Non-Stationary Components Fixing Jacobi Iteration

JUN SRISUTAPAN
Department of Biostatistics, Mahidol University, Bangkok 10400. THAILAND

SIRIKUL BUNDITSAOVAPAK
Department of Mathematics and Computer Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520. THAILAND

Abstract: - This paper presents an effective extension to the classical Jacobi iteration. The non-stationary components fixing Jacobi iteration was designed to save the computational cost per iteration. This done by skip evaluating the solution components that probably need no computation in a certain iteration. The experiment using PDE data set confirm our method. Moreover, the quality of the obtained solutions are acceptable.

Key-Words: - Stationary iterative methods, Non-stationary iterative methods, Jacobi iteration, Gauss-Seidel, Components fixing, Convergence ratio, Binary convergence table, Approximate Jacobi iteration.

1 Introduction
We proposes an adaptive algorithm for the Jacobi iteration to solve a linear system arises from computational fluid dynamics. Section 2 gives literature reviews. Section 3 reviews Jacobi iteration in our notations. Section 4 describes the motivation of the problem in term of our notations together with introduces some terminology used in Section 5. Section 5 explains our method and Section 6 gives experimental results.

2 Literature Reviews
Solving a system of linear equations[1,2,3] is a fundamental work in scientific computing. This line of work can be categorised into direct and indirect (iterative) method.

The term iterative method refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system after each step. This paper will covers two types of iterative methods: 1) stationary methods, which are older, simpler, and easy to implements and 2) non-stationary methods, which are relatively new.

Below are short descriptions of each of the methods:

2.1 Stationary Methods
The Jacobi method solves for every variable locally with respect to the other variables. One iteration of the method corresponds to solving for every variable once.

The Gauss-Seidel method is similar to the Jacobi method except that it employs updated values as soon as they are available which make this method converges faster than the Jacobi.

Successive Over-Relaxation (SOR) can be derived from the Gauss-Seidel method by introducing an extrapolation parameter and may converge faster than Gauss-Seidel.

2.2 Non-stationary Methods
The Conjugate Gradient method (CG) generates a sequence of orthogonal vectors. These vectors are the residuals of the iterates. They are also the gradients of a quadratic function, the minimisation of which is equivalent to solving the linear system. The CG method has been adapt for various conditions of various types of problem.

Chebyshev Iteration recursively determines polynomials with coefficients chosen to minimise the norm of the residual in a min-max sense.

This paper focuses on Jacobi and Gauss-Seidel iteration methods because these iterative methods are known as the fundamental theoretical and application framework for all iterative methods. Furthermore, these methods are still widely used in many classes of applications, making their improvement greatly appreciable.

3 Classical stationary Jacobi iteration
An iterative method to solve a linear system $Ax = b$, where $A = \{a_{ij}\} \in \Re^{n \times n}$ and $b = \{b_i\} \in \Re^n$,

starts with an initial approximation $x^{(0)} \in \Re^n$ to approach the solution $x = \{x_i\} \in \Re^n$ by produces a sequence of vectors $\left\{x^{(k)}\right\}_{k=0}^{\infty} \in \Re^n$ that would converge to the solution x. The method can be expressed in the form $x = Tx + c$ where $T = \{t_{ij}\} \in \Re^{n \times n}$ and $c = \{c_i\} \in \Re^n$. In stationary methods, neither T nor c depends upon the iteration count k. In contrast, for non-stationary iterative methods, T and/or c can depends upon the iteration count k.

For each iteration k and components i, Jacobi iteration generates a sequence $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ by using the generating function $\Gamma_{Jacobi}(k,i)$ with the assignment $x_i^{(k)} = \Gamma_{Jacobi}(k,i)$. Γ_{Jacobi} is defined by (4) whereas Gauss-Seidel employs $\Gamma_{Gauss-Seideli}(k,i)$ in (5).

$$\Gamma_{Jacobi}(k,i) = \frac{-\sum_{j=1, j \neq i}^{N} a_{ii} x_j^{(k-1)} + b_i}{a_{ii}} \tag{1}$$

$$\Gamma_{Gauss-Seidel}(k,i) = \frac{-\sum_{j=1}^{i-1} a_{ii} x_j^{(k)} - \sum_{j=i+1}^{N} a_{ii} x_j^{(k-1)} + b_i}{a_{ii}} \tag{2}$$

Definition For each iteration k and component i we define $G^{(k)} := \left\{i = 1 \ldots N : \left\| x_i^{(k)} - x_i^{(k-1)} \right\|_{\infty} < \varepsilon \right\}$

$G^{(k)}$ can be used to check whether the linear convergence is converge at iteration k, e.g., iff $\left|G^{(k)}\right| = N$. The infinity norm in $G^{(k)}$ can be replaced with any proper norm. The algorithmic of the classical stationary Jacobi then can be written as $\Gamma_{Jacobi}(k,i)$ and $G^{(k)}$, which can be replaced with Gauss-Seidel or different norm(*).

```
Algorithm Jacobi Iteration. Solving $Ax = b$
order $N$ with initial solution $x^{(0)}$ and
infinity-norm and TOLERANCE $\varepsilon$
begin
• converge := false;
• $k := 0$;
  while (not(converge)) do
  {
     $k := k + 1$;
     for $i := 1 \ldots N$ do
        $x_i^{(k)} := \Gamma_{Jacobi}(k,i)$; (*)
     $G^{(k)} := \left\{i = 1 \ldots N : \left\| x_i^{(k)} - x_i^{(k-1)} \right\|_{\infty} < \varepsilon \right\}$; (*)
     if $\left|G^{(k)}\right| = N$ then
        converge := true;
  }
end;
```

Fig.1 Formalism of classical Jacobi iteration

4 Motivation

This section describes the root of the problem. For example, let we solve the linear system (3) of order $N = 8$ using Jacobi iteration with infinity norm where tolerance ε =0.05.

$$\begin{bmatrix} 9 & 0 & 1 & 2 & 0 & 0 & 2 & 1 \\ 0 & 9 & 1 & 1 & 0 & 0 & 3 & 0 \\ 1 & 1 & 9 & 1 & 2 & 0 & 1 & 2 \\ 2 & 1 & 1 & 9 & 1 & 2 & 1 & 1 \\ 0 & 0 & 2 & 1 & 9 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 & 0 & 9 & 1 & 0 \\ 2 & 3 & 1 & 1 & 2 & 1 & 9 & 1 \\ 1 & 0 & 2 & 1 & 1 & 0 & 1 & 9 \end{bmatrix} x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{3}$$

We track how the linear system reaches the convergence point.

4.1 Error

In Fig.2 seven iterations are need to make (3) converges. The Root Mean Square Error (RMSE), which is a quality of the solution, is 0.223.

4.2 Computational Cost

Jacobi iteration pays static cost for using generating function at cost N per iteration. It employs the component-updating equation (1) for 56=8×7 times shown within Fig.2.

4.3 Binary Convergence Table

Definition *Binary convergence table* is a table displaying how each components reach the convergence. The rows represent iteration count and columns represent component.

$k \backslash i$	1	2	3	4	5	6	7	8
1	×	×	×	×	×	×	×	×
2	×	×	×	×	×	•	×	×
3	×	×	×	×	×	•	×	×
4	×	×	×	×	×	•	×	×
5	•	•	×	×	•	•	×	•
6	•	•	•	•	•	•	×	•
7	•	•	•	•	•	•	•	•

Fig.2 Binary convergence table of solving (3) using classical Jacobi Iteration

For $k = 2$ and $i = 6$ we have $x_6^{(2)} \in G^{(2)}$, i.e., $\left\| x_6^{(2)} - x_6^{(2-1)} \right\|_{\infty} < \varepsilon$ so we mark • at element (2, 6) but for $k = 2$ and $i = 8$ we have $\left\| x_8^{(2)} - x_8^{(2-1)} \right\|_{\infty} \geq \varepsilon$ or $x_8^{(2)} \notin G^{(2)}$ so we mark × at element (2, 8).

Moreover for $k \geq 5$ we also have $x_1^k, x_2^k, x_5^k, x_6^k, x_8^k \in G^{(k)}$. However it is not always that $x_m^{(k)} \in G^{(k)}$ imply $x_m^{(k+1)} \in G^{(k+1)}$ but by observation this is probably occur. The binary convergence table gives a lot of useful information.

4.4 Convergence Ratio

From binary convergence table it is natural to define some quantity.

Definition *convergence ratio*, π_k at iteration k is defined as $\pi_k = \left|G^{(k)}\right| / N$.

5 The Method

Our method was inspired by the binary convergence table. The method consists of *fixing*, *flushing*, and *convergence determining*.

5.1 Fixing

One simple way to reduce the computational cost is to simply *fix* the component $x_i^{(k)}$. Unconditionally fixing or modifying a component can be viewed as giving an unusual noise to the system. Fixing is a very low-cost operation. It is definitely effective if it introduces a little noise to the system but yield an acceptable result.

In case of N=3, standard Jacobi searches the solution within R^3. Fixing one component reduce the search space to R^2. Fixing one more component would reduce it further to R^1. Therefore if we know in prior that the noise added is low enough, we can reduce our search space one dimension at a time. In fact, binary convergence table provides us some useful heuristic. Here we do fix $x_i^{(k)}$ only if when $x_i^{(k)} \in G^{(k)}$. This can be done via defining a new generating function, $\Gamma_{\Phi-Jacobi}(k,i)$ for Jacobi extension and $\Gamma_{\Phi-Gauss-Seidel}(k,i)$ for Gauss-Seidel.

$$\Gamma_{\Phi-Jacobi}(k,i) = \begin{cases} \Gamma_{Jacobi}(k,i) & ;\left(x_i^{(k)} \notin G^{(k)}\right) \\ x_i^{(k-1)} & ;otherwise \end{cases} \tag{4}$$

$$\Gamma_{\Phi-Gauss-Seidel}(k,i) = \begin{cases} \Gamma_{Gauss-Seidel}(k,i) & ;\left(x_i^{(k)} \notin G^{(k)}\right) \\ x_i^{(k-1)} & ;otherwise \end{cases} \tag{5}$$

$k \backslash i$	1	2	3	4	5	6	7	8
1								
2						•		
3						▼		
4								
5	•	•			•			•
6	▼	▼	•	•	▼			▼
7			▼	▼			•	

Fig.3 Example of component fixing from iteration 2→3, 5→6, 6→7

In Fig.3 at iteration $k = 3$, there use only 8-1 calls of Γ_{Jacobi} At iteration $k = 6$ for 8-4=4 times of calling Γ_{Jacobi} are required. Let us define some term related to the fixing.

Definition For iteration k, $F^{(k)}$ is defined to be the total number of components that have been fixing from iteration $k-1$ to k.

Therefore, in Fig.3, $F^{(1)}$ = 0, $F^{(2)}$=0, $F^{(3)}$=1, $F^{(4)}$=0, $F^{(5)}$=0, $F^{(6)}$=4, $F^{()}$=2.

5.2 Convergence

Test for convergence of classical version uses $\left|G^{(k)}\right| = N$. If we still use this in the component fixing (i.e. fix some dimension of the solution), we cannot be sure whether what we fixed is really the correct dimension. One way that would ensure us is to consider fixing no component before making a decision for convergence.

Hence, the system is said to be converge if and only if $\left|G^{(k)}\right| = N$ and $F^{(k)} = 0$.

5.3 Flushing

Flushing lets the system search for the solution in N-dimensional space. Two flushing strategies designed for this work are i*mmediately-flushing* and *proportional-flushing.*

5.3.1 Immediately-Flushing (Jacobi-I)

At iteration k, do flush when $F^{(k-1)}$ > 0. In Fig.4, every components from iteration 4 must be computed into iteration 5.

k/i	1	2	3	4	5	6	7	8	$F(k$-1)
1									
2						•			
3						▼			1 > 0
*4**									

Fig.4 Immediately-Flushing at iteration 4

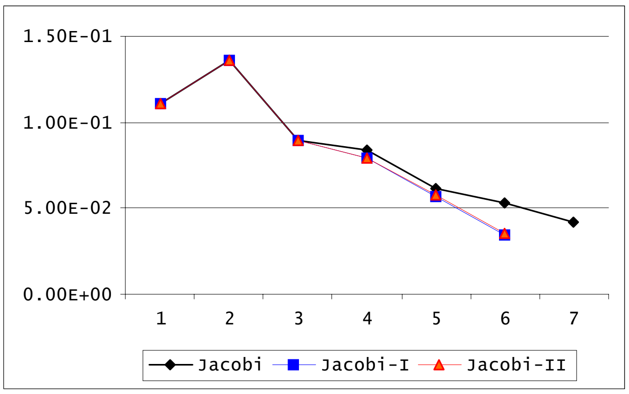

Fig.5 Error norm v.s. iteration of solving (3) using classical Jacobi, Jacobi-I(5.3.1), and Jacobi-II(5.3.2)

5.3.2 Flushing Proportional to $\frac{F^{(k-1)}}{N}$ (Jacobi-II)

Do flushing only if the number of fixed element, $F^{(k-1)}$ is large enough. As we try to make our model simple, we try to avoid defining what is large. Instead, we assign the probability to flush dues to the ratio $F^{(k-1)}/N$. The larger $F^{(k-1)}$, the greater probability to flush the system.

In Fig.6, at iteration 6, the flushing will occurred since a uniform random number from $\chi^2_{[0,1]}$ gives a value greater than $F^{(5)}/8 = 0.5$.

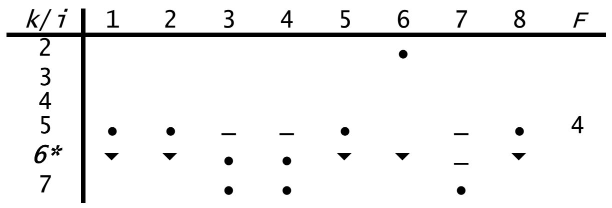

Fig.6 Flushing Proportionally at iteration 6

5.4 Computational Cost

In all Jacobi's, the cost per iteration k is actually $N - F^{(k)}$. In classical Jacobi $N - F^{(k)} = N - 0 = N$. Fig.7 shows the computational cost of all Jacobi iterations.

5.5 Error

The graph of error within the process of all Jacobi iterations are shown in Fig.5. They are not much different for (3). However, in some equation, our method also decrease the error dramatically.

5.6 The quality of the solution

After linear systems are converged, RMSE is used to show the quality. The RMSE of the three methods comparing to the exact solution are given in Fig.8

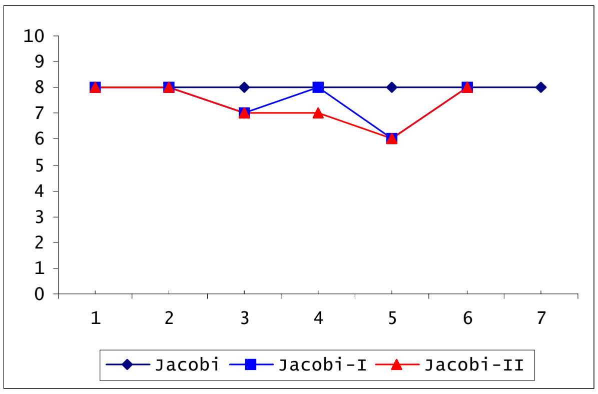

Fig.7 Computational cost v.s. iteration of classical Jacobi, Jacobi-I, and Jacobi-II .

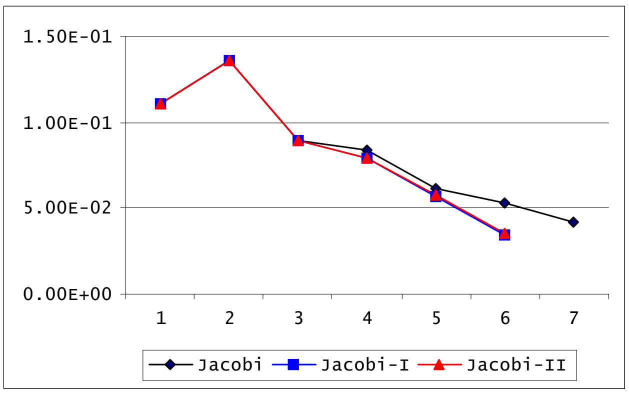

Fig.8 RMSE v.s. iteration of Jacobi Family

Firstly we expected the RMSE of our fixed method are slighter higher than classical one, but the experiment show even the RMSE is reduced.

5.7 Algorithm

```
Algorithm Components-Fixing Jacobi Iteration. Solving $Ax = b$ order $N$ with
initial solution $x^{(0)}$ with infinity-norm and TOLERANCE
begin
• converge := false;
• $k := 0$;
• $G^{(k)} := \{\ \}$;
• $F^{(k)}$ := 0;
  while (not(converge)) do
  {
     $k := k + 1$;
     $F^{(k)}$ := 0;
     if ( $F^{(k-1)}$ > 0) then  (**)
       for $i = 1 \ldots N$ do
         $x_i^{(k)} := \Gamma_{Jacobi}(k,i)$ ;  (**)
     else
     {
       for $i = 1 \ldots N$ do
         if $i \in G^{(k-1)}$ then
           $F^{(k)} := F^{(k)} + 1$;
         else
           $x_i^{(k)} := \Gamma_{Jacobi}(k,i)$ ; (**)
     }
     $G^{(k)} := \{ i = 1 \ldots N : \| x_i^{(k)} - x_i^{(k-1)} \|_\infty < \varepsilon \}$;
     if ( $F^{(k)}$ = 0) and $|G^{(k)}| = N$ then
       converge := true;
  } // while
end;
```

Fig.9 Algorithm for Jacobi Family

(**) The term $x_i^{(k)} := \Gamma_{Jacobi1}(k,i)$ can be replaced by Gauss-Seidel and $F^{(k-1)} > 0$ can be substituted by another flushing condition.

6 Experimental Results

6.1 Data sets

Three computational fluid dynamics data sets from `http://www.math.nist.org` are used to test our method.

6.2 Parameters and results

We solve $Ax = b$, using infinity-norm with different tolerance ε and a couple of different b. The random number seed is fixed as 1 and initial guess $x^{(0)} = (0 \ldots 0)^T$. Number computed is the total number of calling (1) or (2). All results are show in Fig.10-Fig.15 (method 1 denotes Jacobi, 2 for Jacobi-I, 3 for Jacobi-II ,4 for Seidel, 5 for Seidel-I, 6 for Seidel-II). Fig.16 plots norm v.s. iteration of solving PDE225 with Jacobi's $\varepsilon = 0.05$ shown in Fig.13. Fig.17 plots norm v.s. iteration of solving PDE225 with Gauss-Seidel's $\varepsilon = 0.05$ shown in Fig.13.

7 Conclusion

The component fixing-flushing strategy appeared in this paper is determined by $G^{(k-1)}$ and $F^{(k-1)}$. The strategy is very simple yet provides a noticeable improvement. Determining more than one iteration can give better result.

In general, there might be fixing-flushing strategies that can reduce as much computational effort for updating solution component while retains the same or better solution. Although we don't aim our method to beat SOR or conjugated-gradient, we wish our method would be an additional improvement on many data set.

References:

[1] Burden, R.L. and Faires, J.D. Numerical Analysis, Brooks/Cole, London, 6th ed. ,1997.

[2] Fröberg, C-E. Numerical Mathematics, Benjamin/Cummings Publishing Company, Sydney, 1985.

[3] Endre Süli and David F.Mayers. An Introduction to Numerical Analysis, Oxford University Press, 2003.

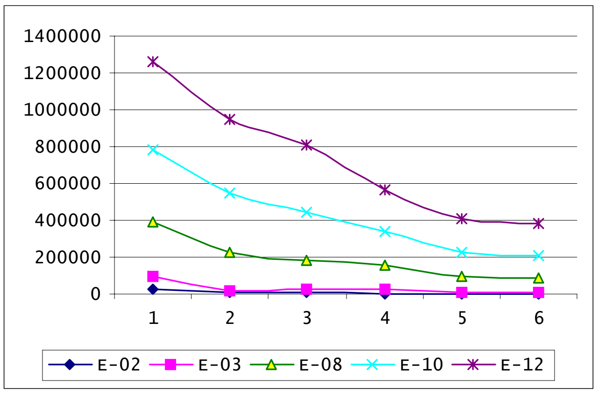

Fig.10 Number computed for PDE225: $b = (+1\ -1\ +1\ \ldots)^T$

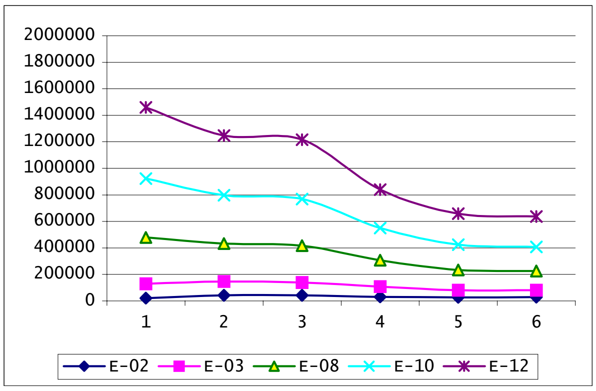

Fig.11 Number computed for PDE900: $b = (+1\ -1\ +1\ \ldots)^T$

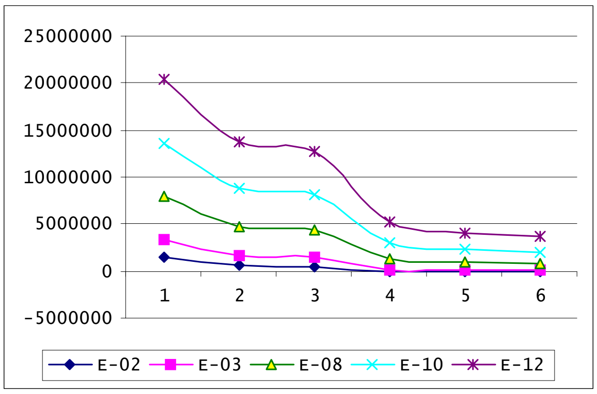

Fig.12 Number computed for PDE2961: $b = (+1\ -1\ +1\ \ldots)^T$

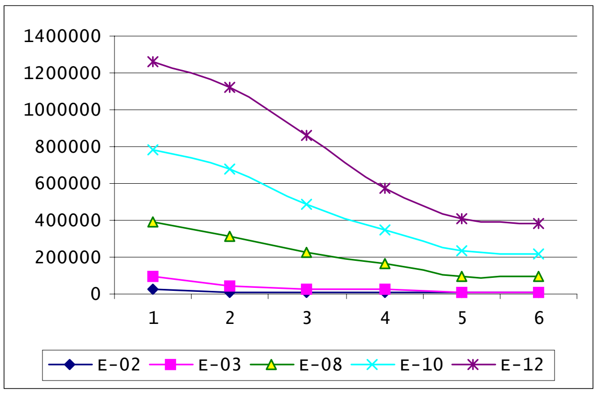

Fig.13 Number computed / PDE225: $b = (1\ \ 1\ \ 1\ \ ...)^T$

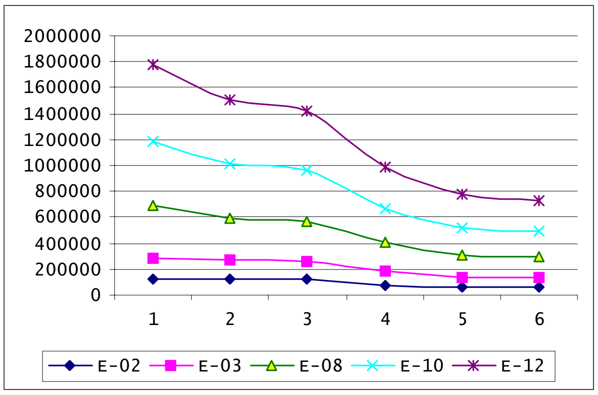

Fig.14 Number computed / PDE900: $b = (1\ \ 1\ \ 1\ \ ...)^T$

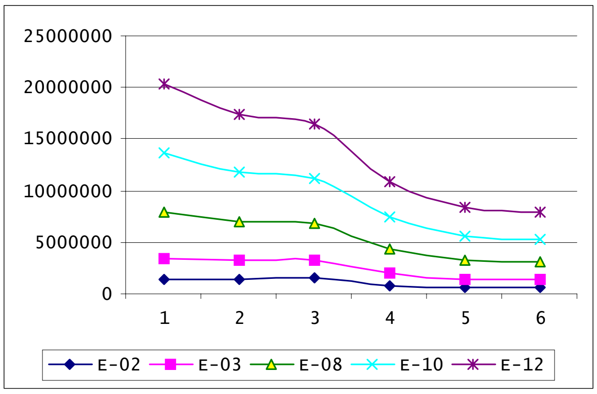

Fig.15 Number computed/PDE2961: $b = (1\ \ 1\ \ 1\ \ ...)^T$

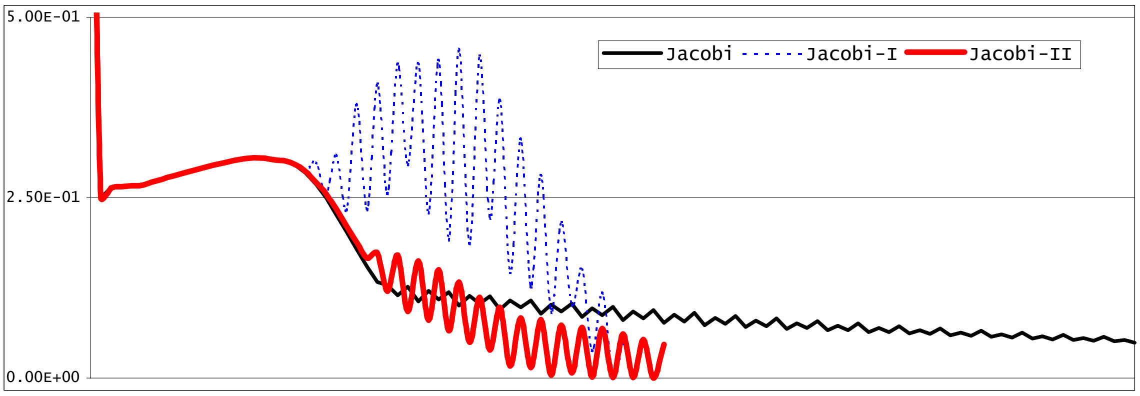

Fig.16 Norm v.s. iteration of solving PDE225 with Jacobi's $\varepsilon = 0.05$ shown in Fig.13.

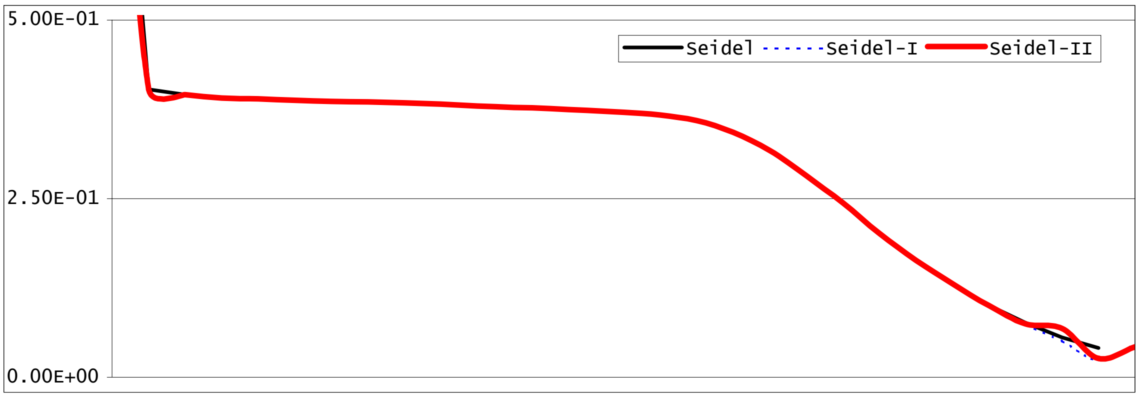

Fig.17 Norm v.s. iteration of solving PDE225 with Gauss-Seidel's $\varepsilon = 0.05$ shown in Fig.13.