

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

```
E 18 #include "clad/Differentiator/Differentiator.h"      ■ Unknown argument: '-fno-lifetime-dse'
17
16 constexpr double sq(double x) {
15     return x*x;
14 }
13
E 12 consteval double fn(double x, double y, double z) {  ■ Unknown type name 'consteval'; did you mean 'constexpr'? (fix available)
11     double res = sq(x) + sq(y) + sq(z);
10     return res;
9 }
8
7 int main() {
6     auto d_fn = clad::gradient(fn);
5
4     double dx = 0;
3     double dy = 0;
2     double dz = 0;
1
19     d_fn.execute(3, 4, 5, &dx, &dy, &dz);
W 1     printf("Gradient vector: [%2f, %2f, %2f]", dx, dy, dz);  ■ Do not call c-style vararg functions
2
3     return 0;
4 }

constant-expr.cpp 19,41 All

mihail@desktop ~/dev/clad/build (reverse-mode-non-differentiable) $ /home/mihail/dev/llvm-18.1.5/build/bin/clang++ -DCLAD_NO_NUM_DIFF -Xclang -add-plugin -Xclang clad -Xclang -load -Xclang /home/mihail/dev/clad/bu
ild/lib/clad.so constant-expr.cpp -I/home/mihail/dev/clad/include -std=c++20 -oConstantExpr.out
constant-expr.cpp:13:32: error: cannot take address of consteval function 'fn' outside of an immediate invocation
13 |     auto d_fn = clad::gradient(fn);
   |                               ^
constant-expr.cpp:7:18: note: declared here
7 |     consteval double fn(double x, double y, double z) {
   |                   ^
1 error generated.
```

Benefits of this project

- By now both `constexpr` and `constexpr` are quite widely 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/constexpr` 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/constexpr if keywords:

- Added in C++17, but not widely used yet, so implementation isn't a priority.
- Shouldn't require much work, if constexpr/constexpr are implemented.

Thank you for your attention!

Questions?