IMPLEMENTATION OF DIFFERENCE SOLUTIONS OF MAXWELL'S EQUATIONS ON THE GPU BY METHOD OF PYRAMID

L.V. Yablokova, D.L. Golovashkin

Samara National Research University, Samara, Russia

Abstract. The work is dedicated to the development of the method of pyramids in the annex to the decision of Maxwell's equations in the time domain by means of the finite difference (FDTD) and its implementation on the GPU. This method allows you to remove the restriction on the amount of memory the graphics computing device.

Keywords: FDTD, Maxwell's equations, method of pyramids, GPU, CUDA.

Citation: Yablokova LV, Golovashkin DL. Implementation of difference solutions of Maxwell's equations on the GPU by method of pyramid. CEUR Workshop Proceedings, 2016; 1638: 469-476. DOI: 10.18287/1613-0073-2016-1638-469-476

Introduction

Difference solution of Maxwell's equations method in the time domain (FDTD) [1] is widely used in modern research in the simulation: the propagation of radiation [2], materials with new properties [3], the interaction of radiation with biological objects [4] and in many other applications.

Its popularity is due to methodological clarity (based on the replacement of derivatives by difference quotients), wide scope of application (rigorous theory of diffraction) and an abundance of ready-made software implementations (many commercial and open source packages for different architectures and operating systems). The price for the universality of the method - high demands on processor speed and amount of RAM. Therefore developed half a century ago, the method has been actively used in computational practice until now.

The development of computer technology hardware base faced with difficulties ("silicon dead end"). It is not by increasing clock speed of processors. Produces special architectural solutions. They allow to increase the speed of certain types of operations. The most notable successes achieved in this direction during the development of the architecture of graphics processors (GPU) and related software tools (CUDA [5], the OpenCL [6]), allowing multiple accelerate action with vectors and matrices.

From a mathematical point of view, based on the FDTD method is an operation of subtraction of matrices. Modern implementations FDTD-method on GPU [7, 8] characterized by an increase in productivity of 10 times compared to the central computing processor. However, a limited amount of video memory (2-6 Gb. compared with 4-32 Gb. RAM) does not allow full use of the advantage of GPU performance.

Overcoming this limitation - an urgent task. To solve this problem in this paper, we propose to apply the pyramid method [9,10,11]. It is based on reducing the intensity of communication between RAM and VRAM due to duplication of arithmetic operations.

A similar problem applied to other differential equations of the previously successfully solved in the works [12,13,14].

Yee algorithm

Illustrating the application of the method of the pyramids to the times-difference solution of Maxwell's equations, the authors settled on the classical Yee scheme for the three-dimensional case [15], which is currently the most popular tool for computational electrodynamics [1]. The main feature of this scheme is to separate the location of the nodes of the mesh region for each projection of the electric and magnetic fields, providing increased order of approximation of the initial differential problem by time and space.

Table 1. The coefficients are grid to projections of the electric and magnetic fields

A	d	M_t	a	I_x	b	J_y	c	K_z
E_x			0,5	1				
E_y					0,5	1		
E_z							0,5	1
H_x	0,5	1			0,5	1	0,5	1
H_y	0,5	1	0,5	1			0,5	1
H_z	0,5	1	0,5	1	0,5	1		

We considered three-dimensional space of the area of computational experiment D^3 ($0 < t \le T$, $0 \le x \le L_x$, $0 \le y \le L_y$, $0 \le z \le L_z$) the grid structure is superimposed on the area D_h^3 . In the nodes $\{ (t_{m+d}, x_{i+a}, y_{j+b}, z_{k+c}) : t_{m+d} = (m+d)h_t$, $m = 0,1,\ldots,$ $M - M_t$, $(M = T / h_t)$, $x_{i+a} = (i+a)h_x$, $i = 0,1,\ldots, I - I_x$, $\left(I = L_x / h_x\right)$, $y_{j+b} = (j+b)h_y$, $j = 0,1,\ldots, J - J_y$, ($J = L_y / h_y$), $z_{k+c} = (k+c)h_z$,

$k = 0,1,..,K - K_z$, $K = L_z / h_z$ } of grid defined projection of the field $A^{m+d}_{i+a,j+b,k+c}$. The values of coefficients for different grid projections of electric and magnetic fields are presented in table 1. The empty cells correspond to zero value of the coefficient. The difference scheme for this region is represented by the equations:

$$H^{m+0.5}_{x_{i,j+0.5,k+0.5}} = D_{a_{i,j+0.5,k+0.5}} H^{m-0.5}_{x_{i,j+0.5,k+0.5}} - D_{b_{i,j+0.5,k+0.5}} \left(\frac{E^m_{z_{i,j+1,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{h_y} - \frac{E^m_{y_{i,j+0.5,k+1}} - E^m_{y_{i,j+0.5,k}}}{h_z} \right)$$

$$H^{m+0.5}_{y_{i+0.5,j,k+0.5}} = D_{a_{i+0.5,j,k+0.5}} H^{m-0.5}_{y_{i+0.5,j,k+0.5}} - D_{b_{i+0.5,j,k+0.5}} \left(\frac{E^m_{x_{i+0.5,j,k+1}} - E^m_{x_{i+0.5,j,k}}}{h_z} - \frac{E^m_{z_{i+1,j,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{h_x} \right)$$

$$H^{m+0.5}_{z_{i+0.5,j+0.5,k}} = D_{a_{i+0.5,j+0.5,k}} H^{m-0.5}_{z_{i+0.5,j+0.5,k}} - D_{b_{i+0.5,j+0.5,k}} \left(\frac{E^m_{y_{i+1,j+0.5,k}} - E^m_{y_{i,j+0.5,k}}}{h_x} - \frac{E^m_{x_{i+0.5,j+1,k}} - E^m_{x_{i+0.5,j,k}}}{h_y} \right)$$

$$E^{m+1}_{x_{i+0.5,j,k}} = C_{a_{i+0.5,j,k}} E^m_{x_{i+0.5,j,k}} + C_{b_{i+0.5,j,k}} \left(\frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i+0.5,j-0.5,k}}}{h_y} - \frac{H^{m+0.5}_{y_{i+0.5,j,k+0.5}} - H^{m+0.5}_{y_{i+0.5,j,k-0.5}}}{h_z} \right)$$

$$E^{m+1}_{y_{i,j+0.5,k}} = C_{a_{i,j+0.5,k}} E^m_{y_{i,j+0.5,k}} + C_{b_{i,j+0.5,k}} \left(\frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j+0.5,k-0.5}}}{h_z} - \frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i-0.5,j+0.5,k}}}{h_x} \right)$$

$$E^{m+1}_{z_{i,j,k+0.5}} = C_{a_{i,j,k+0.5}} E^m_{z_{i,j,k+0.5}} + C_{b_{i,j,k+0.5}} \left(\frac{H^{m+0.5}_{y_{i+0.5,j,k+0.5}} - H^{m+0.5}_{y_{i-0.5,j,k+0.5}}}{h_x} - \frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j-0.5,k+0.5}}}{h_y} \right)$$

Where, $C_{a_{i,j,k}} = \dfrac{1 - \dfrac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}}{1 + \dfrac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}}$, $C_{b_{i,j,k}} = \dfrac{\dfrac{h_t}{\varepsilon_{i,j,k}\varepsilon_0}}{1 + \dfrac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}}$, $D_{a_{i,j,k}} = \dfrac{1 - \dfrac{\sigma^*_{i,j,k}\, h_t}{2\mu_{i,j,k}\mu_0}}{1 + \dfrac{\sigma^*_{i,j,k}\, h_t}{2\mu_{i,j,k}\mu_0}}$,

$D_{b_{i,j,k}} = \dfrac{\dfrac{h_t}{\mu_{i,j,k}\mu_0}}{1 + \dfrac{\sigma^*_{i,j,k}\, h_t}{2\mu_{i,j,k}\mu_0}}$. ε_0 , μ_0 it electric and magnetic constants, $\varepsilon_{i,j,k}$, $\mu_{i,j,k}$ it grid of the relative electrical and of the magnetic permittivity of linear isotropic environment without dispersion, $\sigma_{i,j,k}$, $\sigma^*_{i,j,k}$ it grid-specific of the electric and of the magnetic conductivity.

Software implementation of FDTD-method

Consider the propagation of a plane wave in the three-dimensional region of space, bounded absorbing layers CPML [1]. We consider the Maxwell equations and difference scheme Yee for them [1]. The solution of difference equations is a means of modeling.

Define the grid area the size of $256 \times 256 \times 256$ samples in space. The width of the absorbing layer select equals 11 counts. It should be 467 Mb to accommodate such a task in the memory card.

We apply the method to a one-dimensional pyramid decomposition [12] and carry out the partition of the original grid area of one chosen direction. The following is the author's code to CUDA C to Yee algorithm in [1]:

```
__host__ void raschetGPU_pyramid_1d()
{
 ...
  // The number of launched blocks
  int SIZEx = ((Imax-1)%BLOCK_DIMx==0) ?
  (Imax-1)/BLOCK_DIMx : (Imax-1)/BLOCK_DIMx+1;
  int SIZEy = ((Jmax-1)%BLOCK_DIMy==0) ?
  (Jmax-1)/BLOCK_DIMy : (Jmax-1)/BLOCK_DIMy+1;
  dim3 gridSize = dim3(SIZEx,SIZEy, 1);
  dim3 blockSize = dim3(BLOCK_DIMx,BLOCK_DIMy,
  BLOCK_DIMz);
  create_temp_fields();
  // Copying of overlapping regions into a temporary
  // buffer
  int countPyramidsK = ceil((Kmax-1) /
  (pyramidBaseLengthK + 0.0f));
  // The number of pyramids
  int currentTime = 1;// The current step of time
  int timeOfPass =((nmax)%PASS==0)?
  nmax/PASS : nmax/PASS+1;
  // The number of passes
  // The loop of passes
  for (int pass = 1; pass <= PASS; pass++)
  {
   int currentBaseLengthK; // the width of the upper base
   //of the current pyramid
   int leftOffsetK; // the width of the left overlap of
   // the bottom base of the pyramid
   int rightOffsetK;  // the width of the right overlap of
   //the bottom base of the pyramid
   int fullBaseLengthK;// the width of the bottom base of
   //the current pyramid
   int leftPyramidBorderK; // the index of the left
   // border of the base of the pyramid
   int rightPyramidBorderK;// the index of the right
   // border of the base of the pyramid
```

```
   int durationOfTimePass = timeOfPass * pass; // the
   // time step corresponding to the end of the current
   // passage

   // Copying into a temporary buffer initial field
   //values to pass
   copyFields(...);

   // The loop for pyramid
   for(int pyramidIdK = 0; pyramidIdK < countPyramidsK;
   pyramidIdK++)
   {
    int startPyramidBasePositionK = pyramidIdK *
    pyramid BaseLengthK; // The starting index of the
    //upper base of the pyramid in the grid
    getOffsets(pyramidIdK, pyramidBaseLengthK,
    &currentBaseLengthK, &leftOffsetK, &rightOffsetK);
    getBorders(currentBaseLengthK, leftOffsetK,
    rightOffsetK, &fullBaseLengthK, &leftPyramidBorderK,
    &rightPyramidBorderK);

    // Copying fields on a graphics card
    create_arrays_on_GPU(Imax,Jmax,fullBaseLengthK);
    copy_temp_arrays_to_GPU(0,0,leftPyramidBorderK,
    Imax,Jmax,fullBaseLengthK);
    copy_constant_to_device();
    // The loop of time inside passage
    for (int n = currentTime; n <= durationOfTimePass
    && n <=nmax; n++)
    {
     // Kernel for conversion of magnetic field
     //components on the GPU
     kernelH_pyramid1<<<gridSize,blockSize>>>(...);
     cudaEventSynchronize(syncEvent);
     // Kernel for conversion of electric field
     //components on the GPU
     kenelE_pyramid1<<<gridSize,blockSize>>>(...);
     cudaEventSynchronize(syncEvent);
    }

   // Copying data from video memory
   copy_arrays_from_GPU_with_offset(currentBaseLengthK,
   tartPyramidBasePositionK, leftOffsetK);
   delete_arrays_on_GPU();
   }
   currentTime = durationOfTimePass +1;
  }
 ...
}
```

Grid subdomain at the lower base of the pyramid (containing the initial data for each pass) intersect. Therefore, these areas are in a temporary buffer (implemented procedure create_temp_fields) must be copied to the beginning of the next pass. Then the values necessary for the processing of the adjacent pyramid will not be lost.

Computational experiments

Numerical calculations to determine the duration of the experiments were carried out on the video card NVIDIA GeForce GTX 660 Ti, and the CPU Intel Core 2 Duo E8500. The study was carried out in the Ubuntu operating system. Exploring the conditions of efficiency of the method of the pyramids will conduct a series of computational experiments. Limit the amount of video memory of 245 MB, which will build up to 3 pyramids with different base width and height. The purpose of the experiments - determination dependence on the acceleration of calculations of these parameters.

Fig. 1 shows the results of measuring the duration of computation.

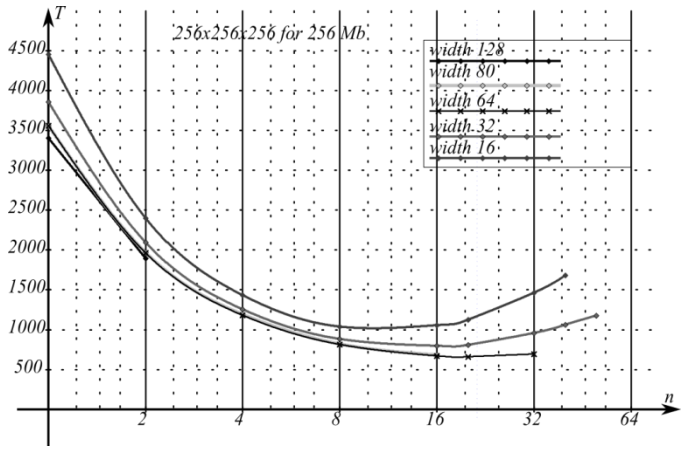

Fig. 1. The duration of the calculation y (ms) as a function of height x (number of samples) of the pyramid for varying the width of the base

Minimum of time calculation is obtained by achieving a balance between reducing the number of communications and increasing the number of arithmetic operations. This is obtained (Fig. 1) using a pyramid base width 64 and height 20 in the total number of time layers equal 256. With further increase of the height of the pyramid calculation time begins to increase due to increased volume of duplicate data. This is illustrated by the U-shaped dependence of the computation time of the height of the pyramid.

Conclusion

Implementation of FDTD method with the use of the pyramids allows to overcome the limitation on the amount of memory of the GPU. This allows you to use the GPU advantage in speed. Pyramid method is based on reducing the intensity of communication between main and video memory by duplicating arithmetic operations.

In applying the method of the pyramids is important to strike a balance between the volume of communication and memory volume. To do this, you must find the optimum width and height of the base of the pyramid.

Acknowledgment

This work was supported by grant RFBR 14-07-00291-a.

References

1. Taflove A, Hagness S. Computational Electrodynamics: The Finite-Difference Time-Domain Method. Boston: Arthech House Publishers (Thrid Edition), 2005; 1006 p.
2. Kotlyar VV, Stafeev SS, Feldman AYu. Photonic nanojets formed by square microsteps. Computer Optics, 2014; 38(1): 72-80 [in Russia].
3. Tiranov AD, Kalachev AA. Collective spontaneous emission in a waveguide with close to zero refractive index. Bulletin of the Russian Academy of Sciences, 2014; 78(3): 271-275 [In Russian].
4. Perov SYu, Bogacheva EV. Theoretical and experimental dosimetry in the evaluation of biological effects of electromagnetic fields of portable radio stations. Message 1. Flat phantoms. Radiation Biology. Radioecology, 2014; 54(1): 57-61.
5. Boreskov AV, Harlamov AA. The basics of working with CUDA technology. Moscow: “DMK Press” Publisher, 2010; 232 p. [in Russian].
6. OpenCL – The open standard for parallel programming of heterogeneous systems. URL: http://www.khronos.org/opencl/.
7. B-CALM - Belgium California Light Machine. URL: http://b-calm.sourceforge.net/
8. FDTD solver. URL: http://www.acceleware.com/fdtd-solvers.
9. Lamport L. The parallel execution of DO loops. Communications of the ACM, 1974; 17(2): 83-93.
10. Valkovski VA. Parallel execution cycles. The method of the pyramids. Cybernetics, 1983; 5: 51-55.
11. Golovoshkin DL. The solution of grid equations on graphics processing devices. The method of the pyramids. Modern problems of applied mathematics and mechanics: theory, experiment and practice. International conference dedicated to the 90th anniversary from the birthday of academician NN. Yanenko, Novosibirsk, Russia, 30 May – 4 June 2011 - Novosibirsk, ICT SB RAS, 2011, N 0321101160, http://conf.nsc.ru/files/conferences /niknik-90/fulltext/37858/ 46076/kochurov_final.pdf.

12. Kochurov AV, Golovashkin DL. GPU implementation of Jacobi Method and Gauss-Seidel Method for Data Arrays that Exceed GPU-dedicated Memory Size. Journal of Mathematical Modelling and Algorithms in Operations Research, 2015; 14(4): 393-405.
13. Kochurov AV, Vorotnikova DG, Golovashkin DL. GPU implementation of Jacobi method for data arrays that exceed GPU-dedicated memory size. CEUR Workshop Proceedings, 2015; 1490: 414-419. DOI: 10.18287/1613-0073-2015-1490-414-419.
14. Golovashkin DL, Kochurov AV. Pyramid method for GPU-aided finite difference method. Proceedings of the 13th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2013 24–27 June, 2013; 746-756.
15. Yee KS. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Trans. Antennas Propag., 1966; 14: 302-307.