数理解析研究所講究録
第 673 巻 1988 年 22-39

Some Experiments on New ACRITH
- Self-validating SOR Algorithm -

Yoshihiro Tanamachi(棚町 芳弘)
Akira OKuda (奥田 晃)
IBM Japan Ltd., Tokyo

Abstract:

The successive overrelaxation method(SOR) is well known as a fundamental method of iteratively solving a large sparse linear system of equations. Obviously, however, the method by itself cannot give a guaranteed solution. We have developed a simple and practical self-validating SOR algorithm which gives a guaranteed interval solution, using new release of ACRITH, IBM's product for high-accuracy computation.

In this paper, we will describe our algorithm and programming considerations for its implementation, and give the results of some numerical experiments. At the stage of interval SOR iteration of our algorithm, we restrict the acceleration parameter to 1.0 in order to maintain numerical stability,though at first sight it seems better to do overrelaxation. Further study and experiment is needed to discover how to accelerate convergence at this stage.

Contents:

1. Introduction

The successive overrelaxation method(SOR) is well known as a basic, indirect, and iterative method of solving large sparse linear systems such as those derived from finite difference approximations of elliptic differential equations. It gives very good approximate solutions for the linear systems, but in general, it is not known quantitatively how good they are.

In this paper,we propose a self-validating SOR algorithm that gives an approximate solution in the form of a 'guaranteed interval', an interval including the exact solution.

W. F. Ames and R. C. Nicklas ([1]) gave numerical examples where all or nearly all the components of a converged solution by the usual SOR method (point SOR), are not included in the guaranteed interval solution gained by the SOR method using interval arithmetic (interval SOR). However, those examples are not as strange as they appear, because there is no way in witch the point SOR method can guarantee the exact solution location.
This is the reason why we need the self-validating SOR algorithm.

The above mentioned authors state that they developed the interval SOR method with an acceleration factor ω=1.0, utilizing IBM's product ACRITH, thouth its algorithm description is not shown in the article.

Motivated by this article, we have developed an interval SOR algorithm using the newest release of ACRITH (release 3), and did some numerical experiments. In section 2, we describe our algorithm and programming considerations. Numerical examples are given in section 3, and the interval SOR acceleration is discussed in section 4. The appendix is a very brief introduction to ACRITH.

2. Self-validating SOR algorithm

Let a system of linear equations below be given.

$Ax=b$, $A=(a_{ij})$; n by n square matrix

We use the following notations:

- Scalar variable: ω, x_i, a_{ij}
- Vector variable: b, q, x, y, $\tilde{x}$
- Matrix variable: A, D, I, L, U
- Interval data type: $[\varepsilon]$, $[x]$, $[A]$; interval data are specified by the brackets, and the lower and upper bounds of the interval are specified by superscripts l and u respectively, as $[x]=[x^l, x^u]$.

2.1 Algorithm description

Our algorithm consists of two stages:

(1) Point SOR stage (PSOR)

An approximate solution $\tilde{x}$ is obtained by the point SOR method with a given optimal acceleration factor ω. Usually, starting vector is zero.

(2) Interval SOR stage (ISOR)

Taking the above $\tilde{x}$ as the initial interval vector, a guaranteed solution $[\hat{x}]$ is obtained using interval arithmetic with an acceleration factor of 1.0 (i.e. of the Gauss-Seidel type).

Figure 1 shows the overall flow, and Figure 2 describes the ISOR algorithm in pseudo-code form.

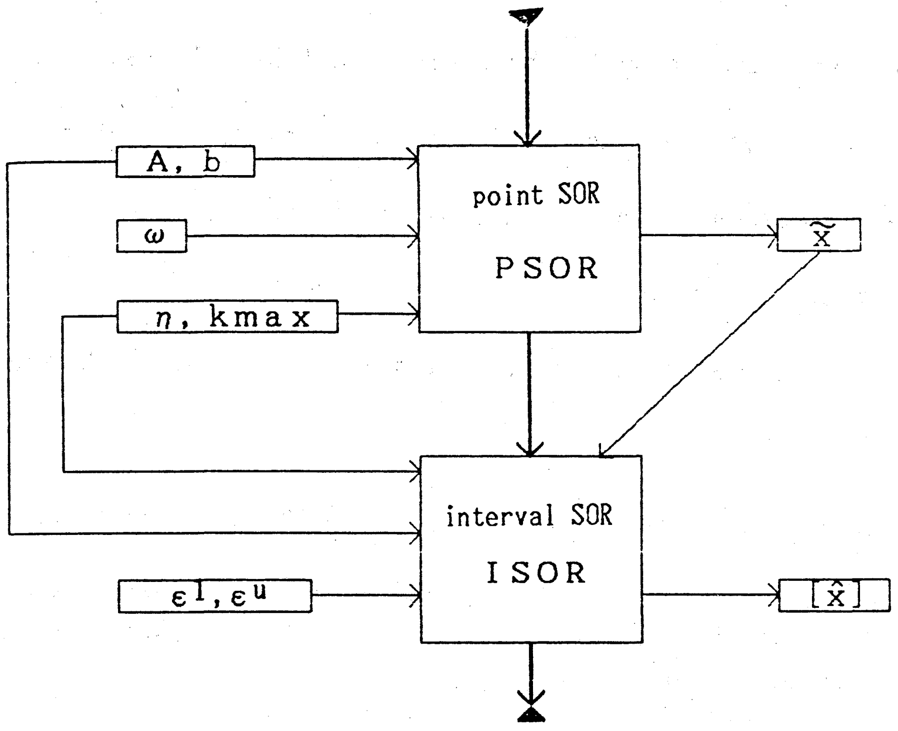


Figure 1. Overall flow

```
$[x^{(0)}] \leftarrow [\tilde{x}, \tilde{x}]$
 do   k=0, kmax
    do  i=1, n
      $[y_i] \leftarrow \{b - \sum_{j>i} a_{ij} \times [x_j^{(k+1)}] - \sum_{j<i} a_{ij} \times [x_j^{(k)}]\} / a_{ij}$ (※)
         if ( $[y_i] \subseteq [x_i^{(k)}]$ ) then
              $[x_i^{(k+1)}] \leftarrow [y_i]$
         else
              $[x_i^{(k+1)}] \leftarrow [y_i] \times [\varepsilon]$
         endif
    enddo (i)
    if( $[y_i] \subseteq [x_i^{(k)}]$ for all i) then
       if( $\max_i (y_i^u - y_i^l) < \eta$ ) then
             'converged'
              return  ( $[\hat{x}] \leftarrow [x^{(k+1)}]$ )
         endif
    endif
 enddo (k)
 'not  converged'
 return
```

Figure 2.   ISOR algorithm

[Notes]

- η is the convergence criterion for both PSOR and ISOR. In PSOR, it is used as the criterion for the maximum norm of the difference of two successive iteration vectors. In ISOR, it is used as the criterion for interval width.
- kmax is the maximum number of iterations permitted, both for PSOR and ISOR.
- In ISOR,the calculated interval$[y_i]$is replaced by the inflated interval $[y_i]\times[\varepsilon]$, where $[\varepsilon]=[\varepsilon^l, \varepsilon^u]$, $\varepsilon^l<1.0$, $\varepsilon^u> 1.0$, in order to reach a guaranteed interval sooner.

The algorithm is practical in the sense that we simply apply the usual SOR method with an optimal acceleration factor, and then we can get guaranteed interval solution with additional interval arithmetic computation. In the ISOR stage, we always use the acceleration parameter 1.0 to maintain numerical stability. The parameter choice is discussed in section 4.

2.2 Programming considerations

The key to the computation in our algorithm is the statement specified by symbol (*) in Fig. 2, where high-accuracy arithmetic is needed. Here we use ACRITH subroutines: DIDOT for the interval inner product, and DIDIV for the division of interval data by a scalar value.

We have also tried to use the 'interval accumulator', one of the new functions in release 3, for direct control of interval calculation. Although this function works well, we have not implemented it in our final program, because the subroutine DITOT function is enough for our purpose.

Another subroutine for arithmetic expression evaluation, FVAL, seemed useful at first sight, but it is not appropriate in our case, as it accepts only point data as input. We would need to enhance FVAL to be able to specify interval data as input.

3. Numerical examples

For our numerical experiments, we used the same problem as Armes and Nicklas, the example problem IV in [1].

[Problem] Poisson equation with Dirichlet boundary condition.

$-(u_{xx}+u_{yy}) = \exp[-(x-1/2)^2-(y-1/2)^2]$, $0<x<1$, $0<y<1$, $u=0$ on boundary

The linear system of equations is derived by five-point difference approximation, dividing the unit square into N×N subsquares.

Armes and Nicklas showed that the point SOR convergent solution does not lie in the guaranteed interval solution gained by their interval SOR method for all inner mesh points. Part of their results are as follows:

mesh point	x	y	point SOR	interval SOR	
				(lower)	(upper)
(1,1)	0.125	0.125	0.15150487E-1	0.151518E-1	0.151520E-1
(4,4)	0.500	0.500	0.67912579E-1	0.679184E-1	0.679192E-1

[Computational conditions]

- N=8
- acceleration factor in point SOR $\omega=1.0$
- $\eta=1.0E-4$

Our results obtained by the algorithm in Figures 1 and 2 are shown in Figure 3.

```
    INTERVAL SOR ALGORITHM
     FOR MODEL PROBLEM
 * PARAMETERS:
          NUMBER OF INNER GRIDS =    7 * 7
          RELAXATION FACTOR ω   =    1.0000
          CONVERGENCE CRITERIA  =    0.10000000D-09
          EPSILON INFLATION     =    0.99999900D-00  0.10000010D+01

 * RESULTS    :

   (I,J)                 LOWER                          UPPER

  (1) DIRECT
      (1,1)  (    0.1515980974458414840-01 ,     0.1515980974458414940-01)
      (4,4)  (      0.67952329636803870-01 ,       0.67952329636803900-01)

  (2) ISOR    :IT =  47 (ω= 1.0000)
      (1,1)  (            0.1515980973D-01 ,              0.1515980976D-01)
      (4,4)  (            0.6795232959D-01 ,              0.6795232969D-01)

  (3) PSOR    :IT = 119 (ω= 1.0000)
      (1,1)               0.151598096233563255D-01
      (4,4)               0.679523291220316822D-01
```

Figure 3. Results for $\omega=1.0$

[Output description]

- PARAMETERS
  - RELAXATION FACTOR ω: the acceleration factor for PSOR; this value is also printed in the '(3) PSOR' line.
  - CONVERGENCE CRITERIA: the value of η described in [Notes] of §2.1.
  - EPSILON: the values of ε^l and ε^u described in [Notes] of §2.1.
- RESULTS
  - DIRECT: the guaranteed interval obtained by ACRITH subroutine DSSSB, a direct solving routine for band linear system, one of new functions of release 3.
  - IT: the number of iterations needed for convergence.
- The lower and upper bounds of the resultant interval are printed out using ACRITH subroutine DIOUT. DIOUT gives the minimum decimal interval that includes the inner hexadecimal interval, and shows the significant digits of decimal lower and upper bounds as long as there is a difference between them.
- All computations are executed in long precision.

Figure 4 is the result for ω=1.4465, the optimal acceleration factor in the case of N=8 in the model problem. The optimal ω is given by the formula below ([4]).

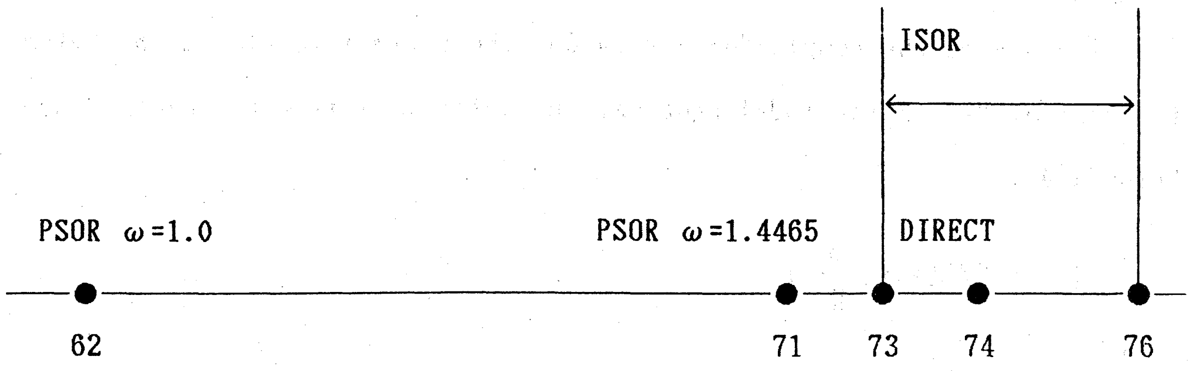
$$\omega_{opt} = 2/(1+\sin\frac{\pi}{N})$$

```
    INTERVAL SOR ALGORITHM
     FOR MODEL PROBLEM
* PARAMETERS:
          NUMBER OF INNER GRIDS =   7 * 7
          RELAXATION FACTOR ω   =   1.4465
          CONVERGENCE CRITERIA  =   0.10000000D-09
          EPSILON INFLATION     =   0.99999900D+00  0.10000010D+01

* RESULTS   :

  (I,J)                  LOWER                          UPPER

  (1) DIRECT
      (1,1)  (     0.151598097445841484D-01 ,     0.151598097445841494D-01)
      (4,4)  (       0.6795232963680387D-01 ,       0.6795232963680390D-01)

  (2) ISOR   :IT = 47 (ω= 1.0000)
      (1,1)  (             0.1515980973D-01 ,             0.1515980976D-01)
      (4,4)  (             0.6795232959D-01 ,             0.6795232969D-01)

  (3) PSOR   :IT = 32 (ω= 1.4465)
      (1,1)                0.151598097180568525D-01
      (4,4)                0.679523296184764564D-01
```

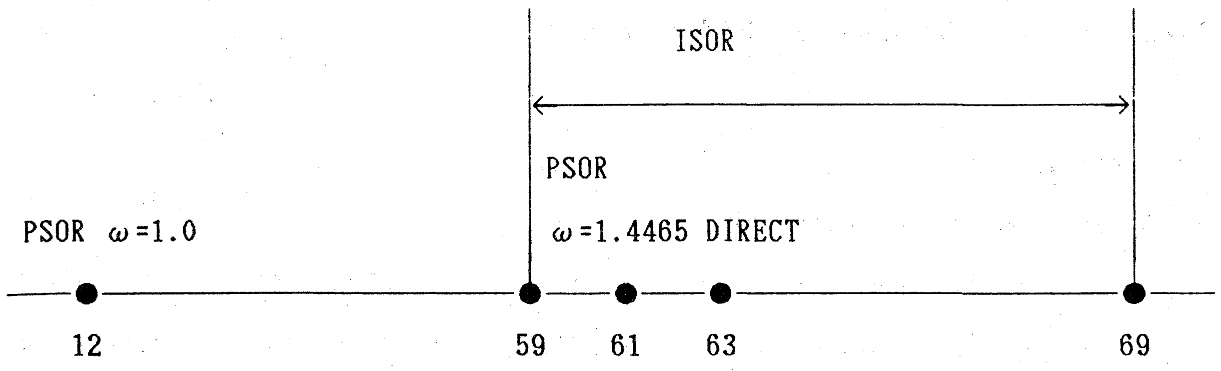
Figure 4. Results for ω=1.4465

The relative locations of derived values are depicted.

• (1,1) mesh point: 2 digits ** of 0.015159809 $\boxed{**}$ are shown under the line.



• (4,4) mesh point: 2 digits ** of 0.067952329 ** are shown under the line.



[Observations]

• In general, PSOR converged solutions with the same criterion vary according to the acceleration parameters. Better approximations are obtained by overrelaxation.

• The ISOR interval always includes the exact solution (the guaranteed interval by the direct method).

In Table 1, we summarize the numbers of iterations for PSOR and ISOR, and also PSOR converged values at the mesh point (1,1) with varying parameter ω.

Table 1. Numbers of iterations and (1,1) converged values

ω	PSOR	(1,1) converged value	ISOR
0.8	177	0.01515980956	47
0.9	145	59	〃
1.0	119	62	〃
1.2	78	68	〃
1.4	43	70	〃
1.4465	32	71	〃
1.5	33	74	〃

In this example problem, the numbers of ISOR iterations and convergent intervals are the same for different ω's.

4. Discussion

4.1 Overrelaxation and interval SOR

In our ISOR stage, the acceleration factor is always set at 1.0. Using the optimal acceleration factor ω of PSOR in the ISOR stage as well seems to give faster convergence.

The only modification necessary is to replace the statement(*) in Fig. 2 with the following two statements.

$$\boxed{\begin{aligned}&[z_i] \leftarrow \{b - \sum_{j>i} a_{ij} \times [x_j^{(k+1)}] - \sum_{j<i} a_{ij} \times [x_j^{(k)}]\} / a_{ij}\\ &[y_i] \leftarrow (1-\omega)[x_i^{(k)}] + \omega[z_i]\end{aligned}}$$

With this modification and taking ω=1.4, ISOR produces a diverging vector, as shown in Figure 5.

```
* RESULTS   :


  (I,J)                   LOWER                          UPPER

 (1) DIRECT

     (1,1)   (    0.151598097445841484D-01 ,     0.151598097445841494D-01)

     (4,4)   (      0.6795232963680387D-01 ,       0.6795232963680390D-01)


 (2) ISOR    :IT = 201 (ω= 1.4000)

     (1,1)   (                   -0.461D+68 ,                    0.461D+68)

     (4,4)   (                   -0.438D+70 ,                    0.438D+70)


 (3) PSOR    :IT = 43 (ω= 1.4000)

     (1,1)              0.151598096877621769D-01

     (4,4)              0.679523295482247908D-01
```

Figure 5. Results for ω=1.4, in PSOR and ISOR

On the other hand, the modified ISOR always converges for $0<\omega\leq 1$. The overrelaxation technique for accelerating convergence seems to work in the diverging direction when applied to interval computation. That is, the coefficients $1-\omega\,(<0)$ and $\omega\,(>1)$ in the second statement of the preceding modification, $(1-\omega)[x_i^{(k)}]+\omega[z_i]$, tend to contribute to the divergence.

4.2 Another method for interval SOR

The SOR method with the acceleration parameter ω can be considered as an iterative improvement process,

$$x^{(k+1)} \leftarrow x^{(k)} + R(b-Ax^{(k)}) = Rb + (I-RA)x^{(k)}$$

where $R=(1/\omega\ D+L)^{-1}$. Here, $A=D+L+U$ is the splitting of the coefficient matrix A to the matrices of the diagonal D, the lower triangular L, and the upper triangular U.

Direct method ACRITH subroutines such as DSSSB, for band matrices mentioned in section 2, and DLIN, for general matrixces, are also based on the above iterative improvement process, where a good approximation of A^{-1} is taken as R. In general,the convergency is essentially determined by the spectral radius $\rho(I-RA)$. The key to this process is to compute the right hand side with high-accuracy.

We have developed a program based on the above method. Its algorithm, which we call the matrix form interval SOR(MISOR), is as follows.

1) Initialization:

$$[R] \leftarrow (\frac{1}{\omega}D+L)^{-1}$$

$$[S] \leftarrow (I, -[R])\begin{pmatrix} I \\ A \end{pmatrix} \quad (+)$$

$$[P] \leftarrow ([R],\ [S])$$

$$[x^{(0)}] \leftarrow [0,\ 0]$$

$$k \leftarrow 0$$

2) Iterations:

$$[q] \leftarrow \begin{pmatrix} b \\ [x^{(k)}] \end{pmatrix}$$

$$[x^{(k+1)}] \leftarrow [P][q] \quad (+)$$

The test of inclusion, ε-inflation, and convergence test are the same as in ISOR.

For high-accuracy computation of the parts (+) above, we use ACRITH matrix multiplication subroutine DIMAM.

[Observations]

- In the case of $\omega=1.2$, MISOR does not diverge. However, the inclusion cirteria:

  $[x_i^{(k+1)}] \subseteq [x_i^{(k)}]$

  holds for only about a half of the components in every iteration cycle.

- In the case of $\omega=1.4$, MISOR eventually diverges.

Numerical examples show that MISOR is slightly better than ISOR in terms of non-divergence property, but cannot be a numerically stable method. In terms of computational work, MISOR has a much heavier load than ISOR.

S.M.Rump proposed the following iterative improvement method ([5]),which aims to get a guaranteed interval [d] of the error, the difference between t he exact solution and the approximation $\tilde{x}$.

$$[d] \leftarrow (\frac{1}{\omega}D+L)^{-1}\,\{b-A\tilde{x} - (U+D-\frac{1}{\omega}D)[d]\}$$

We did try it during this study, but this method is also likely to diverge.

4.3 Spectral radius of interval SOR iteration matrix

The iteration operator matrix of PSOR is given by (I-RA) where $R=(\frac{1}{\omega}D-L)^{-1}$. Those for interval ISOR and MISOR are considered as follows ([6]).

ISOR: $R^{*}(\frac{|1-\omega|}{\omega}D+|U|)$

MISOR: $|I-R^{*}A|$

Here, $R^{*}=(\frac{1}{\omega}D-|L|)^{-1}$.

In our example problem, R^{*} coincides with R. Table 2 shows the spectral radii of PSOR,ISOR and MISOR,with the parameter ω varying from 0.8 to 1.44. The values of the spectral radii correspond closely to the results of numerical experiments, except the case of ω=1.2 in MISOR.

Table 2. Spectral radius (for N=8)

ω	PSOR	ISOR	MISOR
0.8	0.902	0.902	0.902
0.9	0.880	0.880	0.880
1.0	0.854	0.854	0.854
1.2	0.778	1.604	0.850
1.4	0.611	2.407	1.236
1.44	0.511	2.575	1.340

5. Concluding remarks

We have developed a simple and practical self-validating SOR algorithm. In this paper, we gave details of our algorithm, programming considerations, and numerical examples.

In order to accelerate convergence in the interval SOR stage, overrelaxation iteration seems promising, as in the case of the point SOR stage. However, numerical experiments show a tendency to diverge in the ISOR stage. Further study, especially of this divergence problem, is needed in order to build up a truly practical self-validating SOR method.

[References]

[1] Ames,W.F.and Nicklas,R.C.,Accurate Elliptic Differential Equation Solver, in Miranker,W.L.and Toupin,R.A., edited by, Accurate Scientific Computations, Lecture Notes in Computer Science, Springer-Verlag, 1985.

[2] IBM High-Accuracy Arithmetic Subroutine Library Program Description and User's Guide, Third Edition, 1986. (Order number: SC33-6164-02)

[3] Okuda,A. and Tanamachi,Y., High-Accuracy Subroutine Library ACRITH, 'A New Direction in Scientific Computation Technologies' Workshop, at Ehime Univ., 1986.

[4] Varga, R.S., Matrix Iterative Analysis, Prentice Hall, 1962.

[5] Rump, S.M., Solving Algebraic Problems with High Accuracy, in Kulisch, U.W.and Miranker, W.L., edited by, A New Approach to Scientific Computation, Academic Press, 1983.

[6] Alefeld,G. and Herzberger,J., Introduction to Interval Computations, Academic Press, 1983.

[Appendix] Overview of ACRITH ([2],[3])

1. ACRITH, IBM High-Accuracy Arithmetic Subroutine Library, is designed for high accuracy computation with automatic verification,using interval arithmetic. Its aim is to provide quality assurance in numerical computation. The library is built up hierachically; it consists of full precision primitives, basic arithmetic routines, and problem solving routines.
2. ACRITH implements new methods such as the following.
   - inner product computation treated as basic arithmetic.
   - accumulator(s) for full precision computation.
   - four types of rounding

3. ACRITH consists of 3 hierachical levels.
   - Level 1: Problem Solving Routines
     - Polynomials
     - Dense systems of linear equations and matrix inversion
     - Sparse linear systems
     - Linear Programming
     - Eigenvalues and Eigenvectors
     - Nonlinear equations
     - Standard mathematical functions
   - Level 2: Basic Arithmetic Routines
     - Scalar operations
     - Vector operations
     - Scalar product
     - Matirix product

- Level 3: Full Precision Primitives
  - Accumulator preparation
  - Accumulator arithmetic operations
  - Store from accumulator

4. New functions in release 3
   - Extension to complex number
     - Accumulator arithmetic
     - Basic arithmetic
     - Polynomials
     - Dense systems of linear equations and matrix inversion
   - Interval accumulator (real and complex)
   - Sparse linear systems (real)
   - Nonlinear system of equations (real, long)