Design and analysis of high performance conjugate gradient and Lanczos type algorithms

Tyler Chen

Committee
Anne Greenbaum (chair),
Tom Trogodon,
Sasha Aravkin,
Maryam Fazel (GSR)

A thesis proposal
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics
University of Washington

Abstract

The conjugate gradient and Lanczos algorithms are among the most commonly used methods for symmetric eigenvalue problems. In this thesis proposal we provide an overview of the theory of these algorithms in exact arithmetic and in finite precision in the context of high performance scientific computing. We identify a range of related open questions suitable for study during a PhD and provide an indication of the approaches required to solve these problems. We additionally outline some more difficult problems of personal interest, which may be suitable for future research.

Notation

We denote scalars with Greek letters and vectors with bold Roman letters. The standard (Euclidian) inner product is denoted by $\langle \cdot, \cdot \rangle$. The i, j entry of a matrix $\mathbf{A}$ is denoted $[\mathbf{A}]_{i,j}$, and colons may be used to denote spans of rows or columns; for instance, $[\mathbf{A}]_{:,j}$ denotes the j-th column of $\mathbf{A}$. The transpose of a matrix $\mathbf{A}$ is denoted $\mathbf{A}^\mathsf{T}$. The matrix product is defined in the standard way.

We additionally make the following definitions.

$$
\begin{aligned}
\mathbf{e}_k &:= k\text{-th standard unit vector} \\
\|\mathbf{x}\| = \|\mathbf{x}\|_2 &:= \langle \mathbf{x}, \mathbf{x} \rangle^{1/2} \\
\lambda(\mathbf{A}) &:= \{\lambda : \lambda \text{ is an eigenvalue of } \mathbf{A}\} \\
\|\mathbf{A}\| &:= \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \\
\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{B}} &:= \langle \mathbf{x}, \mathbf{B}\mathbf{y} \rangle = \langle \mathbf{B}\mathbf{x}, \mathbf{y} \rangle \text{ where } \mathbf{B} \text{ is symmetric positive definite} \\
\lambda_\delta(\mathbf{A}) &:= \{\lambda + x : \lambda \in \lambda(\mathbf{A}), x \in [-\delta, \delta]\} \\
\mathcal{I}_\eta(\mathbf{A}) &:= [\lambda_{\min}(\mathbf{A}) - \eta, \lambda_{\max}(\mathbf{A}) + \eta] \\
\kappa_\eta(\mathbf{A}) &:= \frac{\lambda_{\max}(\mathbf{A}) + \eta}{\lambda_{\min}(\mathbf{A}) - \eta} \\
\mathcal{P}_k &:= \{p : \deg p < k\} \\
\mathcal{P}_k^0 &:= \{p : \deg p \leq k, p(0) = 1\} \\
\mathrm{T}_{\text{mv}} &:= \text{computation time for matrix vector product} \\
\mathrm{C}_{\text{mv}} &:= \text{communication time for matrix vector product} \\
\mathrm{C}_{\text{gr}} &:= \text{communication time for global reduction}
\end{aligned}
$$

Contents

1 Introduction

The overarching motivation for this thesis proposal is the fact that *the less time scientists have to wait for code to run, the more time they can spend thinking about the problems they are tackling.* Whether used directly, or as the building blocks for more complex algorithms, solving linear systems and computing matrix functions are at the core of modern computational science. Therefore, developing methods to solve linear systems and compute matrix functions quickly and accurately will enable scientists from a range of disciplines to work more efficiently to solve the problems they care about.

The conjugate gradient (CG) and Lanczos algorithms are among the most popular methods for solving symmetric positive definite linear systems, and computing the eigenvalues (and matrix functions) of a symmetric matrix. However, while both algorithms have many nice theoretical properties, they often behave very differently in finite precision than exact arithmetic theory predicts. Thus, the study of the effect of finite precision arithmetic on these algorithms has been an active area of study since their introductions.

Computing environments are constantly evolving. For instance, today's cell phones are thousands of times more powerful than the Apollo Guidance Computer used to take humans to the moon [Ken19]. Likewise, the physical capabilities of supercomputers have improved drastically. Today, the largest supercomputers are able to compute more than one *exaflop* per second (that's a billion billion (10^{18}) floating point operations each second) [Orn]. These supercomputers are able to achieve such performance by linking hundreds or thousands of smaller computers together with high speed connections so that they each unit can work as part of the whole.

These advances in hardware have naturally brought about massive changes in the problems which are now tractable numerically. Problems considered "large" a few decades ago can now be solved in real time on a laptop, and problems which were previously too computationally expensive to even attempt can be solved in days or weeks on large clusters. As such, many new types of problems are now being solved, and new fields such as machine learning and data science are emerging. Of course, the changes in hardware also bring about the need for new algorithms designed to be efficient on the computers of today and of tomorrow.

Traditionally the runtime of numerical methods has been measured in terms of the number of floating point operations they require. However, on today's supercomputers the dominant costs are moving and reading data rather than floating point operations. For example, the speed of light limits the rate messages can be sent between two points. Similarly, memory has physical limitations about how fast it can be read from and written to. This has a significant effect on the CG and Lanczos algorithms. Indeed, when CG is run on most supercomputers it is only able to achieve 1-2% of the maximum number of floating point operations per unit time [Hpc].

To address this bottleneck, many *mathematically equivalent* variants of the CG algorithm have been introduced; see for instance [Ros83; Saa85; Chr87; Meu87; Saa89; CG89; SG06; GV14; EG16; CCV19b, etc.]. Broadly speaking, these variants aim to rearrange the standard CG algorithm in such a way that the communication occurs less frequently, or is overlapped with other computations. As a result, the time per iteration of these methods is reduced on parallel machines.

Notably, however, many communication hiding methods suffer numerical problems due to the rearranging of computations within each iteration. Indeed, it is well known that the conjugate gradient algorithm is particularly sensitive to rounding errors, and any modification to the CG algorithm will have an effect on numerical behavior. Specifically, both the rate

of convergence (the number of iterations to reach a given level of accuracy) and the maximal attainable accuracy of any CG implementation may be *severely* impacted by carrying out computations in finite precision.⌋

Ultimately, however, it is neither the time per iteration nor the number of iterations which practitioners care about. Rather, the important quantity is the *time to solution*. For iterative methods this amounts to the product of the time per iteration with the total number of iterations required to reach the desired accuracy. Thus, when designing fast iterative methods it is important to balance speedups in the time per iteration with the number of additional iterations required due to numerical deficiencies.

Organization

The remainder of this document is organized as follows. In sections 2.1 and 2.2, I provide a standard background on some relevant theoretical properties of the Lanczos and conjugate gradient algorithms in exact arithmetic, and in sections 3.2 and 3.3 I provide a summary of relevant theory relating to the numerical analysis of these algorithms. In section 4 I provide an overview of recent work on high performance conjugate gradient variants. Finally, in section 6 I present a plan for future work and provide an overview of my past contributions.

This document is not meant to be self contained; for instance, I only describe my past work at a high level leaving full details to the papers in the appendix. It is, however, meant to demonstrate that I have the necessary background knowledge to begin a successful thesis, and to outline the goals I have for my future work.

Some portions of this document are taken more or less verbatim from other works I have written. These sections are surrounded by the symbols ⌈⌋. Copies of the original works from which the quotes have been taken are included as supplementary material.

2 Exact arithmetic theory

In exact arithmetic, the theory of Krylov subspace methods for symmetric matrices is well understood and relatively straightforward. In this section we provide an overview of some basic results about the Lanczos and conjugate gradient algorithms in exact arithmetic.

Suppose we are given an arbitrary real symmetric matrix $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$, where $\mathbf{U}$ is an orthogonal matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues, and vector $\mathbf{b}$. A common task in scientific computing is to compute $f(\mathbf{A})\mathbf{b}$ for some function $f : \mathbb{R} \to \mathbb{R}$, where $f(\mathbf{A}) := \mathbf{U}f(\mathbf{\Lambda})\mathbf{U}^\mathsf{T}\mathbf{b}$, and $f(\mathbf{\Lambda})$ is simply f applied to the entries of the diagonal matrix $\mathbf{\Lambda}$. For instance, the case $f(x) = 1/x$ corresponds to solving a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. We note that while we use f for the matrix function, it is not technically the same as the real valued function $f : \mathbb{R} \to \mathbb{R}$, although we use the same notation as any real valued function will induce a matrix function through the previous definition.

A straightforward way to compute $f(\mathbf{A})\mathbf{b}$ would be to compute the eigendecomposition of $\mathbf{A}$ and follow the definition above. However, since the ultimate aim is to compute $f(\mathbf{A})\mathbf{b}$, there is some hope that we do not need to compute $f(\mathbf{A})$ explicitly. For instance, if $f(z) = c_0 + c_1 z + \cdots + c_k z^k$ is a polynomial, it is easy to verify that $f(\mathbf{A}) = c_0\mathbf{I} + c_1\mathbf{A} + \cdots c_k\mathbf{A}^k$. In this case, we can easily compute $f(\mathbf{A})\mathbf{b}$ using only matrix vector products with $\mathcal{O}(n)$ additional storage in time $\mathcal{O}(k\,\mathrm{T}_{\mathrm{mv}})$.

Thus, a simple approach to computing $f(\mathbf{A})\mathbf{b}$ would be to approximate f by a polynomial p

and then compute $p(\mathbf{A})\mathbf{b}$, which we can evaluate using only matrix vector products, which is made more precise by theorem 2. Even so, This provides some motivation to study the spaces spanned by polynomials of increasing degree.

Definition 1. *Given a square matrix* $\mathbf{A}$ *and vector* $\mathbf{b}$*, the* k*-th Krylov subspace generated by* $\mathbf{A}$ *and* $\mathbf{b}$ *is defined as*

$$\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \operatorname{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \ldots, \mathbf{A}^{k-1}\mathbf{b}\} = \{p(\mathbf{A})\mathbf{b} : \deg p < k\}.$$

⌈Suppose at the beginning of step k that we have already computed an orthonormal basis $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{k-1}\}$ for $\mathcal{K}_{k-1}(\mathbf{A}, \mathbf{b}) = \{\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{k-2}\mathbf{b}\}$. If we use the Gram-Schmidt algorithm, we would obtain $\mathbf{q}_k$ by orthogonalizing $\mathbf{A}^{k-1}\mathbf{b}$ against each of the vectors in the basis $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{k-1}\}$.⌋ However, $\mathbf{A}^{k-1}\mathbf{b}$ and $\mathbf{A}^{k-2}\mathbf{b}$ may be close to linearly dependent as both will eventually converge to the direction of the dominant eigenvector of $\mathbf{A}$. In finite precision this can cause issues, as information about the Krylov subspaces will be degraded.

An alternate approach is to instead orthogonalize $\mathbf{A}\mathbf{q}_{k-1}$ against $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{k-1}\}$. It is trivial to verify that the vector obtained with this approach is some scalar multiple of the vector obtained by the Gram-Schmidt approach. This approach is referred to as the Arnoldi algorithm and yields a factorization

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{H}_k + h_{k+1,k}\mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T}$$

where $\mathbf{H}_k$ is an upper Hessenburg matrix.

2.1 Lanczos

⌈If $\mathbf{A}$ is symmetric, then $\mathbf{H}_k = \mathbf{Q}_k^\mathsf{T}\mathbf{A}\mathbf{Q}_k$ is also symmetric. This means that $\mathbf{H}_k$ is upper Hessenburg and symmetric, so it must be tridiagonal.⌋ We use $\mathbf{T}_k$ to denote this symmetric tridiagonal matrix. It is clear that we can save some work in the orthogonalization step described above by taking advantage of fact that $\mathbf{A}\mathbf{q}_j$ will be orthogonal to $\mathbf{q}_i$ for $i < j - 1$; i.e. if $\mathbf{T}_k$ has diagonal entries $\{\alpha_j\}_{j=1}^k$ and off diagonal entries $\{\beta_j\}_{j=1}^{k-1}$, then

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{T}_k + \beta_k\mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T}. \tag{1}$$

That is to say, the columns of $\mathbf{Q}_k$ satisfy the symmetric three term recurrence

$$\mathbf{A}\mathbf{q}_j = \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j\mathbf{q}_j + \beta_j\mathbf{q}_{j+1}, \qquad j = 1, 2, \ldots, k-1.$$

For notational convenience, we will often write $\mathbf{Q}$ and $\mathbf{T}$ when it is clear what k is.

When $\mathbf{A}$ is symmetric, it is easy to envision that the Arnoldi algorithm described at the end of the previous section can be simplified somewhat by taking advantage of this three term recurrence.

The Lanczos algorithm was introduced in 1950 by Lanczos in [Lan50]. It was initially designed as a direct method for computing eigenvalues and vectors of a symmetric matrix. However, while the algorithm has many nice theoretical properties its early use was limited due to numerical instabilities. However, as these instabilities became better understood, the algorithm gained popularity as an iterative method.

The theory described in this section is based on the decomposition eq. (1) which in exact arithmetic can be obtained through the standard implementation Lanczos.

Algorithm 1 Lanczos

procedure LANCZOS($\mathbf{A}$,$\mathbf{r}$)
  $\mathbf{q}_0 = \mathbf{0}$, $\beta_{-1} = 0$, $\mathbf{q}_1 = \mathbf{r}/\|\mathbf{r}\|$
  for $k = 1, 2, \ldots$ **do**
    $\tilde{\mathbf{q}}_{k+1} = \mathbf{A}\mathbf{q}_k - \beta_{k-1}\mathbf{q}_{k-1}$
    $\alpha_k = \langle \tilde{\mathbf{q}}_{k+1}, \mathbf{q}_k \rangle$
    $\mathbf{q}_{k+1} = \tilde{\mathbf{q}}_{k+1} - \alpha_k \mathbf{q}_k$
    $\beta_k = \|\mathbf{q}_{k+1}\|$
    $\mathbf{q}_{k+1} = \mathbf{q}_{k+1}/\beta_k$
  end for
end procedure

2.1.1 Ritz values and vectors

Suppose $\mathbf{T}$ is the tridiagonal matrix generated at step k of a Lanczos recurrence with corresponding basis vectors $\mathbf{Q}$. We can compute the eigendecomposition of $\mathbf{T}$,

$$\mathbf{T}\mathbf{S} = \mathbf{S}\mathbf{\Theta}.$$

Then, defining $\mathbf{y}_i := \mathbf{Q}\mathbf{s}_i$ it is easy to observe

$$\mathbf{A}\mathbf{y}_i = \mathbf{Q}\mathbf{T}\mathbf{y}_i + \beta_k \mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T}\mathbf{s}_i = \theta_i \mathbf{y}_i + \beta_k \mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T}\mathbf{s}_i.$$

Thus, if β_k or the k-th component of $\mathbf{s}_i$ is not too large then θ_i and $\mathbf{y}_i$ will be good approximations to the eigenpairs of $\mathbf{A}$. These quantities are referred to as Ritz values and vectors respectively.

In particular at step n the Lanczos vectors generated will span $\mathbb{R}^n$ so that $\beta_n = 0$. In this case the Ritz values and vectors will give exactly the eigenvalues and vectors of $\mathbf{A}$. Thus, the eigenproblem with $\mathbf{A}$ is reduced to an eigenproblem with the tridiagonal matrix $\mathbf{T}$.

2.1.2 Matrix function approximation

The Lanczos algorithm can be used to approximate functions of the matrix $\mathbf{A}$ applied to $\mathbf{b}$ in a "black box" manner; i.e. without any knowledge of the eigenvalues and vectors of $\mathbf{A}$. We now outline the basic theory for this.

Theorem 1. *Apply Lanczos to $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$ for k steps to obtain $\mathbf{Q}$ and $\mathbf{T}$. Then, for any polynomial $p : \mathbb{R} \to \mathbb{R}$ of degree less than k, and vector $\mathbf{b}$,*

$$p(\mathbf{A})\mathbf{b} = \|\mathbf{b}\|\mathbf{Q}p(\mathbf{T})\mathbf{e}_1.$$

Proof. Fix $k > 2$. Note that $\mathbf{T}$ is tridiagonal, so $\mathbf{T}^i$ has half bandwidth at most i. In particular, for $i < k$, $[\mathbf{T}^i]_{k,1} = \mathbf{e}_k^\mathsf{T}\mathbf{T}^i\mathbf{e}_1 = 0$. Thus, for any nonnegative integers i and j with $i < k$,

$$\begin{aligned}
\mathbf{A}^{j+1}\mathbf{Q}\mathbf{T}^i\mathbf{e}_1 &= \mathbf{A}^j(\mathbf{A}\mathbf{Q})\mathbf{T}^i\mathbf{e}_1 \\
&= \mathbf{A}^j(\mathbf{Q}\mathbf{T} + \delta_k \mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T})\mathbf{T}^i\mathbf{e}_1 \\
&= \mathbf{A}^j\mathbf{Q}\mathbf{T}^{i+1}\mathbf{e}_1 + \delta_k \mathbf{A}^j\mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T}\mathbf{T}^i\mathbf{e}_1 \\
&= \mathbf{A}^j\mathbf{Q}\mathbf{T}^{i+1}\mathbf{e}_1.
\end{aligned}$$

Then, for any $j < k$, by inductively applying the above result we find

$$\mathbf{A}^j\mathbf{q}_1 = \mathbf{A}^j\mathbf{Q}\mathbf{e}_1 = \mathbf{Q}\mathbf{T}^j\mathbf{e}_1.$$

The theorem then follows by setting $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$. □

With theorem 1 in mind, it's reasonable to expect that if $f : \mathbb{R} \to \mathbb{R}$ is well approximated by a polynomial of degree less than k, that $f(\mathbf{A})\mathbf{b}$ should be well approximated by

$$\mathbf{x}_k = \|\mathbf{b}\|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1.$$

The degree to which this is the case is made explicit by the following theorem.

Theorem 2. *Apply Lanczos to $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$ for k steps to obtain $\mathbf{Q}$ and $\mathbf{T}$ and set $\mathbf{x}_k = \|\mathbf{b}\|\mathbf{Q}f(\mathbf{T})\mathbf{e}_1$. Then,*

$$\|f(\mathbf{A})\mathbf{b} - \mathbf{x}_k\| \leq 2\|\mathbf{b}\| \min_{p \in \mathcal{P}_{k-1}} \left\{ \max_{x \in \mathcal{I}(\mathbf{A})} |f(x) - p(x)| \right\}, \qquad \mathcal{I}(\mathbf{A}) = [\lambda_{min}(\mathbf{A}), \lambda_{max}(\mathbf{A})].$$

Proof from [MMS18]. For any polynomial p of degree less than k, by a simple application of the triangle inequality and the submultiplicative property of the spectral norm, we have,

$$\begin{aligned}
\|f(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}f(\mathbf{T})\mathbf{e}_1\| &= \|p(\mathbf{A})\mathbf{q}_1 - \mathbf{Q}p(\mathbf{T})\mathbf{e}_1 + (f(\mathbf{A}) - p(\mathbf{A}))\mathbf{q}_1 - \mathbf{Q}(f(\mathbf{T}) - p(\mathbf{T}))\mathbf{e}_1\| \\
&\leq \|(f(\mathbf{A}) - p(\mathbf{A}))\mathbf{q}_1\| + \|\mathbf{Q}(f(\mathbf{T}) - p(\mathbf{T}))\mathbf{e}_1\| \\
&\leq \|f(\mathbf{A}) - p(\mathbf{A})\|\|\mathbf{q}_1\| + \|\mathbf{Q}\|\|f(\mathbf{T}) - p(\mathbf{T})\|\|\mathbf{e}_1\| \\
&= \|f(\mathbf{A}) - p(\mathbf{A})\| + \|f(\mathbf{T}) - p(\mathbf{T})\|.
\end{aligned}$$

Moreover, by definition of matrix norm,

$$\|f(\mathbf{A}) - p(\mathbf{A})\| = \max_{x \in \lambda(\mathbf{A})} |f(x) - p(x)| \leq \max_{x \in \mathcal{I}(\mathbf{A})} |f(x) - p(x)|.$$

Note now that $\mathbf{Q}^\mathsf{H}\mathbf{A}\mathbf{Q} = \mathbf{T}$ and $\mathbf{Q}^\mathsf{H}\mathbf{Q} = \mathbf{I}$ so that $\lambda(\mathbf{T}) \subseteq \mathcal{I}(\mathbf{A})$. Thus,

$$\|f(\mathbf{T}) - p(\mathbf{T})\| = \max_{x \in \lambda(\mathbf{T})} |f(x) - p(x)| \leq \max_{x \in \mathcal{I}(\mathbf{A})} |f(x) - p(x)|.$$

Since the previous bounds hold for all polynomials p of degree less than k, they must hold for the optimal polynomial of degree less than k. □

2.2 Conjugate Gradient

The conjugate gradient algorithm, HS-CG, was introduced in 1952 by Hestenes and Stiefel in [HS52] for solving symmetric positive definite linear systems. Like the Lanczos algorithm it was initially indended as a direct method but gained popularity as an iterative method. The popularity of CG is due in part to the fact (i) that it is matrix free, meaning that the algorithm never needs to access the entries of $\mathbf{A}$ explicitly, only to be able to compute the product $\mathbf{v} \mapsto \mathbf{A}\mathbf{v}$, (ii) that it requires only $\mathcal{O}(n)$ storage and floating point operations each iteration (in addition to the matrix vector product), and (iii) the development of effective preconditioners. However, as stated in the introduction, on high performance machines, CG suffers from communication bottlenecks.

Algorithm 2 Hestenes and Stiefel Conjugate Gradient (preconditioned)

procedure HS-CG($\mathbf{A}, \mathbf{M}, \mathbf{b}, \mathbf{x}_0$)
  INITIALIZE()
  for $k = 1, 2, \ldots$ **do**
    $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
    $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$, $\tilde{\mathbf{r}}_k = \mathbf{M}^{-1}\mathbf{r}_k$
    $\nu_k = \langle \tilde{\mathbf{r}}_k, \mathbf{r}_k \rangle$
    $\beta_k = \nu_k / \nu_{k-1}$
    $\mathbf{p}_k = \tilde{\mathbf{r}}_k + \beta_k \mathbf{p}_{k-1}$
    $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
    $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle$
    $\alpha_k = \nu_k / \mu_k$
  end for
end procedure

In this section we provide convergence theory for CG. As in the previous section, this theory is built entirely on the assumption that there exists an algorithm which produces the CG iterates defined below (such as HS-CG in exact arithmetic).

Definition 2. *The k-th CG iterate, denoted by $\mathbf{x}_k$, is the unique vector in $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ minimizing the $\mathbf{A}$-norm of the distance to the true solution of linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. That is,*

$$\mathbf{x}_k := \operatorname*{argmin}_{\mathbf{x} \in \mathcal{K}_k(\mathbf{A}, \mathbf{b})} \|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}\|_{\mathbf{A}}.$$

With this definition in mind, we can provide error bounds for the accuracy of the CG iterates $\mathbf{x}_k$. In particular, we provide a sharp bound based on the minimax polynomial on the set of eigenvalues of $\mathbf{A}$, and a relaxation of this bound giving a rate of convergence dependent only on the condition number of $\mathbf{A}$.

Theorem 3. *The k-th CG iterate, $\mathbf{x}_k$, satisfies the minimax error bound*

$$\frac{\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}}}{\|\mathbf{A}^{-1}\mathbf{b}\|_{\mathbf{A}}} \leq \min_{p \in \mathcal{P}_k^0} \left\{ \max_{x \in \lambda(\mathbf{A})} |p(x)| \right\}.$$

Proof. By definition, the conjugate gradient iterates $\mathbf{x}_k$ are optimal over the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ in the $\mathbf{A}$-norm. Therefore, since $\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \{p(\mathbf{A})\mathbf{b} : \deg p < k\}$,

$$\begin{aligned}
\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}} &= \min_{p \in \mathcal{P}_{k-1}} \|\mathbf{A}^{-1}\mathbf{b} - p(\mathbf{A})\mathbf{b}\|_{\mathbf{A}} \\
&= \min_{p \in \mathcal{P}_{k-1}} \|\mathbf{A}^{-1/2}(\mathbf{I} - \mathbf{A}^{1/2}p(\mathbf{A})\mathbf{A}^{1/2})\mathbf{A}^{-1/2}\mathbf{b}\|_{\mathbf{A}} \\
&= \min_{p \in \mathcal{P}_k^0} \|p(\mathbf{A})\mathbf{A}^{-1/2}\mathbf{b}\|
\end{aligned}$$

where we have used the fact that

$$\{1 - x^{1/2}p(x)x^{1/2} : \deg p < k\} = \{p(x) : \deg p \leq k, p(0) = 1\}.$$

Now note that

$$\|p(\mathbf{A})\mathbf{A}^{-1/2}\mathbf{b}\| \leq \|p(\mathbf{A})\|\|\mathbf{A}^{-1/2}\mathbf{b}\| = \|p(\mathbf{A})\|\|\mathbf{A}^{-1}\mathbf{b}\|_{\mathbf{A}}$$

and that

$$\|p(\mathbf{A})\| = \max_{x \in \lambda(\mathbf{A})} |p(x)|.$$

This gives the bound

$$\frac{\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}}}{\|\mathbf{A}^{-1}\mathbf{b}\|_{\mathbf{A}}} \leq \min_{p \in \mathcal{P}_k^0} \left\{ \max_{x \in \lambda(\mathbf{A})} |p(x)| \right\}.$$

□

We note that this bound is independent of the right hand side vector $\mathbf{b}$, and therefore cannot be expected to be tight for a given right hand side. However, for any iteration k, there exists some right hand side which attains the bound [Gre79].

To obtain a tight bound which takes the right hand side into account would result in a weighted least squares problem, with weights depending on the size of $\mathbf{b}$ in the direction of the eigenvectors of $\mathbf{A}$.

Corollary 1. *The k-th CG iterate, $\mathbf{x}_k$, satisfies the Chyebyshev ($\sqrt{\kappa}$) error bound,*

$$\frac{\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}}}{\|\mathbf{A}^{-1}\mathbf{b}\|_{\mathbf{A}}} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k.$$

Proof. Note that $\lambda(\mathbf{A}) \subseteq \mathcal{I}(\mathbf{A})$ so that finding the degree $< k$ minimax polynomial on the interval $\mathcal{I}(\mathbf{A})$ is a relaxation of the problem of finding the minimax polynomial on the eigenvalues of A. The minimax polynomial on an single closed interval is the Chebyshev polynomial whose maximum absolute deviation from zero on the interval is well known. Specifically,

$$\min_{p \in \mathcal{P}_k^0} \left\{ \max_{x \in \lambda(\mathbf{A})} |p(x)| \right\} \leq \min_{p \in \mathcal{P}_k^0} \left\{ \max_{x \in \mathcal{I}(\mathbf{A})} |p(x)| \right\} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k$$

□

While the bound in corollary 1 is typically quite pessimistic, we include it as a similar bound holds in finite precision. The same convergence bound holds for Chebyshev iteration, which requires that estimates of $\lambda_{\min}$ and $\lambda_{\max}$ are known ahead of time. However, as Chebyshev iteration does not require inner products, the cost per iteration may be much lower than that of CG on parallel machines.

2.2.1 Preconditioning

[Notice that to obtain the solution $x = \mathbf{A}^{-1}\mathbf{b}$, we could instead solve

$$\mathbf{R}^{-\mathsf{T}}\mathbf{A}\mathbf{R}^{-1}\mathbf{y} = \mathbf{R}^{-\mathsf{T}}\mathbf{b} \tag{2}$$

and then set $\mathbf{x} = \mathbf{R}^{-1}\mathbf{y}$, where $\mathbf{R}$ is any full rank square matrix.

If $\mathbf{A}$ is symmetric positive definite, then this new system is also symmetric positive definite. Thus, if the spectrum of $\mathbf{R}^{-\mathsf{T}}\mathbf{A}\mathbf{R}^{-1}$ is "better behaved" than the spectrum of $\mathbf{A}$, the bound in theorem 3 using $\mathbf{R}^{-\mathsf{T}}\mathbf{A}\mathbf{R}^{-1}$ will be stronger than the bound using just $\mathbf{A}$. In this case, the conjugate gradient method can be expected to converge significantly faster on the preconditioned system than on the original system.

By writing out the unpreconditioned CG algorithm for the preconditioned system in eq. (2) it is easy to show that we only need to be able to evaluate the map $\mathbf{v} \mapsto \mathbf{M}^{-1}\mathbf{v}$, where $\mathbf{M}^{-1} = \mathbf{R}^{-\mathsf{T}}\mathbf{R}^{-1}$. In particular, the individual factors $\mathbf{R}^{-\mathsf{T}}$ and $\mathbf{R}^{-1}$ need not be known [Gre97b]. This gives the preconditioned conjugate gradient algorithm, displayed in algorithm 2.]

2.3 Relationship between CG and Lanczos

The conjugate gradient algorithm applied to $\mathbf{Ax} = \mathbf{b}$ is mathematically equivalent to using the Lanczos algorithm with $\mathbf{A}$ and $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$ to compute $f(\mathbf{A})\mathbf{b}$ where $f(x) = 1/x$ in the sense that the important quantities from each algorithm can be recovered from the other. In this special case, we can obtain stronger optimality bounds of the iterates produced than those in theorem 2, and no longer need to store all Lanczos vectors $\mathbf{Q}$.

While the CG algorithm can be entirely derived from the Lanczos algorithm [Gre97b, Section 2.5], we simply show that the iterates produced are equivalent, and show how to obtain the Lanczos algorithm from CG. This relationship is important for the analysis of the CG algorithm in finite precision as discussed in section 3.3.

Theorem 4. *The iterates $\mathbf{x}_k = \|\mathbf{b}\|\mathbf{Q}\mathbf{T}^{-1}\mathbf{e}_1$ produced by using the Lanczos algorithm to compute $f(\mathbf{A})\mathbf{b}$ with $f(x) = 1/x$ are CG iterates.*

Proof. Recall that $\mathbf{Q}$ spans the Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$. Thus,

$$\mathbf{x}_k = \min_{\mathbf{y}} \|\mathbf{A}^{-1}\mathbf{b} - \mathbf{Q}\mathbf{y}\|_{\mathbf{A}} = \min_{\mathbf{y}} \|\mathbf{A}^{-1/2}\mathbf{b} - \mathbf{A}^{1/2}\mathbf{Q}\mathbf{y}\|$$

which has solution

$$\mathbf{y} = (\mathbf{Q}^{\mathsf{H}}\mathbf{A}^{1/2}\mathbf{A}^{1/2}\mathbf{Q})^{-1}(\mathbf{Q}\mathbf{A}^{1/2})\mathbf{A}^{-1/2}\mathbf{b} = \mathbf{Q}^{\mathsf{H}}\mathbf{A}\mathbf{Q}\mathbf{b} = \|\mathbf{b}\|\mathbf{T}^{-1}\mathbf{e}_1. \qquad \square$$

Similarly, the Lanczos algorithm can be derived from the CG algorithm [Gre97b, Exercise 2.6].

Theorem 5. *Both $\mathbf{Q}$ and $\mathbf{T}$ from the Lanczos algorithm applied to the normalized initial CG residual $\mathbf{r}_0/\|\mathbf{r}_0\| = \mathbf{b}/\|\mathbf{b}\|$ can be obtained from quantities computed in the CG algorithm.*

In particular, the columns of $\mathbf{Q}$, and the diagonal and the off-diagonal entries of $\mathbf{T}$ are respectively given by

$$\mathbf{q}_{j+1} = (-1)^j \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}, \qquad \gamma_j = \left(\frac{1}{\alpha_{j-1}} - \frac{\beta_{j-1}}{\alpha_{j-2}}\right), \qquad \delta_j = \frac{1}{\alpha_{j-1}}\frac{\|\mathbf{r}_j\|}{\|\mathbf{r}_{j-1}\|}.$$

Proof. Define $\mathbf{q}_{j+1}$ as above. These must be (up to a sign) the vectors produced by the Lanczos algorithm, since the $\mathbf{r}_j$ span the Krlyov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$.

Then, substituting the recurrences for $\mathbf{r}_j$ and $\mathbf{p}_j$ as necessary, and using the identity $\|\mathbf{r}_j\|^2 = \beta_j$, it is not hard to show that,

$$\mathbf{A}\mathbf{q}_j = = \frac{1}{\alpha_{j-1}}\frac{\|\mathbf{r}_j\|}{\|\mathbf{r}_{j-1}\|}\mathbf{q}_{j+1} + \left(\frac{1}{\alpha_{j-1}} - \frac{\beta_{j-1}}{\alpha_{j-2}}\right)\mathbf{q}_j + \frac{1}{\alpha_{j-2}}\frac{\|\mathbf{r}_{j-1}\|}{\|\mathbf{r}_{j-2}\|}\mathbf{q}_{j-1}$$

This is a three term symmetric recurrence for the $\mathbf{q}_j$'s, so it must be the Lanczos recurrence. $\square$

3 Finite precision

Up to this point we have not considered the effects of finite precision on the Lanczos and CG algorithms. For this reason we have not discussed the implementation of these algorithms,

as such details do not affect the convergence theory of such algorithms in exact arithmetic, provided that the implementations are mathematically equivalent. However, in finite precision, implementation details turn out to have a fairly significant effect on these algorithms.

Recall that in exact arithmetic the basis vectors $\mathbf{q}_k$ generated by the Lanczos procedure are orthonormal; i.e. $\langle \mathbf{q}_j, \mathbf{q}_k \rangle = 0$ if $j \neq k$ and 1 if $j = k$. In finite precision, it is easy to observe that this orthogonality may be lost if either Lanczos or HS-CG is used. While this is perhaps not surprising considering that both algorithms rely on inductive arguments to avoid orthogonalizing against all previous basis vectors, it does have a significant effect on their behavior in finite precision.

3.1 Finite precision model

⌈We assume the standard model of floating point arithmetic,

$$|\operatorname{fp}(\alpha \circ \beta) - \alpha \circ \beta| \leq \epsilon |\alpha \circ \beta| \tag{3}$$

for floating point numbers α and β and standard operations $\circ \in \{+, -, \times, \div\}$, where ϵ is the unit roundoff of the machine. From this, to first order,

$$\|\operatorname{fp}(\mathbf{x} + \alpha\mathbf{y}) - (\mathbf{x} + \alpha\mathbf{y})\| \leq \epsilon \left(\|\mathbf{x}\| + 2|\alpha|\|\mathbf{y}\|\right) \tag{4}$$

$$\|\operatorname{fp}(\mathbf{A}\mathbf{x}) - \mathbf{A}\mathbf{x}\| \leq \epsilon\, c\, \|\mathbf{A}\|\|\mathbf{x}\| \tag{5}$$

$$\|\operatorname{fp}(\langle \mathbf{x}, \mathbf{y} \rangle) - \langle \mathbf{x}, \mathbf{y} \rangle\| \leq \epsilon\, n\, \|\mathbf{x}\|\|\mathbf{y}\| \tag{6}$$

where c is a dimensional constant depending on the method of matrix multiplication and n is the length of the vectors $\mathbf{x}$ and $\mathbf{y}$.⌋

3.2 Lanczos

When Lanczos is run in finite precision, it is often observed that while the Ritz values and vectors as defined in section 2.1.1 continue toprovide good estimates of eigenvalues and vectors, that multiple copies of a single eigenvalue appear. This is due in part to the fact that the Lanczos vectors $\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \ldots\}$ lose orthogonality allowing the recurrence to continue past iteration n without terminating.

Much of what is understood about the finite precsion behavior of Lanczos comes from understanding a general perturbed Lanczos recurrence. Such a recurrence can be written in the form

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{T}_k + \beta_k \mathbf{q}_{k+1}\mathbf{e}_k^\mathsf{T} + \mathbf{F}_k. \tag{7}$$

Defining $\mathbf{f}_j := [\mathbf{F}_k]_{:,j}$ we can equivalently write

$$\mathbf{A}\mathbf{q}_j = \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j \mathbf{q}_j + \beta_j \mathbf{q}_{j+1} + \mathbf{f}_j.$$

3.2.1 Paige's theory

Much of what is known about the Lanczos algorithm in finite precision is due to Paige in his PhD thesis and related works. These anlyses provide information about the direction in which orthogonality is lost. They also show that if Lanczos is run in finite precision, that the corresponding Lanczos vectors are nearly unit norm, that perturbation term from eq. (7) are small, and that successive Lanczos vectors are nearly orthogonal, and that the spectum of $\mathbf{T}$ is more or less contained in the interval bounded by the extremal eigenvalues of $\mathbf{A}$.

Theorem 6 (Paige taken from [Gre89, Section 5]). *At any step k of a perturbed Lanczos recurrence,*

$$\langle \mathbf{q}_{k+1}, \mathbf{y}_i \rangle = \epsilon \frac{\|\mathbf{A}\|\gamma_{k,i}}{\beta_{k,i}}, \qquad \beta_{k,i} = \beta_k \mathbf{e}_k^T \mathbf{s}_i, \qquad \gamma_{k,i} \leq \mathcal{O}(J)$$

and

$$|1 - \|\mathbf{y}_i\|^2| \leq \frac{J(J-1)\gamma\epsilon}{\mu_i}, \qquad \mu_i = \min_{j \neq i} \frac{|\theta_i - \theta_j|}{\|\mathbf{A}\|}, \qquad \gamma \leq \mathcal{O}(J).$$

The first condition implies that a Lanczos vector $\mathbf{q}_{k+1}$ is approximately orthogonal to a Ritz vector $\mathbf{y}_i$ as long as $\|\mathbf{y}_i\|$ and $|\beta_{k,i}|$ are not too small. The second condition implies that the size of $\|\mathbf{y}_i\|$ will be near one (not small) provided that μ_i is too small; i.e. that θ_i is well separated from other Ritz values. We note that

$$|\beta_{k,i}| = |\beta_k \mathbf{e}_k^T \mathbf{s}_i| = \|\mathbf{A}\mathbf{y}_i - \theta_i \mathbf{y}_i - \mathbf{F}_k \mathbf{s}_j\| / \|\mathbf{q}_{k+1}\| \approx \|\mathbf{A}\mathbf{y}_i - \theta_i \mathbf{y}_i\|$$

where the last approximate equality follows by assuming $\|\mathbf{q}_{k+1}\| \approx 1$ and $\mathbf{F}_k \approx 0$. Thus, $|\beta_{k,i}|$ will not be small unless the Ritz value θ_i has converged to an eigenvalue of $\mathbf{A}$.

The previous theorm applies to any perturbed Lanczos recurrence. However, there is also theory about the output of Lanczos which gurantees the assumptions made above.

Theorem 7 ([Pai76]). *Apply* Lanczos *to $\mathbf{A}$ for k steps to obtain a perturbed recurrence of the form eq. (7). Then,*

$$\begin{aligned} |\|\mathbf{q}_j\| - 1| &\leq \epsilon_0 \\ \|\mathbf{f}_j\| &\leq \|\mathbf{A}\|\epsilon_1 \\ \beta_{J+1}|\langle \mathbf{q}_J, \mathbf{q}_{J+1} \rangle| &\leq 2\|\mathbf{A}\|\epsilon_0 \end{aligned}$$

where

$$\epsilon_0 := (n+4)\epsilon, \qquad \epsilon_1 := (7 + m\||\mathbf{A}|\|/\|\mathbf{A}\|)\epsilon$$

and m is the maximum number of nonzeros per row of $\mathbf{A}$. It is assumed that $\epsilon_0 < 1/12$ and $k(3\epsilon_0 + \epsilon_1) < 1$.

Theorem 8 ([Pai80]). *Apply* Lanczos *to $\mathbf{A}$ for k steps to obtain a perturbed recurrence of the form eq. (7). Then*

$$\lambda(\mathbf{T}) \subset \mathcal{I}_\eta(\mathbf{A}), \qquad \eta = k^{5/2}\|\mathbf{A}\|\epsilon_2, \qquad \epsilon_2 := \sqrt{2}\max\{6\epsilon_0, \epsilon_1\}.$$

Theorem 9 (Paige take from [Gre89, Section 4]). *Let $\mathbf{T}_k$ be the tridiagonal matrix generated at step $k > n$ of a perturbed Lanczos recurrence of the form eq. (7) with $\beta_k = 0$. Then every eigenvalue of $\mathbf{T}_k$ lies within,*

$$\sigma\, k\, \xi\, \|\mathbf{A}\|$$

of an eigenvalue of $\mathbf{A}$ if the perturbation terms $\mathbf{f}_j := [\mathbf{F}_k]_{:,j}$ satisfy,

$$\max_j \|\mathbf{f}_j\| \leq \xi\, \|\mathbf{A}\|$$

where σ is a constant independent of ξ, n, k, and $\|\mathbf{A}\|$.

Theorem 10 ([MMS18, Theorem 6.2]). *Run* Lanczos *in finite precision on $\mathbf{A} \in \mathbb{R}^{n\times n}$ symmetric and $\mathbf{b} \in \mathbb{R}^n$ for k steps. Pick $\beta \in (0,1]$, $\eta = nk\|\mathbf{A}\|\epsilon/\beta$, and suppose $|f(x)| \leq C$ for all $x \in \mathcal{I}_\eta(\mathbf{A}) = [\lambda_{min}(\mathbf{A}) - \eta, \lambda_{max}(\mathbf{A}) + \eta]$. Then*

$$\frac{\|f(\mathbf{A})\mathbf{b} - \mathbf{x}_k\|}{\|\mathbf{b}\|} \leq 7k \left(\min_{p \in \mathcal{P}_k} \left\{ \max_{z \in \mathcal{I}_\eta(\mathbf{A})} |f(z) - p(z)| \right\} \right) + \epsilon\, C.$$

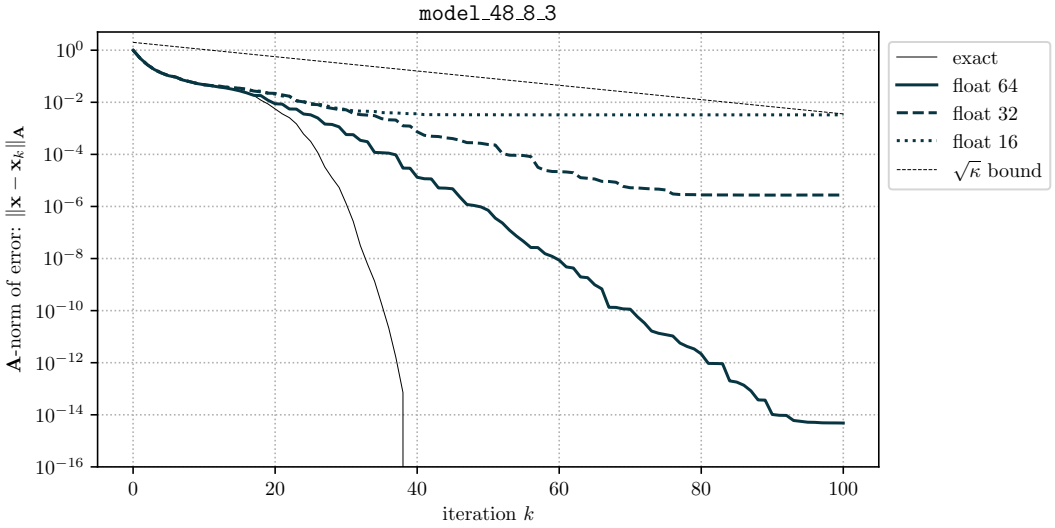

Figure 1: Convergence of finite precision conjugate gradient

3.2.2 Greenbaum's theory

The main result of Greenbaum's work on the Lanczos algorithm [Gre89] is demonstrating that a perturned Lanczos recurrence of the form eq. (7) can be extended to a perturbed Lanczos recurrence

$$\mathbf{A}\bar{\mathbf{Q}}_{N+m} = \bar{\mathbf{Q}}_{N+m}\bar{\mathbf{A}} + \bar{\mathbf{F}}_k$$

where

$$[\bar{\mathbf{Q}}_{N+m}]_{:,:k} = \mathbf{Q}_k, \qquad [\bar{\mathbf{A}}]_{:k,:k} = \mathbf{T}_k$$

and the eigenvalues of $\bar{\mathbf{A}}$ are near those of $\mathbf{A}$.

Applying the Lanczos algorithm with initial vector $\mathbf{e}_1$ to any tridiagonal matrix produces that same matrix. In particular, applying exact Lanczos to $\bar{\mathbf{A}}$ for k steps will produce the tridiagonal matrix $\mathbf{T}$. Thus, this highly non-trivial result allows a "backwards" type analysis for Lanczos and CG, allowing the finite precision behavior of CG and Lanczos to be related to a particular "nearby" exact arithmetic computation.

3.3 Conjugate Gradient

Like Lanczos, the behavior of CG in finite precision can be significantly different than in exact arithmetic. In particular, both the rate of convergence (the number of iterations to reach a given level of accuracy) and the maximal attainable accuracy of any CG implementation may be severely impacted by carrying out computations in finite precision. Both of these effects are clearly exhibited in fig. 1.

3.3.1 Rate of convergence

Most of what is known about the analysis on the rate of convergence of CG in finite precision is due to Greenbaum's analysis on perturbed Lanczos recurrences. In section 2.3 we showed

the equivalence of the Lanczos and conjugate gradient algorithms, in particular showing that the scaled residuals $\mathbf{q}_j := (-1)^j \mathbf{r}_j / \|\mathbf{r}_j\|$ satisfy a three term Lanczos recurrence. In finite precision this is no longer the case, but we can still define $\mathbf{q}_j$ in this way. Then, our (scaled) updated residuals $\mathbf{q}_j$ will satisfy a perturbed Lanczos recurrence of the form eq. (7).

If the sufficient conditions for Greenbaum's theory to apply are satisfied, then we can obtain a larger matrix $\bar{\mathbf{A}}$ whose eigenvalues are near those of $\mathbf{A}$. This extended matrix $\bar{\mathbf{A}}$ is tridiagonal, so exact CG applied to this matrix with right hand side $\mathbf{e}_1$ will produce the same tridiagonal matrix. Since the residual norms of a CG method are determined entirely by the entries of the tridiagonal matrix produced, this trivially implies that the residual norms of the original finite precision conjugate gradient method will match *exactly* those of exact CG applied to the larger matrix $\bar{\mathbf{A}}$.

The important fact is that if the eignevalues of $\bar{\mathbf{A}}$ are near those of $\mathbf{A}$, then the error norms can also be expected to approximately match. This is characterized more precisely in [Gre89, Theorem 3]. Interestingly, as discussed in section 5.2.4, the error norms of finite precision computations seem to match very closely with the exact error norms of CG applied to an extended matrix constructed using Greenbaum's approach, *even when the eigenvalues are not particualrly close*.

Roughly speaking, the analysis then shows that for some δ, which the analysis in [Gre89] defines more precisely, the error norms of the CG iterates in *finite precision* satisfy the relaxed minimax bound,

$$\frac{\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}}}{\|\mathbf{A}^{-1}\mathbf{b}\|_{\mathbf{A}}} \leq \min_{p \in \mathcal{P}_k^0} \left\{ \max_{x \in \lambda_\delta(\mathbf{A})} |p(x)| \right\}.$$

This can be viewed as the analogue to theorem 3. With this in mind it is clear that the Chebyshev error bound from corollary 1 holds with $\kappa(\mathbf{A})$ replaced with $\kappa_\delta(\mathbf{A})$.

We note that a similar result to the resulting finite precision Chebyshev error bound can more easily be obtained by applying theorem 10 to the special case $f(x) = 1/x$ which gives the bound

$$\frac{\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} \leq 7k \cdot 2 \left(\frac{\sqrt{\kappa_\eta(\mathbf{A})} - 1}{\sqrt{\kappa_\eta(\mathbf{A})} + 1} \right)^k + \epsilon\, C.$$

3.3.2 Final accuracy

In exact arithmetic, the CG method will find the exact solution in at most n steps. This is a trivial result of the Cayley–Hamilton Theorem, which states that a matrix satisfies its own characteristic polynomial.

However, in finite precision this may not be the case. Indeed, both $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$ and $\|\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k\|_{\mathbf{A}}$ are observed to eventually stagnate. The level at which these quantities stop decreasing is referred to as the maximal accuracy, and may be dependent on the problem as well as the specific implementation. For instance, it is well known that certain high performance varaints, such as GV-CG, fail to reach a sufficiently accurate solution on some problems.

The maximal attainable accuracy of a CG implementation in finite precision is typically analyzed in terms of the residual gap

$$\Delta_{\mathbf{r}_k} := (\mathbf{b} - \mathbf{A}\mathbf{x}_k) - \mathbf{r}_k.$$

This expression was introduced by Greenbaum in [Gre89, Theorem 2] and studied in further detail in [Gre97a; SVF94]. Analysis of the final accuracy of a variant typically involves computing an explicit formula for the residual gap in terms of roundoff errors introduced in a finite precision algorithm. The form of these expressions gives some indication for how large the residual gap can be expected to become. For instance, in the case of HS-CG we can observe that

$$\begin{aligned}\Delta_{\mathbf{r}_k} &= (\mathbf{b} - \mathbf{A}(\mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1} + \delta_{\mathbf{x}_k})) - (\mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1} + \delta_{\mathbf{r}_k}) \\ &= \Delta_{\mathbf{r}_{k-1}} - \mathbf{A}\delta_{\mathbf{x}_k} - \delta_{\mathbf{r}_k} + \alpha_{k-1}\delta_{\mathbf{s}_{k-1}} \\ &\vdots \\ &= \Delta_{\mathbf{r}_0} - \sum_{j=1}^{k} \left(\mathbf{A}\delta_{\mathbf{x}_j} + \delta_{\mathbf{r}_j} - \alpha_{j-1}\delta_{\mathbf{s}_{j-1}}\right)\end{aligned}$$

where $\delta_{\mathbf{x}_k}$, $\delta_{\mathbf{r}_k}$, and $\delta_{\mathbf{s}_k}$ represent errors made while computing $\mathbf{x}_k$, $\mathbf{r}_k$, and $\mathbf{s}_k$ in finite precision.

For many variants, such as HS-CG, it is observed experimentally that the size of the updated residual $\mathbf{r}_k$ decreases to much lower than the machine precision, even after the true residual has stagnated. As a result, the size of the residual gap $\Delta_{\mathbf{r}_k}$ can be used to estimate of the size of the smallest true residual which can be attained in finite precision, thereby giving an estimate of the accuracy of the iterate $\mathbf{x}_k$. Similar analyses have been done for a three-term CG variant [GS00], and for CG-CG, GV-CG and other pipelined conjugate gradient variants [Coo+18; Car+18]. However, for some variants such as GV-CG, the updated residual $\mathbf{r}_k$ may not decrease to well below machine precision, so some care must be taken when analyzing the final accuracy of such varaints.

4 Past efforts towards reducing communication

[On large machines, the cost of moving data dominates the costs of floating point arithmetic. From the presentation of HS-CG in algorithm 2, it is clear that the algorithm is highly sequential. Specifically, in each iteration, the preconditioning step, the first inner product, the matrix vector product, and the final inner product must occur one after the other. Thus, the communication costs for each of these steps will add to the time it takes to compute a single iteration.

Inner products. Each inner product requires a global synchronization of all nodes involved in the computation. This means that the next computation cannot begin until all nodes have completed their computations and the results are aggregated. Such synchronizations are typically the most expensive component of each iteration on parallel machines [Dem+08; VDGQO08].

Matrix vector product. While a matrix vector product typically requires more floating point operations than a single inner product, if $\mathbf{A}$ has an exploitable structure (i.e. $\mathbf{A}$ is sparse, is a DFT operator, etc.) the communication costs can be much lower [VDGQO08]. In fact, even in the extreme case that the matrix $\mathbf{A}$ is dense, computing the matrix vector product still only requires one global synchronization, since each row of the output can be computed simultaneously and independently.

If $\mathbf{A}$ has a reasonable sparsity strucutre, it is possible to efficintly compute "matrix powers" such as $[\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots, \mathbf{A}^k\mathbf{x}]$ simultaneously and much more efficiently than it would be to

naievely compute each vector [Dem+08; Hoe10]. This is one of the main advantages of s-step methods.

Preconditioning. Traditional preconditioners based on incomplete factorizations are applied through a triangular solve. However, even if the triangular factors are sparse so that they require only $\mathcal{O}(n)$ floating point operations, the operations for the solved are mostly sequential and inherently difficult to parallelize. For this reason, there has been a range of work on finding classes of preconditioners which can be efficiently applied on parallel machines; for an overview see [Saa03].⌋

Roughly speaking, the conjugate gradient varaints meant to reduce communication fall into two categories, (i) communication hiding, and (ii) communication avoiding. Communicating hiding varaints introduce auxillary vectors to allow expensive computations to be overlapped. On the other hand, communication avoiding methods expand the Krylov subspace s-dimensions at a time, reducing the number of synchronizatinos by a factor of $\mathcal{O}(s)$.

In the remainder of this section we provide a very breif overview of these methods.

4.1 Communication hiding methods

Much of the early work on reducing communication costs in the CG method focused on computing the two inner products of HS-CG simultaneously [Saa85; Chr87; Meu87; Saa89; CG89]. This is typically done by substituting recurrences for the vectors involved in an inner product, and expanding. The resulting expresion, which is equivalent in exact arithmetic, involves the inner products of vectors known earlier in the iteration and therefore can be overlapped with the other inner product. Two such varaints are CG-CG and M-CG from [CG89] and [Meu87] respectively.

More recent work has focused on overlapping the matrix vector product and preconditioning steps with global reductions. Through the introduction of auxillary vectors, the well known "pipelined" CG (GV-CG) allows the matrix vector prodcut to be computed at the same time as both inner products, resulting in only one global reduction per iteration [GV14]. Another variant, which overlaps the matrix product with one inner product and preconditinoing with other inner product, has also been considered [EG16].

Even more recently, there has also been a variety of work on "deep pipelined" variants, which allow multiple matrix products to be overlapped with a single global reduction, reducing the cost per iteration further in the case that the matrix vector product is cheap relative to the global reduction [CCV19b].

In general, the introduction of auxillary vectors and reordering of computations within an iteration means that the numerical properties of such variants will be impacted. Indeed, pipelined CG methods are well known to suffer from both a loss of accuracy and delay of convergence. Various approaches such as residual replacement [Coo+18] and shifted recurrences [CCV19a] have been suggested to improve numerical performace.

However, these approaches all rely on the use of heuristics or on user supplied hyperparameters not required by HS-CG. Thus, their use as general purpose solvers may be limited. The predict-and-recompute variants discussed further in section 5.1.1 address this concern.

For a more comprehensive overview of such methods we refer readers to [Car+18].

4.2 Communication avoiding (s-step) methods

The other main approach to reducing communication costs are the so called s-step methods [Chr87; CG89; Car15] which work by expanding the Krylov subspace s steps at a time. While there are multiple different approaches, the main idea is to construct a basis for Krylov subspace of dimension s which will contain the iterates for the next s steps. This allows the communication which would be associated with these s iterations in HS-CG to occur at once, effectively reducing the communoication costs by a factor of $\mathcal{O}(s)$.

However, like communication hiding methods, s-step methods encounter a range of numerical difficultites. These are primarily associated with expanding the Krylov subsapace with an ill conditioned basis. Approaches to dealing with these issues include residual replacement [CD14], and the use of different bases for expanding the Krylov subspace[PR12], both of which may occur adaptively [Car19]. However, these approaches also tend to rely on heuristics and user supplied parameters. For an overview of s-step methods we refer readers to [Car15].

5 Thesis work

During the remainder of my time in the PhD program I intend to work on a variety projects in the intersection of numerical linear algebra and computational science.

The primary aim of my thesis will be to develop new conjugate gradient and Lanczos type algorithms which are (i) fast in practice, and (ii) demonstrate provable behavior. However, it should be noted that the second goal is quite ambitious in that the conjugate gradient and Lanczos algorithms are notoriously difficult to analyze in finite precision. Thus, a more realistic outcome may be the development of algorithms which are stable in practice, even if not provably so. The secondary aim of my thesis is to analyze some of the various phenomenon relating to the conjugate gradient algorithm in finite precision in general which have not yet been explained.

5.1 Design of new methods

5.1.1 Predict-and-recompute methods

The idea to recompute a quantity in a CG iteartion first appears, to the best of our knowledge, in [Meu87].

In [CC20] we (myself and Erin Carson) introduce several "predict-and-recompute" type conjugate gradient variants, which decrease the runtime per iteration by overlapping global synchronizations and matrix vector products. The main contribution is the idea to use recursively updated quantities as a predictor for their true values, and then recompute them directly at a later point in the iteration when they can be overlapped with other work. This allows the same communication complexity as GV-CG, but improves the numerical properties.

We provide a rounding error analysis of the predict-and-recompute variants in order to help explain both the improved rate of convergence and final accuracy. The form of the residual gap as well as the perturbations to the Lanczos recurrence are "nicer" than those for GV-CG which supports the observation that these varaints have better numerical properties than GV-CG on *every* problem we tested; see appendix A.2 for more details.

One of the primary bennefits of the methods we introduce is that they require exactly the

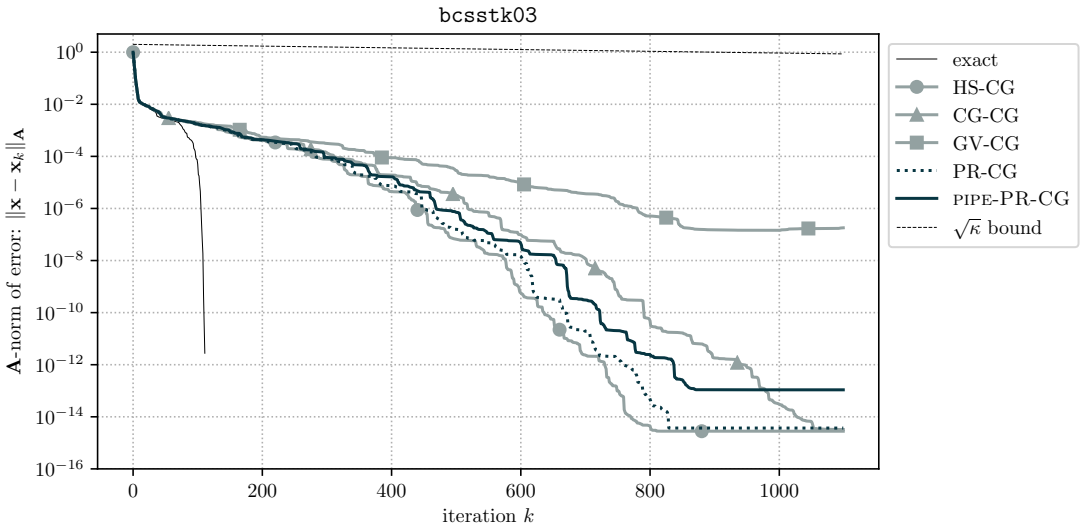

Figure 2: Convergence of finite precision conjugate gradient

same inputs as HS-CG in order to achieve good convergence. This is in contrast to other high performance CG methods which rely on heuristics or optimizing hyperparameters in order to prevent numerical issues. The convergence of the new variants (dark blue) is compared to previously known variants in fig. 2.

The most significant limitation of the predict-and-recompute variants from [CC20] is that they only improve the time per iteration by a factor of at most three. As discussed in section 4, two approaches which improve allow for greater speedups are s-step methods and deep pipelined methods. However, both approaches suffer from numerical issues when the hyperparameters controlling the speedup are increased too much.

There is some hope that predict-and-recompute ideas can be incorporated into s-step methods. For instance, in s-step CG a Gram matrix is constructed to allow the inner loop to work with vectors of size $\mathcal{O}(s)$. The coefficients in this inner loop are equivalent to those in exact CG. Thus, when the inner loop has completed, it may be possible to recompute the iterate and residual using a matrix powers kernel with the polynomials corresponding to these coefficients. Computing these vectors directly in this way may reduce some of the rounding errors associated with the ill-conditioned Gram matrix.

5.1.2 Reorthogonalization in HPC context

In their seminal 1952 paper [HS52], Hestenes and Stiefel suggested that the updated CG residuals should be orthogonalized against all previous residuals in order to prevent a loss of orthogonality. In a HPC context, these orthogonalizainons can occur simultaneously using a regular Gram-Schmidt type approach meaning the number of global communications can be kept the same as HS-CG. The downside of this approach is that the algorithm would then require $\mathcal{O}(n^2)$ storage and $\mathcal{O}(nk)$ work per iteration.

This drawback lead to the notion of partial reorthogonalization [PS79]. Such approaches attempt to identify some smaller subset of vectors to save for reorthogonalization using Paige's analysis about the direction in which orthogonality is lost in a Lanczos recurrence.

Such work may still be applicable to HPC. Indeed, all previous residuals can be simultaneously orthogonalized against using only a single global communication.

Reorthogonalization is also theoretically compatibale with s-step methods, which would allow for the same $\mathcal{O}(s)$ speedup.

5.2 Numerical analysis

CG in finite precision exhibits a range of interesting behaviors. Many of these phenomena are consistently observed in practice, but proofs remain elusive.

5.2.1 Analysis of rate of convergence

In [Gre89] an analogy was shown between a perturbed Lanczos computation satisfying certain condtions and a specific larger exact Lanczos recurrence. However, for this analysis to apply to CG, it must be proved that the perturbed Lanczos recurrence obtained from the normalized CG residuals satisfies the given conditions. This has not been proved for any commonly used variants.

In [GLC19], we (Anne Greenbaum, Hexuan Liu, and myself) analyze the rate of convergence of HS-CG, CG-CG, and GV-CG in terms of the conditions required by the analysis in [Gre89]. In this paper we explain numerically why the rate of convergence of all variants is similar on some problems but different on others. The conclusion is that on problems where the minimax polynomial on small intervals about the eigenvalues depends heavily on the size of the interval, that the different variants will behave differently, but on problems where the size of the intervals do not affect the minimax polynomial, the variants behave similarly.

We additionally provide a rounding error analysis of the amount by which the three term symmetric Lanczos recurrence in eq. (7) fail to be satisfied in finite precision for HS-CG, CG-CG, and GV-CG. The form of these expressions gives some intuition for why GV-CG is observed not to satisfy the three term recurrence as well as HS-CG and CG-CG, as well as some indication that the rate of convergence of GV-CG on hard problems may lower than the other variants.

5.2.2 Norm of updated residuals

In many variants it is observed that the norm of the updated residual $\|\mathbf{r}_k\|$ decreases well past machine precision. However, in GV-CG it is easy to observe that this is not the case. Since the study of the final accuracy of a method done in terms of the residual gap typically assumes that the updated residual eventually is small [Gre97a; Car+18; Coo+18], proving that the updated residual of a given variant does indeed become very small is of significant theoretical interest.

To the best of our knowledge this has not been proved for any varaint based on short recurrences.

5.2.3 Successive orthogonality of CG residuals

Orthogonality of successive vectors is a classic result for Lanczos (theorem 7) [Pai76]. A similar result has been proved for the s-step Lanczos method provided that the condition number of the basis used to expand the Krylov subspace is not too large [CD15]. The condition

number of the standard monomial basis grows exponentially with s, which means that either s must be kept small or a different basis must be used. In the context of CG there has been a range of work on approaches to selecting the paramter s as well as an appropraite basis [Car18; Car19].

In [GLC19] we observed numerically that HS-CG, CG-CG, and GV-CG do keep the successive residuals nearly orthogonal, however this has not be proved for any of these variants. As successive orthogonality of residuals is one of the necessary conditions for the analysis from [Gre89], this is of theoretical interest.

Orthogonality of successive residuals was shown for a modified version of HS-CG with coefficients designed to enforce this orthogonality [Gre89]. However, as this varaint is not commonly used in practice this result is mostly of academic interest. Expressions for the orthogonality of successive residuals of HS-CG are derived in [Meu06, Proposition 5.19], however they are messy and it is not clear that they of practical use or can easily fit into the framework of [Gre89].

A derivation of simple bounds similar to those of theorem 7 is of both theoretical and practical interest.

5.2.4 CG error for perturbed Lanczos

Given any perturbed Lanczos recurrence, the method from [Gre89] for constructing the larger matrix $\bar{\mathbf{A}}$ can be applied. If the perturbed Lanczos recurrence does not satisfy the conditions of Greenbaum's analysis then there are no gurantees about how the eigenvalues of $\bar{\mathbf{A}}$ relate to those of $\mathbf{A}$.

In [GLC19] we observed that if this method was applied to the tridiagonal matrix produced by a finite precision CG variant that the $\bar{\mathbf{A}}$-norm of the error of exact CG applied to the extended matrix $\bar{\mathbf{A}}$ matched nearly identialy to the $\mathbf{A}$-norm of the error of the original finite precision computation for all varaints. While it is obvious that the residal norms will agree, since the eigenvalues of $\mathbf{A}$ and $\bar{\mathbf{A}}$ ae not necessarily close, it is interesting that the error norms also agree.

The analsis in recent work showing that the residual norms and error norms can be simultanously prescribed may be of use [Meu19]. In particular, it seems possible that the analysis shows that the error norm at step j is more or less determined by the tridiagonal matrix generated at step J if J is enough larger than j. This may also simplify the proof of Theorem 3' in [Gre89].

6 Future plans

6.1 Other research during the PhD

In addition to my thesis work, I hope to develop techniques for analyzing the finite precision behaviour of randomized algorithms. Algorithms involving randomness have become commonplace, and in practice these algorithms are often run in finite precision. As a result, some of their theoretical properties based on the use of exact samples from continuous distributions, can no longer be guaranteed. Even so, many randomized algorithms appear to perform as well in practice as predicted by theory [HMT11], suggesting that errors resulting from sampling such distributions in finite precision are negligible. At the same time, especially in the

case of Monte Carlo simulations, it is not typically clear how to differentiate the possible effects of rounding errors from the effects of sampling error. In fact, in many areas (such as the numerical solution to stochastic differential equations) this problem is typically addressed by ignoring the effects of rounding errors under the assumption that they are small [KP92]. However, with the recent trend towards lower precision computations, and with the massive increase in the amount of data available, understanding the effects of rounding errors on random variables and the interplay between rounding and sampling error has become increasingly important.

Broadly speaking, the goal of such projects is to provide provable bounds for numerical algorithms involving randomness when they are run in finite precision.

6.2 Other goals during the PhD

In addition to the concrete goals of my thesis, I have a few "soft goals" for the research which I will produce during the duration of my time here. Specifically, I intend to (i) collaborate with a range of people, both within and outside of the department, (ii) facilitate the reproducibility of my work and promote open science, and (iii) disseminate scientific research to a broader audience.

6.3 Career goals

My goal is to become a professor. Roughly speaking, in order of importance, I would like to work at an instutution (i) in a city (3M+), (ii) with a strong research reputation, and (iii) emphasis on undergraduate education; e.g. Tufts, Northeastern, Boston College, etc. Prior to this, I hope to do a postdoc at large east coast school (e.g. NYU, MIT, Columbia) or abroad (Canada, UK, or China).

6.4 Career questions

I outlined my intentions for my thesis in section 5. However, as a result of the time constraints associated with a PhD, the suggested thesis work is somewhat limited in scope. In this section I provide an overview of some of the more ambitious and open ended problems which I hope to study over the course of my academic career.

What effect do rounding errors have on algorithms involving randomness?

Alorithms involving randomness have become commonplace. However, their behavior in finite precision has not been well studied. While traditional deterministic rounding error analysis can be applied, it seems reasonable that the randomness involved in the algorithms could lead to stronger bounds or simpler analysis.

Can finite precision be viewed as an exact computation in some space with an algebraic structure?

The theory of the Lanczos and conjugate gradient algorithms in exact arithmetic relies heavily on certain algebraic properties such as orthogonality. However, as we have discussed, these properties may not hold in finite precision, in which case exact arithmetic theory is longer applicable. However, Greenbaum showed an analogy between finite precision Lanczos (and conjugate gradient) computations, and a certain exact precision computation [Gre89]. Can this type of analogy be constructed for any sequence of linear algebra operations?

Can we develop algorithms meant to perform well in finite precision without starting with an algorithm which is meant to work well in exact precision?

In exact arithmetic, the conjugate gradient algorithm uses very little work/memory to minimize the A-norm of the error over successive Krylov subspaces. However, in finite precision it does not do a very good job of this. On the other hand, while Chebyshev iteration is much less ambitious than CG in exact arithmetic, it works almost exactly as well in finite precision as it does in exact arithmetic. Perhaps it is possible to develop new algorithms which lie in the middle of these extremes; i.e. algorithms which have moderately ambitious goals in exact precision, but which are able to get closer to those goals in finite precision. ?

7 Acknowledgements

I'm extremely greatful to my advisor, Anne Greenbaum, for providing persistent enthusiasm and guidance during my time in the department.

I would also like to thank Erin C. Carson for detailed feedback on my work which has led to what I hope will be an ongoing collaboration.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[CC20] Tyler Chen and Erin C. Carson. "Predict-and-recompute conjugate gradient variants". In: (2020). arXiv: 1905.01549 [cs.NA].

[CCV19a] Siegfried Cools, Jeffrey Cornelis, and Wim Vanroose. "Numerically Stable Recurrence Relations for the Communication Hiding Pipelined Conjugate Gradient Method". In: *IEEE Transactions on Parallel and Distributed Systems* 30.11 (2019), pp. 2507–2522.

[CCV19b] Jeffrey Cornelis, Siegfried Cools, and Wim Vanroose. *The Communication-Hiding Conjugate Gradient Method with Deep Pipelines*. 2019.

[CD14] Erin C. Carson and James Demmel. "A Residual Replacement Strategy for Improving the Maximum Attainable Accuracy of s-Step Krylov Subspace Methods". In: *SIAM Journal on Matrix Analysis and Applications* 35.1 (2014), pp. 22–43.

[CD15] Erin C. Carson and James W. Demmel. "Accuracy of the s-Step Lanczos Method for the Symmetric Eigenproblem in Finite Precision". In: *SIAM Journal on Matrix Analysis and Applications* 36.2 (2015), pp. 793–819.

[CG89] A.T. Chronopoulos and Charles William Gear. "s-step iterative methods for symmetric linear systems". In: *Journal of Computational and Applied Mathematics* 25.2 (1989), pp. 153 –168.

[Car+18] Erin C. Carson et al. "The Numerical Stability Analysis of Pipelined Conjugate Gradient Methods: Historical Context and Methodology". In: *SIAM Journal on Scientific Computing* 40.5 (2018), A3549–A3580.

[Car15] Erin C. Carson. "Communication-avoiding Krylov subspace methods in theory and practice". PhD thesis. UC Berkeley, 2015.

[Car18] Erin C. Carson. "The Adaptive s-Step Conjugate Gradient Method". In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (2018), pp. 1318–1338.

[Car19] Erin C. Carson. *An Adaptive s-step Conjugate Gradient Algorithmwith Dynamic Basis Updating*. 2019.

[Chr87] A. Chronopoulos. "A Class of Parallel Iterative Methods Implemented on Multiprocessors". PhD thesis. Chicago, IL, USA, 1987.

[Coo+18] Siegfried Cools et al. "Analyzing the Effect of Local Rounding Error Propagation on the Maximal Attainable Accuracy of the Pipelined Conjugate Gradient Method". In: *SIAM Journal on Matrix Analysis and Applications* 39 (Mar. 2018), pp. 426–450.

[Dem+08] James Demmel et al. "Avoiding communication in sparse matrix computations". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. 2008, pp. 1–12.

[EG16] Paul R. Eller and William Gropp. "Scalable Non-blocking Preconditioned Conjugate Gradient Methods". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '16. Salt Lake City, Utah: IEEE Press, 2016, 18:1–18:12. ISBN: 978-1-4673-8815-3.

[GLC19] Anne Greenbaum, Hexuan Liu, and Tyler Chen. *On the Convergence Rate of Variants of the Conjugate Gradient Algorithm in Finite Precision Arithmetic*. 2019.

[GS00] M. Gutknecht and Z. Strakos. "Accuracy of Two Three-term and Three Two-term Recurrences for Krylov Space Solvers". In: *SIAM Journal on Matrix Analysis and Applications* 22.1 (2000), pp. 213–229.

[GV14] Pieter Ghysels and Wim Vanroose. "Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm". In: *Parallel Computing* 40.7 (2014), pp. 224 –238.

[Gre79] A. Greenbaum. "Comparison of splittings used with the conjugate gradient algorithm". In: *Numerische Mathematik* 33.2 (1979), pp. 181–193.

[Gre89] Anne Greenbaum. "Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences". In: *Linear Algebra and its Applications* 113 (1989), pp. 7 –63.

[Gre97a] Anne Greenbaum. "Estimating the Attainable Accuracy of Recursively Computed Residual Methods". In: *SIAM Journal on Matrix Analysis and Applications* 18.3 (1997), pp. 535–551.

[Gre97b] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM review* 53.2 (2011), pp. 217–288.

[HS52] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*. Vol. 49. NBS Washington, DC, 1952.

[Hoe10] Mark Hoemmen. "Communication-avoiding Krylov subspace methods". PhD thesis. UC Berkeley, 2010.

[Hpc] *Novenber 2019 HPCG Results*. 2019. URL: https://www.hpcg-benchmark.org/custom/index.html?lid=155&slid=302.

[KP92] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin Heidelberg, 1992.

[Ken19] Graham Kendall. *Your Mobile Phone vs. Apollo 11's Guidance Computer*. 2019. URL: https://www.realclearscience.com/articles/2019/07/02/your_mobile_phone_vs_apollo_11s_guidance_computer_111026.html.

[Lan50] Cornelius Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: 1950.

[MMS18] Cameron Musco, Christopher Musco, and Aaron Sidford. "Stability of the Lanczos Method for Matrix Function Approximation". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 1605–1624.

[Meu06] Gérard. Meurant. *The Lanczos and Conjugate Gradient Algorithms*. Society for Industrial and Applied Mathematics, 2006.

[Meu19] Gérard Meurant. "On prescribing the convergence behavior of the conjugate gradient algorithm". In: *Numerical Algorithms* (2019).

[Meu87] Gérard Meurant. "Multitasking the conjugate gradient method on the CRAY X-MP/48". In: *Parallel Computing* 5.3 (1987), pp. 267 –280.

[Orn] *ORNL Launches Summit Supercomputer*. 2018. URL: https://www.ornl.gov/news/ornl-launches-summit-supercomputer.

[PR12] Bernard Philippe and Lothar Reichel. "On the generation of Krylov subspace bases". In: *Applied Numerical Mathematics* 62.9 (2012). Numerical Analysis and Scientific Computation with Applications (NASCA), pp. 1171 –1186.

[PS79] Beresford Neill Parlet and David St. Clair Scott. "The Lanczos algorithm with selective orthogonalization". In: *Mathematics of Computation* 33.145 (1979), pp. 217–238.

[Pai76] Chris Conway Paige. "Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix". In: *IMA Journal of Applied Mathematics* 18.3 (Dec. 1976), pp. 341–349.

[Pai80] Christopher Conway Paige. "Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem". In: *Linear Algebra and its Applications* 34 (1980), pp. 235 –258.

[Ros83] John Van Rosendale. "Minimizing Inner Product Data Dependencies in Conjugate Gradient Iteration." In: *ICPP*. 1983.

[SG06] Robert Strzodka and Dominik Goddeke. "Pipelined Mixed Precision Algorithms on FPGAs for Fast and Accurate PDE Solvers from Low Precision Components". In: *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. 2006, pp. 259–270.

[SVF94] Gerard Sleijpen, Henk A. van der Vorst, and Diederik R. Fokkema. "BiCGstab(l) and other hybrid Bi-CG methods". In: *Numerical Algorithms* 7.1 (1994), pp. 75–109.

[Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003.

[Saa85] Yousef Saad. "Practical Use of Polynomial Preconditionings for the Conjugate Gradient Method". In: *SIAM Journal on Scientific and Statistical Computing* 6.4 (1985), pp. 865–881.

[Saa89] Yousef Saad. "Krylov Subspace Methods on Supercomputers". In: *SIAM Journal on Scientific and Statistical Computing* 10.6 (1989), pp. 1200–1232.

[VDGQO08] Robert A Van De Geijn and Enrique S Quintana-Ortí. *The science of programming matrix computations*. Citeseer, 2008.

A Supplementary Materials

A.1 On the convergence rate of variants of the conjugate gradient algorithm in finite precision arithmetic (CS.NA 1905.05874)

Abstract

We consider three mathematically equivalent variants of the conjugate gradient (CG) algorithm and how they perform in finite precision arithmetic. It was shown in [*Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Lin. Alg. Appl., 113 (1989), pp. 7-63] that under certain conditions involving local orthogonality and approximate satisfaction of a recurrence formula, that *may* be satisfied by a finite precision CG computation, the convergence of that computation is like that of exact CG for a matrix with many eigenvalues distributed throughout tiny intervals about the eigenvalues of the given matrix. We determine to what extent each of these variants satisfies the desired conditions, using a set of test problems, and show that there is significant correlation between how well these conditions are satisfied and how well the finite precision computation converges before reaching its ultimately attainable accuracy. We show that for problems where the interval width makes a significant difference in the behavior of exact CG, the different CG variants behave differently in finite precision arithmetic. For problems where the interval width makes little difference or where the convergence of exact CG is essentially governed by the upper bound based on the square root of the condition number of the matrix, the different CG variants converge similarly in finite precision arithmetic until the ultimate level of accuracy is achieved, although this ultimate level of accuracy may be different for the different variants.

A.2 Predict-and-recompute conjugate gradient variants (CS.NA 1905.01549)

Abstract

The standard implementation of the conjugate gradient algorithm suffers from communication bottlenecks on parallel architectures, due primarily to the two global reductions required every iteration. In this paper, we introduce conjugate gradient variants which decrease the runtime per iteration by overlapping global synchronizations, and in the case of our pipelined variants, matrix vector products. Through the use of a predict-and-recompute scheme, whereby recursively-updated quantities are first used as a predictor for their true values and then recomputed exactly at a later point in the iteration, our variants are observed to have convergence behavior nearly as good as the standard conjugate gradient implementation on every problem we tested. We provide a rounding error analysis which provides insight into this observation. It is also verified experimentally that our variants do indeed reduce the runtime per iteration in practice, and that they scale similarly to previously-studied communication hiding variants. Finally, because our variants achieve good convergence without the use of any additional input parameters, they have the potential to be used in place of the standard conjugate gradient implementation in a range of applications.

A.3 An Introduction to Modern Analysis of the Conjugate Gradient Algorithm in Exact and Finite Precision

This document is intended to explain some of the important ideas relating to high performance conjugate gradient variants to a broader audience. An online version of this document can be found at: https://chen.pw/research/cg.