Expression Templates in OpenFOAM

Mattijs Janssens1

1ESI-OpenCFD, ext-mjanssens@esi-group.com

**Keywords:** C++, expression templates, GPU

This abstract describes implementing expression templates in OpenFOAM. OpenFOAM is implemented to allow finite-volume modelling using an expression-like syntax. This comes with an overhead in terms of memory management. Especially in modern computer architectures there is a large mismatch between the processing speed of the computation unit and the memory access speed, especially with regards to latency. This can be mitigated by increasing the memory speed through e.g. fast-memory caching or by making the bottom level implementation more memory-access aware. In this work we demonstrate a near-transparent implementation of the latter using modern C++.

Expression Templates

In expression templates any expression becomes an instantiation of a templated class. Each class implements the indexing operator ‘[]’. Only when ‘evaluating’ the expression is this actual `[]’ operator used. Consider a typical OpenFOAM expression

scalarField A = B \* C – D;

This will allocate a temporary for the result of multiplying B and C, subtract D from it and assign the contents to A. Using expression templating instead a class hierarchy is built up mirroring the expression:

class subtract<multiply<scalar, scalar>, scalar>

where its indexing operator ‘[]’ does the actual work: b[i]\*c[i]-d[i]. In this implementation of expression templates the main OpenFOAM containers have been extended with a function to ‘wrap’ the contents in an expression form. Above expression can hence be writen as

scalarField A = B.expr() \* C.expr() – D.expr();

This same syntax can also be used for GeometricFields (field plus boundary conditions) and the most often used containers. There are two categories: can they be wrapped into expressions and can they be assigned to using an expression (as in above expression)

Table 1: Container expression template support

|  |  |  |
| --- | --- | --- |
| Class | Expression | Assignment |
| List | .expr() | yes |
| Field | .expr() | yes |
| GeometricField (e.g. volScalarField, surfaceVectorField) | .expr() | yes |
| tmp<Field> | .expr() | no |
| tmp<GeometricField> |  |  |
| DimensionedType (for constant Field) | .expr(<size>) | no |
| DimensonedType (for constant GeometricField) | .expr(<GeoField>) | no |
| fvMatrix | .expr() | yes |

Above framework supports most field functions (e.g. min, sqrt). The exception is the fvMatrix expression class which only supports linear operations.

Typical Application

A typical use of above expression template would be in complex modelling. As a test the kOmegaSST turbulence model has been fitted with above expressions. As an example one of the base functions ‘F1’ becomes a templated function

auto arg1 = min

(

min

(

max

(

(one/betaStar)\*sqrt(k\_.expr())/(omega\_.expr()\*y\_.expr()),

fiveHundred

\*(this→mu().expr()/this→rho\_.expr())

/(sqr(y\_.expr())\*omega\_.expr())

),

fourAlphaOmega2\*k\_.expr()/(CDkOmegaPlus\*sqr(y\_.expr()))

),

ten

);

return tanh(pow4(arg1));

where all the constants (one, betaStar etc) are pre-declared as a DimensionedField(..).expr().

Future WORK

The are still some missing operations – e.g. ones that change the return type. The expression template framework has been extended to use parallel execution through the C++17 ‘execution policy’ construct. This requires changing a single loop only (in the expression template evaluation for the basic `List’ container) since all the work is done through the indexing operator[] and iterators. This will benefit vectorisation and/or GPU offloading. The other ongoing effort is to move the expression templates further up into the code into the actual discretisation:

Table 2: Discretisation support

|  |  |
| --- | --- |
| Function | Returns |
| Interpolation | expression template |
| uncorrected Gauss Laplacian | fvMatrix |

**Acknowledgements**

The authors thank all those involved in the organisation of this OpenFOAM Workshop and to all the contributors that will enrich this event.

**References**

1. OpenCFD, OpenFOAM: The Open Source CFD Toolbox. User Guide Version 1.4, OpenCFD Limited. Reading UK, Apr. 2007.