2020-03-12: Eigen

Steve H opened the meeting by announcing that today will be a focused topic discussion about the Eigen C++ package, led by Ryan.

Ryan then presented the following slides

2020-03-12 - Eigen.pptx

What follows is a brief summary - see the slides for further details.

Ryan began by introducing Eigen as not only a useful tool but also a good demonstration of modern C++ coding practices and capabilities. As such, he encouraged users to explore its documentation and tutorials, as well as the code itself. It is a headers-only library so one can peruse the include directories to see the code itself.

Ryan also emphasized that we use the code already in oops. So, if you want to use it, you can just include the desired header files. There is no need to add dependencies.

Ryan started by demonstrating how EIgen facilitates array and matrix operations. Jim R asked if arrrays defined by Eigen are guaranteed to be contiguous in memory. Ryan confirmed that they are guaranteed to be contiguous.

Ryan then highlighted other features of Eigen arrays. In particular, dynamic allocation allows objects to be resized. Eigen also allows you to choose whether you want multi-dimensional arrays to be stored in row-major or column-major format, which can be useful for mixed Fortran/C++ applications.

Ryan also emphasized that Eigen produces very efficient code "under the hood". For example, matrix operations are coded in a way that allows the compiler to highly optimize them. The resulting application is likely to run faster than if you were to code these low-level operations yourself.

Chris H asked if Eigen support OpenMP for the autogenerated loops. Ryan responded that Eigen has some support for OpenMP threading, particularly for sophisticated matrix operations, but most of the parallel optimization is achieved through vectorization.

Ryan pointed out that there is a thorough tutorial available from the Eigen web site for working with the Matrix class in particular (see slides).

Eigen also supports view operations, which allow you to access data in different ways without making a copy. For example, you can choose to access data as a one-dimensional array or a multi-dimensional array. He also mentioned the .data() method (slide 11), which allows you to convert an Eigen array or matrix to a standard C++ vector. This is important for JEDI. It means, for example, you may wish to use Eigen array and matrix classes in your own classes, methods, and functions, but then when you pass the data to other components of JEDI you are able to pass it as a standard data type, like a vector. With time, more JEDI classes may use eigen classes as arguments but this is currently rare.

Ryan then briefly discussed the Eigen Tensor class, which is in development.   In C++, one way to work with multi-dimensional data structures is to define them as vectors of vectors of vectors...The Eigen tensor class lets you avoid that. It's use as a container in this sense is stable. What is in development now are more sophisticated mathematical operations. Steve H asked if simple operations are stable for the Tensor class, like A+B. Ryan confirmed that they are, but more complex operations may not work yet.

After Ryan finished his presentation, the meeting was opened for questions. Jim R asked what happens if you do something wrong, like coding A+B for matrices of different rank. Ryan said it depends on whether the objects in question are allocated statically during compile time or dynamically during run time. If they are static, you'll get a compile-time (static_assert) error. If they are dynamically allocated objects, the code execution will be terminated and you'll get an appropriate error message.

Clementine then asked what happens if you try to allocate large arrays that don't fit into memory. Ryan said that Eigen arrays and matrices are generally allocated on the heap so this isn't a common problem. He said he has allocated arrays of up to 5 GB without any issues.

Since there were no more questions, Steve H asked if there were any other issues that anyone wanted to discuss.

Mark M added a quick update for Discover users. Discover is currently migrating to a new operating system, SLES12. Mark has two jedi stacks available now on SLES12, namely a gnu-impi stack and an intel19-impi stack.   The gnu-impi stack is ready to go - all fv3-bundle tests pass. The intel-impi stack is still having problems. fv3-bundle compiles but about 12% of the tests fail. Apparently this has to do with openmp threading operations in bump. Mark continues to investigate this. If you want to use either stack on SLES12, email Mark for instructions on how to load them.

Meeting adjourned.