2005 International Symposium on Nonlinear
Theory and its Applications (NOLTA2005)
Bruges, Belgium, October 18-21, 2005

A Method of Generating Linear Systems with an Arbitrarily Ill-conditioned Matrix and an Arbitrary Solution

Shinya Miyajima[†], Takeshi Ogita[‡,†] and Shin'ichi Oishi[†,‡]

†Faculty of Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
‡CREST, Japan Science and Technology Agency
Email: shinya_miyajima@aoni.waseda.jp, {ogita,oishi}@waseda.jp

Abstract—A method of generating linear systems is proposed for testing validity and efficiency of numerical verification methods. Using the proposed method, condition number and the exact solution of a linear system can arbitrarily be set by user. In this test, Rump's method of generating matrices with arbitrary condition number can be utilized. Moreover the error-free vector transformation is applied for setting the exact solution. Finally numerical examples are presented to show the usefulness of the proposed method.

1. Introduction

In this paper, we consider generating a linear system

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \ \ b \in \mathbb{R}^n \tag{1}$$

whose true solution has been known in advance for testing validity and efficiency of numerical verification methods (cf. e.g. [3, 4]). By generating such a linear system, we can grasp the exact error of an approximate solution of $Ax = b$.

The purpose of this paper is to propose a method which generates a linear system whose coefficient matrix $A \in \mathbb{F}^{n\times n}$ and right-hand side vector $b \in \mathbb{F}^n$ for $\mathbb{F}$ denoting a set of floating-point numbers when the dimension p ($p \le n$), an anticipated condition number of A and a true solution for $Ax = b$ are set in advance. Although it does not necessarily hold $p = n$, it can be expected that $p \approx n$ by the proposed method. We assume that no underflow occurs.

In the proposed method, a fast and accurate dot product algorithm [2] is applied. By this application, the proposed method generates such a linear system using only ordinary floating-point operations supported by hardware. Additionally in this method, Rump's method [5] which generates a matrix with an arbitrary condition number is utilized. By this utilization, the proposed method can also supply a linear system whose coefficient matrix is arbitrarily ill-conditioned. Finally numerical examples are presented to show the usefulness of the proposed method.

2. Error-Free Vector Transformation

A fast algorithm `DotK` is proposed by Ogita, Rump and Oishi, which computes accurate dot products using only ordinary floating-point operations supported in hardware [2, Algorithm 5.10]. By applying `DotK` to $p, q \in \mathbb{F}^n$, we can compute $\tilde{s} \in \mathbb{F}$ and $v_1, \ldots, v_m \in \mathbb{F}$, $0 \le m \le 2n-1$ such that

$$p^T q = \tilde{s} + \sum_{i=1}^{m} v_i \tag{2}$$

where $\tilde{s}$ is an approximation of the dot product $p^T q$ and $v_1, \ldots, v_m$ correspond to the error of $\tilde{s}$. Using `DotK`, we can usually obtain sufficiently small m by the iterative process.

3. Proposed Method

In this section, we propose the method which generates a linear system with an arbitrarily ill-conditioned matrix and an arbitrary solution.

3.1. Concrete Steps and Properties

In this subsection, we present the concrete steps of the proposed method and some properties of a generated linear system.

First we present Algorithm 1 with respect to the concrete steps of this method.

Algorithm 1 Generation of an n-dimensional linear system $Ax = b$ with $n := p + m$. Assume that an anticipated dimension p and condition number $\kappa \in \mathbb{F}$ of A, and an arbitrary solution $\hat{x} = (x_1, \ldots, x_p)^T \in \mathbb{F}^p$ which becomes the leading p entries of x, have been set in advance.

Step 1 Generate a random matrix $M \in \mathbb{F}^{p\times p}$ whose condition number is approximately equal to κ by Rump's method [5].

Step 2 Compute $b_i \in \mathbb{F}$ and $c_{i1}, \ldots, c_{im} \in \mathbb{F}$, $i = 1, \ldots, p$ such that

$$M(i, 1:p)\hat{x} = b_i + \sum_{j=1}^{m_i} c_{ij} \tag{3}$$

applying `DotK`. Set $m = \max_{1\le i\le n} m_i$. If $m_i < m$, then set $c_{ij} = 0$ for $j = m_i + 1, \ldots, m$.

Step 3 Set $A \in \mathbb{F}^{n\times n}$ with $n = p + m$ as

$$A = \begin{pmatrix} M & -C \\ O_{mp} & I_m \end{pmatrix}, \quad C = \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{p1} & \cdots & c_{pm} \end{pmatrix} \tag{4}$$

where O_{mp} denotes the $m \times p$ matrix of all zeros and I_m the $m \times m$ identity matrix. Set also x and b as

$$x = (x_1, \ldots, x_p, 1, \ldots, 1)^T \in \mathbb{F}^n \tag{5}$$

and

$$b = (b_1, \ldots, b_p, 1 \ldots, 1)^T \in \mathbb{F}^n. \tag{6}$$

From these we obtain a linear system $Ax = b$. □

By (3)–(6), it holds strictly that $Ax = b$. Computational cost of **Step 1** is $O(p^3)$ flops and that from **Step 2** to **Step 3** is $O(p^2)$flops. Therefore computational cost of this algorithm is $O(p^3)$ flops in total.

Next we present Theorem 1 with respect to the condition number of A.

Theorem 1 *Let A be defined as in (4) for a nonsingular matrix $M \in \mathbb{R}^{p\times p}$. Then it holds that*

$$\mathrm{cond}_\alpha(M) \le \mathrm{cond}_\alpha(A), \quad \alpha \ge 1 \tag{7}$$

where $\mathrm{cond}_\alpha(A) := \|A\|_\alpha \cdot \|A^{-1}\|_\alpha$.

Proof. For any matrix $B \in \mathbb{R}^{m\times n}$, it is well-known (e.g. [1, p. 57]) that

$$\|B(i_1 : i_2, j_1 : j_2)\|_\alpha \le \|B\|_\alpha \tag{8}$$

where $1 \le i_1 \le i_2 \le m$ and $1 \le j_1 \le j_2 \le n$. Moreover, it is straightforward to show that

$$A^{-1} = \begin{pmatrix} M^{-1} & M^{-1}C \\ O_{mp} & I_m \end{pmatrix}. \tag{9}$$

From (8) and (9), we obtain $\|M\|_\alpha \le \|A\|_\alpha$ and $\|M^{-1}\|_\alpha \le \|A^{-1}\|_\alpha$ which proves Theorem 1. □

Finally in this subsection, we mention how large m is. Let $\beta \in \mathbb{N}$ and $f_m \in \mathbb{N}$ be the base number and the number of mantissa bits of working floating-point number. Let also $b^* = (b_1^*, \ldots, b_n^*)^T$ and $f_{b_i^*} \in \mathbb{N}$ be as $b^* := M\hat{x}$ and the number of figure (binary if $\beta = 2$) of b_i^*. Then it holds that

$$m = \left\lceil \left(\max_{1\le i\le n} f_{b_i^*} \right) / f_m \right\rceil - 1. \tag{10}$$

3.2. Scaling of Generated Linear Systems

Let $\kappa(A)$ denote the condition number of A as $\kappa(A) := \|A\|_2 \cdot \|A^{-1}\|_2$. Magnitude of the elements of C can become much larger than that of M. In this case $\kappa(A)$ can also become much larger than the anticipated condition number κ so that it may cause instability to solve $Ax = b$ by Gaussian elimination. Therefore in this subsection, we consider the scaling of $Ax = b$ for numerical stability.

Let $D \in \mathbb{F}^{n\times n}$ with $n = p + m$ be as

$$D = \mathrm{diag}(1, \ldots, 1, s_1, \ldots, s_m) \tag{11}$$

where

$$s_i := \beta^{-\alpha_i}\gamma, \quad i = 1, \ldots, m, \tag{12}$$

$$\alpha_i := \left\lceil \log_\beta \left(\max_{1\le j\le m} c_{ij} \right) \right\rceil, \tag{13}$$

$$\gamma := \beta^{-f_m} \min(\beta^\delta, 1) \tag{14}$$

and

$$\delta := \left\lfloor \log_\beta \|M\|_\infty \right\rfloor. \tag{15}$$

Utilizing D we consider the following scaling of $Ax = b$:

$$Gy = h \tag{16}$$

where

$$G := D^{-1}AD = \begin{pmatrix} M & -CD_s \\ O_{mp} & I_m \end{pmatrix}, \tag{17}$$

$$D_s := \mathrm{diag}(s_1, \ldots, s_m), \tag{18}$$

$$y := D^{-1}x = (x_1, \ldots, x_p, \frac{1}{s_1}, \ldots, \frac{1}{s_m})^T \tag{19}$$

and

$$h := D^{-1}b = (b_1, \ldots, b_p, \frac{1}{s_1}, \ldots, \frac{1}{s_m})^T. \tag{20}$$

It follows that

$$G^{-1} = \begin{pmatrix} M^{-1} & M^{-1}CD_s \\ O & I_m \end{pmatrix}. \tag{21}$$

From (12)–(15) it holds approximately that

$$\|CD_s\|_\alpha \lesssim \beta^{-f_m}\|M\|_\alpha \le 1 \tag{22}$$

and

$$\|M^{-1}CD_s\|_\alpha \le \|M^{-1}\|_\alpha\|CD_s\|_\alpha \lesssim \|M^{-1}\|_\alpha. \tag{23}$$

Considering (17) and (21), it can be expected that $\|CD_s\|_\alpha$ and $\|M^{-1}CD_s\|_\alpha$ do not enlarge the condition number significantly, i.e. $\|G\|_\alpha \approx \|M\|_\alpha$ and $\|G^{-1}\|_\alpha \approx \|M^{-1}\|_\alpha$. Therefore it holds approximately that $\kappa(G) \approx \kappa$.

Based on the above discussions, we present Algorithm 2 with respect to the generation of $Gy = h$. Here, $\mathrm{fl}(\cdot)$ denotes the result of floating-point computations, where all operations inside parentheses are executed by ordinary floating-point arithmetic fulfilling rounding mode instruction, especially $\mathrm{fl}_\square(\cdot)$ in round-to-nearest and $\mathrm{fl}_\triangledown(\cdot)$ in round-downward.

Algorithm 2 Generation of a scaled linear system $Gy = h$. Assume that A, x and b have been obtained in advance from Algorithm 1.

Step 1 Compute α_i, $i = 1, \ldots, m$ from (13). Compute also δ, γ and s_i, $i = 1, \ldots, m$ as

$$\delta = \left\lfloor \log_\beta \left(\mathrm{fl}_\triangledown \left(\|M\|_\infty \right) \right) \right\rfloor, \tag{24}$$

$$\gamma = \mathrm{fl}_\square \left(\beta^{-f_m} \min \left(\beta^\delta, 1 \right) \right) \tag{25}$$

and

$$s_i = \mathrm{fl}_\square \left(\beta^{-\alpha_i} \gamma \right). \tag{26}$$

Step 2 Compute $E \in \mathbb{F}^{p \times m}$ and $t_j \in \mathbb{F}$, $j = 1, \ldots, m$ as

$$E(1:p, j) = \mathrm{fl}_\square \left(-s_j \cdot C(1:p, j) \right) \tag{27}$$

and

$$t_j = \mathrm{fl}_\square \left(1/s_j \right). \tag{28}$$

Step 3 Obtain G, y and h as

$$G = \begin{pmatrix} M & E \\ O_{mp} & I_m \end{pmatrix}, \tag{29}$$

$$y = (x_1, \ldots, x_p, t_1, \ldots, t_m)^T \tag{30}$$

and

$$h = (b_1, \ldots, b_p, t_1, \ldots, t_m)^T. \tag{31}$$

From these we obtain a linear system $Gy = h$. □

In this algorithm, powers of β are utilized so that there is no rounding error within the all result obtained from $\mathrm{fl}_\square(\cdot)$. Moreover the results of $\lceil \log_\beta (\cdot) \rceil$ and $\lfloor \log_\beta (\cdot) \rfloor$ is able to be obtained by checking the exponential bits of the corresponding floating-point so that there is also no rounding error within these results. Computational cost of this algorithm is $O(p^2)$ flops and it is negligible compared with that of Algorithm 1.

4. Numerical Examples

In this section, we present numerical examples to show the usefulness of the proposed method. Our computer environment is Pentium IV 3.4GHz CPU. We use Matlab 7.0 with ATLAS and IEEE 754 double precision for all computations. In this environment $\beta = 2$ and $f_m = 53$.

4.1. Numerical Result 1

Let $p = 3$, $\kappa =$ 1e+14 and $\hat{x} = (1, 1, 1)^T$. Using Rump's method, we obtained

$$M = \begin{pmatrix} 0.0184 & 0.1507 & 0.1851 \\ 0.1092 & -0.0172 & -0.2726 \\ -0.4781 & -0.8406 & -0.1840 \end{pmatrix}$$

whose condition number was approximately equal to κ.

Applying `DotK`, it became that $m = 1$. b_1, b_2, b_3 and C were generated as

$$(b_1, b_2, b_3)^T = (0.3542, -0.1807, -1.5027)^T$$

and

$$C = (-0.1388\text{e-}16, -0.0694\text{e-}16, -0.5551\text{e-}16)^T.$$

Therefore we obtained

$$A = \begin{pmatrix} 0.0184 & 0.1507 & 0.1851 & 0.1388\text{e-}16 \\ 0.1092 & -0.0172 & -0.2726 & 0.0694\text{e-}16 \\ -0.4781 & -0.8406 & -0.1840 & 0.5551\text{e-}16 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$b = (0.3542, -0.1807, -1.5027, 1)^T$$

and

$$x = (1, 1, 1, 1)^T.$$

Next, we executed a scaling for M based on Algorithm 2. α_1 and δ were computed as $\alpha_1 = -54$ and $\delta = 1$. In this case, γ was computed as

$$\gamma = \mathrm{fl}_\square \left(2^{-53} \min \left(2^1, 1 \right) \right) = 2^{-53}.$$

Utilizing α_1 and γ, s_1 was computed as

$$s_1 = \mathrm{fl}_\square \left(2^{54} \cdot 2^{-53} \right) = 2.$$

From s_1, E and t_1 were computed as

$$E = (0.2776\text{e-}16, 0.1388\text{e-}16, 1.1102\text{e-}15)^T$$

and $t_1 = 0.5$. From E and t_1, we obtained

$$G = \begin{pmatrix} 0.0184 & 0.1507 & 0.1851 & 0.2776\text{e-}16 \\ 0.1092 & -0.0172 & -0.2726 & 0.1388\text{e-}16 \\ -0.4781 & -0.8406 & -0.1840 & 1.11102\text{e-}15 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$y = (1, 1, 1, 0.5)^T,$$

and

$$h = (0.3542, -0.1807, -1.5027, 0.5000)^T.$$

When we solved $Gy = h$ with iterative refinement (e.g. [1, p.127]), we obtained a numerical solution $\tilde{y}$ as

$$\tilde{y} = (1.0000, 1.0000, 1.0000, 0.50000)^T.$$

Applying the verification method proposed by Oishi and Rump [3], we obtained

$$\|G^{-1}h - \tilde{y}\|_\infty \leq 7.3657\text{e-}3 =: \varepsilon.$$

We can confirm that the interval $[\tilde{y}_i - \varepsilon, \tilde{y}_i + \varepsilon]$ includes y_i for $i = 1, \ldots, 4$. Here, $\kappa(A) \approx$ 1.0006e+14 and $\kappa(G) \approx$ 1.0006e+14. it can be seen that $\kappa(A) \approx \kappa$ and $\kappa(G) \approx \kappa$. These results identify Theorem 1 and the consideration in Subsection 3.2.

Table 1: t_R, t_e and t_e/t_R for various p.

p	t_R	t_e	t_e/t_R
500	6.03	0.64	0.11
1000	46.0	2.72	0.059
1500	153.1	5.84	0.038
2000	367.9	12.2	0.033

Table 2: t_R, t_e and t_e/t_R for various κ.

κ	t_R	t_e	t_e/t_R
1e+20	11.5	0.16	0.014
1e+30	14.0	0.16	0.011
1e+40	11.3	0.16	0.014
1e+50	10.1	0.16	0.015

Table 3: m, t_R, t_e and t_e/t_R for various $\hat{x}$.

$\hat{x}$	m	t_R	t_e	t_e/t_R
$(2^1, \ldots, 2^{100})$	2	0.063	0.031	0.49
$(4^1, \ldots, 4^{100})$	4	0.063	0.062	0.98
$(8^1, \ldots, 8^{100})$	6	0.063	0.078	1.23
$(16^1, \ldots, 16^{100})$	8	0.062	0.094	1.52
$(32^1, \ldots, 32^{100})$	10	0.062	0.20	3.27
$(64^1, \ldots, 64^{100})$	12	0.063	0.28	4.46

$\hat{x}$	$\kappa(A)$	$\kappa(G)$
$(2^1, \ldots, 2^{100})$	1.78e+34	1.00e+10
$(4^1, \ldots, 4^{100})$	8.52e+71	1.00e+10
$(8^1, \ldots, 8^{100})$	2.92e+99	1.00e+10
$(16^1, \ldots, 16^{100})$	5.88e+129	1.00e+10
$(32^1, \ldots, 32^{100})$	1.08e+162	1.00e+10
$(64^1, \ldots, 64^{100})$	2.64e+194	1.00e+10

4.2. Numerical Result 2

We next consider computational speed of our method for various dimensions. Let t_R and t_e be the computing time (sec) of Rump's method and the other part in Algorithms 1 and 2. Table 1 displays t_R, t_e and their ratio t_e/t_R for various p. Here, we set $\hat{x} = (1, \ldots, 1)^T$ and κ = 1e+10.

It became that $m = 1$ for all examples in Table 1. By Table 1, it can be seen that t_e/t_R becomes smaller as p increases. From this we can confirm that computational cost of the part except for Rump's method becomes relatively smaller as p increases for fixed $\hat{x}$ and κ.

4.3. Numerical Result 3

We next treated the cases for various condition numbers. Let $\hat{x} = (1, \ldots, 1)^T$ and $p = 250$. Table 2 displays t_R, t_e and t_e/t_R for various κ.

It became that $m = 1$ for all examples in Table 2. By Table 2, we can confirm that computational cost of Algorithms 1 and 2 do not depend on κ for fixed $\hat{x}$ and p.

4.4. Numerical Result 4

We finally considered the cases for varying $\hat{x}$. Let p = 100 and κ = 1e+10. Table 3 displays m, t_R, t_e, t_e/t_R, $\kappa(A)$ and $\kappa(G)$ for various $\hat{x}$.

By Table 3, it can be seen that t_e becomes larger as the exponential orders of the entries of $\hat{x}$ disperse. The reason is that the number of iterations in `DotK` increases as they disperse.

We can confirm that m increases as the exponential orders of the entries of $\hat{x}$ disperse. A reason of this result is that all entries of A were almost the same order of magnitude for all examples in Table 3. The result of each example in Table 3 coincides with (10).

$\kappa(A)$ became much larger than κ as the exponential orders of the entries of $\hat{x}$ disperse. The reason is that the magnitude of the elements in C became extremely large. On the other hand, it can be seen that $\kappa(G) \approx \kappa$ for all examples in Table 3. This result identifies the consideration in Subsection 3.2 and shows the efficiency of Algorithm 2.

5. Conclusion

In this paper, we proposed the method which generates a linear system when the dimension, condition number and a true solution are set in advance. Finally we presented numerical examples to show the usefulness of the proposed method.

References

[1] G. H. Golub and C. F. van Loan, *Matrix Computations*, Third edition, The Johns Hopkins University Press, Baltimore and London, 1996.

[2] T. Ogita, S. M. Rump, S. Oishi, "Accurate Sum and Dot Product," *SIAM J. Sci. Comp.*, 26:6, pp. 1955–1988, 2005.

[3] S. Oishi and S. M. Rump, "Fast Verification of Solutions of Matrix Equations," *Numer. Math.*, 90:4, pp. 755–773, 2002.

[4] S. M. Rump, "Verification Methods for Dense and Sparse Systems of Equations," *Topics in Validated Computations (J. Herzberger, ed.)*, Elsevier, Amsterdam, pp. 63–135, 1994.

[5] S. M. Rump, "A Class of Arbitrarily Ill-conditioned Floating-Point Matrices," *SIAM J. Mat. Anal. Appl.* 12:4, pp. 645–653, 1991.