Adding constexpr and consteval support to Clad

Mentors: Vaibhav Thakkar, Petro Zaritskyi, Vassil Vassilev

Student: Mihail Mihov
Overview of Clad

Clad is an automatic differentiation (AD) plugin for Clang.

Looks for calls to clad::differentiate, clad::gradient (and a few others).

Recursively “visits” the Clang-generated AST and finds the derivatives where possible.
Overview of constexpr and consteval

Both are used to define expressions as constant expressions.

constexpr specifier:
- Added in C++11.
- Can be applied to variables, functions, constructors, destructors.

consteval specifier:
- Added in C++20.
- Can be applied to variables, functions, constructors.
Clad current behavior
Benefits of this project

- By now both constexpr and consteval are quite widely used and their usage will only grow.
- Not having this forces projects to either skip on Clad or reduce the quality of their code by removing usage of those keywords.
Goals of my GSoC project

- Support compilation of code which has constexpr/consteval functions.
- Evaluate when it’s possible to preserve the “constant-ness” of a given function.
- Figure out how to communicate to the user if a certain derivative will behave differently to the original function (in terms of compiler- vs. run-time evaluation).
Implementation steps

Getting these specifiers to work will mostly require changes to clad::CladFunction.

In the first half of my project I will look at forward mode and then at reverse mode.
Possible API changes to Clad

The user currently has no control of what attribute is applied to the generated derivatives.
Possible further work

constexpr if/consteval if keywords:

- Added in C++17, but not widely used yet, so implementation isn’t a priority.
- Shouldn’t require much work, if constexpr/consteval are implemented.
Thank you for your attention!

Questions?