Conference Paper, Published Version

**Gisen, David**

# How to benefit from OpenFOAM capabilities in custom software projects

Verfügbar unter/Available at: https://hdl.handle.net/20.500.11970/105359

Vorgeschlagene Zitierweise/Suggested citation:
Gisen, David (2018): How to benefit from OpenFOAM capabilities in custom software projects. In: Bundesanstalt für Wasserbau (Hg.): BAW-Workshop OpenFOAM® in Hydraulic Engineering. Karlsruhe: Bundesanstalt für Wasserbau. S. 3-3.

# How to benefit from OpenFOAM capabilities in custom software projects

*Author: David Gisen, Bundesanstalt für Wasserbau*

OpenFOAM consists of numerous libraries written in C++, providing a toolbox for all kinds of numerical calculation. For engineers with little training in programming, it can be difficult to find and apply these tools because the authors distributed the code in small chunks for reusability (Jasak et al. 2007). I aim to lower this hurdle by describing the approach I use. For my PhD thesis, I built an individual-based model of fish moving in a laboratory flume (Gisen 2018). Model fish took input stored on an OpenFOAM mesh, e.g. velocity, which had to be interpolated to the fish center. But how?

I found the most convenient ways of searching the code to be the OpenFOAM C++ Source Code Guide (https://cpp.openfoam.org, starting with 3.0.1) and the GitHub repository (https://github.com/OpenFOAM?tab=repositories, starting with 2.0.x). The guide is recommended for basic research, as its results are clearer and it contains descriptions of member functions. The repository is more suitable for digging through classes located in neighbour directories, as its navigation layout is clearer.

The first step to find suitable functions is to search for keywords similar to the desired function name in the guide. For all results, member function text descriptions have to be evaluated. If they match the task, they can be inserted in custom C++ code. Following inclusion of the headers, the code has to be compiled and linked. This can be done by using OpenFOAM's wmake tool with an options file or a custom makefile. Anyway, they have to contain the paths of the header files (.H) and compiled libraries (.so) of every function library included. The basic path is known from the guide. Then, the corresponding folder lnInclude has to be added to the EXE_INC variable. The same goes for the respective libraries at $FOAM_LIBBIN, which are added to EXE_LIBS. After successful compilation and linkage, the new executable can be applied to produce data, for example model fish tracks (Figure 1).

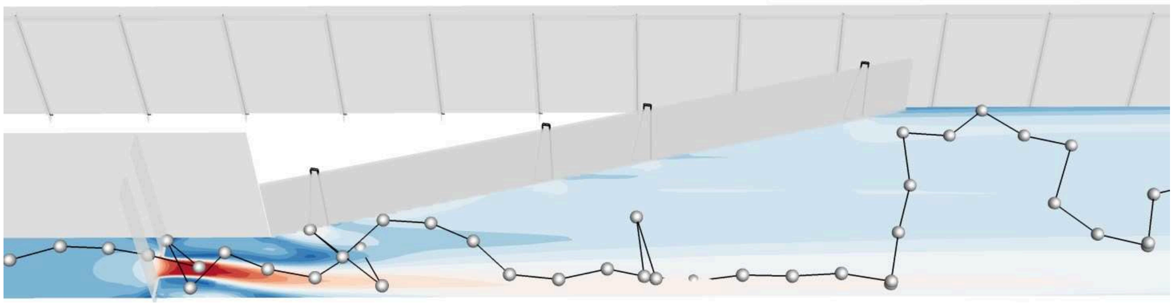

*Figure 1: Model fish track in a hydraulic flume. Flow from left to right, fish movement vice versa.*

## Literature

Gisen, D.C. (2018): Modeling upstream fish migration in small-scale using the Eulerian-Lagrangian-agent method (ELAM). Dissertation Universität der Bundeswehr München, in press, https://hdl.handle.net/20.500.11970/105158.

Jasak, H.; Jemcov, A.; Tuković, Ž. (2007). OpenFOAM: A C++ library for complex physics simulations. In: Proc. International Workshop on Coupled Methods in Numerical Dynamics. IUC, Dubrovnik, Croatia.