A PRECISION OF COMPUTATION IN THE PROJECTIVE SPACE

VACLAV SKALA[1], VIT ONDRACKA
[1]Department of Computer Science and Engineering
University of West Bohemia
Univerzitni 8, CZ 306 14 Plzen
CZECH REPUBLIC
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Abstract: - Precision of computation and stability are the key issues in all computational methods. There are a lot of problems that lead to a "nearly singular" formulation and if standard approaches are taken wrong results are usually obtained. The projective formulation of many computational problems seems to be very appealing as the division operation is not needed if result(s) can remain in the projective representation.
This paper focuses on computational precision using the projective space representation. Properties of this approach are demonstrated on an inversion of the Hilbert matrix, as the inverse is known analytically and determinant converges to zero. Also, we will compare the proposed approach with the standard method for solving linear systems of equations – the comparison is based on pivoted Gaussian method and its projective variant, using the previously developed library PLib for the .NET environment.
The paper proves that elimination of the division operation is entirely possible while preserving the precision of the calculation and simplicity of code. This could even lead to a significant performance boost with appropriate hardware support.

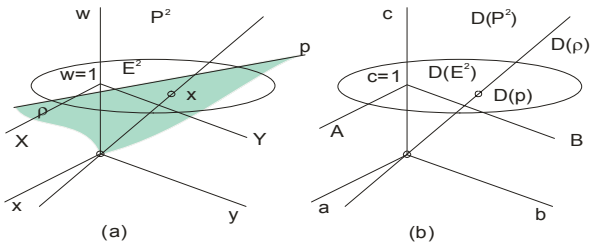
Key-Words: - computer graphics, numerical algorithms, projective space, numerical library

1 Introduction

Many problems solved in computer graphics, computer vision, visualization etc. require fast and robust computations, usually using the Euclidean representation. Many of these problems can be transformed into the projective representation, using the homogeneous coordinates. This often leads to improvement in terms of stability, robustness and/or speed of the calculation.
The projective space is often used in computer graphics and computer vision fields to represent geometric transformations. Recently it was shown that the solution of linear systems of equations is equivalent to the generalized cross product [10]-[12], where no division operation is needed. The projective space also leads to new formulation of known problems, often resulting in new more robust and stable algorithms [10], [13], [14].

2. Projective space and duality

Homogeneous coordinates are widely used in computer graphics applications, usually connected with geometric transformations, such as rotation, scaling, translation and projection, etc. In many cases, homogeneous coordinates are only seen as a "mathematical tool" that makes a simple description of geometric transformations possible. There are many "invisible" impacts on the algorithm design that may lead to new, faster and robust algorithms, which can also be supported in GPU hardware. Fig.1.a. presents a geometrical interpretation of Euclidean and projective spaces.



Euclidean, projective and dual space representations
Figure 1

The point **x** is defined as a point in E^2 with coordinates **X**=(X,Y) or as a point with homogeneous coordinates $[x,y,w]^T$, where w usually equals 1. The point **x** is actually a "line" without the origin $[0,0,0]^T$ in the projective space P^2, and $X = x/w$ and $Y = y/w$ where $w \neq 0$. It can be seen that the line $\mathbf{p} \in E^2$ is actually a plane **ρ** without the origin $[0,0,0]^T$ in the projective space P^2, i.e. a line **p** in the Euclidean is defined as:

$$aX + bY + c = 0 \qquad (1)$$

Any $w \neq 0$ can multiply this equation without any effect on the geometry and we get a representation in the projective P^2 space as follows:

$$ax + by + cw = 0, \quad w \neq 0 \tag{2}$$

In dual representation, see Fig.1.b, the plane ρ can be represented as a line $D(\rho) \in D(P^2)$ or as a point $D(\mathbf{p}) \in D(E^2)$, when a projection is made, e.g. for c = 1. A complete theory on projective spaces can be found in [1], [3], [4], [15]. On the other hand, there is a principle of duality that is useful when deriving a formula. The principle states that any theorem remains true when we interchange the words "point" and "line", "lie on" and "pass through", "join" and "intersection" and so on. Once the theorem has been established, the dual theorem is obtained as described above [6]. In other words, the principle of duality in E^2 says that in all theorems it is possible to substitute the term "point" by the term "line" and the term "line" by the term "point" and the given theorem stays valid. This helps a lot in solving some geometrical cases.

Definition 1

The cross product of the two vectors $\mathbf{x}_1 = [x_1, y_1, w_1]^T$ and $\mathbf{x}_2 = [x_2, y_2, w_2]^T$ is defined as:

$$\mathbf{x}_1 \times \mathbf{x}_2 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \tag{3}$$

where: $\mathbf{i} = [1,0,0]^T$, $\mathbf{j} = [0,1,0]^T$, $\mathbf{k} = [0,0,1]^T$

Please, note that the homogeneous coordinates are used.

Theorem 1

Let two points $\mathbf{x}_1$ and $\mathbf{x}_2$ be given in the projective space. Then the coefficients of the line $\mathbf{p}$, which is defined by those two points, are determined as the cross product of their homogeneous coordinates.

$$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 \tag{4}$$

Proof 1

Let the line $\mathbf{p} \in E^2$ be defined in homogeneous coordinates as

$$\mathbf{ax + by + cw = 0} \tag{5}$$

We are actually looking for a solution to the following equations:

$$\mathbf{p}^T \mathbf{x}_1 = 0 \quad \text{and} \quad \mathbf{p}^T \mathbf{x}_2 = 0 \tag{6}$$

where: $\mathbf{p} = [a,b,c]^T$

It means that any point $\mathbf{x}$ that lies on the line $\mathbf{p}$ must satisfy both the equations above and the equation

$$\mathbf{p}^T \mathbf{x} = 0 \tag{7}$$

in other words the vector $\mathbf{p}$ is defined as

$$\mathbf{p} = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \tag{8}$$

We can write

$$\left(\mathbf{x}_1 \times \mathbf{x}_2\right)^T \mathbf{x} = 0 \tag{9}$$

i.e.

$$\det \begin{bmatrix} x & y & w \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0 \tag{10}$$

Then evaluating the determinant, we get the line coefficients as:

$$a = \det \begin{bmatrix} y_1 & w_1 \\ y_2 & w_2 \end{bmatrix} \quad b = -\det \begin{bmatrix} x_1 & w_1 \\ x_2 & w_2 \end{bmatrix} \quad c = \det \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} \tag{11}$$

Note: For w = 1 we get the standard cross product formula and the cross product defines the line $\mathbf{p}$, i.e.

$$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 \tag{12}$$

where: $\mathbf{p} = [a,b,c]^T$

Theorem 2

Let two lines $\mathbf{p}_1$ and $\mathbf{p}_2$ be given in the projective space. Then the homogeneous coordinates of the point $\mathbf{x}$ at the intersection of those two lines are given by the cross product of vectors of their coordinates

$$\mathbf{x} = \mathbf{p}_1 \times \mathbf{p}_2 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix} \tag{13}$$

where: $\mathbf{i} = [1,0,0]^T$, $\mathbf{j} = [0,1,0]^T$, $\mathbf{k} = [0,0,1]^T$

Note: Actually two equations

$$\mathbf{p}_1^T \mathbf{x} = 0 \quad \text{and} \quad \mathbf{p}_2^T \mathbf{x} = 0 \tag{14}$$

are solved.

Proof 2

An immediate result of Theorem 1 and the duality principle.

The E^3 case is a little bit more complex as a point is dual to a plane and vice versa. It should be noted that a line in E^3 **is not** dual to a line in the dual space, for details see [6], [15].

In the E^3 case the plane ρ is given by three points $\mathbf{X}=(X,Y,Z)$ or by points in the homogeneous coordinates $\mathbf{x}=[x,y,z,w]^T$.

Theorem 3
Let three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ be given in the projective space. Then the coefficients of the plane ρ, which is defined by those three points, are determined by the cross product of their homogeneous coordinates

$$\rho = \mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 \qquad (15)$$

where: $\rho = [a,b,c,d]^T$
and the cross product is defined as follows:

$$\mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{bmatrix} \qquad (16)$$

where: $\mathbf{i} = [1,0,0,0]^T$, $\mathbf{j} = [0,1,0,0]^T$, $\mathbf{k} = [0,0,1,0]^T$, $\mathbf{l} = [0,0,0,1]^T$
The proof is left to the reader, as it is similar to Proof 1.

Theorem 4
Let three planes ρ_1, ρ_2 and ρ_3 be given in the projective space. Then the homogeneous coordinates of the point $\mathbf{x}$ at the intersection of those three planes are given by the cross product of their coordinates

$$\mathbf{x} = \rho_1 \times \rho_2 \times \rho_3 \qquad (17)$$

i.e.:

$$\rho_1 \times \rho_2 \times \rho_3 = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{bmatrix} \qquad (18)$$

where: $\mathbf{i} = [1,0,0,0]^T$, $\mathbf{j} = [0,1,0,0]^T$, $\mathbf{k} = [0,0,1,0]^T$, $\mathbf{l} = [0,0,0,1]^T$
The proof is left to the reader, as it is similar to Proof 2.

These theorems are very important as they enable us to handle some problems defined in the homogeneous coordinates efficiently and make the computations more robust and effective.

3 Gaussian Elimination

The Gauss-Jordan elimination is a well-known method for solving systems of *n* linear equations:

$$\mathbf{Ax} = \mathbf{b} \qquad (19)$$

A solution exists if $\boldsymbol{A}$ is a regular matrix; furthermore in this paper we are interested specifically in the case of a *square regular matrix*. We can rearrange the equations and the right hand side vector into an augmented matrix $\boldsymbol{A}|\boldsymbol{b}$ (with coefficients $a_{i,j}$) to get our problem reformulated into this form:

$$\sum_{j=1}^{n} a_{i,j} x_j = a_{i,n+1} \qquad i = 1,2,\ldots,n \qquad (20)$$

The solution is then found by transforming the given matrix $\boldsymbol{A}$ into upper triangular matrix $\boldsymbol{A'}$, (with coefficients $a'_{i,j}$). The components x_i of the solution vector $\boldsymbol{x}$:

$$x_i = a'_{i,n+1} - \sum_{j=i+1}^{n} a'_{i,j} \qquad j = n, n-1,\ldots,1 \qquad (21)$$

One of the problems of this unmodified, standard approach is its numerical stability. This can be solved by pivoting, a method which selects rows with higher values for elimination of the others. This means that the coefficients are kept at lower order, which – because of floating point numbers storage pattern – helps to preserve the precision.

2.1. A projective extension. One of the very basic properties of the projective space is that the vectors are equivalent if they are multiplied *including the homogeneous coordinate* by any value except zero:

$$[a_1, a_2, \ldots, a_i : w]^T = [ka_1, ka_2, \ldots, ka_i : kw]^T \qquad (22)$$

If we expand matrix $\boldsymbol{A}|\boldsymbol{b}$ to $\boldsymbol{A}|\boldsymbol{b}|\boldsymbol{w}$, we get a new – projective – matrix, with the same properties for the solving of the system.
Since the Gaussian elimination is an iterative algorithm, we can run into problems with numerical overflow of vector components, if we want to eliminate the use of the division operation. Using homogeneous coordinates, we can easily solve this problem. Based on (22), the following formula is valid for vector with homogeneous coordinate:

$$\begin{aligned} &\left[r_1 2^{s_1}, \ldots, r_i 2^{s_i} : r_w 2^{s_w}\right]^T = \\ &\left[r_1 \frac{2^{s_1}}{2^{s_w}}, \ldots, r_i \frac{2^{s_i}}{2^{s_w}} : r_w \frac{2^{s_w}}{2^{s_w}}\right]^T = \\ &\left[r_1 2^{s_1 - s_w}, \ldots, r_i 2^{s_1 - s_w} : r_w\right]^T \end{aligned} \qquad (23)$$

We will call this operation the *exponent normalization* further in this paper.

4 Experiments

We have chosen a Hilbert matrix for the coefficients of our system. Its coefficients are given by the following formula:

$$a_{i,j} = \frac{1}{i+j-1} \qquad i, j \in \langle 1..n \rangle \qquad (24)$$

The determinant of the Hilbert matrix is very quickly falling towards zero as the dimension of the matrix grows, making it a perfect candidate for the precision testing.

The right hand vector is chosen as:

$$b_i = i - 1 \quad i \in \langle 1..n \rangle \tag{25}$$

Size of the problem was limited to n $n \in \langle 2..20 \rangle$ for the following comparisons, as the Hilbert matrix is becoming very ill-conditioned with growing n (the determinant value for $n = 5$ is of order 10^{-15}).

Two methods were implemented for the comparison purposes:

- pivoting Gaussian elimination
- non-pivoting projective Gaussian elimination

The first comparison of the methods is based upon the back substitution of the resulting vectors – $\boldsymbol{x}$ for the standard and $\boldsymbol{x}_p$ for the projective variant. The differences of the newly obtained right hand vectors $\boldsymbol{b'}$ ($\boldsymbol{b'}_p$ respectively) is then calculated and normalized:

$$\delta = \frac{1}{n}\sum_{i=1}^{n}\left|b'_i - b_i\right| \tag{26}$$

$$\delta_p = \frac{1}{n}\sum_{i=1}^{n}\left|b'_{p_i} - b_i\right| \tag{27}$$

The second comparison is based upon the exact solution of the given system. We can compute this using the matrix inverse $\boldsymbol{H}^{-1}$:

$$\mathbf{x}_e = \mathbf{H}^{-1}\mathbf{b} \tag{27}$$

since the formula for the coefficients is known [1]:

$$\mathbf{H}^{-1}{}_{(i,j)} = (-1)^{i+j}(i+j-1) \binom{n+i-1}{n-j}\binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2 \tag{28}$$

The numbers of this matrix are integers. This means – supposing we chose the integer right hand vector – that the $\boldsymbol{x}_e$ components will be integers as well.

As one can imagine, the multiplication of large numbers causes an imprecision problem, as the length of the mantissa is limited. Our second comparison is based upon this fact – the imprecision can be understood as a round-off error, and we will calculate it as sum of the differences from the vector obtained by the analytic solution:

$$\varepsilon = \sum_{i=1}^{n}\left|x_i - x_{ei}\right| \tag{29}$$

$$\varepsilon_p = \sum_{i=1}^{n}\left|x_{p_i} - x_{ei}\right| \tag{30}$$

However, even the exact solution values are affected by the round-off error for $n > 11$, therefore we introduce the expected result error:

$$\xi = \sum_{i=1}^{n}\left|x_{ei} - \left[x_{ei}\right]\right| \tag{31}$$

5 Results

The need for rapid development and verification of various algorithms' projective variants led us to implement a library for computation in the projective space – PLib [8]. It allows users to perform various arithmetical operations on projective vectors, while providing natural notation and therefore making the code more readable, and makes algorithms easier to implement. However this comes at a price of a performance penalty. Implementation of the algorithm for these tests was done using C# and the .NET platform.

The Table 1 shows that there are just minor differences in error rates of the methods and therefore they can be considered as equal in terms of precision from the first criterion standpoint.

The Fig 1 shows the data from Table 1 in a chart. There is an interesting change in trend for $n > 14$, but both algorithms are already far off the correct result – round-off errors are too large.

As for the second criterion, the results are summed up in the Table 2. Again, we can see very similar values for both error rates.

Also, the values of the expected error of the analytic solution ξ are significantly smaller compared to ε and ε_p error rates, therefore they should not be affected by the imprecise values of the analytical solution.

6 Conclusion

The experiment described in this paper proved that choosing a projective variant of the existing algorithm for solving the system of linear equations does not result in any loss of precision, while the division operation can be avoided completely. Although in the current state of the hardware support there is a performance penalty for the chosen approach (because of no support for the vector normalization), the future performance gains could be substantial.

The precision testing results nevertheless clearly proved that the precision of both the original pivoting Gauss-Jordan elimination and its projective non-pivoting variant is very similar, but the projective approach is clear and simple to implement.

Table 1 – *The normalized error of the right hand side vector (26),(27) for both Euclidean and projective variant.*

N	δ	δ_p
3	1,13687E-13	1,13687E-13
4	4,83169E-13	5,68434E-12
5	2,41016E-11	3,27418E-11
6	1,14596E-10	2,18279E-11
7	5,42059E-09	6,16274E-09
8	1,32131E-08	4,24334E-08
9	1,50176E-08	1,61817E-08
10	2,41213E-07	2,73809E-07
11	1,63913E-06	8,3819E-06
12	1,21891E-05	1,08033E-05
13	2,33352E-05	0,0001266
14	0,003219604	0,000446826
15	4,39584E-05	1,8537E-05
16	0,000948489	0,000533044
17	0,000926971	0,000339508
18	0,038101196	0,001551539
19	0,008105278	0,000868797
20	0,000365913	0,000473663

Table 2 - *The absolute solution errors and the analytic solution's expected error.*

N	ε	ε_p	ξ
3	9,9476E-13	9,9476E-13	0
4	4,82288E-10	3,42681E-10	0
5	1,13388E-08	8,73861E-08	0
6	5,48076E-05	0,000114997	0
7	0,014617973	0,033832397	0
8	0,718061665	2,935712333	0
9	982,6737948	625,4853351	0
10	222632,4052	260728,5907	0
11	40196386,46	55722836,66	0
12	5,16119E+12	5,16056E+12	0,517578125
13	4,44831E+13	4,43733E+13	0,501983881
14	3,64103E+14	3,5566E+14	0,127941132
15	2,96255E+15	2,96225E+15	0,2433815
16	2,27245E+16	2,27254E+16	10,28186035
17	1,79964E+17	1,79965E+17	166,4783936
18	1,34053E+18	1,34054E+18	1780,544922
19	1,0235E+19	1,0235E+19	9886,289063
20	7,44466E+19	7,44466E+19	110468,0039

7 Acknowledgement

The authors would like to thank colleagues and students at the University of West Bohemia for their comments and suggestions. This project was supported by the grants VIRTUAL No 2C06002, LC-CPG No LC06008 of the MŠMT Czech Republic.

References

[1] Bloomenthal,J., Rokne,J.: *Homogeneous Coordinates*, The Visual Computer, Vol.11, No.1, pp.15-26, 1994.

[2] Choi, M.D., *Tricks or Treats with the Hilbert Matrix*, American Mathematical Monthly 90, p301-312, 1983

[3] Coxeter,H.S.M.: *Introduction to Geometry*, John Wiley, 1969.

[4] Hartley,R, Zisserman,A.: *MultiView Geometry in Computer Vision*, Cambridge Univ. Press, 2000.

[5] Jimenez,J.J., Segura,R.J., Feito,F.R.: *Efficient Collision Detection between 2D Polygons*, Journal of WSCG, Vol.12, No.1-3, 2003

[6] Johnson,M: *Proof by Duality: or the Discovery of New Theorems*, Mathematics Today, 12,1996.

[7] Richardson, T.M.: *The Filbert Matrix*, 1999, http://arxiv.org/abs/math.LA/9905079/ 6/2007

[8] Skala, V., Kaiser, J., Ondracka, V.: *A library for computation in the projective space*, in 6th International Conference Proceedings of Aplimat 2007, STU Bratislava , 2007

[9] Skala, V.: *Length, Area and Volume Computation in Homogeneous Coordinates*, International Journal of Image and Graphics, Vol.6, No.4, p.625-639, 2006

[10] Skala, V.: *Barycentric Coordinates Computation in Homogeneous Coordinates*, in Computers & Graphics, Vol.32, No.1, p120-127, Elsevier, 2008

[11] Skala, V.: *GPU Computation in Projective Space Using Homogeneous Coordinates*, Game Programming Gems 6 (Ed.Dickheiser,M.), p137-147, Charles River Media, 2006

[12] Skala,V.: *Computation in Projective Space*, MAMECTIS conference, La Laguna, Spain, WSEAS, pp.152-157, 2009

[13] Skala,V.: *Duality and Intersection Computation in Projective Space with GPU support*, Applied Mathematics, Simulation and Modelling - ASM 2010, NAUN, Corfu, Greece, pp.66-71, 2010

[14] Skala,V:*Duality and Intersection Computation in Projective Space with GPU Support*, WSEAS Trans.on Mathematics, Vol.9.No.6.pp.407-416, 2010

[15] Stolfi,J: *Oriented Projective Geometry*, Academic Press, 2001.

[16] *PLib: Projective Library*, http://www.kiv.zcu.cz/vyzkum/software/index.html

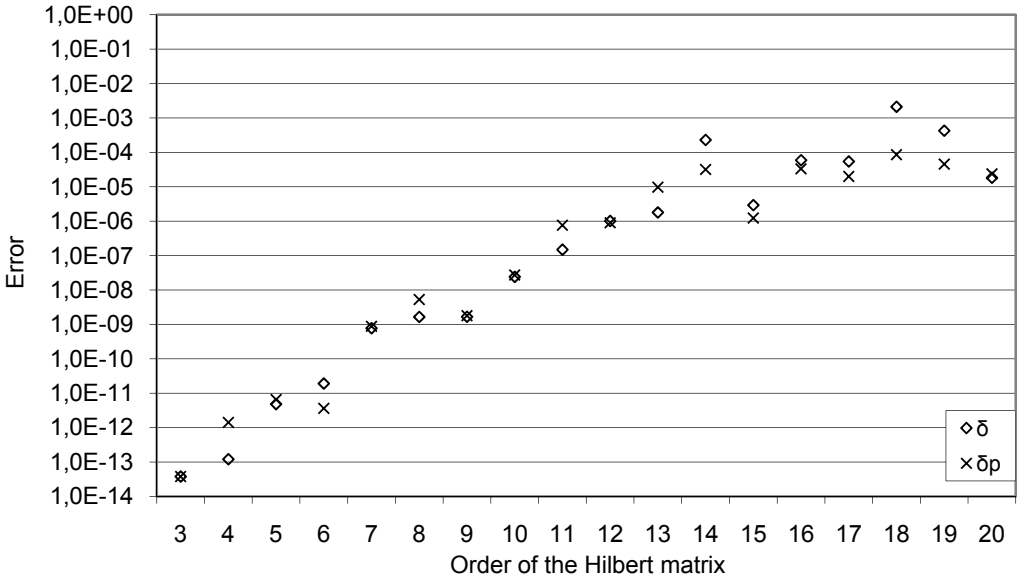


Fig 2. Error of the right hand side vector

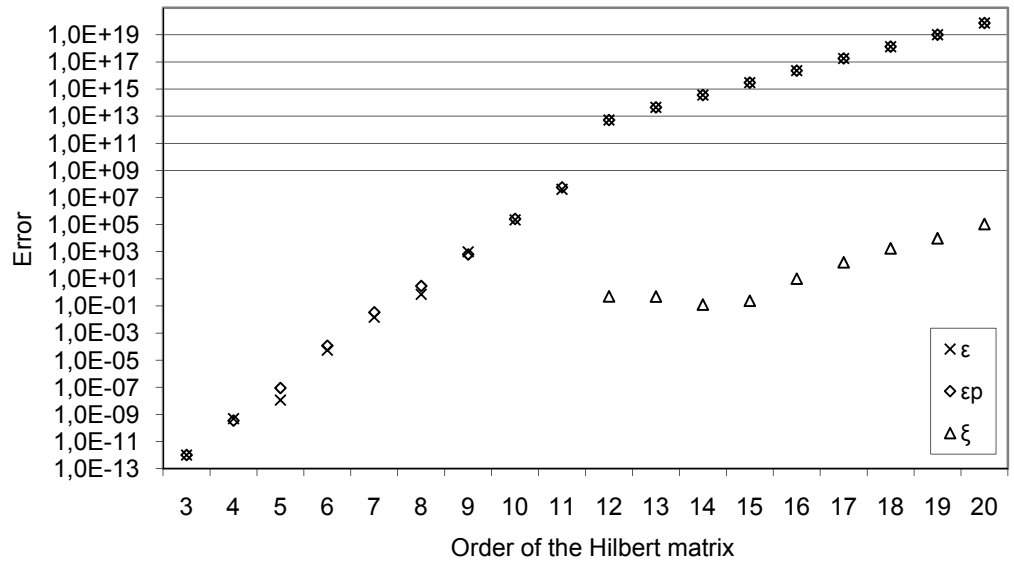


Fig 3. Absolute errors of the solution and expected analytical solution error

Appendix A

The cross product in 4D defined as

$$\mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 = \det \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix} \qquad \text{(A1)}$$

can be implemented in Cg/HLSL on GPU as follows:

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
{
    return ( dot(x1.yzw, cross(x2.yzw, x3.yzw)),
    -dot(x1.xzw, cross(x2.xzw, x3.xzw)),
    dot(x1.xyw, cross(x2.xyw, x3.xyw)),
    -dot(x1.xyz, cross(x2.xyz, x3.xyz)) );
}
```