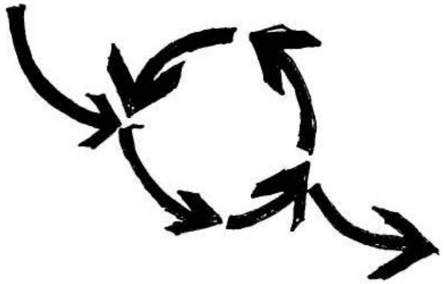
The Successive Over-Relaxation Method:

Developing the Method and Exploring Modern Applications

John Anthony Bowllan, Josh Rubin, Robert Bernhardt

Middlebury College, Spring 2017



$Ax = b$: a step in a larger process

Motivating Problem

Problem

Given A (SPD) and vector b, efficiently and accurately solve the system

$$Ax = b$$

- Given a large system, Gaussian Elimination is impractical
- FOR: Given $\rho > 0$, solve new iterates by $x^{(n+1)} = x^{(n)} + \frac{1}{\rho} r^{(n)}$.
    - $\rho_{opt} = \frac{\lambda_{max} + \lambda_{min}}{2}$
- FOR inefficient in practical applications

Goal

Build accurate and more efficient methods for solving $Ax = b$.

New approach: different matrix decomposition!

Matrix Decomposition

★ Matrix decomposition: $A = M - N$
where $M = \rho I$, $N = \rho I - A$

★ Different matrix decomposition from now on: $\mathrm{A} = \mathrm{D} - \mathrm{L} - \mathrm{U}$
where D diagonal, L lower triangular and U upper triangular

FOR

- ❑ Algorithm: $Mx^{(n+1)} = Nx^{(n)} + b$

Jacobi Method → Gauss-Seidel Method

Jacobi Method

- ❑ Algorithm: $x_{JAC}^{(n+1)} = T_{JAC}\ x_{JAC}^{(n)} + b_{JAC}$

  where $T_{JAC} = D^{-1}(L+U)$, $b_{JAC} = D^{-1}b$

Gauss-Seidel Method

- ❑ Algorithm: $x_{GS}^{(n+1)} = T_{GS}\ x_{GS}^{(n)} + b_{GS}$

  where $T_{GS} = (D-L)^{-1}U$, $b_{GS} = (D-L)^{-1}b$

Road to SOR

1) Jacobi Method
2) Gauss-Seidel Method
3) SOR Method

Jacobi and Gauss-Seidel Methods

Precursor Methods of SOR

Jacobi Algorithm

Gauss-Seidel Algorithm

Component Forms

$$
\begin{aligned}
x_1^{(n+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(n)} - a_{13}x_3^{(n)} - \cdots - a_{1n}x_n^{(n)}),\\
x_2^{(n+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(n)} - a_{23}x_3^{(n)} - \cdots - a_{2n}x_n^{(n)}),\\
&\vdots\\
x_n^{(n+1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(n)} - a_{n2}x_2^{(n)} - \cdots - a_{n,n-1}x_{n-1}^{(n)})
\end{aligned}
$$

$$
\begin{aligned}
x_1^{(n+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(n)} - a_{13}x_3^{(n)} - \cdots - a_{1n}x_n^{(n)})\\
x_2^{(n+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(n+1)} - a_{23}x_3^{(n)} - \cdots - a_{2n}x_n^{(n)})\\
&\vdots\\
x_n^{(n+1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(n+1)} - a_{n2}x_2^{(n+1)} - \cdots - a_{n,n-1}x_{n-1}^{(n+1)})
\end{aligned}
$$

- ❏ Only uses components of previous iterate

- ❏ Uses most recent approximations of the *x* components.
- ❏ Faster convergence than Jacobi method

SOR - A Weighted Average:

SOR Algorithm

$$x_{SOR}^{(n+1)} = \omega x_{GS}^{(n+1)} + (1-\omega)x_{SOR}^{(n)}$$

where ω is the weighting factor.

$$x^{(n+1)} = (D-\omega L)^{-1}(\omega U + (1-\omega)D)x^{(n)} + (D-\omega L)^{-1}\omega b$$

- Improves upon Gauss-Seidel
- Altering values of ω alters weights of two iterates
- $\omega > 1$ over-relaxation, $\omega < 1$ under-relaxation

Convergence of SOR

★ Must place **restrictions** on ω for convergence

★ Following Lemma required to prove convergence theorem

Lemma

Let $A \in R^{n \times n}$.

Then $\lim_{k \to \infty} A^k = 0_{n \times n}$ if and only if $\rho(A) < 1$.

Theorem of SOR Convergence

SOR converges for $\omega \in (0, 2)$ and diverges otherwise

Illustration of Convergence Criteria

$$A = \begin{bmatrix} 3 & -2 & 5 \\ 4 & -7 & -1 \\ 5 & -6 & 4 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 19 \\ 13 \end{bmatrix}, x_{\text{TRUE}} = \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}, x_{\text{INITIAL}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, tol = 10^{-6}$$

Omega = 1.5, Convergence within tol in 95 iterations

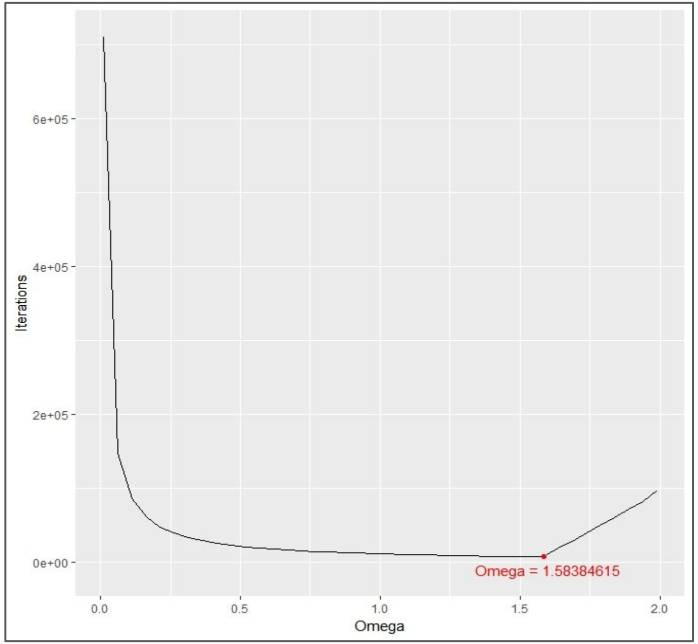
Iterations	x_1	x_2	x_3
1	0.6667	-2.3333	-1.0833
2	1.0417	-1.8869	-0.8824
3	0.7984	-2.1851	-1.0255
4	0.9796	-1.9713	-0.9314
$\vdots$	$\vdots$	$\vdots$	$\vdots$
94	0.9999	-2.0000	-1.0000
95	1	-2	-1

Omega = -1, Convergence not achieved

Iterations	x_1	x_2	x_3
1	0.6667	-2.3333	-1.0833
2	0.4167	-2.6310	-1.2173
3	-0.1081	-3.2595	-1.5041
4	-1.2167	-4.5913	-2.1166
$\vdots$	$\vdots$	$\vdots$	$\vdots$
94	-6.2120×10^{29}	-7.4850×10^{29}	-3.4625×10^{29}
95	-1.3205×10^{30}	-1.5911×10^{30}	-0.7360×10^{30}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
872	-3.8798×10^{284}	-4.6749×10^{284}	-2.1626×10^{284}
873	-8.2473×10^{284}	-9.9374×10^{284}	-4.5969×10^{284}
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Apply SOR to 1DMPP

MATLAB Demonstration



Iterative Performance of the SOR algorithm for $\omega \in (0, 2)$

Conclusions

- $\omega_{optimal} \approx 1.6$
  approximately 7,800 iterations
- Choice of ω crucial for iterative efficiency
- *A* SPD, formula for $\omega_{optimal}$ is

$$\omega_{optimal} = \frac{2}{1 + \sqrt{1 - (spr(T_{JAC}))^2}}$$

- In practice, $\omega_{optimal}$ attained by trial and error

Next Step (Part 2): Conjugate Gradient Method

Conjugate Gradient Method

- ❑ Global optimization as opposed to local minimization of Steepest Descent
- ❑ Converges in N steps in exact arithmetic

- SOR and Symmetric Successive Over-Relaxation (SSOR) used as preconditioners for CG.
- Preconditioner lowers condition number, increasing iterative efficiency.
- SOR to increase iterative efficiency of CG when dimension is large.

Performance of Iterative Methods on 1DMPP Problem	
Iterative Method	Iterations for Convergence
SOR ($\omega_{optimal} \approx 1.6$)	7,809
Gauss-Seidel	11,860
Jacobi	22,277
FOR ($\rho = 2$)	37,574
2-step FOR ($\rho_1 = 1, \rho_2 = 3$)	14,088
Steepest Descent	38,214
Conjugate Gradient	100

References

- Kendall Atkinson. *Introduction to Numerical Analysis*. Wiley, 2 edition, 1989.
- Brian Bradie. *A Friendly Introduction to Numerical Analysis*. Pearson, 1 edition, 2005.
- W. Kahan. *Numerical Linear Algebra*. Canadian Mathematical Society, 2 edition, 1966.
- Abdelwahab Kharab and Ronald B. Guenther. *An Introduction to Numerical Methods, A MATLAB Approach*. CRC Press, 3 edition, 2011.
- Rainer Kress. *Graduate Texts in Mathematics, Numerical Analysis*. Springer, 1 edition, 1998.
- William Layton and Myron Sussman. *Numerical Linear Algebra*. Lulu, 2014. https://www.lulu.com/spotlight/Layton_Sussman.
- MathWorks. MATLAB. Version R2017a. https://www.mathworks.com

Extra Information (Matrix Derivations)

Jacobi Method

$$
\begin{aligned}
(D - L - U)x &= b \\
Dx - Lx - Ux &= b \\
Dx &= Lx + Ux + b \\
x^{(n+1)} &= D^{-1}(L + U)x^{(n)} + D^{-1}b
\end{aligned}
$$

Gauss-Seidel Method

$$
\begin{aligned}
(D - L - U)x &= b \\
Dx - Lx - Ux &= b \\
Dx - Lx &= Ux + b \\
x^{(n+1)} &= (D - L)^{-1}Ux^{(n)} + (D - L)^{-1}b
\end{aligned}
$$

SOR Method

$$
\begin{aligned}
\omega(D - L - U)x &= \omega b \\
\omega Dx - \omega Lx - \omega Ux &= \omega b \\
(\omega D + D - D)x - \omega Lx - \omega Ux &= \omega b \\
(D - \omega L)x &= (\omega U + (1 - \omega)D)x + \omega b \\
x^{(n+1)} &= (D - \omega L)^{-1}(\omega U + (1 - \omega)D)x^{(n)} + (D - \omega L)^{-1}\omega b
\end{aligned}
$$