U3D based computer graphics in PDE PTK

Ilyin V.P., Solovev S.A.
Intel Novosibirsk Laboratory,
Novosibirs, Russia
{Valery.p.ilyin , Sergey.a.solovev}@intel.com

Abstract

The structure and functionality of postprocessing in PDE Performance Toolkit for 3D physical modeling is described. In complicated computational domains the computing results are presented by node and edge values at the nonuniform grids. Algorithms for calculating isolines and isosurfaces, vector fields and gradients of scalar fields are described. The obtained graphics information is transformed into U3D format and is drawn by means of corresponding tools. Examples of visualization pictures are presented.

***Keywords**: 3D physical modeling, computational domain, isoline, isosurface, field gradient, vector field, U3D format.*

1. INTRODUCTION

Partial Differential Equation Performance Toolkit (PDE PTK) is the set of numerical methods, technologies and codes for scientific computation and physical modeling in various industrial applications. The predecessor of this code was the Applied Program Package (APP) BASIS, see [1], for complex simulation of technological processes in an aluminum electrolizer. It includes routines for three-dimensional mathematical modeling of electrical fields, heat transfer and diffusion-convection problems, thermo-elasticity stress analyses and solving Navier-Stokes system of differential non-linear equations which describe the flow of a viscous incompressible fluid.

A computational domain can have complicated topology and consist of any number of subdomains with different material properties, i.e. coefficients of PDEs are piece-wise smooth in general, constant or variable in each sub-domain, and have jumps at the interface between subdomains. On the different segments of external boundary various types of boundary conditions can be stated. In principle, different systems of PDEs can be solved in different subdomains.

Discretization of the considered boundary value problems (BVPs) is made by adapted regular structured piece-wise uniform grids which provide a simple grid data structure (GridDS), with full necessary information about grid objects: nodes, edges, faces and finite elements (volumes). Approximation of BVPs is implemented by mixed finite volume methods (FVMs, [2]) which result in an algebraic data structure (ADS) for systems of linear algebraic systems (SLAEs) with sparse matrices of very large order, symmetric or non-symmetric. Numerical solution of SLAEs is realized by a preconditioned iterative algorithm, on the basis of incomplete factorization and generalized conjugate direction methods, [3].

PDE PTK includes also preprocessing which provides control and modification of user input data and forms geometrical and functional data structures (GeomDS & FDS) for computational modules of the Toolkit.

An important component of PDE PTK is a problem-oriented post-processor which is responsible for computer graphics in the Toolkit. It provides an implementation of numerical algorithms for computing the point coordinates for isolines and isosurfaces, function graphs and solution vectors as well as formation of geometrical data in accordance with U3D specifications [4]. This standard defines the syntax and semantics of the U3D file format, an extensible format for downstream 3D CAD repurposing and visualization, useful for many mainstream business applications.

The conception and niche of PDE PTK are the following. In the world market of applied software, there are many powerful commercial APPs similar to ANSYS [5], which have strong computational tools for multi-physics modeling and extended graphic user interface (GUI) communicated with CAD/CAE systems. Usually, such APPs have been implemented and developed during several decades by many people and are largely fixed in terms of their data structure as well as adaptation to new tasks and computational technologies. In this sense, PDE PTK is a light APP which provides fast solving of a wide class of 3D mathematical problems, by advanced computational approaches, and can be used also for industrial applications. The limitations of PDE PTK consist in the absence of curvilinear boundary surfaces and non-structured grids. But it helps developers and users to apply simple enough and flexible data structures which provide high efficiency of algorithms and software product.

This paper is organized as the following. In section 2, we present the general structure of PDE PTK , computer algorithms and technologies for simulation of various physical processes as well as the current issues for postprocessing and visualization of mathematical modeling results. Section 3 presents numerical methods for definition of the main lines and surfaces, on the basis of obtained grid solutions, and principles of interface with existing U3D graphic tools. In conclusion, we demonstrate several pictures which present the functionality of PDE PTK postprocessing.

2. THE ARCHITECTURE AND DATA STRUCTURE OF PDE PTK

The architecture of PDE-PTK is presented by the set of code units (tools) unified by the consistent flexible data structures which realize input, internal, and output program interfaces.

PTK consists of the following main components:

- Preprocessor: identifying, controlling, and preprocessing initial information, i.e., geometrical and functional data structures. These two DSs define completely the topology and geometry of a computational domain and its subdomains, type of PDEs to be solved as well as the problem coefficients, boundary conditions on the different parts of boundaries, and some necessary data for the algorithms. The content of FDS, in general, is different for various types of boundary value problems (BVPs) and it is strongly connected with the GeomDS objects; the last ones of which are independent of the FDS objects.

- Discretizator (Mesh Generator): construction of an adaptive non-uniform grid and generation of grid data structure including the full information about mesh objects and necessary references to the geometrical and functional objects; grid generation stage and content of GridDs do not depend, in some sense, on the type of the considered BVPs and PDEs.

- Approximator: implementation of the algorithms that approximate any given boundary value problem for PDEs, and construction of the algebraic data structure (ADS) resulted in the system of linear or nonlinear algebraic equation (SLAE or SNAE) in some conventional sparse matrix format. This computational stage is realized by a separate library, i.e., the set of independent procedures oriented to the specific type of differential equations and boundary conditions.

- Algebraic Solver: realization of iterative algorithms defined by used preconditioner, acceleration approach ((bi)conjugate gradient or (bi)conjugate residual, (Bi)CG or (Bi)CR, for example), stopping criteria, matrix format, and some optimizing iterative parameters;. Each solver presents independent procedure implemented for a particular kind of the method and program interface.

- Postprocessing: analysis of the resulting grid solution, computing of the coordinates of isolines and vector fields in the given cross-sections as well as preparing of the information for isosurfaces, 3D pictures of solutions, tables, histograms, graphs, and other resulting data structures (ResDS) for visualization.

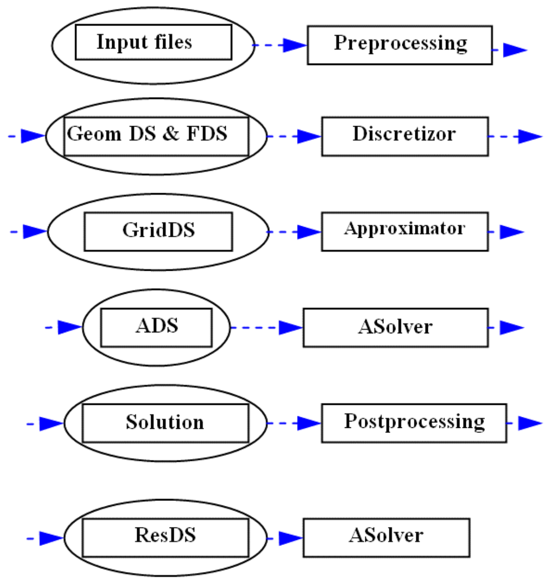

Figure 1: Block-chart of Physical Modeling.

3. The considered tools realize modern computational and informational technologies of mathematical modeling for complicated physical problems ([6], [7]).

In general, a block-chart of computational processes for physical modeling can be presented in the following way, see Fig 1:

3. POSTPROCESSING

Postprocessing provides visualization of the results which are calculated using PDE-PTK. Users can form isolines of the scalar fields in cross-sections, 3D isosurfaces in full computational domain, vector gradients of scalar fields and vector fields for data on different grid structures. Simultaneously with modeling results, users can draw the computational domain. Postprocessing's results are transformed into U3D format and then are visualized in a Graphic User Interface.

In multi-physical modeling, the typical objects for visualization are the following: equipotential lines in cross-section and space equipotential surfaces in electric field simulation, isotherms in calculated temperature distributions, vector stress field in elasticity, and vector velocity field for fluid flow. In considered FVM, the values of scalar fields (potentials, temperature, density) are defined in grid nodes, but in modeling the vector field the situations are different: all components of the displacement vector are computed at the same points in thermo-elasticity problem, but in solving Navier-Stokes equation different velocity components are defined in the middles of the different edges of staggered grids.

Because of such peculiarities, it is necessary to construct a representative set of algorithms for graphic visualization, which we describe briefly below. The U3D file format uses conventional graphic primitives, i.e. direct line segments and triangles for drawing curves and surface. So the goal of the following algorithms is to provide the necessary set of points at the isolines and isosurface, by means of interpolating grid functions, in order to give qualities presentation of computed physical fields.

3.1 Mathematical algorithms

Here we describe algorithms for constructing isolines, isosurfaces and vector fields.

We suppose that numerical scalar function $\left\{f_{i,j,k}\right\}$ are set in grid nodes of non-uniform parallelepiped grid $\Omega^h = \left(\left\{x_i\right\}_{i=1}^{L}, \left\{y_j\right\}_{j=1}^{M}, \left\{z_k\right\}_{k=1}^{K}\right)$. For vector fields, numerical vector function $\vec{p} = \{u, v, w\}$, $\vec{p} = u\vec{e}_x + v\vec{e}_y + w\vec{e}_z$ is defined in grid edges: on the middles of x-edges are set *u* components, on the y-edges – *v*, on the z-edges – *w*.

3.1.1 Isolines

Let $\{c_n\}_{n=1}^{N}$ be a set of real numbers. Need to construct lines $\{l_n = (x_n(t), y_n(t), z_n(t)\}_{n=1}^{N}$ in some crossection, under conditions $\tilde{f}(x_n(t), y_n(t), z_n(t)) = c_n$. Function $\tilde{f}$ is some interpolation of *f*, $\tilde{f}(x_i, y_j, z_k) = f_{i,j,k}$. To construct isolines of scalar function in crossection we interpolate function *f* as the following. Let x-crossection is $x=x_s$, than interpolating function *u* is defined by 2D set of values $u_{j,k}$:

$$u_{j,k} = \alpha f_{i,j,k} + (1-\alpha) f_{i+1,j,k},$$

$$x_i \le x_s \le x_{i+1}, \quad \alpha = \frac{x_{i+1} - x_s}{x_{i+1} - x_i}.$$

To search isoline $u=c_n$ for given constant c_n, we determinate does this isoline belongs or not to each finite element $E_{j,k} = \{(y_j \le y \le y_{j+1}), (z_k \le z \le z_{k+1})\}$ (set flag true or false). In can be done easy: c_n belongs to $E_{j,k}$ if $c_n \in [\breve{u}_{j,k}, \hat{u}_{j,k}]$, where

$$\breve{u}_{j,k} = \min(u_{j,k}, u_{j+1,k}, u_{j,k+1}, u_{j+1,k+1}),$$

$$\hat{u}_{j,k} = \max(u_{j,k}, u_{j+1,k}, u_{j,k+1}, u_{j+1,k+1}).$$

Then, for elements with flag=true, the function u is interpolated in the middle of element. In each $E_{j,k}$ we obtain four triangle, in the each of them u are interpolated ($\tilde{u}(y,z), \tilde{u}(y_j, z_k) = u_{j,k}$). Then search across function $\tilde{u}$ with 8 different edges, see Fig.2. By means of backward linear interpolation, we find the cross-point of isoline $u = c_n$ with the edges. There are four variants of isoline behavior in finite elements see Fig.2.

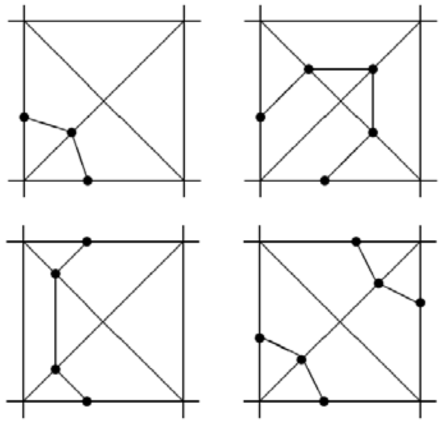

Figure 2: Four variants of isoline in finite element.

Each isoline has been stored as set of segments.

When isolines are visualized, need to fill domain between l_n and l_{n+1} isolines in b_n colour ($\{b_n\}_{n=1}^{N}$ – certain set of colours).

Let domain between c_{n-1} and c_n was filled (see Fig.3a). For pain in b_i color we will do three steps:

1) If $c_n < \breve{u}_{j,k}$ and $\hat{u}_{j,k} < c_{n+1}$, then finite elements $E_{j,k}$, are filled in b_i,

2) If $c_n < \breve{u}_{j,k}$ and $\breve{u}_{j,k} < c_{n+1} < \hat{u}_{j,k}$, then finite elements $E_{j,k}$, are filled in b_i too (fig. 3b),

3) If $\breve{u}_{j,k} < c_n < \hat{u}_{j,k}$, then finite element $E_{j,k}$ divide by l_n in to parts:

$$E_{j,k} = \breve{E}_{j,k} \cup \hat{E}_{j,k},$$

$$\{\tilde{u}(x,y,z) < c_n | (x,y,z) \in \breve{E}_{j,k}\},$$

$$\{\tilde{u}(x,y,z) > c_n | (x,y,z) \in \hat{E}_{j,k}\}.$$

Elements $\breve{E}_{j,k}$ are filling in b_n colour (Fig. 3c).

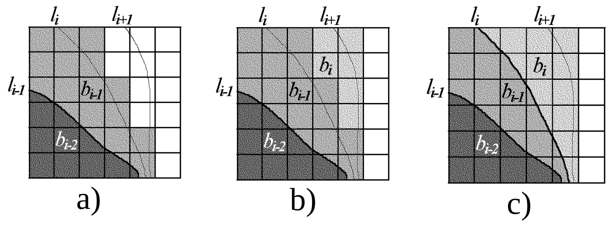

Figure 3: Constrain isolines and fill domains (*i* on fig. mean *n*).

When we say "*fill in b*", we mean, that corresponding triangles are constructed and saved with this color.

After fill in b_n color, we consider b_{n+1}.

3.1.2 Isosurfaces

Let we have grid function $f = \{f_{i,j,k}\}$ on parallelepiped grid Ω^h and constant *c*, which equal *f* on isosurface. Need to find locus *(x,y,z)* on which $\tilde{f}(x,y,z) = c$, where function $\tilde{f}$ is some interpolation of *f*, $\tilde{f}(x_i, y_j, z_k) = f_{i,j,k}$.

As in the case for isoline, for isosurface, first we determine whether any isoline belongs to each finite element $E_{i,j,k} = \{(x_i \le x \le x_{i+1}), (y_j \le y \le y_{j+1}), (z_k \le z \le z_{k+1})\}$ i.e. set flag true or false.

If finite element $E_{i,j,k}$ intersects with surface, we divide it into five tetrahedrons. Function *f* is interpolated in these and we search intersection isosurface and tetrahedron. There are two types of behavior isosurface in tetrahedron see.fig.4.

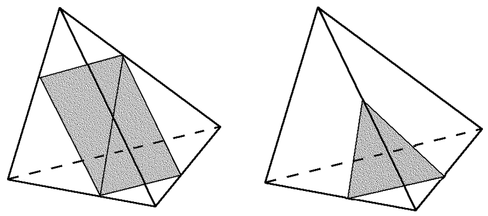

Figure 4: Two variants of isosurface in tetrahedron.

3.1.3 Vector fields

As in the case of isolines, let us have 2-dimensional function $\{f_{i,j}\}$ on non-uniform rectangle grid Ω^h. Also we have a set of points $\{P_l = (x_l, y_l, z_l)\}_{l=1}^{n}$, in which need to visualize the gradient of f, i.e. vector field $\vec{p}$. For computing $\vec{p} = \nabla f$ f we use the following formulas:

$$u_l = \frac{1}{h_i^x}\left(\alpha(f_{i+1,j} - f_{i,j}) + (1-\alpha)(f_{i+1,j+1} - f_{i,j+1})\right),$$

$$\alpha = \frac{y_{j+1} - y_l}{h_j^y},$$

$$v_l = \frac{1}{h_j^y}\left(\beta(f_{i,j+1} - f_{i,j}) + (1-\beta)(f_{i+1,j+1} - f_{i+1,j})\right),$$

$$\beta = \frac{x_{i+1} - x_l}{h_i^x},$$

$$\left|\nabla f_l\right| = \sqrt{(u_l)^2 + (v_l)^2},$$

$$x_i < x_l \le x_{i+1},\ y_j < y_l \le y_{j+1},\ h_i^x = x_{i+1} - x_i,\ h_j^y = y_{j+1} - y_j.$$

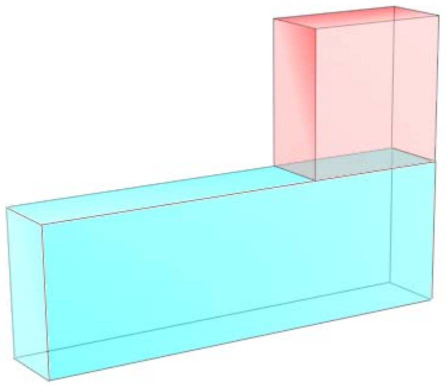
Figure 5: Geometry (the part of aluminum electrolytic cell).

For visualize gradients, they are scaled to minimum step of grid $\min(x_i, y_j)$. Let us note, what set $\{P_l\}_{l=1}^n$ are middles of finite elements of new, uniform grid $\tilde{\Omega}^h = \left(\{\tilde{x}_i\}_{i=1}^L, \{\tilde{y}_j\}_{j=1}^M, \{\tilde{z}_k\}_{k=1}^K\right)$. The grid $\tilde{\Omega}^h$ cover grid Ω^h and coincide with it if Ω^h is uniform one.

Let us have vector field $\vec{p} = (u, v, w)$ on non-uniform rectangle grid Ω^h. There are various data types of $\vec{p}$: components u,v,w can be set in grid nodes or in the middles of finite elements. Also each component can be set on the edges: u on x-edges, v on y and w on z. For all of these data types we use simple common algorithm for visualize: each component $\vec{p}$ is interpolated to points of set $\{P_l\}_{l=1}^n$.

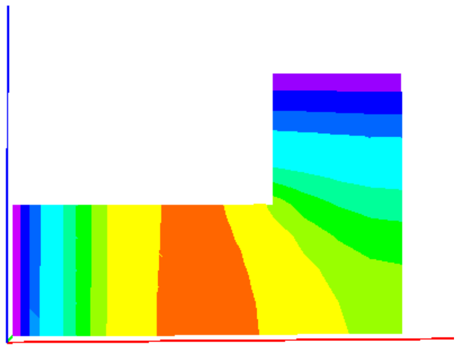
Figure 6: Isolines of temperature field in the crossection of the part of aluminum electrolytic cell.

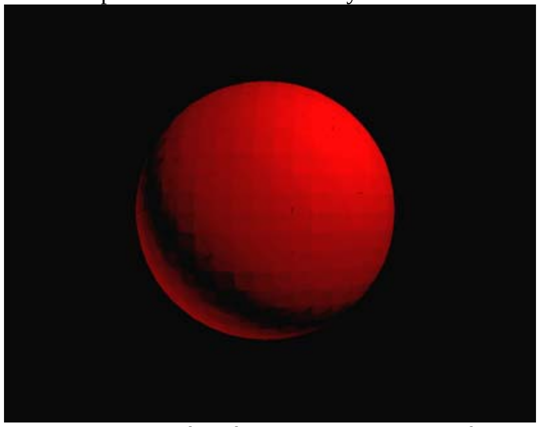
Figure 7: Isosurface for spherical symmetric field.

3.2 Postprocessing Examples

In this section we demonstrate the results of postprocessing. The standard of U3D presents an extensible format for downstream 3D CAD repurposing and visualization. So, the aim of postprocessor is to convert the results of the above algorithms into necessary form, in order to provide existing GUI tools by input data.

In Fig.5 the geometry of the computational domain (the part of aluminum electrolytic cell: anode, blooms) was visualized. In Fig.6 we demonstrate isolines of the temperature field for heat-distribution problem. We have done computation of the problem distribution of combined electrostatic and temperature field. Figure 7 and Figure 8 illustrate the visualization of the isosurface and vector field for other physical examples.

In order to light a scene we use two point sources of white light. Colors of objects are generated automatically. Textures of surfaces are set very simply. In conclusion, let us note, that visualization by U3D tools may be more presentable, if the scene parameters are set correctly.

In conclusion it is possible to say that U3D format presents the reach graphic possibilities and the next work would be oriented for extending the usage of these tools for PDE-PTK

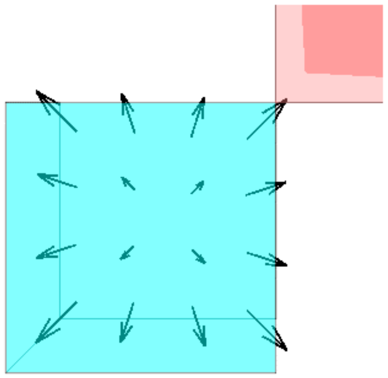
Figure 8: Vectors field.

4. REFERENCES

[1] Golosov I.S., Gorbenko N.I., Gurieva Y.L., Il'in V.P. and other. Program package BASIS-A for simulation of technological processes in an aluminum cell.-Krasnoyarsk: NTZ "Light metals". Proceedings of the 8th International Conference "Aluminum of Siberia - 2002", 2002, 187-192.

[2] Il'in V.P. Finite difference and finite volume methods for elliptic equations (in Russian). – Novosibirsk, NCC Publ., 2001

[3] Il'in V.P. Incomplete factorization methods. Singapore, World Sci. Publ., 1992

[4] Ecma-363/ 2nd Edition / August 2005 / Universal 3D File Format/(http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-363.pdf).

[5] Basov K.A. ANSYS in the examples and tasks. (in Russia) / Moscow: Computer Press, 2002. – p. 224.

[6] Il'in V.P. Computational informational technologies of mathematical modeling. – Avtometriya, N1, 2000, 3-13

[7] Il'in V.P. Geometrical and functional modeling in mathematical physics problems (in Russian). – Novosibirsk, Computational Technologies, v.6, N2, 2001, 315-321

About the authors

Sergey Solovev is a Ph.D. He is Software Engineer in Intel A/O, Novosibirsk R&D Branch. His contact email is Sergey.a.solovev@intel.com .

Valery Ilyin is a professor at Institute of Computational Mathematics and Mathematical Geophysics, Department of Mathematical Problems in Physics and Chemistry. His contact phone numbers are: +7(383) 3306062. He is Senior Software Engineering in Intel A/O, Novosibirsk R&D Branch. His contact email is Valery.p.ilyin@intel.com