ICNAAM 2013, Rhodos, Greece, AIP Conf.Proceedings, No.1558, pp.1936-1939, AIP Publishing, http://dx.doi.org/10.1063/1.4825912, 2013

Modified Gaussian Elimination without Division Operations

Vaclav Skala

University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering
Univerzitni 8, CZ 306 14 Plzen, Czech Republic

Abstract. A new modified method based on the Gaussian elimination method for solution of linear system of equations in the projective space is formulated. It is based on application of projective extension of the Euclidean space and use of homogeneous coordinates. It leads to an elimination of division operation and higher precision due to division operation elimination. The approach is based on understanding that a solution of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ is equivalent to the extended cross-product, i.e. $\boldsymbol{x} = \boldsymbol{a}_1 \times ... \times \boldsymbol{a}_n$. As it can be seen there no division is needed. Use of the projective representation enables to avoid division operation and use advantages of the matrix-vector architectures. Division operations have to be used only if the final result of computation has to be in the Euclidean representation. The proposed method was implemented in C# and C++ and experimentally verified. It is especially convenient for computations on GPUs based architectures.

Keywords: Numerical methods (mathematics), partial differential equations, numerical linear algebra
PACS: 02.60.-x , 02.30.Jr , 02.60 Dc

INTRODUCTION

Many physical and computational problems result into a solution of linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$. There are many methods already developed. However, two fundamental classes can be distinguished. Explicit methods, e.g. Gaussian elimination method [1], [6], are of $O(N^3)$ computational complexity in general, while iterative methods are of $O(kN^2)$ computational complexity, where: N is a matrix size and k is number of iterations. Another group of problems lead to homogeneous system of equations, i.e. $\boldsymbol{Ax} = \boldsymbol{0}$. The solution of $\boldsymbol{Ax} = \boldsymbol{b}$ can be converted to a problem $\boldsymbol{A\tilde{x}} = \boldsymbol{0}$, where $\tilde{\boldsymbol{x}}$ is a vector in the projective space. Iterative methods are preferred as an approximate solution is usually required and in this case they are faster if a solution converges faster due to stability issues.

A solution of a linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ can be transformed to $\tilde{\boldsymbol{x}} = \boldsymbol{a}_1 \times ... \times \boldsymbol{a}_n$, using the extended cross-product [10], [11], where $\tilde{\boldsymbol{x}}$ is a vector in the projective space. This evokes a question: "***Why a solution of a linear system of equations requires division operations***"?

It is known that the precision of computation is given by rules as stored values x resp. y actually represents all values in intervals, i.e. $x \in [a, b]$, resp. $y \in [c, d]$. The precision of operations is given as [1], [9]

$$\begin{aligned} x + y &\in [a + c, b + d] \qquad x - y \in [a - d, b - c] \\ x \times y &= [min(ac, ad, bc, bd), max(ac, ad, bc, bd)] \\ x/y &= [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)] \ \textbf{if}\ y \neq 0 \end{aligned} \tag{1}$$

It can be seen that the division operation causes significant imprecision in computations except of addition/subtraction with numbers with significantly different exponents.

The mutual conversions between the Euclidean space and the projective space, i.e. the projective extension of the Euclidean space, is given as

$$x = \tilde{x}/\tilde{w} \text{ and } y = \tilde{z}/\tilde{w} \quad \tilde{w} \neq 0 \tag{2}$$

where: $[\tilde{x}, \tilde{y}: \tilde{w}]$ are homogenous coordinates of the Euclidean coordinates (x, y) if we consider that geometrically. In mathematics a different notation $\tilde{\boldsymbol{x}} = [\tilde{x}_0: \tilde{x}_1, ..., \tilde{x}_n]$ is used, where $\tilde{x}_0$ is the homogeneous coordinate, i.e. the $\tilde{w}$ value, $\tilde{x} = \tilde{x}_1$, and $\tilde{y} = \tilde{x}_2$ etc. There are different notations used and in computer graphics and related fields the first notation is common.

GAUSSIAN ELIMINATION METHOD

The Gaussian elimination method for solving linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ relies on a strategy of the gradual upper triangular matrix generation. When done, there is a backward cycle computing the $\boldsymbol{x}$ vector values.

The Gaussian elimination method can be described as [3]:

```
for k := 1 to n-1 do
{   # Find pivot for column k #
    ii := max_arg (abs(a[i, k]), i = k ... n);   #Finds the maximum pivot #
    if abs(a[ii, k]) ≤ eps then {ERROR ("Matrix is singular!"); exit }
    swap_rows (k, ii);     # swaps rows k, ii #
    # for all rows below pivot #
    for i := k + 1 to n do
    {   # for all remaining elements in the current row #
        for j := k + 1 to n+1 do
            a[i, j] := a[i, j] - a[k, j] * (a[i, k] / a[k, k]);
        # fill lower triangular matrix with zeros if needed #
        a[i, k] := 0
    }
};
for i := n downto 1 do              # backward cycle #
{   s:=0;
    for j := i+1 to k do
        s := s + a[i,j] * x[j];
    x[i] := (a[i,n+1) - s) / a[i,i];
}
```

Algorithm 1. Gaussian elimination

where: a[i,j] are elements of the extended matrix, i.e. a matrix which last column is the $\boldsymbol{b}$ vector. It can be seen that the there is a division by the a[k,k] diagonal element, which can lead to division by a value equal or close to zero. The Gaussian elimination has $O(N^3)$ computational complexity as the pivot finding is of $O(N)$ complexity only [4]. Also there is a question what is the ***eps*** value, i.e. value saying that the matrix is close to singular.

PROJECTIVE MODIFICATION OF THE GAUSSIAN ELIMINATION METHOD

The Gauss elimination method for solving linear needs to find a maximum pivotal element for a denominator in order to avoid the division operation by a value closed to or equal to zero. It is possible to use projective representation [8], [13], [12]. Let us consider the following data structure constructions:

- projective scalar $\tilde{a} = [a_0 : a]$
- projective vector $\tilde{\boldsymbol{a}} = [a_0 : a_1, \dots, a_n]$
- matrix of projective vectors $\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{a}_{1,0} & \tilde{a}_{1,1} & \cdots & \tilde{a}_{1,n} \\ \tilde{a}_{2,0} & \tilde{a}_{2,1} & \cdots & \tilde{a}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{n,0} & \tilde{a}_{n,1} & \cdots & \tilde{a}_{n,n} \end{bmatrix}$, i.e. each row is a projective vector

Now a projective reformulation of the Gaussian elimination of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ can be made. Let an extended matrix $\tilde{\boldsymbol{A}}$, i.e. containing the $\boldsymbol{b}$ vector, is defined as a matrix of projective vectors

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{a}_{1,0} & \tilde{a}_{1,1} & \cdots & \tilde{a}_{1,n} & \tilde{a}_{1,n+1} \\ \tilde{a}_{2,0} & \tilde{a}_{2,1} & \cdots & \tilde{a}_{2,n} & \tilde{a}_{2,n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{a}_{n,0} & \tilde{a}_{n,1} & \cdots & \tilde{a}_{n,n} & \tilde{a}_{n,n+1} \end{bmatrix} \tag{3}$$

where: $\tilde{a}_{i,0}$ is the homogeneous value for the whole i[th] row (for a simplicity let $\tilde{a}_{i,0} = 1$ for all i), $\tilde{a}_{*,n+1}$ represents the $\boldsymbol{b}$ vector and $a_{i,j} = \frac{\tilde{a}_{i,j}}{\tilde{a}_{i,0}}$ for all i, j. The computational step of the Gaussian elimination can be modified to:

$$a_{i,j} = a_{i,j} - \frac{a_{i,k}a_{k,j}}{a_{k,k}} = \frac{\tilde{a}_{i,j}}{\tilde{a}_{i,0}} - \frac{\frac{\tilde{a}_{i,k}\tilde{a}_{k,j}}{\tilde{a}_{i,0}\tilde{a}_{k,0}}}{\frac{\tilde{a}_{k,k}}{\tilde{a}_{k,0}}} = \frac{\tilde{a}_{i,j}\tilde{a}_{k,k} - \tilde{a}_{i,k}\tilde{a}_{k,j}}{\tilde{a}_{i,0}\tilde{a}_{k,k}} \triangleq [\tilde{a}_{i,0}\tilde{a}_{k,k} : \tilde{a}_{i,j}\tilde{a}_{k,k} - \tilde{a}_{i,k}\tilde{a}_{k,j}] \tag{4}$$

where: $\triangleq$ means projectively equivalent.

It can be seen that the precision of computation is better due to fraction representation with the floating point representation, but exponents tend to grow or decreases that could lead to an exponent overflow or underflow. The special routine "Normalize_row_exponents" has to be called at the end of each row computation. It subtracts exponent of the homogeneous value $\tilde{a}_{i,0}$ from the other $\tilde{a}_{i,*}$ values in the given row. This is actually made by bitwise masking operations. It should be noted that $\tilde{a}_{i,0}\tilde{a}_{k,k}$ is a common factor for the whole i[th] row.

```
for k := 1 to n-1 do          # the principal algorithm – no optimization; a[i, j] = ã_{i,j} #
{   ii := max_arg_projective (abs(a[i, k]), i = k ... n);   #Finds the maximum pivot #
    if abs(a[ii, k]) ≤ eps then {ERROR ("Matrix is singular!"); exit }
    swap_rows (k, ii);     # swaps rows k, ii #
    # for all rows below pivot #
    par for i := k+1 to n do # cycle steps can be performed in parallel #
    {   for j := k+1 to n+1 do
            a[i,j] := a[i,j] * a[k,k] - a[i,k] * a[k,j];
        a[i,0] := a[i,0] * a[k,k];
        # normalize exponents of the i^th row according to a[i,0] exponent value #
        Normalize_row_exponents (a[i,j], j = k+1 ... n)
    }
}; # this MUST be done completely in the projective representation #
for i := n downto 1 do   # this MUST be done completely in the projective representation #
{   s:=0; #symbolic code only #
    for j := i+1 to k do
        s := s + a[i,j] * x[j];
    x[i] := (a[i,n+1) - s) / a[i,i];
}
```

Algorithm 2. Proposed projective modification of the Gaussian elimination

The "max_arg_projective" procedure finds maximum pivot. This is a little bit tricky as it has to be made in the projective representation and it is based on a test $a_{i,k} * a_{i+j,0} > a_{i+j,k} * a_{i,0}$ - no division is needed.

ALGORITHM COMPLEXITY ANALYSIS

The proposed modification actually converts the division operation to a multiplication and the division operation is actually hidden into the homogeneous coordinate. Let us consider the relative timing of operations in the floating point representation, see Table 1.

TABLE 1. Relative timing of floating operations for a 1,5GHz Intel CPU based on clocks

Timing	±	*	/	<	:=
Clocks	31	31	78	149	16

It means that the division operation is approx. 2 times slower than multiplication in the floating point representation. In both cases the algorithms are of $O(N^3)$ computational complexity. However in the proposed algorithm the division is replaced by a multiplication and update of the homogeneous value is made for each row, i.e. with $O(N^2)$ computational complexity only. The exponent normalization is fast as it is actually implemented as a bitwise **and**/**or** operation. If this operation would be supported by hardware, timing is negligible.

TABLE 2. Main computational complexities

Gaussian elimination	Proposed algorithm
for j := k+1 **to** n+1 **do** # $O(N^3)$ #	**for** j := k+1 **to** n+1 **do** # $O(N^3)$ #
a[i,j] := a[i,j] - a[i,k] * a[k,j] / a[k,k];	a[i,j] := a[i,j] * a[k,k] - a[i,k] * a[k,j];
	a[i,0] := a[i,0] * a[k,k]; # $O(N^2)$ #
	Normalize (a[i,j], k+1, n) # $O(N^3)$ #

Table 2 presents principal computational complexities showing that the proposed algorithm is to be faster. It should be noted that implementation of the "Normalize_row_exponents" procedure is a little bit tricky as it modifies exponent value directly in the floating point representation and low-level programming is needed.

EXPERIMENTAL RESULTS

The proposed modification was tested and properties experimentally proved on Hilbert's matrix inversion.
The Hilbert's matrix is given as

$$H_{ij} = \frac{1}{i+j-1} \tag{5}$$

and the inversion is exactly given as

$$H_{ij}^{-1} = (-1)^{i+j}(i+j-1)\binom{n+i-1}{n-j}\binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2 \tag{6}$$

This gives us an advantage of precision and stability evaluation as the conditional number of the $n \times n$ Hilbert matrix grows as $O((1+\sqrt{2})^{4n}/\sqrt{n})$.

The experimental results proved speed up higher than 10% in spite of the fact that the "Normalize_row_exponents" procedure was not optimized as it depends on implementation of the floating point representation and hardware used. As far as the stability issues are concerned the stability of Hilbert matrix inverse was order of magnitude better than the original Gaussian elimination.

CONCLUSION

A new modification of Gaussian elimination method based on projective representation was introduced. The proposed approach uses projective extension of the Euclidean representation and generally eliminates division operation, which is the slowest operation in the floating point representation except of the comparison operation. As a projective scalar value is represented as two rational numbers in the floating point representation, the final precision is nearly equivalent to double length mantissa use, similarly for projective vectors.

The proposed modification was tested and verified by the inverse Hilbert's matrix computation. Higher precision and speed-up of computation were proved. The proposed projective modification is also convenient for parallel processing and use on GPU similar architectures. Other interesting applications of the projective representation can be found in [7], [10], [11].

ACKNOWLEDGMENTS

The author thanks to students and colleagues at the University of West Bohemia, Plzen and VSB-Technical University, Ostrava for their critical comments, discussions and especially to Vit Ondracka and Jan Kaiser for another implementation, optimization C++ code and verification of the proposed algorithm. This project was supported by the Ministry of Education of the Czech Republic, projects No.LH12181 and LG13047.

REFERENCES

1. Atkinson,K.A.: An Introduction to Numerical Analysis, New York: John Wiley & Sons, ISBN 978-0-471-50023-0, 1989
2. Agoston,M.K.: Computer Graphics and Geometric Modeling - Mathematics, ISBN 1-58233-817-2, Springer, 2005
3. Bronson,R., Costa,G.B.: Matrix Methods: Applied Linear Algebra, Academic Press, ISBN 978-0-12-374427-2, 2009
4. Fang,X.G., Havas,G.: On the worst-case complexity of integer Gaussian elimination, Proceedings of the Symposium on Symbolic and Algebraic Computation. ISSAC '97, ACM. pp. 28–31, 1997
5. Goldman,R.N., Sederberg,T.V., Anderson,D.C.: Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves, Computer Aided geometric design, Vol.1, pp.327-356, 1984
6. Grcar,J.: Mathematicians of Gaussian elimination, Notices of the American Mathematical Society 58 (6): 782–792, 2011
7. Johnson,M.: Proof by Duality: or the Discovery, of "New" Theorems, Mathematics Today, December 1996
8. Press,W.H, Teukolsky,S.A., Vetterling,W.T., Flannery, B.P.: Numerical Recipes: The Art of Scientific Computing (3rd ed.), New York, Cambridge University Press, ISBN 978-0-521-88068-8, 2007
9. Rump,S.M.: Realiability in Computing, The role of Interval Methods in Scientific Computing, Academic Press, 1988
10. Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, ISSN 0097-8493, Vol. 32, No.1, pp.120-127, 2008
11. Skala,V.: Projective Geometry and Duality for Graphics, Games and Visualization - Course SIGGRAPH Asia 2012, Singapore, ISBN 978-1-4503-1757-3, 2012
12. Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, 6th Int.Conf. Aplimat, Bratislava, ISBN 978-80-969562-00-8, pp. 125-130, 2007
13. Yamaguchi,F.:Computer-Aided Geometric Design: A Totally Four-Dimensional Approach, Springer, 2002