ISSN 1063-7796, Physics of Particles and Nuclei, 2024, Vol. 55, No. 3, pp. 313–316. © Pleiades Publishing, Ltd., 2024.

# Real-Time Simulation, Decomposition of Mathematical Models and Algorithms of Interactive Control and Visualization

**A. Degtyarev**[a, b, *] **and V. Khramushin**[b, **]

[a] *St. Petersburg State University, Saint-Petersburg, Russia*

[b] *Krylov Science-Engineering Shipbuilding Society, St. Petersburg, Russia*

**e-mail: a.degtyarev@spbu.ru*

***e-mail: Khram@mail.ru*

Received September 18, 2023; revised November 6, 2023; accepted December 1, 2023

**Abstract**—The issues of interactive control of resource-intensive algorithms seem to be topical in the formulation and design of simulation. The possibility of dynamic reconfiguration of models with independent visual control of three-dimensional physical phenomena and processes in real time is important. Direct simulation allows to achieve practical engineering solutions without traditional analytical limitations. Only the volume of spatial meshes and the accuracy of approximation of the studied engineering objects is an optimization condition.

**DOI:** 10.1134/S1063779624030304

## INTORUCTION

Modern computing systems largely provide practical modeling of processes and phenomena for situations where the physical description is not in doubt [1]. In this case, it is possible to perform real time simulation instead of physical experiments with the same degree of adequacy in obtaining results. The advantages of such approach are clear. Firstly, we have not scale effect in comparison with model experiments. Secondly, simulations are carried out under the conditions of accurate knowledge of the state of the object and the external environment. Thirdly, any simulation conditions can be provided, including extreme ones, which cannot be achieved in a full-scale experiment for safety reasons, or in a model experiment due to the limited capabilities of experimental facilities.

At the same time, the conduct of any experiment implies the possibility of its planning and management, the possibility to intervene in the process, to change these or those parameters, to examine in more detail, as through a magnifying glass, the individual components, in some cases, to repeat the experiment from a certain point, etc. Let us consider this issue on the example of hydroaerodynamics problems. In the case of direct simulation, this imposes certain requirements on the models and simulation algorithms themselves [2–4].

Thus, in fact, we are talking about the development of special computational models, which, without any contradictions, are reduced to strictly defined complexes of functional objects and algorithmic operations. They claim, if successful, to be special descriptions of physical phenomena [2]. Accordingly, special language tools suitable for strictly describing the results aimed at their subsequent theoretical and technological development are also in demand [3].

In general, such simulation is formed by three conditionally independent components [4]:

• separation of solutions by physical processes in the mathematical representation of hydromechanics (physics) processes;

• multi-window toolkit for graphic visualization of simulated phenomena and processes, independent of the computing environment;

• time synchronization system with interactive control of computational model parameters.

These three logically independent components of simulation are linked algorithmically. Further, the features of design and implementation of simulation for interactive real-time modelling are discussed.

## TIME SEPARATION OF COMPUTING PROCESSES WITH INTERACTIVE INTERFACE

In the implementation of the above tasks, it is acceptable to divide software and hardware developments into three conditionally independent directions:

(1) *Actually mathematical modeling*—only arithmetic-logic block with multiple computational cores and shared access to big amounts of RAM for fast reading and modification is involved.

(2) *Automatic control* of the simulation is realized in interrupt handling procedures by one or a group of interval timers.

(3) *Graphical visualization of the results* is organized and executed in interactive communication with external hardware peripherals, including interrupts from timers, cursor indicators, and the computer keyboard.

In this case, the initial orientation on the construction of extremely efficient algorithms of the actual numerical modeling is allowed.

A traditional simulation is built on modeling the processes of development and transformation of complex physical phenomena in time, where each computation clock cycle is unconditionally small. Interrupt processing from timers as well as from external peripherals can be executed in order of priority in time intervals free from modeling.

There are always peculiarities in the operation of graphic visualization procedures. In parallel execution, they involve data arrays from RAM that are simultaneously modified during simulation. However, usually, it does not introduce much distortion in the current results.

The coordination within the procedures of interactive processing of interrupts from external peripheral devices is much more complicated. If graphic scenes are changed or local image fragments are rebuilt, the simulation may not be paused. However, when the parameters of the simulation change, and especially when the state of the simulated environment changes completely, it may be necessary to either immediately pause the simulation or completely restart the entire computations.

## SET OF INTERVAL TIMER PROCEDURES

Let us consider two realizations of interactive control for simulation.

Variant (A). The main attention is execution of a resource-intensive experiment. Usually, this variant is typical for the situation when interactive control of parameters of mathematical models is primary.

Variant (B) is oriented to processing with group of timers. This variant is optimal for numerical modelling in real time.

In variant (A) the adjustment of graphical visualization of current results is implemented as needed. Formally, one interval timer (WaitTime) is sufficient here. Interactive control of the experiment and graphical procedures are performed sequentially when the actual numerical simulation is paused.

*long* **WaitTime**(

*long wait, // delay for independent interrupt processing*

*bool( ****inFree** *)() = null, // free function of the computational experiment*

*long work = 0 ); // control time to execute the calculation cycle [ms]*

For dynamic control of simulation in the program environment, direct access to the computer clock is further defined. Example is optimization of wait and work time intervals.

*long* ***StartTime****, // computer start time of the whole program*

***RealTime,*** *// current execution time of the inFree procedure within WaitTime*

***GetTime****(), // query the exact time in milliseconds (GetTickCount)*

***ElapsedTime****(); // querying the program runtime(**μs**)*

Control operations, including positioning an OpenGL graphics scene, can be performed simply from the keyboard. However, at least one graphics window must be pre-opened and activated.

*byte Window::***WaitKey***()// stop and waiting of new symbol from keyboard*

*byte Window::***GetKey***()// querying and selecting a symbol without stopping the program*

*byte Window::***ScanKey***()// symbol polling without stopping and without sampling from the queue*

*byte Window::***ScanStatus***()// getting associated keys code from buffer*

In option (B), the applied toolkit focuses more on parallel threads for mathematical models, graphical visualization and interactive control.

Here it is possible to use windows as object-oriented programming classes. The same separation is required for independent customization of the context-sensitive OpenGL graphics package. Here, physical phenomena and processes of the actual simulation are similarly separated.

virtual Window& Window::**Timer**() // *virtual procedure for timer processing*

Window& Window::**SetTimer**( μsec, bool( *inFree) ()=null ) //*interval and transaction*

Window& Window::**KillTimer**() // *timer reset*

Here procedure Window::SetTimer also activates virtual call to interrupt procedure Window::Timer() directly inside numerical object. This correctly supports automatic processing of prologue and epilogue algorithms at the beginning and end of reentrant procedures and makes it possible to coordinate the asynchronous execution of interrupts from various external devices using logical flags. The Window::KillTimer()

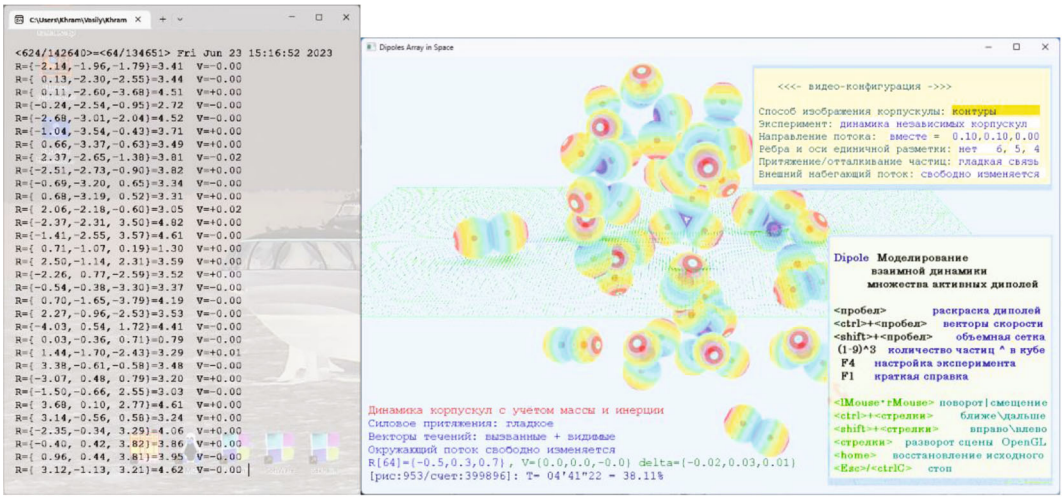


**Fig. 1.** A cloud of mobile and actively interacting polarized particles is presented. On the right menu with the parameters of simulation, below the hint on the control commands are presented. On the text console on the left, control parameters for establishing a solution are displayed without activating the graphics environment.

procedure terminates the interval timer's interrupt loop. Pre-execution of all current interrupts inside for the active program as a whole is provided.

Interrupts from external devices are processed in the same way.

## EXAMPLES OF PROGRAM ENVIRONMENT USING FOR SIMULATION

(A) In setting up a computational simulation in corpuscular mechanics, there is no need to harmonize physical time with a computer clock. Algorithms are optimized for extremely efficient and fast calculations. At the same time, there is no loss of opportunities for detailed visualization of simulated physical processes. Fast response to external interactive control actions is supported.

Direct simulation with detailed drawing of moving spatial objects is useful for visual interpretation of properties and features of models of physical phenomena, including those that cannot be verified by exact analytical solutions. Interactive control of simulation allows to evaluate the adequacy of mathematical models, dynamically determine the areas of existence of physical phenomena and stability of mechanical processes (see Fig. 1).

(B) Carry out of direct simulation for modeling, for example, the storm hydromechanics of a ship, is fundamentally required to coordinate all operations with real time. This case is more determined by the verification of the nature of the modeling results with the ideas of good marine practice. It is assumed that conducting such experiments in the storm seaworthiness of the ship will form a practical testbed for testing and practicing maneuvers in stormy seas. Such experiments provide also on-board intelligence system knowledge base filling. It is possible to use it for front-end engineering in the process of ship hull and naval architecture optimization.

In the present implementation, a simulated ship with hydrodynamic state estimates is displayed in one graphical field. In another graphical window, the sea surrounding the ship with indications of course, speed, running drift and rudder position are shown. This is required for interactive control of storm maneuvering. General parameters of simulation are updated on text consol. In each graphics screen standalone interactive control systems act with the help of cursor, keyboard and context menus. Context-sensitive tooltips are associated with each graphics window (see Fig. 2).

## CONCLUSIONS

Thus, the paper formulates the minimum permissible and sufficient tools for effective statement and objective control of the process and results of simulation. The minimum necessary instrumental tools for visual control of adequacy of execution of mathematical models are considered. Taking into account the architecture of supercomputers, the design of experiments includes the requirements for the execution of calculations and visualization of results in real time. Based on the possibilities of parallel and fast execution of graphical procedures in OpenGL environment, variants of direct and virtual interactive interfaces are proposed. The new toolkit is necessary to analyze the adequacy of mathematical modeling in local subareas.

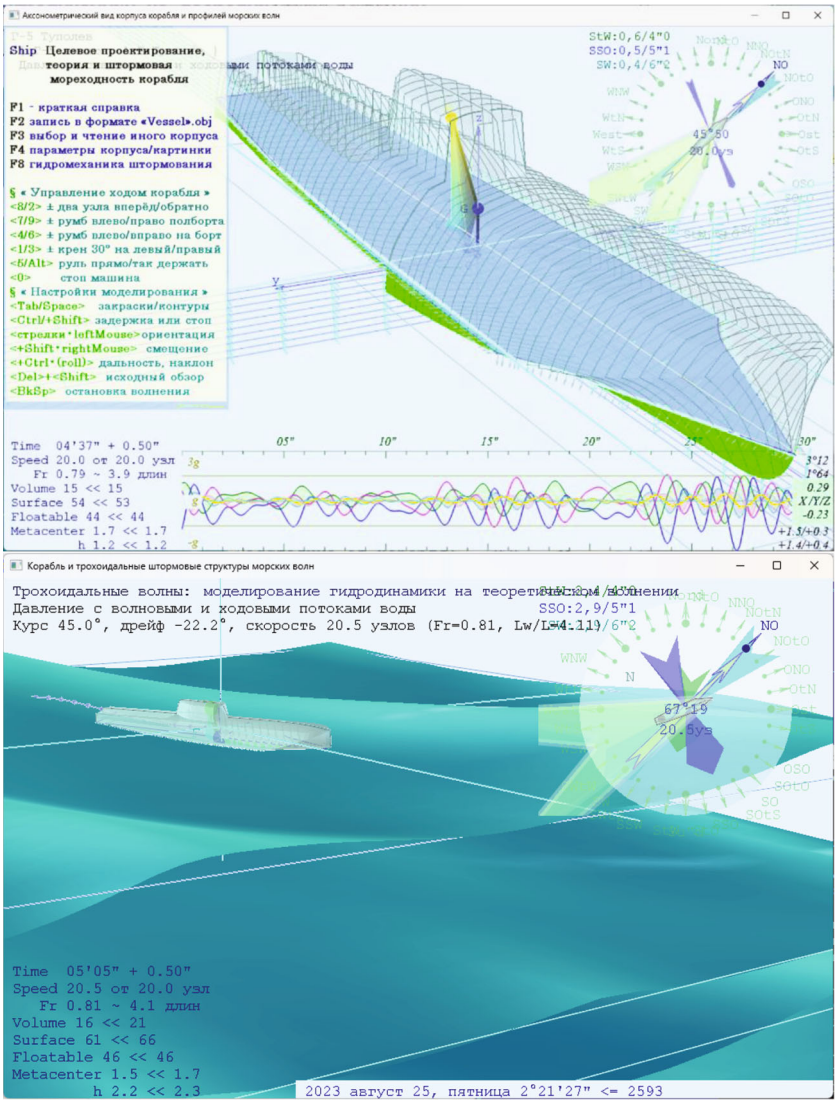


**Fig. 2.** Representation of storm sea and numerical ship state control parameters in a local ship reference frame. On the left prompts for interactive control of simulation, below numerical and graphical information on the hydromechanics of maneuvering are shown. The bottom image shows a maneuvering ship in maritime fixed coordinate system.

## FUNDING

The research was partly supported be St. Petersburg State University's project NP_GZ_2021-3.

## CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

## REFERENCES

1. A. Bogdanov, A. Degtyarev, I. Gankevich, V. Khramushin, and V. Korkhov, "Virtual testbed: Concept and applications," Lect. Notes Comput. Sci. **12254**, 3–17 (2020).
2. A. Bogdanov and V. Khramushin, "Tensor arithmetic, geometric and mathematic principles of fluid mechanics in implementation of direct computational experiments," EPJ Web Conf. **108**, 02013 (2016).
3. A. Degtyarev and V. Khramushin, "Coordinate systems, numerical objects and algorithmic operations of computational experiment in fluid mechanics," EPJ Web Conf. **108**, 02018 (2016).
4. A. Degtyarev, V. Khramushin, and J. Shichkina, "Tensor methodology and computational geometry in direct computational experiments in fluid mechanics," AIP Conf. Proc. **1863**, 11006 (2017).