Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Français
Italiano
日本語
Русский
Edit links

Successive over-relaxation

From Wikipedia, the free encyclopedia

In numerical linear algebra, the method of **successive over-relaxation** (**SOR**) is a variant of the Gauss–Seidel method for solving a linear system of equations, resulting in faster convergence. A similar method can be used for any slowly converging iterative process.

It was devised simultaneously by David M. Young, Jr. and by H. Frankel in 1950 for the purpose of automatically solving linear systems on digital computers. Over-relaxation methods had been used before the work of Young and Frankel. An example is the method of Lewis Fry Richardson, and the methods developed by R. V. Southwell. However, these methods were designed for computation by human calculators, and they required some expertise to ensure convergence to the solution which made them inapplicable for programming on digital computers. These aspects are discussed in the thesis of David M. Young, Jr.[1]

Contents [hide]

1 Formulation
2 Algorithm
3 Symmetric successive over-relaxation
4 Other applications of the method
5 See also
6 Notes
7 References
8 External links

Formulation [edit]

Given a square system of *n* linear equations with unknown **x**:

$$A\mathbf{x} = \mathbf{b}$$

where:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Then *A* can be decomposed into a diagonal component *D*, and strictly lower and upper triangular components *L* and *U*:

$$A = D + L + U,$$

where

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

The system of linear equations may be rewritten as:

$$(D + \omega L)\mathbf{x} = \omega\mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}$$

for a constant $\omega > 1$, called the *relaxation factor*.

The method of successive over-relaxation is an iterative technique that solves the left hand side of this expression for **x**, using previous value for **x** on the right hand side. Analytically, this may be written as:

$$\mathbf{x}^{(k+1)} = (D + \omega L)^{-1}\big(\omega\mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}^{(k)}\big) = L_\omega \mathbf{x}^{(k)} + \mathbf{c},$$

where $\mathbf{x}^{(k)}$ is the *k*th approximation or iteration of $\mathbf{x}$ and $\mathbf{x}^{(k+1)}$ is the next or *k* + 1 iteration of $\mathbf{x}$. However, by taking advantage of the triangular form of (*D*+*ωL*), the elements of $\mathbf{x}^{(k+1)}$ can be computed sequentially using forward substitution:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$

The choice of relaxation factor *ω* is not necessarily easy, and depends upon the properties of the coefficient matrix. In 1947, Ostrowski proved that if A is symmetric and positive-definite then $\rho(L_\omega) < 1$ for $0 < \omega < 2$. Thus convergence of the iteration process follows, but we are generally interested in faster convergence rather than just convergence.

Algorithm [edit]

Since elements can be overwritten as they are computed in this algorithm, only one storage vector is needed, and vector indexing is omitted. The algorithm goes as follows:

Inputs: *A*, *b*, *ω*
Output: ϕ

```
Choose an initial guess $\phi$ to the solution
repeat until convergence
    for i from 1 until n do
        $\sigma \leftarrow 0$
        for j from 1 until n do
            if j ≠ i then
                $\sigma \leftarrow \sigma + a_{ij}\phi_j$
            end if
        end (j-loop)
        $\phi_i \leftarrow (1-\omega)\phi_i + \frac{\omega}{a_{ii}}(b_i - \sigma)$
    end (i-loop)
    check if convergence is reached
end (repeat)
```

Note:

$(1-\omega)\phi_i + \frac{\omega}{a_{ii}}(b_i - \sigma)$ can also be written $\phi_i + \omega\left(\frac{b_i - \sigma}{a_{ii}} - \phi_i\right)$, thus saving one multiplication in each iteration of the outer *for*-loop.

Symmetric successive over-relaxation [edit]

The version for symmetric matrices *A*, in which

$$U = L^T,$$

is referred to as **Symmetric Successive Over-Relaxation**, or (**SSOR**), in which

$$P = \left(\frac{D}{\omega} + L\right) \frac{1}{2-\omega} D^{-1} \left(\frac{D}{\omega} + U\right),$$

and the iterative method is

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma^k P^{-1}(A\mathbf{x}^k - \mathbf{b}),\ k \geq 0.$$

The SOR and SSOR methods are credited to David M. Young, Jr..

Other applications of the method [edit]

Main article: Richardson extrapolation

A similar technique can be used for any iterative method. If the original iteration had the form

$$x_{n+1} = f(x_n)$$

then the modified version would use

$$x_{n+1}^{\mathrm{SOR}} = (1-\omega)x_n^{\mathrm{SOR}} + \omega f(x_n^{\mathrm{SOR}}).$$

Note however that the formulation presented above, used for solving systems of linear equations, is not a special case of this formulation if x is considered to be the complete vector. If this formulation is used instead, the equation for calculating the next vector will look like

$$\mathbf{x}^{(k+1)} = (1-\omega)\mathbf{x}^{(k)} + \omega L_*^{-1}(\mathbf{b} - U\mathbf{x}^{(k)}).$$

Values of $\omega > 1$ are used to speed up convergence of a slow-converging process, while values of $\omega < 1$ are often used to help establish convergence of a diverging iterative process or speed up the convergence of an overshooting process.

There are various methods that adaptively set the relaxation parameter ω based on the observed behavior of the converging process. Usually they help to reach a super-linear convergence for some problems but fail for the others.

See also [edit]

- Jacobi method
- Gaussian Belief Propagation
- Matrix splitting

Notes [edit]

1. ^ Young, David M. (May 1, 1950), *Iterative methods for solving partial difference equations of elliptical type* (PDF), PhD thesis, Harvard University, retrieved 2009-06-15

References [edit]

- This article incorporates text from the article Successive_over-relaxation_method_-_SOR on CFD-Wiki that is under the GFDL license.

- Abraham Berman, Robert J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, 1994, SIAM. ISBN 0-89871-321-8.
- Black, Noel and Moore, Shirley, "Successive Overrelaxation Method", *MathWorld*.
- Yousef Saad, *Iterative Methods for Sparse Linear Systems*, 1st edition, PWS, 1996.
- Netlib's copy of "Templates for the Solution of Linear Systems", by Barrett et al.
- Richard S. Varga 2002 *Matrix Iterative Analysis*, Second ed. (of 1962 Prentice Hall edition), Springer-Verlag.
- David M. Young, Jr. *Iterative Solution of Large Linear Systems*, Academic Press, 1971. (reprinted by Dover, 2003)

External links [edit]

- Module for the SOR Method
- Tridiagonal linear system solver based on SOR, in C++

v · t · e	Numerical linear algebra [hide]
Key concepts	Floating point · Numerical stability
Problems	Matrix multiplication (algorithms) · Matrix decompositions · Linear equations · Sparse problems
Hardware	CPU cache · TLB · Cache-oblivious algorithm · SIMD · Multiprocessing
Software	BLAS · Specialized libraries · General purpose software

Categories: Numerical linear algebra | Relaxation (iterative methods)