31st International Conference Nuclear Energy for New Europe
NENE 2022 September 12-15 PORTOROŽ SLOVENIA

Towards Refactoring Of The TOKES Tokamak Plasma Transient Code

Leon Bogdanović[a], Sergey Pestchanyi[b], Leon Kos[a]
[a]Faculty of Mechanical Engineering, University of Ljubljana
Aškerčeva 6
1000 Ljubljana, Slovenia

[b]Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen, Germany
leon.bogdanovic@lecad.fs.uni-lj.si, serguei.pestchanyi@kit.edu, leon.kos@lecad.fs.uni-lj.si

ABSTRACT

The TOKES ("Tokamak Equilibrium and Surfaces") code simulates numerically dynamics of the thermonuclear deuterium-tritium (D-T) plasma in ITER core, in scrape-off layer (SOL) calculates heat flux to the tokamak walls and heat transport inside the solid walls. It takes into account phase transitions of the wall material, i.e., tungsten (W), including vaporization. After vaporization start, TOKES simulates dynamics of vaporized W in vacuum vessel, its ionization and W-D-T plasma dynamics, including photonic radiation. The code features a numerical meshing out to all wall surfaces with the possibility of spatially variable grid resolution on the mesh. It includes standard surface interactions such as sputtering, but also surface vaporization and, importantly, vapour shielding. TOKES has been extensively used for several years in specific ITER studies, covering in particular simulations of disruption mitigation by massive gas and shattered pellet injection and the impact of heat fluxes due to non-mitigated disruptions and ELMs (Edge-Localized Modes), including vapour shielding effects. The code is also being applied to JET-ILW (ITER-like wall) and to the EU DEMO PFC design activities considering, in particular the impact of disruptions and sacrificial limiters. Compared with more conventional boundary codes, it has the advantage of rapid run times, permitting extensive parametric studies even at the reactor scale like required in the DEMO design phase. Developed over almost two decades at KIT, the source code of TOKES in Pascal is still compiled in Delphi, a commercial Integrated Development Environment (IDE), under Windows on single machines. For the benefit of TOKES preservation and availability to the fusion community, refactoring of its source code to an open-source solution is needed. This paper presents the progress on TOKES code refactoring under Free Pascal on Linux as well as some examples of simulation results visualization in a prototype Graphical User Interface (GUI).

1 INTRODUCTION

The TOKES code was developed in the last two decades at Karlsruhe Institute of Technology (KIT) aiming at an integrated simulation of plasma equilibrium states and surface processes in tokamaks, what its acronym actually reflects: "TOKamak Equilibrium and Surfaces" [1]. There are several publications that describe the features of TOKES and its capabilities of applications for the future tokamaks ITER and DEMO, e.g., simulation of plasma shielding effect during disruption, simulation of gas and pellet injection for disruption

mitigation and simulation of the impact of unmitigated disruptions on sub-dome structure in ITER operation ([2], [3], [4]), as well as validation of vapor shield simulations against plasma gun experiments [5]. To secure the continued deployment of the TOKES code and its availability to a wider fusion community, refactoring from commercial Delphi Pascal [6] to an open-source solution, e.g., Free Pascal, is essential.

2 ORIGINAL TOKES SOURCE CODE AND PROGRAM

2.1 Basic features of TOKES

For TOKES the toroidal symmetry is assumed, which its models keep as a basic feature of the tokamak principle. The code simulates plasma evolution in time *t*. The plasma stays in some slowly changing equilibrium with the confining magnetic field. The evolution of the confinement equilibrium occurs due to internal dissipative processes in the plasma, such as the diffusion across the field lines, and variations of external parameters, such as the Deuterium-Tritium (DT) inflow. The intermediate states are described with diverse functions of cylindrical coordinates, *r* and *z*, where *z* is the coordinate along the axis of toroidal symmetry and *r* the distance from the *z*-axis. Another important frame in TOKES is magnetic flux coordinates. At *t* approaching infinity, this 2D system may eventually reach a steady equilibrium state. Such final state is not essential, and TOKES can simulate in many details non-steady tokamak processes. The simulation involves dynamical changes of plasma shape and actual electric currents both in plasma and in external coils of poloidal magnetic field (PF coils) that control the stability of the whole configuration in respect to toroidally symmetric modes. The virtual tokamak of TOKES assumes a lexicon of real physical processes that can naturally be used. In this context, TOKES steadily "creates" hot DT plasma by fuelling a confinement region (the core) inside the tokamak vessel with DT atoms. The fusion reaction produces helium ions, fast neutrons, and heat. Impurities from the vessel wall contaminate the plasma. The code simulates impurity ions in the whole vessel as it does for the main plasma components, with a common multi-fluid plasma model for all species. The plasma diffuses across nested magnetic surfaces of the trap to the periphery of the core. Then it passes a narrow scrape-off layer (SOL), i.e., the core's shell, and dumps along magnetic field lines onto vessel walls. The wall surface also absorbs the electromagnetic radiation from the plasma, neutral atoms and the neutrons as well. The surface responses to the impact, backscattering impacting ions and emitting sputtered atoms and, if the load is very large, the vapour of wall material. The erosion products, which are emitted atoms of wall materials, freely propagate in the vessel before having been ionized in SOL and the core. For more efficient simulations of fast transient processes, like ELMs and disruptions, TOKES uses an assumption that the magnetic configuration does not change during one or a few milliseconds of the transient. In the original mode with magnetic configuration evolution the code calculates the evolution during the ramp up until reaching a steady state [1].

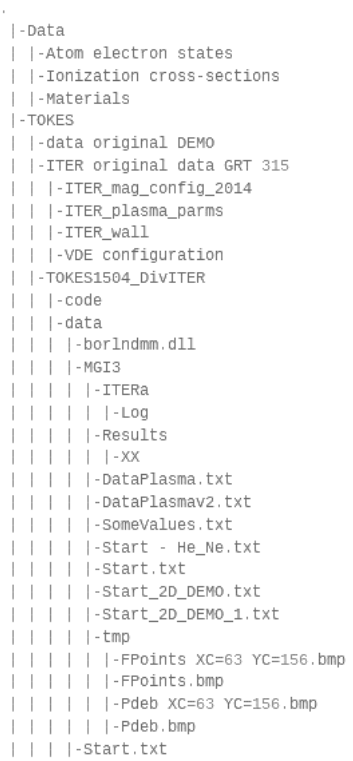


Figure 1: TOKES source code.

2.2 Structure of the original TOKES source code in Delphi Pascal

The original source code of TOKES is contained in the code subdirectory of TOKES/TOKES1504_DivITER, as displayed in the tree structure on Figure 1. It consists of Delphi units and forms for the Graphical User Interface (GUI) of the application in MS Windows (Table 1) and units of Pascal code (*.pas), which are basically the core for TOKES calculations (Table 2). This code is compiled and built with Delphi into an executable. The executable needs input data from the Data directory as well as from the data subdirectory. The device data is contained in the ITERa subdirectory of data/MGI3.

Table 1: TOKES GUI units

Unit name	**Description**
GraphicU	functions and routines for GUI operations
UInterfa	code for main form of TOKES
ShowU	code for secondary (showing) form of TOKES

Table 2: TOKES core units

Unit name	**Description**
BaseLibU	basic data types
ComplexU	complex data types, elementary functions of complex variables, Fourier transforms
FunctioU	function definitions
GeometrU	2D and 3D geometry, circles, linear algebraic equations
GraphU	graph algorithm
ListU	list operations
PhysicsU	plasma physics
UAtoms	database of physical constants of the atoms
UCrosdif	plasma diffusive transport across magnetic surfaces
UField	calculation of toroidally symmetric magnetic field both inside and outside the tokamak vessel
UGround	main unit of TOKES
ULongit	longitudinal transport of magnetized plasma
UPlasma	plasma implementation
UProcess	process
URays	transport of neutral rays in the vessel
UTask_MGI3	user task in TOKES
UVessel	triangle meshes and magnetic flux coordinates inside tokamak vessel
UWall	wall definition and coordinates as well as processes inside the wall

2.3 Performing simulations with the original TOKES application

After starting the executable and clicking on the Task button at the bottom of the window, the Start.txt input file is displayed (Figure 2). This file contains the input parameters for the TOKES simulation and the output directory definition where the results are stored. The user can change these parameters and save them. By clicking on the Start button, TOKES is initialized with the chosen parameters. The simulation is started by selecting Tools and then Run in the menu bar. The user can observe the simulation running in the main window, i.e., the current time step and total time of simulation, some parameters shown on the graphs and the tokamak vessel displayed (Figure 3). The simulation can be stopped by selecting Tools and then Stop in the menu bar or by clicking on the total time label. From logs, which are automatically produced, simulations can be resumed if they have been previously stopped or the executable crashed for any reason.

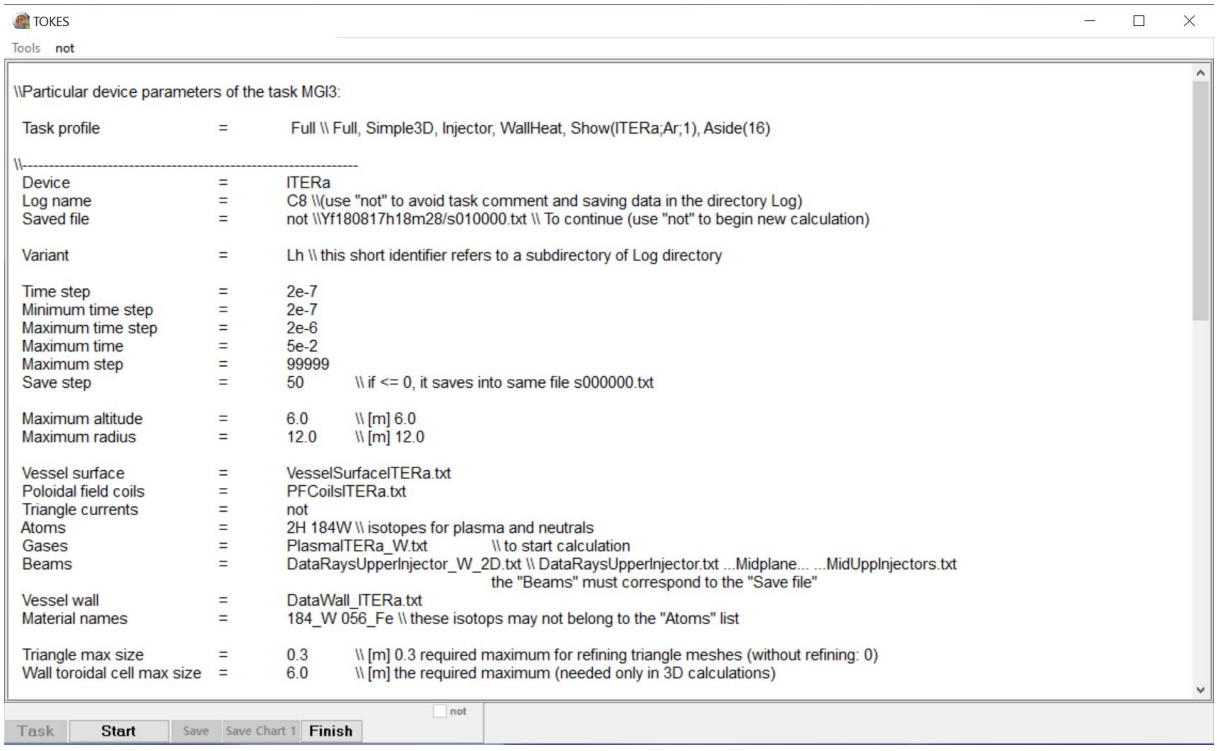


Figure 2: Input parameters for a TOKES simulation.

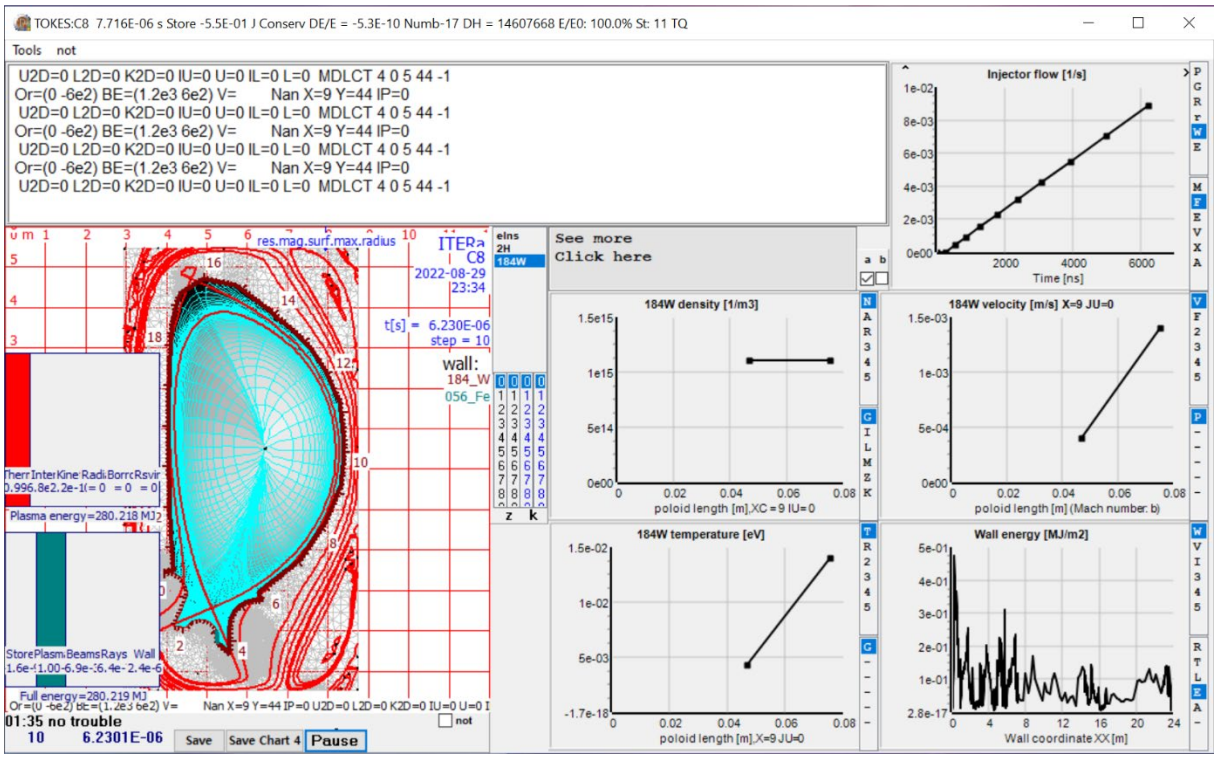


Figure 3: A running TOKES simulation.

2.4 Analysing TOKES simulation results

A separate application, the vw2DFMC viewer, which is also coded and compiled in Delphi, is used to analyse TOKES simulation results. Results are stored in *.P2DT files, e.g., at every 50th time step, if Save step = 50 is set in Start.txt. The user can open either single or multiple results files, depending on the analysis performed with the viewer. Generally, single files are used for displaying 2D plots of the parameter of interest in the tokamak vessel region, while multiple plots are used for time dependent analysis or distributions along the wall for the chosen parameter.

3 REFACTORING OF THE TOKES CORE AND VIEWER SOURCE CODE

By definition, code refactoring is the process of restructuring existing computer code without changing its external behaviour. In this regard the core of TOKES in Pascal for calculations/simulations will be preserved, while the GUI will be replaced according to the open-source solution used. The functionality of the vw2DFMC viewer, which is currently a separate program, is planned to be included in a single refactored TOKES program/application for simulations and visualizations. In this paper first attempts at refactoring of current separate programs in Lazarus are presented. Lazarus is an open-source Delphi compatible cross-platform Integrated Development Environment (IDE) for Rapid Application Development. It has variety of components ready for use and a graphical form designer to easily create complex graphical user interfaces. It runs on Windows, macOS, Linux and many other platforms [7]. Lazarus uses Free Pascal as its language which is an Object Pascal dialect [8].

3.1 First attempt at refactoring of the TOKES core

While Lazarus is a Delphi compatible IDE, automatic conversion from a Delphi project to a Lazarus project seldom works. This is also the case for the TOKES source code, hence the best approach for porting the TOKES core from Delphi is to create a new project in Lazarus with appropriate GUI forms and then include the Pascal units (see Table 2) into it. Also, some differences in linking and compiling the source code exist, e.g., in Lazarus the included files are set under Compiler Options > Paths > Include files, while in Delphi under Delphi Compiler > Search path. When porting Delphi source code to Lazarus one should also set Delphi (-MDelphi) under Compiler Options > Parsing > Syntax mode.

The GUI of the TOKES core application in Lazarus is shown on Figure 4. Its structure follows the structure of the original TOKES application (Figure 2), although many graphical elements (display of the tokamak vessel and graphs during the simulation) are omitted.

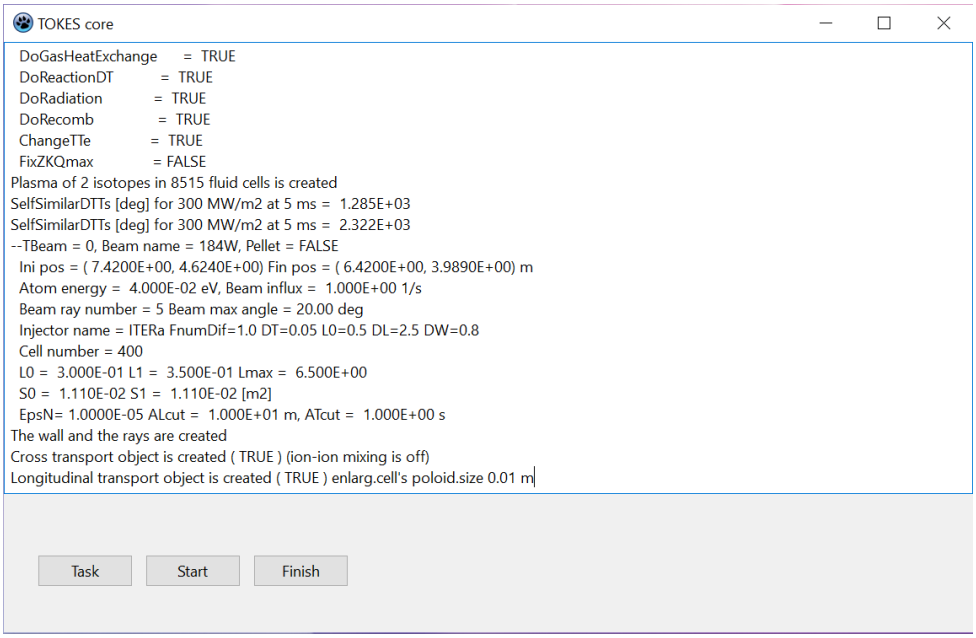


Figure 4: TOKES core GUI in Lazarus.

Currently, only the start of the TOKES simulation in the TOKES core application in Lazarus is implemented, i.e., the application produces the result file at the start of the simulation ($t = 0$). Simulation runs will be implemented in a future version.

3.2 First attempt at refactoring of the TOKES viewer

The vw2DFMC viewer was refactored with the same approach as the TOKES core. Figure 5 shows the GUI of the TOKES viewer. Currently, only the 2D plot type menu is implemented.

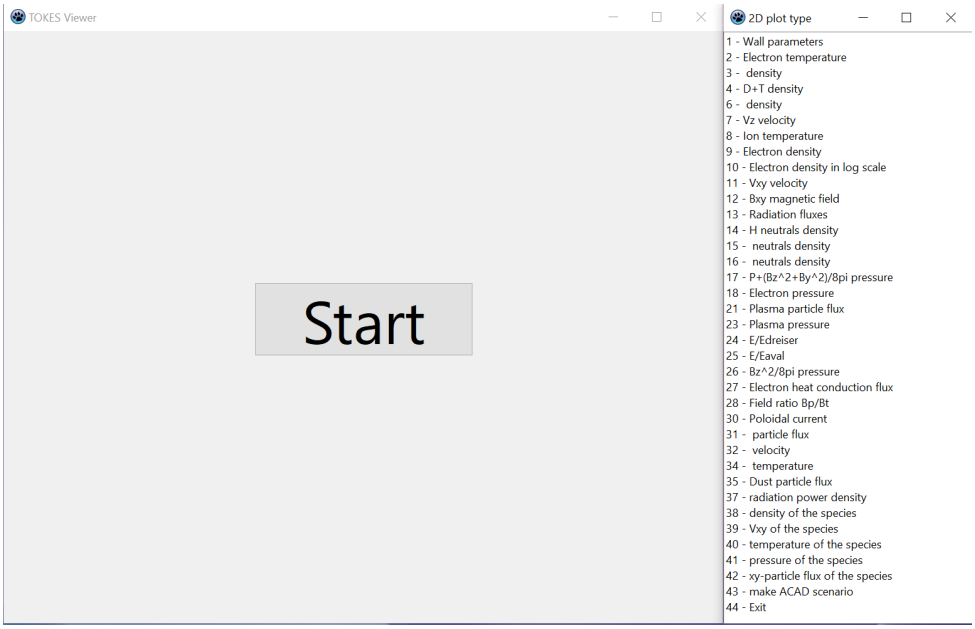
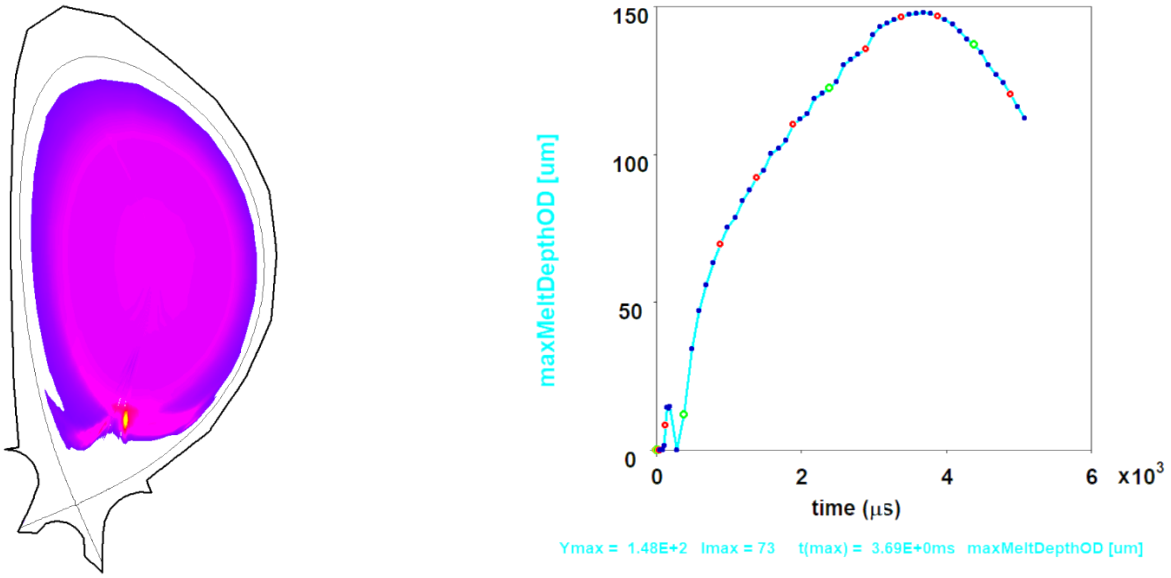


Figure 5: TOKES viewer GUI in Lazarus.

Selecting 4 – D+T density from the menu for a previously chosen result input file produces the result shown on Figure 6 (a). In a future version of the TOKES viewer, also time dependent visualizations will be implemented (Figure 6 (b)).



(a) (b)

Figure 6: TOKES results of unmitigated disruptions for 350 MJ initial plasma energy in ITER. (a) D+T density in the tokamak vessel region at end of simulation. (b) Time dependent maximum melt depth in the tokamak wall.

4 CONCLUSION

First attempts at refactoring of the TOKES Tokamak Plasma Transient Code with Lazarus, an open-source solution, were presented. The first versions of the TOKES core and

TOKES viewer, although far from being complete and hence not really a replacement for the original codes, show that TOKES can be ported from Delphi Pascal to Free Pascal. Future versions are planned which will include all the capabilities of the original codes. While the Lazarus IDE is a viable alternative to the commercial Delphi RAD for porting the TOKES source code to other platforms, such as Linux, also other solutions or approaches will be investigated. One such approach is to retain the core units (backend) in Pascal, while developing the interface (frontend) with Python graphics libraries, such as PyQt or PySide.

ACKNOWLEDGMENTS

The author LB is supported by EUROfusion Engineering Grant EEG21-19 "Refactoring and deployment of the TOKES tokamak plasma transient code".

REFERENCES

[1] I. S. Landman, "Tokamak code TOKES. Models and implementation", Forschungszentrum Karlsruhe in der Helmholtz-Gemeinschaft Wissenschaftliche Berichte, FZKA-7496 (September 2009).

[2] S. Pestchanyi, R. Pitts, M. Lehnen, "Simulation of divertor targets shielding during transients in ITER", Fusion Engineering and Design, Vol. 109-111, Part A, 2016, pp. 141-145, ISSN 0920-3796, https://doi.org/10.1016/j.fusengdes.2016.02.105.

[3] S. Pestchanyi, et al., "TOKES simulations to compare gas and pellet injection for disruption mitigation in ITER", Fusion Engineering and Design, Vol. 136, Part A, 2018, pp. 29-33, ISSN 0920-3796, https://doi.org/10.1016/j.fusengdes.2017.12.016.

[4] S. Pestchanyi, et al., "Simulations of Energy Loads and their Mitigation during Disruptions and Runaway Electron Formation in ITER, Part-1: TOKES Simulations", Final Report of Work incl. Final Conclusions & Recommendations (incl. work on VDEs & Vapour Shielding), KIT Document Ref. No. MOD-PEW-145842-RD-2D04 (November 2020).

[5] S. Pestchanyi, R.A. Pitts, V. Safronov, "Validation of TOKES vapor shield simulations against experiments in the 2MK-200 facility", Fusion Engineering and Design, Vol. 124, 2017, pp. 401-404, ISSN 0920-3796, https://doi.org/10.1016/j.fusengdes.2017.02.048.

[6] Delphi: a software product that uses the Delphi dialect of the Object Pascal programming language, https://www.embarcadero.com/products/delphi, 2022. Accessed: 2022-08-16.

[7] Lazarus: a Delphi compatible cross-platform IDE for Rapid Application Development, https://www.lazarus-ide.org/, 2022. Accessed: 2022-08-16.

[8] Free Pascal: an open source Pascal compiler, https://www.freepascal.org/, 2022. Accessed: 2022-08-16.