Journal of Discrete Algorithms 10 (2012) 98–109

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

A condensation-based application of Cramer's rule for solving large-scale linear systems

Ken Habgood *, Itamar Arel

Department of Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, TN, USA

ARTICLE INFO

Article history:
Received 4 April 2010
Received in revised form 27 June 2011
Accepted 27 June 2011
Available online 1 July 2011

Keywords:
Cramer's rule
Linear systems
Matrix condensation

ABSTRACT

State-of-the-art software packages for solving large-scale linear systems are predominantly founded on Gaussian elimination techniques (e.g. LU-decomposition). This paper presents an efficient framework for solving large-scale linear systems by means of a novel utilization of Cramer's rule. While the latter is often perceived to be impractical when considered for large systems, it is shown that the algorithm proposed retains an $\mathcal{O}(N^3)$ complexity with pragmatic forward and backward stability properties. Empirical results are provided to substantiate the stated accuracy and computational complexity claims.

1. Introduction

Systems of linear equations are central to many science and engineering application domains. Fast linear solvers generally use a form of Gaussian elimination [8], the most common of which is LU-decomposition. The latter involves a computation complexity of $W_{LU} \approx \frac{2}{3}N^3$ [11], where N denotes the number of linearly independent columns in a matrix. The factor 2 accounts for one addition and one multiplication. If only multiplications are considered, then $W_{LU} \approx \frac{N^3}{3}$, which is the operation count sometimes quoted in the literature.

In most implementations of Gaussian elimination, the row with the largest lead value and the first row are interchanged during each reduction. This is referred to as partial pivoting and facilitates the minimization of truncation errors. Historically, Cramer's rule has been considered inaccurate when compared to these methods. As this paper will discuss, the perceived inaccuracy does not originate from Cramer's rule but rather from the method utilized for obtaining determinants.

This paper revisits Cramer's rule [4] and introduces an alternative framework to the traditional LU-decomposition methods offering similar computational complexity and storage requirements. To the best of the authors' knowledge, this is the first work to demonstrate such characterization. In utilizing a technique similar to partial pivoting to calculate determinant values, the algorithm's stability properties are derived and shown to be comparable to LU-decomposition for asymmetric systems.

The rest of this paper is structured as follows. Section 2 describes the algorithm and presents its computational complexity. Section 3 discusses the stability of the algorithm including forward and backward error analysis. Section 4 presents empirical results to support the proposed complexity assertions. Finally, a brief discussion and drawn conclusions are provided in Section 5.

* Corresponding author.
E-mail addresses: habgood@eecs.utk.edu (K. Habgood), itamar@eecs.utk.edu (I. Arel).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2011.06.007

2. Revisiting Cramer's rule

The proposed algorithm centers on the mathematically elegant Cramer's rule, which states that the components of the solution to a linear system in the form $Ax = b$ (where A is invertible) are given by

$$x_i = \det\big(A_i(b)\big)/\det(A), \tag{1}$$

where $A_i(b)$ denotes the matrix A with its ith column replaced by b [10]. Unfortunately, when working with large matrices, Cramer's rule is generally considered impractical. This stems from the fact that the determinant values are calculated via minors. As the number of variables increases, the determinant computation becomes unwieldy [3]. The time complexity is widely quoted as $\mathcal{O}(N!)$, which would make it useless for any practical application when compared to a method like LU-decomposition at $\mathcal{O}(N^3)$. Fortunately, Cramer's rule can be realized in far lower complexity than $\mathcal{O}(N!)$. The complexity of Cramer's rule depends exclusively on the determinant calculations. If the determinants are calculated via minors, that complexity holds. In an effort to overcome this limitation, matrix condensation techniques can reduce the size of the original matrix to one that may be solved efficiently and quickly. As a result, Cramer's rule becomes $\mathcal{O}(N^3)$ similar to LU-decomposition.

The other concern with Cramer's rule pertains to the numerical instability, which is less studied [10]. A simple example put forward in [12] suggests that Cramer's rule is unsatisfactory even for 2-by-2 systems, mainly because of round error difficulties. However, that argument heavily depends on the method for obtaining the determinants. If an accurate method for evaluating determinants is used then Cramer's rule can, in fact, be numerically stable. In fact, a later paper [5] revisited the cited example and provided an example where Cramer's rule yielded a highly accurate answer while Gaussian elimination with pivoting a poor one. We thus provide a detailed treatment of the stability and accuracy aspects of the proposed determinant calculations employed by the proposed algorithm.

2.1. Chio's condensation

Chio's condensation [6] method reduces a matrix of order N to order $N-1$ when evaluating its determinant. As will be shown, repeating the procedure numerous times can reduce a large matrix to a size convenient for the application of Cramer's rule. Chio's pivotal condensation theorem is described as follows. Let $A = [a_{ij}]$ be an $N \times N$ matrix for which $a_{11} \neq 0$. Let D denote the matrix obtained by replacing elements a_{ij} not in the lead row or lead column by $\left|\begin{smallmatrix} a_{11} & a_{1j} \\ a_{i1} & a_{ij} \end{smallmatrix}\right|$, then it can be shown that $|A| = \frac{|D|}{a_{11}^{n-2}}$ [6].

Note that this process replaces each element in the original matrix with a 2×2 determinant consisting of the a_{11} element, the top value in the element's column, the first value in the element's row and the element being replaced. The calculated value of this 2×2 determinant replaces the initial $a_{i,j}$ with $a'_{i,j}$. The first column and first row are discarded, thereby reducing the original $N \times N$ matrix to an $(N-1) \times (N-1)$ matrix with an equivalent determinant. As an example, we consider the following 3×3 matrix:

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad \text{and its condensed form:}$$

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & (a_{11}a_{22} - a_{21}a_{12}) & (a_{11}a_{23} - a_{21}a_{13}) \\ 0 & (a_{11}a_{32} - a_{31}a_{12}) & (a_{11}a_{33} - a_{31}a_{13}) \end{vmatrix} = \begin{vmatrix} \times & \times & \times \\ \times & a'_{22} & a'_{23} \\ \times & a'_{32} & a'_{33} \end{vmatrix}$$

Obtaining each 2×2 determinant requires two multiplications and one subtraction. However, if the value of $a_{1,1}$ is one, then only a single multiplication is required. In the example above we note that $a_{1,1}$ is used in each element as a multiplier to the matrix element, for example, the equation for the matrix element in position $(2, 2)$ is $a_{11}a_{22} - a_{21}a_{12}$. If in this situation $a_{11} = 1$, then the equation changes to $a_{22} - a_{21}a_{12}$. This holds true for every element in the matrix. Therefore for each condensation step k, if $a_{kk} = 1$ then $(N-k)^2$ multiplications are removed.

In order to guarantee $a_{kk} = 1$, an entire row or column must be divided by a_{kk}. This value would need to be stored because the determinant value calculated by Chio's condensation would be reduced by this factor. To find the true value at the end of the condensation, the calculated answer would need to be multiplied by each a_{kk} that was factored out. Multiplying all of these values over numerous condensation steps would result in an extremely large number that would exceed the floating point range of most computers. This is where the elegance of Cramer's rule is exploited. Cramer's rule determines each variable by a ratio of determinants, $x_i = \det(A_i(b))/\det(A)$. Given that both determinants are from the same condensation line, they both are reduced by the same a_{kk} values. The a_{kk} values factored out during Chio's condensation cancel during the application of Cramer's rule. This allows the algorithm to simply discard the a_{kk} values in the final computations. The actual determinant values are not correct, however the ratio evaluated at the core of Cramer's rule remains correct.