Integrated software environment for numerical modeling of experiments on tokamaks

D.Yu. Sychugov[1] , I.V. Zotov[1] , S.Yu. Solov'ev[1] , L.I. Vysotsky[1] ,V.E. Lukash[2] , R.R. Khayrutdinov[2] , A.D. Sadykov[3]

[1] Faculty of CMC, Lomonosov Moscow State University, 119991 Moscow, Russia

[2] National Research Centre "Kurchatov Institute", 123182 Moscow, Russia

[3] Institute of Atomic Energy of NNC RK, 071100 Kurchatov, Kazahstan

Introduction. The current development of nuclear fusion devices and planning the experiments on them is unimaginable without mathematical modeling. Hundreds of numerical codes have been developed and successfully used for modeling the most substantial processes in plasma. These programs were written at different times by different research teams, thus they are not quite well suited for integration into a single software system. Hence there exists an essential problem to create an integrated environment for tokamak design and subsequent experiment support. This software system should unite several numerical codes to model the whole life cycle of the tokamak. We have been working on such a system, and this paper describes its current state.

Requirements. The description of requirements to the integrated software system should start from the concept of "discharge scenario". It is the sequence of the following phases: breakdown, increase of the current, separatrix formation (for the devices with divertors), quasi-stationary phase and shutdown. To support the plasma heating experiments all the listed phases have to be numerically modeled. During all the discharge phases one should take into account the possibility of disruption. To prevent the disruption the plasma shape and location has to be thoroughly controlled. This problem leads to the necessity to simulate the diagnostic and control systems. Thus the minimum experiment support module set includes:

1. the codes for evaluation the MHD equilibrium and evolution of plasma, its stability and energy and particle balance;
2. simulators of diagnostic, plasma control and magnetic systems of the device.

These codes have to be embedded into an integral software environment (ISE) with a uniform interface and uniform and (semi-)automatic data exchange system.

Structure. The requirements from the previous section define the set of numerical codes to include into the ISE [1-9]. The system is currently divided into informational part (hosted at plasma-fusion.ru) and functional part (hosted at nfusion.cs.msu.ru). The latter has the client-server architecture. The server application is currently run on a midrange computer, whereas the client application can be run on PCs, laptops and mobile devices. The server part of the ISE:

1. performs the client authorization (data and computational resources access control);
2. executes and interacts with the computational modules;
3. performs the input/output data transformation between the particular module and ISE-wide data formats.

The client part is responsible for drawing the graphical user interface (including graphs, tables and images), interaction with it and data exchange with the server.

The server part of the ISE currently uses the Apache HTTP-server spawning scripts written in the Python programming language. The scripts execute the computational modules which are written in different languages (C, C++, Fortran etc.) and compiled into binary executables. The

client application uses solely web-technologies: the static part of graphical user interface (GUI) is written in HTML and CSS, the interaction and network data exchange logic is implemented in JavaScript. The client-server interaction is done over HTTP; the main data format is XML.
The separation of the ISE into two parts has several advantages.

1. The authors of modules don't have to distribute them in source code; instead, the source code is stored only on developers' machines and on the server. Moreover, the users don't have direct access even to the executables and can invoke them only through specific API commands.
2. The users don't have to be concerned about any software environment setup; it all suffices to a modern web-browser. This argument is even more convincing if the heterogeneity of the numerical codes is taken into account: compilation of most of them requires quite specific environment and set of libraries.
3. It allows for almost unlimited scalability: the current midrange server can be seamlessly replaced with a cluster or even a supercomputer.

Several examples of client applications views are presented in Fig. 1 (MHD equilibria code), Fig. 2 (plasma evolution computation code) and Fig. 3 (reconstruction of plasma boundary code).

New module addition. Currently the addition of new modules into the ISE demands a bit of creativity and sometimes significant effort. The reason is aforementioned heterogeneity of programming languages, data formats and GUI libraries used in original program. Despite that, there are some general recipe of the module addition.

1. The original program is transformed into “ISE-friendly” one, i.e. a console application with unified input and output data formats. The data is usually read from and written to files passed in command line arguments.
2. The server part is extended with new classes and functions. It usually suffices to implementation of an interface with commands of form “set input data”, “run the module” and “get output data”.
3. The client application is extended with the new module specific parts of GUI and with the logic of interaction with the server.

During the ISE development we understood that significant part of the code can be moved to separate libraries and reused. The creation of these libraries substantially simplified the second and third stages of the the system extension. But the adaption of the original program for the ISE (first stage) still remains the most demanding and unautomated.

Automatic data exchange. One of the fundamental task emerging during the development of a integrated system similar to described in this work is organizing data exchange between the computational modules. As we have already mentioned, because of their different origin the modules' input/output data formats differ dramatically. The typical manifestation of the problem is the necessity to transfer the output of one module as input to another one. If the ISE consists of small number of modules or all of them comply to the same standard, the problem may not be obvious. But as the number of modules and their diversity grow the difficulties become explicit.

The only possible solution is to have a single “omniscient” format that can store all the data produced and consumed by all the codes of the ISE. Indeed, unless one has such a universal format, the only way to solve the data exchange problem is to write a transformation tool for each pair of modules (i.e. number of tools is proportional to n^2, where n is the number of modules). It's absolutely infeasible for a large ISE especially when the transformation process is only semi-automatic (because of the difference in input information necessary for different modules). On the contrary, having one universal data format requires writing only $2n$ transformation tools, at the same time stimulating the developers of new numerical codes to use this format.

The main requirements for the universal data format are flexibility and extensibility, which implies it has to be hierarchical (e.g., it should be possible to store entries with the same name if they correspond to distinct modules). On the other hand, there are some concepts common to the domain and they must correspond to unique entries in the format. In the case of tokamak devices the examples of such concepts are: the number of poloidal field coils, their location, size and number of turns; the geometry and material of the first wall; the number of probes, their type, location and precision; the full plasma current, its beta, inductance etc.

Conclusion. The integrated software environment consists of:

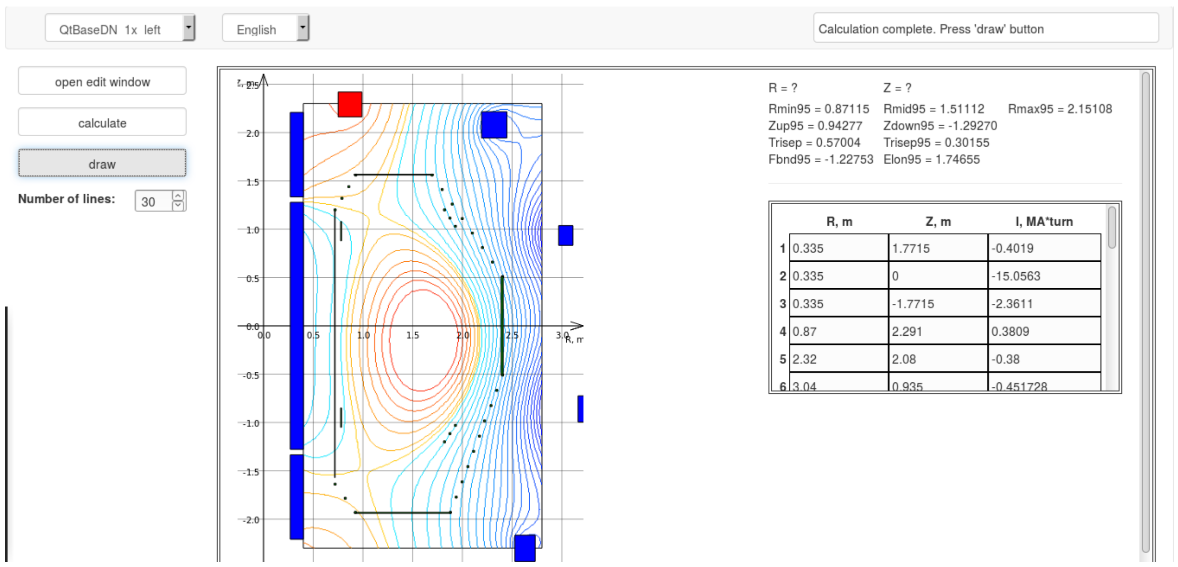
1. computational modules modeling processes in plasma;
2. applications allowing remote multi-user access and work with these modules;
3. means of data exchange between the modules;
4. informational resource.

The proposed resource is open for public use. Despite the system's narrow focus, its basic principles can be applied for the development of virtual analogues of other complex systems.

This work was supported by RFBR grants No. 17-07-00544-a, 17-07-00883-a.

References

1. Sadykov A.D., Sychugov D.Yu., Shapovalov G.V., Chektybaev B.Zh., Skakov M.K. and Gasilov N.A. 2015 Nuclear Fusion, **55**, N. 4, 55043017.
2. Sychugov D.Yu. 2008 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, No. 4, p.85.
3. Sychugov D.Yu., Amelin V.V., Gasilov N.A. 2010 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **33**, No. 3, p.46.
4. Sadykov A.D., Shapovalov G.V., Chectybaev B., Sychugov D.Yu., Gasilov N.A. 2013 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **36**, No. 4, p.94.
5. Khayrutdinov R.R., Lukash V.E. 2009 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **32**, No. 3, p.57.
6. Khayrutdinov R.R., Lukash V.E. 2010 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **33**, No. 3, p.50.
7. Lukash V.E., Khayrutdinov R.R. 2011 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **34**, No. 3, p.88.
8. Lukash V.E., Khayrutdinov R.R. 2012 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **35**, No. 4, p.93.
9. Zotov I.V., Belov A.G. 2014 Problems of Atomic Science and Technology. Ser. Thermonuclear Fusion, **37**, No. 1, p.97.



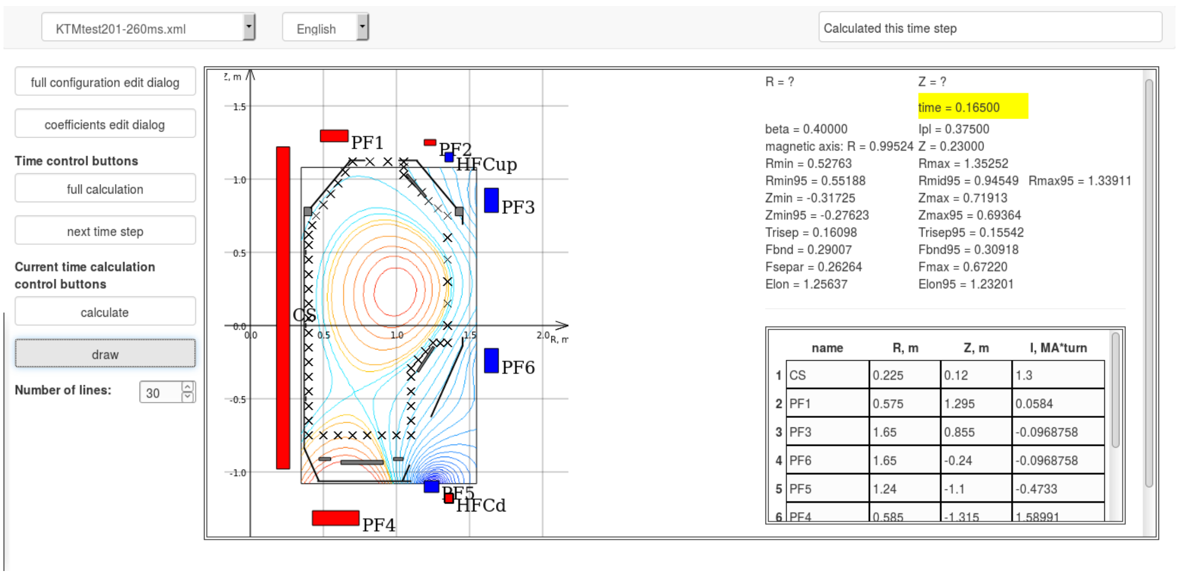
Calculation complete. Press 'draw' button

R = ? Z = ?

Rmin95 = 0.87115 Rmid95 = 1.51112 Rmax95 = 2.15108
Zup95 = 0.94277 Zdown95 = -1.29270
Trisep = 0.57004 Trisep95 = 0.30155
Fbnd95 = -1.22753 Elon95 = 1.74655

	R, m	Z, m	I, MA*turn
1	0.335	1.7715	-0.4019
2	0.335	0	-15.0563
3	0.335	-1.7715	-2.3611
4	0.87	2.291	0.3809
5	2.32	2.08	-0.38
6	3.04	0.935	-0.451728

Fig. 1: *Interface of Tokameq module (MHD equilibria). The double-null magnetic configuration of the T-15MD tokamak at the time moment t=2.5s.* I_p = 1.85 MA, R_{mid95} = 1.51 m, a=0.64 m, k = 1.746, β_p = 0.18



Calculated this time step

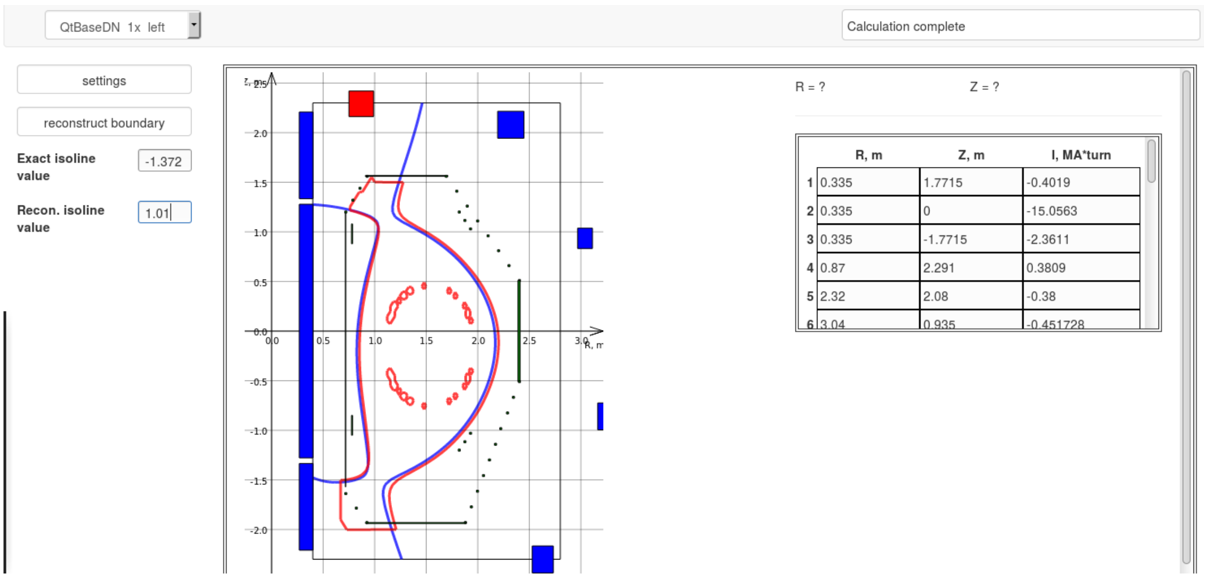
R = ? Z = ?

time = 0.16500

```
beta = 0.40000                      Ipl = 0.37500
magnetic axis: R = 0.99524 Z = 0.23000
Rmin = 0.52763                      Rmax = 1.35252
Rmin95 = 0.55188                    Rmid95 = 0.94549   Rmax95 = 1.33911
Zmin = -0.31725                     Zmax = 0.71913
Zmin95 = -0.27623                   Zmax95 = 0.69364
Trisep = 0.16098                    Trisep95 = 0.15542
Fbnd = 0.29007                      Fbnd95 = 0.30918
Fsepar = 0.26264                    Fmax = 0.67220
Elon = 1.25637                      Elon95 = 1.23201
```

	name	R, m	Z, m	I, MA*turn
1	CS	0.225	0.12	1.3
2	PF1	0.575	1.295	0.0584
3	PF3	1.65	0.855	-0.0968758
4	PF6	1.65	-0.24	-0.0968758
5	PF5	1.24	-1.1	-0.4733
6	PF4	0.585	-1.315	1.58991

Fig. 2: *Interface of TokScen module (plasma evolution). The magnetic configuration of the KTM tokamak at the time moment t=0.165 s. I_p = 0.375 MA, R_{mid95} = 0.945 m, a=0.41 m, k = 1.256, β_p = 0.4.*



Calculation complete

R = ? Z = ?

	R, m	Z, m	I, MA*turn
1	0.335	1.7715	-0.4019
2	0.335	0	-15.0563
3	0.335	-1.7715	-2.3611
4	0.87	2.291	0.3809
5	2.32	2.08	-0.38
6	3.04	0.935	-0.451728

Fig.3: *Interface of* RPB *module (reconstruction of plasma boundary from magnetic measurements). Reconstruction of the separatrix in the T-15MD tokamak. Time moment t=0.76 s. Measurements error 1 %, 44 two-component magnetic sensors . Blue and red lines – exact and reconstructed separatrix.*