SUBSPACE RECURSIVE PROJECTION METHOD AND THEORY

*

Abstract. This paper improves the Recursive Projection Method (RPM) as first discussed in [1]. In particular, we will present a subspace version that will effectively utilize parallelism as well as conclude with some theoretical results regarding an equivalent preconditioner expression for RPM that is useful for nested inner-outer preconditioner analysis and convergence rate results ([11]). Furthermore, we include a discussion and suggestions for the heretofor neglected coupling factor in RPM.

Key words. Preconditioning, Invariant subspace, Deflation, Richardson

AMS subject classifications. 65F10, 65F08, 65B99, 65N12, 65Y05

1. Introduction.

Computational techniques for solving linear system of equations such as $Ay = b$, for $A \in \Re^{n\times n}, b \in \Re^{n\times 1}$ are frequently highly dependent on the spectra of the initial matrix A or various consequent matrix formulations such as the iteration matrix. Many theoretical results that compute the solution to a linear system even necessitate some *a priori* bound of the spectral radius for either the originating matrix A or resultant matrices utilized implicitly or explicitly during the course of many different numerical computations.

One possible method, explored initially by [19] and finally in the form of the Recursive Projection Method (RPM) in [1], uses some form of deflation. Deflation strategies allow us to manipulate eigenvalues to improve overall performance in solving linear systems while at the same time cautiously exempting oneself from the prohibitive costs of direct eigenvalue computation.

RPM is in turn based on iterative strategies for solving linear systems, including one of the oldest: Richardson iteration [20]. It is based on the fact that the slow convergence (or even divergence) of Richardson's method is dependent entirely on the eigenvalues of highest modulus of its Jacobian operator. RPM approximates two spaces $\mathbf{P}$ and $\mathbf{Q}$ that correspond respectively to an unstable and stable space. The unstable space can then be deflated out in order to improve overall performance.

There exist similar strategies that apply deflation to GMRES or other Krylov methods [2, 5, 6, 15, 17, 18]. We note that RPM can be used as a preconditioning subroutine in such algorithms (similar to [7, 9, 10, 12, 14]), or in specific cases can show robustness akin to RPM [1].

However, RPM currently has issues that we hope to build upon in order to add to its own theoretical framework, but also to other algorithms that may call or nest it as an efficient dependency (for example, using preconditioned GMRES with RPM as the preconditioner). One of these issues with RPM is a lack of a parallel implementation. This can be an issue for many iterative algorithms that solve linear systems, as many iterative algorithms contain dependencies to previous iterations at each current iteration that make effective parallelism across iterates very difficult to achieve. We instead do this by introducing a few, strategically placed block vectors inside of the deflation process by rewriting the algorithm to allow the replacement of a power method lying at the heart of the original RPM algorithm with a subspace method.

*(SUBSPACERPM@OPTOUTOFDATABASES.33MAIL.COM). QUESTIONS, COMMENTS, OR CORRECTIONS TO THIS DOCUMENT MAY BE DIRECTED TO THAT EMAIL ADDRESS.

This improvement permits us a parameter by which we can now have an effective parallelization.

We describe a number of a number of possible theoretical improvements to the underlying convergence theory of RPM including an explicit description of the rate of convergence of RPM. In previous analyses of RPM, the theoretical dependence of the algorithm was traced from nonlinear stabilization procedures to the specific application of Richardson methods. This in turn created a simplified expression for the Jacobian of the algorithm [1]. However, this is not enough to help completely describe the convergence properties of RPM, and in order to explicitly describe the rate of convergence of RPM, we introduce a new preconditioner expression for RPM. This by itself is an important result, as we may now model RPM as a preconditioner, which further means that the possibility of analysis of RPM as a subroutine in other algorithms is now drastically simplified.

The rest of the paper is organized as follows. In section 2 we introduce and review the Recursive Projection Method (RPM). Then we provide new theoretical results and correct previous theoretical results in section 3, before introducing a new generalized convergence theorem. After this, we recall the implementation details and definitions in section 4, and use this to propose a parallel Recursive Projection Method. In section 5 we discuss numerical results, and finally we conclude in section 6.

2. The Recursive Projection Method.

The Recursive Projection Method is a general methods that attempts to enhance finding the fixed points of given functional F, solving the problem

$$y = F(y) : \Re^N \times \Re \to \Re^N \tag{2.1}$$

where F can be any continuous functional. Here, we will solely consider the fixed iteration functional based on the Richardson iteration. The Richardson iteration attempts to solve a linear system $Ay = b$ by splitting the matrix $A = M - N$. It can be expressed in the form of a fixed iteration functional by

$$F(y) = M^{-1}Ny + M^{-1}b \tag{2.2}$$

RPM also uses a stabilization procedure defined by deflating where the Jacobian of (2.2) (which in this case is simply $M^{-1}N$) has spectra less than 1 or greater than 1.

In particular, we let $\mathbf{P}$ be the *invariant* subspace of where $M^{-1}N$ has spectra > 1, $\mathbf{Q}$ be the orthogonal complement to this space (we do not consider the case where 1 is an eigenvalue) so that $\mathbf{P} + \mathbf{Q} = \Re^n$, P and Q be the projectors on to the spaces $\mathbf{P}$ and $\mathbf{Q}$ respectively, $p := Py, q := Qy$, then we denote

$$\begin{aligned} p = f(p,q) &:= PF(y) \\ q = g(p,q) &:= QF(y). \end{aligned} \tag{2.3}$$

where p, q are meant to express the vectors, and f, g the projection of the functional F. With this, we split the iteration so that on the 'good' subspace corresponding to $\mathbf{Q}$ we continue with a fixed point iteration (recalling that for Jacobian having spectra < 1 we are guaranteed convergence to a fixed point), and perform a modified chord method on the 'bad' subspace $\mathbf{P}$. This results in the following iteration:

$$\begin{aligned}(I - f_p^{(0)})(p^{(k+1)} - p^{(k)}) &= f(p^{(k)}, q^{(k)}) - p^{(k)} \\ q^{(k+1)} &= g(p^{(k)}, q^{(k)})\end{aligned} \tag{2.4}$$

(f_p is the derivative of f with respect to the subspace $\mathbf{P}$, and the 0 superscript denotes that we take the first iterate's approximation to the derivative, i.e., if $J^{(0)} := J(p^{(0)}, q^{(0)})$ is the Jacobian of F at $(p^{(0)}, q^{(0)})$, then $f_p^{(0)} = PJ^{(0)}P$).[19]

We now simplify by applying (2.2) to (2.4), performing the necessary algebraic simplifications, letting Z be an orthogonal basis for $\mathbf{P}$ of dimension m (so that $P = ZZ^T, I = Z^T Z$), $H := M^{-1}N$, noting that $\mathbf{P}$ was chosen invariant under H thus $QHP = 0$ (at the moment, this assumption will not be used until section 3), and letting $u =: Z^T y$. Using these definitions and notations, we may simplify the first equation of (2.4) as:

$$\begin{aligned}(I - PHP)(p^{(k+1)} - p^{(k)}) &= P(Hy^{(k)} + M^{-1}b) - p^{(k)} \\ (I - ZZ^T HZZ^T)(p^{(k+1)} - p^{(k)}) &= ZZ^T(Hy^{(k)} + M^{-1}b) - p^{(k)} \\ (I - ZZ^T HZZ^T)p^{(k+1)} &= ZZ^T(Hy^{(k)} + M^{-1}b) - ZZ^T HPp^{(k)} \\ (I - ZZ^T HZZ^T)p^{(k+1)} &= ZZ^T(Hy^{(k)} - Hp^{(k)} + M^{-1}b) \\ (Z^T - Z^T HZZ^T)p^{(k+1)} &= Z^T(Hq^{(k)} + M^{-1}b) \\ (I - Z^T HZ)Z^T y^{(k+1)} &= Z^T(Hq^{(k)} + M^{-1}b) \\ (I - Z^T HZ)u^{(k+1)} &= Z^T(Hq^{(k)} + M^{-1}b)\end{aligned} \quad . \tag{2.5}$$

Likewise for the second equation in (2.4):

$$\begin{aligned}q^{(k+1)} &= g(p^{(k)}, q^{(k)}) \\ q^{(k+1)} &= Q(Hy^{(k)} + M^{-1}b) \\ q^{(k+1)} &= Q(Hq^{(k)} + HZu^{(k)} + M^{-1}b)\end{aligned} \tag{2.6}$$

In summary, we obtain:

$$\begin{aligned}(I - Z^T HZ)u^{(k+1)} &= Z^T(Hq^{(k)} + M^{-1}b) \\ q^{(k+1)} &= Q(Hq^{(k)} + HZu^{(k)} + M^{-1}b)\end{aligned} \tag{2.7}$$

(Notice that because we have assumed that 1 is not an eigenvalue that $(I_r - Z^T HZ)$ is nonsingular and therefore this is well-defined).

In order to generalize, we introduce i, j components in order to vary the codependent nature on the $\mathbf{Q}, \mathbf{P}$ subspaces respectively (2.7):

Algorithm 2.1 (RPM Iteration).

$$\begin{aligned}(I - Z^T HZ)u^{(k+1)} &= Z^T(Hq^{(i)} + M^{-1}b) \\ q^{(k+1)} &= Q(Hq^{(k)} + HZu^{(j)} + M^{-1}b)\end{aligned} \tag{2.8}$$

The (i, j) pair is known as the coupling factor (as defined in [1]). The coupling factor has an important influence on the resulting Jacobian of our iteration, and the most common couplings are named below.

i	j	coupling
k	k	Jacobi
k	k+1	Gauss-Seidel (GS)
k+1	k	Reverse Gauss-Seidel (RGS)

[1]

This coupling is a rather important factor for the computational convergence of the projectors P and Q. This interplay is not fully discussed in [1, 11], and we touch on it lightly on here. We will return to this discussion after we note how to efficiently calculate expressions involving Z, P, and Q for the Jacobi coupling in the implementation section.

3. Theoretical Convergence of RPM. All of our theoretical results depend on the appropriate calculation of the Jacobian of RPM. We review this result now (note: RGS' Jacobian is a correction on [1]). The following is derived from basic linear algebra, and (2.1).

THEOREM 3.1 (The Jacobian of RPM). *Denote the error vector as the appropriate difference in both projection* **P** *and* **Q**, *respectively, as*

$$e^{(k)} = \begin{pmatrix} p^{(k)} - p \\ q^{(k)} - q \end{pmatrix} \tag{3.1}$$

Each of the three couplings' iterations can be expressed as $Je^{(k)} = e^{(k+1)}$ *[1] where:*

$$J_J = \begin{pmatrix} 0 & C \\ E & B \end{pmatrix} \tag{3.2}$$

$$J_{GS} = \begin{pmatrix} 0 & C \\ 0 & EC + B \end{pmatrix} \tag{3.3}$$

$$J_{RGS} = \begin{pmatrix} CE & CB \\ E & B \end{pmatrix} \tag{3.4}$$

Where

$$E := QHP, B := QHQ, C := P(Z(I - Z^T HZ)^{-1} Z^T)PHQ \tag{3.5}$$

Proof.

We begin with the Jacobi Coupling, and show that $C(q^{(k)} - q) = p^{(k+1)} - p$:

$$\begin{aligned} C(q^{(k)} - q) \quad = \\ & P(Z(I - Z^T HZ)^{-1} Z^T)PHQ(q^{(k)} - q) \\ & Z(I - Z^T HZ)^{-1} Z^T HQ(q^{(k)} - q) \\ & Z(I - Z^T HZ)^{-1} Z^T Hq^{(k)} - Z(I - Z^T HZ)^{-1} Z^T Hq \end{aligned} \tag{3.6}$$

We apply $(I - Z^T HZ)u^{(k+1)} = Z^T(Hq^{(k)} + M^{-1}b)$ (from equation 2.1)

$$\begin{aligned}
C(q^{(k)}-q) &= Z(I-Z^THZ)^{-1}(I-Z^THZ)u^{(k+1)} \\
&\quad -Z(I-Z^THZ)^{-1}Z^TM^{-1}b - Z(I-Z^THZ)^{-1}Z^THq \\
&= Zu^{(k+1)} - Z(I-Z^THZ)^{-1}Z^T(M^{-1}b+Hq) \\
&= Zu^{(k+1)} - Z(I-Z^THZ)^{-1}(I-Z^THZ)u \\
&= Zu^{(k+1)} - Zu \\
&= p^{(k+1)} - p
\end{aligned} \tag{3.7}$$

Now we show that $E(p^{(k)}-p)+B(q^{(k)}-q) = q^{(k+1)}-q$:

$$\begin{aligned}
& E(p^{(k)}-p)+B(q^{(k)}-q) \\
&= QHP(p^{(k)}-p)+QHQ(q^{(k)}-q) \\
&= Q(Hp^{(k)}+Hq^{(k)}) - Q(Hp+q) \\
&= Q(HZu^{(k)}+Hq^{(k)}+M^{-1}b) - Q(HZu+q+M^{-1}b)
\end{aligned} \tag{3.8}$$

This satisfies equation 2.1 exactly

$$\begin{aligned}
& Q(HZu^{(k)}+Hq^{(k)}+M^{-1}b) - Q(HZu+q+M^{-1}b) \\
&= q^{(k+1)}-q
\end{aligned} \tag{3.9}$$

We proceed similarly for the other two couplings. Starting now with Gauss-Seidel, we have already shown that $p^{(k+1)}-p = C(q^{(k)}-q)$ in the above Jacobi Coupling case, since the (i) component is the same for the Jacobi coupling as the Gauss-Seidel.

Therefore, we only show that $(EC+B)(q^{(k)}-q) = q^{(k+1)}-q$, the proof is similar to the Jacobi case with the following replacing $E(p^{(k)}+p)+B(q^{(k)}-q)$ in 3.8

$$EC(q^{(k)}-q)+B(q^{(k)}-q) \tag{3.10}$$

We have already shown that $p^{(k+1)}-p = C(q^{(k)}-q)$:

$$\begin{aligned}
&= QHP(p^{(k+1)}-p)+QHQ(q^{(k)}-q) \\
&= Q(Hp^{(k+1)}+Hq^{(k)}) - Q(Hp+q) \\
&= Q(HZu^{(k+1)}+Hq^{(k)}+M^{-1}b) - Q(HZu+q+M^{-1}b) \\
&\quad Q(HZu^{(k)}+Hq^{(k)}+M^{-1}b) - Q(HZu+q+M^{-1}b) \\
&= q^{(k+1)}-q
\end{aligned} \tag{3.11}$$

Finally, the Reverse Gauss-Seidel case. The (j) component is the same for the Jacobi coupling as the Reverse Gauss-Seidel, thus $E(p^{(k)}-p)+B(q^{(k)}-q) = q^{(k+1)}-q$.

All that remains to be shown is that $CE(p^{(k)}-p)+CB(q^{(k)}-q)$ which we already know is equal to $C(q^{(k+1)}-q) = p^{(k+1)}-p$, again applying equation 2.1 where needed.

$$\begin{aligned}
& P(Z(I-Z^THZ)^{-1}Z^T)PHQ(q^{(k+1)}-q) \\
&= Z(I-Z^THZ)^{-1}Z^THQ(q^{(k+1)}-q) \\
&= Z(I-Z^THZ)^{-1}Z^THq^{(k+1)} - Z(I-Z^THZ)^{-1}Z^THq \\
&\quad Z(I-Z^THZ)^{-1}(I-Z^THZ)u^{(k+1)} - Z(I-Z^THZ)^{-1}Z^TM^{-1}b \\
&\quad -Z(I-Z^THZ)^{-1}Z^THq \\
&= Zu^{(k+1)} - Z(I-Z^THZ)^{-1}Z^T(M^{-1}b+Hq) \\
&\quad Zu^{(k+1)} - Z(I-Z^THZ)^{-1}(I-Z^THZ)u \\
&= Zu^{(k+1)} - Zu \\
&= p^{(k+1)} - p
\end{aligned} \tag{3.12}$$

□

With this result, we may now state the central new theoretical result of this paper.

THEOREM 3.2 (RPM Preconditioner). *k steps of RPM is equivalent to preconditioning the original system (i.e., that $C^{-1}b$ approximates the solution to $Ay = b$) by:*

$$C^{-1} := (I, I)J^k \begin{pmatrix} P \\ Q \end{pmatrix} + (I, I)(I - J^k) \begin{pmatrix} P \\ Q \end{pmatrix} A^{-1} \tag{3.13}$$

Where $J = J_J, J_{GS}$, or J_{RGS} according to the chosen coupling, and assuming that Q is chosen so that J has no eigenvalues ≥ 1.

Proof. As stated in the previous theorem, RPM can be expressed via $Je_k = e_{k+1}$, thus we define:

$$\begin{aligned} v : &= \begin{pmatrix} Py \\ Qy \end{pmatrix} \\ v_k : &= \begin{pmatrix} Py_k \\ Qy_k \end{pmatrix} \\ e_k &= v_k - v \end{aligned} \tag{3.14}$$

$$Jv_k + (v - Jv) = v_{k+1} \tag{3.15}$$

One obtains the actual error vector by premultiplying (3.15) by (I, I) (since $(I, I) \begin{pmatrix} Py \\ Qy \end{pmatrix} = Py + Qy = y$). Thus, after premultiplying by (I, I) and then recursively expanding telescoping (3.15), we obtain:

$$(I, I)(J^k v_0 + \Sigma_{i=0}^{k-1} J^i (I - J)v) \tag{3.16}$$

Therefore, if we specify the right hand side in solving a system $Ay = b$, namely with $v_0 = \begin{pmatrix} P \\ Q \end{pmatrix} b$, and $v = \begin{pmatrix} P \\ Q \end{pmatrix} A^{-1}b$, we get our result that k steps of RPM is the same as preconditioning the original system by (Since postmultiplication of (3.17) with b results in (3.16)):

$$C^{-1} = (I, I)J^k \begin{pmatrix} P \\ Q \end{pmatrix} + (I, I)\Sigma_{i=0}^{k-1} J^i (I - J) \begin{pmatrix} P \\ Q \end{pmatrix} A^{-1} \tag{3.17}$$

Since J has no eigenvalues ≥ 1, then $\Sigma_{i=0}^{k-1} J^i = (J^k - I)(J - I)^{-1}$:

$$C^{-1} = (I, I)J^k \begin{pmatrix} P \\ Q \end{pmatrix} + (I, I)(I - J^k) \begin{pmatrix} P \\ Q \end{pmatrix} A^{-1} \tag{3.18}$$

□

We can restate this more clearly with the following corollary.

COROLLARY 3.3 (RPM Convergence Criteria). *k steps of RPM convergence rate bound is determined by* $(QH)^{k+1}$

Proof. We recall that the interior preconditioner of RPM (the matrix used in the Richardson splitting) is M, and the equivalent preconditioner expression of k steps of RPM as described by the previous theorem is C.

Without loss of generality, the preconditioner inside RPM (M) may be set to be I. We may say this because M in RPM is the same as using this first on A (in other words, applying RPM to the system $M^{-1}Ax = M^{-1}b$) and letting $M = I$ in (3.16).

Thus $A = I - H$ (where $H := I - M^{-1}A$).

Using the previous thorem (and now using that theoretically $E = 0 = C$):

$$
\begin{aligned}
I - C^{-1}A &= I - C^{-1}(M^{-1}A) \\
&= I - (I, I)\begin{pmatrix} 0 & 0 \\ 0 & (QHQ)^k \end{pmatrix}\begin{pmatrix} P \\ Q \end{pmatrix} M^{-1}A \\
&\quad -(I, I)\begin{pmatrix} P \\ Q \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & (QHQ)^k \end{pmatrix}\begin{pmatrix} P \\ Q \end{pmatrix} \\
&= (QHQ)^k(I - M^{-1}A) = (QH^{k+1})
\end{aligned}
\tag{3.19}
$$

□

So far, we have only detailed the convergence and rate of convergence for the Jacobi, Gauss-Seidel, and Reverse Gauss-Seidel. We now build off of the work in [11] to prove that for all couplings where $(i, j) \leq (k + 1, k + 1)$ we have convergence. In a style similar to the convergence results of [8], we define the following three $2N$ (where ideally one should choose $N > k$) dimensional matrices using GNU Octave and MATLAB's diagonal notation to simplify the presentation where $diag(v, k)$ indicates the matrix whose kth superdiagonal (subdiagonal if negative) has entries in the vector v and is otherwise 0[4, 13].

$$
\begin{aligned}
N_1 &= diag((0, 1, 0, 1, 0, 1, \cdots), -1) \\
N_2 &= diag((0, 1, 0, 1, 0, 1, \cdots), -j) \\
N_3 &= diag((1, 0, 1, 0, 1, 0, \cdots), -i)
\end{aligned}
\tag{3.20}
$$

And we define the following error vector:

$$
e_N = \begin{pmatrix} e^{(0)} \\ e^{(1)} \\ \vdots \\ e^{(k)} \\ \vdots \\ e^{(N)} \end{pmatrix} = \begin{pmatrix} (p^{(0)T} - p^T, q^{(0)T} - q^T)^T \\ (p^{(1)T} - p^T, q^{(1)T} - q^T)^T \\ \vdots \\ (p^{(k)T} - p^T, q^{(k)T} - q^T)^T \\ \vdots \\ (p^{(N)T} - p^T, q^{(N)T} - q^T)^T \end{pmatrix}
\tag{3.21}
$$

We then note that the RPM iteration as expressed in equation (2.1) can instead be modeled by the kth entry of

$$
R_N \cdot e_N := (N_1 \otimes B + N_2 \otimes C + N_3 \otimes E)^k \cdot e
\tag{3.22}
$$

Where B, C, E are defined in equation (3.5) and $\otimes$ is the Kronecker product

We may do this since we may restate (2.1) as

$$
\begin{aligned}
p^{(k+1)} - p &= P(Z(I - Z^T HZ)^{-1} Z^T)PHQ(q^{(i)} - q) \\
q^{(k+1)} - q &= QHQ(q^{(k)} - q) + QHP(p^{(j)} - p)
\end{aligned} \tag{3.23}
$$

Therefore repeated iterations are modeled by repeated multiplications of R_N, and if we show that $R_N^m \to 0$ independent of N, then we will show that all couplings $(i,j) \le (k+1, k+1)$ converge.

THEOREM 3.4 (General RPM Coupling Convergence). *$R_N^m \to 0$ in 2-norm independent of N.*

Proof.

Note that C^2 contains a product of Q and P, and therefore $= 0$, likewise, theoretically $E = 0$. Thus

$$
\begin{aligned}
||R_N^m||_2 &= ||(N_1 \otimes B + N_2 \otimes C + N_3 \otimes E)^m||_2 \\
&= ||(N_1 \otimes B + N_2 \otimes C)^m||_2 \\
&= ||\Sigma_{n=0}^m \tbinom{m}{n} (N_1^m \otimes B^m)(N_2^{n-m} \otimes C^{n-m}||_2 \\
&\le ||N_1^m \otimes B^m||_2 + m||I \otimes B^{m-1}||_2 ||N_1^{m-1} \otimes I||N_2 \otimes C||_2
\end{aligned} \tag{3.24}
$$

Since N_i simply zeros out certain elements of the shift operator, $||N_i||_2 \le 1$, thus $||N_1^{m-1} \otimes I||_2 ||N_2 \otimes I||_2 ||I \otimes C||_2 \le ||I \otimes C||_2 \le \sigma_{max}(C) = K$, which is independent of N.

All that needs to be shown if that $||I \otimes B^{m-1}||/m \to 0$. But since $B = QHQ$, and Q was chosen to correspond to be the projection to the eigenspace where the eigenvalues of H are < 1. Therefore, the spectra of B^{m-1} and likewise the maximum singular value of B^{m-1} decays exponentially to 0 and thus the above $\to 0$ independent of N.

□

4. Implementation.

In order to create an efficient implementation, we need first to discuss how to compute the projectors P, Q efficiently. To do this, we calculate the Z matrix recursively as in [1]. We first observe (using the invariance of P under H)

$$
\begin{aligned}
q^{(k+1)} &= Q(M^{-1}b + Hq^{(k)}) \\
q^{(k+1)} - q^{(k)} &= (QHQ)(q^{(k)} - q^{(k-1)})
\end{aligned} \tag{4.1}
$$

This implies that $q^{(k+1)} - q^{(k)}$ will lie in the dominant eigenspace of QHQ. Therefore, we can use the power method on the successive q vectors to progressively obtain the Q projector ([21]). The above equation can be used to approximate the dominant eigenspace of QHQ by computing a small window of $q^{(k+1)} - q^{(k)}$ for $k = j - wind + 1, \cdots, j$, computing an orthonormal basis S of this space, and then using the Schur vectors T (i.e., the columns of the orthonormal matrix of the Schur decomposition, which are needed to ensure that P is an *invariant* subspace) of the dominant eigenspace $S^T HS$ so that ST approximates the Schur vectors of H [1].

There are various ways to implement the deflation step. We proceed by first reviewing the implementation in [1] before describing our own. In [1], the vectors that comprise the Krylov subspace to be deflated are denoted by $\Delta q^{(k)} := q^{(k+1)} - q^{(k)}$, and we compute a small window of $wind$ many of these difference vectors, $\{\Delta q^{(k)}\}_{j-wind+1}^{j}$. The user chooses this $wind$ value, along with the number of eigenvalues to be deflated, denoted as def. We use a $freq$ parameter to denote during which iterations the eigenvalues are to be deflated, i.e., the frequency at which the deflation process takes place in the overall algorithm. Finally, we have a parameter denoted by *numeig* to denote the maximum number of eigenvalues total that are to be deflated (in the above, we assumed that $\mathbf{Q}$ only contained all of the eigenvalues of modulus < 1, but in implementation there is no limitation against choosing a radius smaller than 1).

With this, we may now recall the outline of the algorithm presented in [1]:

Algorithm 4.1 (RGS RPM).

```
Choose a splitting $A = M - N$ and some $y^{(0)}$. Let $H = M^{-1}N$, and $c = M^{-1}b$
# Create the initial vectors.
do k=0:freq-1
    $y^{(k+1)} = c + Hy^{(k)}$
    $\Delta^{(k=1)} = y^{(k+1)} - y^{(k)}$
enddo
$Z = \{\}$
$u^{(0)} = 0$
$q^{(0)} = y^{(0)} - Zu^{(0)}$
$t^{(0)} = c + Hq^{(0)}$
$k = 0$
while not converged
    # Extract Schur vectors for $Z$ nonmaximal
        if $size(Z, 2) < numeig$ and $mod(k, freq) = 0$
        $S = \{\Delta^{(freq)}, \cdots, \Delta^{(freq-wind+q)}\}$
        Orthogonalize $S$
        Perform Schur factorization on $S^T HS$ to obtain $def$ schur vectors $T$
        $Z1 = ST$
        $Z = (Z, Z1)$
        Orthogonalize $Z$
        $W = I - Z^T HZ$
    endif
    # Perform the RGS coupling iteration of (2.1).
    $q^{(k+1)} = (I - ZZ^T)(t^{(k)} + (HZ)u^{(k)})$
    $t^{(k+1)} = c + Hq^{(k+1)}$
    $u^{(k+1)} = W^{-1}(Z^T t^{(k+1)})$
    $\Delta^{(k+1)} = q^{(k+1)} - q^{(k)}$
    $y^{(k+1)} = Zu^{(k+1)} + q^{(k+1)}$
    $k = k + 1$
endwhile
```

However, we can improve this algorithm by introducing parallelism, instead of keeping a window of the subsequent vectors $q^{(k+1)} - q^{(k)}$ and applying the power method to extract the eigenspace, we suggest using a block of vectors and using subspace iteration, which amount to allowing q to be a $n \times m$ matrix (similar to [15, 16, 18]). This means that the *wind* parameter is no longer necessary and is instead replaced by the subspace size. Thus, in what follows all of the corresponding y, u, q, t, c vectors from above are replaced with Y, u, q, t, C which are all **block** $n \times m$ matrices. And one can use block methods to calculate the corresponding Schur vectors [3], and parallel matrix-matrix multiplies.

Then with minor changes from algorithm 4.1 (and retaining the def and $freq$ parameters defined above), this yields the following algorithm.

Algorithm 4.2 (Subspace Iteration RPM).

```
Choose some random $n \times m$ block of initial linearly independent vectors $Y$.
Choose a splitting $A = M - N$. Let $H = M^{-1}N$, and $C$ to be a $n \times m$ matrix with each column $= M^{-1}b$
do k=0:freq-1
    $Y^{(k+1)} = C + HY^{(k)}$
    # The subscript in $\Delta^{(k+1)}$ from algorithm 4.1 is no longer necessary, since we store all of the vectors as a block matrix $S$.
    $S = Y^{(k+1)} - Y^{(k)}$
enddo
$Z = \{\}$
$u^{(0)} = 0$
$q^{(0)} = Y^{(0)} - Zu^{(0)}$
$t^{(0)} = C + Hq^{(0)}$
$k = 0$
while not converged
    if $size(Z, 2) < numeig$ and $mod(k, freq) = 0$
        Orthogonalize $S$
        Perform Schur factorization on $S^T HS$ to obtain $def$ schur vectors $T$
        $Z1 = ST$
        $Z = (Z, Z1)$
        Orthogonalize $Z$ over itself
        $W = I - Z^T HZ$
    endif
    $q^{(k+1)} = (I - ZZ^T)(T^{(k)} + (HZ)u^{(k)})$
    $t^{(k+1)} = C + Hq^{(k+1)}$
    Reorthogonalize $t$
    $u^{(k+1)} = W^{-1}(Z^T t^{(k+1)})$
    $S = q^{(k+1)} - q^{(k)}$
    $Y^{(k+1)} = Zu^{(k+1)} + q^{(k+1)}$
    $k = k + 1$
endwhile
Extract the first column of $Y^{(k+1)}$[11]
```

As we have already discussed, all we have done is to take algorithm 4.1, and to replace the appropriate vectors by block matrices wherever possible. This necessitated some interesting introductions. For example, the matrix C contains redundant copies of $M^{-1}b$ to ensure that the appropriate corrections are made to *every* column in the block matrix. The Δ variable now vanishes entirely, its purpose having been subsumed by the block operations allow one to calculate S directly. However, the most important introduction by far is the reorthogonalization that takes place. This was done to ensure separation of the dominant eigenspace, as is done in subspace iteration [16]. However, choosing to reorthogonalize the t block matrix as opposed to u or q was done *a posteriori*, and how this influences the convergence is still not entirely clear.

We may now return to a discussion of the coupling. If we increase i relative to j, then note that by (2.1), we will compute more successive powers of Q before the next orthogonalization to obtain the subspace is performed. This means that the coupling factor in the block formulation of RPM really is a means to control the stability of the process illustrated in equation (4.1). Precisely how this stability depends on the coupling is a complex interaction between the range of spectra being computed in the current iteration inside the algorithm and most importantly the stability of the multiplications and eigenspace calculation (which will be discussed below in the numerical experiments).

5. Numerical Results. There are two central claims to verify of RPM. First is that with deflation we might be able to make previously nonconverging iterations converge. The second is that according to our convergence result above, further increases in number of eigenvalues deflated should only speed up convergence further. In all except the final two tests, we use the RGS coupling.

Before showing these numerical results, an important discussion on how to decide to choose the various parameters in the subspace RPM method is called for. These parameter choices were used in the experiments that follow.

The easiest parameter to determine is the subspace size (i.e., the number of columns of the Y, S, t, u, q matrices). Ideally, the subspace size will be determined by the number of available processors, and what communication limits these processors have. Usually, setting the subspace size to be the number of processors or some small multiple thereof is sufficient.

There is a balance between the subspace size and frequency. Part of this needs to take into account the number of nodes (if it is decided that the algorithm is to be implemented in parallel) and the distribution of the spectra of the iteration matrix. However, one of the advantages to setting the subspace size greater than the $freq$ parameter is that further iterations smooth out errors apparent earlier in the iterative deflation process. Therefore, as stated in [11], *subspace size is to be preferred over frequency* (this assumes that the spectra of the matrix is widely distributed enough that subspace iteration is a significant advantage over the power method). Thus, frequency should be kept at 1 or 2.

Some other parameters in the algorithm left to describe as in [1] is the maximal number of deflated eigenvalues (which we denote as *numeig*), and the number of eigenvalues deflated at each iteration (which we denote by def). Typically, the number of eigenvalues to deflate at each step should be 2 because if it were larger, then it would necessitate a higher frequency parameter, a larger number of RPM steps in order to accurately generate subspace projection, or a larger subspace to compensate. If it were smaller, then it introduces numerical issues from the necessity of deflating out complex eigenvalues pairs. However, there are cases in which a deflation size of 1 is desirable. Notably, in practice subspace RPM has to very carefully keep track of matrix sizes. One frequent problem that may occur in poorly designed subspace RPM implementations is that during the orthogonalization stage, the Z vector might hit upon two vectors that lay in nearly the same subspace. If this happens, then it means that the primary eigenspaces are extremely dominant. Fortunately, this happens early enough in the code that if this should happen, then the code should break and replace the deflation parameter from a 2 down to a 1. Furthermore, this is another reason to limit the size of the deflation parameter.

The only user-dependent parameter is *numeig*, and is completely dependent on how much cost the user is willing to endure. If it is possible to perform a simple Richardson method without any deflation, and if the user has any knowledge that the higher modulus spectra are not well-spaced, then *numeig* should be 0, which amounts to simply using the Richardson method. However, reality is a harsh mistress; the vast majority of cases require some amount of deflation. As such, this question really depends on the spectra of the matrix in question. An ideal application would be to use subspace RPM where there is a succession of matrices, each changing slightly based on a parameter (which is why nearly all of the tests in this section are on Poisson and Helmholtz problems that differ by a slightly changing diagonal parameter). In this way, if one is assured that the process succeeds in one parameter domain, the parameter domain may be extended slightly by increasing the *numeig* quantity. In short, *numeig* should be as small as one can get away with and be used in such a way or application that it can gradually be increased as the need arises.[11]

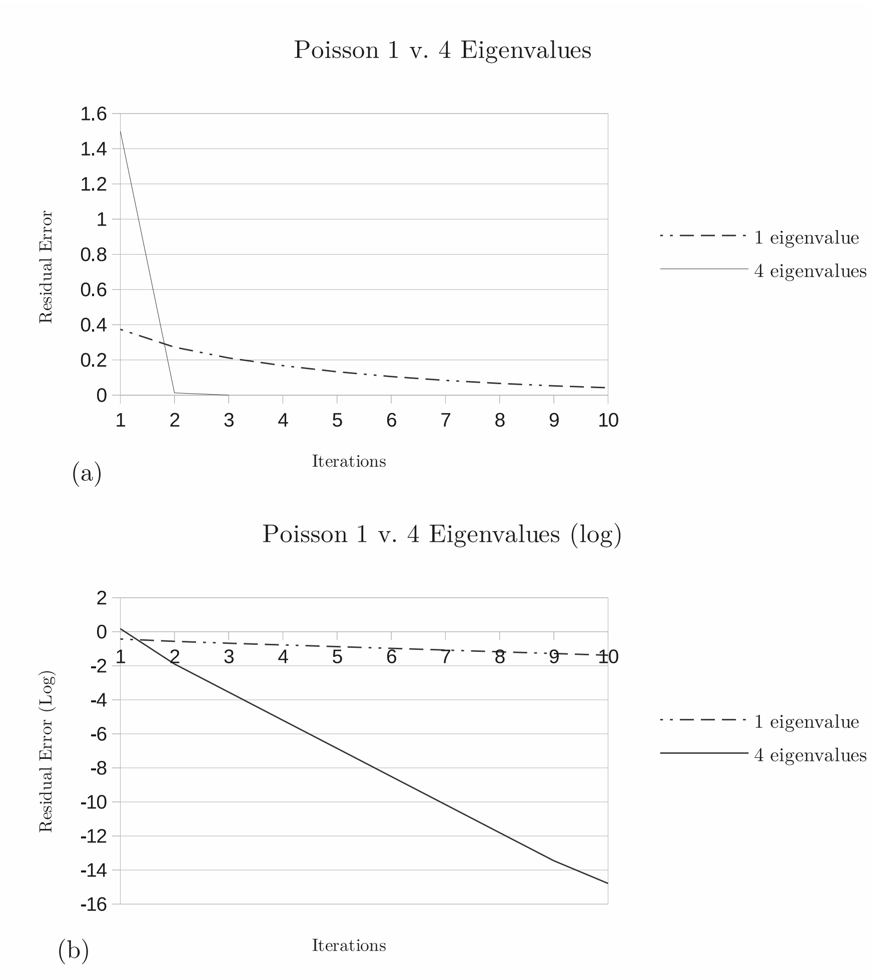

Figure 1. (a),(b): Deflation speeding up convergence[11]

We exhibit the claim that we can speed up convergence in figure 1 performed sequentially in GNU Octave [4] on a simple home desktop machine. This shows the residual norm (all norms are the standard 2-norm) vs. iterations with a very standard toy matrix setup, a Poisson matrix of size 100 simply for purposes of illustration, reordered with reverse Cuthill-McKee and using a banded matrix for our splitting. The maximum number of eigenvalues to be deflated is 4, the subspace size is 4, the frequency of deflation is 1, and the number of eigenvalues deflated at each step is 2. (note that there might be a delay in convergence initially. This is due to needing a few more initial steps in order to appropriately calculate the projector; this is a patten that will continue throughout these numerical results). It takes a few iterations for convergence to be assured (Fig. (a)), and then ensure the speed-up of RPM upon further deflation (Fig. (b)).

As stated above, this algorithm can also make divergent iterations convergent. We show this by keeping all the parameters the same, except we change the problem into a Helmholtz problem by decreasing the value of the diagonal entries down from 4 to 3.6. Again, we compare RPM with only 1 eigenvalue deflated, and with 4 eigenvalues deflated:

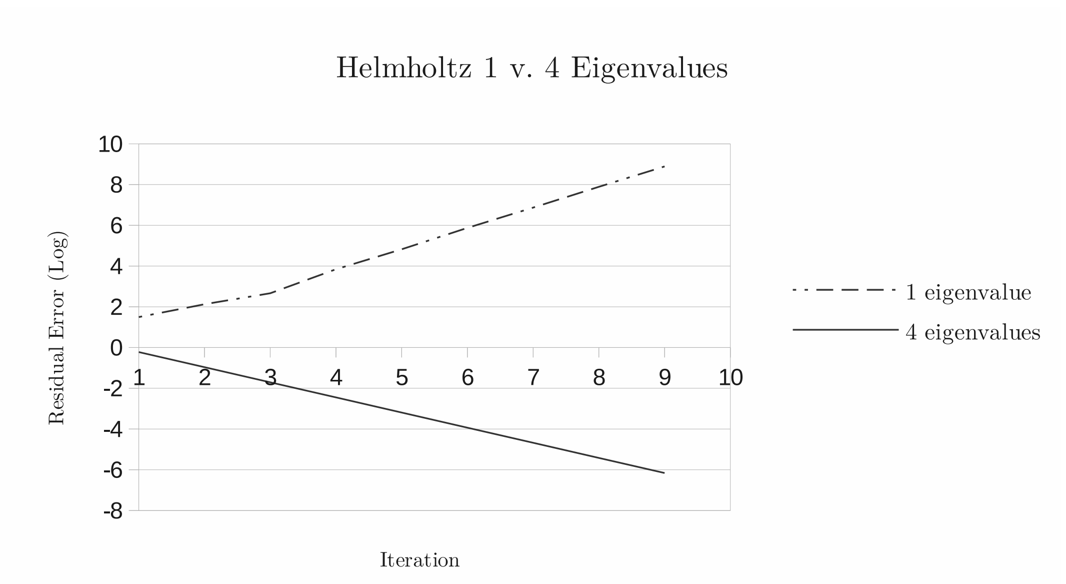

Figure 2. Deflation preventing divergence[11]

Finally, the following not only also illustrates the speed-up upon deflating more eigenvalues, but also demonstrates algorithm 4.2 and the claim of parallelism underlying the algorithm. Using this, larger sizes become much more feasible, and the following Poisson system has dimension 4194304 with approximately 400 million nonzeros running in parallel on an Intel based machine with 8 cores (we recall from the parameter discussion above that this means that the block sizes are set to be 8) and using a MPI/FORTRAN implementation. Here, we compare block Jacobi RPM with 2 eigenvalues deflated, and block Jacobi RPM with 4 eigenvalues deflated.

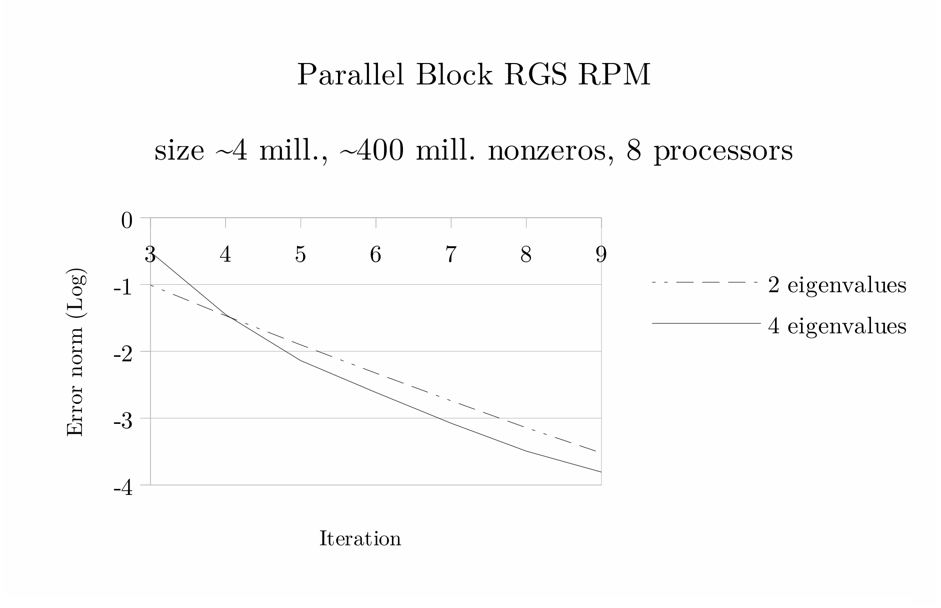

Figure 3. Parallel deflation speeding up convergence[11]

The question of the optimal coupling is a much more difficult problem and unlike the above, it has not been adequately addressed in the literature. This is partly because inherent in this question are stability issues of this algorithm, since in practice 4.2 needs a reorthogonalization step, as we mentioned that normal subspace iteration also needs. We show the stability issues with the following test, performed on a desktop machine in GNU Octave [4] with a randomly generated dense matrix in order to enhance possible stability issues

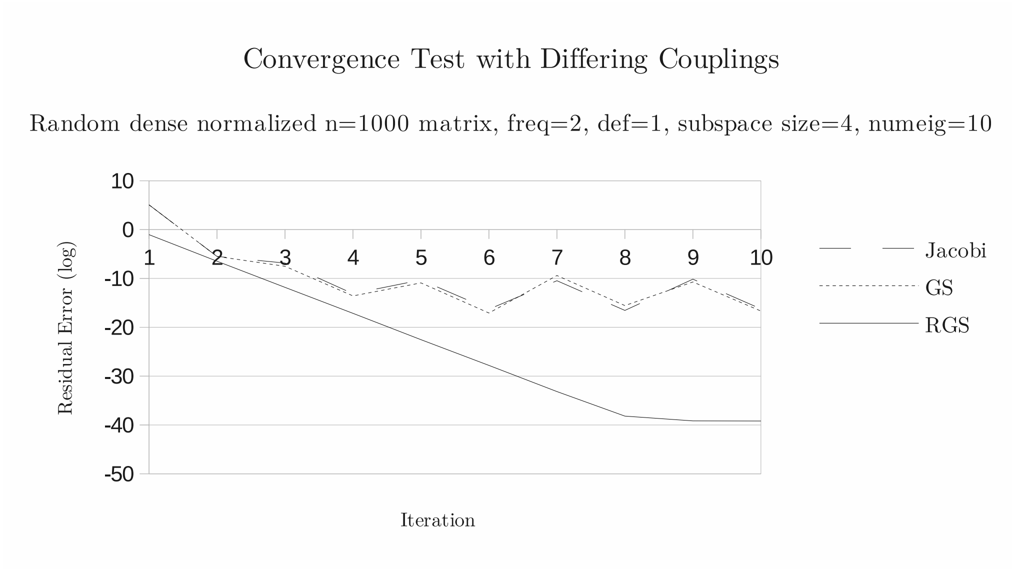

Figure 4. The influence of coupling numbers on convergence[11]

What we see is a stark performance benefit for the RGS coupling, but not for other couplings. We hypothesize that the reason for why the RGS coupling performs so much better than the other couplings is because the delay in receiving the u vector in the iteration allows the q vector to obtain an extra multiplication by H, which has the result of increasing the accuracy of the subspace iteration. As far as we have seen, the RGS coupling appears to be an ideal choice for most test cases.

6. Conclusion.

Splitting based methods frequently exhibit poor convergence or robustness. We have added to a method that aids in the alleviation of this problem through the deflation of troublesome eigenvalues implicit in the calculations of such iterative methods. In particular, we have introduced changes that allow effective parallelization while retaining the robustness of the previous RPM algorithm. We not only improved the practical implementation of subspace RPM, but also improved its theoretical background by correcting its Jacobian expression and using this to show a central preconditioner result. It was with this result that we were able to clarify the convergence behavior of subspace RPM. Finally, we provided a number of numerical examples and experiments and provided an extensive discussion of efficacious choices for the parameters in subspace RPM. With these improvements, subspace RPM can remain an effective subroutine both practically and theoretically.

Furthermore, we have left a possibility for more research into the coupling factor. Because of its ability to control the stability in determining the dominant eigenspace, it does not make sense for it to stay fixed during each iteration. It may be possible to change the algorithm slightly so that it is made more stable and the coupling factor is chosen adaptively (a similar proposal was originally suggested in [1], but not yet implemented). If at a given step the algorithm is trying to compute the nth eigenvalue to deflate against with a subspace k, then the difference in $i - j$ of the coupling factor should depend on the ratio of the eigenvalues of H, $|\frac{\lambda_n}{\lambda_{n+k}}|$ (if the hypothesis stated regarding the performance of the RGS coupling performs is true). At the moment, the accuracy of the implicit eigenvalue problem is a serious bottleneck that is typically overcome with nesting strategies as in [11]. Therefore, this remains an important avenue of research.

REFERENCES

[1] Burrage, K., Erhel, J., Pohl, B., & Williams, A. (1998). A deflation technique for linear systems of equations. *SIAM Journal on Scientific Computing*, 19(4), 1245-1260.

[2] Chapman, A., & Saad, Y. (1997). Deflated and augmented Krylov subspace techniques. *Numerical linear algebra with applications*, 4(1), 43-66.

[3] Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. SIAM.

[4] Eaton, J. W., Bateman, D., & Hauberg, S. (1997). Gnu octave. Network thoery.

[5] Erhel, J., Burrage, K., & Pohl, B. (1996). Restarted GMRES preconditioned by deflation. *Journal of computational and applied mathematics*, 69(2), 303-318.

[6] Erhel, J., & Frdric, G. H. (1997). An augmented subspace conjugate gradient.

[7] Frommer, A., & Szyld, D. B. (1992). H-splittings and two-stage iterative methods. *Numerische Mathematik*, 63(1), 345-356.

[8] Kloster, K., & Gleich, D. F. (2013). A Fast Relaxation Method for Computing a Column of the Matrix Exponential of Stochastic Matrices from Large, Sparse Networks. arXiv preprint arXiv:1310.3423.

[9] Golub, G. H., & Overton, M. L. (1982). *Convergence of a two-stage Richardson iterative procedure for solving systems of linear equations* (pp. 125-139). Springer Berlin Heidelberg.

[10] Golub, G. H., & Ye, Q. (1999). Inexact preconditioned conjugate gradient method with inner-outer iteration. SIAM Journal on Scientific Computing, 21(4), 1305-1320.

[11] Imberti, D. (2013). *Methods for Increasing Domains of Convergence in Iterative Linear System Solvers* (Doctoral Dissertation). Purdue University. Lafayette, Indiana, 2013.

[12] Lanzkron, P. J., Rose, D. J., & Szyld, D. B. (1990). Convergence of nested classical iterative methods for linear systems. *Numerische Mathematik*, 58(1), 685-702.

[13] MATLAB and Statistics Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.

[14] Nichols, N. K. (1973). On the convergence of two-stage iterative processes for solving linear equations. *SIAM Journal on Numerical Analysis*, 10(3), 460-469.

[15] Nicolaides, R. A. (1987). Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2), 355-365.

[16] Rutishauser, H. (1970). Simultaneous iteration method for symmetric matrices. *Numerische Mathematik*, 16(3), 205-223.

[17] Saad, Y. (1997). Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2), 435-449.

[18] Saad, Y., Yeung, M., Erhel, J., & Guyomarc'h, F. (2000). A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5), 1909-1926.

[19] Shroff, G. M., & Keller, H. B. (1993). Stabilization of unstable procedures: the recursive projection method. *SIAM Journal on Numerical Analysis*, 30(4), 1099-1120.

[20] Stein, P., & Rosenberg, R.L. (1948). On the Solution of Linear Simultaneous Equations by Iteration, *Journal of London Mathematical Society*, 23, 111-118.

[21] Wilkinson, J.H. (1965). *The Algebraic Eigenvalue Problem*. Oxford Science Publications.