C++ Review DUNE 💻

Una organización donde compartir notas acerca de C++ con PDFs escritos en LaTeX.

19 de marzo del 2022

📃 Pad de apuntes

📎 Liga del PDF

🎬 Sesión grabada en `diode.zone`

📅 Cronograma de esta semana

`$ date` (🇵🇪 Lima, 🇨🇴 Bogotá, 🇲🇽 Ciudad de México -1)

- `Sat Mar 19 07:00:00 AM -05 2022`.
- `Sun Mar 20 07:00:00 AM -05 2022`.

Grid interface

recipe-integration.cc

#include <dune/common/foo.hh>

- mpihelper.hh File Reference
- parametertreeparser.hh File Reference
- timer.hh File Reference

#include <dune/geometry/foo.hh>

- quadraturerules.hh File Reference
- Dune::Geo::ReferenceElement<Implementation> Class Template Reference

#include <dune/grid/foo.hh>

- yaspgrid.hh File Reference

Code snippet

```cpp
// always include the config file
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <dune/common/parallel/mpihelper.hh>
#include <dune/common/parametertreeparser.hh>
#include <dune/common/timer.hh>

#include <dune/geometry/referenceelement.hh>
#include <dune/geometry/quadraturerules.hh>

#include <dune/grid/yaspgrid.hh>
```

recipe-integration.cc

Dune::MPIHelper Class Reference

- A.

Dune::YaspGrid<dim,Coordinates> Class Template Reference

- B.

Dune::FieldVector<K,SIZE> Class Template Reference

- C.

Code snippet

```cpp
// Maybe initialize Mpi
Dune::MPIHelper &helper =
    Dune::MPIHelper::instance(argc, argv);

// [set up grid]
const int dim = 4;
using Grid = Dune::YaspGrid<dim>;

Dune::FieldVector<double, dim> len;

for (auto &l : len)
  l = 1.0;

std::array<int, dim> cells;

for (auto &c : cells)
  c = 5;

Grid grid(len, cells);
```

recipe-integration.cc

`Dune::MPIHelper` Class Reference

- A.

Code snippet

```cpp
// [small vectors and matrices]
// make a vector
Dune::FieldVector<double, 4>
    x({1, 2, 3, 4});

// copy constructor
auto y(x);

// scaling
y *= 1.0 / 3.0;

// scalar product
auto s = x * y;

// Euclidean norm
auto norm = x.two_norm();
```

recipe-integration.cc

Dune::FieldMatrix<K,ROWS,COLS> Class Template Reference

- A.

Code snippet

```cpp
// make a matrix
Dune::FieldMatrix<double, 4, 4>
    A({{1, 0, 0, 0},
       {0, 1, 0, 0},
       {0, 0, 1, 0},
       {0, 0, 0, 1}});

// matvec: y = Ax
A.mv(x, y);

// axpy: y += 0.5*Ax
A.usmv(0.5, x, y);
```

recipe-integration.cc

`leafGridView()`

- A.

`elements()`

- B.

`geometry()`

- C.

`center()`

- D.

`volume()`

- E.

Code snippet

```cpp
// [a function to integrate]
auto u = [](const auto &x)
{ return std::exp(x.two_norm()); };

// [integration with midpoint rule]
double integral = 0.0;

// extract the grid view
auto gv = grid.leafGridView();

for (const auto &e : elements(gv))
  integral +=
      u(e.geometry().center()) *
      e.geometry().volume();

std::cout << "integral = "
          << integral
          << std::endl;
```

recipe-integration.cc

Dune::QuadratureRules<ctype,dim> Class Template Reference

- A.

type()

- B.

global()

- C.

integrationElement()

- D.

position()

- E.

weight()

- F.

Code snippet

```
// [integration with quadrature rule]
double integral2 = 0.0;
using QR =
    Dune::QuadratureRules<Grid::ctype, dim>;

for (const auto &e : elements(gv))
{
  auto geo = e.geometry();
  auto quadrature = QR::rule(geo.type(), 5);
  for (const auto &qp : quadrature)
    integral2 +=
        u(geo.global(qp.position())) *
        geo.integrationElement(qp.position()) *
        qp.weight();
}

std::cout << "integral2 = "
          << integral2
          << std::endl;
```

recipe-integration.cc

`intersections()`

- A.

`neighbor()`

- B.

`centerUnitOuterNormal()`

- C.

Code snippet

```cpp
// [integrating a flux]
auto f = [](const auto &x)
{ return x; };

double divergence = 0.0;

for (const auto &i : elements(gv))
{
  for (const auto &I : intersections(gv, i))
    if (!I.neighbor())
    {
      auto geoI = I.geometry();
      divergence +=
          f(geoI.center()) *
          I.centerUnitOuterNormal() *
          geoI.volume();
    }
}

std::cout << "divergence = "
          << divergence
          << std::endl;
```

Referencias

- A generic grid interface for parallel and adaptive scientific computing. Bastian, P., Blatt, M., Dedner, Andreas, Engwer, C., Klöfkorn, R., Ohlberger, M. and Sander, O. (2008)
- The DUNE Grid Interface An Introduction. Christian Engwer
- AMDiS Workshop 2021. Simon Praetorius
- The DUNE Grid Interface. Simon Praetorius
- The Dune Framework: Basic Concepts and Recent Developments. Peter Bastian, Markus Blatt, Andreas Dedner, Nils-Arne Dreier, Christian Engwer, René Fritze, Carsten Gräser, Christoph Grüninger, Dominic Kempf, Robert Klöfkorn, Mario Olberger, Oliver Sander
- DUNE/PDELab course
- The Distributed and Unified Numerics Environment (DUNE) Grid Interface HOWTO
- The Distributed and Unified Numerics Environment (DUNE)