UDC 536.24 DOI: 10.15222/TKEA2018.2.33

Ph.D. V. E. TROFIMOV, Ph.D. A. L. PAVLOV, Y. G. MAMYKIN

Ukraine, Odessa National Polytechnic University
E-mail: vovic@ukr.net

CAD/CAE METHOD OF SOLVING THE HYDRODYNAMIC PROBLEM WHILE DEVELOPING POWERFUL ELECTRONIC DEVICES

The article presents examples of the solution of the hydrodynamic problem that arises in the development of powerful electronic devices requiring liquid cooling using the CAD/CAE modeling method. The authors consider poorly documented or undocumented features of such solution based on the use of free software packages — SALOME, OpenFOAM and ParaView for the CAELinux operating system platform.

Keywords: liquid cooling of electronic devices, CAD/CAE-modeling, CAELinux, SALOME, OpenFOAM, ParaView.

While designing electronic devices, the need to solve the hydrodynamic problem arises in the case when, in order to ensure a given thermal regime of electronic components, it is necessary to use forced liquid cooling realized as coolers of a particular design. For a developer of electronic devices, the end result of solving the hydrodynamic problem is not so much an estimate of the flow velocity and liquid flow rate in the cooler as the choice of a pump capable of overcoming the hydrodynamic resistance of the cooler and providing the velocity and flow rate with given values obtained during thermal calculations. It is not difficult to choose the pump parameters when the hydrodynamic characteristic is known for the cooler, i.e. the dependence of the liquid pressure drop at the inlet of the cooler versus the flow rate or velocity of the liquid flowing through it. With streamlined flow, when the hydrodynamic resistance of the cooler is determined by frictional forces, the relationship between the pressure drop and the velocity of the liquid is expressed by the Darcy's law [1], which, taking into account the hydrodynamic resistance at the inlet to the cooler and at the outlet from it, is represented by the well-known equation [2]

$$\Delta p = \left[\xi_{\text{in}} + \lambda\left(\frac{l}{d}\right) + \xi_{\text{out}} \right] \frac{\rho V^2}{2}, \qquad (1)$$

where Δp — pressure drop of the coolant at the cooler inlet and at its outlet;
ξ_{in}, ξ_{out} — coefficients of hydrodynamic resistance at the inlet and outlet of the cooler, respectively;
λ — coefficient of hydrodynamic resistance (friction) along the length l;
l — length of the coolant flow path in the cooler;
d — characteristic cooler size;
ρ, V — density and liquid velocity in the cooler, respectively.

The seeming simplicity of this equation conceals considerable difficulties in determining the coefficients of hydrodynamic resistance it includes. Known analytical dependencies providing satisfactory results for practical applications are obtained for coolers in which the flow of liquid occurs in rectilinear channels of regular shape (circular or rectangular cross-section) [3]. For cases when there are some form of obstructions in the coolant channels, local constrictions, extensions and rotations, the equations for calculating are empirical or semi-empirical, are oriented to concrete design solutions and have a strictly limited field of application [3, 4].

The trend to complicate the design of coolers due to the realization in them of nontraditional flows [5—10] makes it impossible to use equation (1) for practical calculations and requires the use of other approaches. One of them is mathematical modeling, which implements the following algorithm for solving the hydrodynamic problem [7]:

— to develop a 3D geometric model, the composition, shape, dimensions and applied materials of which adequately reflect the design of the cooler;

— to determine the distribution of flow velocity and fluid pressure in the geometric model of the cooler by solving within its boundaries the system of nonstationary three-dimensional continuum and Navier—Stokes equations with allowance for the specified hydrodynamic effects, using the finite element method;

— to present the results of the solution in a form convenient for their interpretation and subsequent intellectual analysis (tables, graphs, diagrams, visualization and animated pictures).

Obviously, from the point of view of the developer of electronic devices, implementing such an algorithm should not be about writing their own computational procedures in any algorithmic language, but about the use of specialized software related to CAD/CAE systems.

Today there is a variety of such software. There is a classification of the selection criteria for the software specifying the features and main characteristics of each criterion [11]. However, the fundamental requirement to use exclusively legal software in the design process takes the proprietary criterion to the first place.

It is known that proprietary CAD/CAE systems have owners who provide chargeable control over systems development, distribution, modification and use. For this reason, the functionality and reliability of proprietary CAD/CAE systems are not questioned. Such software for solving the hydrodynamic problem includes, for example, ANSYS CFX, SolidWorks, Elmer. Its use requires the purchase of a license, the cost of which reaches the price of a small economy car.

An alternative to proprietary are free CAD/CAE systems, which are considered to be less functional and less reliable software. However, our experience of its successful use in the design of sufficiently complex coolers [5—10] allows us to conclude that this judgment is wrong.

To solve the hydrodynamic problem, the developer can use free software, such as the SALOME geometric modeling system [12], the OpenFOAM mathematical modeling system [13], and the ParaView parallel computing visualizer [14]. The SALOME tools allow you to create a virtual 3D geometric model of the cooler and generate its finite-element representation (meshing). OpenFOAM solvers (functions in terms of C++) allow you to import from SALOME a meshed model of the cooler, set the initial and boundary conditions for the problem, and solve the above-mentioned equations of mathematical physics within the finite element model.

The solution obtained is an array of numerical values in nodes of the meshed model, the number of which can reach $10^5—10^6$. ParaView functions allow using this array of values to determine the hydrodynamic characteristic of the cooler and to present it in the form of a table, a two-dimensional graph, a three-dimensional diagram, a visualization picture of the change in pressure and velocity in space, and an animated picture of their change in time.

Thus, these software packages interact with each other by transmitting the information about the cooler from SALOME to OpenFOAM and then to ParaView, and together they form a CAD/CAE system that implements the above algorithm for solving the hydrodynamic problem. This system is a complete analog of known proprietary systems, without any loss of functionality.

One of the main obstacles to the active use of SALOME, OpenFOAM and ParaView for solving the hydrodynamic problem is their weak documentation. Official user manuals from developers [12—14] help mastering the basic techniques using the simplest examples, which are far from the real designs of coolers. Known applications of this software repeat the main declarations of official manuals and are focused on solving other problems for other modeling objects [15—17].

The goal of this paper is to show practical examples of undocumented or poorly documented features of using this CAD/CAE system for the solution of the hydrodynamic problem based on the experience of designing liquid coolers for such electronic devices as microprocessors. SALOME, OpenFOAM and ParaView are cross-platform software products and can run under different operating systems. However, the rational approach is to use them on the CAELinux operating system platform [18] for several reasons: first, it corresponds to the spirit of free software; secondly, the installation distribution of CAELinux already includes these software packages, which eliminates the need for additional settings to ensure the correct transmission of information between the packages.

Example 1
Fundamentals of solving the hydrodynamic problem by the CAD/CAE method

Let us consider a water cooler used for thermal management of the Intel Core i7 microprocessor. The cooler is a closed rectilinear channel with rectangular cross section, whose base is in direct thermal contact with the pedestal of the microprocessor body, serving for heat removal (**Fig. 1**, *a*). The flow volume of the cooler is a parallelepiped, symmetric with respect to the plane parallel to the $Y0Z$ plane (Fig. 1, *b*), with the following parameters: width $X = 60$ mm, height $Y = 15$ mm and length $Z = 70$ mm. Water flows along the Z axis at a temperature of 20°C, the water velocity at the cooler inlet is $U = 0.01$ m/s.

It is required to determine the pressure drop of water at the inlet to the cooler using the icoFoam solver, which is designed to solve the above equations of mathematical physics.

The solution of the problem is carried out in accordance with the basic techniques outlined in [12—14, 18], in the following order.

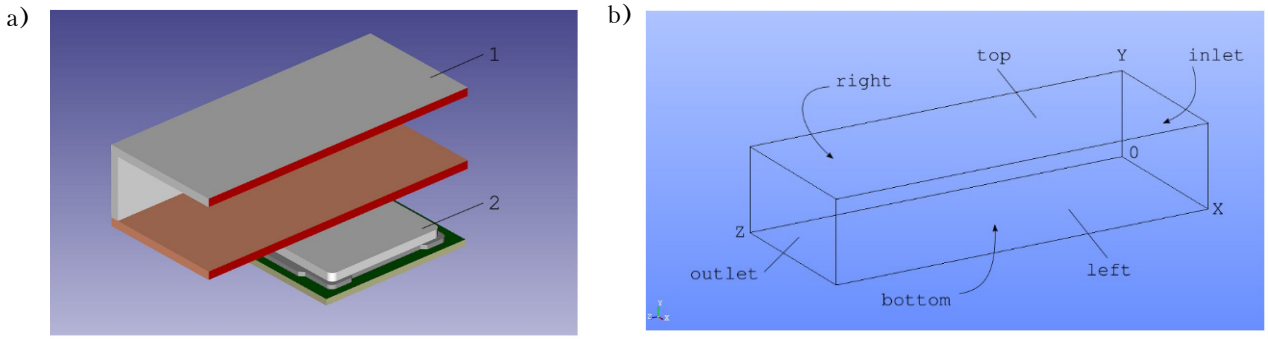


Fig. 1. Model of the microprocessor with the cooler (*a*) and the flow volume of the cooler (*b*): *1* — symmetrical half of the cooler; *2* — microprocessor

1. Create a project directory

1.1. Create a project directory with a name, for example, *rect_cooler* (rectangular cooler) in the system directory *Home* of the CAELinux operating system.

1.2. Place directories and files of icoFoam solver to the project directory *rect_cooler*. To do this, go to the system directory of the CAELinux operating system */opt/openfoam211/tutorials/incompressible/icoFoam/cavity* and copy directories named *0*, *constant* and *system* to the directory *rect_cooler* **(Fig. 2)**.

1.3. Go to the directory *constant* and empty the *polyMesh* directory. To do this, delete the *blockMeshDict* and *boundary* files in the *polyMesh* directory.

2. Create a geometric model of the cooler flow volume in SALOME

2.1. Run SALOME and go to geometric module Geometry.

2.2. Using the Box graphic primitive to create a geometric model of the flow volume in accordance with the specified dimensions in the form of its symmetrical half with a width of $X/2 = 30$ mm, and assign it a name, for example, coinciding with the name of the *rect_cooler* project directory.

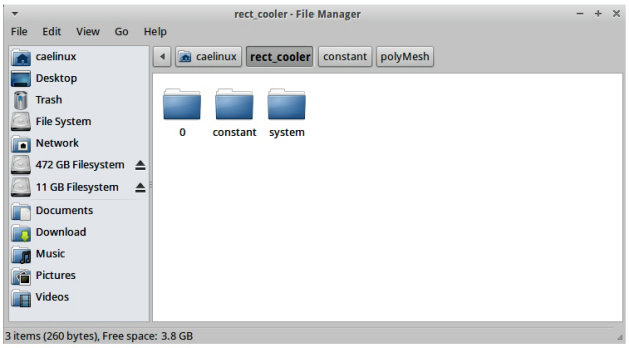


Fig. 2. The contents of the project directory *rect_cooler* before solving the problem

2.3. For the *rect_cooler* geometric model, use the Create Group operation to select the surfaces (see Fig. 1, *b*) that correspond to the specified boundary conditions of the problem:

— the water inlet into the flow volume occurs through the surface inlet, perpendicular to the *Z* axis and coinciding with the *Y*0*X* plane;

— the water outlet from the flow volume occurs through the surface outlet, perpendicular to the *Z* axis and spaced from the *Y*0*X* plane at a distance equal to the length of the flow volume;

— the *top* and *bottom* surfaces of the flow volume, perpendicular to the *Y* axis, as well as the *right* surface, perpendicular to the *X* axis and coinciding with the plane *Y*0*Z*, are solid impermeable walls;

— the *left* surface of the flow volume, perpendicular to the *X* axis and spaced from the plane *Y*0*Z* at a distance equal to the width of the model, is the plane of symmetry.

2.4. Save the result of creating a geometric model of the flow volume of the cooler to the *rect_cooler_geometry.hdf* file in the *rect_cooler* project directory, making sure that the Object Browser window of the SALOME desktop contains all the surfaces created as part of the *rect_cooler* model **(Fig. 3**, *a*).

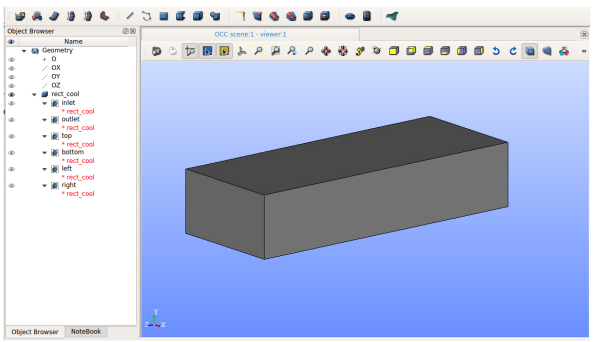
3. Mesh the geometric model of the flow volume of the cooler in SALOME

3.1. Without quitting SALOME, go to the Mesh meshing module.

3.2. Since the geometrical model of the flow volume of the cooler has the form of a parallelepiped, use Hexahedron finite elements in the form of hexahedrons. Using the Create Mesh operation, set the following meshing parameters:

— Wire Discretization algorithm for dividing the edges of the model, Automatic Length hypothesis of dividing the edges of the model with the quality value, e.g., 0.5;

a)



b)

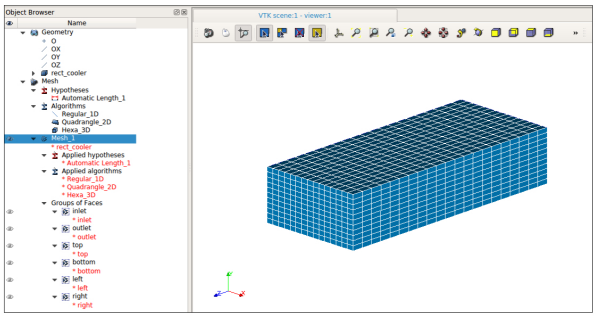


Fig. 3. The geometric model of the flow volume of the cooler (*a*) and the result of its meshing (*b*)

— Quadrangle (Mapping) algorithm for model surface partitioning;

— Hexahedron (*i, j, k*) algorithm for partitioning the volume of the model.

3.3. With the help of the Compute operation, perform a meshing of the model and its surfaces that correspond to the boundary conditions of the problem (see 2.3). The result of meshing the *rect_cooler* geometric model of the cooler flow volume is shown in Fig. 3, *b*. For meshing parameters given in 3.2 the model contains 3465 finite elements.

4. Conversion of the flow volume model of the cooler into the OpenFOAM format

4.1. While in the meshing module Mesh, export the cooler model to the I-DEAS (integrated design and engineering analysis software) intermediate .unv format. To do this, select the Mesh_1 object in the Object Browser window (see Fig. 3, *b*) and select File → Export → UNV file in the main menu of the SALOME desktop.

4.2. Save the result in the *rect_cooler* project directory in the file with a name, for example, *Mesh_1.unv*. Minimize the window or exit the SALOME geometric modeling system (optional).

4.3. Open the OpenFOAM terminal window by selecting Applications → caelinux → OpenFOAM 2.2.1 terminal in the main menu of the CAELinux

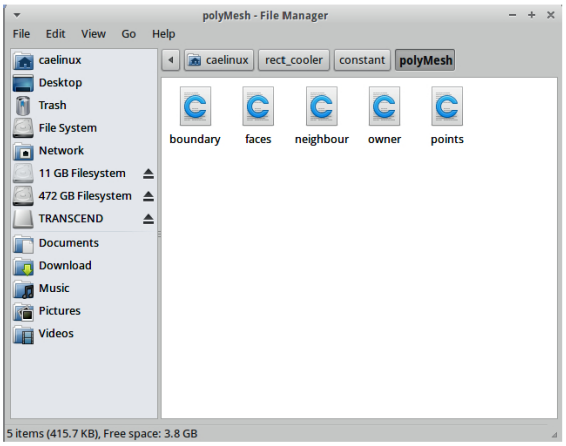


Fig. 4. The contents of the subdirectory *rect_cooler/constant/polyMesh* after converting the cooler model from SALOME to OpenFOAM

operating system desktop, and enter the *rect_cooler* project directory.

4.4. Convert the cooler model from the intermediate I-DEAS format to the OpenFOAM format by executing the `ideasUnvToFoam Mesh_1.unv` command in the terminal window. Make sure that the protocol for executing the `ideasUnvToFoam` command was completed with the `End` string, indicating that the conversion was successful without errors. Also, make sure that the result of the conversion was the creation of *boundary*, *faces*, *neighbor*, *owner* and *points* files in the *rect_cooler/constant/polyMesh* project subdirectory **(Fig. 4)**.

4.5. Convert the dimensions of the cooler model specified in the SALOME geometric modeling system in millimeters to meters, accepted in the OpenFOAM mathematical modeling system by default. To do this, execute the `transformPoints -scale '(0.001 0.001 0.001)'` command in the terminal window and make sure that there are no errors.

5. Prepare the flow volume model of the cooler to the computation by OpenFOAM

5.1. Define the types of surfaces of the cooler model that correspond to the boundary conditions of the problem (see 2.3). To do this, open the *rect_cooler/constant/polyMesh/boundary* file in any text editor and make the following changes:

— for the *inlet* and *outlet* surfaces, specify the *patch* type;

— for the *top*, *bottom* and *right* surfaces, specify the *wall* type;

— for the *left* surface, specify the *symmetryPlane* type.

The *rect_cooler/constant/polyMesh/boundary* file should have the following form (the order

of the surfaces and the value of the `startFace` parameter can be different):

```
6
(
    left
    {
        type            symmetryPlane;
        nFaces          231;
        startFace       9684;
    }
    right
    {
        type             wall;
        nFaces           231;
        startFace        9915;
    }
    inlet
    {
        type            patch;
        nFaces          165;
        startFace       10146;
    }
    outlet
    {
        type            patch;
        nFaces          165;
        startFace       10311;
    }
    top
    {
        type             wall;
        nFaces          315;
        startFace       10476;
    }
    bottom
    {
        type             wall;
        nFaces          315;
        startFace       10791;
    }
)
```

5.2. Set the initial and boundary values of the water velocity, corresponding to the initial data and the boundary conditions of the problem (see 2.3). To do this, open the *rect_cooler/0/U* file in any text editor and make the following changes:

— on the *inlet* surface, set the velocity value at the inlet to the flow volume of the model: 0.01 m/s along the *Z* axis, and 0 m/s along the remaining axes;

— define the *outlet* surface as a surface with zero gradient;

— on the *top*, *bottom* and *right* surfaces, set zero velocity value in all coordinate axes;

— define the *left* surface as a plane of symmetry.

The *rect_cooler/0/U* file should have the following form (the order of the surfaces may be different):

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0.01);
    }
    left
    {
        type            symmetryPlane;
    }
    right
    {
        type            fixedValue;
          value           uniform (0 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    top
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    bottom
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}
```

5.3. Set the initial and boundary values of the water pressure, corresponding to the boundary conditions of the problem (see 2.3). To do this, open the *rect_cooler/0/p* file in any text editor and make the following changes:

— set the *inlet, top, bottom* and *right* surfaces as surfaces with zero gradient;

— set zero pressure value on the *outlet* surface;

— define the *left* surface as a plane of symmetry.

The *rect_cooler/0/p* file should have the following form (the order of the surfaces may be different):

```
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    left
    {
        type            symmetryPlane;
    }
    right
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
}
```

5.4. Set the physical properties of the water according to the conditions of the problem. When using the icoFoam solver, it is sufficient to specify only the kinematic viscosity value *nu*. At a temperature of 20°C, *nu* = 0.000001 m²/s. In any text editor open the *rect_cooler/constant/transportProperties* file and enter the value of this parameter. The file should contain:

```
nu    nu [0 2 -1 0 0 0 0] 0.000001;
```

5.5. Set computation parameters. To do this, open the *rect_cooler/system/controlDict* file in any text editor and make the following entries:

— set the start time of the computation (*startTime*) to zero;

— the end time of the computation (*endTime*) is equal to the time when the steady flow of water in the cooler sets in. To quantify it, an empirical rule is used, according to which for a given flow velocity at the inlet, the flow volume must be updated 10 times, i.e. with an inlet flow velocity of 0.01 m/s and a cooler length in the direction of flow of 0.07 m the time of setting the steady state is 70 s;

— the *deltaT* time step of the computation is accepted so that during the computation the maximum value of the Courant Number (*Courant Number max*) displayed in the OpenFOAM terminal window for the count protocol does not exceed 1.0. For the initial data received, the value of *deltaT* should be taken as 0.1 s;

— interval of recording the intermediate results of the *writeInterval* computation should be set. For example, if the computation time is 70 s, the time step is 0.1 s, the interval of recording intermediate results is 20, then 35 subdirectories containing velocity and pressure values for the cooler model for every 2 seconds will be created in the project's directory.

The rest of the parameters remain unchanged [13].

The *rect_cooler/system/controlDict* file should look like this:

```
application         icoFoam;
startFrom           startTime;
startTime           0;
stopAt              endTime;
endTime             70;
deltaT              0.1;
writeControl        timeStep;
writeInterval       20;
purgeWrite          0;
writeFormat         ascii;
writePrecision      6;
writeCompression    off;
timeFormat          general;
timePrecision       6;
runTimeModifiable   true;
```

6. Running the task and obtaining a solution

6.1. Run the task for computation. To do this, execute the `icoFoam` command in the OpenFOAM terminal window while in the project's directory.

6.2. Observe the computation process watching the count protocol which is dynamically displayed in real time during the solution of the problem in the terminal window. The final part of the count protocol is shown in **Fig. 5**. The protocol contains the current time (*Time*) from the moment of the beginning of the water flow with the time step *deltaT*, the average (*Courant Number mean*) and the maximum (*Courant Number max*) values of the Courant number, initial (*Initial residual*) and final (*Final residual*) values of the calculation of the remaining projections of velocity *U* and pressure *p* for the corresponding current time *Time*, the time of task solving (CPU time) *ClockTime* and other parameters.

6.3. Wait for the successful completion of the task, when the output of the count protocol data stops in the terminal window at the *endTime* moment, and the last line of the count protocol contains the *End* value. The content of the *rect_cooler* project directory for completed computation is shown in **Fig. 6**. Depending on the processor performance and the size of the computer's main memory, the time for solving the problem can vary from 4 to 15 seconds or more.

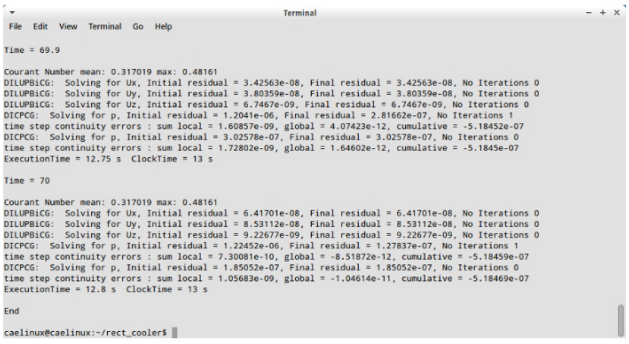


Fig. 5. The final part of the count protocol

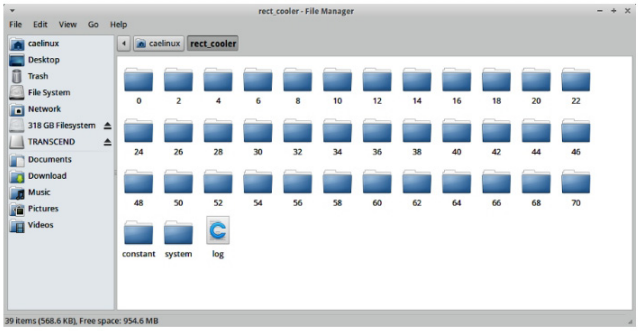


Fig. 6. The contents of the *rect_cooler* project directory after the computation is completed

6.4. Make sure that 35 subdirectories are created in the *rect_cooler* project directory, the names of which correspond to the given time intervals of the flow in 2-second increments, containing the required values of the velocity U and the pressure p at the nodes of the meshed model of the flow volume of the cooler (see Fig. 6).

7. Analysing the results of solving the problem in ParaView

7.1. Download the results of the solution of the problem, i.e. the contents of the *rect_cooler/2 ... rect_cooler/70* subdirectories to the ParaView parallel computing visualizer. To do this, run the `paraFoam` command in the OpenFOAM terminal window. As a result of the command, the main window of the ParaView will open on the desktop of the CAELinux operating system, and an empty file named *rect_cooler/rect_cooler.OpenFOAM* will be created in the project's directory, through which the visualizer will access the data of subdirectories of the *rect_cooler* project directory.

7.2. Display the image of the geometric model of the cooler on the desktop of the visualizer and visualize the pressure distribution p in it **(Fig. 7)**.

The pressure values obtained as a result of the solving the problem are the specific values related to the density of the cooling liquid and have the dimension m^2/s^2. At a temperature of 20°C, the density of water is 998.2 kg/m^3. Thus, in order to ensure a given velocity at the inlet to the cooler, the water pressure drop must be $\Delta p = 998.2 \cdot 0.000105 = 0.1$ Pa.

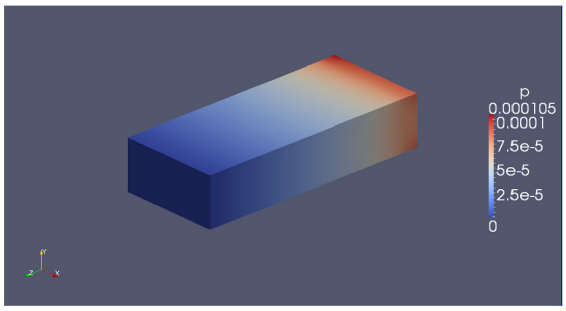


Fig. 7. Pressure distribution p in the flow volume of the cooler

Example 2
Cooler design with hose connections

In the design of the cooler, two cylindrical hose connections are added to connect the chiller with a diameter of 9 mm and a length of 13 mm **(Fig. 8**, *a*).

The water flow rate through the cooler is assumed to be the same as in Example 1, in order to be able to compare the results, so that its inlet velocity is set to U = 0.141 m/s. The solution of the problem was carried out according to the method described in the previous example, with the following features:

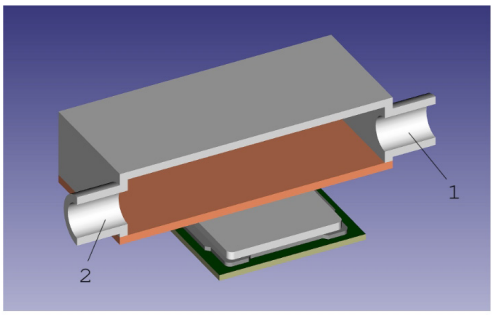
— the flow volume of the cooler has a complex shape, its geometric model (Fig. 8, *b*) is created in SALOME using the *Fuse* logical operation of combining of the Box graphic primitive and two Cylinder graphic primitives;

— the model has 10 boundary surfaces: inlet, outlet, symmetry plane and seven solid impermeable walls (see Fig. 8, *b*);

— *Tetrahedron* finite elements in the form of tetrahedrons (pyramids) were used for meshing, with the value of the edge partitioning quality of the model 0.6 **(Fig. 9)**, the number of finite elements was 63715.

The result of the solution of the problem, which is a visualization of the pressure distribution in the flow volume of the cooler, is shown in **Fig. 10**. It can be seen that in order to ensure a predetermined water flow rate through the cooler, the pressure drop should be equal to $\Delta p = 998.2 \cdot 0.0129 = 12.9$ Pa. Comparing this value with the result obtained in Example 1, one can see that as the flow velocity at the inlet to the cooler increases by an order of magnitude, the pressure drop increases by two orders of magnitude, which corresponds to the

a)



б)

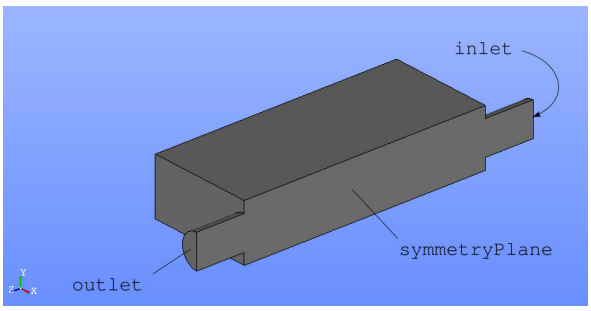


Fig. 8. Model of a microprocessor with a symmetrical half of the cooler with hose connections (*a*) and the flow volume of the cooler (*b*)

1 — inlet hose connection; *2* — outlet hose connection

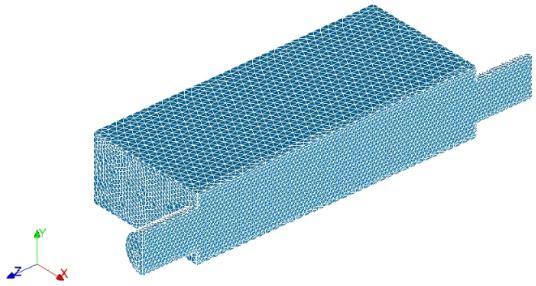


Fig. 9. The result of meshing the geometric model of the flow volume of the cooler with hose connections

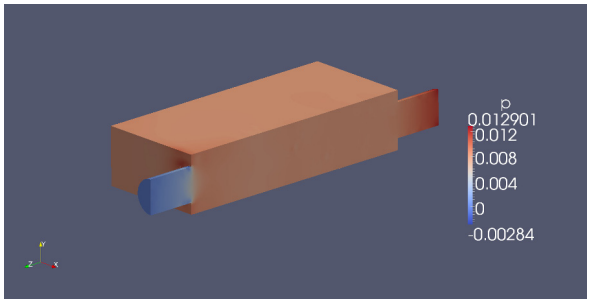


Fig. 10. Pressure distribution p in the flow volume of the cooler

quadratic character of the hydrodynamic characteristic described by equation (1). This result is one of the reasons for trusting the software used and the method proposed.

Conclusions

Considered on simple examples, the CAD/CAE method for solving the hydrodynamic problem using free software SALOME, OpenFOAM and ParaView on a platform of the CAELinux operating system was successfully used by authors when designing more complex coolers — not only symmetrical but also containing dozens and hundreds of planar and curvilinear boundary surfaces, with a sudden change in the direction of flow and velocity of cooling liquid, when the value of CPU time for obtaining a solution for only one variant of the initial data were tens of hours.

Due to the limited size of the journal article, many important issues were not considered here, such as the adequacy of the geometric model to the real design, the choice of algorithms and meshing hypotheses, their effect on the solution result, the use of a simpleFoam solver that takes into account the turbulence of the coolant flow, ensuring convergence of solution, parallelization of the task to reduce the time of computing, presentation of the solution results not only with visualization pictures, the adequacy of the solution to the experimental data. The authors plan to devote their future publications to these issues.

REFERENCES

1. Berd R., St'yuart V., Laitfut E. *Yavleniya Perenosa* [Transport Phenomena]. Moscow, Khimiya, 1974, 688 p. (Rus)

2. Spokoiny Yu. E., Trofimov V. E., Gidalevich V. B. *Teplomassoobmen v REA* [Heat and Mass Transfer in REA: Collection of tasks. Kiev, Odessa, Lybid, 1991, 224 p. (Rus)

3. Idel'chik I. E. *Aerogidrodinamika tekhnologicheskikh apparatov (Podvod, otvod i raspredelenie potoka po secheniyu apparatov)* [Aerohydrodynamics of Technological Apparatuses (Approach, Diversion and Distribution of Flow Along the Cross-Section of Apparatuses)]. Moscow, Mashinostroenie, 1983, 351 p. (Rus)

4. Reznikov G. V. *Raschet i Konstruirovanie Sistem Okhlazhdeniya EVM* [Calculation and Design of Cooling Systems EVM]. Moscow, Radio i Svyaz', 1988, 224 p. (Rus)

5. Spokoiny M., Trofimov V., Qiu X., Kerner J.M. Enhanced heat transfer in a channel with combined structure of pins and dimples. *Proc. of the 9th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, San Francisco, CA, 2006, pp. 1-21.

6. Spokoiny M., Trofimov V. Collider jets cooling method of microprocessors. *Proc. of the International Microelectronics and Packaging Society ATW on Thermal Management, Session 12 "Liquid, phase-change and refrigeration cooling"*, 2011, Palo Alto, CA, USA, pp. 1-18.

7. Spokoiny M. Yu., Trofimov V. E., Shevchuk M. V. CFD modeling of heat transfer in a rectangular channel with dimplepin finning. *Tekhnologiya i Konstruirovanie v Elektronnoi Apparature*, 2013, no. 2-3, pp. 33-38. (Rus)

8. Trofimov V. E., Pavlov A. L. Animation of contrary jets interaction in the radiator for microprocessor liquid cooling. *Proc. of the 15th International Scientific-Practical Conference "Modern Information and Electronic Technologies"*, Ukraine, Odessa, 2014, pp. 26-27. (Rus)

9. Trofimov V. E., Pavlov A. L., Zhmud E. V. Visualization of the interaction of a jet with a dead-end cavity of the radiator for liquid cooling of a microprocessor. *Proc. of the 16th International Scientific-Practical Conference "Modern Information and Electronic Technologies"*, Ukraine, Odessa, 2015, pp. 160-161. (Rus)

10. Trofimov V. E., Pavlov A. L. Intensification of heat transfer in liquid heat exchangers with dimple-pin finning. *Tekhnologiya i Konstruirovanie v Elektronnoi Apparature*, 2016, no. 1, pp. 23-26. (Rus) http://dx.doi.org/10.15222/TKEA2016.1.23

11. Shehovtsova V. I. [The problem of choice and criteria for evaluating the automated design]. *Visnik NTU "KHPI"*, 2014, no. 26 (1069), pp. 101-108. (Rus)

12. SALOME. *The Open Source Integration Platform for Numerical Simulation* [Electronic resource]. http://www.salome-platform.org (Date of the application 04.04.2018.)

13. OpenFOAM. *The open source CFD toolbox* [Electronic resource]. http://www.openfoam.com (Date of the application 04.04.2018).

14. *ParaView. The open source multi-platform data analysis and visualization application* [Electronic resource]. http://www.paraview.org (Date of the application 04.04.2018).

15. Lazarev T.V. Simulation of thermoelectric state by means of OpenFOAM. *Proceedings of the NTUU "Igor Sikorsky KPI". Series: Chemical Engineering, Ecology and Resource Saving*, 2013, no. 1, pp. 26-30. (Rus)

16. M. de Groot. *Flow prediction in brain aneurysms using OpenFOAM*, 2014 [Electronic resource]. https://www.utwente.nl/en/eemcs/sacs/teaching/Thesis/masterthesis_meindert_de_groot.pdf (Date of the application 06.04.2018.)

17. Juan Marcelo Gimenez, Axel Larreteguy, Santiago Márquez Damián, Norberto Nigro. *Short course on OpenFOAM*

development. ENIEF 2014, Instituto Balseiro, Bariloche, Argentina, [Electronic resource]. https://cimec.org.ar/foswiki/pub/Main/Cimec/CursoCFD/OF_Developers_Course.pdf (Date of the application 06.04.2018.)

18. *CAELinux. Open-source powered engineering* [Electronic resource]. http://caelinux.com (date of the application 04.04.2018.)

Received 02.04 2018

DOI: 10.15222/TKEA2018.2.33
УДК 536.24

К. т. н. В. Є. ТРОФІМОВ, к. т. н. А. Л. ПАВЛОВ, Я. Г. МАМИКІН

Україна, Одеський національний політехнічний університет
E-mail: vovic@ukr.net

CAD/CAE-МЕТОД ВИРІШЕННЯ ГІДРОДИНАМІЧНОЇ ЗАДАЧІ ПРИ РОЗРОБЦІ ПОТУЖНИХ ЕЛЕКТРОННИХ ПРИЛАДІВ

Необхідність вирішення гідродинамічної задачі виникає в тому випадку, коли для забезпечення теплового режиму електронного приладу необхідним є застосування примусового рідинного охолодження, реалізованого у вигляді охолоджувачів тієї чи іншої конструкції. Результатом її рішення є вибір насоса, здатного подолати гідродинамічний опір охолоджувача при заданих значеннях витрати рідини. Вибрати насос не важко, коли відома гідродинамічна характеристика охолоджувача — залежність надлишкового тиску рідини на вході від швидкості рідини, яка протікає через охолоджувач. Тенденція до ускладнення конструкцій охолоджувачів з нетрадиційними течіями виключає використання відомих аналітичних залежностей і вимагає застосування інших методів, наприклад математичного моделювання. Його основні етапи: розробка 3D геометричної моделі охолоджувача; визначення в її межах швидкості і тиску рідини шляхом вирішення системи диференціальних рівнянь нерозривності і Нав'є — Стокса; представлення результатів рішення в вигляді, зручному для аналізу.

Для реалізації такого підходу потрібне спеціалізоване програмне забезпечення (ПЗ) типу CAD/CAE. Дорожнеча пропрієтарних CAD/CAE-систем разом з вимогою використання при проектуванні виключно легального ПЗ викликали необхідність вирішення зазначеної задачі за допомогою вільних програмних продуктів, наприклад системи геометричного моделювання SALOME, системи математичного моделювання OpenFOAM і візуалізатора паралельних обчислень ParaView.

Серед основних перешкод застосування зазначеного ПЗ — його слабка документованість. Мета даної роботи — показати на практичних прикладах, заснованих на досвіді проектування рідинних охолоджувачів для мікропроцесорів, недокументовані особливості використання SALOME, OpenFOAM і ParaView на платформі операційної системи CAELinux. У роботі на двох прикладах докладно розглянуто методику рішення задачі, що дозволяє успішно застосовувати даний підхід при проектуванні складних реальних конструкцій охолоджувачів електронних приладів.

Ключові слова: рідинне охолодження електронних приладів, CAD/CAE-моделювання, CAELinux, SALOME, OpenFOAM, ParaView.

К. т. н. В. Е. ТРОФИМОВ, к. т. н. А. Л. ПАВЛОВ, Я. Г. МАМЫКИН

Украина, Одесский национальний политехнический университет
E-mail: vovic@ukr.net

CAD/CAE-МЕТОД РЕШЕНИЯ ГИДРОДИНАМИЧЕСКОЙ ЗАДАЧИ ПРИ РАЗРАБОТКЕ МОЩНЫХ ЭЛЕКТРОННЫХ ПРИБОРОВ

Приведены примеры решения гидродинамической задачи, которая возникает при разработке мощных электронных приборов, требующих жидкостного охлаждения, методом CAD/CAE-моделирования. Рассмотрены не документированные или слабо документированные особенности ее решения на основе использования свободного программного обеспечения SALOME, OpenFOAM и ParaView на платформе операционной системы CAELinux.

Ключевые слова: жидкостное охлаждение электронных приборов, CAD/CAE-моделирование, CAELinux, SALOME, OpenFOAM, ParaView.

Cite the article as:
Trofimov V. E., Pavlov A. L. , Mamykin Y. G. CAD/CAE method of solving the hydrodynamic problem while developing powerful electronic devices. Tekhnologiya i Konstruirovanie v Elektronnoi Apparature, 2018, no. 2, pp. 33-41. http://dx.doi.org/10.15222/TKEA2018.2.33