MAX PLANCK INSTITUTE FOR DYNAMICS OF COMPLEX TECHNICAL SYSTEMS MAGDEBURG
CSC COMPUTATIONAL METHODS IN SYSTEMS AND CONTROL THEORY
MaRDI
Universität Münster

Improving Interoperability in Scientific Computing via MaRDI Open Interfaces

Dmitry I. Kabanov, Stephan Rave, Mario Ohlberger

Institute for Analysis and Numerics, University of Münster, Münster, Germany

Summary

Computational scientists often face two obstacles while conducting numerical experiments:

- **Multiple programming languages** are used for numerical solvers
- Multiple solvers have **different interfaces** for the same problem types

Combination of solvers, thus, could lead to significant programming effort on scientist's side: writing bindings and adapting interfaces takes time. MaRDI Open Interfaces aim to improve interoperability in Scientific Computing by alleviating these obstacles by replacing pairwise bindings between languages and implementations with decoupled bindings.

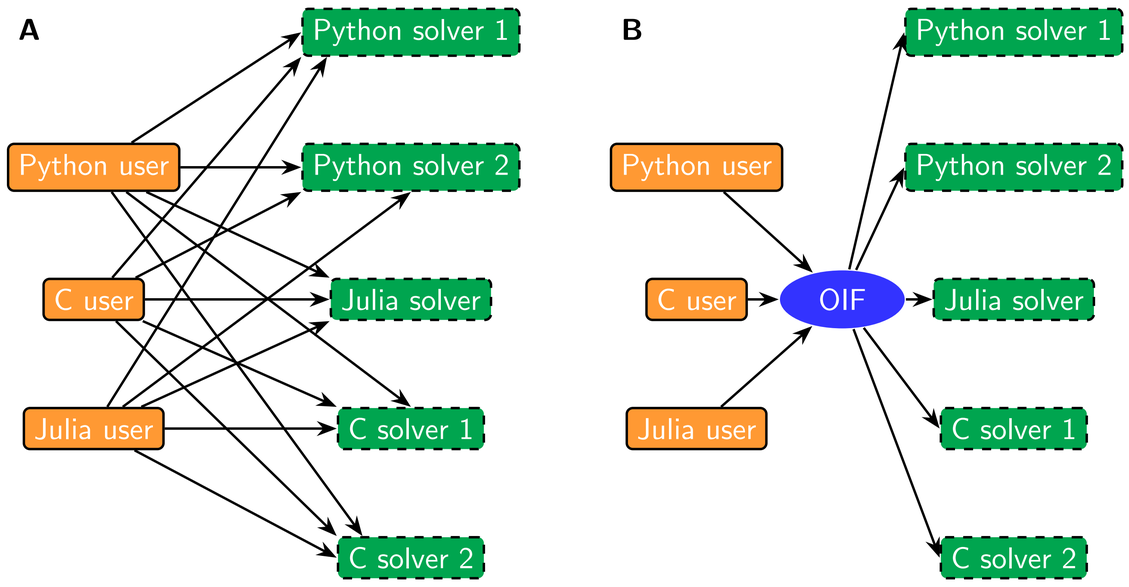


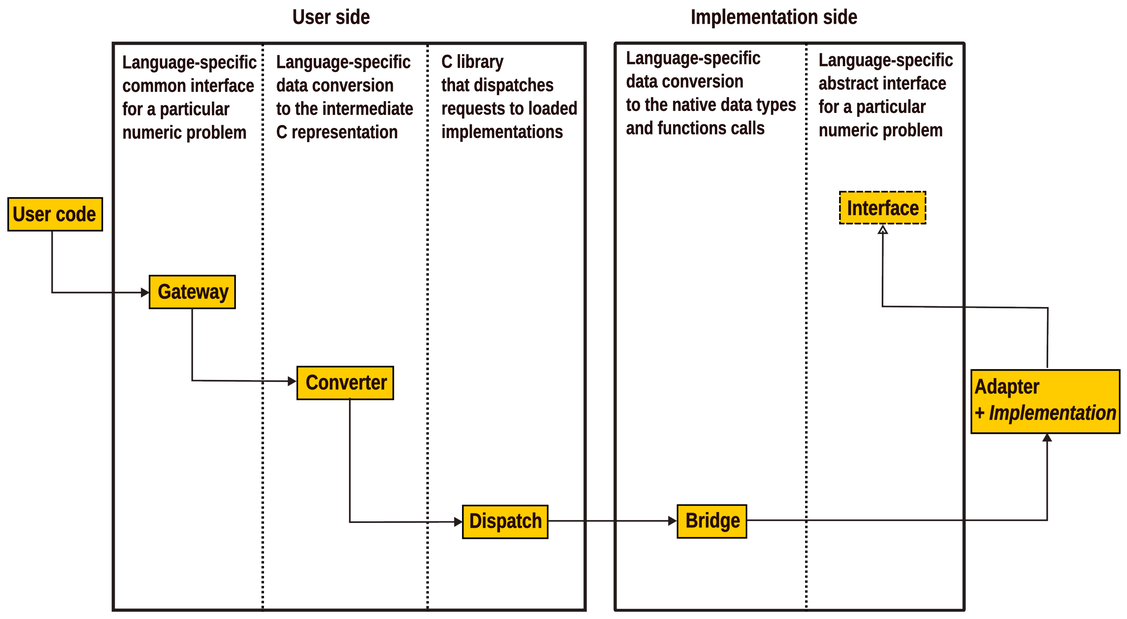
Figure 1: Schematic comparison of two approaches to the problem of multiple languages/multiple implementations. **A** Standard pairwise bindings **B** Bindings via MaRDI Open Interfaces.

Objectives

- Develop library that passes data between languages automatically
- Develop generic interfaces for typical numerical problems
- Encourage the community to program against these interfaces

Similar to software packages such as pyMOR (Milk, Rave, and Schindler, 2016), SUNDIALS (Gardner et al., 2022), or preCICE (Chourdakis et al., 2022), we aim to abstract out discrepancies between different solvers, so that computational scientists would spend less time connecting them.

Software Architecture and Data Flow



Example usage

Consider the inviscid Burgers' equation problem:

$$\frac{\partial u}{\partial t} + \frac{\partial \left(u^2/2\right)}{\partial x} = 0, \quad t \in [0,2], \quad x \in [0,2]$$
$$u(t,0) = 0.5 - 0.25\sin(\pi x) \text{ and } u(t,0) = u(t,2),$$

which we integrate using an open interface for Initial-Value Problems (IVP) for ordinary differential equations.

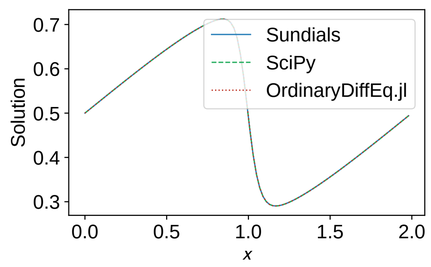


Figure 2: Solution of the Burgers' problem using open interface IVP with three implementations: SUNDIALS CVODE, SciPy (Virtanen et al., 2020), and OrdinaryDiffEq.jl (Rackauckas and Nie, 2017).

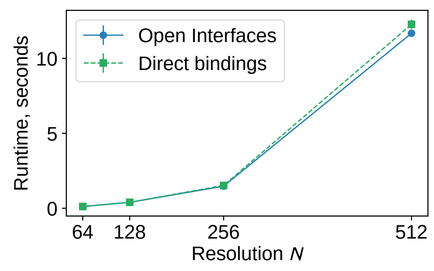
User code in Python:

```python
from oif.interfaces.ivp import IVP
...

# `BurgersEquationProblem` is a utility user class.
impl = "scipy_ode"  # or "sundials_cvode", or "jl_diffeq"
problem = BurgersEquationProblem(N=1001)
s = IVP(impl)
s.set_initial_value(problem.y0, problem.t0)
s.set_rhs_fn(problem.compute_rhs)

times = np.linspace(problem.t0, problem.tfinal, num=11)
soln = [y0]
for t in times[1:]:
    s.integrate(t)
    soln.append(s.y)
```

Performance study



Do we sacrifice high-performance computing? *No.* Integration of a 2D Gray–Scott reaction–diffusion system using MaRDI Open Interfaces and SUNDIALS CVODE solver and using the same solver via direct bindings (from Python) takes virtually the same time.

Figure 3: Comparison of runtimes, averaged over 30 runs, between using the same solver via MaRDI Open Interfaces and directly.

Conclusions

We demonstrated how MaRDI Open Interfaces can be used to improve interoperability in Scientific Computing. Using this software package, computational scientists can connect numerical solvers written in different languages, without explicitly writing bindings to these solvers, enabling complex experiment workflows.

Besides, the library allows switching between different implementations of a numerical algorithm without extensive modifications of user's code, which facilitates benchmarking numerical software in terms of performance and correctness.

Acknowledgments

The authors would like to thank for the provided funding the National Research Data Infrastructure, project number 460135501, NFDI 29/1 "MaRDI – Mathematical Research Data Initiative" and the German Research Foundation, Germany's Excellence Strategy EXC 2044-390685587, "Mathematics Münster: Dynamics–Geometry–Structure".

References

Chourdakis, G. et al. (2022). "preCICE v2: A sustainable and user-friendly coupling library". DOI: 10.12688/openreseurope.14445.2.

Gardner, D. J. et al. (2022). "Enabling New Flexibility in the SUNDIALS Suite of Nonlinear and Differential/Algebraic Equation Solvers". DOI: 10.1145/3539801.

Milk, R., S. Rave, and F. Schindler (2016). "pyMOR – Generic Algorithms and Interfaces for Model Order Reduction". DOI: 10.1137/15m1026614.

Rackauckas, C. and Q. Nie (2017). "DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia". DOI: 10.5334/jors.151.

Virtanen, P. et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing In Python". DOI: 10.1038/s41592-019-0686-2.