A Model for Countercurrent Exchange Using Iterative Methods

Chernega, A; Anderson, J; Brown, J
Eastern Washington University

Objective

To model multistage chemical extraction via matrices by solving for mass fractions of target chemical in water stream at each reactor stage. Iterative methods will be applied contingent on the properties of the matrix.

Physical Background

Multistage chemical extraction exploits the physics of countercurrent exchange to transfer a target chemical from an aqueous solution to a target solvent. The countercurrent exchange occurs in a user-specified number of stages, based off of physical capacity of the reactor. Water and solvent are pumped at rates W [kg/hr] and S [kg/hr], respectively. At the initial stage the water must necessarily contain an initial mass fraction, x(in), of chemical. At the final stage the user can optionally pump solvent with a non-zero mass fraction, y(in), depending on the user's objective. Note that mass fractions and mass fraction ratios are unitless. We solve for the mass fraction of chemical in the solvent at every single stage, so that it may be monitored and/or extracted at target stages.

Presentation

Due to the nature of our physical setup, we will always be working with a weakly diagonally dominant matrix; fortunately, our iterative methods will still converge. Jacobi and SOR methods will be applied, with SOR sampling a range of w values. If the matrix happens to be symmetrically positive definite, conjugate gradient method will be applied as well. Extensive analysis of convergence rate will follow with discussion of optimal strategies of analysis. Finally, Gaussian elimination will be performed on well- and ill-conditioned matrices as a result of user input, with real-time feedback to warn the user if the matrix is indeed ill-conditioned.

Multistage Chemical Extraction Formula

The target matrix will be built around the following system of equations that model the chemical extractor:

$$-(W + Sm)x_1 + Smx_2 = -Wx_{in}$$
$$Wx_{i-1} - (W + Sm)x_i + Smx_{i+1} = 0 \quad \text{for} \quad (i = 2, 3, 4, \ldots, n-1)$$
$$Wx_{n-1} - (W + Sm)x_n = -Sy_{in}$$

- x_i = mass fraction of a chemical exiting the ith stage of extraction
- W = flow rate in the water stream
- S = flow rate in the solvent stream
- m = the ratio of th mass fraction of chemical in solvent stream / mass fraction of chemical in water stream
- x_{in} = the mass fraction in the input water stream
- y_{in} = the mass fraction in the input solvent stream

Defining Terms and Theorems

- Spectral radius: a square matrices' largest absolute eigenvalue. Note that the spectral radius of an n x n matrix is less than or equal to any natural norm.
- Let A be an n x n matrix, the following are equivalent:
    1. $p(A) < 1$
    2. $A^k \to 0$ as $k \to \infty$
    3. $A^k x \to 0$ as $k \to \infty$ for any x
- Strictly Diagonally Dominant (SDD) Matrices: A matrix in which the magnitude of its diagonal element is strictly larger than the sum of the absolute values in the same row, minus the diagonal.
- Symmetric Positive Definite (SPD) Matrix: a symmetric matrix whose leading principal submatrices determinant is greater than 0 and $x^TAx > 0$.

Mathematical Basis for Iterative Methods

Iterative methods like Jacobi and SOR rely on a fixed point iteration scheme to solve systems of the form $Ax = b$:

$$x^{(k+1)} = Tx^{(k)} + c$$

Matrix T and vector c are constructed based on relevant splitting methods (detailed below.) A critical requirement for these methods will consistently be that the spectral radius of T is less than 1.

Conditions for Convergence - Error Evolution

The error evolution equation for iterative methods will be explored as a basis for defining convergence conditions. Let:

$$e^{(k)} = x^{(k)} - x$$

Subtracting $x = Tx + c$ from the kth estimate of x yields:

$$e^{(k+1)} = x^{(k+1)} - x = Tx^{(k)} + c - (Tx + c) = T(x^{(k)} - x) = Te^{(k)} = T^2e^{(k-1)} = \dots = T^{k+1}e^{(0)}$$

Per theorems from slide 7, $e^{(k+1)} = T^{k+1}e^{(0)}$ will converge to 0 when the spectral radius of T is less than 1. The condition on spectral radius will further reinforce during discussion of rate of convergence

Convergence to Unique Solutions

Given the presence of a fixed point, the iteration scheme ultimately produces:

$$x = Tx + c \Rightarrow x - Tx = c \Rightarrow (I - T)x = c$$

Thus, the fixed point scheme will have a unique solution only if $(I - T)$ is nonsingular. A sufficient condition for $(I - T)$ to be nonsingular is that the spectral radius of T is less than 1 (see appendix for detailed proof).

Consistency of Splitting Methods with Ax = b

To show consistency, it must be true that $(I- T)^{-1}c = A^{-1}b$. The splitting methods discussed here define:

$$A = M - N = D - L - U$$

$$T = M^{-1}N,$$

$$c = M^{-1}b$$

Note: $I - T = I - M^{-1}N = M^{-1}(M - N) = M^{-1}A$.

Order of Convergence

We have established that $e^{(k+1)} = Te^{(k)}$. Taking the natural norm of both sides yields:

$$||e^{(k+1)}|| \leq ||T||\ ||e^{(k)}||$$

This inequality suggests a linear order of convergence with an asymptotic error constant less than or equal to T. Since the spectral radius of any matrix must be smaller than or equal to any natural norm for that matrix, the spectral radius of T can be used for a greatest lower bound on all natural matrix norms of T. For example, when the error constant is less than or equal to the spectral radius of T, which is less than or equal to the norm of T, ||T||. This justifies the use of the spectral radius as the asymptotic error constant.

Rate of Convergence

It was shown in the previous slide that the iterative methods being discussed converge linearly with asymptotic error constant (λ) equal to the spectral radius of T. It is known that the rate of convergence for linearly convergent methods is $O(\lambda^n)$, with λ representing the spectral radius of T in this case. This definition reinforces the condition that the spectral radius of T be less than 1. For $\lambda \geq 1$, the rate of convergence fails to approach 0 as n approaches infinity.

Jacobi Method

The Jacobi method requires no zeros on main diagonal to ensure nonsingularity of M. Convergence is guaranteed for SDD matrices (see Appendix for proof). Calculating matrices:

$$M = D$$
$$N = (L + U)$$
$$T = D^{-1}(L + U)$$
$$C = D^{-1} b$$

By examining the construction of T, it can be shown that the main diagonal elements are always zero and all other $T_{i,j} = -a_{i,j} / a_{i,i}$. It then follows that:

$$||T|| = \max_{1 \leq i \leq n} \sum |a_{i,j} / a_{i,i}| = \max_{1 \leq i \leq n} (1/|a_{i,i}|) * \sum |a_{i,j}|$$

Note: sums iterate for $1 \leq j \leq n$, $j \neq i$

Since $\sum |a_{i,j}| < |a_{i,i}|$ for SDD matrices, the infinity norm of T is less than 1. Finally, since the spectral radius of any matrix must be less than or equal to any natural matrix norm, the spectral radius of T must be less than 1.

Gauss - Seidel

The Gauss-Seidel method builds upon the Jacobi Method, using the vector $x^{(k+1)}$ as soon as it is calculated, rather than using the approximation vector for every row in the target matrix. Calculating matrices:

$$M = D - L$$
$$N = U$$
$$T = (D - L)^{-1}U$$
$$c = (D - L)^{-1}b$$

Note: Gauss - Seidel is not in the graphs because it's bounded between Jacobi and SOR, and not all that exciting.

Successive Over - Relaxation (SOR)

Successive Over-Relaxation (SOR) uses the relaxation parameter ω to weigh the two components of the iterating equation. Note that SOR becomes the Gauss-Seidel method when $\omega = 1$. Calculating matrices:

$$M = (1/\omega)D - L$$
$$N = [(1/\omega)-1]D + U$$
$$T = [(1/\omega)D - L]^{-1}([(1/\omega)-1]D + U)$$
$$c = [(1/\omega)D - L]^{-1}b$$

Finding the optimal ω can be done when A is positive definite and tridiagonal.
The equation is:

$$\omega = \frac{2}{1 + \sqrt{1 - [\rho(T_{jac})]^2}}$$

Conjugate Gradient

The conjugate gradient method is found via minimization of the functional of the form:

$$f(x) = \tfrac{1}{2}\mathbf{x}A\mathbf{x}^{\mathbf{T}} - \mathbf{b}^{T}\mathbf{x} + \mathbf{c}$$

Minimization yields:

$$\nabla f(x) = 0 = A\mathbf{x} - \mathbf{b}$$
$$0 = A\mathbf{x} - \mathbf{b}$$
$$A\mathbf{x} = \mathbf{b}$$

Therefore minimizing f(x) gives us to the solution of the equation A**x** = **b.** Through further analysis we find that the numerical iterative method for finding minimum is:

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \lambda_m\mathbf{d}^{(m)}$$

where λ_m is the step size and $\mathbf{d}^{(m)}$ is the search direction along the surface of the functional. With residual **r**, step size is minimal when:

$$\lambda_m = -(\ \mathbf{d}^{T(m)}\mathbf{r}^{(m)}\ ) / (\ \mathbf{d}^{T(m)}A\mathbf{d}^{(m)}\ )$$
$$\mathbf{r} = A\mathbf{x} - \mathbf{b}$$

Direction is chosen with:

$$\mathbf{d}^{(m+1)} = -\mathbf{r}^{(m+1)} + \alpha_m\mathbf{d}^{(m)}$$

where α is chosen so that $\mathbf{d}^{(m+1)}$ is A-conjugate to the direction $\mathbf{d}^{(m)}$. Note: two vectors, **u** and **v**, are A-conjugate if $\mathbf{u}^{T}A\mathbf{v} = 0$, where A is SPD. This method converges ridiculously fast if the target matrix is SPD; even with smaller matrices (smaller number of reactor stages) the method gets within tolerance at superior rates compared to Jacobi and SOR. We are guaranteed to converge within n iterations to the exact solution, provided we use exact arithmetic. Therefore solutions are wildly accurate and are only limited by the operating machines precision. Matrix *must* be SPD because that is the equivalent of a “positive” matrix, paralleled to a positive number in the reals. This ensures one unique local minimum. This extends to higher dimensions but can be visualized in 3D, as will be shown.

Analysis of Methods

All three methods were compared by relative error, absolute error, difference between an approximate solution and true solution, and by their behaviors as they converge to their respective convergence constants. SPD Matrix A was built with the following input information:

Water stream flow rate W = 200

Solvent stream flow rate S = 50

Mass fraction of chemical in input water stream $x_{in} = 0.075$

Mass fraction of chemical in input solvent stream $y_{in} = 0$

Ratio of mass fraction in streams m = 4

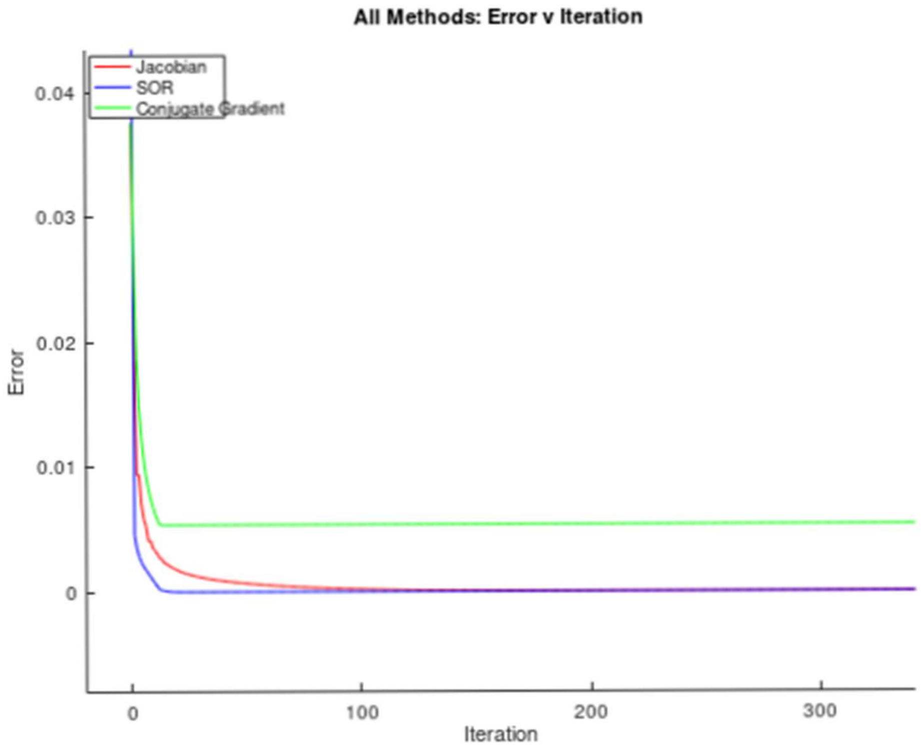
Number of stages n = 13

Error Tolerance = 5 x 10^{-12}

This will generate a tridiagonal 13x13 matrix with -400 along the center diagonal and 200 along the upper and lower diagonals, and a 13x1 **b** vector with $\mathbf{b}_1$ = 15 and $\mathbf{b}_1$ = 0 everywhere else. An optimal value of ω was used for SOR. After running the program, we perform the following analyses:

Relative Error

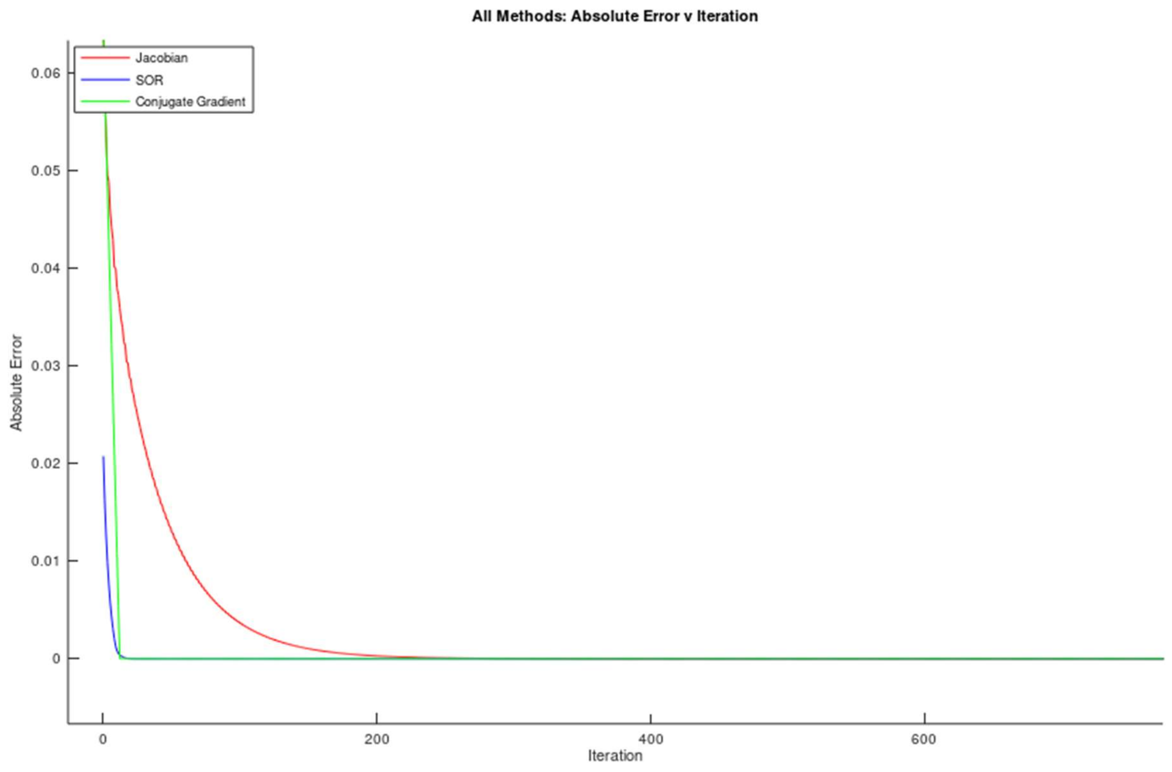
Calculated by finding the maximum of the absolute value of the difference between the current and previous approximations of **x**.



Comparing relative error, we see that all three methods converge rapidly to relatively low relative error, however Jacobi and SOR methods will never converge to the actual value. Note the unnatural curve of the conjugate gradient method; this is due to the method converging too drastically, meaning the penultimate approximation of **x** was far off of the true solution, but the final approximation of **x** was well within tolerance. The error relationship is clearer in the analysis of absolute error.

Absolute Error

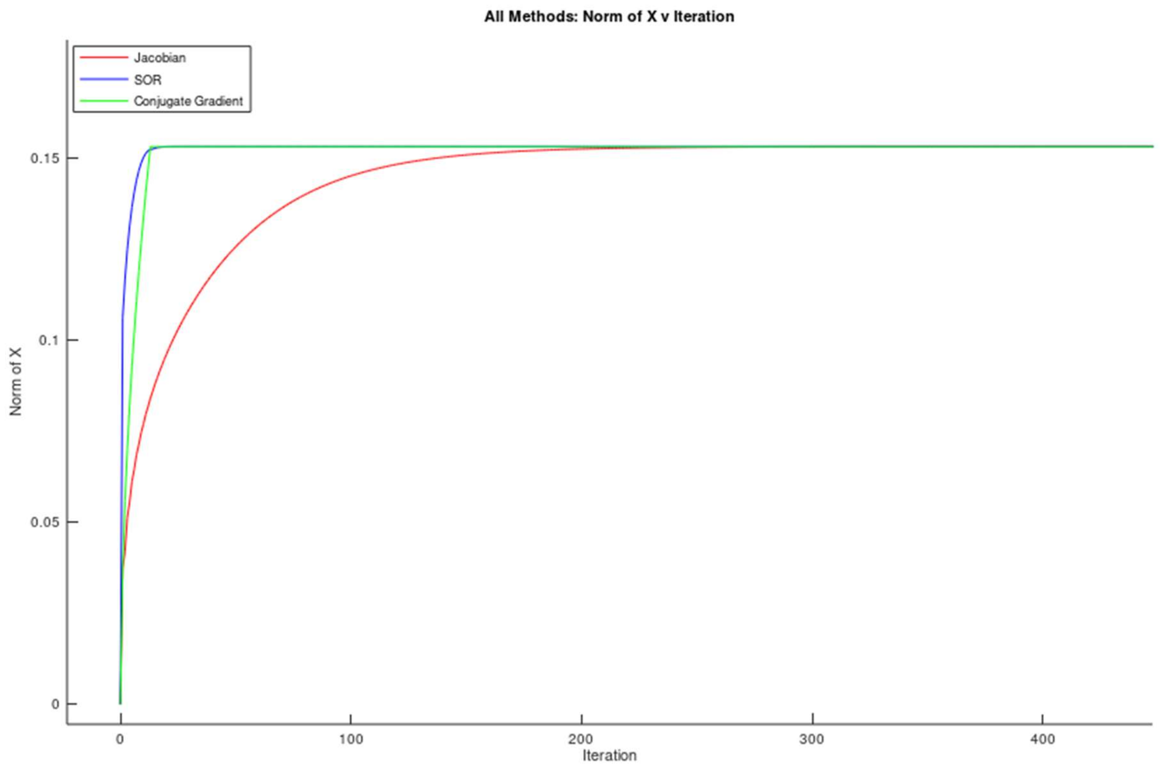
Calculated finding the maximum of the absolute value of the difference between the current approximation and true value of **x.**



It is clear from the graph that the conjugate gradient method converges to an absolute error of zero astronomically quickly. SOR offers similar performance but wastes a number of iterations converging absolute errors within our exceedingly tight tolerance, however it offers an excellent approximation with lower tolerances. Jacobi takes a significantly longer time to converge and is clearly least efficient. The difference between methods is drastic when iterating over vastly larger matrices, such as 100x100 or 1000x1000 matrices.

Norm of Current Approximation

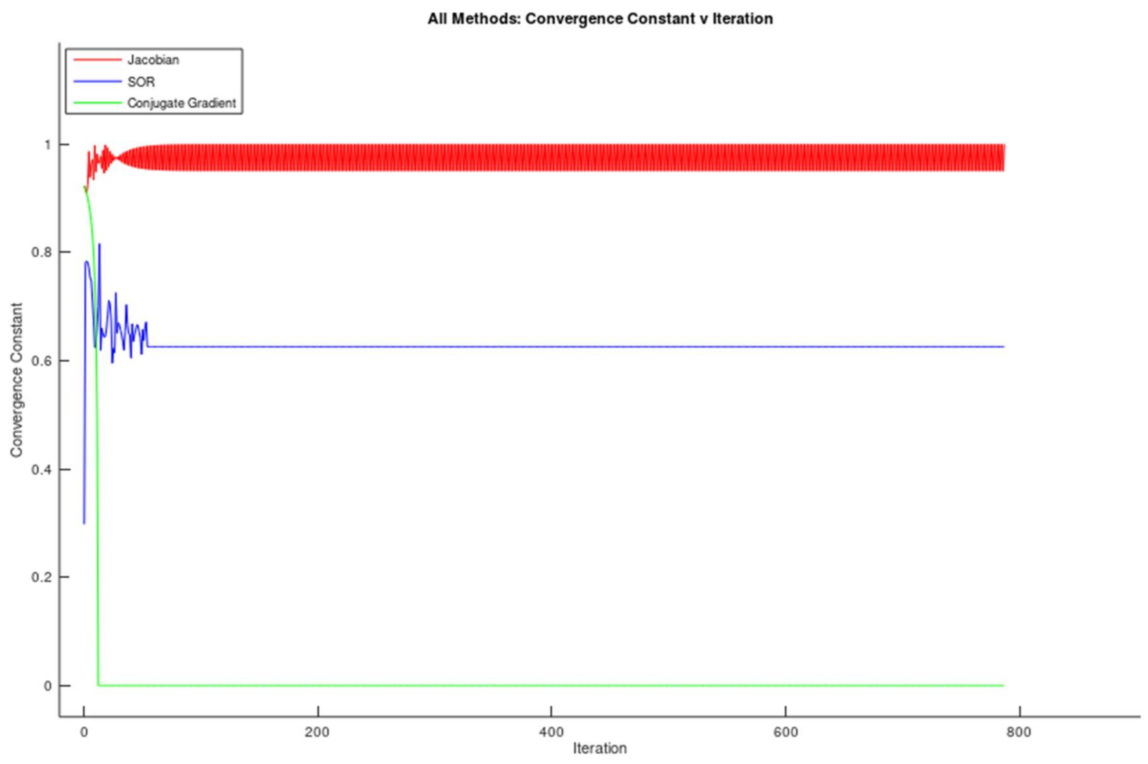
Calculated at each iteration with MatLab's built-in Euclidean norm function.



Fundamentally this is a similar graph to our comparison of absolute errors, but instead shows the Euclidean norm of the current approximation of **x** at each iteration. The primary advantage of this model is to highlight the evolution of approximations for the user.

Converge Constant

Calculated by subtracting the current approximation from the true value of **x**, finding the infinity norm, and dividing by a similar infinity norm subtraction of the *previous* approximation and true value of **x**.



From analysis of each method's convergence to their respective convergence constants, we are able discern the repeatable behaviors of each method, adding valuable insight to the interpretation of our program's results. The conjugate gradient method is cut and dry, rapidly converging to its expected convergence constant without any fluctuation. The SOR method, on the other hand, appears to fluctuate with random amplitude and frequency, which we believe to be a natural consequence of weighing our iteration equation with ω. Finally, analysis of the Jacobi method reveals a pattern that appears to follow hidden, definite rules. We were unable to discern exactly these rules however we concluded that we can extract the convergence constant by finding the average value of all data points from the harmonic portion of the wave. Interestingly, the behavior of the Jacobi curve appears to be dependent on whether the dimensions of the target matrix A are odd or even, as demonstrated:

Figure 1. Number of stages n = 13

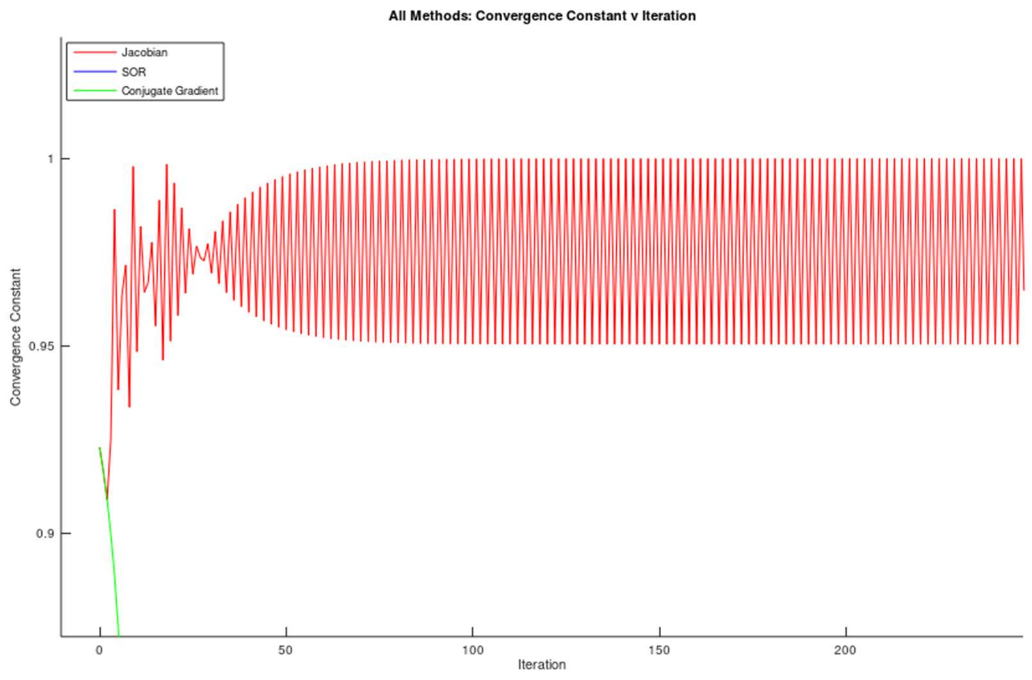
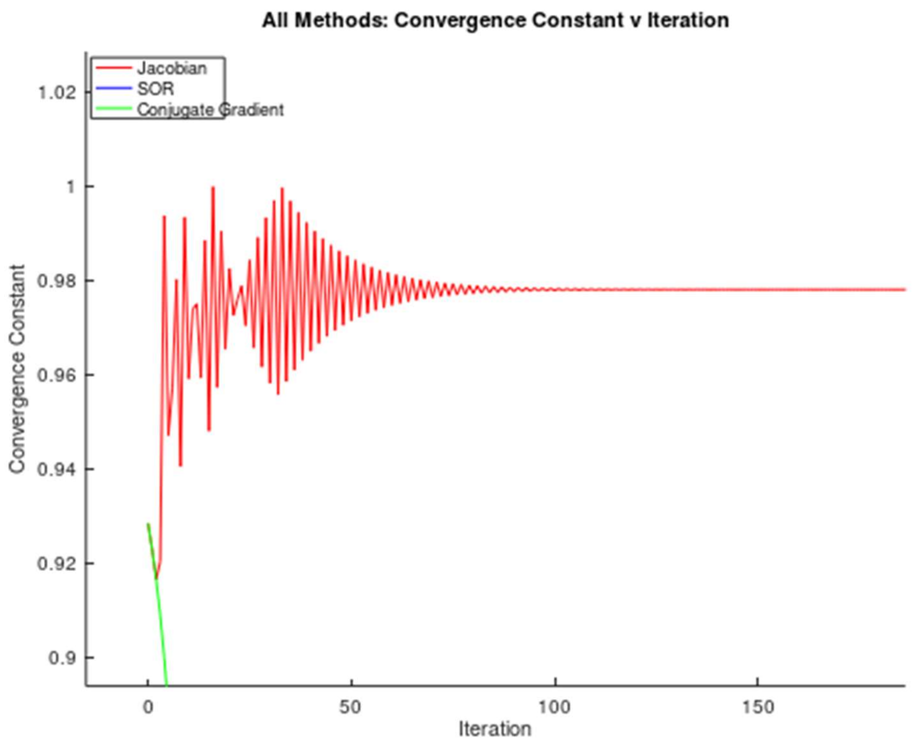


Figure 2. Number of stages n = 14



Conclusions

We conclude that the conjugate gradient method is absolutely the preferred method, provided the target matrix is SPD. The Jacobi and SOR methods converge to relative precision quickly with smaller number of stages but struggle to get within low tolerances. Error between iterations is small for Jac/SOR, but absolute error takes forever to reach zero. If the target matrix is not SPD, SOR offers significant efficiency over the Jacobi method. Analyzing the Jacobian method, we noted it produces interesting results; convergence constant behavior appears to be dependent on whether n is positive/negative, but the relationship breaks down the larger n is. Lastly, Gaussian elimination is appeared to be very accurate for approximating x even with larger conditio number values k(A), but the conjugate gradient is still the preferred method as it is less affected by ill-conditioning.

Pseudocode

Jacobi Method

```
while error > tolerance
    temp = xNew;
    xOld = temp;
    for row = 1:n
        sum1 = 0;
        sum2 = 0;
        for col = 1:n
            if col < row
                sum1 = sum1 + A(row, col) * xOld(col);
            end
            if col > row
                sum2 = sum2 + A(row, col) * xOld(col);
            end
        end
        xNew(row) = 1 / A(row, row) * [b(row) - sum1 - sum2]
    end
    error = max( |xNew - xOld| );
end
```

SOR Method

```
while error > tolerance
    for i=1:n
        for j= 1:i-1
            Lower = Lower + A(i,j) * $\mathbf{x}$New(j);
        end
        for j=i+1:n
            Upper = Upper + A(i,j) * $\mathbf{x}$Old(j);
        end
        $\mathbf{x}$New(i) = (1-$\omega$) * $\mathbf{x}$Old(i) + $\omega$*($\mathbf{b}$(i) - Lower - Upper) / A(i,i);
    end
    $\mathbf{x}$Old=$\mathbf{x}$New;
end
```

Conjugate Gradient Method

```
$r^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$
$\mathbf{d}^{(0)} = -\mathbf{r}^{(0)}$
$\delta^{(0)} = \mathbf{r}^{(0)T}\mathbf{r}^{(0)}$
For $m$ = 0, 1, 2,...
    $\mathbf{u} = A\mathbf{d}^{(m)}$
    $\lambda_m = \delta^{(m)} / \mathbf{d}^{(m)T}\,\mathbf{u}$
    $\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \lambda_m\mathbf{d}^{(m)}$
    $\mathbf{r}^{(m+1)} = \mathbf{r}^{(m)} + \lambda_m\mathbf{u}$
    set $\delta^{(m+1)} = \mathbf{r}^{(m+1)T}\,\mathbf{r}^{(m+1)}$
    If sqrt($\mathbf{r}^{(m+1)T}\mathbf{r}^{(m+1)}$) < TOL, OUTPUT $\mathbf{x}^{(m+1)}$
        $\alpha_m = \delta^{(m+1)} / \delta^{(m)}$
        $\mathbf{d}^{(m+1)} = -\mathbf{r}^{(m+1)} + \alpha_m\mathbf{d}^{(m)}$
    end
end
```

Appendix of Proofs

Guarantee of unique solutions of iterative methods via nonsingularity of (I - T):
We prove (I - T) is nonsingular provided spectral radius of T is less than 1.
Proof by contradiction: let spectral radius of T be less than 1 and assume (I - T) is singular. If this is true, there must be some nonzero vector x such that (I -T)x = 0:

$$(I - T)x = 0$$
$$\Rightarrow Ix - Tx = 0$$
$$\Rightarrow Tx = x$$

This means that 1 is an eigenvalue of T, which is a contradiction of the provided assumption that the spectral radius of T is less than 1. Thus, (I - T) is nonsingular when the spectral radius of T is less than 1 ■

Guaranteed Convergence of the Jacobi Method with SDD Matrices

After splitting matrix A into matrices D, L, and U for the creation of matrices M = D and N = (L + U), T is constructed according to $T = D^{-1} * (L + U)$. Since (L + U) will have zeros along the main diagonal and D will have zeros everywhere but the main diagonal, T will also have zeros along the main diagonal based on matrix multiplication. Further examination of T shows that for each element of T:

$$T_{i,j} = - A_{i,j} / A_{i,i}$$

It then follows that the infinity norm of $T = \max_{1 \leq i \leq n} \sum |A_{i,j} / A_{i,i}| = \max_{1 \leq i \leq n} (1/|A_{i,i}|) * \sum |A_{i,j}|$ (Note: Sums iterate for $1 \leq j \leq n$, $j \neq i$).

Since $\sum |A_{i,j}| < |A_{i,i}|$ for SDD matrices, the infinity norm of T is less than 1.

Finally, since the spectral radius of any matrix must be less than or equal to any natural matrix norm, the spectral radius of T must be less than 1 and the Jacobi method is guaranteed to converge ■

References

Bradie, B. (2006). *A Friendly Introduction to Numerical Analysis.* Upper Saddle River, NJ: Pearson Education, Inc.

Taroudaki, V. (2019). *Gaussian.m*. [online] Canvas.ewu.edu. Available at: https://canvas.ewu.edu/courses/1252663/files/folder/Matlab [Accessed 17 Mar. 2019].