Technische Universität Berlin


Application of object-oriented design to modeling coupled processes in hydro- and environmental systems

I. Özgen, F. Amann, F. Tügel, J. Zhao, R. Hinkelmann | Chair of Water Resources Management and Modeling of Hydrosystems | BIMoS Day „Shallow Water Flow Simulations“, May 22, 2017

Motivation: Flash flood in Wadi Bili

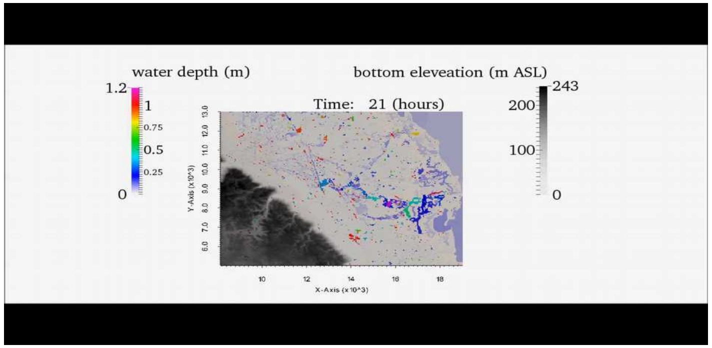
El Gouna, Wadi Bili, 27 Oct 2017

Motivation: Flash flood in Wadi Bili

El Gouna, Wadi Bili, 27 Oct 2017

Motivation: Flash flood in Wadi Bili

Motivation: Flash flood in Wadi Bili



Tügel *et al.* (2017)

Motivation

- How can we add other processes such as infiltration, sediment transport and morphodynamics, contamination transport?

Object-oriented programming (OOP)

- OOP is a programming technique that supports objects, classes, and inheritance.
- OOP concepts:
  - Class encapsulation
  - Polymorphism
  - Class inheritance
  - Generic programming
- Aims to enable abstraction, modularity of code and reusability

- Some well known OOP languages: C++, Java, C#
- Languages that support OOP: Modern Fortran, MATLAB

Criticism of OOP

- Less efficient
- Encourage unneeded complexity
- Produces „bloated“ code base

- However: The high-performance scientific computing suite DUNE is written in C++ using OOP.

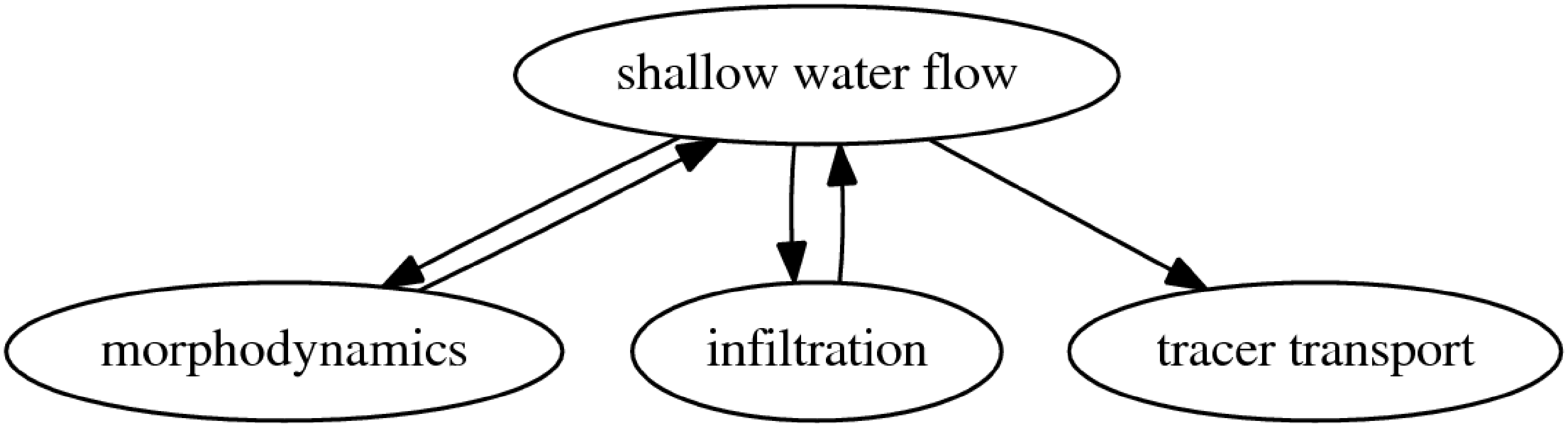
Hydroinformatics Modeling System (hms)

- Java-based OOP modeling framework (suite)
- Focus on modeling of coupled processes related to shallow water flow
- Focus on enabling fast prototyping of methods

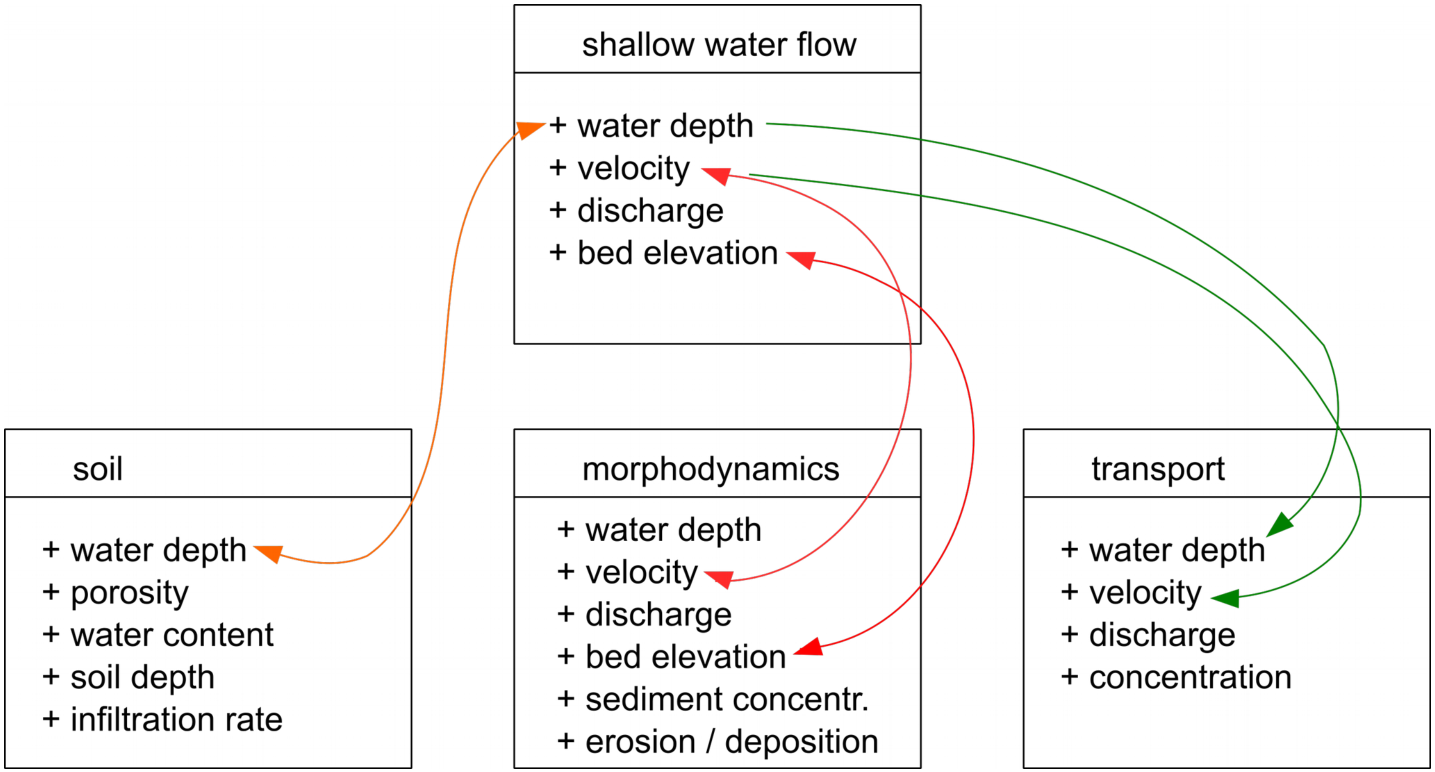
- Solvers and numerical algorithms have to be written in the most general way possible to apply them to as many physical processes as possible (modularity and reusability)

- **Aim of this talk** is to present the numerics module of hms, as an example of OOP design applied to modeling coupled processes.

Shallow water flow and related processes



hms: Encapsulation of physical processes



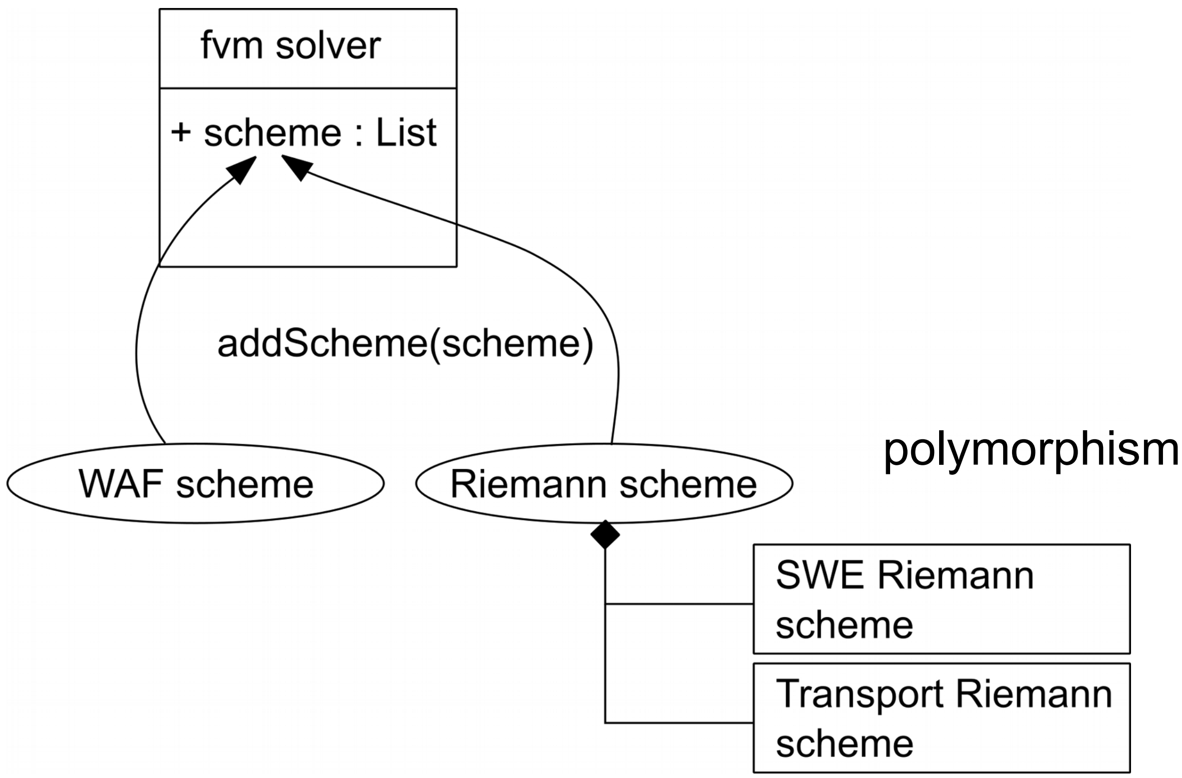
Technische Universität Berlin

hms: A generalized finite-volume solver

Note: Every cell and every edge is an object, stored in a list. State variables are stored in cell objects.

1. initialize state variables
2. preparation step prior to global loop
3. compute values of conserved variables (loop over cells)
4. compute flux terms (loop over edges)
5. compute source terms (loop over cells)
6. update independent state variables (loop over cells)
7. update dependent state variables (process coupling)

hms: A generalized finite-volume solver

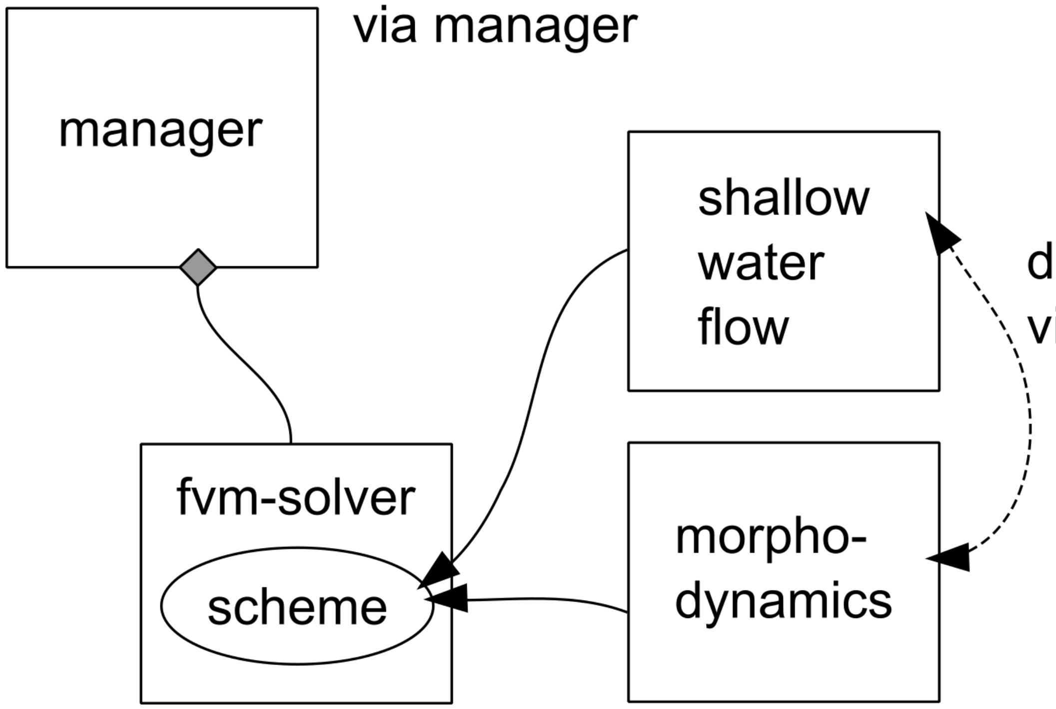


Example: Dam-break flow over movable bed

- A numerical experiment by Cao et al. (2004)
- Governing equations from (Simpson & Castelltort, 2006)
- Domain: 50,000 m long, 200 m wide
- Cell size: 10 m
- Manning coefficient: n = 0.03 ms$^{-1/3}$
- Sediment diameter: d = 8 mm

Morphodynamics: Object-oriented solution

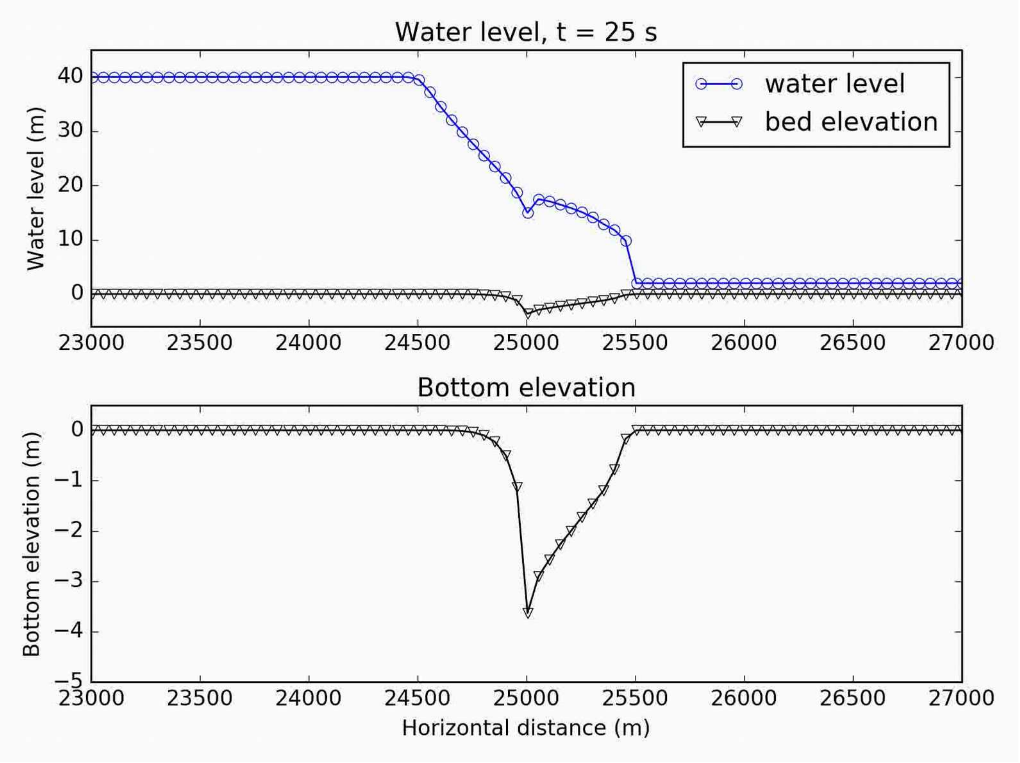
add/remove schemes
via manager



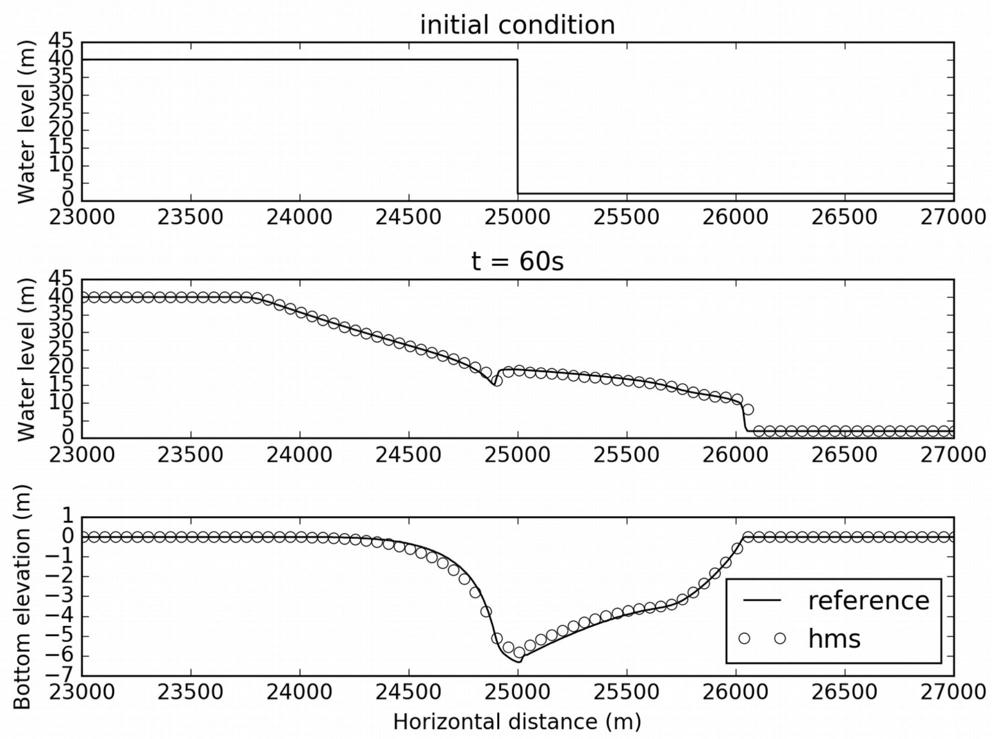
data exchange
via manager

strategy pattern

Results and discussion



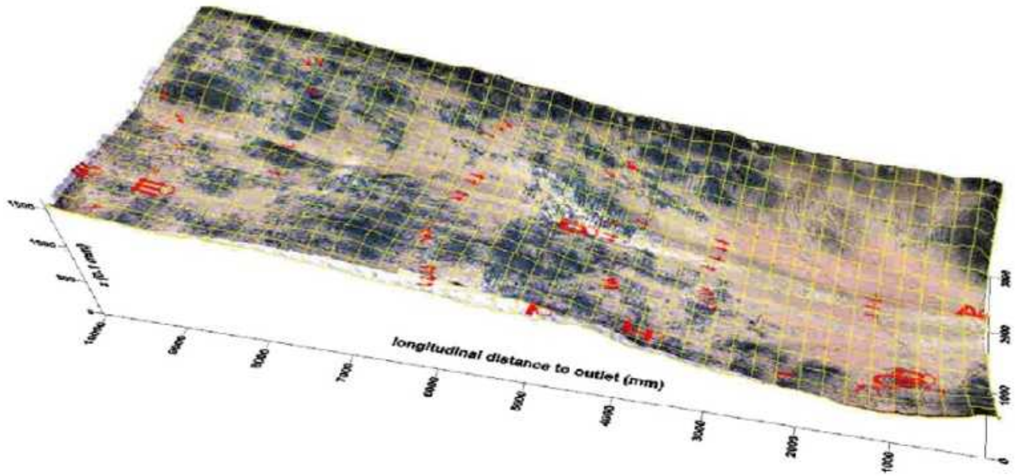
Results and discussion



Example: Tracer experiments in Thies catchment, Senegal

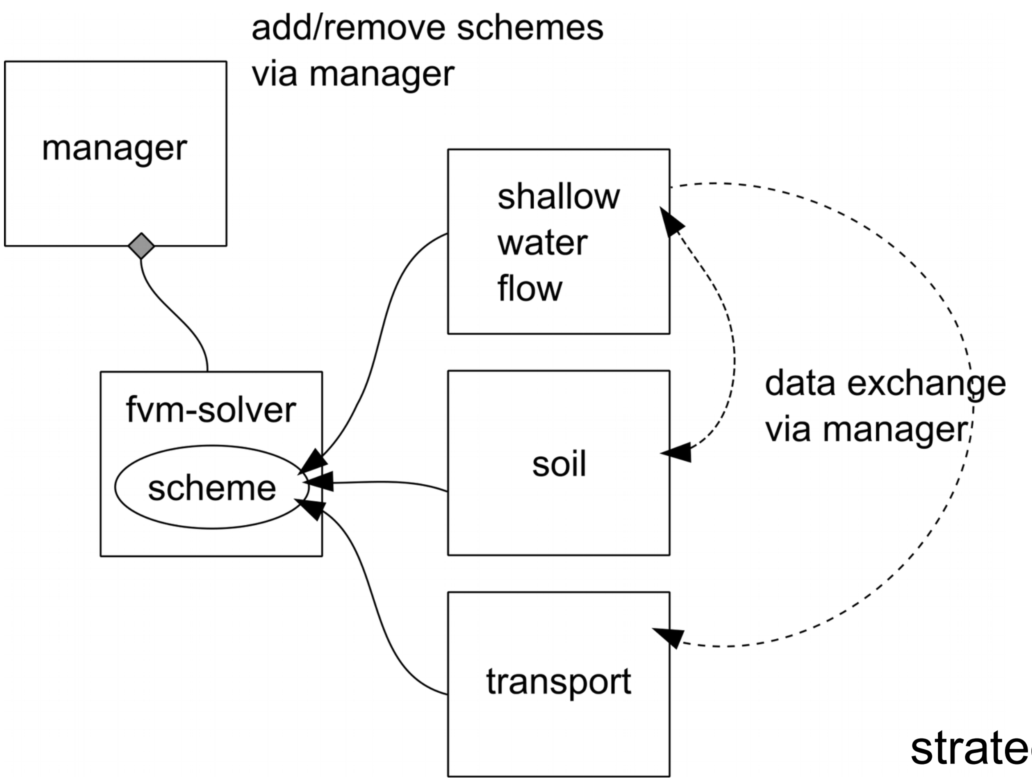
- Experiments have been carried out by Mügler *et al.* (2011)
- Domain: 10 m long by 4 m wide, with a 1% slope
- Rainfall intensity: 70 mm/h = 1.944×10^{-5} m/s (constant)
- Cell size: 0.05 m
- Variable Manning roughness: n=0.014 $ms^{-1/3}$, d_0=0.0045 m, e=0.1
- Infiltration rate: 7.36×10^{-6} m/s (constant)
- Tracer is injected after 300 sec, with a rate of 1 g/s for 30 sec
- Diffusion coefficient: 2.982×10^{-9} m^2s^{-1}

Example: Tracer experiments in Thies catchment, Senegal



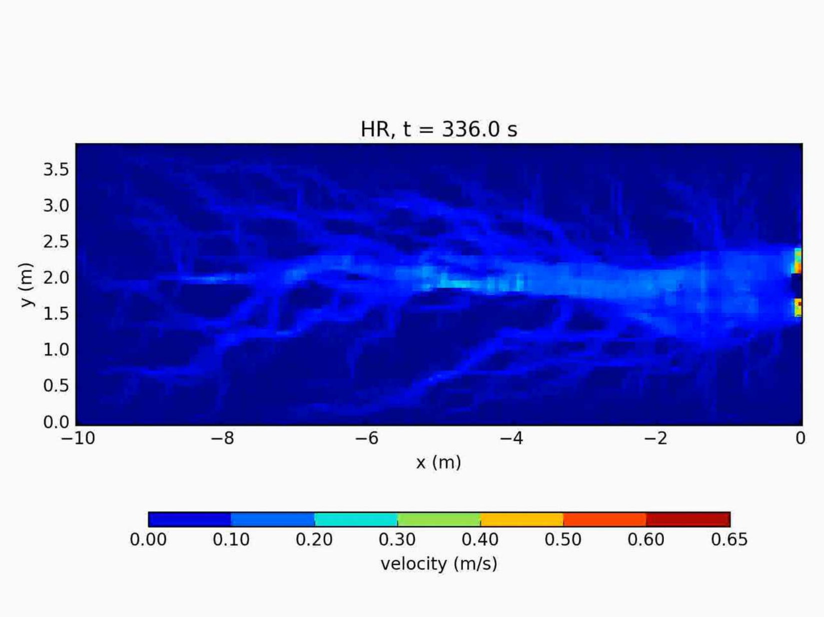
Mügler *et al.* (2011)

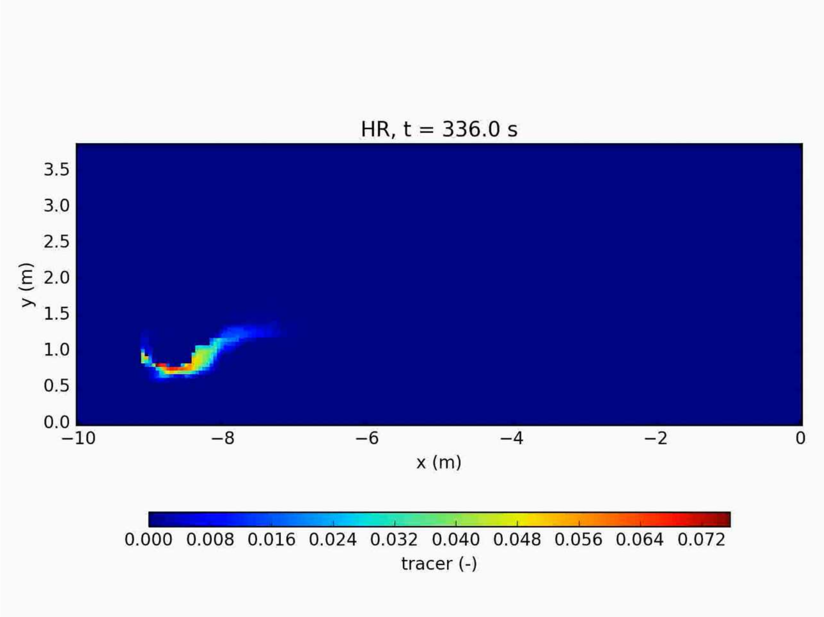
Tracer transport and infiltration: Object oriented solution



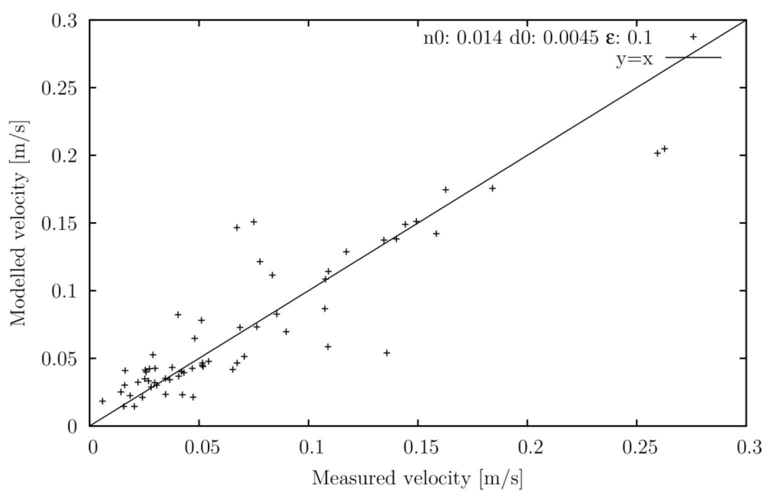
strategy pattern

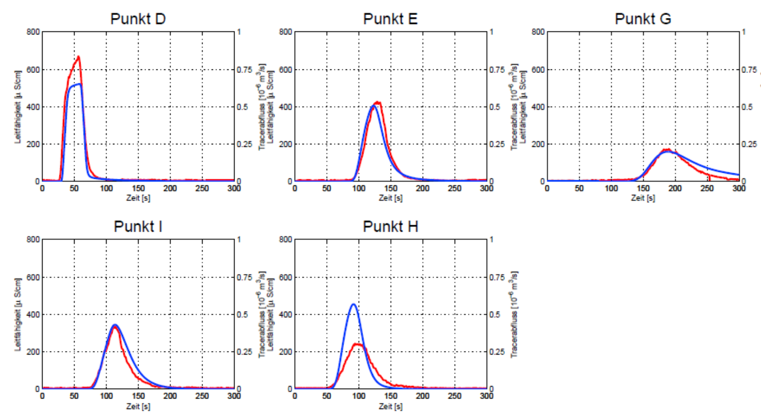
Results and discussion





Results and discussion





(Simons et al., 2014)

(Adamczak, 2012)

Technische Universität Berlin

Discussion: computational efficiency

- Scientific code usually targets computational efficiency and is limited to compute a limited number of specific processes
- The benefit of OOP in maintenance cost and flexibility may outweigh the performance penalty

- Comparison of hms with the CLAWPACK suite (procedural code in Fortran) in several shallow water benchmarks suggests that CLAWPACK is up to 20 times faster than hms → coupling processes in CLAWPACK requires writing a new solver
- DUNE suite (OOP code in C++) utilizes template metaprogramming techniques and is a highly efficient high-performance code → basically unreadable

Discussion: complexity of code

- Generalized solvers, generalized schemes, generalized data objects require generic programming (polymorphism)

- Writing everything as general as possible requires several levels of abstraction
- Perceived code complexity increases

- *Factory patterns* can be applied to provide simpler interfaces and hide complexity → might also increase code complexity (information hiding)

- *How much abstraction do you need?* or *How general should your solver be?*
  → *„Worse is better“ vs. „The MIT approach“*

Conclusions

- OOP enables a comfortable way to simulate coupled physical processes
- OOP increases the computational overhead and the complexity of code
- OOP increases the reusability and flexibility of the code

Technische Universität Berlin

Further

Busse, T. (2017) ``HMS – A Component-Based Hydroinformatics Modelling System for Flexible Model Coupling and Integration", Doctoral thesis, Technische Universität Berlin.

Özgen, I., Zhao, J., Amann, F., Hinkelmann, R. (2017) ``Applying object-oriented design concepts for the coupled numerical simulation of shallow water flow and related processes", E-proceedings of the 37th IAHR World Congress, August 13 – 18, 2017, Kuala Lumpur, Malaysia.

Simons, F., Busse, T., Hou, J., Özgen, I., Hinkelmann, R. (2014) ``A model for overland flow and associated processes within the Hydroinformatics Modelling System", Journal of Hydroinformatics, 16, 375-391.

Acknowledgment

This work is supported by the DFG Research Training Group ‚Urban Water Interfaces' (DFG-GRK2032).

Urban Water Interfaces (UWI)

Technische Universität Berlin

References

Adamczak, C. (2012), Hochauflösende Strömungs- und Transportsimulation für ein Beregnungs- und Tracerexperiment. Student research project, Technische Universität Berlin, unpublished.

Busse, T., Simons, F., Mieth, S., Hinkelmann, R. & Molkenthin, F. (2012), HMS: A generalised software design to enhance the modelling of geospatial referenced flow and transport phenomena. *In: Proceedings of the 10th International Conference on Hydroinformatics – HIC 2012*, Hamburg, Germany.

Cao, Z., Pender, G., Wallis, S. & Carling, P. (2004), Computational dam-break hydraulics over erodible sediment bed. *Journal of Hydraulic Engineering*, 130, 689-703.

Mügler, C., Planchon, O., Patin, J., Weill, S., Silvera, N., Richard, P. & Mouche, E. (2011). Comparison of roughness models to simulate overland flow and tracer transport experiments under simulated rainfall at plot scale. *Journal of Hydrology*, 402, 25-40.

Simpson, G. & Castelltort, S. (2006), Coupled model of surface water flow, sediment transport and morphological evolution. *Computers & Geosciences*, 32, 1600-1614.

References

Tügel, F., Özgen, I., Hadidi, A., Tröger, U. & Hinkelmann, R. (2017), Modelling of flash floods in wadi systems using a robust shallow water model – Case study El Gouna, Egypt. SimHydro Conference 2017, 14-16 June 2017, Nice, France.