Direct and Iterative Solver of Linear Systems

Numerical Analysis II

David Thau
Lucius Anderson

Introduction

- In this paper, we will be evaluating numerical methods for direct and iterative solvers of linear systems. From class we have discussed the various methods; Gauss elimination with pivoting techniques, Jacobi Iterative Method, Gauss-Seidel Iterative Method, Successive Over-Relaxation Method, Iterative Refinement Method, and Conjugate Gradient Method. In this paper, using Python programming language, we will discuss how each method evaluates various linear systems of equations, and then we will discuss the complexity, accuracy, and stability of each method.
- Complexity will be judged based on the standard of Big O notation. Meaning we will discuss the methods makeup and then show the Big O notation of that individual method and compare each method. Showing quantitively evidence of the complexity as well.
- Accuracy is judged based on error calculated from each method. An accurate approximation is judged desirable when being below a tolerance value of 10^{-15}. This standard of accuracy is chosen because we feel that in academia, and in application that error less than this tolerance provides a relatively accurate approximation.
- Stability, in our project, is judged upon convergence. Meaning a method is declared stable if the norm converges to zero or the residuals converge to zero. This important because this converges shows that method is becoming more and more accurate over time. Divergence of the norm or residuals declares that a method is unstable.

Table of Contents

Gaussian Elimination with Pivoting

- Gaussian elimination is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix. In our project we use Gaussian elimination with pivoting to speed up the convergence rate process and reduce total round off error with approximating x.

- **Convergence:**

  - Gaussian elimination technique, with pivoting is almost guaranteed to have a finite sequential step count that will have a given matrix A converge. When using pivoting techniques for the method, converge towards a solution is found faster and the use of multipliers helps reduce round off error. However, the biggest downside Gaussian elimination is number of iterative steps needed to approximate a value and that to use the method, the matrix must be invertible, which is very hard to do in application.
  - Gaussian elimination with pivoting is also a very complex approach to solving linear systems. Gaussian elimination when approximating uses many different arithmetic operations: multiplication, division, addition, subtraction, and even row interchanges. Because of this Gaussian elimination has complexity of $O(n^3)$. With a breakdown table provided here:
  -

Division	Multiplication	Subtraction
n(n+1) /2	(2n^3 + 3n^2 -5n)/6	(2n^3+3n^2-5n)/6

- **Accuracy**:
  - Gaussian elimination with pivoting is a highly accurate direct method, and as n steps increases it becomes more and more accurate. The downside to this accuracy is the computational cost.
- **Stability:**
  - Gaussian elimination with pivoting is stable direct method. Because of the pivoting round off error is decreased and this projects and even accurate direct approximation as n steps increases.
- **Scalability:**
  - As far as scalability, this is where the gaussian elimination with pivoting method falls short. The computational cost is very high at small matrix sizes, implicating this method into real world application could prove very costly and even still this method may not produce desirable results. Because of the method being operated on the basis of an invertible matrix this could cause problems in application. Many real-world applications are very large and producing an inverse may not be feasible.

General Mathematical Backing for Iterative Methods

Theorem

For any initial $x^{(0)}$, the sequence $\{x^{(k)}\}_k$ defined by $x^{(k)} = Tx^{(k-1)} + c$ converges to the unique solution of $x = Tx + c$ iff $\rho(T) < 1$.

Proof.

($\Leftarrow$) Suppose $\rho(T) < 1$. Then

$$\begin{aligned} x^{(k)} &= Tx^{(k-1)} + c = T(Tx^{(k-2)} + c) + c \\ &= T^2x^{(k-2)} + (I + T)c \\ &= \cdots \\ &= T^kx^{(0)} + (I + T + T^2 + \cdots + T^k)c \end{aligned}$$

Let $k \to \infty$, we know $T^k \to 0$ and $(I + T + T^2 + \cdots + T^k) \to (I - T)^{-1}$, so $x^{(k)} \to (I - T)^{-1}c$, which is the unique solution of $x = Tx + c$. □

Theorem

For any initial $x^{(0)}$, the sequence $\{x^{(k)}\}_k$ defined by $x^{(k)} = Tx^{(k-1)} + c$ converges to the unique solution of $x = Tx + c$ iff $\rho(T) < 1$.

Proof.

($\Rightarrow$) Suppose for any initial $x^{(0)}$, the sequence $\{x^{(k)}\}_k$ defined by $x^{(k)} = Tx^{(k-1)} + c$ converges to the unique solution of $x = Tx + c$ iff $\rho(T) < 1$.

Let x^* be the solution of $x = Tx + c$. Then for any $z \in \mathbb{R}$, we set initial $x^{(0)} = x^* - z$. Then

$$\begin{aligned} x^* - x^{(k)} &= (Tx^* + c) - (Tx^{(k-1)} + c) = T(x^* - x^{(k-1)}) \\ &= \cdots = T^k(x^* - x^{(0)}) = T^kz \to 0 \end{aligned}$$

This implies $\rho(T) < 1$. □

Corollary

If $\|T\| < 1$ *for any matrix norm* $\|\cdot\|$, *and* c *is given, then* $\{x^{(k)}$ *generated by* $x^{(k)} = Tx^{(k-1)} + c$ *converges to the unique solution* x^* *of* $x = Tx + c$. *Moreover*

- $\|x^* - x^{(k)}\| \leq \|T\|^k \|x^* - x^{(0)}\|$.
- $\|x^* - x^{(k)}\| \leq \frac{\|T\|^k}{1-\|T\|} \|x^{(1)} - x^{(0)}\|$.

Jacobi's Iterate Method

The Jacobi Iterative Method is an iterative technique to solve the $n \times n$ linear system $Ax = b$ starts with an initial approximation $x^{(0)}$ to the solution x and generates a sequence of vectors $\{x^k\}_{k=0}^{\infty}$ that converges to x.

- **Convergence:**
  - The standard convergence condition, for the Jacobi method, is when the spectral radius of the iteration matrix is less and 1. A sufficient but not necessary condition for the Jacobi's method to converge is that matrix A has to be strictly or irreducibly diagonally dominant.
  - The Jacobi Method still sometimes converges even when these conditions are not met.
  - In our examples, all the matrices are strictly diagonally dominate for simplicity. We can see that this method takes the longest to converge with 48 iterations before finding the actual solution of x in our first example, 42 iterations in our second example, and it didn't even finish converging after 50 iterations in our third example.
  - As the system becomes larger, the convergence time increases significantly as the complexity at which Jacobi runs is $O(n^2)$.
    - Therefore
  - However certain remarks are noted about convergence for the Jacobi method:
    - Jacobi method requires nonzero diagonal entries: $a_{ii} \neq 0$ for all i.
    - Faster convergence occurs when the absolute value of the pivot is large.
- **Accuracy:**
  - Jacobi method is not a highly accurate iterative method. Usually for accurate results Jacobi's method requires a matrix to be strictly diagonally dominate, which in application, matrices rarely are. Because of this and the general makeup of the method it is not fairly accurate, and as the iteration number increases the accuracy of the Jacobi method increases only if the matrix fits the convergence criteria.
- **Stability:**
  - For practical stability Jacobi's method heavily relies upon the properties of a matrix being strictly diagonally dominant, and even without this property, over large iterations Jacobi's method is not fairly stable.
- **Scalability:**
  - Jacobi's method is not practically scalable. Because of its criteria of needing a strictly diagonally dominate matrix, and even still having low accuracy at high iterations, this method is not practical for scalability. Implementing this method with a large matrix in application would prove very difficult.

Successive Over-Relaxation

- **Convergence:** SOR method provides a good convergence rate. The method has a Big O notation of O(n^2) for complexity so the method does well to converge with bigger data sets. The complexity of the algorithms provides decent method to iterate with. However, the problem lies in choosing a relaxation parameter. The choice of relaxation factor ω is not necessarily easy and depends upon the properties of the coefficient matrix.
    - **From Wikipedia,** convergence rate of the SOR method can be analytically derived shown here:

Convergence Rate [edit]

The convergence rate for the SOR method can be analytically derived. One needs to assume the following

- the relaxation parameter is appropriate: $\omega \in (0, 2)$
- Jacobi's iteration matrix $C_{\text{Jac}} := I - D^{-1}A$ has only real eigenvalues
- Jacobi's method is convergent: $\mu := \rho(C_{\text{Jac}}) < 1$
- a unique solution exists: $\det A \neq 0$.

Then the convergence rate can be expressed as[2]

$$\rho(C_\omega) = \begin{cases} \frac{1}{4}\left(\omega\mu + \sqrt{\omega^2\mu^2 - 4(\omega - 1)}\right)^2, & 0 < \omega \leq \omega_{\text{opt}} \\ \omega - 1, & \omega_{\text{opt}} < \omega < 2 \end{cases}$$

where the optimal relaxation parameter is given by

$$\omega_{\text{opt}} := 1 + \left(\frac{\mu}{1 + \sqrt{1 - \mu^2}}\right)^2.$$

- **Accuracy:** The SOR method is fairly accurate given that the omega chosen is desirable.

- **Stability:** Stability of the method is fairly decent, given a relaxation factor is chosen correctly. The method relies solely on this so problems for stability arise at higher iterative steps when a relaxation factor is not properly chosen.

- **Scalability:** This method can be scalable, but it is not all time wise to scale a system of linear solvers based on this method. Because of the relaxation factor. A relaxation number that is too small or too large can produce undesired results and when applied this can be very costly.

Gauss-Seidel Iterative Method

The Gauss-Seidel Iterative Method is very similar to the Jacobi's Iterative Method, it solves the $n \times n$ linear system, where $b \in \mathbb{R}$. Gauss-Seidel mechanism works by:

- Initializing $x^{(0)} \in \mathbb{R}^n$. Set L to the lower triangular part of A and U = A-L
- Repeat the following for k = 0,1, . . . until convergence:
  - $x^{k+1} = L^{-1}\left(b - Ux^{(k)}\right)$

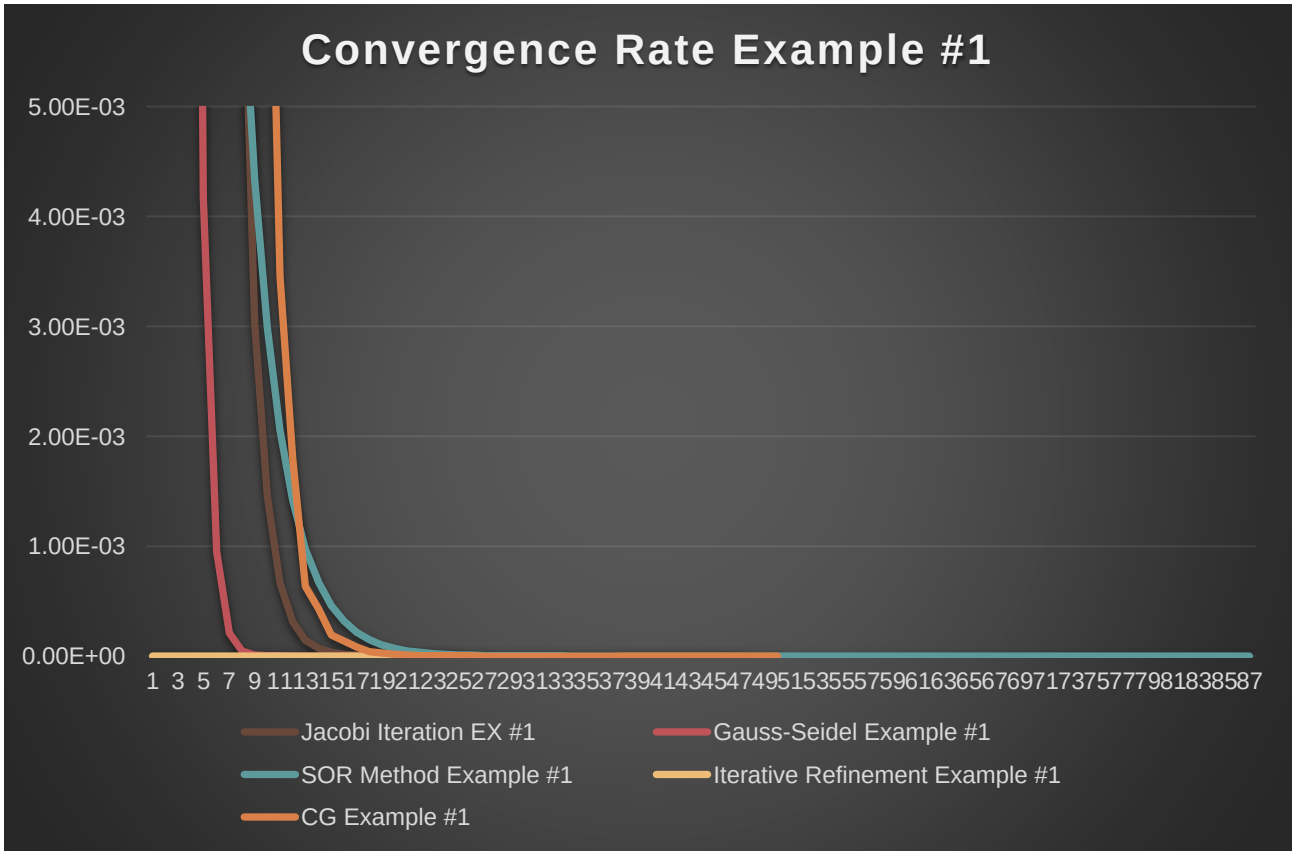
- **Convergence:**
  - The convergence properties of the Gauss-Seidel method are dependent on the matrix A
    - A is symmetric positive definite
    - A is strictly or irreducibly diagonally dominant then from any initial $x^{(0)}$ generate sequences that converge to the unique solution $Ax = b$.
  - If both Jacobi and Gauss-Seidel methods converge, Gauss-Seidel converges almost twice as fast as the Jacobi method.
    - As we can see in our examples, it only took 25 iterations in the first example to converge, 16 iterations in the second example, while it took Jacobi 48, and 42 iterations respectively.
  - The complexity of Gauss-Seidel method is the same as Jacobi, $O(n^2)$, therefore as the system gets increasingly larger, the convergence time for that system increases dramatically.
- Accuracy:
  - Gauss-Seidel method is not used in real word applications because it becomes highly inaccurate when matrix A is not strictly diagonally dominant. The accuracy of Gauss-Seidel method only increases as the iteration increases when the matrix fits the convergence criteria.
- Stability:
  - The Gauss-Seidel heavily relies on the matrix being strictly diagonally dominate. Without it following this criteria, the method can be highly unstable.
- Scalability:
  - As the system size increases, Gauss-Seidel will start to become highly inaccurate and the convergence time will increase significantly as Big O is $O(n^2)$. Also as Gauss-Seidel method relies on the matrix being strictly diagonally dominant therefore implementing this method for real applications would be almost impossible.

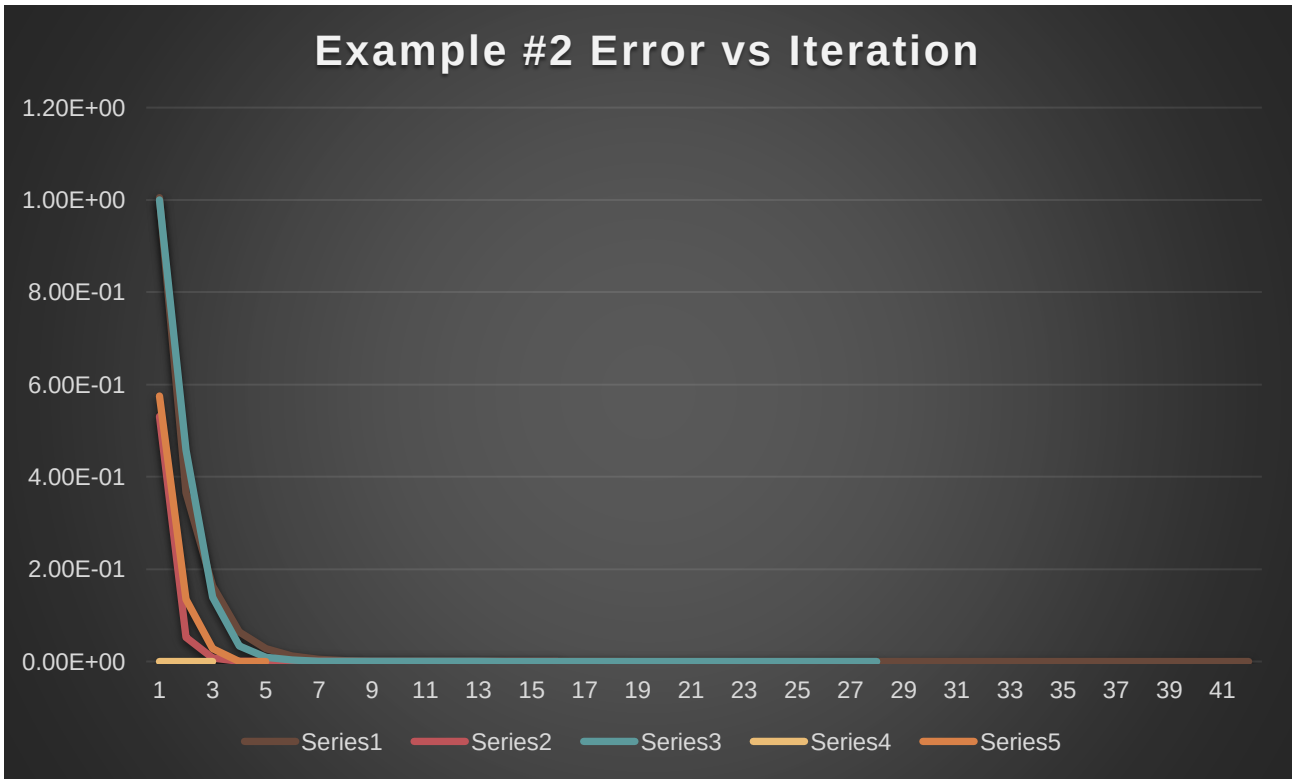
Iterative Refinement

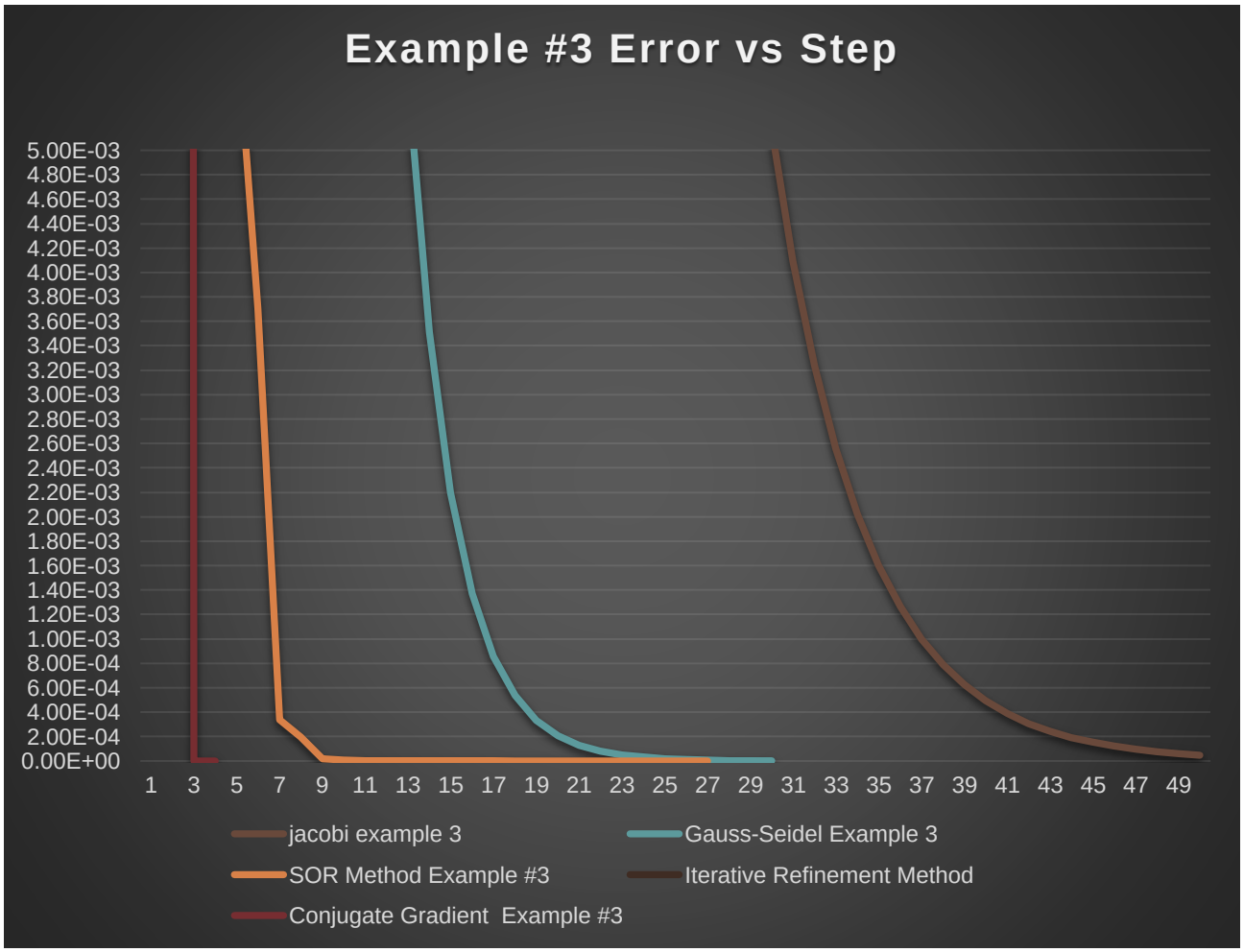
- **Convergence:** The iterative refinement method proves to have a fairly good use of convergence and the method is useful in practice. However, the method heavy relies on a matrix being well conditioned. Meaning the if the conditional norm of a matrix is closer to 1 the iterative method will find an accurate solution in an extremely timely manner, but if not, method still can be used to iterate correctly but it will take a longer time.
- **Accuracy:** if the matrix being used is well conditioned then the matrix is accurate if not, then it isn't using this method, this also applies to stability because of the prereqs for the matrix.

Experimental Results and Charts

- From our experiments, our results fairly agree with the qualitative backing provided earlier in the documentation.




Example #2 Error vs Iteration
1.20E+00
1.00E+00
8.00E-01
6.00E-01
4.00E-01
2.00E-01
0.00E+00
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41
Series1
Series2
Series3
Series4
Series5


Example #3 Error vs Step
5.00E-03
4.80E-03
4.60E-03
4.40E-03
4.20E-03
4.00E-03
3.80E-03
3.60E-03
3.40E-03
3.20E-03
3.00E-03
2.80E-03
2.60E-03
2.40E-03
2.20E-03
2.00E-03
1.80E-03
1.60E-03
1.40E-03
1.20E-03
1.00E-03
8.00E-04
6.00E-04
4.00E-04
2.00E-04
0.00E+00
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
jacobi example 3
Gauss-Seidel Example 3
SOR Method Example #3
Iterative Refinement Method
Conjugate Gradient Example #3

Conclusion

- In conclusion we can say that comparing the iterative methods, the most practical method is the PCG method, because of its low computational cost, high accuracy, and fast convergence. However, because a preconditioned matrix is usually required, problems will arise too creating this matrix. In regard to comparisons iterative refinement method is a good method for use but the odds of a matrix being well conditioned in application are very low.