AGROS - SOFTWARE FOR PHYSICAL FIELDS SIMULATIONS

P. Karban*, D. Pánek*, J. Kaska*

*University of West Bohemia, Czech Republic
{karban, panek50, kaskaj}@fel.zcu.cz

Keywords: FEM, simulation, modeling, scripting, open source software, physical fields

Abstract

Agros is an open-source software designed for physical simulations. The aim of this contribution is to present features of a new version of this software. It discuss the architecture of the software, used third party codes, basic principles of scripting and possibilities of solving optimization tasks.

1 Introduction

Agros is a software for simulating physical fields using the finite element method. It has been in development for more than 14 years. The first versions were based on the library for finite element modelling Hermes. First, electrostatic and magnetic fields were implemented. Gradually, other physical fields and their coupling possibilities were added. Since 2014, a gradual transition to a new core -- the Deal.II library has begun. This transition has been successfully completed and a new version of the agros software is currently being prepared for release. The aim of this paper is to introduce the features and new functionality of the new version of agros. Note that source code of the project argos is available at [1].

2 Architecture

When developing the current version of agros, one of the requirements was to achieve maximum modularity, so that new components could be added in the simplest way possible. The structure of agros is shown in Figure 1.

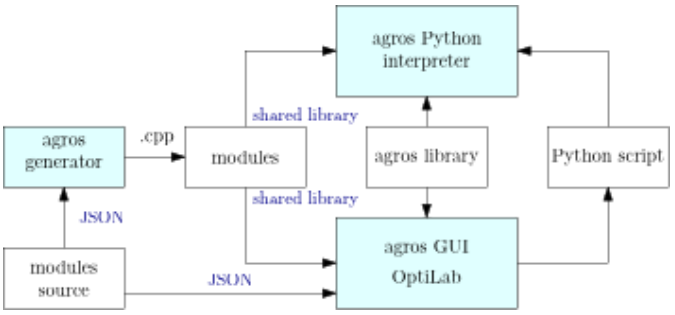

Figure 1: Structure of agros.

2.1 Graphical user interface

The program consists of a graphical user interface allowing to create two dimensional geometries, set material properties, and boundary conditions, depicted in Figure 2, and also to analyse results. Besides specifying model, running fem calculations and analysing results, the GUI also allows to set and run optimization tasks.

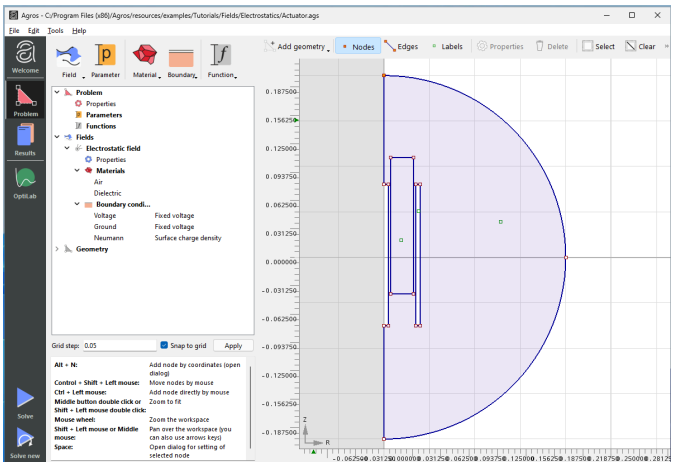

Figure 2: Graphical user interface.

2.2 argos library

Agros library is an intermediary between the high level represented by the GUI or agros wrapper and the low level represented by the Deal.II library and matrix solvers. The library allows, based on the mathematical description of the physical field through the Deal.II, to construct a system of equations, which is then solved using the matrix solver. The matrix solver is again solved as a separate module, which allows adding new solvers. Currently, it is possible to use the internal solver of the Deal.II library and the external solvers MUMPS and AMGCL.

2.3 argos generator

Relatively independent part of the argos project is *argos generator*, which produces C++ code from JSON files describing the mathematical formulations of physical fields. This code can be compiled, dynamically linked and directly used from GUI. Using module (JSON) files and the agros generator is a way to relatively simply add a new physical field.

For example the volume integral (first term) from the weak formulation of electrostatic field in the form

$$\int_{\Omega} \varepsilon\left(\frac{\partial \varphi}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial \varphi}{\partial y}\frac{\partial v}{\partial y}\right)dxdy - \int_{\Gamma} \varepsilon g_N v ds = 0$$

can be expressed in the source (JSON) module file by so called matrix form as

```
"matrix_forms": {

"laplace": {el_epsr * EPS0 * (udx * vdx + udy * vdy), "i": 1, "j": 1 }

}
```

2.4 argos Python wraper

In the current version of agros, scripting has been separated from the GUI asa separate package, which can be downloaded from the PYPI system using

```
pip install agrossuite
```

The scripting in agrossuite is relatively simple. The first step is importing and setting basic properties.

```
from agrossuite import agros

problem = agros.problem(clear = True)
problem.coordinate_type = "planar"
problem.mesh_type = "triangle"

self.electrostatic =
problem.field("electrostatic")

self.electrostatic.analysis_type =
"steadystate"

self.electrostatic.number_of_refinements = 1
self.electrostatic.polynomial_order = 2
self.electrostatic.solver = "linear"
```

The next step is defining boundary conditions and materials used in the problem.

```
self.electrostatic.add_boundary("Neumann",
"electrostatic_surface_charge_density",
{"electrostatic_surface_charge_density" : 0})

self.electrostatic.add_material("Dieletric 1",
{"electrostatic_permittivity"       :       3,
"electrostatic_charge_density" : 0})
```

Than it is possible to add geometry in similar manner as in GUI using nodes and edges

```
geometry = problem.geometry()

geometry.add_edge(0, -R_Air, R_Air, 0, angle =
90, boundaries = {"magnetic" : "Zero"})
```

Material in the certain domain can be specified using labels.

```
geometry.add_label(12.3992, 0.556005,
materials = {"electrostatic" : "Dieletric 1"})
```

Now the problem is set and it is possible to solve it.

```
self.computation = problem.computation()
self.computation.solve()
```

The last step is analyzing results

```
solution =
self.computation.solution('electrostatic')

local_values = solution.local_values(5.06,
7.537)

volume_integrals =
solution.volume_integrals([4])
```

2.5 OptiLab

Optilab is a tool for solving optimization problems. At the moment it allows solving both single and multi criteria optimization problems using most of the basic classes of optimization methods. The argos contain the following third party libraries to solve the optimization tasks: NLopt [2] library for standard methods, BayesOPT [3] for Bayessian methods and Pagmo [4] for genetic and swarm algorithms.

Acknowledgements

This publication was created with the support of the internal project SGS-2024-025 at the University of West Bohemia.

References

[1] P. Karban at all. agros [online]. Available from: https://github.com/artap-framework/agrossuite.git [Accessed 29 December 2024].

[2] Steven G. Johnson. The NLopt nonlinear optimization package. 2007. Available at: https://nlopt.readthedocs.io/ [Accessed 29 December 2024].

[3] Ruben Martinez-Cantin. BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. J. Mach. Learn. Res., 15(1):3735–3739, January 2014.

[4] Biscani, Francesco, and Dario Izzo. "A parallel global multiobjective framework for optimization: pagmo." *Journal of Open Source Software*, vol. 5, no. 53, 2020, p. 2338. The Open Journal. DOI: 10.21105/joss.02338.