieved (even with the additional safety scale factor). For the L-curve criterion, in some cases, there is no L-shaped corner in the solution norm versus residual norm plot (it is in fact a straight line). In other instances, even when a corner is present, it is not uncommon for many values of k to make up the single corner, including k for which the error has begun increasing significantly. For the GCV technique, tests show that, for some problems where the optimal value of k occurs before the residual bottoms out, the GCV criterion does not stop early enough. For some problems with the NCP analysis technique, there is no transition to high frequency noise, and, in multiple cases, the values of k that fall within the white noise range are those for which the error in the computed approximate solution is increasing.

5.2 POTENTIAL QUANTITIES FOR DETECTING TERMINATION

Although we cannot monitor the error, we can monitor a variety of other relevant quantities throughout the incremental process, such as the individual entries and the norms of the approximate solution vector $\boldsymbol{x}_k$, the vector of coefficients $\boldsymbol{z}_k$, and the residual vector $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$, and the diagonal entries r_{kk} in the incremental QR factorization.

While it may seem that some of these quantities could provide useful information for detecting the effective rank of the projected problem or the representation of the solution, most are ill-suited for use as a termination criterion. For instance, one could monitor $\|\boldsymbol{x}_k\|_2$, as the solution norm may increase as noise begins to dominate the solution. However, there is no guarantee the solution norm will increase, or that it will increase only when noise dominates the solution. The coefficients $\boldsymbol{z}_k$ indicate the contribution of the basis vectors, but as the bases are not generally orthogonal, adding a new basis vector often changes the entire vector $\boldsymbol{z}_k$, making it difficult to monitor the contribution of each individual basis vector. Furthermore, not all basis vectors will contribute to the solution, so a small coefficient in magnitude does not necessarily indicate convergence (in fact, often due to an even or odd solution every other basis vector makes no contribution to the solution). Similarly, although a nearly zero diagonal entry r_{kk} can indicate near rank deficiency of $\boldsymbol{Y}_k$, there is no guarantee that subsequent basis vectors will not extend the subspace.

Utilizing many of these quantities as termination indicators is insufficient due to the unpredictable nature for future basis vectors. However, the residual, although its value is not a direct indicator of convergence, is guaranteed to be monotonically nonincreasing as k increases (absent rounding error). Thus, the residual may be a suitable choice if additional information regarding the progress towards the solution can be detected based on the detailed behavior of the residual.

5.3 PROPOSED STOPPING CRITERION

We present a multi-faceted, heuristic stopping criterion that attempts to stop at the point (k) of diminishing returns in the incremental version of the algorithm. Similar to some of the stopping criteria summarized in Section 5.1.1, the criterion we present is motivated by the idea of extracting additional information from the residual. In particular, although the value of the residual alone may not be a sufficient stopping criterion for the reasons previously discussed, the detailed behavior of the decay of the residual can provide additional information.

Specifically, as Algorithm 1.2 computes a least squares approximation to the solution over a nondecreasing sequence of subspaces, the 2-norm of the residual is guaranteed to be monotonically nonincreasing as k increases (absent rounding effects). Although the residual is monotonically nonincreasing, there is often a "bend" (sharp change in curvature) in the plot of the residual 2-norm versus k when plotted on a semilogarithmic plot, using a logarithmic scale on the vertical axis (i.e., semilogy). For some problems and noise levels, this bend occurs when the residual "bottoms out." In other instances, however, a bend occurs at a change in the behavior of the residual before the residual has permanently flat-lined, and in such cases, there is often a second distinct bend when the residual finally flattens permanently.

The natural question, then, is how and why such bends can be relevant to detecting an appropriate termination point? In the case of two bends, the initial bend indicates a transition from capturing the overall behavior of the solution of the continuous problem to making additional progress towards solving the linear system more accurately, which may not necessarily improve the overall accuracy and may even increase the error for integral equations. More specifically, for problems that do not benefit from regularization, our test problems have illustrated that when the residual exhibits two distinct bends, the first bend indicates the point at which the accuracy of the approximate solution to the continuous problems is around the discretization error, and once the approximate solution is at the level of the dominant discretization error, no additional accuracy will be gained with respect to the original continuous problem. Intuitively, such an initial bend in the residual curve is due to the suddenly diminished returns to be gained from further iterations once the approximate solution is already approximately within the discretization error. In cases with a single bend, which occurs when the residual stagnates, the linear system has been solved as accurately as possible and the incremental process should be terminated as there is no additional progress to be made to the true solution to the continuous problem. Either scenario indicates that the incremental process should be terminated at the first bend detected, regardless of whether there may be more than one bend.

For example, consider Figure 5.1, for which we see two distinct bends in the semilogy plot of the residual. The first bend occurs at approximately $k = 7$, where the rate of decay of the 2-norm of the relative residual transitions from quite rapid to a still distinct, but far less rapid, decay. Correspondingly, the error attains its minimum and begins to increase as the approximate solution transitions from capturing the behavior of the true continuous solution to progressing towards approximating the solution to the linear system. A second bend occurs at approximately $k = 70$, after which the behavior of the residual is dominated by rounding error and is no longer monotonically nonincreasing.

For the single stopping criterion to be effective for both integral equation problems and boundary value problems, we present a multi-faceted stopping criterion that combines detecting a desired bend with more traditional aspects, such as a residual tolerance. The primary aspects of the stopping criterion are:

- Residual tolerance: stop if a provided relative residual tolerance is met
- Residual increasing: stop if the 2-norm of the relative residual increases significantly
- Bend in residual plot: stop if a valid bend has been detected in the semilogy relative residual plot
- Residual flat-lining: stop if a valid flatlining of the relative residual on the semilogy plot has been detected

These primary aspects must detect changes in the behavior of the residual in a robust manner with respect to other behaviors often present in the residual decay that are not indicative of termination. For instance, in Figure 5.1, while the overall behavior of the residual shows two clear bends, a stair-stepping behavior (due to odd/even effects) is also present that is not indicative of a larger change in the decay of the residual. To be effective, the bend detection must be robust to such stair-stepping. Specifics on overcoming such issues are discussed in the details for each of the individual components.

The full stopping criterion is illustrated in Figure 5.2 and details of the individual components are described in the following sections.

5.3.1 Traditional Residual Components

The first two components of the multi-faceted stopping criterion are more traditional components based solely on the value of the residual rather than the overall behavior of the residual. The first component is the traditional relative residual tolerance often used as

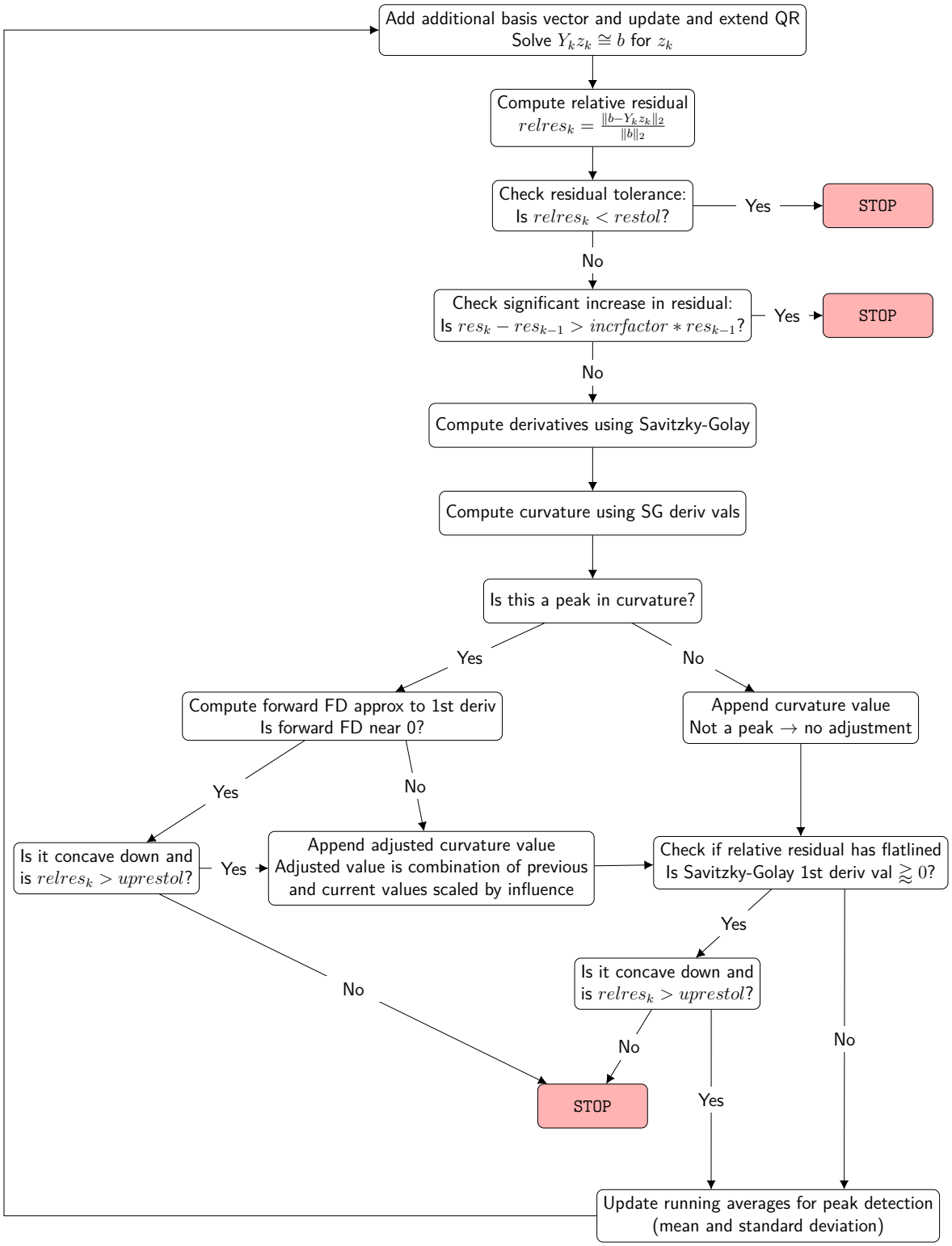


Figure 5.2: Diagram of heuristic stopping criterion decision flow.

the stopping criterion in many iterative methods. The default value is set sufficiently low that this component is unlikely to trigger before the relative residual has reached the level of rounding error. This component is primarily included so users can easily incorporate a priori knowledge about a relative residual tolerance after which it is not worthwhile continuing. This component is a sufficient condition for stopping, but not necessary; it will not force the method to keep going until the tolerance is met if one of the other components detects that the incremental process should terminate.

The second component is motivated by the fact that absent effects of rounding error and noise, the 2-norm of the residual is guaranteed to be monotonically nonincreasing. While rounding error may cause the 2-norm of the residual to increase slightly, if the 2-norm of the residual increases significantly, the incremental process should stop. To ensure that rounding error does not cause this component to indicate that the method should stop prematurely, this check is implemented by testing whether the increase is sufficiently large relative to the 2-norm of the previous residual.

To complement these, an upper relative residual tolerance is also included. Unlike the traditional residual tolerance, which indicates termination when the relative residual falls below the given tolerance, the upper relative residual tolerance prevents stopping too early by specifying a relative residual that must be achieved before the method will stop due to satisfying the other, more complex components discussed in the next sections. The default value is set to ensure the method has made some progress towards solving the linear system before allowing the stopping criterion to signal to stop, and it is designed to prevent problems that are initially vastly underresolved from stopping prematurely.

5.3.2 Residual Decay Behavior

Detecting a change in the behavior of the residual decay (i.e., identifying the location of a bend described in Section 5.3) is similar to detecting the corner of the L-curve in the L-curve criterion. Accordingly, we adopt a similar technique applied to the semilogy plot of the relative residual 2-norm versus k, as we wish to stop at the point of maximum curvature in this plot. Unlike the L-curve criterion, we cannot have access to the entire plot or even a few values of k spread out over the entire plot, as running with a large value of k will nullify any potential computational effectiveness. Instead, we must adaptively find bends "on the fly," based primarily on past information and requiring information from at most a few steps further in Algorithm 1.2. We must first, however, approximate the curvature and then detect for which k the curvature is at a peak.

5.3.3 Curvature Approximation

The curvature of a 1D function $y = f(x)$ is defined as

$$\text{curvature} = \frac{|y''|}{(1 + y'^2)^{\frac{3}{2}}}.$$

Therefore, to approximate the curvature, we must have approximations of the first and second derivatives. While there are a variety of ways to approximate derivatives and the curvature, we must account for the fact that the semilogy residual plot is inherently not smooth because k is discrete. Furthermore, in some instances, such as in Figure 5.1, the residual plot may exhibit stair-stepping, with no change every other k, due to other aspects (such as approximating an even or odd function with polynomials, where every other basis vector has no effect).

Initial Smoothing

The unsmooth or potentially jagged nature of the semilogy relative residual plot has the effect of creating many small "knees" that do not actually indicate a more general change. To address this issue, we utilize the Savitzky-Golay convolution filter coefficients [56] to compute the smoothed log relative residual value, along with first and second derivative values, at a given point. The Savitzky-Golay convolution coefficients allow for computation of the function value, 1st, and 2nd derivatives obtained from fitting a low degree polynomial over a certain window of data.

In general, we utilize the Savitzky-Golay filter with a window of five points and a polynomial degree of two, where the derivatives and function values are evaluated at the center of the interval. However, this prevents the ability to determine smoothed values for $k = 1$ and $k = 2$. To address this issue, for $k = 1$, a 2nd degree polynomial is fit over the log relative residual values for $k = 1, 2, 3$ and then the function and its derivatives are evaluated at the initial endpoint. Likewise, for $k = 2$, a 2nd degree polynomial is fit over the four values $k = 1, 2, 3, 4$, and the function and its derivatives are evaluated at $k = 2$. In all cases, the window extends two beyond the current k. So, to compute the smoothed log relative residual value or the derivatives at a point k, residual information through $k + 2$ is required.

Peak Curvature Detection

Once the smoothed log relative residual values and 1st and 2nd derivatives for a given point are determined, they can be used to compute an approximate curvature value. To detect a peak in the curvature on the fly, we use an adaptation of the smoothed z-score algorithm presented in [57], which is designed to take a vector of data values and determine whether a given value is a peak (or signal as it is called in [57]) based only upon the previous data values. When given a new data point, the algorithm compares the new data point by computing the number of standard deviations the new data point lies away from the moving mean (where the length of the moving mean is controlled by `lag`). If this z-score is above a preset threshold, then the new data point is declared a peak. To ensure that a single peak does not prevent future peaks from triggering, when computing the new moving mean and moving standard deviation, a filtered value of the peak is used. Specifically, the filtered value is a scaled combination of the current value and the previous filtered value, where the scaling is controlled by the `influence` parameter.

For use as a component of the stopping criterion, we make a few minor adjustments to the smoothed z-score algorithm as it is presented in [57]. Specifically, the original algorithm detects "signals" that may be higher or lower than the mean. However, as we are looking only for peaks, we adapt the results to look only for curvature values above the moving average by at least `threshold` standard deviations. Additionally, the algorithm as presented cannot begin detecting peaks until the `lag` + 1 data point (as `lag` data points are required for the moving average and standard deviation). While this is important for forming a baseline mean, it prevents the ability to detect a peak in the first few k. As some of the numerical results in Chapter 3 illustrate, there are problems for which the optimum solution is at k less than `lag`. The inability to detect any peaks in the curvature for small k could cause the peak to be missed entirely. To counteract this, when k is less than `lag` but at least two, we simply compute a moving average of the $k-1$ previous curvature values, switching to a moving average of length `lag` when $k >$ `lag`.

5.3.4 Additional Components

In addition to just checking for a peak, checks are included within the stopping criterion to verify that a peak is a "good" peak before stopping. First, it is possible that there could be multiple peaks close together, and stopping at the first peak may not be the most desirable choice. To account for this, we check a 2nd-order accurate forward difference approximation to the first derivative to verify that the logged relative residual is reasonably

flat immediately following the peak. This additional check requires two Savitzky-Golay smoothed log relative residual values beyond the current. Therefore, because each Savitzky-Golay smoothing window requires two values beyond the current k, in total the stopping criterion requires information for four values beyond the current k to decide whether the current k is the appropriate point at which to stop.

Also, as it is possible for the curvature to be large at any bend in the plot, it is important to consider the individual shape of any point of maximum curvature. If the current k is a peak in the curvature and the semilogy relative residual plot is concave down at that point, the concave down aspect is indicative of a transition from a slowly decaying residual to a more quickly decaying residual, which would generally indicate a poor place to stop. So, the stopping criterion includes a check on the second derivative when a peak is detected, to ensure that it does not stop at a point of high curvature where it is concave down.

In addition to those checks, in a few cases, it is possible for there to be no "good" peaks in the curvature before the residual effectively flatlines. To ensure this is detected, if for a given k no good peak is detected, the first derivative value computed using Savitzky-Golay is checked and if it is near zero or positive and the second derivative does not indicate it is concave down, then stop, as this indicates the residual has effectively not changed for five values of k in a row, and that there is no indication it will start decreasing soon.

5.3.5 Specific Parameter Values

The smoothed z-score algorithm for robust peak detection presented in [57] requires multiple input parameters to determine what is detected as a peak and control its influence. Additionally, the other checks described in Section 5.3.1 and Section 5.3.4 also require individual tolerances. For our tests, the tolerance values and parameters in Table 5.1 are used. We intentionally use an unrealistically low relative residual tolerance to test the effectiveness of the remaining components of the stopping criterion. While not requiring user specification, as the other components of the stopping criterion are designed to detect when the residual ceases decreasing, this tolerance would likely be user specified based on additional information regarding the problem or a desired residual known by the user. Similarly, while the upper relative residual tolerance is designed to be sufficient with the default value, if a user has additional knowledge about the problem and the desired solution, this tolerance could also be adjusted by the user.

threshold	3.0
influence	0.4
lag	4
relative residual tolerance	$< 10^{-16}$
upper relative residual tolerance	10^{-1}
residual increasing factor	0.5
forward difference deriv tolerance	> -0.25
Savitzky-Golay deriv tolerance	$> -.01$
concave down tolerance	$< -.02$

Table 5.1: Stopping criterion parameter and tolerance values.

5.4 SELECTED RESULTS

As with previous numerical results presented, all tests are performed using Python 2.7 with use of functions and data structures from the NumPy and SciPy packages wherever appropriate. This includes utilizing the Savitzky-Golay filter functions in `scipy.signal` for all of the standard window scenarios (beyond the initial two iterations, where one-sided or lopsided windows are used for which the coefficients are calculated by actually fitting the log relative residual values and differentiating the resulting polynomial). To limit unnecessary clutter, we illustrate results only for the two best performing compression bases: Chebyshev polynomials and hierarchical Gaussians with $c = 0.125$.

As discussed in Appendix B, in practice the right-hand sides for ill-posed problems are often contaminated by noise. Thus, a reasonable automated stopping criterion must be robust for various noise levels. Accordingly, for integral equations, the results presented here have noise added to the right-hand side for various noise levels. The noise level will be specified as η (see Appendix B for details on the type of noise added and how η controls the level of noise).

For each of the test problem and noise level combinations, we continue as in the previous plots illustrating both the relative error (left) and the relative residual (right). The plots illustrating relative error are included only to evaluate the effectiveness of the stopping criterion; neither the true solution nor the error are utilized as part of the stopping criterion. As with the comprehensive results, the red solid, dashed, and dotted lines represent TSVD, GMRES, and a direct solver and are included for comparison purposes. The other lines illustrate results for Algorithm 1.2. For each of these lines, a star denotes the k at which

the stopping criterion indicates it should stop. The solid line indicates the values of k for which actual results were obtained in Algorithm 1.2 to make the determination to stop at the starred value of k. As described in Section 5.3.4, in total the incremental algorithm must have information for four values of k beyond the current unless the stopping criterion triggers termination based on the residual tolerance or the increasing residual. So, in most cases, the solid line will extend four values beyond the starred k. The dotted lines continuing after the end of the solid line are included solely to provide context and illustrate the behavior shortly after the k at which the stopping criterion signals to stop. The k range in the dotted portion of the line is not required to be run in order to stop at the starred value of k.

For comparison, we first illustrate the results of the stopping criterion in Figure 5.3 for the noiseless `designed_sine_ie` problem illustrated as motivation in Figure 5.1. This clearly illustrates that the stopping criterion is able to detect the change in the behavior of the residual and signal for the method to stop at a relatively small k, even though the residual is still decreasing. Examining the error, we see that the error is nearly at its minimum at the point the stopping criterion signaled to stop.

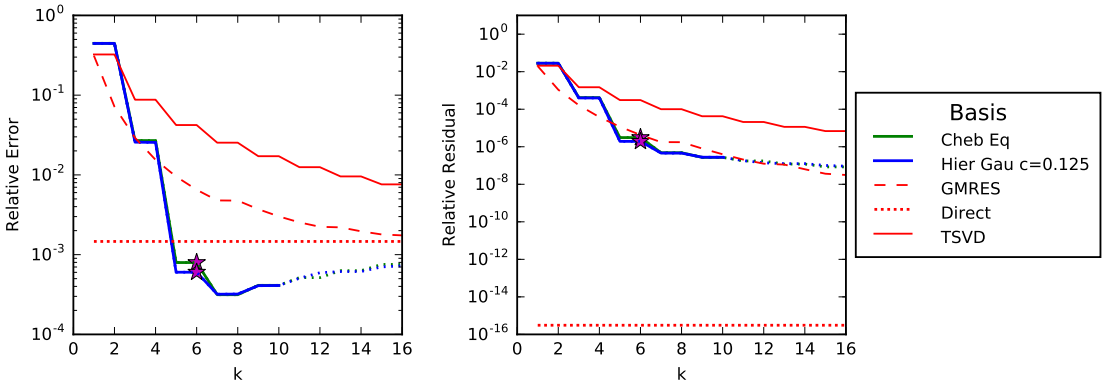
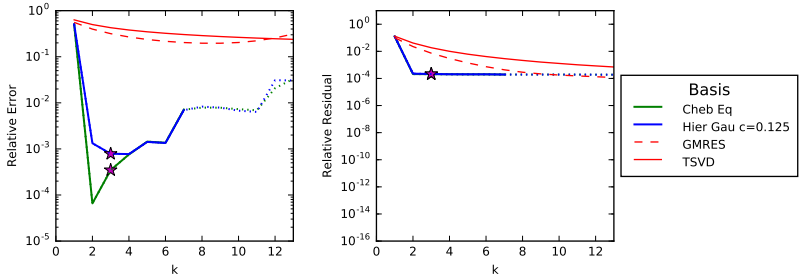


Figure 5.3: Stopping criterion results for `designed_sine_ie` with $n = 128$ discretized using composite trapezoid quadrature with no noise.

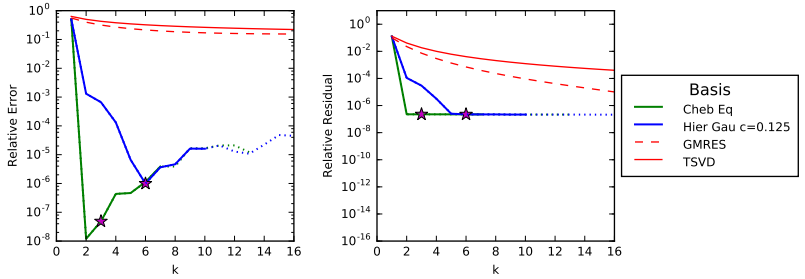
Figure 5.4 illustrates the performance on the integral equation test problems considered in Section 3.6. Each row corresponds to a problem, and each column represents a different noise level. Examining Figure 5.4, it is clear that, in all cases, the stopping criterion chooses a reasonable value of k at which to stop. While the value of k chosen for each may not be the truly optimum k, it is often very close to the optimum k, and the error is reasonably low at the k determined by the stopping criterion. Examining across the rows, although the actual results for the solution of the problem may change, the performance of the stopping criterion stays roughly the same. For instance, in Figure 5.4a the minimum error attained is approximately three to four orders of magnitude higher than in Figure 5.4b due to the

higher noise level in the former; however, in both cases the stopping criterion chose a k close to the optimum k to stop.

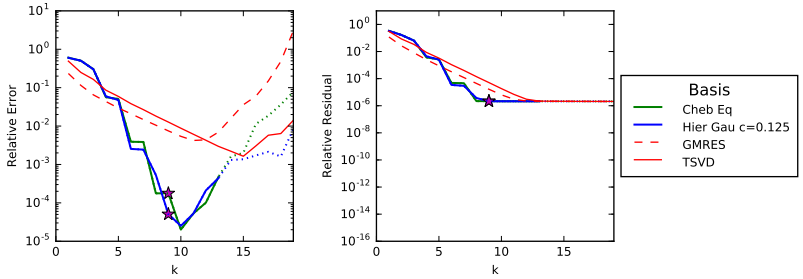
Figure 5.5 illustrates the results of the stopping criterion on the various boundary value problems considered in Section 3.6. Unlike integral equations, boundary value problems do not normally have noise in the right-hand side, so no artificial noise is added, and there are no longer multiple sets of results for each problem. In Figure 5.5, we see that for the majority of the problems, the stopping criterion is sufficient. Although it may not capture the exact k when the error reaches the minimum, it still indicates a reasonably low k. Unlike ill-posed problems, the boundary value problems do not exhibit the increase in error while the residual continues to decrease, so continuing a few k further is not as problematic because the solution is not degrading. If one had knowledge of a sufficient relative residual, the relative residual tolerance could be adjusted; however, even with the default tight tolerance the stopping criterion determines a reasonable value at which to stop.



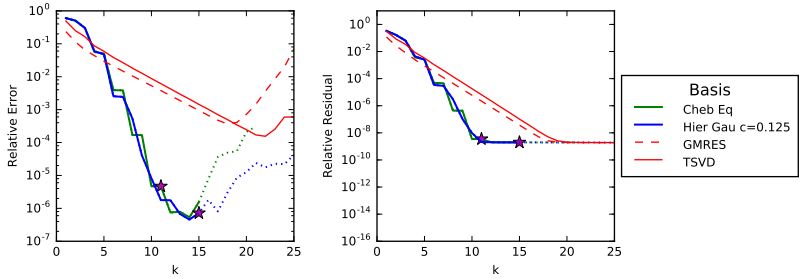
(a) Results for `deriv2_c1` with noise level $\eta = 10^{-5}$.



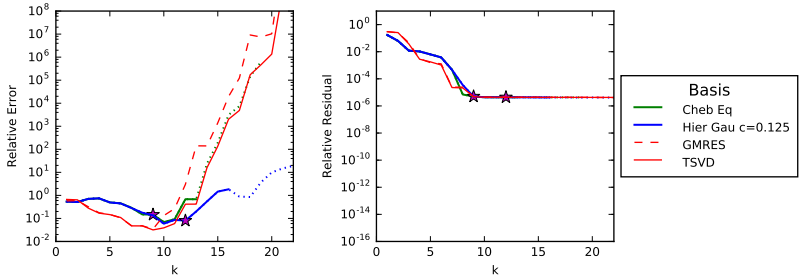
(b) Results for `deriv2_c1` with noise level $\eta = 10^{-8}$.



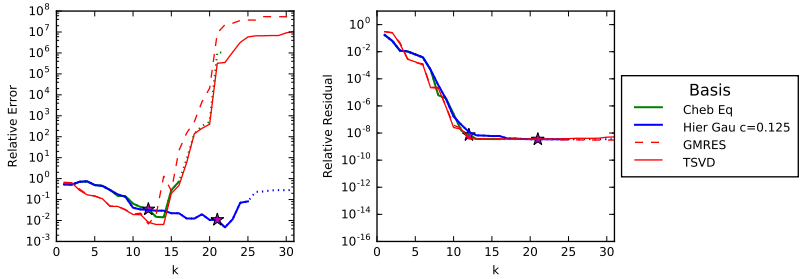
(c) Results for `gravity_c1` with noise level $\eta = 10^{-5}$.



(d) Results for `gravity_c1` with noise level $\eta = 10^{-8}$.

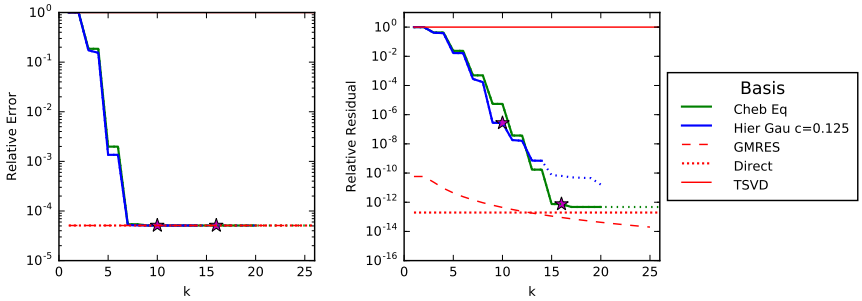


(e) Results for `shaw` with noise level $\eta = 10^{-5}$.

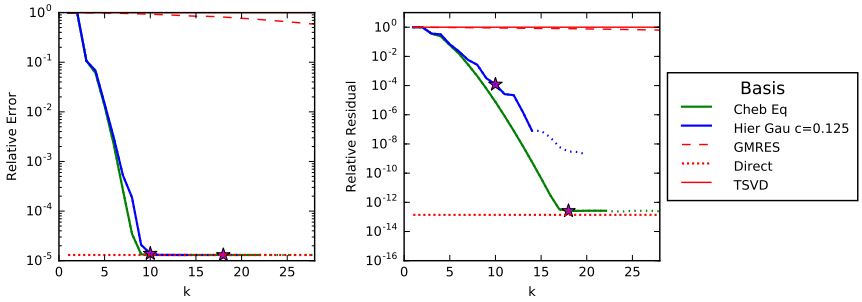


(f) Results for `shaw` with noise level $\eta = 10^{-8}$.

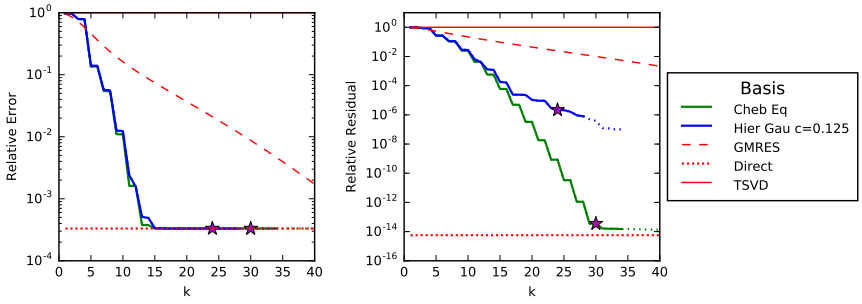
Figure 5.4: Stopping criterion performance results for integral equation problems with varying noise levels. All problems are discretized using composite trapezoid quadrature with $n = 128$.



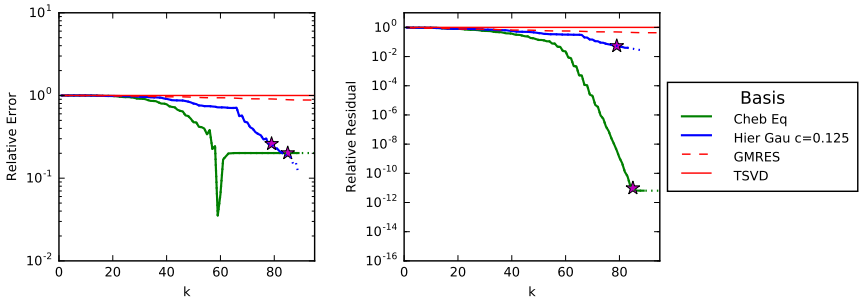
(a) Results for `designed_sine_bvp` with $n = 128$.



(b) Results for `fornberg` with $n = 128$.



(c) Results for `greengard-ex1` with $n = 128$.



(d) Results for `greengard-ex3` with $n = 512$.

Figure 5.5: Stopping criterion performance results for boundary value problems discretized using finite differences with equally spaced points.

In addition to 1D test problems, we also illustrate that the stopping criterion is effective in 2D. To examine its effectiveness, we first remark that Figure 4.4a, for the 2D test problem `texas2d-ex1`, exhibits similar behavior to Figure 5.1, with a generally jagged plot and a

distinct bend early in the incremental process. Although on the scales plotted the jagged behavior looks quite similar, the width of the stair steps is greater for 2D problems. Because of the greater jaggedness in 2D, a wider smoothing window is necessary in order to ensure that the stopping criterion is not fooled into stopping prematurely by the wide stair steps, which are introduced by the inherent need for a linear ordering of the 2D grid of basis functions. Figure 5.6 illustrates results of the stopping criterion applied to a uniform finite element discretization of `texas2d-ex1` with a Savitzky-Golay smoothing window size of 9. Although the stopping criterion goes slightly beyond the visible large bend, which is expected due to the additional smoothing required, it detects a reasonable point to stop after which the accuracy has reached the level of discretization error and only marginal additional progress to the solution of the linear system is gained with increasing k. In particular, when compared to the full results for the compression-based method and traditional methods shown in Figure 4.4a, not only is the compression-based method outperforming GMRES at the detected stopping point, but GMRES would continue to iterate far beyond discretization error, until the residual reaches rounding error or an arbitrary residual tolerance. Thus, the compression-based method can have a significant edge over GMRES, even beyond potentially achieving a lower error at a given k.

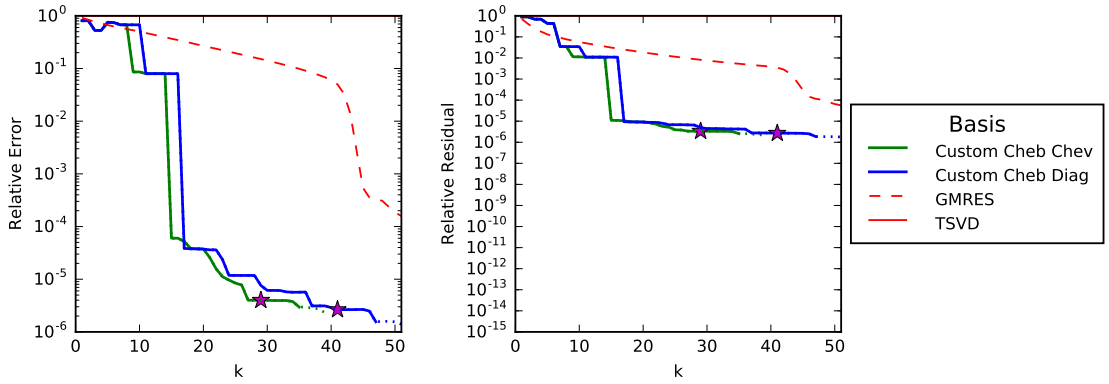


Figure 5.6: Stopping criterion results for `texas2d-ex1` discretized using finite elements with a uniform mesh with 1024 total degrees of freedom.

5.5 SUMMARY

The presented multi-faceted, heuristic stopping criterion combines traditional residual-based stopping criteria with additional components designed to detect a significant change in the behavior of the residual. This stopping criterion requires no knowledge of the magnitude of the noise in the right-hand side (if applicable) and can be performed on the fly,

requiring only information for a few k further. We have shown experimental results with this stopping criterion, illustrating its ability to perform well in detecting a value of k at which to stop in Algorithm 1.2 for many problems and noise levels (when applicable). We have shown its applicability to both 1D and 2D problems and both integral equations and boundary value problems. For all types of problems, it effectively chooses a termination point after which little additional progress would be made towards the true continuous solution, often terminating when the accuracy of the computed solution to the continuous problem is approximately within the discretization error.

Unlike traditional termination criteria for methods such as GMRES, the presented stopping criterion is effective without manual input from the user on the value of specific tolerances, although the ability to provide them is incorporated if the user has additional information and wishes to provide them. Although the presented stopping criterion is heuristic and is not guaranteed to perform well in all scenarios, its overall performance for integral equation problems is comparable to, and in some cases better than, the other regularization parameter choice techniques, and it is applicable to boundary value problems as well.

Chapter 6: Summary

This dissertation presents a method for exploiting compression when solving discretized linear systems. The method exploits compression by solving for an approximate solution to the discretized linear system that is a linear combination of the discrete basis vectors that form the compression basis. To make the basic algorithm more practical, we formulated an incremental version of the algorithm that allows for the compression space to be built-up incrementally, allowing for dynamic determination of the dimension of the compression space.

We have investigated the impact of a variety of issues that arise when choosing a basis, such as the choice and ordering of evaluation points and the choice and ordering of basis vectors. We have illustrated results of numerical tests that confirm the importance of matching the basis evaluation points to the discretization points.

We have also presented Table 2.1, based on approximation theory, to help guide the choice of compression basis for a given problem and discretization basis. We have verified the guidance in the table with experimental results. The table is merely a guide, however, and does not guarantee a given compression basis will perform well. In particular, the table requires an appropriate basis, taken in the correct order.

With the guidance of the table and the motivation of compression, we have investigated a wide variety of bases for use in the compression-based method. While some of the bases we consider have been used for solving continuous problems of similar forms, such as Chebyshev polynomials with spectral methods, we have illustrated that these bases can be effective for solving the linear systems from already discretized problems.

Overall, we have demonstrated the potential of the method to achieve a good approximate solution with relatively few basis vectors for linear systems resulting from the discretization of a variety of 1D and 2D continuous problems. We have shown that the properties of the solution will affect the overall efficacy of the method for a particular problem. To achieve a good approximation with relatively few basis vectors, the basis must be chosen carefully and any details of the discretization of the problem or any known properties of the solution should be utilized when choosing a basis and forming ordered, discrete basis vectors. For ill-conditioned problems, the method can provide a regularizing effect and prevent undesired noise from contaminating the solution.

We have illustrated that the method can often achieve a lower error with smaller k than other methods such as GMRES and TSVD. We have shown how the performance of the method scales as the size of the problem increases, with the number of basis vectors required

to achieve a good approximate solution increasing little, if any, as n increases. In addition to being potentially competitive in cost with standard methods for solving linear systems, for some problems, the compression-based method can achieve better results (i.e., a more accurate solution) than conventional methods.

We have presented a multi-faceted heuristic stopping criterion based on the norm of the residual that can be computed on the fly, requiring results for only a few values of k beyond the k at which it detects to stop. Unlike a simple residual tolerance that is often used as the stopping criterion for many traditional iterative methods for solving linear systems, the presented stopping criterion relies on monitoring both the value and the detailed behavior of the residual norm. The main component of the stopping criterion is designed to detect changes in the behavior of the residual that indicate diminishing returns as k increases. We have performed numerical experiments to test the multi-faceted heuristic stopping criterion, which verify that it usually correctly identifies an appropriate value of k at which to stop, often terminating when the accuracy of the computed solution to the continuous problem is approximately within the discretization error.

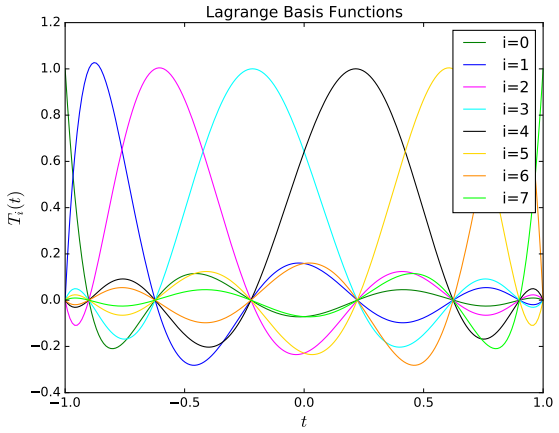
Appendix A: Details of Bases

A.1 POLYNOMIALS

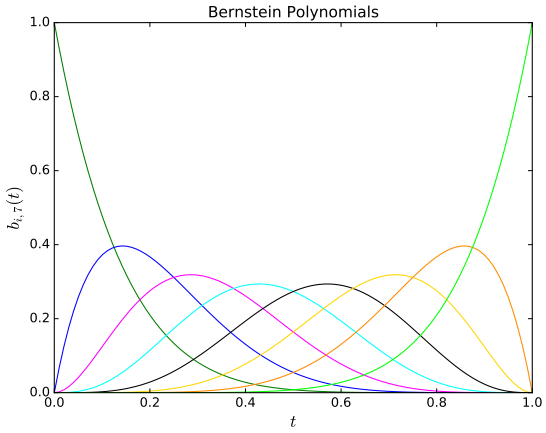
A.1.1 Modified Chebyshev Polynomials

As discussed in Section 3.2, for some discretizations, the discretization basis functions must be homogeneous. Let the i-th Chebyshev polynomial be $T_i(t)$. Although the standard Chebyshev polynomials are not homogeneous, they can easily be modified to be homogeneous. The modified Chebyshev polynomials, $\phi_i(t)$ can be taken as

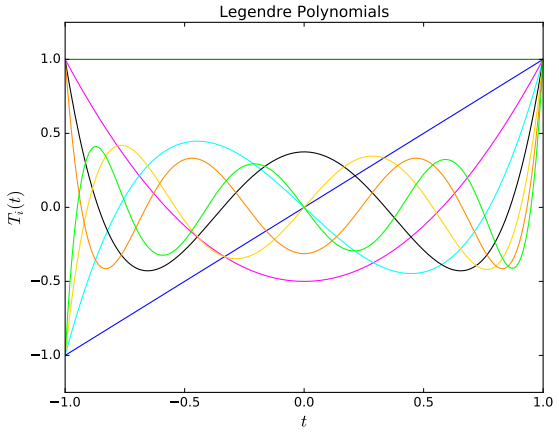
$$\phi_i(t) = \begin{cases} T_i(t) & \text{for } i = 1, 2 \\ T_i(t) - T_{i-2}(t) & \text{for } i > 2 \end{cases}.$$



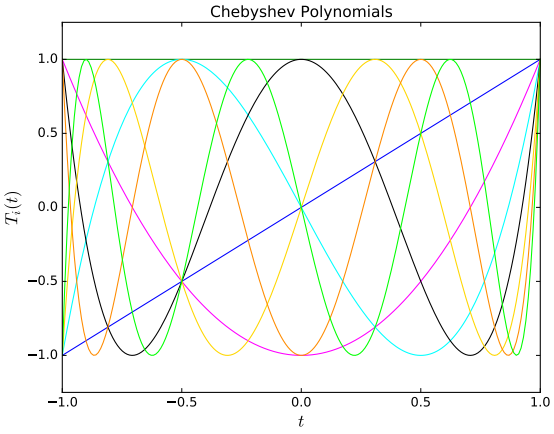
(a) Lagrange basis functions for $n = 8$.



(b) First eight Bernstein polynomials.

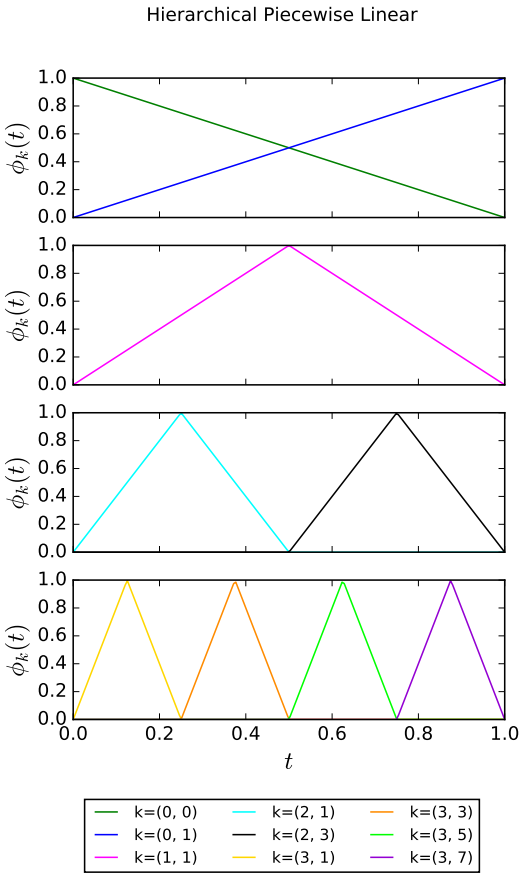


(c) First eight Legendre polynomials.

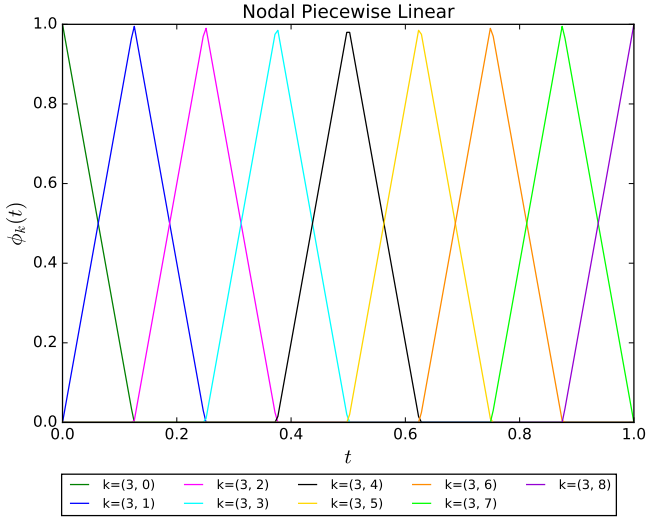


(d) First eight Chebyshev polynomials.

Figure A.1: Illustrations of a few polynomial bases.



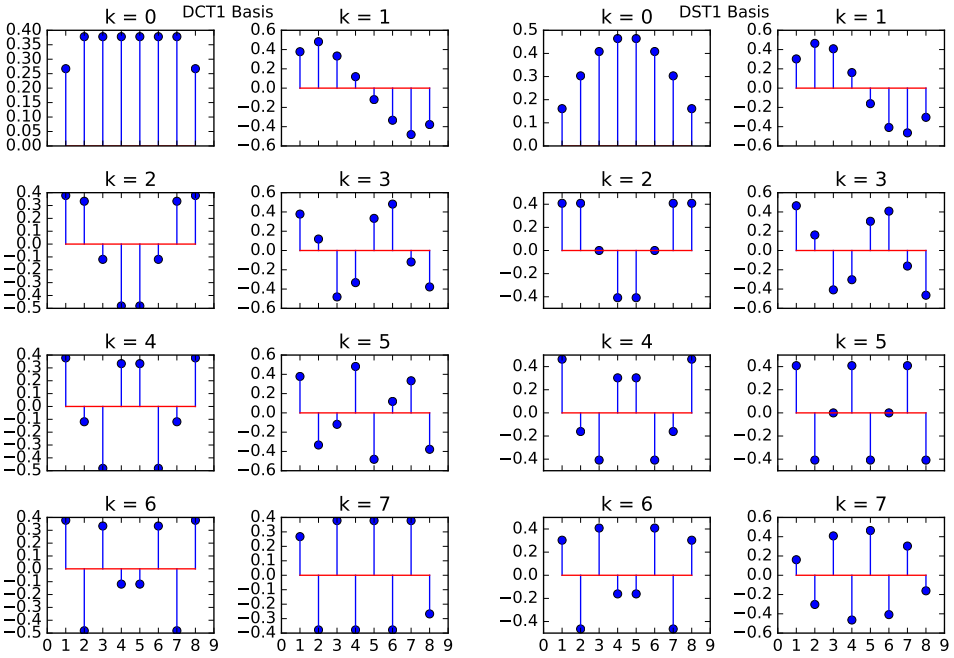
(a) Hierarchical piecewise linear basis functions for $n = 9$.



(b) Nodal piecewise linear basis functions for $n = 9$

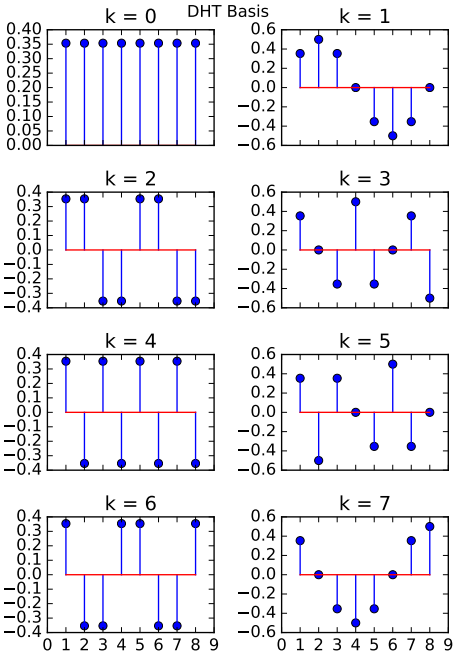
Figure A.2: Illustrations of two piecewise polynomial bases.

A.3 SINUSOIDS



(a) DCT1 basis functions for $n = 8$.

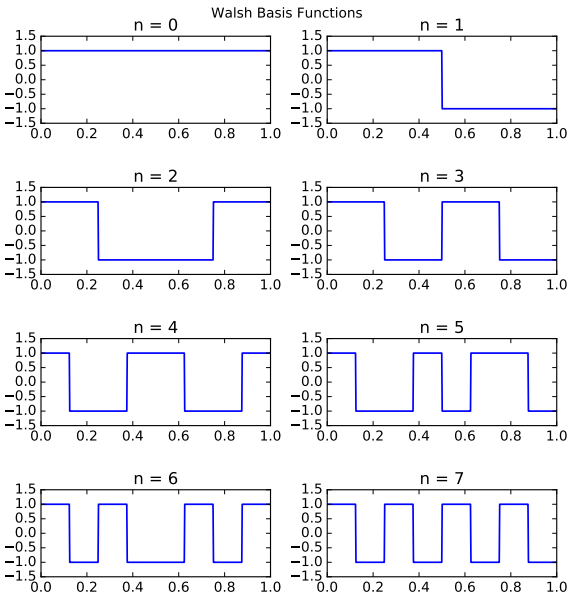
(b) DST1 basis functions for $n = 8$.



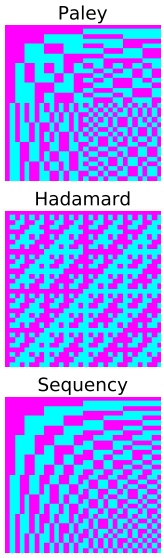
(c) DHT basis functions for $n = 8$.

Figure A.3: Illustrations of a few sinusoidal bases.

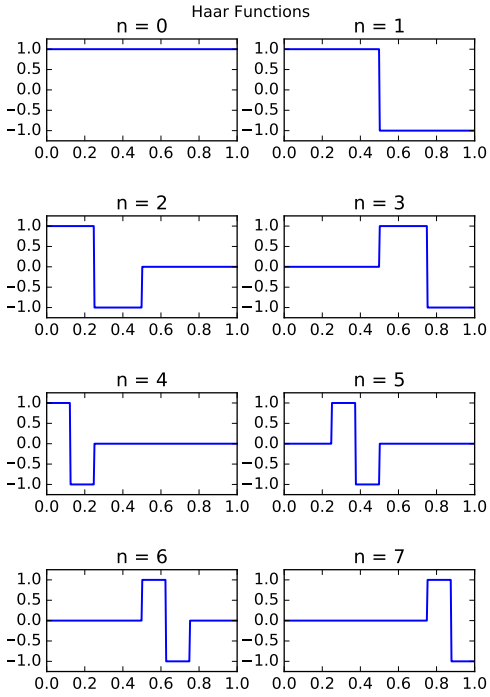
A.4 SQUARE WAVES



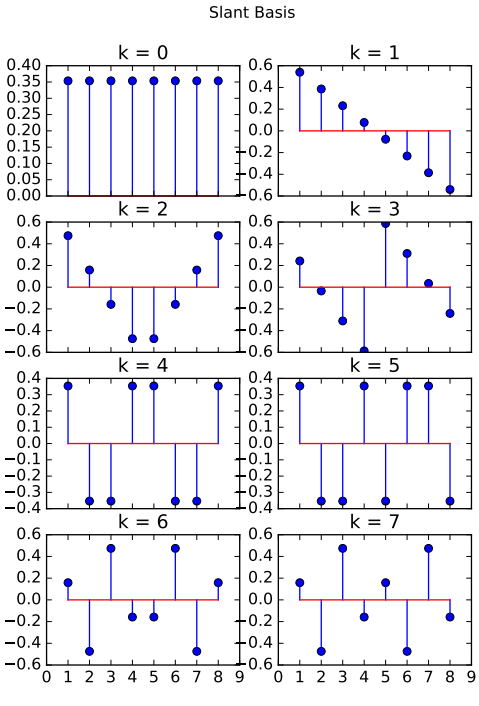
(a) First eight Walsh basis functions.



(b) Various orderings of Walsh basis functions for $n = 32$.

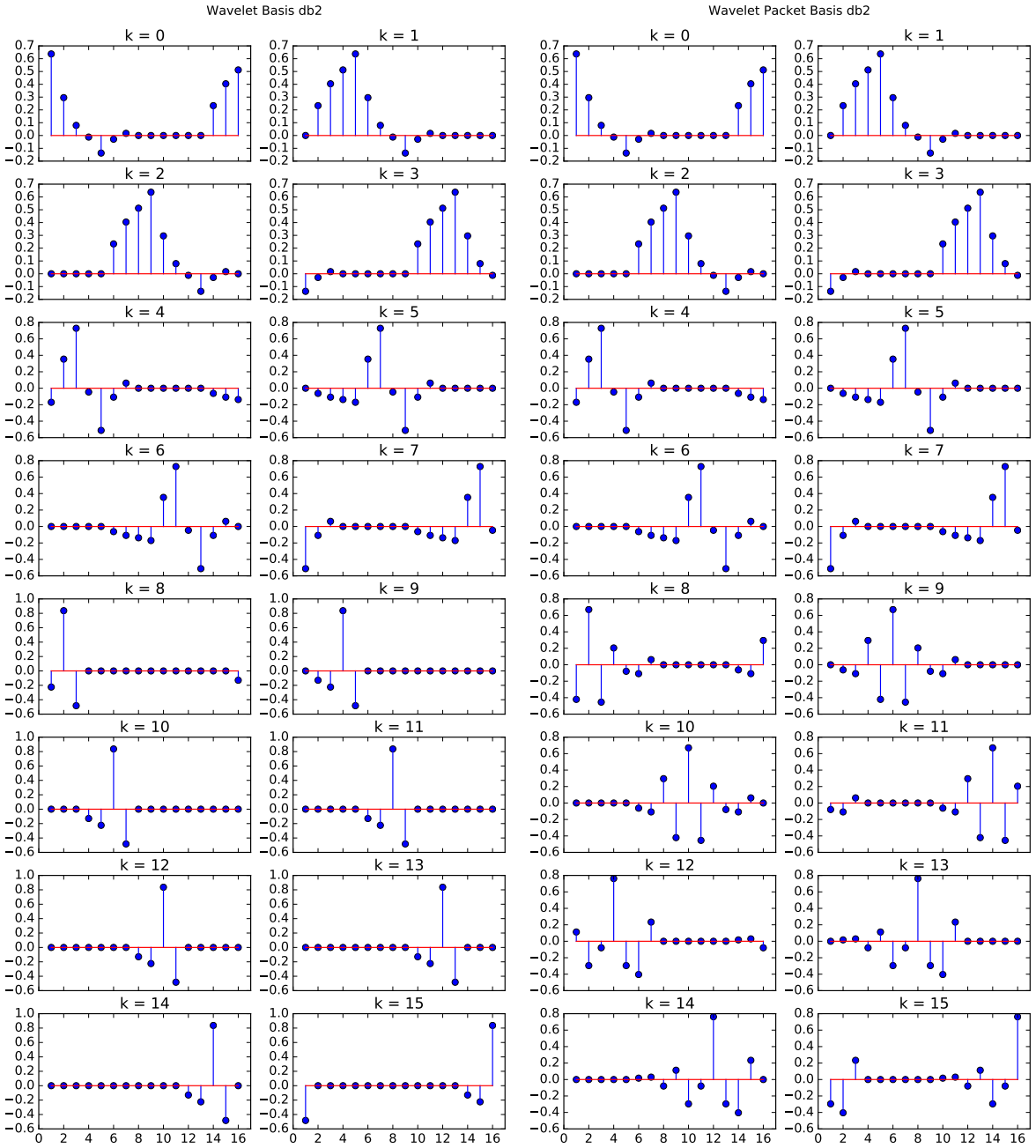


(c) First eight Haar basis functions.



(d) Slant basis functions for $n = 8$.

Figure A.4: Illustrations of various square wave bases.



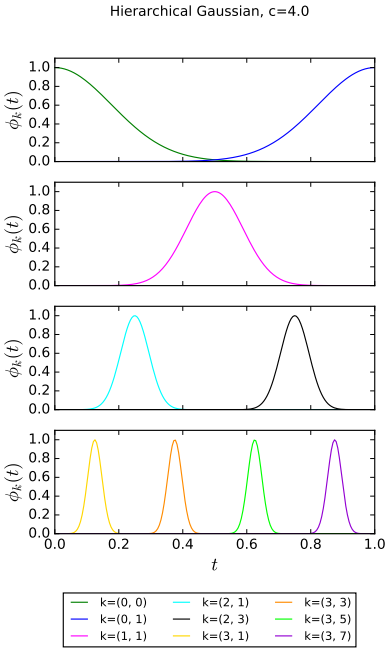
(a) Basis functions from the multilevel wavelet decomposition with Daubechies2 wavelet for $n = 16$.

(b) Basis functions from the full tree wavelet packet decomposition with Daubechies2 wavelet for $n = 16$.

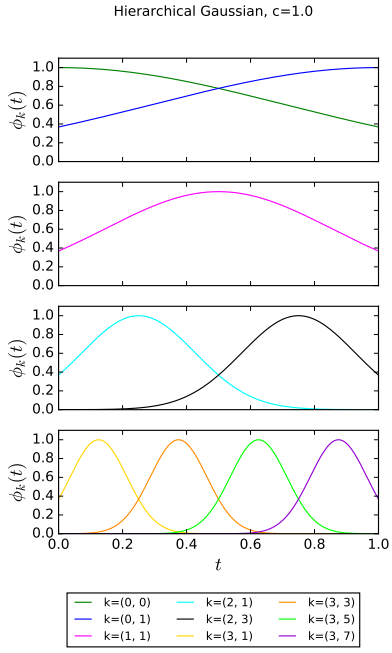
Figure A.5: Illustrations of various wavelet bases.

A.6 GAUSSIANS

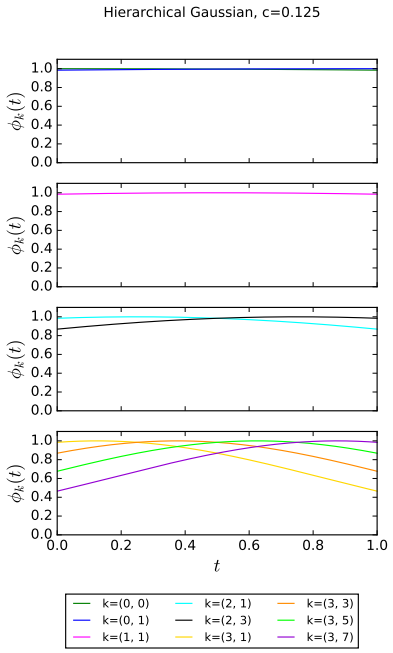
A.6.1 Hierarchical



(a) Hierarchical Gaussian basis functions with essentially no overlap ($c = 4.0$) for $n = 9$



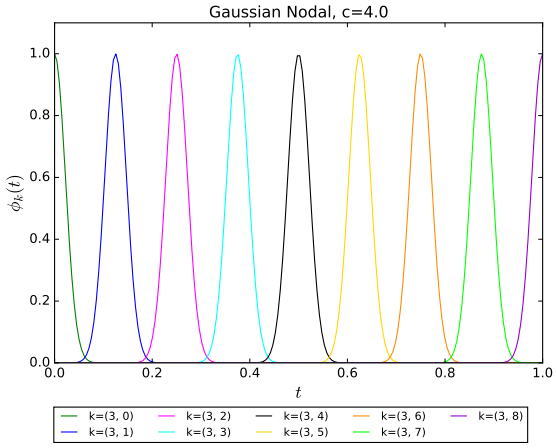
(b) Hierarchical Gaussian basis functions with moderate overlap ($c = 1.0$) for $n = 9$



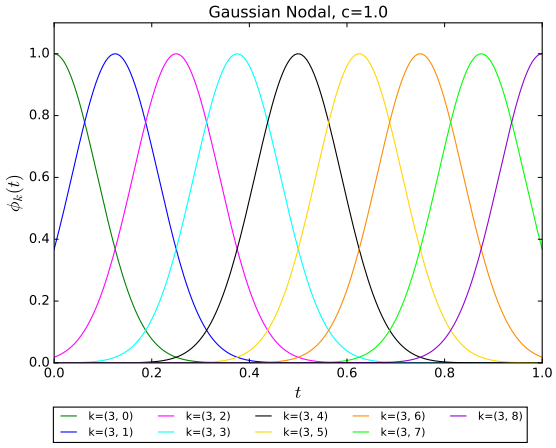
(c) Hierarchical Gaussian basis functions with substantial overlap ($c = 0.125$) for $n = 9$

Figure A.6: Illustrations of hierarchical Gaussian bases with various amounts of overlap.

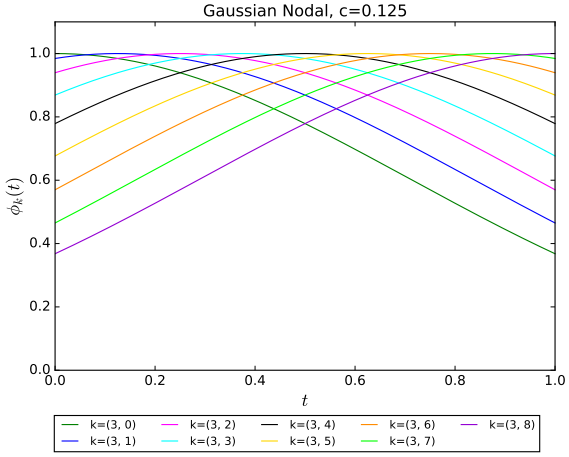
A.6.2 Nodal



(a) Nodal Gaussian basis functions with essentially no overlap ($c = 4.0$) for $n = 9$



(b) Nodal Gaussian basis functions with moderate overlap ($c = 1.0$) for $n = 9$



(c) Nodal Gaussian basis functions with substantial overlap ($c = 0.125$) for $n = 9$

Figure A.7: Illustrations of nodal Gaussian bases with various amounts of overlap.

A.6.3 Overlap Parameter

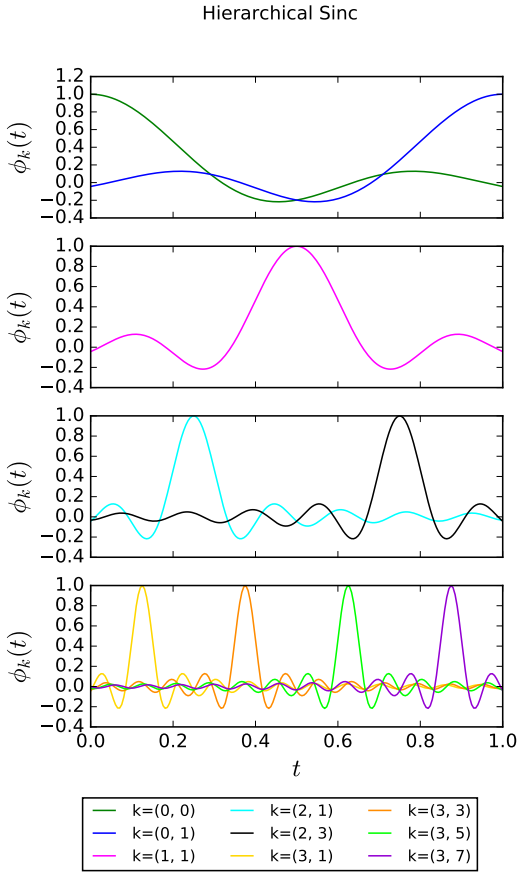
A standard Gaussian function is of the form

$$f(t) = \exp\left(-\frac{(t-b)^2}{2\sigma^2}\right),$$

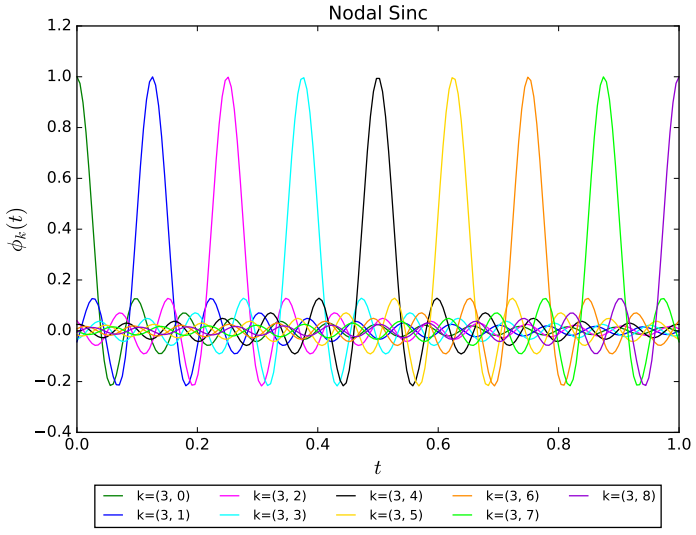
where σ is the standard deviation. The parameter c we use to vary the overlap is inversely proportional to σ. Specifically,

$$\sigma^2 = \frac{1}{2(c(n-1))^2},$$

where n is the number of points (at the given level) and accounts for the different scaling of the width of the Gaussian functions at the various levels in the hierarchical Gaussian basis.



(a) Hierarchical sinc basis functions for $n = 9$.



(b) Nodal sinc basis functions for $n = 9$

Figure A.8: Illustrations of sinc bases.

Appendix B: Details of 1D Integral Equation Test Problems

The 1D integral equation test problems we consider are Fredholm integral equations of the first kind, which take the general form

$$\int_a^b K(s,t)u(t)\,dt = f(s), \tag{B.1}$$

where the kernel $K(s,t) : [c,d] \times [a,b] \to \mathbb{R}$ and right-hand side $f(s) : [c,d] \to \mathbb{R}$ are given, and $u(t) : [a,b] \to \mathbb{R}$ is the solution to be determined. We consider Nystrom (quadrature), collocation, and Galerkin discretizations for integral equation problems, and details for each can be found in Section B.1.

Defining equations for each integral equation test problem are provided in Section B.2. For problems with no $f(s)$ listed, the right-hand side is constructed by multiplying the matrix resulting from the discretization with the true solution. In order to discretize such problems using collocation or Galerkin, we first interpolate the true solution to obtain "true" coefficients and then use these coefficients to compute a right-hand-side vector. For each problem, the numerical rank (computed by NumPy using the default tolerance) and condition number (if full rank) are provided for a few selected discretization and basis combinations.

In practice, integral equation problems typically have noise in the right-hand side as the right-hand side often comes from physically measured data, where the measurement tools are subject to noise. To simulate this, it is common to add artificial noise to the discretized integral equation problems. For some tests, we add such artificial noise. Specifically, when we add artificial noise, Gaussian random noise with mean 0 and standard deviation η is added to the right-hand-side vector and we classify the noise level by η.

B.1 DISCRETIZATIONS

B.1.1 Nystrom (Quadrature)

The Nystrom (quadrature) method is a standard technique for solving integral equations numerically by using a quadrature rule to approximate the integral by a finite sum. Let the nodes and weights of the quadrature rule be denoted by t_j and w_j, respectively, $j = 1, \ldots, n$. Additionally, choose n points s_i (which can be different, but are often the same as t_j). Then,

the approximation to Equation B.1 becomes

$$\sum_{j=1}^{n} w_j K(s_i, t_j)\, u(t_j) = f(s_i), \quad i = 1, \ldots, n,$$

which is a system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $a_{ij} = w_j K(s_i, t_j)$, $b_i = f(s_i)$, and $x_j \approx u(t_j)$. This linear system can be solved for the vector $\boldsymbol{x}$, whose components are discrete samples of the approximate values of the solution function u.

B.1.2 Collocation

To discretize an integral equation by collocation, the solution u is approximated by a linear combination of basis functions ϕ_j with coefficients x_j, giving

$$u(t) \approx v(t, \boldsymbol{x}) \equiv \sum_{j=1}^{n} x_j \phi_j(t).$$

We then choose collocation points, s_i, $i = 1, \ldots, n$. Substituting the approximate solution into Equation B.1 and forcing it to satisfy the integral equation exactly at the collocation points gives a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where

$$a_{ij} = \int_a^b K(s_i, t)\phi_j(t)\, dt, \quad b_i = f(s_i), \quad i = 1, \ldots, n,$$

for which the solution $\boldsymbol{x}$ is a vector of coefficients of the basis functions. Depending on the complexity of the kernel and the basis functions, the integrals defining the entries of $\boldsymbol{A}$ can be computed analytically or approximated using numerical quadrature.

B.1.3 Galerkin

To discretize an integral equation by the Galerkin method, the solution u is approximated by a linear combination of basis functions ϕ_j with coefficients x_j, giving

$$u(t) \approx v(t, \boldsymbol{x}) \equiv \sum_{j=1}^{n} x_j \phi_j(t).$$

We then substitute the approximate solution into Equation B.1 and require the residual to be orthogonal to the subspace spanned by ϕ_i, $i = 1, \ldots, n$. Doing so gives a linear system

$\boldsymbol{Ax} = \boldsymbol{b}$, where

$$a_{ij} = \int_a^b \int_a^b K(s,t)\phi_i(s)\phi_j(t)\, ds\, dt, \quad b_i = \int_a^b f(s)\phi_i(s)\, ds, \quad i = 1, \ldots, n.$$

The necessary integrals can be computed analytically or approximated using numerical quadrature.

`designed_sine_ie`

This problem is specifically designed by us. The true solution closely resembles the true solution to `designed_poly`, the other designed problem.

$$
\begin{aligned}
K(s,t) &= \begin{cases} 1+s-t & s < t \\ 1+t-s & s \geq t \end{cases} \\
f(s) &= \frac{1 + 2s\cos(\pi s) + (2(\sin(\pi s) - \pi s\cos(\pi s)))/\pi}{\pi} \\
u(t) &= \sin(\pi t) \\
t &\in [0,1], \qquad s \in [0,1]
\end{aligned}
$$

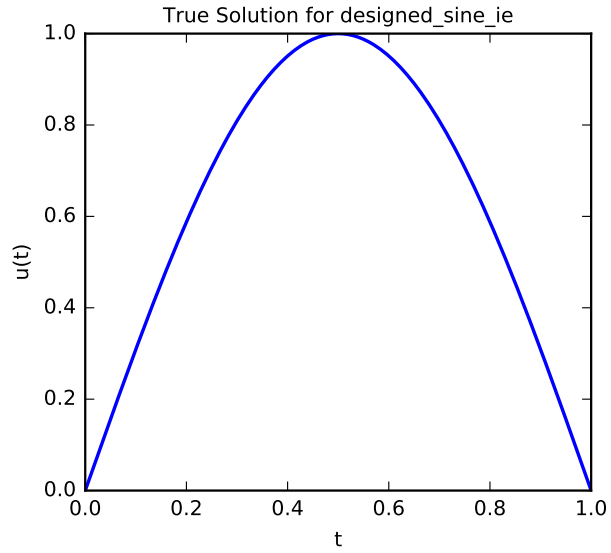


Figure B.1: Illustration of true solution for `designed_sine_ie`.

Discretization	N	Numerical Rank	Condition #
Nystrom (composite trapezoid)	128	128	2.2×10^4
Galerkin (modified Chebyshev)	128	120	–
Galerkin (nodal PL)	129	129	1.6×10^5

Table B.1: Condition numbers of discretized `designed_sine_ie` for a few selected discretizations.

designed_poly_ie

This problem is specifically designed by us. The true solution is a low degree polynomial that closely resembles $\sin(\pi t)$, the solution to the other designed problem, `designed_sine`. Using Hermite interpolation, let $p(t)$ be the polynomial of degree four whose coefficients are given by the solution to the linear system

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1/2 & 1/4 & 1/8 & 1/16 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \pi \\ -\pi \end{bmatrix}.$$

The integral equation is then given by

$$\begin{aligned} K(s,t) &= \begin{cases} 1+s-t & s < t \\ 1+t-s & s \geq t \end{cases} \\ f(s) &= c_1(1+2s-2s^2)/2 + c_2(1+3s-2s^3)/6 + c_3(1+4s-2s^4)/12 \\ &\quad + c_4(1+5s-2s^5)/20 + c_5(1+6s-2s^6)/30 \\ u(t) &= p(t), \qquad t \in [0,1], \qquad s \in [0,1] \end{aligned}$$

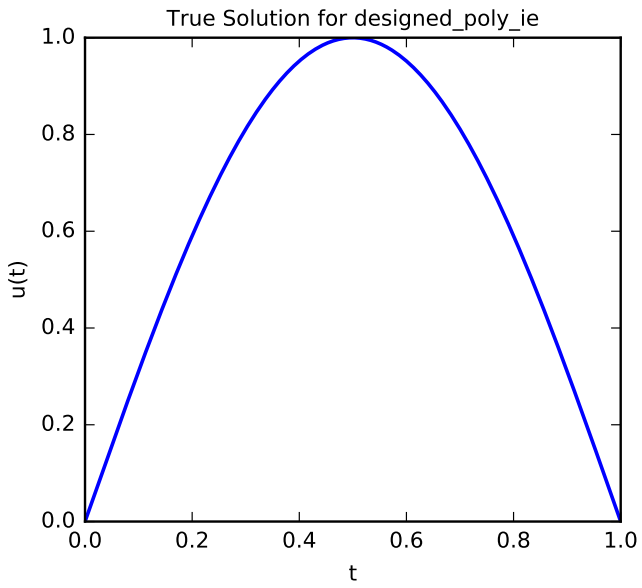


Figure B.2: Illustration of true solution for `designed_poly_ie`.

Discretization	N	Numerical Rank	Condition #
Nystrom (composite trapezoid)	128	128	2.2×10^4
Galerkin (modified Chebyshev)	128	120	–
Galerkin (nodal PL)	129	129	1.6×10^5

Table B.2: Condition numbers of discretized `designed_poly_ie` for a few selected discretizations.

`deriv2_c1`

This problem is from [58]; it was originally presented in [59, p. 315].

$$
\begin{aligned}
K(s,t) &= \begin{cases} s\,(t-1) & s < t \\ t\,(s-1) & s \geq t \end{cases} \\
f(s) &= \frac{s^3 - s}{6} \\
u(t) &= t \\
t &\in [0,1], \qquad s \in [0,1]
\end{aligned}
$$

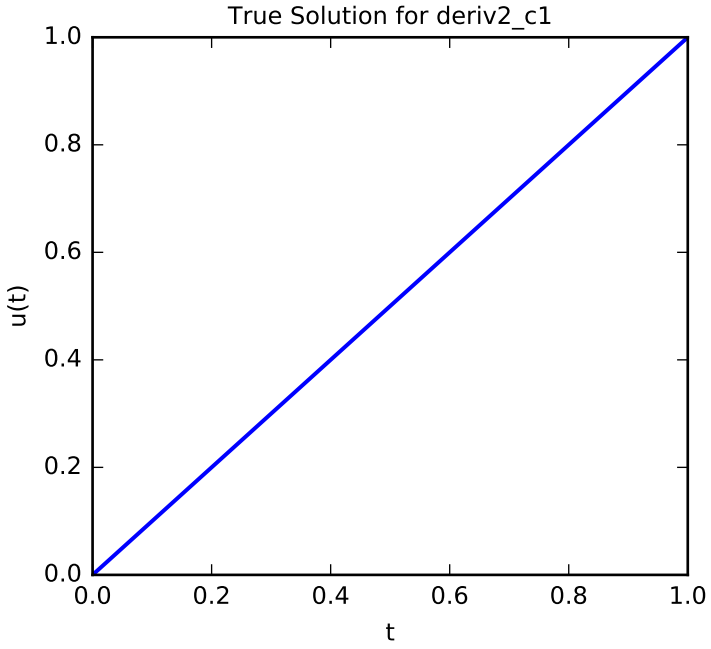


Figure B.3: Illustration of true solution for `deriv2_c1`.

Discretization	N	Numerical Rank	Condition #
Nystrom (composite trapezoid)	128	126	–
Galerkin (modified Chebyshev)	128	118	–
Galerkin (nodal PL)	129	129	1.7×10^5

Table B.3: Condition numbers of discretized `deriv2_c1` for a few selected discretizations.

`gravity_c1`

This problem is from [58]; it was originally presented in [60].

$$
\begin{aligned}
d &= 0.25 \\
K(s,t) &= d\left(d^2 + (s-t)^2\right)^{-1.5} \\
u(t) &= \sin(\pi t) + 0.5\sin(2\pi t) \\
t &\in [0,1], \qquad s \in [0,1]
\end{aligned}
$$

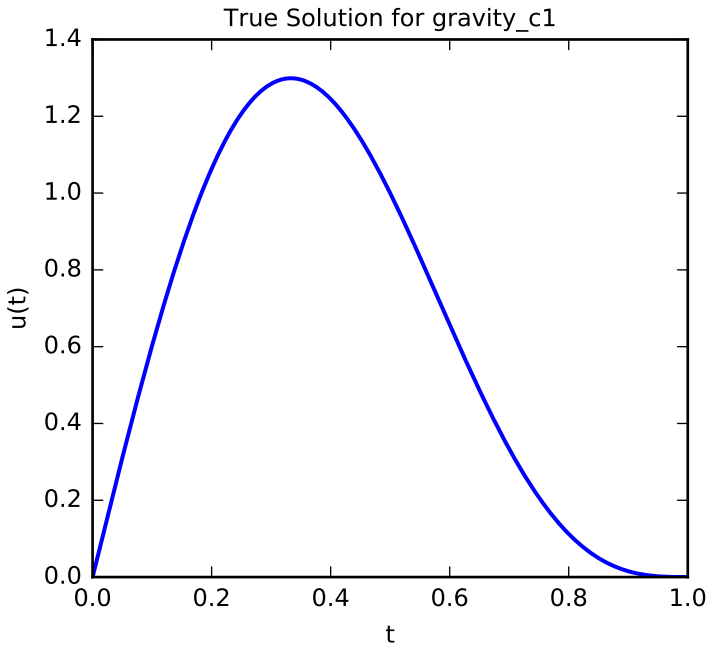


Figure B.4: Illustration of true solution for `gravity_c1`.

Discretization	N	Numerical Rank	Condition #
Nystrom (composite trapezoid)	128	47	–
Galerkin (modified Chebyshev)	128	45	–
Galerkin (nodal PL)	129	47	–

Table B.4: Condition numbers of discretized `gravity_c1` for a few selected discretizations.

shaw

This problem is from [58]; it was originally presented in [61].

$$K(s,t) = (\cos(s) + \cos(t))^2 \left(\frac{\sin(v)}{v} \right)^2$$

$$v = \pi \left(\sin(s) + \sin(t) \right)$$

$$u(t) = a_1 e^{-c_1 (t-t_1)^2} + a_2 e^{-c_2 (t-t_2)^2}$$

$$a_1 = 2, \qquad a_2 = 1, \qquad c_1 = 6, \qquad c_2 = 2$$

$$t_1 = .8, \qquad t_2 = -.5$$

$$t \in [-\pi/2, \pi/2], \qquad s \in [-\pi/2, \pi/2]$$

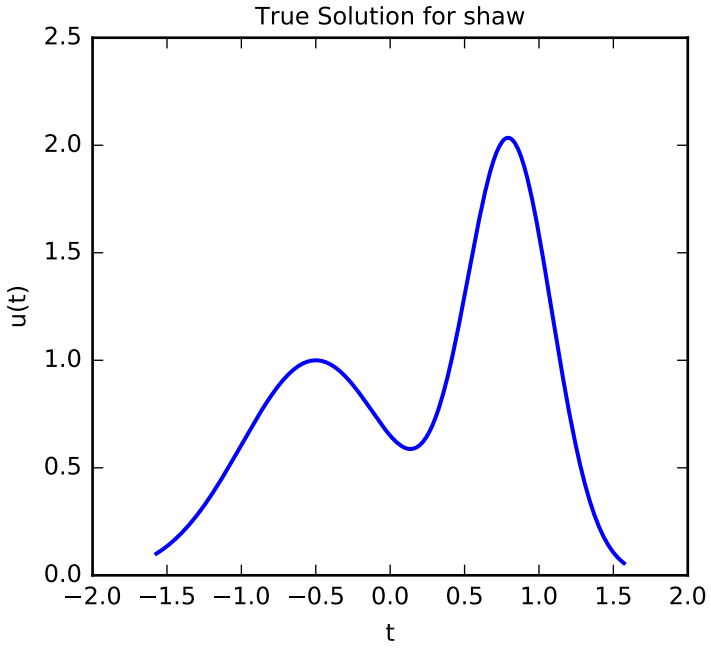


Figure B.5: Illustration of true solution for **shaw**.

Discretization	N	Numerical Rank	Condition #
Nystrom (composite trapezoid)	128	20	–
Galerkin (modified Chebyshev)	128	10	–
Galerkin (nodal PL)	129	10	–

Table B.5: Condition numbers of discretized **shaw** for a few selected discretizations.

Appendix C: Details of 1D Boundary Value Test Problems

The boundary value problems considered are second-order, linear, constant coefficient boundary value problems (BVPs) in one dimension. Such problems take the general form

$$pu''(t) + qu'(t) + ru(t) = f(t), \tag{C.1}$$

where p, q, and r are constants and $t \in [a, b]$, with additional equations to specify boundary conditions.

We consider finite difference, collocation, and Galerkin discretizations, and an overview of each is provided in Section C.1. To ensure consistent treatment of boundary conditions, they are always explicitly enforced through equations in the linear system (even if the equations are trivial). For example, when enforcing Dirichlet boundary conditions in a finite difference discretization, we explicitly include the trivial equation $x_0 = u(a)$ as its own row in the matrix instead of modifying the remaining matrix and incorporating the effect of the boundary condition into the right-hand-side vector. Similarly, when using collocation or Galerkin, we explicitly introduce an equation to enforce the boundary conditions, such as $\sum_i \phi_i(t_0) = u(a)$ for a Dirichlet condition at the left endpoint, and include it directly in the linear system, even if it is trivial.

Defining equations for each BVP test problem are provided in Section C.2. For each problem, the numerical rank (computed by NumPy using the default tolerance) and condition number (if full rank) are provided for a few selected discretization and basis combinations.

C.1 DISCRETIZATIONS

C.1.1 Finite Differences

With the finite difference method, n evenly spaced mesh points, t_i, are introduced in $[a, b]$ with mesh spacing $h = (b - a)/(n - 1)$. The derivatives in the differential equation are then replaced by finite difference approximations at each interior mesh point. For instance, substituting second-order accurate centered difference approximations for the derivatives in Equation C.1 gives

$$p\left(\frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}\right) + q\left(\frac{x_{i+1} - x_{i-1}}{h}\right) + rx_i = f(t_i), \quad i = 2, \ldots, n-1,$$

which is a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ whose solution gives approximate solution values at the mesh points, $x_i \approx u(t_i)$. Note that additional equations are needed to enforce the boundary conditions, and their form depends on the types of the individual boundary conditions. For Neumann boundary conditions, we specifically use a second-order centered difference approximation at the endpoint.

C.1.2 Collocation

To discretize a BVP by collocation, the solution u is approximated by a linear combination of basis functions ϕ_j defined on $[a, b]$ with coefficients x_j, giving

$$u(t) \approx v(t, \boldsymbol{x}) \equiv \sum_{j=1}^{n} x_j \phi_j(t).$$

We then choose collocation points, t_i in $[a, b]$, $i = 1, \ldots, n$. We then substitute the approximate solution into Equation C.1 and force it to satisfy the ODE at the interior collocation points and the boundary conditions at the endpoints. Doing so gives

$$\sum_{j=1}^{n} x_j \left[p\phi_j''(t_i) + q\phi_j'(t_i) + r\phi_j(t_i) \right] = f(t_i), \quad i = 2, \ldots, n-1$$

as the linear system for the interior points, with additional equations used to satisfy the boundary conditions.

C.1.3 Galerkin

To discretize a BVP by the Galerkin method, the solution u is approximated by a linear combination of basis functions ϕ_j defined on $[a, b]$ with coefficients x_j, giving

$$u(t) \approx v(t, \boldsymbol{x}) \equiv \sum_{j=1}^{n} x_j \phi_j(t).$$

We then substitute the approximate solution into Equation C.1 and require the residual to be orthogonal to the subspace spanned by ϕ_i, $i = 1 \ldots, n$. Doing so gives a linear system

$\boldsymbol{Ax} = \boldsymbol{b}$, where

$$a_{ij} = \int_a^b -p\phi_j'(t)\phi_i'(t) + q\phi_j'(t)\phi_i(t) + r\phi_j(t)\phi_i(t)\,dt,$$
$$b_i = \int_a^b f(t)\phi_i(t)\,dt.$$

These integrals can be computed analytically or approximated using numerical quadrature, depending on the complexity of the forcing function and the basis functions. Additional details regarding Galerkin discretizations of boundary value problems can be found in many numerical analysis textbooks, such as [2, p. 438].

`designed_sine_bvp`

This problem is specifically designed by us. The true solution closely resembles the true solution to `designed_poly`, the other designed problem.

$$u'' = -\pi^2 \sin(\pi t)$$
$$u(0) = u(1) = 0$$

The true solution is

$$u(t) = \sin(\pi t).$$

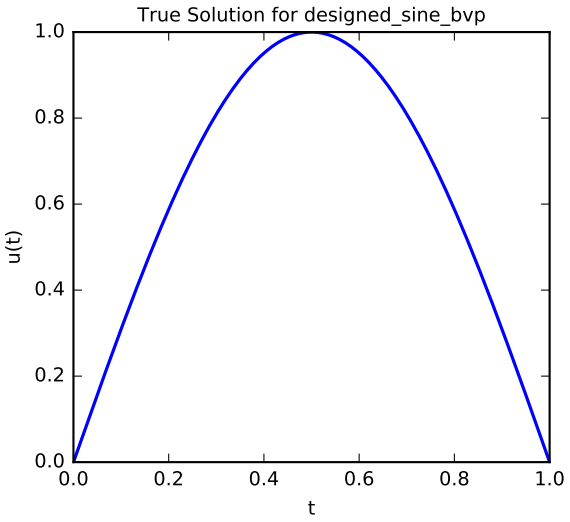


Figure C.1: Illustration of true solution for `designed_sine_bvp`.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	6.5×10^3
Galerkin (Chebyshev)	128	128	1.2×10^5
Galerkin (nodal PL)	129	129	7.6×10^3

Table C.1: Condition numbers of discretized `designed_sine_bvp` for a few selected discretizations.

`designed_poly_bvp`

This problem is specifically designed by us. The true solution is a low degree polynomial that closely resembles $\sin(\pi t)$, the solution to the other designed problem `designed_sine`. Using Hermite interpolation, let $p(t)$ be the polynomial of degree four whose coefficients are given by the solution to the linear system

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1/2 & 1/4 & 1/8 & 1/16 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \pi \\ -\pi \end{bmatrix}.$$

Then, the boundary value problem is

$$\begin{aligned} u'' &= 2c_3 + 6c_4 t + 12c_5 t^2 \\ u(0) &= u(1) = 0 \end{aligned}$$

with true solution $u(t) = p(t)$.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	6.5×10^3
Galerkin (Chebyshev)	128	128	1.2×10^5
Galerkin (nodal PL)	129	129	7.6×10^3

Table C.2: Condition numbers of discretized `designed_poly_bvp` for a few selected discretizations.

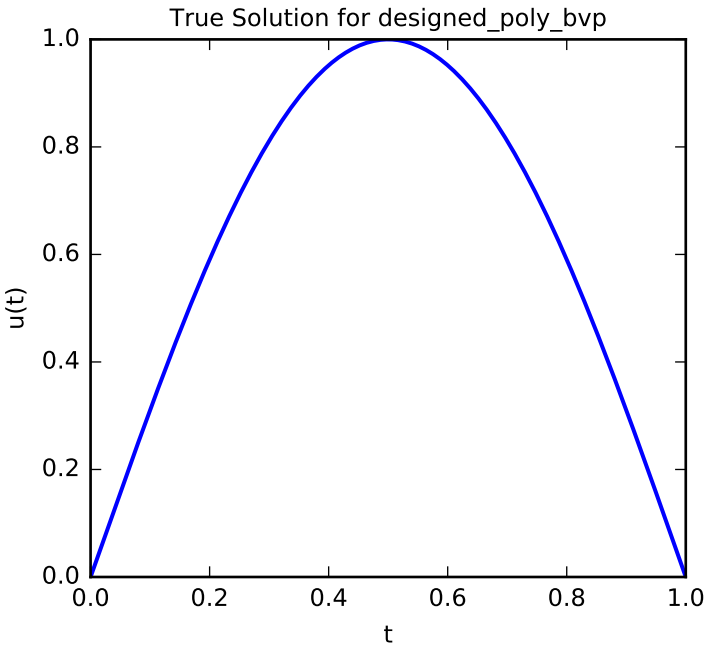


Figure C.2: Illustration of true solution for `designed_poly_bvp`.

`greengard-ex1`

This problem is from [62]; it was originally presented in [63].

$$-u'' + 400u = -400\cos^2(\pi t) - 2\pi^2\cos(2\pi t)$$
$$u(0) = u(1) = 0$$

The true solution is

$$u(t) = \frac{e^{-20}}{1+e^{-20}}e^{20t} + \frac{1}{1+e^{-20}}e^{-20t} - \cos^2(\pi t).$$

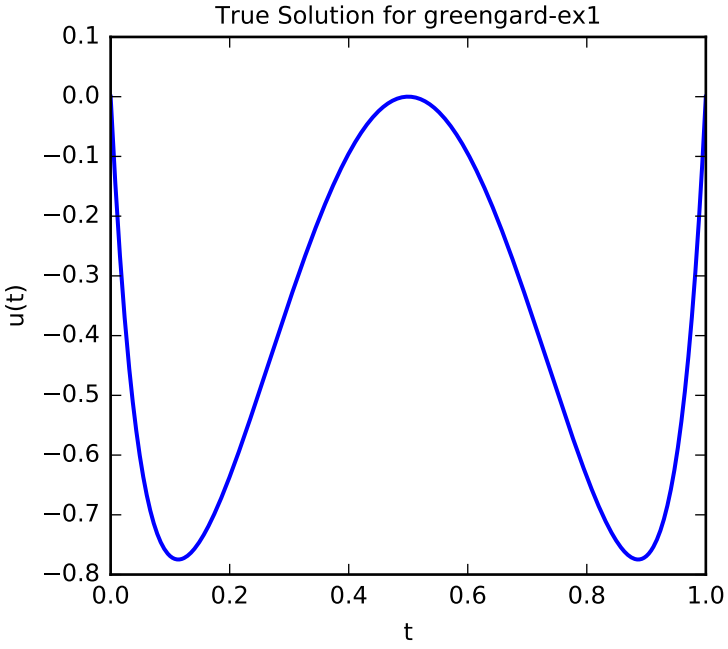


Figure C.3: Illustration of true solution for `greengard-ex1`.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	1.6×10^2
Galerkin (Chebyshev)	128	128	1.6×10^5
Galerkin (nodal PL)	129	129	9.9×10^2

Table C.3: Condition numbers of discretized `greengard-ex1` for a few selected discretizations.

`greengard-ex2`

This is a boundary layer problem from [62]. This is a slightly modified version of the problem; the original specified $\epsilon = 10^{-5}$.

$$\begin{aligned} \epsilon u'' - u &= 0, & \epsilon &= 10^{-4} \\ u(-1) &= 1, & u(1) &= 2 \end{aligned}$$

The true solution is

$$u(t) = \frac{e^{-100(t-1)}(-e^{200t} + 2e^{200(t+1)} - 2 + e^{200})}{e^{400} - 1}$$

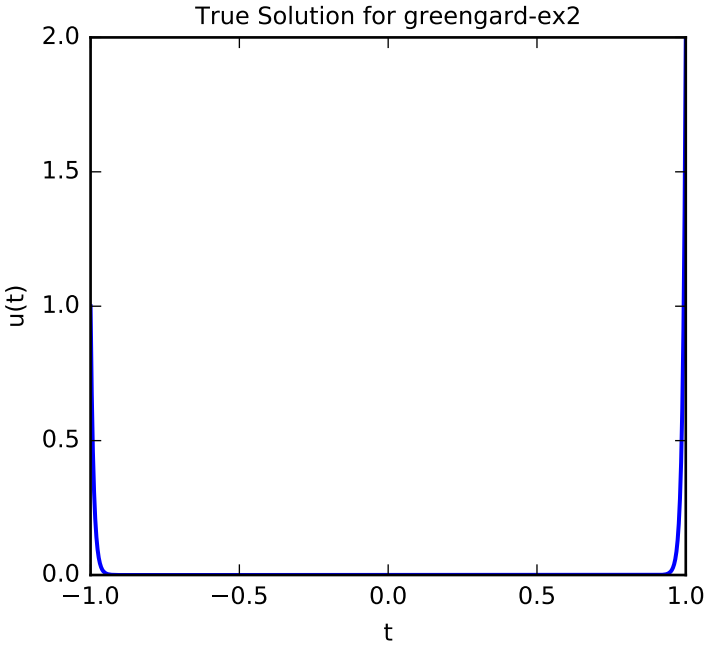


Figure C.4: Illustration of true solution for `greengard-ex2`.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	1.6×10^4
Galerkin (Chebyshev)	128	128	6.2×10^2
Galerkin (nodal PL)	129	129	1.2×10^2

Table C.4: Condition numbers of discretized `greengard-ex2` for a few selected discretizations.

greengard-ex3

This problem is from [62]. This problem has a highly oscillatory solution.

$$u'' + 5u' + 10000u = -500\cos(100t)e^{-5t}$$
$$u(0) = 0, \qquad u(1) = \sin(100)e^{-5}$$

The true solution is

$$u(t) = \sin(100t)e^{-5t}.$$

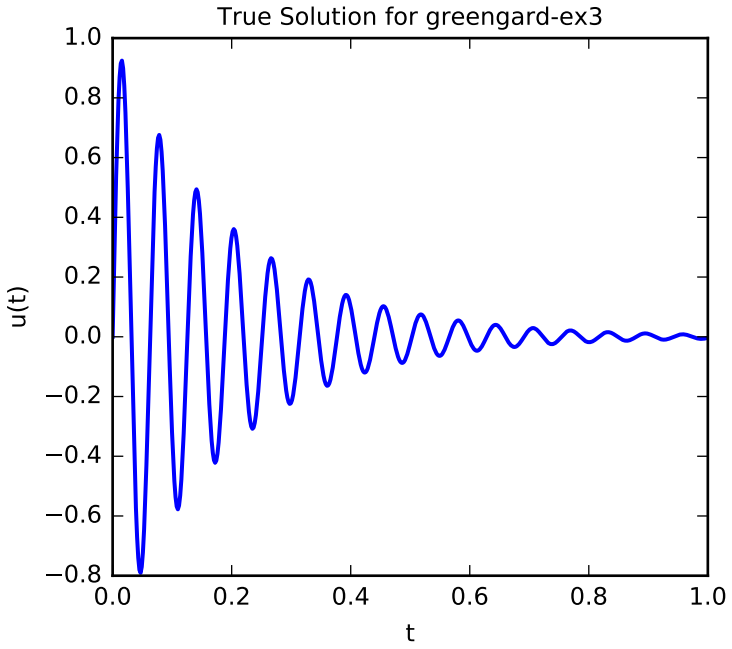


Figure C.5: Illustration of true solution for `greengard-ex3`.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	9.1×10^2
Galerkin (Chebyshev)	128	128	2.7×10^6
Galerkin (nodal PL)	129	129	8.5×10^4

Table C.5: Condition numbers of discretized `greengard-ex3` for a few selected discretizations.

`fornberg`

This problem is from [64].

$$u'' + u' - 2u = -2$$
$$u(-1) = u(1) = 0$$

The true solution is

$$u(t) = 1 - \frac{\sinh(2)e^t + \sinh(1)e^{-2t}}{\sinh(3)}.$$

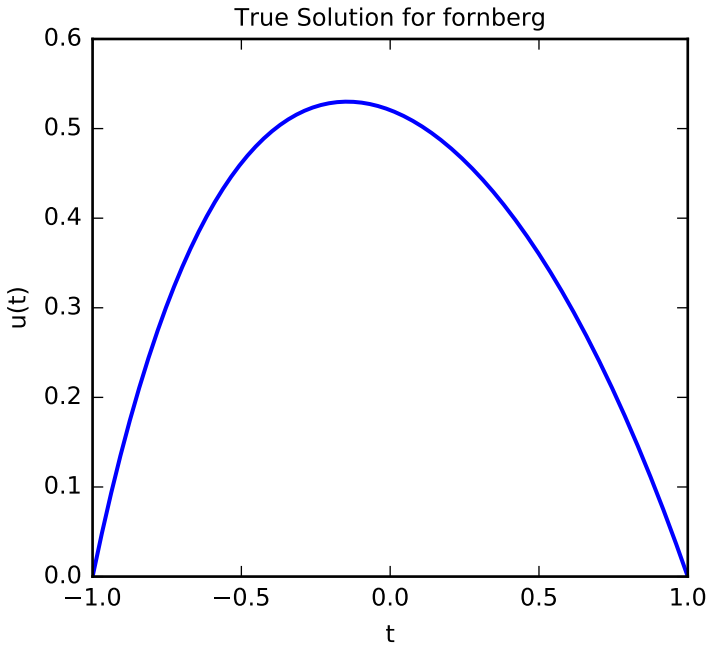


Figure C.6: Illustration of true solution for `fornberg`.

Discretization	N	Numerical Rank	Condition #
Finite differences	128	128	3.7×10^3
Galerkin (Chebyshev)	128	128	3.3×10^4
Galerkin (nodal PL)	129	129	3.6×10^3

Table C.6: Condition numbers of discretized `fornberg` for a few selected discretizations.

Appendix D: Details of 2D Test Problems

These test problems come from a variety of different problem domains. As many test problems are taken directly from other packages in their discretized form, most test problems do not contain details of the continuous problem, but instead contain information on the source of the problem and how to generate it.

For problems that specify they are discretized using finite differences or Nystrom, conceptually the solution is defined on an $n \times n$ grid. To form the linear system, however, the grid is flattened to a 1D solution vector in row-major order. For all other problems, the ordering of the solution vector is determined by the source package (for problems named `ir-*`) or by the discretization software (for problems discretized using FEniCS [36, 37]).

For each problem, the numerical rank (computed by NumPy using the default tolerance) and condition number (if full rank) are provided.

`2d-designed-sine-bvp`

This problem was designed by us to be the 2D analogue of `designed-sine-bvp`. It is a general 2D boundary value problem of the form

$$-\Delta u = -u_{t_0 t_0} - u_{t_1 t_1} = f(\boldsymbol{t}),$$

with sufficient boundary conditions, where $u(\boldsymbol{t})$ is the unknown function to be determined. The specific functions defining this problem are given below.

$$f(\boldsymbol{t}) = 2\pi^2 \sin(\pi t_0) \sin(\pi t_1)$$
$$u(t_0, 0) = u(t_0, 1) = u(0, t_1) = u(1, t_1) = 0$$

The true solution is

$$u(\boldsymbol{t}) = \sin(\pi t_0) \sin(\pi t_1).$$

We discretize this problem using finite differences on a uniform 2D mesh. With $n = 32$ ($n^2 \times n^2$ linear system), the resulting matrix is full rank, and its condition number is 3.9×10^2.

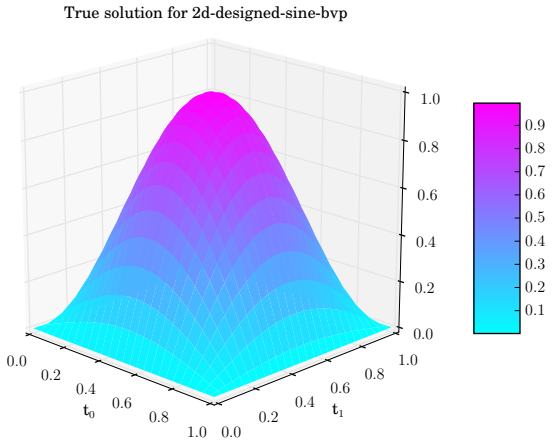


Figure D.1: Illustration of true solution for `2d-designed-sine-bvp`.

su-ex3

This is a 2D integral equation problem that is example three presented in [65, p. 195]. It is a 2D Fredholm integral equation of the first kind that takes the general form

$$\int_{a_1}^{b_1} \int_{a_0}^{b_0} K(s_0, s_1, t_0, t_1) u(t_0, t_1)\, dt_0\, dt_1 = f(s_0, s_1) \tag{D.1}$$

where

$$K(s_0, s_1, t_0, t_1) = \frac{1}{\sqrt{(t_0 - s_0)^2 + (t_1 - s_1)^2}}$$

and the true solution is given by

$$u(t_0, t_1) = 1 + (t_0 - 0.5)t_0(t_1 + 0.5)\cos(\frac{\pi}{3}t_1) - t_1(1 - t_1).$$

We discretize this problem using composite trapezoid quadrature using midpoints for the collocation points. As $f(s_0, s_1)$ is not provided for this problem, the right-hand-side vector is generated by computing $\boldsymbol{Ax}$, where $\boldsymbol{x}$ is obtained by sampling the true solution function at the discretization points flattened in row-major order. With $n = 32$ ($n^2 \times n^2$ linear system), the resulting matrix is full rank, and its condition number is 1.9×10^3.

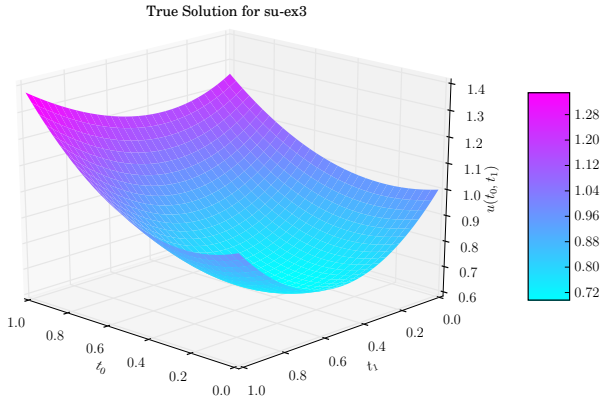


Figure D.2: Illustration of true solution for `su-ex3`.

`texas2d-ex1`

This problem is example one in [66] and is a general, continuous boundary value problem of the form

$$-\Delta u = -u_{t_0 t_0} - u_{t_1 t_1} = f(\boldsymbol{t}),$$

with sufficient boundary conditions, where $u(\boldsymbol{t})$ is the unknown function to be determined. The specific functions defining this problem are

$$f(\boldsymbol{t}) = -1(6t_0 t_1(1-t_1) - 2t_0^3)$$
$$u(\boldsymbol{t}) = u_D(\boldsymbol{t}), \qquad \boldsymbol{t} \text{ on } \mathrm{d}\Omega$$

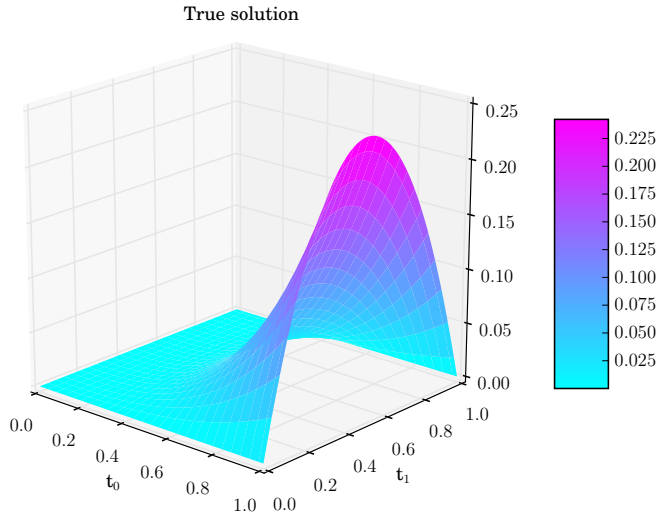
where $\mathrm{d}\Omega$ is the boundary of the domain, $\Omega = [0,1] \times [0,1]$, and u_D is the true solution function on $\mathrm{d}\Omega$. The true solution is

$$u(\boldsymbol{t}) = t_1(1-t_1)t_0^3.$$

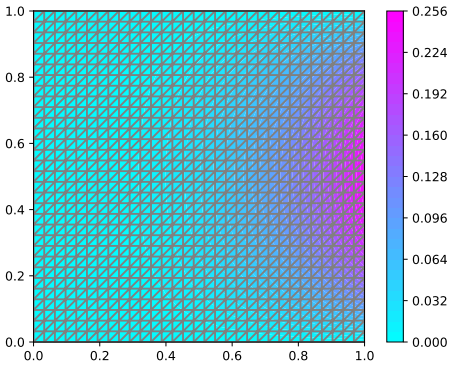
We consider this problem discretized using finite differences on a uniform 2D mesh. We also consider this problem discretized using finite elements with P1 finite elements on both a uniform and irregular mesh over the unit square. The rank and condition number of the resulting matrices are shown in Table D.1, and the true solution function and the solution of the finite element discretization of the problem with the corresponding meshes are illustrated in Figure D.3.

Discretization	DOF	Numerical Rank	Condition #
Finite differences	1024 (32×32 grid)	1024	3.9×10^2
Finite elements (uniform mesh)	1024 (32×32 grid)	1024	3.9×10^2
Finite elements (irregular mesh)	249	249	8.1×10^1

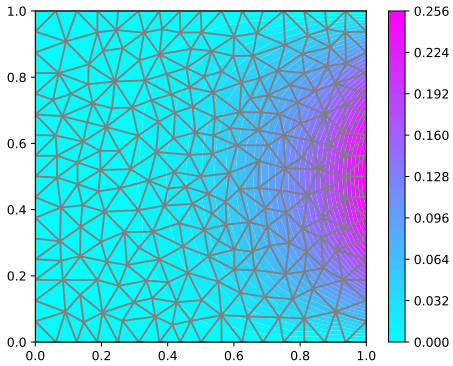
Table D.1: Condition numbers of discretized `texas2d-ex1` for a few selected discretizations.



(a) Illustration of true solution function.



(b) Illustration of solution and mesh for finite element discretization with uniform mesh with $n = 32$ (1024 degrees of freedom).



(c) Illustration of solution and mesh for finite element discretization with irregular mesh with 249 degrees of freedom).

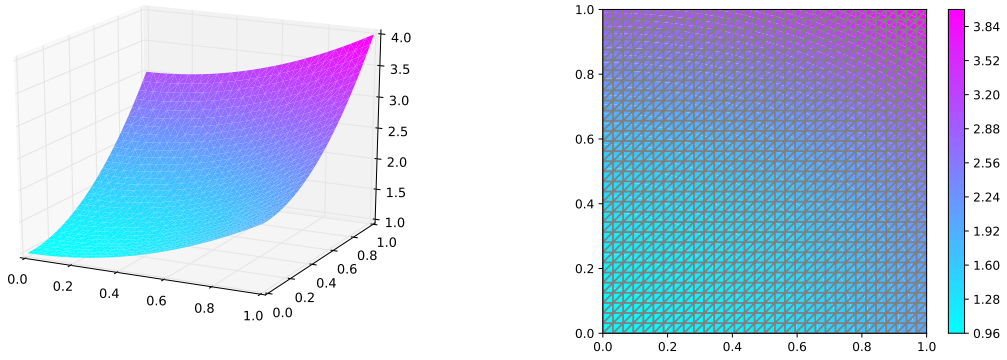
Figure D.3: Illustrations of true solution function and solutions of finite element discretizations of `texas2d-ex1`.

`fenics-simple`

This problem is taken from the FEniCS tutorial, [40]. It is the simple 2D Poisson problem whose discretization code is provided in [40, sections 2.1, 2.2, and 2.3]. This is a Poisson problem on a uniform finite element mesh over the unit square $[0,1]\times[0,1]$ with P1 elements. As we are not interested in utilizing FEniCS to solve the problem, it is used solely to for creating the test problem and the additional code in Listing D.1 is used to extract the explicit discretized linear system. With $n = 32$, (1024 degrees of freedom), the resulting matrix is full rank, and its condition number is 3.9×10^2.

Listing D.1: Code to extract linear system from FEniCS.

```
A = assemble(a)
b = assemble(L)
bc.apply(A,b)
Anp = A.array()
bnp = b.array()
```



(a) Illustration of true solution.

(b) Solution with overlaid mesh (2D representation).

Figure D.4: Illustrations of true solution for `fenics-simple` with $n = 32$ (i.e., 1024 total degrees of freedom).

`ir-inverse-interpolation`

This is a 2D inverse interpolation problem from the IR tools package [35]. With the package, the following MATLAB code can be used to generate the problem information and extract an explicit form of the matrix $\boldsymbol{A}$. With $n = 32$ ($n^2 \times n^2$ linear system), the resulting matrix is not full rank (rank = 924).

Listing D.2: Matlab generation code for `ir-inverse-interpolation`

```
n = 32
[A, b, x, ProbInfo] = PRinvinterp2(n);
I = eye(n^2);
res = zeros(n^2,n^2);
for i = 1:n^2
res(:,i) = A(I(:,i), 'notransp');
end
```

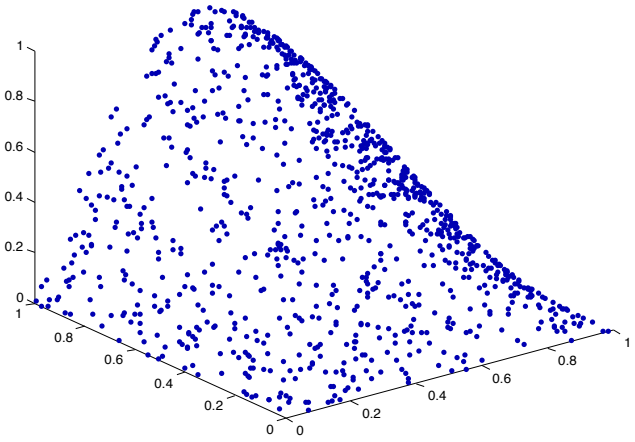


Figure D.5: Illustration of true solution for `ir-inverse-interpolation`.

ir-image-deblurring

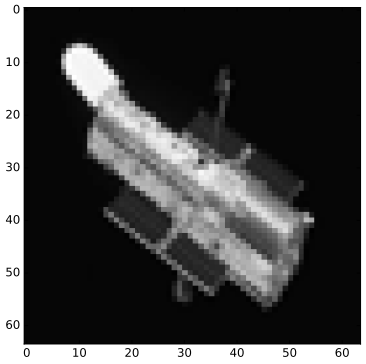
This is an image deblurring problem from the IR tools package presented in [35] based on a spatially variant rotational blur. With the package, the following MATLAB code can be used to generate the problem information with various images. With $n = 64$ (64×64 image), the resulting matrix is full rank, and its condition number is 1.3×10^4.

Listing D.3: Matlab generation code for `ir-image-deblurring-*`

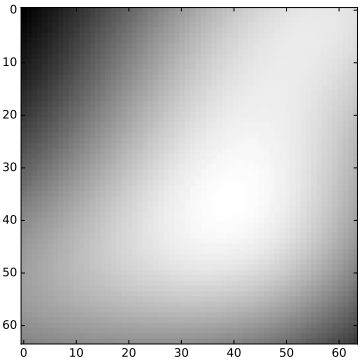
```
options = PRblurrotation('defaults')

% Uncomment one of the below lines to set the image
%options.trueImage = 'hst'
%options.trueImage = 'smooth'

n = 64
[A, b, x, ProbInfo] = PRblurrotation(n, options);
```



(a) Unblurred image for `ir-image-deblurring-hst`.



(b) Unblurred image for `ir-image-deblurring-smooth`.

Figure D.6: Images used with `ir-image-deblurring-*` test problems.

`ir-inverse-diffusion`

This is a 2D inverse diffusion problem from the IR tools package [35]. With the package, the following MATLAB code can be used to generate the problem information and extract an explicit form of the matrix $\boldsymbol{A}$. With $n = 32$ (32×32 image), the rank of the resulting matrix is 200 (it is not full rank).

Listing D.4: Matlab generation code for `ir-inverse-diffusion`

```
n = 32
[A, b, x, ProbInfo] = PRdiffusion(n);
I = eye(n^2);
res = zeros(n^2, n^2);
for i = 1:n^2
        res(:,i) = A(I(:,i), 'notransp');
end
```

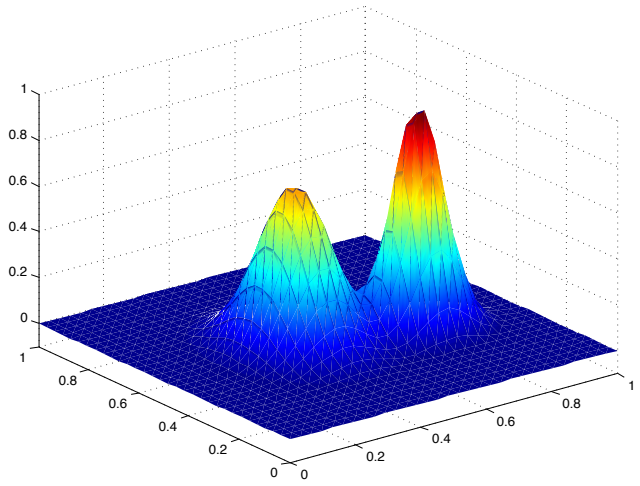


Figure D.7: Illustration of true solution for `ir-inverse-diffusion`.

References

[1] E. Carrier and M. Heath, "A sampling-based method for solving linear systems," 2018, 15th Copper Mountain Conference on Iterative Methods. [Online]. Available: https://easychair.org/smart-program/CM2018/2018-03-26.html#talk:64984

[2] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed. New York: McGraw-Hill, 2002.

[3] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia: SIAM, 2003.

[4] M. E. Hochstenbach and L. Reichel, "Subspace-restricted singular value decompositions for linear discrete ill-posed problems," *J. Comput. Appl. Math.*, vol. 235, pp. 1053–1064, 2010.

[5] S. Morigi, L. Reichel, and F. Sgallari, "A truncated projected SVD method for linear discrete ill-posed problems," *Numer. Algor.*, vol. 43, pp. 197–213, 2006.

[6] P. C. Hansen, *Discrete Inverse Problems: Insight and Algorithms*. Philadelphia: SIAM, 2010.

[7] D. Calvetti, L. Reichel, and A. Shuibi, "Enriched Krylov subspace methods for ill-posed problems," *Linear Alg. Appl.*, vol. 362, pp. 257–273, 2003.

[8] Y. Dong, H. Garde, and P. C. Hansen, "R^{3}GMRES: Including prior information in GMRES-type methods for discrete inverse problems," *Electronic Trans. Numer. Anal.*, vol. 42, pp. 136–146, 2014.

[9] Y. Saad, "Analysis of augmented Krylov subspace methods," *SIAM. J. Matrix Anal. Appl.*, vol. 18, pp. 435–449, 1997.

[10] Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM. J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, 1986.

[11] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd ed. New York: Dover, 2000.

[12] G. M. Phillips, *Interpolation and Approximation by Polynomials*. New York: Springer, 2003.

[13] M. Powell, *Approximation Theory and Methods*. New York: Cambridge University Press, 1981.

[14] L. N. Trefethen, *Approximation Theory and Approximation Practice*. Philadelphia: SIAM, 2013.

[15] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, Jan 2011.

[16] J. Garcke, "Sparse grid tutorial," 2007, [Accessed May 19, 2017]. [Online]. Available: https://pdfs.semanticscholar.org/9367/0b2257d60c866a6d32ad864f2eb640fe26f0.pdf

[17] J. Garcke, "Sparse grids in a nutshell," in *Sparse grids and applications*, ser. Lecture Notes in Computational Science and Engineering, J. Garcke and M. Griebel, Eds. Springer, 2013, vol. 88, pp. 57–80.

[18] J. Garcke, "PHAML methods: hierarchical basis multigrid," [Accessed May 19, 2017]. [Online]. Available: http://math.nist.gov/phaml/hbmg.html

[19] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90–93, Jan 1974.

[20] W. L. Briggs and V. E. Henson, *The DFT: An Owner's Manual for the Discrete Fourier Transform*. Philadelphia: SIAM, 1995.

[21] D. F. Elliott and K. R. Rao, *Fast Transforms: Algorithms, Analyses, Applications*. New York: Academic Press, 1982.

[22] K. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Boston: Academic Press, 1990.

[23] S. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Trans. on Signal Process.*, vol. 42, no. 5, pp. 1038–1051, May 1994.

[24] Z. Wang and B. R. Hunt, "The discrete W transform," *Appl. Math. Comput.*, vol. 16, no. 1, pp. 19–48, 1985.

[25] R. N. Bracewell, "Discrete Hartley transform," *J. Optical Soc. Amer.*, vol. 73, no. 12, pp. 1832–1835, Dec 1983.

[26] K. G. Beauchamp, *Applications of Walsh and Related Functions, with an Introduction to Sequency Theory*. New York: Academic Press, 1984.

[27] W. K. Pratt, W.-H. Chen, and L. R. Welch, "Slant transform image coding," *IEEE Trans. Commun.*, vol. COM-22, no. 8, pp. 1075–1093, Aug 1974.

[28] G. Lee, F. Wasilewski, K. Wohlfahrt, A. O'Leary, H. Nahrstaedt, and Contributors, "PyWavelets - wavelet transforms in Python," 2006, [Accessed November 14, 2017]. [Online]. Available: https://github.com/PyWavelets/pywt

[29] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1997.

[30] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, March 2011.

[31] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing, 2006.

[32] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed December 4, 2018]. [Online]. Available: http://www.scipy.org/

[33] G. Strang, "The discrete cosine transform," *SIAM Rev.*, vol. 41, no. 1, pp. 135–147, 1999.

[34] P. Corr, D. Stewart, P. Hanna, J. Ming, and F. Smith, "Discrete Chebyshev transform: A natural modification of the DCT," in *Proc. 15th Int. Conf. Pattern Recognition*, vol. 3, 2000, pp. 1142–1145.

[35] S. Gazzola, P. C. Hansen, and J. G. Nagy, "IR tools: a MATLAB package of iterative regularization methods and large-scale test problems," *Numerical Algorithms*, Aug 2018.

[36] M. Alnæs, J.B., J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, "The FEniCS project version 1.5," *Arch. of Numer. Software*, vol. 3, no. 100, 2015.

[37] H. Langtangen, *A FEniCS tutorial*. Berlin, Heidelberg: Springer, 2012, pp. 1–73.

[38] H. Sagan, *Space-Filling Curves*. New York: Springer, 1994.

[39] M. Bader, *Space-Filling Curves: An Introduction with Applications in Scientific Computing*. New York: Springer, 2013.

[40] H. P. Langtangen and A. Logg, *Solving PDEs in Python - The FEniCS Tutorial Volume I*. Springer, 2017. [Online]. Available: https://fenicsproject.org/pub/tutorial/pdf/fenics-tutorial-vol1.pdf

[41] M. Arioli, E. Noulard, and A. Russo, "Stopping criteria for iterative methods: applications to PDE's," *CALCOLO*, vol. 38, no. 2, pp. 97–112, Jun 2001.

[42] M. Arioli, "A stopping criterion for the conjugate gradient algorithm in a finite element method framework," *Numer. Math.*, vol. 97, no. 1, pp. 1–24, Mar 2004.

[43] M. Arioli, D. Loghin, and A. J. Wathen, "Stopping criteria for iterations in finite element methods," *Numer. Math.*, vol. 99, no. 3, pp. 381–410, Jan 2005.

[44] S. F. Ashby, M. J. Holst, T. A. Manteuffel, and P. E. Saylor, "The role of the inner product in stopping criteria for conjugate gradient iterations," *BIT*, vol. 41, no. 1, pp. 26–52, Jan 2001.

[45] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia: SIAM, 1994.

[46] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Philadelphia: SIAM, 1997.

[47] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. SIAM, Jan 2002.

[48] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Philadelphia: SIAM, 1998.

[49] V. A. Morozov, "On the solution of functional equations by the method of regularization," *Soviet Math. Dokl.*, vol. 7, pp. 414–417, 1966.

[50] P. C. Hansen, "Analysis of discrete ill-posed problems by means of the L-curve," *SIAM Rev.*, vol. 34, no. 4, pp. 561–580, Dec 1992.

[51] P. C. Hansen and D. P. O'Leary, "The use of the L-curve in the regularization of discrete ill-posed problems," *SIAM J. Sci. Comput.*, vol. 14, no. 6, pp. 1487–1503, Nov 1993.

[52] G. H. Golub, M. Heath, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[53] B. W. Rust, "Truncating the singular value decomposition for ill-posed problems, report nistir 6131," Mathematical and Computational Sciences Division, NIST, Tech. Rep., 1998.

[54] P. C. Hansen, M. E. Kilmer, and R. H. Kjeldsen, "Exploiting residual information in the parameter choice for discrete ill-posed problems," *BIT*, vol. 46, no. 1, pp. 41–59, Mar 2006.

[55] B. W. Rust and D. P. O'Leary, "Residual periodograms for choosing regularization parameters for ill-posed problems," *Inverse Problems*, vol. 24, no. 3, p. 034005, 2008.

[56] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[57] J. van Brakel, "Smoothed z-score algorithm," http://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data, 2016, [Accessed 04 October 2018].

[58] P. C. Hansen, "Regularization Tools: A MATLAB package for analysis and solution of discrete ill-posed problems," *Numer. Algorithms*, vol. 46, pp. 189–194, 2007.

[59] L. M. Delves and J. L. Mohamed, *Computational Methods for Integral Equations*. Cambridge: Cambridge University Press, 1985.

[60] P. C. Hansen, "Deconvolution and regularization with Toeplitz matrices," *Numer. Algorithms*, vol. 29, pp. 323–378, 2002.

[61] C. B. Shaw, "Improvement of the resolution of an instrument by numerical solution of an integral equation," *J. Math. Anal. Appl.*, vol. 37, no. 1, pp. 83–112, 1972.

[62] L. Greengard, "Spectral integration and two-point boundary value problems," *SIAM J. Numer. Anal.*, vol. 28, no. 4, pp. 1071–1080, Aug 1991.

[63] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. New York: Springer-Verlag, 1980.

[64] B. Fornberg, *A Practical Guide to Pseudospectral Methods*. Cambridge: Cambridge University Press, 1996.

[65] C. Su and T. Sarkar, "Adaptive multiscale moment method for solving two-dimensional Fredholm integral equation of the first kind," *J. of Electromagnetic Waves and Appl.*, vol. 13, no. 2, pp. 175–176, 1999.

[66] R. Fitzpatrick, "An example solution of Poisson's equation in 2-d," http://farside.ph.utexas.edu/teaching/329/lectures/node71.html, 2006, [Accessed December 10, 2018].