Application of the pyramid method in difference solution d'Alembert equations on graphic processor with the use of Matlab

L.V. Yablokova[1], D.L. Golovashkin[1,2]

[1]*Samara National Research University, 34 Moskovskoe Shosse, 443086, Samara, Russia*
[2]*Image Processing Systems Institute – Branch of the Federal Scientific Research Centre "Crystallography and Photonics" of Russian Academy of Sciences, 151 Molodogvardeyskaya st., 443001, Samara, Russia*

Abstract

The paper proposes a modification of the pyramid method for constructing algorithms for the difference solution of the d'Alembert equation on a graphics processor in the event of a shortage of video memory. The authors demonstrate the effectiveness of the method on the practical example of dividing the grid area into two sub domains. Acceleration reaches the characteristic for the case of a domain entirely located in the video memory. In the article investigated the effectiveness of using the author's approach depending on the height of the pyramid and showed the boundaries of applicability of the proposed modification.

Keywords: The method of the pyramids; the grid area; difference solution; computing acceleration

---

1. Introduction

The deep interconnectedness of optics and computing technology is due to their mutual influence in the course of which at the turn of the 70-ies and 80-ies of the last century there were two independent branches of science: computer optics associated with the development of numerical methods of calculation and simulation of optical devices on a computer and optical engineering, in which the optical elements are created computing devices. The growth of the relevance of the mentioned industries due to the perfection of the architecture of computers (multithreading, multicore, vectored calculations) and technologies of formation of optical elements (transition from micro to nano-size). The first feature allowed us to use the methods of a rigorous diffraction theory [1] for calculating the nano-sized elements of optical processors, characterized by high computational complexity.

Among the numerical methods of the strict theory of diffraction, FDTD [1], deserving high universality (Maxwell's equations describe all wave electromagnetic processes) and the simplicity of understanding (based on the replacement of derivatives by difference relations) deserved wide popularity. The latter circumstance makes it possible to write the computational procedures of the method in a clear form in the popular language of matrix calculations of MATLAB [2]

Unfortunately, the software implementation of the FDTD method on modern graphics computing devices that provide faster CPU computation by an order of magnitude is encountered, when using this language, with high demands on the amount of video memory: in the production of calculations, it is necessary to use several times more volume than when Work on the central processor. This circumstance is aggravated traditionally by small video memory sizes (up to 2GB in modern budget video cards) in comparison with operating memory (not less than 4GB, even for office computers).

The authors of this publication see the application of the pyramid method as an example of the organization of calculations using the difference scheme Yee [1] on the GPU using CUDA C [3].

2. Difference solution of the d'Alembert equation (one-dimensional case) on a graphics processor

Traditionally, the FDTD method is understood to mean exclusively the difference solution of Maxwell's equations, which is not entirely correct. In the early 80's of the last century [1], the difference solution of the d'Alembert equation was also applied to it, which is still being done [1, 4]. We note that when solving the wave equation the problem of video memory shortage is more acute than for Maxwell's equations because of the necessity of finite-difference approximation of second, not first-order derivatives. However, the decision of the wave equation on the GPU seems more promising due to the high efficiency of vectorization of computational procedures [5].

Outlining the concept of the work, the authors decided to dwell on the one-dimensional equation of d'Alembert, seeking to demonstrate the possibilities of the pyramid method on a simple example.

So known [1] the difference scheme for this equation

$$\frac{E_i^{k+1}-2E_i^k+E_i^{k-1}}{h_t^2}=c^2\,\frac{E_{i-1}^k-2E_i^k+E_{i+1}^k}{h_z^2} \tag{1}$$

is written with respect to the grid function defined on the domain $D^h=\{(t_k,z_i):t_k=kh_t,k=0,1,\ldots,N_t=T/h_t,z_i=ih_t,i=1,\ldots,N_z=L_z/h_z+1\}$, where E the value of the electric field strength is, c is the speed of light in free space, T and L_z are size of the region in time and space.

Below is a fragment of the computational procedure for solving (1) in MATLAB, where $c_1 = c^2 h_t^2 / h_z^2$, $c_5 = 2\pi c\, h_t / \lambda$.

```
% Placement of grid functions on two time layers in video memory
  E1=zeros(1,Nz,'gpuArray'); E2=zeros(1,Nz,'gpuArray');
    for k=1:2:Nt % Passage through time layers of the grid area through one
    E1(2:Nz-1)=2*E2(2:Nz-1)-E1(2:Nz-1)+c1*diff(E2,2); % Calculations on the layer k
      E1(2)=sin(c5*k);       % Hard radiation condition on the layer к
      E2(2:Nz-1)=2*E1(2:Nz-1)-E2(2:Nz-1)+c1*diff(E1,2); % Calculations on the layer k+1
      E2(2)=sin(c5*(k+1));  % Hard radiation condition on the layer k+1
    End
  E=gather(E2); % Transfer of results to RAM
```

For $N_z = 5\times10^7$ and $N_t = 100$ the duration of calculations on the Intel Core i7 CPU was 57.08 s., On the GeForce GTX TITAN X GPU - 5.55 s. (acceleration of 10.29 times) using MATLAB 2015b and the operating system CentOS 7.2. Both used arrays occupied 762 MB in memory, however, during the computations on the CPU, the memory requirements increased by one and a half time, on the GPU the memory requirements increased threefold. Apparently, with the implementation of calculations for the design E1(2:Nz-1)=2*E2(2:Nz-1)-E1(2:Nz-1)+c1*diff(E2,2) on the CPU, the execution of the operation of numerical differentiation diff(E2,2) resulted in allocating additional memory for two copies of the array E2, and the execution on the GPU of the design as a completely required separate area of memory for all the arrays involved and for double copying E2. MATLAB takes about 0.4 gigabytes in RAM, but does not use video memory for its placement. Thus, the execution of the whole algorithm on the CPU was accompanying by the extraction of 1.52 GB. In addition, the execution of the whole algorithm on the GPU was accompanying by the extraction of 2.24 GB. Moreover, if the researcher has a video card with 2 GB of memory (like most popular video processors now) then the organization of calculations on the GPU becomes impossible. In his previous publication [7], using the difference scheme for the Maxwell equations, the CUDA C software tool the authors proposed to solve this problem using the method of pyramids.

3. The pyramid method application

Will this be possible in this case, given that MATLAB is not specialized for working with graphics processors and its tools in this area are very meager?

The essence of the mentioned method in the author's modification consists in splitting the grid domain into overlapping sub regions that fit in the video memory completely, with the subsequent organization of communications in the production of vector computations in each sub region separately. In this case, transfers from RAM to video and vice versa are performed not on each time layer, but through a certain number of them h (the height of the pyramid). This, on the one hand, leads to a reduction in h the number of communications. On the other hand, to the duplication of arithmetic operations in overlapping fragments of grid subdomains (the form of pyramids is available in the two-dimensional case).

A fragment of the computational procedure implementing this strategy in the case under consideration is presented below, where $N = {}^{N_z}\!/_2$.

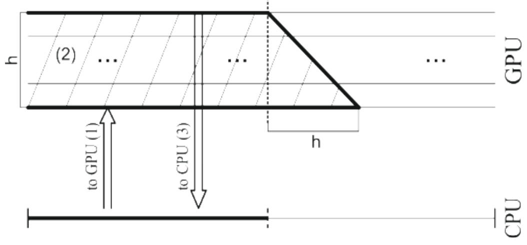

Fig. 1. The scheme of the algorithm of the pyramids to work with the first domain; (1) there is a message to GPU, (2) there is a calculate to GPU, . (1) there is a message to CPU.

```
% creating temporary layers on CPU and GPU
  E1=zeros(1,Nz); E2=zeros(1,Nz); E1m=zeros(1,h); E2m=zeros(1,h);
  E1g=zeros(1,N+h,'gpuArray'); E2=zeros(1,N+h,'gpuArray');
    for t=1:h:Nt % Passage through the pyramids
   % work with the left subdomain
      E1g=gpuArray(E1(1:N+h)); E2g=gpuArray(E2(1:N+h)); % Forwarding CPU ==> GPU
        for k=1:2:h    % Calculations inside the first pyramids
          E1g(2:N+h-k)=2*E2g(2:N+h-k)-E1g(2:N+h-k)+c1*diff(E2g(1:N+h-k+1),2);
          E1g(2)=sin(c5*(t+k-1));
      E2g(2:N+h-k-1)=2*E1g(2:N+h-k-1)-E2g(2:N+h-k-1)+c1*diff(E1g(1:N+h-k),2);
      E2g(2)=sin(c5*(t+k));
    end
  E1(2:N-h)=gather(E1g(2:N-h)); E2(2:N-h)=gather(E2g(2:N-h)); % Forwarding GPU ==> CPU
  E1m(1:h)=gather(E1g(N-h+1:N)); E2m(1:h)=gather(E2g(N-h+1:N));
% work with the right subdomain
  E1g(1:N+h-1)=gpuArray(E1(N-h+1:Nz)); E2g(1:N+h-1)=gpuArray(E2(N-h+1:Nz));
    for t=1:2:h % Calculation by layers of the pyramid
      E1g(t+1:N+h-2)=2*E2g(t+1:N+h-2)-E1g(t+1:N+h-2)+c1*diff(E2g(t:N+h-1),2);
      E2g(t+2:N+h-2)=2*E1g(t+2:N+h-2)-E2g(t+2:N+h-2)+c1*diff(E1g(t+1:N+h-1),2);
    end
  E1(N+1:Nz-1)=gather(E1g(h+1:N+h-2)); % Forwarding GPU ==> CPU
  E2(N+1:Nz-1)=gather(E2g(h+1:N+h-2));
  E1(N-h+1:N)=E1m(1:h); E2(N-h+1:N)=E2m(1:h); % Replenishment of the result
 end
```

In the course of experiments with the new algorithm, the dependence of the calculation time on the height of the pyramid. The Table 1 contains the results.

Table 1. The dependence of the calculation duration of the calculation of the height of the pyramid.

Height of the pyramid, h	Computation time (*s*)	Acceleration
2	53.02	1.08
4	29.58	1.93
10	15.49	3.7
20	10.75	5.31
50	7.9	7.23

4. Conclusion

Thus, the method of pyramids can be effectively using in arranging calculations for solving difference equations with the help of MATLAB on graphic processors in the case when arrays storing values of grid functions do not fit into video memory as a whole. The development of the proposed algorithm for cases of large dimensions will be the next stage of the authors' research in this direction.

Acknowledgements

The research leading to these results has received funding from the Russian Science Foundation grant №16-47-630560-r_a.

References

[1] Taflove A, Hagness S. Computational Electrodynamics: The Finite-Difference Time-Domain Method. Boston: Aerotech House Publishers, 2005; 1006 p.
[2] Elsherbeni A, Demir V. The Finite-Difference Time-Domain Method for Electromagnetics with MATLAB Simulations. Scitech Publishing Inc. 2009; 426 p.
[3] Grigorjev IS, Mejlihov EZ. Physical Values: Reference Book. Moscow: Energoatomizdat Publisher, 1991; 1232 p. (in Russian).
[4] Malysheva SA, Golovashkin DL. Realization of the difference solution of the Maxwell equations on graphic processors by the pyramid method. Computer Optics 2016; 40(2): 179–187. DOI: 10.18287/2412-6179-2016-40-2- 179-187.
[5] Kozlova ES, Kotlyar VV. Simulation of the propagation of a short two-dimensional pulse of light. Computer Optics 2012; 36(2): 158–164.
[6] Vorotnikova DG, Golovashkin DL. Difference solution of the wave equation on graphical processors with repeated use of pairwise sums of the differential pattern. Computer Optics 2017; 41(1): 134–138. DOI: 10.18287/2412-6179-2017-41-1-134-138.
[7] Vorotnikova DG, Golovashkin DL. Algorithms with "long" vectors for solving grid equations of explicit difference schemes. Computer Optics 2015; 39(1): 87–93. DOI: 10.18287/0134-2452-2015-39-1-87-93.