

Dr. Simon Praetorius

AMDiS Workshop 2021

Institute of Scientific Computing

Apr 2021

Introduction

What is this course about?

1. DUNE: The **D**istributed and **U**nified Numerics Environment

[...] a modular toolbox for solving partial differential equations (PDEs) with grid-based methods [...] - <http://www.dune-project.org>

What is this course about?

1. DUNE: The **D**istributed and **U**nified Numerics Environment

[...] a modular toolbox for solving partial differential equations (PDEs) with grid-based methods [...] - <http://www.dune-project.org>

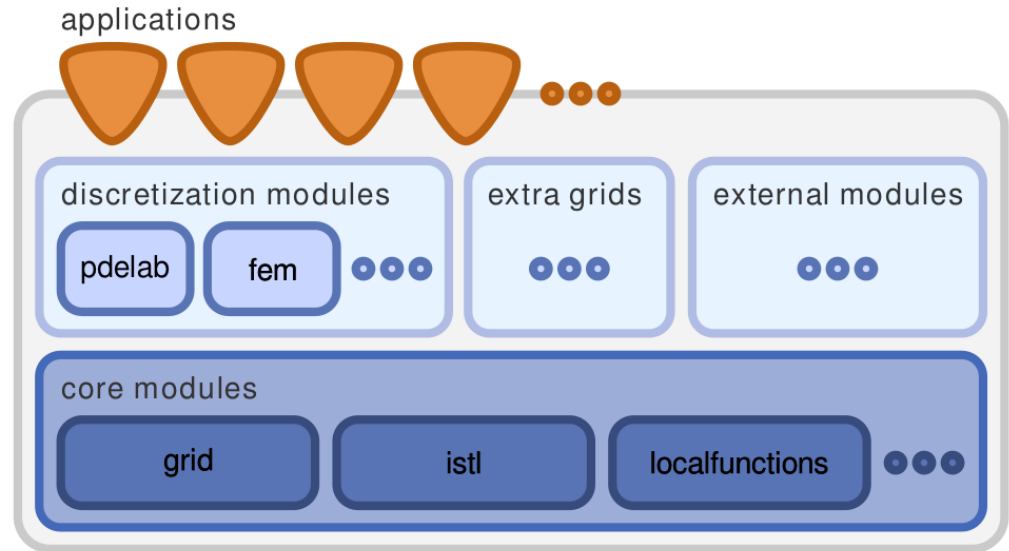
2. AMDiS: **A**daptive **M**ulti-**D**imensional Simulations

A Finite-Element discretization module allowing for fast prototyping of PDEs on adaptively refined grids - <https://gitlab.mn.tu-dresden.de/amdis/amdis-core>

Slides are based on [DUNE-PDELab Course 2021](#)

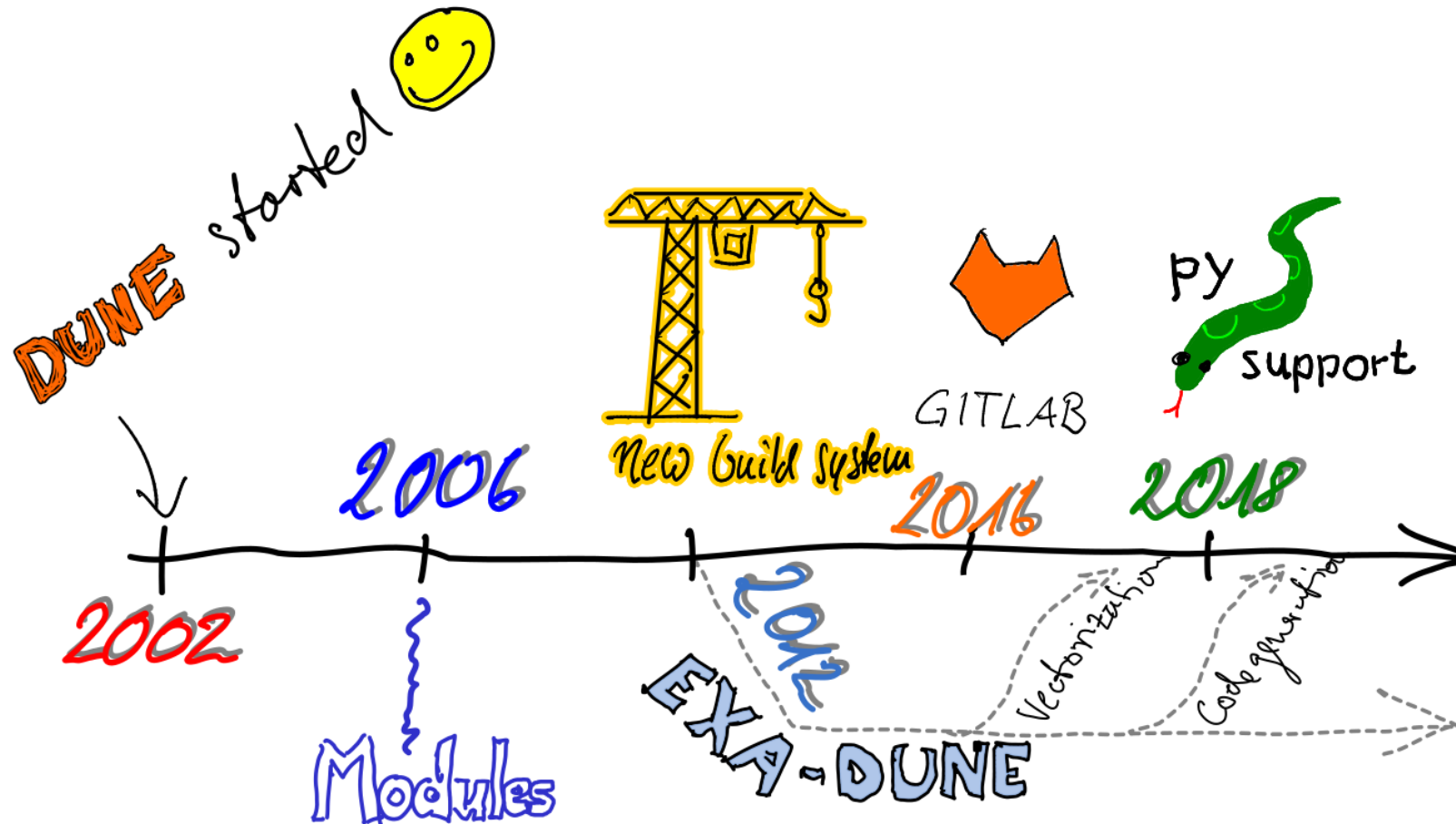
The DUNE Framework

- Modules
 - Code is split into separate modules.
 - Applications use only the modules they need.
 - Modules are sorted according to level of maturity.
 - Everybody can provide her/his own modules.
- Portability
- Open Development Process
- Free Software Licence



P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöforn, M. Ohlberger, O. Sander. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. (2008)

A brief history of DUNE



DUNE Release 2.7.1

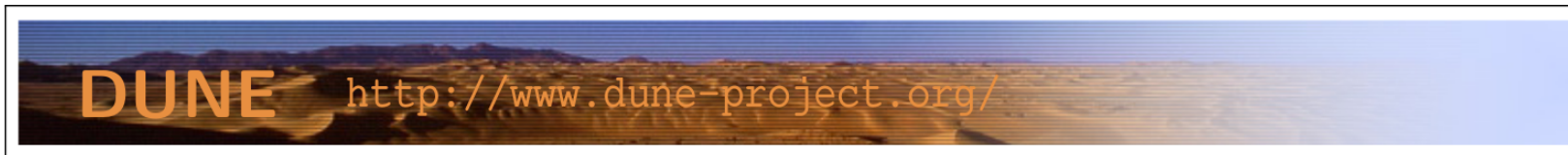
dune-common : foundation classes, infrastructure

dune-geometry : geometric mappings, quadrature rules, reference element

dune-grid : grid interface, grid construction, visualization

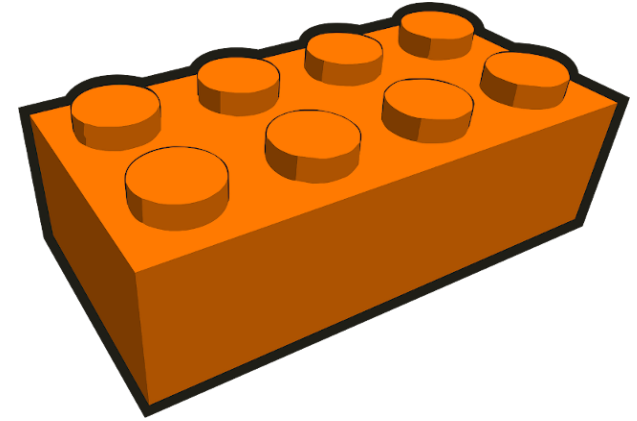
dune-istl : (Iterative Solver Template Library) generic sparse matrix/vector classes, solvers (Krylov methods, AMG, etc.)

dune-localfunctions : generic interface for local finite element functions. Abstract definition following Ciarlet. Collection of different finite elements.



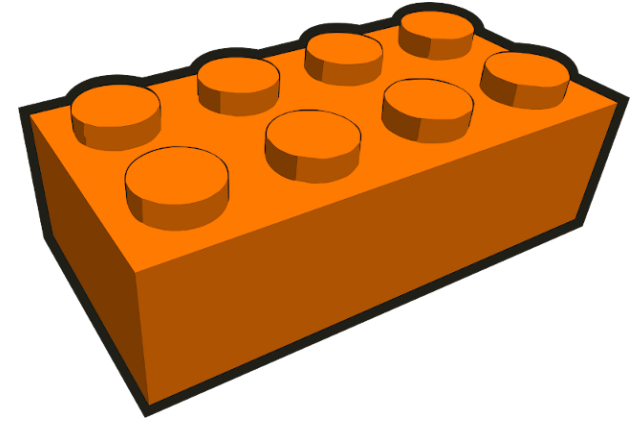
DUN(E)iverse

- modular structure
- write your own DUNE modules
- available under different licenses
- **Discretization Modules**
 - [dune-pdelab](#): Discretization module based on [dune-localfunctions](#).
 - [dune-fem](#): Discretization module based on [dune-localfunctions](#).
 - [dune-functions](#): A new initiative to provide unified interfaces for functions and function spaces.
 - **AMDiS**: based on [dune-functions](#)
- **Additional Grid Implementations...**
- **Extension Modules...**



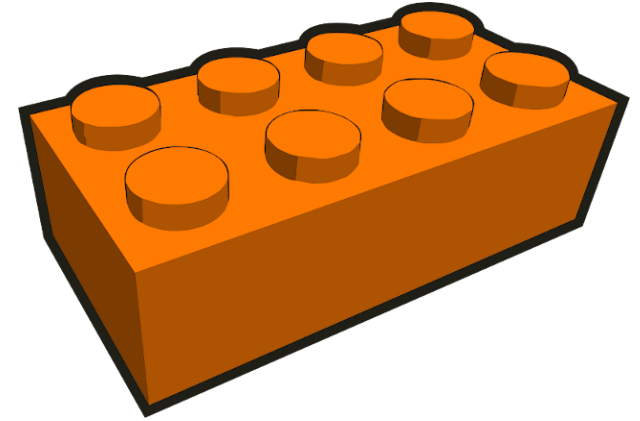
DUN(E)iverse

- modular structure
- write your own DUNE modules
- available under different licenses
- **Discretization Modules**
- **Additional Grid Implementations**
 - `dune-grid-glue`: allows to compute overlapping and nonoverlapping couplings of Dune grids, as required for most domain decomposition algorithms.
 - `dune-subgrid`: allows you to work on a subset of a given DUNE grid.
 - `dune-foamgrid`: non-manifold grids of 1d or 2d entities in higher-dim. world.
 - `dune-prismgrid`: is a tensorgrid of a 2D simplex grid and a 1D grid.
 - `dune-curvedgrid`: Arbitrary parametrization of your element geometries.
- **Extension Modules...**



DUN(E)iverse

- modular structure
- write your own DUNE modules
- available under different licenses
- **Discretization Modules**
- **Additional Grid Implementations**
- **Extension Modules**
 - `dune-python`: python bindings for central DUNE components
 - `dune-typetree`: classes to organize types in trees
 - `dune-dpg`: construct optimal Discontinuous-Petrov-Galerkin test spaces
 - `dune-tpmc`: cut-cell construction using level-sets

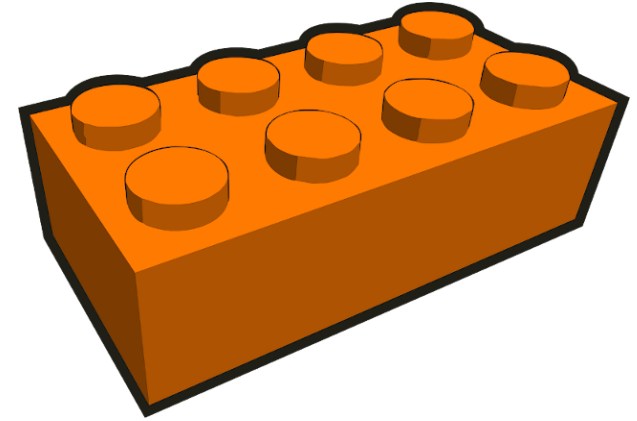


DUN(E)iverse

- modular structure
- write your own DUNE modules
- available under different licenses
- **Discretization Modules**
- **Additional Grid Implementations**
- **Extension Modules**

Goals: allow people to...

- get credit for their innovations
- experiment without breaking the core
- develop at different speeds



Getting the DUNE libraries and sources

1. Install Linux Package

- DUNE is available as system package
- Use your package manager
- Either download `.tar.gz` files with packed sources from <https://dune-project.org>
- Or install directly in Debian systems

```
sudo apt-get install libdune-common-dev libdune-geometry-dev libdune-grid-dev ...
```

Note: Check the DUNE version your distribution provides. Should be ≥ 2.7 !

Getting the DUNE libraries and sources

1. Install Linux Package

- DUNE is available as system package
- Use your package manager
- Either download `.tar.gz` files with packed sources from <https://dune-project.org>
- Or install directly in Debian systems

```
sudo apt-get install libdune-common-dev libdune-geometry-dev libdune-grid-dev ...
```

Note: Check the DUNE version your distribution provides. Should be ≥ 2.7 !

Better: For more control about DUNE versions and to use any existing module, download sources directly

Getting the DUNE libraries and sources

2. Download the Sources from Git

- Central source repository: `<gitlab>:=` <https://gitlab.dune-project.org>
- Modules grouped into `core`, `staging`, `extensions`, discretization/user modules...
- **Suggestion:** Clone all dune modules into same base directory, called `<dune-base>` in the following.

```
$ mkdir <dune-base> && cd <dune-base>
$ git clone <gitlab>/core/dune-[common,geometry,grid,istl,localfunctions]
$ git clone <gitlab>/staging/dune-[functions,typetree,uggrid]
$ git clone <gitlab>/extensions/dune-[alugrid,foamgrid,vtk]
```

Getting the DUNE libraries and sources

2. Download the Sources from Git

- Central source repository: `<gitlab>:=` <https://gitlab.dune-project.org>
- Modules grouped into `core`, `staging`, `extensions`, discretization/user modules...
- **Suggestion:** Clone all dune modules into same base directory, called `<dune-base>` in the following.

```
$ mkdir <dune-base> && cd <dune-base>
$ git clone <gitlab>/core/dune-[common,geometry,grid,istl,localfunctions]
$ git clone <gitlab>/staging/dune-[functions,typetree,uggrid]
$ git clone <gitlab>/extensions/dune-[alugrid,foamgrid,vtk]
```

- Also DUNE modules in other repositories, e.g.,

```
$ git clone https://gitlab.mn.tu-dresden.de/iwr/dune-[curvedgrid,curvedgeometry,gmsh4]
```

Getting the DUNE libraries and sources

2. Download the Sources from Git

- Central source repository: `<gitlab>:=` <https://gitlab.dune-project.org>
- Modules grouped into `core`, `staging`, `extensions`, discretization/user modules...
- **Suggestion:** Clone all dune modules into same base directory, called `<dune-base>` in the following.

```
$ mkdir <dune-base> && cd <dune-base>
$ git clone <gitlab>/core/dune-[common,geometry,grid,istl,localfunctions]
$ git clone <gitlab>/staging/dune-[functions,typetree,uggrid]
$ git clone <gitlab>/extensions/dune-[alugrid,foamgrid,vtk]
```

- Also DUNE modules in other repositories, e.g.,

```
$ git clone https://gitlab.mn.tu-dresden.de/iwr/dune-[curvedgrid,curvedgeometry,gmsh4]
```

Remark: Clone a specific version of dune, e.g., `--branch releases/2.7`

Dependency Management

dunecontrol

- control of module-interplay
- suggestions & dependencies
- integrates with cmake & git
- works with Linux, Mac and Mingw



Note: Dependencies should form a DAG

`dunecontrol cmake` : configure packages via `cmake` , include necessary paths

`dunecontrol make` : build packages in correct order

... works without `make install`

Image source: gnome

Module Management

duneproject

- Create a new module
- Specify dependencies
- Additional information in `dune.module` file:
 - Maintainer
 - Module Version
 - URL, Description
 - python package requirements
- Interactive tool

`duneproject [<project_name>]` : Create a new directory `<project_name>` containing the module information



Image source: gnome

Building the Source

- Run `dunecontrol` from within `<dune-base>`
- It searches all subdirectories recursively for a file `dune.module` describing each module and its dependencies
- Script is provided by `dune-common` module → create alias/link!

```
$ export PATH=<dune-base>/dune-common/bin:$PATH # maybe put into .bashrc
$ cd <dune-base>
$ dunecontrol cmake # configure all found modules
$ dunecontrol make # build all found modules
```

Building the Source

- Run `dunecontrol` from within `<dune-base>`
- It searches all subdirectories recursively for a file `dune.module` describing each module and its dependencies
- Script is provided by `dune-common` module → create alias/link!

```
$ export PATH=<dune-base>/dune-common/bin:$PATH # maybe put into .bashrc
$ cd <dune-base>
$ dunecontrol cmake # configure all found modules
$ dunecontrol make # build all found modules
```

Note: Both steps can be combined to `dunecontrol all`

Building the Source

- Run `dunecontrol` from within `<dune-base>`
- It searches all subdirectories recursively for a file `dune.module` describing each module and its dependencies
- Script is provided by `dune-common` module → create alias/link!
- Pass cmake/make flags in options file

```
$ cd <dune-base>
$ dunecontrol --opts=config.opts cmake    # configure all found modules
$ dunecontrol --opts=config.opts make      # build all found modules

$ cat config.opts
CMAKE_FLAGS="-DCMAKE_BUILD_TYPE=Release"
MAKE_FLAGS="-j4"
```

Building the Source

- Run `dunecontrol` from within `<dune-base>`
- It searches all subdirectories recursively for a file `dune.module` describing each module and its dependencies
- Script is provided by `dune-common` module → create alias/link!
- Pass cmake/make flags in options file
- Build a specific module and all its dependencies

```
$ cd <dune-base>
$ dunecontrol --opts=config.opts --module=dune-grid cmake # configure dune-grid + deps
$ dunecontrol --opts=config.opts --module=dune-grid make  # build dune-grid + deps
```

Building the Source

- Run `dunecontrol` from within `<dune-base>`
- It searches all subdirectories recursively for a file `dune.module` describing each module and its dependencies
- Script is provided by `dune-common` module → create alias/link!
- Pass cmake/make flags in options file
- Build **only** a specific module

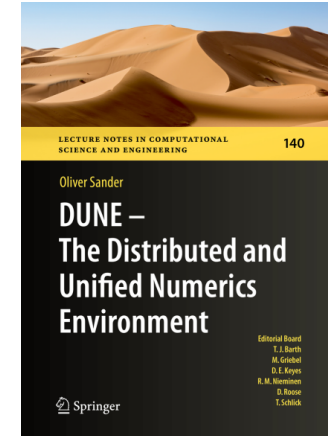
```
$ cd <dune-base>
$ dunecontrol --opts=config.opts --only=dune-grid cmake # configure dune-grid
$ dunecontrol --opts=config.opts --only=dune-grid make  # build dune-grid
```

or

```
$ cd <dune-base>/dune-grid
$ cmake build-cmake/ # configure dune-grid
$ cmake --build build-cmake/ [--target <target>] # build dune-grid
```

Resources for Help

- DUNE - The Distributed and Unified Numerics Environment, O. Sander, (2020)
- The DUNE website <https://www.dune-project.org>
- DUNE-PDELab Course:
<https://dune-pdelab-course.readthedocs.io/en/latest/index.html>
- Mailing list for user questions and discussions:
dune@lists.dune-project.org, [Un-/Subscribe](#)
- The DUNE Core-developer group:
 - Peter Bastian (Universität Heidelberg)
 - Markus Blatt (HPC-Simulation-Software & Services)
 - Ansgar Burchardt (TU Dresden)
 - Andreas Dedner (University of Warwick, UK)
 - Christian Engwer (Universität Münster)
 - Jorrit Fahlke (Jö)
 - Carsten Gräser (Freie Universität Berlin)
 - Christoph Grüninger
 - Dominic Kempf (Universität Heidelberg)
 - Robert Klöfkorn (Lund University, Sweden)
 - Simon Praetorius (TU Dresden)
 - Oliver Sander (TU Dresden)



Exercise 1

Exercise 1

1. Find out the requirements and suggestions of the dune module `dune-grid`.
2. Download and build `dune-grid` and all its dependencies.
3. Create an options-file with cmake and make options:
 - build type `Release`
 - Enable all warnings
 - Compile with 4 processors
 - Change the buildsystem from *Makefile* to *Ninja Build*
 - Set a build directory
4. Recompile with the options passed to `dunecontrol`

See also <https://dune-project.org/doc/installation/> for some more information