Formalization of Asymptotic Convergence for Stationary Iterative Methods

Mohit Tekriwal[⋆], Joshua Miller, and Jean-Baptiste Jeannin

University of Michigan, Ann Arbor MI 48109, USA
{tmohit, joshmi, jeannin}@umich.edu

Abstract. Solutions to differential equations, which are used to model physical systems, are computed numerically by solving a set of discretized equations. This set of discretized equations is reduced to a large linear system, whose solution is typically found using an iterative solver. We start with an initial guess, x_0, and iterate the algorithm to obtain a sequence of solution vectors, x_k, which are approximations to the exact solution of the linear system, x. The iterative algorithm is said to converge to x, in the field of reals, if and only if x_k converges to x in the limit of $k \to \infty$.
In this paper, we formally prove the asymptotic convergence of a particular class of iterative methods called the *stationary iterative methods*, in the Coq theorem prover. We formalize the necessary and sufficient conditions required for the *iterative convergence*, and extend this result to two classical iterative methods: the Gauss–Seidel method and the Jacobi method. For the Gauss–Seidel method, we also formalize a set of *easily testable conditions* for iterative convergence, called the *Reich theorem*, for a particular matrix structure, and apply this on a model problem of the one-dimensional heat equation. We also apply the main theorem of iterative convergence to prove convergence of the Jacobi method on the model problem.

Keywords: Stationary Iterative Methods · Iterative Convergence · Gauss–Seidel method · Jacobi method

1 Introduction

Solutions to differential equations are often obtained numerically, which involves solving a large linear system, $Ax = b$. This system is obtained after discretizing a differential equation in a finite computational domain to obtain a set of discretized equations. Direct methods to solve this linear system, such as Gaussian elimination, usually involve matrix inversion, which is computationally expensive with computational complexity of $\mathcal{O}(N^3)$, where N is the dimension of the linear system. Therefore, low-cost methods such as *iterative methods* [21], which have an average complexity of $\mathcal{O}(N^2)$, are used to obtain an *approximate* solution of the linear system.

[⋆] currently affiliated with the Lawrence Livermore National Laboratory, USA.

The goal of an iterative method is to build a sequence of approximations of the *true numerical solution*, which is defined as $x \overset{\Delta}{=} A^{-1}b$. One starts with an initial guess vector x_0, and builds a sequence of approximate solutions: $\{x_1, x_2, \ldots, x_{k-1}, x_k\}$ for k iterations, with the hope that x_k is close to x. The distance between x_k and x is called the *iterative convergence error*. To ensure the asymptotic convergence of these approximate solutions to the true numerical solution, we need to bound the iterative convergence error, and further show that this error decreases as we increase the number of iterations.

Many general purpose ordinary differential equation (ODE) solvers use some kind of iterative method to solve linear systems. For instance, ODEPACK [10], which is a collection of FORTRAN solvers for initial value problems for ODEs, uses iterative (preconditioned Krylov) methods instead of direct methods for solving linear systems. Another widely used suite of ODE solvers is SUNDIALS [11]. SUNDIALS has support for a variety of direct and Krylov iterative methods for solving systems of linear equations. SUNDIALS solvers are used by the mixed finite element (MFEM) package for solving nonlinear algebraic systems and by NASA for spacecraft trajectory simulation [11]. Because those iterative methods are widely used, it is important to obtain formal guarantees for the convergence of iterative solutions to the "true" solution of differential equations. In this work we use the Coq theorem prover to formalize the convergence guarantees for a class of iterative methods called the *Stationary iterative methods*. The choice of stationary iterative methods for formalization is motivated by the fact that this class of methods is used as building blocks for more complicated iterative solvers like Krylov subspace and conjugate gradient methods [21].

Contributions: We provide an overview of the *stationary iterative methods* in Section 2, followed by a formalization of a generalized iterative convergence theorem in Coq, and its specialization to two classical iterative methods. Overall, this work[1] makes the following contributions:

- we provide a formalization of the necessary and sufficient conditions for iterative convergence in Coq in Section 4;
- we formalize a set of easily testable conditions for convergence of the Gauss–Seidel classical iterative method for a specific matrix structure in Section 5 and prove convergence on a model problem in Section 6.1;
- we then apply the generalized iterative convergence theorem to an example of the Jacobi iteration, another classical iterative method, to prove its convergence in Section 6.2;
- we develop libraries for dealing with complex matrices and vectors, and formalize ℓ^2 norm of a matrix and its spectral properties.

All of the above formalization has been done in the field of real numbers.

[1] Our Coq formalization is available at `https://github.com/mohittkr/iterative_convergence.git`

2 Overview of Stationary Iterative Methods

In this section, we provide an overview of the stationary iterative methods adapted from the textbook [21].

Let x be the true numerical solution, or the direct solution obtained by inverting the linear system $Ax = b$ as

$$x \triangleq A^{-1}b \tag{1}$$

Here, the *coefficient* matrix $A \in \mathbb{R}^{n\times n}$ and the *right hand side* vector $b \in \mathbb{R}^n$ are known to us and we are computing the unknown vector $x \in \mathbb{R}^n$. We assume that the matrix A is non-singular. Thus, there exists a unique solution x of the linear system $Ax = b$. For any iterative algorithm, we start with an initial guess vector x_0 and obtain a sequence of numerical solutions which are an approximation of the solution x. Let x_k be the iterative solution obtained after k iterations obtained by solving the iterative system

$$Mx_k + Nx_{k-1} = b \tag{2}$$

for some choice of initial solution vector x_0. The vector x_{k-1} is the iterative solution obtained after $k-1$ iterations. At the k^{th} iteration step, x_{k-1} is known to us. The matrices M and N are obtained by splitting (*regular splitting* [21]) the original coefficient matrix A such that M is easily invertible. Therefore,

$$A = M + N. \tag{3}$$

The choice of matrices M and N define the choice of an iterative method. For instance, if we choose M to be the lower triangular entries of A and N to be the strictly upper triangular entries of A, we get the Gauss–Seidel iterative method [21]. Thus, for the Gauss–Seidel method, $M = L + D$, and $N = U$, where L, D, and U are illustrated in Figure 1. If we choose M to be the diagonal entries of matrix A and N to be the strictly lower and upper triangular entries of A, we obtain the Jacobi method [21]. Thus, for the Jacobi method, $M = D$, and $N = L + U$. Therefore, the matrices M and N are also known to us based

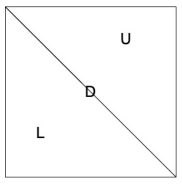

Fig. 1. Initial partitioning of matrix $A = L + D + U$. L is the strictly lower triangular matrix. D is the diagonal matrix. U is the strictly upper triangular matrix.

on the choice of an iterative method. The right hand vector b is also known to us. Thus, the unknown solution vector x_k at k^{th} step is obtained by rearranging terms in the iterative system (2) as

$$x_k = (-M^{-1}N)x_{k-1} + M^{-1}b \tag{4}$$

The quantity $(-M^{-1}N)$ in equation (4) is called an *iterative matrix* and we will denote it as S. Therefore,

$$S \triangleq -M^{-1}N \tag{5}$$

The iterative convergence error after k iterations is defined as

$$e_k^{iterative} \triangleq x_k - x = S^k e_0; \quad e_o \triangleq x_o - x \tag{6}$$

The iterative solution x_k is said to converge to x in the field of reals if and only if

$$\lim_{k\to\infty} ||e_k^{iterative}|| = \lim_{k\to\infty} ||x_k - x|| = 0 \tag{7}$$

where $||.||$ denotes a vector norm. In this paper, we will be using the ℓ^2 vector norm defined as $||x||_2 = \sqrt{\sum_{j=1}^{n} |x_{\{i\}}|^2}$.

In this work, we consider two concrete instances of stationary iterative methods: the Gauss–Seidel method [21], for which the iterative matrix $S_G \triangleq -M^{-1}N = -(L+D)^{-1}U$, and the Jacobi method [21], for which the iterative matrix $S_J \triangleq -M^{-1}N = -D^{-1}(A-D) = I - D^{-1}A$.

3 Generic iterative convergence theorem in the field of reals

The following theorem provides necessary and sufficient conditions for iterative convergence in the field of reals.

Theorem 1. *[21] Let an iterative matrix be defined as (5) for the iterative system (2). The sequence of iterative solutions $\{x_k\}$ converges to the direct solution x for all initial values x_0, if and only if the spectral radius of the iterative matrix $S = -M^{-1}N$ is less than 1, i.e.,*

$$(\forall\ x_o, \lim_{k\to\infty} ||x_k - x|| = 0) \iff \rho(-M^{-1}N) < 1.$$

The spectral radius ρ of a matrix is defined as the maximum eigenvalue in magnitude. We next discuss the proof of Theorem 1 followed by its formalization in the Coq proof assistant. It is noteworthy that while such proofs have been discussed in numerical analysis literature, we found several missing pieces during the formalization. Most facts about intermediate steps in the proof were just stated in the numerical analysis literature without a rigorous proof. In that regard, a contribution of this work is to provide a clean machine-checked proof of the main theorem and any intermediate lemma or fact that was required to close the proof of Theorem 1. Because of a lack of space, we provide most of the informal proofs in the Appendix A and C in the extended version of this paper [24].

To prove Theorem 1, we first need to obtain a recurrence relation for the iterative convergence error at k^{th} step in terms of the initial iteration error $(x_0 - x)$. Therefore,

Proof (Proof of Theorem 1).

$$\begin{aligned} x_k - x &= -M^{-1}Nx_{k-1} + M^{-1}b - x \\ &= -M^{-1}Nx_{k-1} + M^{-1}(Ax) - x \quad [\text{Since, } Ax \triangleq b] \\ &= -M^{-1}Nx_{k-1} + M^{-1}(M+N)x - x \; [\text{Since, } M+N = A] \\ &= -M^{-1}Nx_{k-1} + M^{-1}Mx + M^{-1}Nx - x \\ &= -M^{-1}N(x_{k-1} - x) \quad [\text{Since, } M^{-1}M \triangleq I] \end{aligned}$$

Taking the norm of the vector on both sides, the iterative convergence error at the k^{th} step can be written in terms of the iterative convergence error at $(k-1)$ step as

$$||x_k - x|| = ||(-M^{-1}N)(x_{k-1} - x)|| \tag{8}$$

Since, the system is linear, equation (8) can be written in terms of the initial iteration error as

$$||x_k - x|| = ||(-M^{-1}N)^k(x_0 - x)|| \tag{9}$$

Taking limits of the vector norms on both sides of equation (9),

$$\lim_{k\to\infty} ||x_k - x|| = \lim_{k\to\infty} ||(-M^{-1}N)^k(x_0 - x)|| \tag{10}$$

If $x_0 = x$, the iterative convergence error is zero trivially. The case $x_0 \neq x$ is interesting and we can prove Theorem 1 by splitting it into two lemmas

Lemma 1. *For given matrices $M \in \mathbb{R}^{n\times n}$ and $N \in \mathbb{R}^{n\times n}$ respecting the regular splitting, i.e., $A = M + N$, the sequence of iterative solutions $\{x_k\}$ converges to x for any given initial vector x_0 if and only if the ℓ^2 matrix norm of the iterated product of the iteration matrix, $(-M^{-1}N)^k$ approaches zero as $k \to \infty$, i.e.,*

$$(\forall x_0, \lim_{k\to\infty} ||(-M^{-1}N)^k(x_0 - x)|| = 0) \iff \lim_{k\to\infty} ||(-M^{-1}N)^k|| = 0 \tag{11}$$

Lemma 2. *For given matrices $M \in \mathbb{R}^{n\times n}$ and $N \in \mathbb{R}^{n\times n}$ respecting the regular splitting, i.e., $A = M + N$, the ℓ^2 matrix norm of the iterated product of the iteration matrix, $(-M^{-1}N)^k$ approaches zero as $k \to \infty$ if and only if the spectral radius of the iteration matrix is less than 1, i.e.,*

$$\lim_{k\to\infty} ||(-M^{-1}N)^k|| = 0 \iff \rho(-M^{-1}N) < 1 \tag{12}$$

Therefore, by composing proofs of the Lemma 1 (see Appendix A in the extended version [24]) and the Lemma 2 (see Appendix C in the extended version [24]), we close the proof of the Theorem 1.

4 Formalization of the the generalized iterative convergence theorem (Theorem 1) in Coq

One of the main challenges that we encountered during our formalization was the lack of theories for matrix and vector norm and generic properties about complex vectors and matrices. We will first discuss the formalization of these properties in Coq, followed by their adoption in our formalization on iterative convergence.

4.1 Formalizing properties of complex matrices and vectors

The `complex` theory in the `real_closed` [7] library in MathComp defines complex numbers and basic operations on them. They define complex numbers as a real closed field, thereby allowing us to instantiate a generic field with a complex field. This was useful when we used the `eigenvalue` definition from MathComp matrix algebra library and the canonical forms library by Cano et al. [6]. However, since the basic properties like modulus of a complex number, conjugates, properties of complex matrices and vectors were lacking, we added them in our formalization[2]. We define the modulus of a complex number as

Definition C_mod (x: complex R):= sqrt ((Re x)^+2 + (Im x)^+2).

`Re x` and `Im x` denote the real and imaginary part of `x`, respectively. We proved some basic properties of the modulus, which we enumerate in Table 1. We also found that some theory on complex conjugates were missing in the MathComp library. The Table 2 lists the missing formalization of complex conjugates that we added for this formalization. We also formalize the properties of a conjugate matrix like idempotent property, scaling of a complex matrix, conjugate transpose of matrix multiplication, etc., which we do not list here for brevity.

4.2 Formalization of vector and matrix norms

Another missing piece in the existing linear algebra theory in MathComp was the norm of a vector and a matrix. In this work, we formalize the ℓ^2-norm of a vector and its induced matrix norm. In Coq, we define the ℓ^2-norm of a matrix as

Definition matrix_norm (n:nat) (A: 'M[complex R]_n.+1) := Lub_Rbar (fun x ⇒
  ∃ v: 'cV[complex R]_n.+1, v != 0 ∧ x = (vec_norm_C (A *m v))/ (vec_norm_C v)).

where `vec_norm_C` is the ℓ^2-norm of a complex vector, which we define in Coq as

Definition vec_norm_C (n:nat) (x: 'cV[complex R]_n.+1):= sqrt ($\sum_l$ (C_mod x l 0)^2)

[2] The theory about complex modulus and norms has been added in most recent developments of MathComp after our discussion with the developers. We were pointed to the development of matrix norms in the CoqQ project (`https://github.com/coq-quantum/CoqQ/blob/main/src/mxnorm.v`), which was done concurrently with our development.

Mathematical properties	Formalization in Coq
$\lVert 0\rVert = 0$	**Lemma** C_mod_0: C_mod 0 = 0%Re.
$0 \leq \lVert x\rVert$	**Lemma** C_mod_ge_0: ∀(x: complex R), (0<= C_mod x)%Re.
$\lVert xy\rVert = \lVert x\rVert\ \lVert y\rVert$	**Lemma** C_mod_prod: ∀(x y: complex R), C_mod (x ∗ y) = C_mod x ∗ C_mod y.
$\lVert \frac{x}{y}\rVert = \frac{\lVert x\rVert}{\lVert y\rVert}, y \neq 0$	**Lemma** C_mod_div: ∀(x y: complex R), y <> 0 → C_mod (x / y) = (C_mod x) / (C_mod y).
$\lVert x\rVert \neq 0, \quad \text{if. } x \neq 0$	**Lemma** C_mod_not_zero: ∀(x: complex R), x <> 0 → C_mod x <> 0.
$\lVert 1\rVert = 1$	**Lemma** C_mod_1: C_mod 1 = 1.
$\lVert x^n\rVert = \lVert x\rVert^n$	**Lemma** C_mod_pow: ∀(x: complex R) (n:nat), C_mod (x^+ n) = (C_mod x)^+n.
$\lVert x+y\rVert \leq \lVert x\rVert + \lVert y\rVert$	**Lemma** C_mod_add_leq : ∀(a b: complex R), C_mod (a + b) <= C_mod a + C_mod b.
$\lVert \frac{1}{x}\rVert = \frac{1}{\lVert x\rVert}, \quad \text{if } x \neq 0$	**Lemma** C_mod_inv : ∀x : complex R, x <> 0 → C_mod (invc x) = Rinv (C_mod x).
$\lVert xy\rVert^2 = \lVert x\rVert^2 \lVert y\rVert^2$	**Lemma** C_mod_sqr: ∀(x y : complex R), Rsqr (C_mod (x ∗ y)) = (Rsqr (C_mod x)) ∗ (Rsqr (C_mod y)).
$\lVert -x\rVert = \lVert x\rVert$	**Lemma** C_mod_minus_x: ∀(x: complex R), C_mod (−x) = C_mod x.
$\lVert \sum_{j=0}^{n} u(j)\rVert \leq \sum_{j=0}^{n} \lVert u(j)\rVert$	**Lemma** C_mod_sum_rel: ∀(n:nat) (u : 'I_n.+1 → (complex R)), (C_mod ($\sum_j$ (u j))) <= $\sum_j$ ((C_mod (u j))).

Table 1. Formalization of properties of complex modulus in Coq

Mathematical properties	Formalization in Coq
$\overline{xy} = \bar{x}\ \bar{y}$	**Lemma** Cconj_prod: ∀(x y: complex R), conjc (x∗y)%C = (conjc x ∗ conjc y)%C.
$\overline{x+y} = \bar{x} + \bar{y}$	**Lemma** Cconj_add: ∀(x y: complex R), conjc (x+y) = conjc x + conjc y.
$\|\|x\|\| = \|\|\bar{x}\|\|$	**Lemma** Cconjc_mod: ∀(a: complex R), C_mod a = C_mod (conjc a).
$x = \bar{\bar{x}}$	**Lemma** conj_of_conj_C: ∀(x: complex R), x = conjc (conjc x).
$\bar{x}x = \|\|x\|\|^2$	**Lemma** conj_prod: ∀(x:complex R), ((conjc x)∗x)%C = RtoC (Rsqr (C_mod x)). 1
$Re[x] + Re[\bar{x}] = 2Re[x]$	**Lemma** Re_conjc_add: ∀(x: complex R), Re x + Re (conjc x) = 2 ∗ (Re x).
$\overline{\sum_{j=0}^{n} f(i)} = \sum_{j=0}^{n} \overline{f(i)}$	**Lemma** Cconj_sum: ∀(p:nat) (x: 'I_p → complex R), conjc ($\sum_{j<p}$ x j)= $\sum_{j<p}$ conjc (x j).

Table 2. Formalization of properties of complex conjugates in Coq. [1]Here, RtoC is a coercion from reals to complex.

The definition `Lub_Rbar` is the least upper bound and is already defined in the `Coquelicot` [5] library. Mathematically, `matrix_norm` formalizes the following definition of a matrix norm $||A||_i = \sup_{x\neq 0} \frac{||Ax||}{||x||}$, for a given vector norm $||.||$, which in this case is the ℓ^2 vector norm. In Table 3 and Table 4 we enumerate the properties of vector and matrix norms that we formalized.
An important point to note here is that since we are using the `Coquelicot` definition of an extended real line, `Rbar`, coercion of a quantity of type `Rbar` to real requires us to prove finiteness of that quantity. We therefore have to prove that the matrix norm is finite, which we state as the following lemma in Coq

Lemma matrix_norm_is_finite: ∀(n:nat) (A: 'M[complex R]_n.+1), is_finite (matrix_norm A).

Since we prove asymptotic convergence of component-wise limit of the elements of a Jordan matrix, we have to work with the Frobenius norm of a matrix, which we define in Coq as

Definition mat_norm (n:nat) (A: 'M[complex R]_n.+1) : R := sqrt ($\sum_i \sum_j$ (C_mod (A i j))^2)

We will next discuss the formalization of the Theorem 1.

4.3 Formalization of the Theorem 1 in Coq

We state Theorem 1 in Coq as follows:

Theorem iter_convergence: ∀(n:nat) (A: 'M[R]_n.+1) (b: 'cV[R]_n.+1) (M N : 'M[R]_n.+1),
A \in unitmx → M \in unitmx → A = M + N →
let x := (A^−1) ∗m b **in**
(**let** S_mat := RtoC_mat (− (M^−1 ∗m N)) **in**
(∀ (i: 'I_n.+1), (C_mod (lambda S_mat i) < 1))) ↔
(∀ x0: 'cV[R]_n.+1, is_lim_seq (fun k:nat ⇒ vec_norm ((X_m k.+1 x0 b M N) − x)) 0).

Mathematical properties	Formalization in Coq
$0 \leq \|\|v\|\|$	**Lemma** vec_norm_C_ge_0: ∀ (n:nat) (v: 'cV[complex R]_n.+1), 0<= vec_norm_C v.
$\|\|av\|\| = \|a\| \; \|\|v\|\|, \quad a$ is scalar	**Lemma** ei_vec_ei_compat: ∀ (n:nat) (x:complex R) (v: 'cV[complex R]_n.+1), vec_norm_C (scal_vec_C x v) = C_mod x ∗ vec_norm_C v.
$\|\|v_1 + v_2\|\| \leq \|\|v_1\|\| + \|\|v_2\|\|$	**Lemma** vec_norm_add_le: ∀ (n:nat) (v1 v2 : 'cV[complex R]_n.+1), vec_norm_C (v1 + v2) <= vec_norm_C v1 + vec_norm_C v2.
$v \neq 0 \implies \|\|v\|\| \neq 0$[1]	**Lemma** non_zero_vec_norm: ∀ (n:nat) (v: 'cV[complex R]_n.+1), vec_not_zero v → vec_norm_C v <> 0.

Table 3. Formalization of properties of vector norm in Coq. [1] vec_not_zero is Coq's definition of $v \neq 0$.

Mathematical properties	Formalization in Coq
$0 \leq \|\|A\|\|_i$	**Lemma** matrix_norm_ge_0: ∀ (n:nat) (A: 'M[complex R]_n.+1), 0 <= matrix_norm A.
$\|\|Ax\|\| \leq \|\|A\|\|_i \|\|x\|\|, \quad x \neq 0$[1]	**Lemma** matrix_norm_compat: ∀ (n:nat) (x: 'cV[complex R]_n.+1) (A: 'M[complex R]_n.+1), x != 0 → vec_norm_C (mulmx A x) <= (matrix_norm A) ∗ vec_norm_C x.
$\|\|AB\|\|_i \leq \|\|A\|\|_i \|\|B\|\|_i$	**Lemma** matrix_norm_prod: ∀ (n:nat) (A B: 'M[complex R]_n.+1), matrix_norm (A ∗m B) <= (matrix_norm A) ∗ (matrix_norm B).
$0 \leq \|\|A\|\|_i \leq \|\|A\|\|_F$[2]	**Lemma** mat_2_norm_F_norm_compat: ∀ (n:nat) (A: 'M[complex R]_n.+1), 0 <= matrix_norm A <= mat_norm A.

Table 4. Formalization of properties of matrix norm in Coq.
[1]Here, x is a vector and the relation proves compatibilty relation between a matrix norm and its induced vector norm.
[2]Here, $||A||_F$ is the Frobenius norm and we prove that the 2-norm of a matrix is bounded above by the Frobenius matrix norm. The Frobenius norm of a matrix is defined as $||A||_F = \sqrt{\sum_{j=1}^{n} \sum_{j=1}^{n} |A_{ij}|^2}$

The theorem iter_convergence states that if the square matrix $A \in \mathbb{R}^{n+1 \times n+1}$ is invertible, which is formalized by the condition A \in unitmx and if the sub-matrices $M, N \in \mathbb{R}^{n+1 \times n+1}$ respect the regular splitting property, which is formalized by $A = M + N$, and if M is invertible, which is formalized by M \in unitmx, then $\lim_{k \to \infty} ||x_{k+1} - x|| = 0$ if and only if $\forall\ i, |\lambda_i(S)| < 1$. Here, $|.|$ is the complex modulus of the eigenvalues of the iteration matrix S (S_mat as defined in the let statement of the theorem iter_convergence), which is defined as in equation (5). The is_lim_seq predicate from the Coquelicot library [5] defines limit of a sequence. The reason for having the dimension of A as $(n+1) \times (n+1)$ is that the natural numbers start from 0 in Coq and a square matrix of size 0 does not type check. Thus, instead of having $0 < n$ as pre-condition in the theorem explicitly, we specify this implicitly in the dimension of A. This change also makes our development compatible with the Jordan canonical formalization by Cano et al. [6], who need this constraint on the dimension to define and work with block matrix.

Since we deal with a generic case where a real matrix is allowed to have complex eigenvalues and eigenvectors, we need to transform the real iteration matrix S to a complex matrix, so as to be consistent with types. Thus, given a real matrix A, RtoC_mat transforms a real entry $A_{ij} : \mathbb{R}$ to a complex number $\tilde{A}_{ij} : \mathbb{C} := (A_{ij} + i * 0)$. An important point to note here is that we do not compute the *true numerical solution* x explicitly, but define it as $x \triangleq A^{-1}b$ in the let binding of the theorem statement – **let** x := (A^−1) ∗m b. We define the iterative solution after k steps, x_k from the iterative system (2) using the **Fixpoint** operator in Coq, which lets us define the recurrence relation (4)

```
Fixpoint X_m (k n:nat) (x0 b: 'cV[R]_n.+1) (M N: 'M[R]_n.+1) : 'cV[R]_n.+1:=
match k with
| O ⇒ x0
| S p ⇒ ((− ((M^−1) ∗m N)) ∗m (X_m p x0 b M N)) + ((M^−1) ∗m b)
end.
```

Formalization of eigenvalues One of the main issues we faced was coming up with a scalar definition of eigenvalues, which satisfies the characteristic definition $Av = \lambda v$. The MathComp library defines a non-computable definition of eigenvalues, eigenvalue, which is a predicate stating that λ is an eigenvalue of a matrix A, if the eigenspace corresponding to λ is non-zero. For the scalar definition of eigenvalue, we took an inspiration from the Jordan canonical forms formalization by Guillaume Cano and Maxime Dénès [6]. They define a Jordan form, whose diagonal elements are the characteristic polynomials of the Smith Normal form of a matrix A. Since the diagonal elements of a Jordan form are the eigenvalues of that matrix, we then define a sequence of eigenvalues from these diagonal entries of the Jordan matrix as

```
Definition lambda_seq (n: nat) (A: 'M[complex R]_n.+1) :=
let sizes:= size_sum [seq x.2.−1 | x <− root_seq_poly (invariant_factors A)] in
[seq (Jordan_form A) i i | i <− enum 'I_sizes.+1].
```

where root_seq_poly p returns a sequence of pair of roots and its multiplicity, of the polynomial p. The invariant_factors are the polynomials in the diagonal of the Smith Normal form of a matrix. In this case, the sequence contains the pair of eigenvalues of matrix A and its multiplicity. The i^{th} eigenvalue of A is then defined as the i^{th} component of the sequence of eigenvalues lambda_seq.

```
Definition lambda (n: nat) (A: 'M[complex R]_n.+1) (i: 'I_n.+1) :=
    (@nth _ 0%C (lambda_seq A) i).
```

To take full advantage of the lemmas describing eigenvalues and eigenvectors as defined in MathComp, we had to relate the definition of eigenvalue, `lambda`, and the one defined in MathComp . The lemma Jordan_ii_is_eigen asserts that lambda satisfies the predicate eigenvalue, and is indeed an eigenvalue of a matrix A.

```
Lemma Jordan_ii_is_eigen: ∀(n: nat) (A: 'M[complex R]_n.+1),
  ∀(i: 'I_n.+1), @eigenvalue (complex_fieldType _) n.+1 A (@nth _ 0%C (lambda_seq A) i).
```

Here, `size_sum` is the sum of the algebraic multiplicities of the eigenvalues and equals the total size of the matrix n. We prove this fact using the following lemma statement in Coq

```
Lemma total_eigen_val: ∀(n:nat) (A: 'M[complex R]_n.+1),
(size_sum [seq x.2.−1 | x <− root_seq_poly (invariant_factors A)]).+1 = n.+1.
```

The lemma `total_eigen_val` helps us get around the dimension constraint imposed by the design of the Jordan form of a matrix A.

Formalization of the ratio test As discussed in the informal proof in Appendix C to prove sufficiency condition for iterative convergence, we had to formalize the ratio test for convergence of sequences which was missing in the existing Coq libraries. In Coq, we state the ratio test (Lemma 3 in Appendix C in the extended version [24]) as:

```
Lemma ratio_test: ∀(a: nat → R) (L:R), (0 < L ∧ L < 1) →
(∀ n:nat, (0 < a n)) →
(is_lim_seq ( fun n:nat ⇒ ((a (n.+1))/(a n))) L) →
is_lim_seq (fun n: nat ⇒ a n) 0.
```

Proving convergence of the Jordan block matrix To prove that each element of the Jordan block matrix J^k (see Appendix C in the extended version [24] for more details) converges to zero, i.e.,

$$\forall i\ j, \lim_{k\to\infty} |(J^k)_{(i,j)}|^2 = 0$$

where

$$J^k = \begin{bmatrix} J^k_{m_1}(\lambda_1) & 0 & 0 & \dots & 0 \\ 0 & J^k_{m_2}(\lambda_2) & 0 & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \vdots \\ 0 & \dots & 0 & J^k_{m_{s-1}}(\lambda_{s-1}) & 0 \\ 0 & \dots & \dots & 0 & J^k_{m_s}(\lambda_s) \end{bmatrix}$$

and

$$J_{m_i}^k(\lambda_i) = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \dots & \binom{k}{m_i-1}\lambda_i^{k-m_i+1} \\ 0 & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \dots & \binom{k}{m_i-2}\lambda_i^{k-m_i+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} \\ 0 & 0 & \dots & 0 & \lambda_i^k \end{bmatrix},$$

we prove the following lemma.

Lemma each_entry_zero_lim: ∀ (n:nat) (A: 'M[complex R]_n.+1), (i j: 'I_(size_sum sizes).+1),
let sp := root_seq_poly (invariant_factors A) **in**
let sizes := [seq x0.2.−1 | x0 <− sp] **in**
(∀ i: 'I_(size_sum sizes).+1 , (C_mod (nth 0%C (lambda_seq A) i) < 1)) →
is_lim_seq (fun m: nat ⇒
let block := (fun n0 i1 : nat ⇒ **let** lambda := (nth (0, 0%N) sp i1).1 **in**
\matrix_(i2, j0) (($\binom{m.+1}{j0-i2}$) ∗ (lambda ^ (m.+1 − (j0 − i2))) ∗+ (i2 <= j0)) **in**
(C_mod ((diag_block_mx sizes block) i j))^2) 0.

The lemma each_entry_zero_lim states that if the magnitude of each eigenvalue of a matrix A is less than 1, i.e., $|\lambda_i(A)| < 1,\ \forall i,\ 0 \le i < N$, then the limit of each term in the expanded Jordan matrix is zero as $k \to \infty$. Here, the block diagonal matrix diag_block_mx takes an expanded Jordan block $J_{m_i}^k(\lambda_i), \forall i, 0 \le i < N$ and constructs the Jordan matrix J^k. We then take a modulus of each entry of J^k and prove that its limit is zero as $k \to \infty$. A key challenge we faced when proving the lemma each_entry_zero_lim was extracting each Jordan block of the diagonal block matrix. The diagonal block matrix is defined recursively over a function which takes a block matrix of size μ_i denoting the algebraic multiplicity of each eigenvalues λ_i. We had to carefully destruct this definition of diagonal block matrix and extract the Jordan block and the zeros on the off-diagonal entries, which we formalize using the lemma diag_destruct in Coq. We can then prove the limit on this Jordan block by exploiting its upper triangular structure.

Limits of the off-diagonal elements can then be trivially proven to be zero since each of those elements are zero. This completes the proof of sufficiency condition for convergence of iterative convergence error.

5 Proof of convergence for the Gauss–Seidel method

To prove the convergence of a Gauss–Seidel method, we need to prove that the spectral radius of S_G is less than 1. But computing the eigenvalues of S_G explicitly is almost impossible for a generic matrix. Therefore, we need an easier check to assert that the spectral radius of S_G is indeed less than 1. The *Reich theorem* [20] provides a sufficient condition for the spectral radius of S_G to be less than 1, for a real and symmetric coefficient matrix A, with all of the elements in its main diagonal positive. This condition provides a much easier check, which is linear in time, and when layered with the Theorem 1, provides a sufficient condition for convergence of the Gauss–Seidel iteration for a real and symmetric coefficient matrix A, with all of the elements in its main diagonal positive.

We next discuss the formalization of the *Reich theorem*, followed by the main convergence theorem for the Gauss–Seidel iteration.

5.1 Formalization of easily checkable conditions

Theorem 2 (Reich theorem). *[20] If A is real, symmetric nth-order matrix with all terms on its main diagonal positive, then a sufficient condition for all the n characteristic roots of $(-M^{-1}N)$ to be smaller than unity in magnitude is that A is positive definite.*

From an application point of view, only the sufficiency condition is important. This is because to apply Theorem 1, we only need to know that the magnitude of the eigenvalues are less than 1. Thus, to prove the convergence of Gauss–Seidel iteration, we first apply Theorem 1 to get the eigenvalue condition in the goal and then apply Theorem 2 to complete the proof. Since computing eigenvalues are not very trivial in most cases, the positive definite property of the matrix A provides an easy test for $|\lambda| < 1$ for Gauss–Seidel iteration matrix.

We next present an informal proof of the Reich Theorem followed by its formalization in Coq

Proof. Let z_i be the i^{th} characteristic vector of $-(A_1^{-1}A_2)$ corresponding to the characteristic root μ_i. Then

$$-(A_1^{-1}A_2)z_i = \mu_i z_i \tag{13}$$

Multiplying by $-(\bar{z_i}'A_1)$ on both sides,

$$(-\bar{z_i}'A_1)(-A_1^{-1}A_2)z_i = -\mu_i \bar{z_i}'A_1 z_i \tag{14}$$

where $\bar{z_i}'$ is the conjugate transpose of z_i obtained by taking the conjugate of each element of z_i followed by transpose of the vector. Equation (14) then simplifies to:

$$\bar{z_i}'A_2 z_i = -\mu_i \bar{z_i}'A_1 z_i; \quad [A_1 A_1^{-1} = I] \tag{15}$$

Consider the bi-linear form, $\bar{z_i}'Az_i$,

$$\begin{aligned}\bar{z_i}'Az_i = \bar{z_i}'(A_1 + A_2)z_i &= \bar{z_i}'A_1 z_i + \bar{z_i}'A_2 z_i \\ &= \bar{z_i}'A_1 z_i - \mu_i \bar{z_i}'A_1 z_i \\ &= (1-\mu_i)\bar{z_i}'A_1 z_i \end{aligned} \tag{16}$$

Taking conjugate transpose of equation (16) on both sides,

$$\bar{z_i}'Az_i = (1-\bar{\mu_i})\bar{z_i}'A_1' z_i \tag{17}$$

Let D be the diagonal matrix defined as:

$$D_{ij} = \begin{cases} A_{ij} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{18}$$

It can be shown that

$$A_1' = D + A_2 \tag{19}$$

Substituting equation (19) in equation (17),

$$\begin{aligned}\bar{z_i}'Az_i &= (1-\bar{\mu_i})\bar{z_i}'(D+A_2)z_i \\ &= (1-\bar{\mu_i})\bar{z_i}'Dz_i + (1-\bar{\mu_i})\bar{z_i}'A_2z_i \\ &= (1-\bar{\mu_i})\bar{z_i}'Dz_i + \frac{(1-\bar{\mu_i})}{1-\mu_i}\bar{z_i}'Az_i\end{aligned} \tag{20}$$

Simplifying equation (20),

$$(1-\bar{\mu_i}\mu_i)\bar{z_i}'Az_i = (1-\bar{\mu_i})(1-\mu_i)\bar{z_i}'Dz_i \tag{21}$$

But, $\bar{\mu_i}\mu_i = |\mu_i|^2$ and $(1-\bar{\mu_i})(1-\mu_i) = |1-\mu_i|^2$ Hence, equation (21) simplifies to,

$$(1-|\mu_i|^2)\bar{z_i}'Az_i = (|1-\mu_i|^2)\bar{z_i}'Dz_i \tag{22}$$

Since, the diagonal elements of A is positive, i.e., $A_{ii} > 0$, $\bar{z_i}'Dz_i$ is positive definite, i.e., $\bar{z_i}'Dz_i > 0$. Since $\bar{z_i}'Dz_i > 0$ and $\bar{z_i}'Az_i > 0$, $|\mu_i| < 1$.

Formalization in Coq: We formalize the Theorem 2 in Coq as follows:

```
Theorem Reich_sufficiency: ∀(n:nat) (A: 'M[R]_n.+1),
(∀ i:'I_n.+1, A i i > 0) →
(∀ i j:'I_n.+1, A i j = A j i) →
is_positive_definite A →
(let S_G := − ( (RtoC_mat (M_G A)^−1) *m (RtoC_mat (N_G A))) in
    (∀ i: 'I_n.+1, C_mod (lambda S_G i) < 1)).
```

where positive definiteness of a complex matrix A is defined as: $\forall x \in \mathbb{C}^{n\times 1}$, $Re\ [x^*Ax] > 0$. x^* is the complex conjugate transpose of vector x and $Re\ [x^*Ax]$ is the real part of the complex scalar x^*Ax. The hypothesis ∀i:'I_n.+1, A i i > 0 states that all terms in the main diagonal of A are positive. The hypothesis ∀i j:'I_n.+1, A i j = A j i states that the matrix A is symmetric.

5.2 Proof of convergence for the Gauss–Seidel method

We then apply the theorem iter_convergence with Reich_sufficiency to prove convergence of the Gauss–Seidel iteration method. We formalize the convergence of the Gauss–Seidel iteration method in Coq as

```
Theorem Gauss_Seidel_converges: ∀(n:nat) (A: 'M[R]_n.+1) (b: 'cV[R]_n.+1),
let x := (A^−1) *m b in
A \in unitmx →
(∀ i : 'I_n.+1, 0 < A i i) →
(∀ i j : 'I_n.+1, A i j = A j i) →
is_positive_definite A →
(∀ x0: 'cV[R]_n.+1, is_lim_seq (fun k:nat ⇒
        vec_norm ((X_m k.+1 x0 b (M_G A) (N_G A)) − x)) 0).
```

Thus, to prove convergence of the Gauss–Seidel method for a real, symmetric matrix with all of its diagonal elements positive, we just need positive-definiteness of A, which can be proved by showing positivity of determinants of all upper-left sub-matrices (principal minors) [19]. We illustrate this approach using the following example.

Example 1. To show that the following matrix is positive definite,

$$A_{example} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

we compute the determinants of all possible $k \times k$ upper sub-matrices, i.e.,

$$|2| = 2; \quad \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} = 3; \quad \begin{vmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 4.$$

Since the determinant of all $k \times k$ upper sub-matrices are positive, the matrix $A_{example}$ is positive definite. We will be using this approach to show positive definiteness in Section 6.1. Note that this approach did not involve computing the eigenvalues of S_G and only relies on the structure of matrix $A_{example}$, thereby making it easy to check for positive definiteness and hence convergence of Gauss–Seidel method, by virtue of the Reich theorem (Theorem 2).

Note on sufficient conditions for convergence of the Jacobi method. The Reich theorem, which we formalized in Coq, provides sufficient conditions for convergence of the Gauss–Seidel method for a symmetric and positive definite matrix. Similar sufficient conditions also exist for convergence of the Jacobi method, which relies on showing *strict row diagonal dominance of the matrix*, or *diagonal dominance and irreducibility of the matrix* [1]. A matrix A is said to be strictly diagonally dominant if $|A_{ii}| > \sum_{j \neq i} |A_{ij}|$. This is a much easier check for convergence than computing the eigenvalues of the iterative matrix explicitly. However, we do not formalize this check in this work because the proof of this fact uses the proof of the *Gersǵorin-type theorems* [16]. The Gersǵorin-type theorems have not been formalized in Coq, and therefore their adoption in our work would require a separate proof effort for this theorem.

6 Model problem

We apply our convergence theorems on a concrete linear differential equation $\frac{d^2u}{dx^2} = 1$ for $x \in (0, 1)$, with boundary conditions: $u(0) = u(1) = 0$, as a proof of concept. This differential equation is used to model the heat diffusion in a rod, i.e., 1-D domain. We chose a uniform grid with P points in the interior of the $1-$D domain. The grid has a uniform spacing h. We will be using a centered difference scheme [23] for discretizing the differential equation. Therefore, the difference equation at point x_i in the interior of the $1-$D domain is given by

$$\frac{-u(x_{i+1}) + 2u(x_i) - u(x_{i-1})}{h^2} = -1; \quad h = x_{i+1} - x_i = \frac{1}{P+1} \tag{23}$$

When we stack the equation (23) for all points in the interior of the $1-$D domain, we get a linear matrix system

$$\underbrace{\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & 0 & -1 & 2 & -1 \\ 0 & \dots & 0 & 0 & 0 & -1 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N - 1 \\ u_N \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \\ -1 \end{bmatrix}}_{b} \tag{24}$$

Here, A is the coefficient matrix, b is the right hand side vector and x is the unknown solution vector, which can be exactly obtained by inverting the matrix A, i.e., $x = A^{-1}b$. But, we will obtain an approximation of x using iterative algorithms. We will instantiate two classical iterative algorithms: Gauss–Seidel and Jacobi, with this example problem and apply Theorem 1 to prove convergence of the approximate solutions, obtained using these algorithms, to the exact solution.

6.1 Gauss–Seidel method

We next demonstrate the convergence of the Gauss–Seidel iteration on the example (24). We choose $P = 1$. Thus, we have a symmetric tri-diagonal coefficient matrix of size 3×3, which we will denote as A_{GS}. To show that iterative system for the system $A_{GS}x = b$ converges, we need to show that A_{GS} is positive definite by application of the theorem Gauss_Seidel_converges, which we proved in Section 5.2. In Coq, we prove that A_{GS} is positive definite (by proving positivity of determinant of the principal minors of A_{GS}, as illustrated in Example 1) the following lemma statement

```
Lemma Ah_pd: ∀(h:R), (0<h) → is_positive_definite (Ah 2%N h).
```

Proving that A_{GS} is positive definite using the approach illustrated in Example 1 for a generic N is tedious and does not add much to our line of argument. Hence, we chose to do it for A_{GS} of size 3×3. One can perform this exercise for any choice of N by defining an algorithm for computing the determinant of principal minors of a matrix and get the same result. The statement of convergence of Gauss–Seidel iteration method for the 3×3 matrix is stated in Coq as

```
Theorem Gauss_seidel_Ah_converges: ∀(b: 'cV[R]_3) (h:R), (0 < h) →
let A := (Ah 2%N h) in
let x:= (A^−1) *m b in
∀x0: 'cV[R]_3, is_lim_seq (fun k:nat ⇒ vec_norm ((X_m k.+1 x0 b (M_G A) (N_G A)) − x)) 0.
```

This closes the proof of the convergence of the Gauss–Seidel iteration for the model problem.

6.2 Jacobi method

We next apply the Theorem 1 to prove convergence of the Jacobi iteration on the model problem (24). As discussed earlier, the iteration matrix for a Jacobi iteration method is $S_J = I - D^{-1}A_J$. We choose $P = 1$, thereby obtaining a 3×3 matrix system, like we did for the Gauss–Seidel iteration.

Formalization in Coq: We prove the following theorem in Coq for convergence of the Jacobi method for the model problem

```
Theorem Jacobi_converges: ∀(b: 'cV[R]_3) (h:R), (0 < h) →
let A := (Ah 2h) in
let x := (A^−1) *m b in
∀x0: 'cV[R]_3, is_lim_seq (fun k:nat ⇒ vec_norm ((X_m k.+1 x0 b (M_J 2 h) (N_J 2 h)) − x)) 0.
```

To prove jacobi_converges using the Theorem 1, we need to prove that the modulus of each of the eigenvalues of S_J is less than 1. We prove this using the following lemma statement in Coq

```
Theorem eig_less_than_1: ∀(n:nat) (i: 'I_n.+1) (h:R),
  (0 < h) → (0 < n) → (C_mod (lambda_J i n h) < 1).
```

Since the iteration matrix S_J is tri-diagonal, we can define a closed form expression for lambda_J using the formula

$$|\lambda_i(S_J)| = \left|1 + \frac{h^2}{2}\lambda_i(A_J)\right| = \left|\cos\left(\frac{m\pi}{P+1}\right)\right| \tag{25}$$

A caveat in using lambda_J explicitly is that the definition of eigenvalue, lambda in Theorem 1 is based on the roots of the polynomials in the diagonal of the Smith Normal form of a matrix. However, we use the closed form expression for eigenvalue of the tridiagonal iteration matrix S_J for our model problem. Ideally, we would like to prove that lambda_J equals lambda. Since the proof of this relation is a tangent to the line of argument we are making in this work, we assume that this relation holds in this work, which we state formally in Coq as the following hypothesis

```
Hypothesis Lambda_eq: ∀(n:nat) (h:R) (i: 'I_n.+1),
  let S_mat := RtoC_mat (− ( (M_J n h)^−1 *m (N_J n h) )) in
  lambda S_mat i = lambda_J i n h.
```

This closes the proof of iterative convergence for Jacobi iteration on the model problem.

7 Related work

A number of works have recently emerged in the area of formalization of numerical analysis. This has been facilitated by advancements in automatic and interactive theorem proving [5, 9]. Some notable works in the formalization of numerical analysis include the formalization of the Kantorovich theorem for convergence of Newton methods [18], the formalization of the matrix canonical forms [6], and the formalization of the Perron-Frobenius theorem in Isabelle/HOL [25] for determining the growth rate of A^n for small matrices A. Boldo et al. [2–4] prove consistency, stability and convergence of a second-order centered scheme for the wave equation. Tekriwal et al. [23] formalize the Lax equivalence theorem to guarantee convergence of a generic class of finite difference schemes. Besides Coq, numerical analysis of ordinary differential equations (ODEs) has also been done in Isabelle/ HOL [13]. Immler et al. [14, 15] formalize flows, Poincaré map of dynamical systems, and verified rigorous bounds on numerical algorithms in Isabelle/HOL. In [12], Immler formalized a functional algorithm that computes enclosures of solutions of ODEs in Isabelle/HOL. Deniz et al. [8] formally analyze the problem of heat conduction in Isabelle/HOL, which assumes a closed form solution in a 1-D domain and is thus specific to the problem.

Since most of the existing formalizations on differential equations assume either a closed form solution or use integration schemes which are problem specific, these works do not provide a generalized framework for solving differential equations numerically. Our work however addresses the issue of generalizability by formalizing a framework for solving a linear system iteratively, which is the approach followed by most general purpose linear solvers like ODEPACK [10] and SUNDIALS [11].

8 Conclusion and Future work

In this work we formalized a *generalized theorem* about convergence of the solutions of an iterative algorithm to the *true numerical solution (direct solution)*. In this process, we clarify various details in the proof of convergence, which are missing in the classical numerical analysis literature. We then instantiate this *generalized* theorem to two classical iterative methods – the Gauss-Seidel method, and the Jacobi method on a model problem. Since it is cumbersome to compute the eigenvalues of a generic matrix system, and verify that its magnitude is less than 1, we provide a much easier check for convergence, especially for the Gauss–Seidel method, called the *Reich theorem* [20], which relies on the structure of matrix A. By composing the proof of the Reich theorem with the main iterative convergence theorem, we show convergence of the Gauss–Seidel method for this matrix A. Thus, our approach is *modular*, and can be extended to prove convergence of the Gauss–Seidel method on any desired matrix structure for which one can formalize conditions similar to the Reich theorem. During our fomalization, we develop a library in Coq to deal with complex vectors and matrices. We define absolute values of complex numbers, common properties of complex conjugates and operations on conjugate matrices and vectors. This development leverages the existing formalization [7, 17] of complex numbers and matrices in MathComp. The overall length of the Coq code and proofs is about 8.5k lines of code. It took us about 8 person-months of full time work for the entire formalization.

This work could be extended to develop a generalized end-to-end framework for verification of stationary iterative methods, which includes floating-point error analysis for a concrete implementation of the algorithm in C. In a related work by Tekriwal et al. [22], the authors provide an end-to-end correctness proof for a concrete implementation of the Jacobi iteration, which is an instance of stationary iterative methods. They use a bound on the real iterative solution for the Jacobi method, which relies on a proof of convergence of this real iterative solution to the "true" numerical solution. Thus, a crucial step in extending this result to a generic stationary iterative method, would be to use a *generalized* proof of convergence for a stationary iterative method. The results from this paper can thus be directly used in extending the results from [22] to a concrete implementation of a generic stationary iterative method.

This work could also be extended to verify solutions of non-linear systems. Most physical systems behave non-linearly, and the analysis of these non-linear systems is usually done by linearlizing it around an optimal trajectory. We also plan on extending this work to analyze more practical class of iterative methods called the *Krylov subspace methods* [21], which use stationary iterative methods like Jacobi methods as preconditioners. Thus, our work is foundational in the analysis of these complicated and practical iterative solvers.

References

1. Bagnara, R.: A unified proof for the convergence of jacobi and gauss–seidel methods. SIAM review **37**(1), 93–97 (1995)
2. Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P.: Formal proof of a wave equation resolution scheme: the method error. In: International Conference on Interactive Theorem Proving. pp. 147–162. Springer (2010)
3. Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P.: Wave equation numerical resolution: a comprehensive mechanized proof of a C program. Journal of Automated Reasoning **50**(4), 423–456 (2013)
4. Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P.: Trusting computations: a mechanized proof from partial differential equations to actual program. Computers & Mathematics with Applications **68**(3), 325–352 (2014)
5. Boldo, S., Lelay, C., Melquiond, G.: Coquelicot: A user-friendly library of real analysis for Coq. Mathematics in Computer Science **9**(1), 41–62 (2015)
6. Cano, G., Dénès, M.: Matrices à blocs et en forme canonique. In: JFLA – Journées francophones des langages applicatifs (2013), `https://hal.inria.fr/hal-00779376`
7. Cohen, C.: Construction of real algebraic numbers in Coq. In: Interactive Theorem Proving (2012), `https://hal.inria.fr/hal-00671809`
8. Deniz, E., Rashid, A., Hasan, O., Tahar, S.: On the formalization of the heat conduction problem in hol. In: International Conference on Intelligent Computer Mathematics. pp. 21–37. Springer (2022)
9. Garillot, F., Gonthier, G., Mahboubi, A., Rideau, L.: Packaging mathematical structures. In: International Conference on Theorem Proving in Higher Order Logics. pp. 327–342. Springer (2009)
10. Hindmarsh, A.C.: Odepack, a systematized collection of ode solvers. Scientific computing pp. 55–64 (1983)

11. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) **31**(3), 363–396 (2005)
12. Immler, F.: Formally verified computation of enclosures of solutions of ordinary differential equations. In: Badger, J.M., Rozier, K.Y. (eds.) NASA Formal Methods. pp. 113–127. Springer International Publishing, Cham (2014)
13. Immler, F., Hölzl, J.: Numerical analysis of ordinary differential equations in Isabelle/HOL. In: International Conference on Interactive Theorem Proving. pp. 377–392. Springer (2012)
14. Immler, F., Traut, C.: The flow of ODEs. In: International Conference on Interactive Theorem Proving. pp. 184–199. Springer (2016)
15. Immler, F., Traut, C.: The flow of ODEs: Formalization of variational equation and Poincaré map. Journal of Automated Reasoning **62**(2), 215–236 (2019)
16. Lancaster, P., Tismenetsky, M.: The theory of matrices: with applications. Elsevier (1985)
17. Mahboubi, A., Cohen, C.: Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. Logical Methods in Computer Science **8** (2012)
18. Pasca, I.: Formal Verification for Numerical Methods. Ph.D. thesis, Université Nice Sophia Antipolis (2010)
19. Prussing, J.E.: The principal minor test for semidefinite matrices. Journal of Guidance, Control, and Dynamics **9**(1), 121–122 (1986)
20. Reich, E.: On the convergence of the classical iterative method of solving linear simultaneous equations. The Annals of Mathematical Statistics **20**(3), 448–451 (1949)
21. Saad, Y.: Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 2nd ed. edn. (2003)
22. Tekriwal, M., Appel, A.W., Kellison, A.E., Bindel, D., Jeannin, J.B.: Verified correctness, accuracy, and convergence of a stationary iterative linear solver: Jacobi method. In: International Conference on Intelligent Computer Mathematics. pp. 206–221. Springer (2023)
23. Tekriwal, M., Duraisamy, K., Jeannin, J.B.: A formal proof of the Lax equivalence theorem for finite difference schemes. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NASA Formal Methods. pp. 322–339. Springer International Publishing, Cham (2021)
24. Tekriwal, M., Miller, J., Jeannin, J.B.: Formalization of asymptotic convergence for stationary iterative methods (extended version) (2022), `https://arxiv.org/abs/2202.05587`
25. Thiemann, R.: A Perron-Frobenius theorem for deciding matrix growth. Journal of Logical and Algebraic Methods in Programming **123**, 100699 (2021)