A walk through some OpenFOAM code: Vector

A walk through some OpenFOAM code: Vector

Prerequisites

- You have a basic knowledge in object oriented C++ programming.
- You have a basic knowledge in the structure of OpenFOAM programming.

Learning outcomes

- You will gain experience in reading OpenFOAM classes and figure out how they work.

A walk through some OpenFOAM code: Vector

- Let's have a look at some examples in the OpenFOAM `Vector` class:
  `$FOAM_SRC/OpenFOAM/primitives/Vector`
  (go there while looking at the following slides)
  To which library does it belong?
- We find (version dependent):

```
complexVector   floatVector   lists    Vector.H
doubleVector    labelVector   vector   VectorI.H
```

  - The last two are for the templated `Vector` class (capital first letter `V` means that it is a templated class).
  - Inline functions must be implemented in the class *declaration* file, since they must be inlined without looking at the class *definition* file. In OpenFOAM there are usually files named as `VectorI.H` containing `inline` functions, and those files are included in the corresponding `Vector.H` file. There is no `*.C` file in the `Vector` class, since all functions are inlined.
  - Directories including string `{V,v}ector` are `typedef`s for `Vector` of `complex` numbers, `float`s, `label`s, `double`s and `scalar`s. The directory `lists` defines lists of vectors. What is a scalar? See `$FOAM_SRC/OpenFOAM/primitives/Scalar/scalar/scalarFwd.H`

Vector inheritance

The `Vector.H` file shows that the `Vector` class inherits from (read: "is a") `VectorSpace`:

```
template<class Cmpt>
class Vector
:
    public VectorSpace<Vector<Cmpt>, Cmpt, 3>
{
```

The `VectorSpace` class is found in `$FOAM_SRC/OpenFOAM/primitives/VectorSpace`.

Vector constructor declarations

The constructor declarations are found in `Vector.H`:

```
// Constructors

    //- Construct null
    inline Vector();

    //- Construct initialized to zero
    inline Vector(const Foam::zero);

    //- Construct given VectorSpace of the same rank
    template<class Cmpt2>
    inline Vector(const VectorSpace<Vector<Cmpt2>, Cmpt2, 3>&);

    //- Construct given three components
    inline Vector(const Cmpt& vx, const Cmpt& vy, const Cmpt& vz);

    //- Construct from Istream
    inline Vector(Istream&);
```

Vector constructor definitions

- The constructor definitions are usually found in the corresponding `.C` file, but since the constructors for the `Vector` are inlined they are found in the `VectorI.H` file:

```
template<class Cmpt>
inline Foam::Vector<Cmpt>::Vector
(
    const Cmpt& vx,
    const Cmpt& vy,
    const Cmpt& vz
)
{
    this->v_[X] = vx;
    this->v_[Y] = vy;
    this->v_[Z] = vz;
}
```

  Here, `this` is a pointer to the current object of the current class, i.e. we here set the static data member `v_` (inherited from class `VectorSpace.H`) to the values supplied as arguments to the constructor.

- It is here obvious that the member function `Vector` belongs to the class `Vector`, and that it is a constructor since it has the same name as the class.

Vector access functions

- Some access functions are declared in `Vector.H`:

```
inline const Cmpt& x() const;
inline Cmpt& x();
```

and defined in `VectorI.H`:

```
template<class Cmpt>
inline const Cmpt& Foam::Vector<Cmpt>::x() const
{
    return this->v_[X];
}

template<class Cmpt>
inline Cmpt& Foam::Vector<Cmpt>::x()
{
    return this->v_[X];
}
```

The first one is for `const` objects, and the second one can be used to manipulate the object.

Vector operators

- Some operators are defined in `VectorI.H`:

```
template<class Cmpt>
inline typename innerProduct<Vector<Cmpt>, Vector<Cmpt>>::type
operator&(const Vector<Cmpt>& v1, const Vector<Cmpt>& v2)
{
    return Cmpt(v1.x()*v2.x() + v1.y()*v2.y() + v1.z()*v2.z());
}

template<class Cmpt>
inline Vector<Cmpt> operator^(const Vector<Cmpt>& v1, const Vector<Cmpt>& v2)
{
    return Vector<Cmpt>
    (
        (v1.y()*v2.z() - v1.z()*v2.y()),
        (v1.z()*v2.x() - v1.x()*v2.z()),
        (v1.x()*v2.y() - v1.y()*v2.x())
    );
}
```

They can be changed for a specific to a type of vector, such as in `complexVectorI.H`.

vector/vector.H and vector/vector.C

- Templated class `Vector<scalar>` is `typedef` to `vector` in `vector/vector.H`:

```
typedef Vector<scalar> vector;
```

- Some static members of base class `VectorSpace` are set in `vector.C`:

```
template<>
const char* const Foam::vector::vsType::typeName = "vector";
template<>
const char* const Foam::vector::vsType::componentNames[] = {"x", "y", "z"};
template<>
const Foam::vector Foam::vector::vsType::zero(vector::uniform(0));
template<>
const Foam::vector Foam::vector::vsType::one(vector::uniform(1));
template<>
const Foam::vector Foam::vector::vsType::max(vector::uniform(VGREAT));
template<>
const Foam::vector Foam::vector::vsType::min(vector::uniform(-VGREAT));
template<>
const Foam::vector Foam::vector::vsType::rootMax(vector::uniform(ROOTVGREAT));
template<>
const Foam::vector Foam::vector::vsType::rootMin(vector::uniform(-ROOTVGREAT));
```

VectorSpace

- The `VectorSpace` class does not inherit from any other class:

```
template<class Form, class Cmpt, direction Ncmpts>
class VectorSpace
{
```

However, it is used in many types of vector spaces (search for `VectorSpace` in Doxygen):

```
VectorSpace< Barycentric2D< Cmpt >, Cmpt, 3 >
VectorSpace< Barycentric< Cmpt >, Cmpt, 4 >
VectorSpace< Barycentric< scalar >, scalar, 4 >
VectorSpace< BarycentricTensor< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< CompactSpatialTensor< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< CompactSpatialTensor< scalar >, scalar, Mrows *Ncols >
VectorSpace< CompactSpatialTensorT< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< cubicEqn, scalar, 4 >
VectorSpace< DiagTensor< Cmpt >, Cmpt, 3 >
VectorSpace< DiagTensor< scalar >, scalar, 3 >
VectorSpace< Form, Cmpt, Mrows *Ncols >
VectorSpace< linearEqn, scalar, 2 >
VectorSpace< Polynomial< PolySize >, scalar, PolySize >
VectorSpace< quadraticEqn, scalar, 3 >
VectorSpace< Roots< N >, scalar, N >
VectorSpace< RowVector< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< SpatialTensor< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< SpatialVector< Cmpt >, Cmpt, 6 >
VectorSpace< SpatialVector< scalar >, scalar, 6 >
VectorSpace< SphericalTensor2D< Cmpt >, Cmpt, 1 >
VectorSpace< SphericalTensor< Cmpt >, Cmpt, 1 >
VectorSpace< SymmTensor2D< Cmpt >, Cmpt, 3 >
VectorSpace< SymmTensor< Cmpt >, Cmpt, 6 >
VectorSpace< Tensor2D< Cmpt >, Cmpt, 4 >
VectorSpace< Tensor< Cmpt >, Cmpt, Mrows *Ncols >
VectorSpace< Tensor< scalar >, scalar, Mrows *Ncols >
VectorSpace< Vector2D< Cmpt >, Cmpt, 2 >
VectorSpace< Vector2D< scalar >, scalar, 2 >
VectorSpace< Vector< Cmpt >, Cmpt, 3 >
VectorSpace< Vector< label >, label, 3 >
VectorSpace< Vector< scalar >, scalar, 3 >
VectorSpace< Vector< vector >, vector, 3 >
```

The Vector class in Doxygen

- Use Doxygen to search for `Vector`, click on `Vector`, and click on `Vector< Cmpt >`
- Find all member data and member functions, included inherited ones.