Computing, Suppl. 2, 157 – 164 (1980)

Small Bounds for the Solution of Systems of Linear Equations

S. M. Rump and E. Kaucher, Karlsruhe

Abstract

An algorithm is presented to solve a system of linear equations $Ax = b$ of high order. There are no restrictions for A; A may be a floating-point or interval matrix. The algorithm leads to small, guaranteed bounds for the solution even for ill-conditioned matrices. It takes about six times the computing time needs for the usual floating-point Gaussian algorithm with comparable accuracy.

0. Introduction

Here as throughout the paper "solving" always means giving guaranteed, provable bounds for the exact solution. The approximation of the exact solution may have a big relative error in case that the condition number is large (which in fact is not known in general). If there is no (provable) error estimation one is not able to decide whether an approximation is good or bad. Therefore an algorithm not yielding guaranteed bounds for the exact solution may cause a lot of damage (e.g. repetition of expensive experiments etc.) when a poor approximation is regarded as a good one. With respect to this and other reasons we set a high value on provable bounds to be computed by the algorithm.

Up to now most of those algorithms are either calculating bounds for the solution in a brute-force way (naive interval arithmetic) or depending on a first inclusion of the solution or even for the inverse of the matrix. We are looking for general algorithms working with single precision but giving a solution of high accuracy and moreover we wish to avoid the following lacks as well as possible:

- computing with only single precision or without computing residual corrections produces intervals of a large width
- to start an inclusion of the inverse or of the solution is needed
- the algorithm converges slowly or cannot even start for more bad conditioned problems
- there are certain restrictions for the matrix to be satisfied
- the algorithm needs a lot of time and (or) space
- the algorithm is suitable working only for lower degrees, at most for degrees less than 50
- in general the algorithm cannot use the accuracy of classical methods
- the iteration functions have to be inclusion isotone.

In the following an algorithm is presented satisfying the properties mentioned above. Furthermore several improvements have been introduced, especially

computing bounds for a "residual equation",

going in contrast "from the inner to the outer", that means starting with a certain interval (not necessarily including the solution) and by blowing it up receiving an inclusion of the solution

using good floating-point approximations as well as possible.

The algorithm is working for floating-point as well as for interval systems and has been implemented on the UNIVAC 1108 of the University of Karlsruhe. The algorithm is written in FORTRAN and therefore portable.

Some computational results are given at the end.

1. Theoretical Background

In the paper [4] we presented very general and widely usable theorems concerning Schauder's Fixpoint Theorem. We now present a special form of the cited theorem.

Theorem 1. *Let f: $\mathbb{R}^n \to \mathbb{R}^n$ be a continuous mapping. Let further F: $\mathbb{IR}^n \to \mathbb{IR}^n$ be a given arbitrary function with*

$$\bigwedge_{I \in \mathbb{IR}^n} x \in I \Rightarrow f(x) \in F(I), \tag{1}$$

where $\mathbb{IR}^n$ denotes the set of n-dimensional interval-vectors over $\mathbb{R}$. If for an $\Omega \in \mathbb{IR}^n$ holds

$$F(\Omega) \subseteq \Omega \tag{2}$$

then there exists a fixpoint $\hat{x}$ of f with

$$\hat{x} \in F(\Omega) \tag{3}$$

and furthermore

$$\hat{x} \in \bigcap_{i=0}^{\infty} F^i(\Omega). \tag{4}$$

Note the very weak assumptions and the strong results of the theorem, especially that the interval function F is not assumed to be inclusion monotone or convergent in any sense.

Now let a system of linear equations $Ax = b$ be given with an $n \times n$-matrix A and an n-dimensional vector b. We first assume A and b to be real, i.e. $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$. To apply the theorem above we first have to look for a real function f and then for an interval function F satisfying (1). The first idea for f may be the well-known residual iteration (for invertible A)

$$f^*(y) = y + A^{-1}(b - Ay).$$

Of course the inverse matrix A^{-1} is in general not known; so one tries to replace it by a floating point approximation $R \approx A^{-1}$:

$$\bar{f}(y) = y + R(b - Ay). \tag{5}$$

For a fixpoint $\hat{y}$ of $\bar{f}$ we have

$$\hat{y} = \bar{f}(\hat{y}) = \hat{y} + R(b - A\hat{y}) \equiv R(b - A\hat{y}) = 0.$$

If R is not singular, then $\hat{y}$ is a solution of $Ax = b$. To find an interval function F we write (5) in the following way

$$\begin{aligned} \bar{f}(y) &= x + Rb - RAx + y - x - RAy + RAx \\ &= x + R(b - Ax) + (E \dot{-} RA)(y - x) \end{aligned}$$

where E denotes the $n \times n$ unit matrix. Now let F be defined as

$$F(Y) = x + R(b - Ax) + (E - RA) * (Y - x), \tag{6}$$

then (1) follows immediately. The function (6) occurs for instance in [5]. To get bounds for the solution of $Ax = b$ we need an interval-vector Ω with $F(\Omega) \subseteq \Omega$. To find it we start with the interval Y^0 consisting only of the point $x + R(b - Ax)$ for a certain x. Then we iterate

$$Y^{k+1} := F(Y^k) \quad \text{until} \quad Y^{k+1} \subseteq Y^k. \tag{7}$$

x may be any vector; however, we prefer a floating-point approximation $\tilde{x}$ for the solution $\hat{x}$ to get sharp bounds.

If the spectral radius of $E - RA$ is smaller than one then surely A and R are non-singular. So in this case the last Y^{k+1} of (7) contains the unique solution $\hat{x}$ of $Ax = b$.

2. Improvements

First we give a sample for our algorithm; an improved version will follow later.

1) Compute a floating-point approximation R of A^{-1}
2) Compute $B := E \ominus R \odot A$ by interval arithmetic; **if** $\|B\| \geqslant 1$ **then** stop
3) Let $\tilde{x} := R \cdot b$ be a floating-point approximation of $\hat{x}$
4) $Y^0 := Z := \tilde{x} \oplus R \odot (b \ominus A \odot \tilde{x})$ by interval arithmetic
5) **repeat** $Y^{k+1} := Z \oplus B \odot (Y^k \ominus \tilde{x})$ **until** $Y^{k+1} \subseteq Y^k$.

The operations in a circle always denote an interval operation.

One fundamental improvement is to construct an inclusion for the solution of the residual equation

$$Ay = b - A\tilde{x} \tag{*}$$

instead of the original equation $Ax = b$, where $\tilde{x}$ is a floating-point approximation due to step 3). If y is the solution of the residual equation (*) then $\tilde{x} + y$ is equal to $\hat{x}$, and if an interval Y contains y then $\hat{x}$ is contained in $\tilde{x} \oplus Y$. If furthermore $\tilde{y}$ is a floating-point approximation to y and z the solution of $Az = b - A\tilde{x} - A\tilde{y}$, then $\hat{x} = \tilde{x} + \tilde{y} + z$. If Z is an interval containing z, then $\tilde{x} \oplus \tilde{y} \oplus Z$ contains $\hat{x}$.

This method can be continued in an obvious way. However, these computations make only sense, if the residuals $b - A\tilde{x} - A\tilde{y} - \cdots$ are computed with higher accuracy.

For this purpose for instance the algorithm of Bohlender [2] can be used or a long accumulator as described in [3].

In step 3) of the sketched algorithm it is superior not to take $R \cdot b$ as an approximation of $\hat{x}$ but to iterate with the residual iteration

$$x^0 = R \cdot b;\ x^{k+1} = x^k + R(b - Ax^k). \tag{8}$$

In the case of a floating-point computation the problem arises when to stop the iteration (8). In our algorithm there is no essential dependence on this question. Here a simple stopping criterion is used which, roughly spoken, guarantees a "win of at least one decimal digit" in two iteration steps. So we proceed as follows: First (8) is executed, where the last iterative is noted as $\tilde{x}$. Then

$$y^0 := R(b - A\tilde{x}); \qquad y^{k+1} := y^k + R(b - A\tilde{x} - Ay^k) \tag{9}$$

is executed, where $\tilde{y}$ denotes the last iterative.

With

$$Y^0 := \tilde{y} \oplus R \odot (b \ominus A \odot \tilde{x} \ominus A \odot \tilde{y}) \tag{10}$$

we proceed in step (5) yielding an interval vector Y, thus $\hat{x} \in \tilde{x} \oplus Y$. The residual $b \ominus A \odot \tilde{x} \ominus A \odot \tilde{y}$ in (10) has to be computed in double precision interval arithmetic.

When proceeding in step 5) it turns out that it is better to use a "Einzelschrittverfahren", that means to calculate componentwise and using the computed components at once.

Furthermore an ε-expansion will be introduced. We define

$$I \circ \varepsilon := I + d(I) \cdot \varepsilon * [-1, 1] \tag{11}$$

It follows

$$d(I \circ \varepsilon) = d(I) + 2\varepsilon \cdot d(I) = (1 + 2\varepsilon) \cdot d(I),$$

so the width of the interval I is relatively enlarged by 2ε and can be interpreted as an "artificial rounding". We write step 5) now as follows

$$\begin{aligned} &Y^0 := Z; \qquad k := -1; \\ &\textbf{repeat } k := k + 1; \qquad Y^{k+1} := Y^k := Y^k \circ \varepsilon; \\ &\qquad \textbf{for } i := 1 \textbf{ to } n \textbf{ do } Y_i^{k+1} := Z_i \oplus B_i \odot (Y^k \ominus \tilde{y}) \\ &\textbf{until } \ Y^{k+1} \subseteq Y^k; \end{aligned} \tag{12}$$

Here Y_i and Z_i are the ith component of Y and Z and B_i the ith row of B, resp. In practice it turns out that $\varepsilon = 0.1$ is a good value. It should be noted explicitely that with the ε-expansion we achieve an improvement of the speed of convergence. Regarding (3) one might proceed with

$$Y^{k+1} := \{Z \oplus B \odot (Y^k \ominus \tilde{y})\} \cap Y^k \tag{13}$$

until $Y^{k+1} = Y^k$. However, even computing with double precision interval arithmetic in (13) gives no significant improvement of the bounds.

Obviously, if the matrix A is satisfying special conditions then special improvements can be introduced. If, for instance, the matrix A is strong diagonal dominant, then we can use D^{-1} instead of R as an approximation of A^{-1}. The inverse D^{-1} of the diagonal of A is very easy to compute.

We should mention that if a bound σ of the spectral radius of $E - RA$ is known and $\sigma \ll 1$ holds, one might use the formula

$$\|x^0 - \hat{x}\| \leqslant \frac{1}{1-\sigma} \cdot \|x^0 - x^1\| \tag{14}$$

to give a bound for the solution $\hat{x}$ as is proposed in [1], [6]. But from (14) we get an inclusion "sphere" for $\hat{x}$ independent of the location of its components. Up to now there is not general equilibration method known. Therefore all components of relatively small modulus are overestimated proportional to the maximum difference of the exponents of $\hat{x}$ and so (14) yields in general poor bounds. We avoid this disadvantage for it is more convenient to estimate bounds for $\hat{x}$ using all information (e.g. the whole matrix A) instead of using the only number σ. Furthermore the execution of (12) is of very low cost compared with whole computing time and the gained quality of the bounds.

Moreover for σ nearly 1 or $\sigma \geqslant 1$ or unknown σ the method (14) does not work. Remember that $\sigma < 1$ in our algorithm is only needed to prove R to be non-singular. This can be done e.g. externally or by proving $R \cdot A$ to be of property M or strong diagonal dominant or by some other methods. For the reasons mentioned it is not worthwhile to include (14) in the algorithm, especially not to complicate the program unnecessarily.

3. The Algorithm

With the improvements introduced above we can write down the final version of the algorithm:

---

1) Compute $R \approx A^{-1}$ by floating-point arithmetic.
2) Compute $C = A \odot R$ and $B = I \ominus C$ by interval arithmetic. **If** C is not a matrix of property M (or strong diagonal dominant) and $\|B\| \geqslant 1$ **then** stop.
3) Set $x^0 = R \cdot b$; $k := -1$;
   repeat $k := k + 1$; $x^{k+1} = x^k + R \cdot (b - Ax^k)$
   until $|x^{k+1} - x^k|/|x^k| \geqslant 10^{-k/2}$ or
   $|x^{k+1} - x^k|/|x^k| < 10^{1-t}$;
   the final x^{k+1} is named $\tilde{x}$ and $r := k + 1$.
   Set $y^0 = R(b - A\tilde{x})$; $k := -1$,
   repeat $k := k + 1$; $y^{k+1} = y^k + R(b - A\tilde{x} - Ay^k)$
   until $|y^{k+1} - y^k|/|y^k| \geqslant 10^{-k/2}$ or
   $|y^{k+1} - y^k|/|y^k| < 10^{2-t}$;
   the final y^{k+1} is named $\tilde{y}$ and $r := r + k + 1$.
4) Compute $Y^0 := Z := \tilde{y} \oplus R \odot (b \ominus A \odot \tilde{x} \ominus A \odot \tilde{y})$; $k := -1$
   by interval arithmetic.
5) **repeat** $k := k + 1$; $Y^{k+1} := Y^k := Y^k \circ \varepsilon$,
   for $i := 1$ **to** n **do** $Y_i^{k+1} := Z_i \oplus B_i \odot (Y^k \ominus \tilde{y})$
   until $Y^{k+1} \subseteq Y^k$ or $\{k > 2.25 \cdot r - 1\} =: \text{bool}$;
   if bool **then** stop.
6) The final Y^{k+1} is named Y and $\hat{x} \in \tilde{x} \oplus Y$.

Remarks. In step 1) the Gauss-Jordan algorithm with pivoting is used. It turned out to be better than the Gaussian elimination method. In step 3) and 4) only the residuals are to be computed in double precision. t denotes the number of digits of the mantissa of the computer. The exit stop means that the algorithm fails to compute bounds and may be repeated with higher accuracy.

It is very easy to extend the algorithm to interval equations. Just replace every interval matrix A or interval vector b by the midpoint $m(A)$ or $m(b)$ where A or b is occurring in a floating-point computation, resp.

4. Complexity

Let α_1 be the cost to compute a floating-point approximation for $\hat{x}$ and α_2 be the cost to compute bounds for $\hat{x}$ with the presented algorithm, both with comparable accuracy. As a measure for the additional costs to compute bounds we take the ratio

$$\rho = \frac{\alpha_2}{\alpha_1}.$$

W.l.o.g. we use the Gaussian elimination method with r residual iterations (step 3)) to compute a floating-point approximation. Then (neglecting linear terms in n) we get $\alpha_1 \geqslant n^3/3 + 2rn^2$.

Adding the additional computing times for the steps 1) to 5), where step 5) was executed s times, yields (neglecting linear terms)

$$\begin{aligned}\alpha_2 &\leqslant 2n^3 + n^2(3r + 4s + 4)\\ &\leqslant 6 \cdot \alpha_1 + n^2(-9r + 4s + 4).\end{aligned}$$

So

$$\rho \leqslant 6 + \frac{-9r + 4s + 4}{n/3 + 2r}$$

can be estimated as follows: If $s \leqslant (9r - 4)/4$ then $\rho \leqslant 6$, otherwise $\rho \leqslant 6 \cdot \{1 + (2s - 3.5r)/n\}$. In fact we have $\rho = 6 + 0(1/n)$; in the algorithm $\rho \leqslant 6$ is satisfied. If the algorithm fails (via exit: stop) and is repeated with doubled accuracy etc., the finally ρ is in the most pessimistic case estimated by $\rho \leqslant 6 \cdot \sum_{i=0}^{\infty} 4^{-i} = 6 \cdot \frac{4}{3} = 8$. It is remarkable that ρ is independent of n and the condition number of A.

5. Empirical Results

The empirical results given were computed on the UNIVAC 1108 of the University of Karlsruhe with $t = 8\frac{1}{2}$ decimal digit accuracy for single precision. All computing were done in single precision arithmetic except the residuals, which were computed in double precision but stored in single precision. First we give an example to show what happens when only few iterations are executed in step 3). We take the Hilbert 7×7-matrix, where the coefficients are rational integers and $b = (1, \ldots, 1)$. The condition number is $4.8 \cdot 10^8$ and σ less than 0.55. Due to 3) we have $r = 6$ iterations and we get for the first component

of x^k: 11.658, 5.336, 7.594, 6.788 and

of y^k: 0.2884, 0.1853, 0.2221, 0.209.

With the final values for $\tilde{x}$ and $\tilde{y}$ we get

$$\tilde{x} + \tilde{y} = 6.997,$$

far away from the exact value 7.0. With $Y^0 = [0.2136253^{11}_{00}]$ we get after 9 iterations of step 5)

$$\hat{x} \in [6.999999471, 7.000000294]$$

with an accuracy of 7 correct decimal positions. By the way, using the estimation (14) we would get the bad inclusion:

$$\hat{x} \in [-21.96, 35.95].$$

If we would have 18 iterations in step 3) the relative error bound of the final inclusion would have been bounded by $5 \cdot 10^{-11}$. We see that in special cases it might be better to iterate a bit longer in step 3); however, it is not necessary to achieve good results.

Finally, we give a sample of some bad-conditioned matrices of higher degree.

degree(A) $n =$	cond(A) $\lesssim$ $\|A\| \cdot \|R\| =$	at least guaranteed decimal positions in each component [log($d(Y)/\|Y\|$)]	r	s	ρ
25	$3.1 \cdot 10^7$	12	9	2	3.4
50	$9 \cdot 10^7$	11	10	3	4.0
100	$2.6 \cdot 10^8$	12	16	4	4.1
150	$8 \cdot 10^7$	13	8	2	5.1
200	$6 \cdot 10^7$	13	3	1	5.7
200	$7 \cdot 10^7$	13	4	2	5.7
200	$5 \cdot 10^7$	14	4	1	5.6

The maximum degree 200 is caused by the limited storage of the UNIVAC 1108 and is no bound for the algorithm.

6. Conclusion

A fast direct method to solve an arbitrary system of linear equations is the well-known Gaussian elimination. However, we have seen that the result may be arbitrary false without taking any precautions (see Hilbert$_7$-matrix). Even if the residual iterations "seem" to converge, one is not sure of the true value of the solution. To achieve guaranteed bounds for the solution one needs a special algorithm (using a specified rounding). We presented such an algorithm with the following advantages:

any floating-point algorithm to achieve a good approximation of the solution is usable

there is no restriction on the matrix or the right-hand side

it is working for interval-matrices, too

no inclusion for the solution or the inverse matrix is needed

interval arithmetic is used very late

the algorithm is very fast.

After finishing the algorithm with step 6) it is proved, that

A is not singular

$Ax = b$ is uniquely solvable

exact bounds for the solution are given.

When the residuals are computed with double precision, we got the following experimental results:

matrices with a condition number up to 10^8 can be trieved when single precision has an 8 decimal digit mantissa

in step 4) usually 4, in very bad conditioned cases up to 16 iterations are necessary

in step 5) one iteration is necessary, in very bad conditioned cases up to four

with an 8 decimal digit mantissa at least 10 decimal digits accuracy is achieved in all components of the solution.

The computing time on the UNIVAC 1108 at the University of Karlsruhe were 2 minutes for degree 100 and 12 minutes for degree 200; the computing time for the Gaussian elimination were always $\frac{1}{6}$, independent of the degree. The algorithms were programmed in ALGOL and FORTRAN and are available.

References

[1] Alefeld, G., Apostolatos, N.: Praktische Anwendung von Abschätzungsformeln bei Iterationsverfahren, Bericht des Instituts für Angewandte Mathematik und Rechenzentrum der Universität Karlsruhe, Januar 1968, 9 p.

[2] Bohlender, G.: Floating-point computation of functions with maximum accuracy. IEEE Computer Society, Symposium on Computer Arithmetic, Dallas 1975, pp. 14–23.

[3] Kaucher, E., Klatte, R., Rump, S. M.: Der dynamische Intervallrechner, Bericht des Instituts für Angewandte Mathematik der Universität Karlsruhe, September 1978, 15 p.

[4] Kaucher, E., Rump, S. M.: Generalized iteration methods for bounds of the solution of fixed point operator equations. (This volume.)

[5] Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing 4, 182–201 (1969).

[6] Kulisch, U.: Grundzüge der Intervallrechnung. Überblicke Mathematik 2 (Laugwitz, D., ed.), pp. 51–98. Mannheim: Bibliographisches Institut 1969.

Dipl.-Math. S. M. Rump
Dr. E. Kaucher
Institut für Angewandte Mathematik
Universität Karlsruhe
Kaiserstrasse 12
D-7500 Karlsruhe
Federal Republic of Germany