Future outlook and recommended libraries

Lectures on Modern Scientific Programming
Wigner RCP
23-25 November 2015

C++17 core language

It is easy to create “maps” on parameter packs

```
template<typename F, typename... Ts>
auto apply( F f, Ts... ts )
{
    return std::make_tuple( f(ts)... );
}
```

C++17 core language

But folding cannot be done easily, you must write template recursion for that. But in C++17 you can fold with operators:

```
template<typename F, typename... Ts>
auto add( Ts... ts )
{
    return (... + ts);
}
```

C++17 core language

Concepts! It is really important!

It will completely change how we write generic code.

```
template<typename T>
concept bool Addable = requires (T x) { x + x; };


template<typename T> requires Addable<T>
T add(T a, T b) { return a + b; }
```

C++17 core language

Ranges for the STL!

Standardized networking!

Standardized filesystem access!

Standardized Parallel STL algorithms!

Standardized Concurrency! (better composable std::future)

Some recommended libraries

Some recommended C++ libraries: add your choices!

Linear Algebra:

- Armadillo (TMP, + can use hw optimized libraries)
- Eigen (TMP, explicit SSE vectorization)
- MTL (TMP, GPU-friendly (CUDA)
- clBLAS (GPU-friendly, less user-friendly)
- clSPARSE (GPU-friendly, less user-friendly)

Some recommended libraries

Some recommended C++ libraries: add your choices!

ODE solvers:

- Numerical recipes
- Odeint (now in Boost)

Some libraries

Some other C++ libraries: add your choices!

Boost (Geometry, Quaternions, Octonions, Special functions, etc)

PETSc

Trilinos

FFTW++