30th NENE 2021
September 6 - 9
BLED-SLOVENIA
International Conference Nuclear Energy for New Europe

L2G PFC Heat Loads and Field-line Tracing in the SMITER Framework

Leon Bogdanović, Gregor Simič, Leon Kos
Faculty of Mechanical Engineering, University of Ljubljana
Aškerčeva 6
1000 Ljubljana, Slovenia
leon.bogdanovic@lecad.fs.uni-lj.si, gregor.simic@lecad.fs.uni-lj.si, leon.kos@lecad.fs.uni-lj.si

ABSTRACT

Integrated within the SMITER Framework, L2G is a new software module for magnetic field-line tracing and heat loads mapping on tokamak plasma-facing components (PFC). The core of the module for field-line tracing is written in C++ and makes use of specialized libraries such as ALGLIB, RKF45 and Intel Embree for cubic B-splines interpolation of magnetic equilibrium data, numerical integration of field-lines and ray (field-line segments) casting with PFC triangle meshes, respectively. The interface, Application Programming Interface (API) with the Graphical User Interface (GUI), is written in Python and wraps the C++ core through Cython to give the code C-like overall performance. The GUI provides the input of user-defined parameters and options for running field-line tracing cases. Target and shadow geometries can be imported from the SMITER cases, as well as the magnetic equilibrium files. Field-line tracing is done inversely, i.e., the field-lines start on the target geometry and are traced until an intersection with the adjacent shadow geometry is found or the end of the integration interval is reached. Field-lines with no intersections imply wetted starting triangles on which heat loads are computed based on the single exponential plasma profile formula. Other relevant tracing results are also computed such as field-line connection lengths and incident angles. The results can be visualized in the ParaView module of the SMITER Framework. Optionally, field-lines can also be plotted. Further speed-up of field-line tracing can be achieved through OpenMP parallelization. This paper presents benchmarking of the L2G module against existing field-line tracing codes (SMARDDA and PFCFLUX) and provides results of tests on PFC meshes of 10 millions triangles.

1 INTRODUCTION

One of the main approaches for the design of plasma-facing components (PFC) of a tokamak type fusion experimental reactor is plasma magnetic field-line tracing. Such an approach can define the heat load of the Scrape-Off Layer (SOL) plasma on PFC surfaces and consequently the definition of their optimal shape. Standard codes for magnetic field-line tracing are not optimized for fast simulations in a huge triangle mesh. Intersection tests of millions of field-lines with a PFC mesh with more than 10 millions triangles is computationally very intensive and time consuming, hence a need for parallel codes with state-of-the-art accelerated intersection algorithms arise. Besides being fast such codes should also be easily integrable in open source Computer-Aided Design (CAD) environments.

1.1 Field-line tracing

Field-lines in an axisymmetric magnetic field of a tokamak plasma can be described with a standard equation in cylindrical coordinates (R, ϕ, Z):

$$\frac{\mathrm{d}R}{B_\mathrm{R}} = \frac{\mathrm{d}Z}{B_\mathrm{Z}} = \frac{R\mathrm{d}\phi}{B_\mathrm{T}} \,, \tag{1}$$

where B_R, B_T in B_Z are the radial, toroidal and vertical component of the magnetic field, respectively. By expressing these components in terms of the poloidal flux function $\Psi(R, Z)$ and the flux function $F(\Psi)$:

$$\begin{aligned} B_\mathrm{R} &= -\frac{1}{R}\frac{\partial\Psi(R,Z)}{\partial Z} \,, \\ B_\mathrm{Z} &= \frac{1}{R}\frac{\partial\Psi(R,Z)}{\partial R} \,, \\ B_\mathrm{T} &= \frac{F(\Psi)}{R} \,, \end{aligned} \tag{2}$$

one can obtain the following Ordinary Differential Equations (ODE) system for the field-lines:

$$\begin{aligned} \frac{\mathrm{d}R}{\mathrm{d}\phi} &= -\frac{\partial\Psi(R,Z)}{\partial Z}\frac{R}{F(\Psi)} \,, \\ \frac{\mathrm{d}Z}{\mathrm{d}\phi} &= \frac{\partial\Psi(R,Z)}{\partial R}\frac{R}{F(\Psi)} \,. \end{aligned} \tag{3}$$

This ODE system is solved as an initial value problem, with the initial value set in the barycenter of the triangle from which the field-line is traced backwards. The numerical integration of (3) is performed with the Runge-Kutta-Fehlberg (RKF) algorithm which offers accuracy in respect to computation efficacy. At every integration step, which can be adaptive, the interpolation of $\Psi(R, Z)$ and its derivatives has to be performed, as well as the interpolation of $F(\Psi)$, since their values are generally given as a magnetic equilibrium input on a low resolution grid. A suitable interpolation method is based on de Boor's algorithm [1].

1.2 Heat load model

The heat flux from the SOL is deposited on the target PFC geometry. To determine if a triangle on the target suffers heat loads, a field-line from its barycenter must be traced backwards to the SOL. If the field-line hits the adjacent (shadow) PFC geometry, then the triangle associated to it is considered shadowed, hence the heat load is equal to 0. On the other hand, if the field-line doesn't intersect the shadow geometry, then it's assumed to start from the SOL and the triangle associated to it is considered wetted, hence the heat load can be calculated. The calculation assumes the mapping of SOL heat flux flowing parallel to the magnetic field-lines onto PFC surfaces. The heat flux is assumed to fall off exponentially in the radial direction into the SOL from the primary separatrix (diverted configurations) or from the last closed flux surface (LCFS) in limiter configurations [2]. In any case heat load q_PFC is proportional to the dot product of the magnetic field vector and surface normal $\mathbf{B} \cdot \mathbf{n}$:

$$q_\mathrm{PFC}(\Psi) = \frac{P_\mathrm{SOL}}{4\pi R_\mathrm{m}\lambda_\mathrm{q}} B_\mathrm{pm}\mathbf{B}\cdot\mathbf{n}\exp\left(-\frac{\Psi-\Psi_\mathrm{m}}{R_\mathrm{m}B_\mathrm{pm}\lambda_\mathrm{q}}\right) \,, \tag{4}$$

where P_{SOL}, R_{m}, B_{pm}, Ψ_{m} and λ_{q} are the power flowing into the SOL across the separatrix/LCFS (assumed to enter the SOL at the outer midplane), the radius at the outer midplane, the poloidal component of the magnetic vector at the outer midplane, the poloidal flux function value at the outer midplane and the characteristic width of the exponential decay, respectively. It should be stressed that these parameters are obtained from a magnetic equilibrium input and are constant for the whole grid. The only parameters that vary are the magnetic field vector $\mathbf{B}$ and the poloidal flux function Ψ, both calculated in the barycenter of the triangle, as well as the surface normal $\mathbf{n}$ of the triangle, which is a geometrical parameter.

2 L2G MODULE DESCRIPTION

L2G was developed as a module for field-line tracing in the SMITER framework [2] based on the open source environment SALOME [3]. It consists of three main components:

- **L2G_cpp**: C++ core for field-line tracing
- **L2G_py**: Python/Cython interface
- **L2G API**: Application Programming Interface (API) with the Graphical User Interface (GUI) input dialog within the SMITER framework

2.1 Core for field-line tracing

The core for field-line tracing executes the algorithm shown on Figure 1 for every field-line starting in the barycenter of a triangle in the PFC target geometry. Field-line tracing is done by solving Eq. 3 according to the description in Section 1.1. After every integration step an intersection test of the calculated field-line segment with the shadow geometry stored in the Bounding Volume Hierarchy (BVH) is performed. If an intersection is found, then it's stored in the intersection mask as `true` and tracing is stopped. If an intersection is not found after a maximum number of steps, then it's stored in the intersection mask as `false`. Based on the values in the intersection mask heat loads on the target are calculated according to Eq. 4. Both the intersection mask and the heat load values can be stored on the target geometry as a field or stored in some other format.

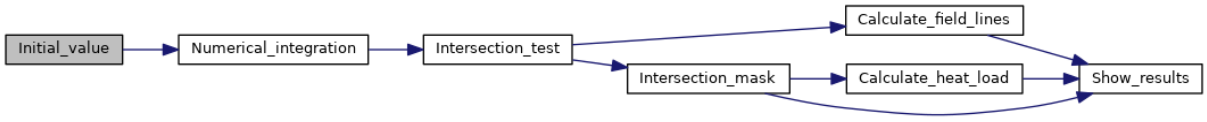


Figure 1: Field-line tracing algorithm.

The core is written in C++ and contains more than 1500 lines of code. For interpolation of $\Psi(R, Z)$, its derivatives and $F(\Psi)$ the C++ library **ALGLIB** [5] is used, while numerical integration is done with the **RKF45** C++ library [6] . For storing the shadow PFC geometry in BVH and for intersection tests the state-of-the-art ray tracing library **Intel Embree** [7] is used.

2.2 Python/Cython interface

Figure 2 shows preprocessing for field-line tracing. This part is generally done through the interface. It comprises the loading of the PFC geometries and magnetic equilibrium data,

as well as the preparation of the cubic B-splines for magnetic parameters interpolation. All the data is available to the core for field-line tracing.

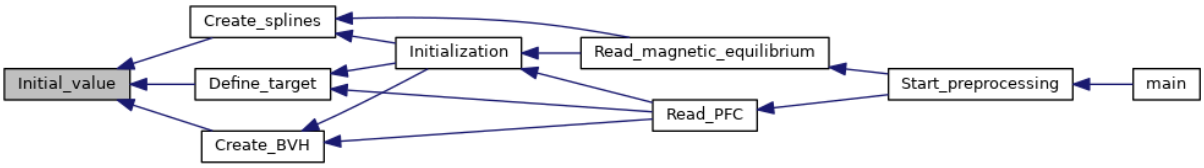


Figure 2: Preprocessing for field-line tracing.

The interface is written in Python mainly for the reason of easier integration into the SMITER framework. For C-like performance of the Python code as well as for wrapping the C++ core in Python, Cython is used. Since every field-line is independent, parallelization paradigms can be used to speed up field-line tracing. L2G uses OpenMP for parallelization. In total the interface (L2G_py + API) contains more than 3600 lines of code.

2.3 L2G API

The integration of the L2G_cpp core and the L2G_py interface of the L2G module into the SMITER framework is done through an Application Programming Interface (API) coded in Python. The API serves mainly as a layer between the L2G module and SMITER, but also defines a GUI dialog for the input of cases to L2G. Figure 3 shows an example of use of the module L2G. Through the L2G Dialog (shown on the left) the user can load SMITER cases for field-line tracing with L2G. Options and parameters can be selected in the tabs of the dialog, while in the tab ”Case log” the user can observe the execution of field-line tracing. After the computation is finished, the results are automatically displayed in the ”ParaView” window (shown on the right) in SMITER. In the example on the figure field-line tracing is done for a PFC target geometry with one panel (in blue) and a shadow geometry with 8 panels (in red). On the target heat load results and field-lines for some selected triangles (in pink) are shown. One can observe that the field-lines emerging from the shadowed triangles (zero heat load q) intersect the adjacent shadow geometry (in red), while the field-lines emerging from the wetted triangles (non-zero heat load q) pass by the shadow geometry indicating their origin from the SOL.

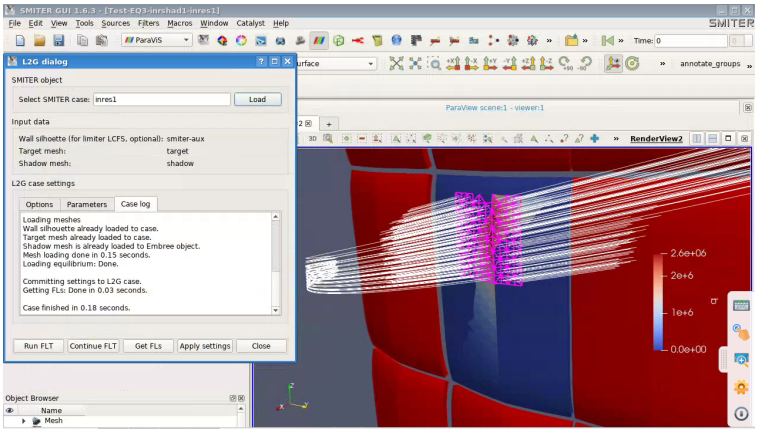


Figure 3: Use of the module L2G for field-line tracing.

3 L2G BENCHMARKING RESULTS

Benchmarking of the L2G module will be presented on the case of an ITER inner wall limiter start-up equilibrium on FWP4 already examined with PFCFLUX [4] and SMARDDA (in SMITER) [2]. Additionally, results of the performance tests on PFC meshes of 10 millions triangles with the use of OpenMP parallelization will be presented.

3.1 ITER NF55 FWP4 case

The **NF55** case is a study of heat loads and incident angles of field-lines on the panels of the ITER tokamak FWP4 first wall [4]. The magnetic equilibrium (ITER `16_97s`) is of an outer wall limiter (OWL) type affecting panels No. 3, 4 and 5 of the FWP4 first wall. The shadow geometry consists of panels No. 1-7 of the FWP4 first wall. Figure 4 shows the last closed flux surface LCFS (blue curve) of the magnetic equilibrium along with the geometry of the limiter of the ITER tokamak (red curve). The target and shadow geometries are shown on Figure 5.

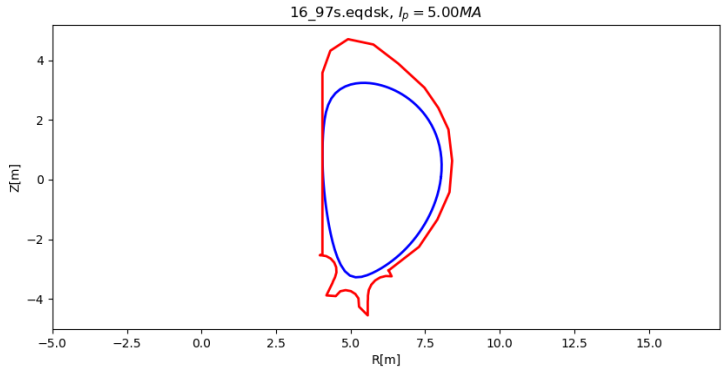
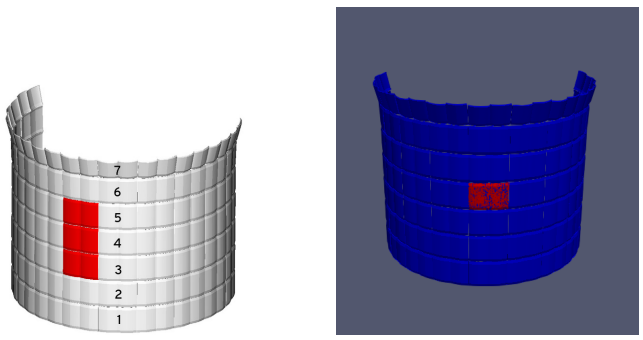


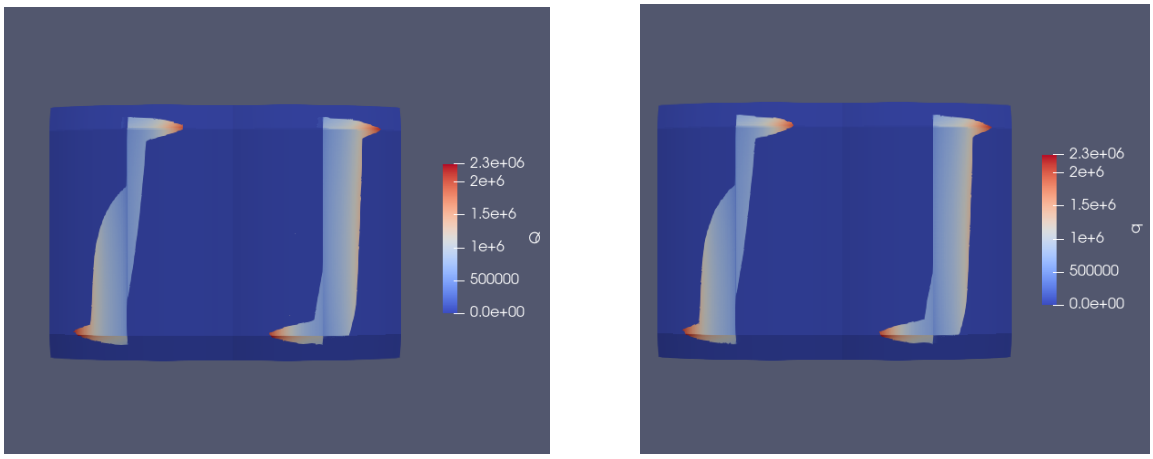
Figure 4: LCFS (blue curve) of the magnetic equilibrium `16_97s` with the given plasma current and the limiter geometry of the ITER tokamak (red curve) for the NF55 case.



(a) **(b)**

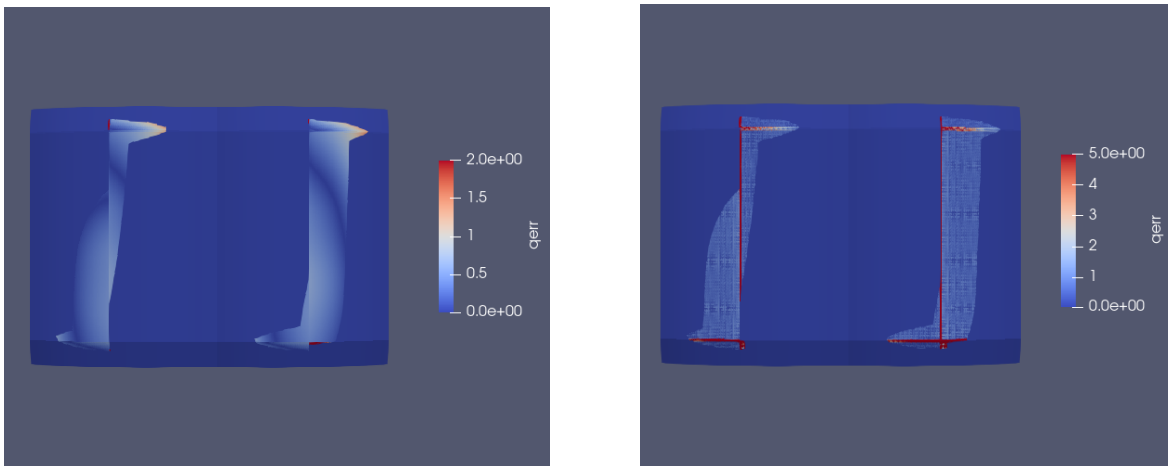
Figure 5: PFC geometry of the NF55 case. (a) Panels No. 1-7 of the FWP4 first wall are the shadow geometry (in white), panels No. 3, 4 and 5 are the target geometry (in red) [4]. (b) Target (in red) and shadow (in blue) mesh geometry in SMITER.

This case was run in the L2G module in SMITER. Figure 6 shows a direct comparison of the heat load results on the target between SMARDDA and L2G. The maximum heat load of 2.3 MW coincides, while the maximum relative error on some triangles exceeds 2% (Figure 7 (a)). The comparison between PFCFLUX and L2G heat load results in terms of relative error is comparable to SMARDDA vs. L2G, although in some areas (especially in the horizontal and vertical apexes of the panel) the error reaches 5% (Figure 7 (b)).



(a) **(b)**

Figure 6: Comparison of the heat load results on the target for the NF55 case. (a) SMARDDA: Q [W]. (b) L2G: q [W].



(a) **(b)**

Figure 7: Heat loads relative error [%] for the NF55 case. (a) SMARDDA vs. L2G. (b) PFCFLUX vs. L2G.

It can be seen that in terms of accuracy the results obtained with the L2G module are comparable to existing field-line tracing solutions.

3.2 Performance tests

The cases in the L2G module are run with 1 OpenMP thread by default. A comparison of execution times for different parts of tracing in SMARDDA and L2G for the NF55 case is shown in Table 1. L2G's performance with just 1 thread is already much better than SMARDDA's performance for NF55.

Table 1: SMARDDA vs. L2G execution times for the NF55 case.

	SMARDDA	L2G
preprocessing	77.657 s	37.82 s
field-line tracing	76.475 s	20.8 s
total	154.132 s	58.62 s

For performance tests on huge meshes the target and shadow geometries from the Inres1 case were remeshed (in the SMESH module in SMITER) to element sizes of approximately 10, 3, 2 and 1 mm. The last size resulted in a target mesh of 3046416 and a shadow mesh of 25309694 triangles. For the original mesh and resized meshes the Inres1 case was run for 1, 2, 4, 8 and 16 OpenMP threads. Figure 8 shows total execution times depending on the mesh size and number of threads used. The results show that many threads execution speeds up field-line tracing on million size meshes for a factor more than 2 with the use of 16 threads.

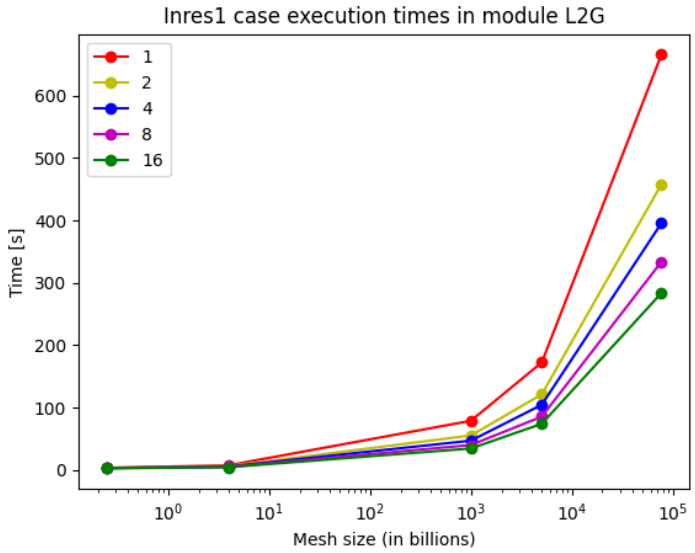


Figure 8: Total execution times depending on the mesh size (product of the target and shadow geometries triangles number) and number of threads used.

4 CONCLUSION

Benchmarking of the developed L2G module for field-line tracing within the SMITER framework shows that this solution can match the accuracy of existing solutions, i.e., SMARDDA and PFCFLUX. Performance tests furthermore imply that the use of OpenMP for execution can significantly speed up field-line tracing on huge meshes of a million triangles size. The L2G module is a good basis for a complete solution for magnetic field-line tracing in tokamak

plasmas. The approach in its development (use of specialized libraries for the C++ core and Python/Cython for the interface and API) offers flexibility and modularity for further development. Upgrades are expected for the solution in the sense of adding different plasma simulation scenarios and capabilities, user-friendly options and also parallelization paradigms to further increase performance. In the end it is expected that a mature solution would replace the current field-line tracing core in the SMITER framework.

ACKNOWLEDGMENTS

The author LB acknowledges funding from the LECAD laboratory (University of Ljubljana, Faculty of Mechanical Engineering) as a member of the Slovenian Fusion Association (SFA) jointly with the EUROfusion consortium for the work in his Master Thesis. The main part of the Master Thesis was the development of the software solution L2G presented in this article.

REFERENCES

[1] W. Arter, V. Riccardo, G. Fishpool, "A CAD-Based Tool for Calculating Power Deposition on Tokamak Plasma-Facing Components", IEEE Transactions on Plasma Science, 42:7, 2014, pp. 1932–1942.

[2] L. Kos, et al., "SMITER: A field-line tracing environment for ITER", Fusion Engineering and Design, 146, 2019, pp. 1796–1800.

[3] SALOME: The open source integration platform for numerical simulation, `http://www.salome-platform.org`, 2021. Accessed: 2021-08-20.

[4] M. Kocan, et al., "Impact of a narrow limiter SOL heat flux channel on the ITER first wall panel shaping", Nuclear Fusion, 55, 2015, pp. 033019.

[5] ALGLIB: A cross-platform numerical analysis and data processing library, `https://www.alglib.net/interpolation/spline3.php`, 2021. Accessed: 2021-08-20.

[6] RKF45: A C++ code which implements the Watt and Shampine RKF45 ODE solver, `https://people.sc.fsu.edu/~jburkardt/cpp_src/rkf45/rkf45.html`, 2021. Accessed: 2021-08-20.

[7] Intel Embree: A collection of high performance ray tracing kernels, `https://www.embree.org`, 2021. Accessed: 2021-08-20.