Adding constexpr and consteval support to Clad

GSoC 2024 - Midterm Progress

What are constexpr and consteval?

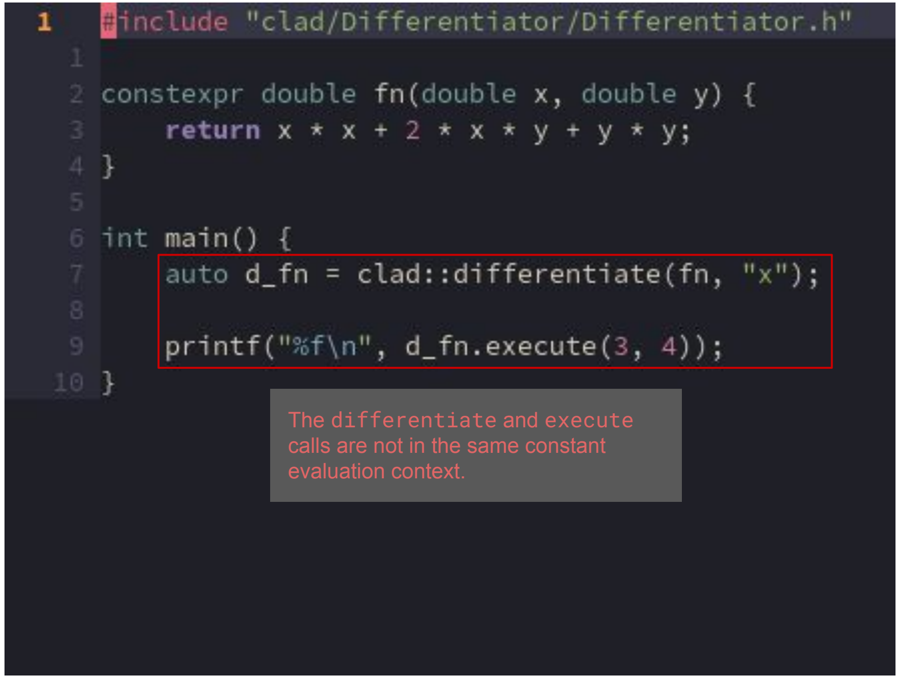
`constexpr` and `consteval` are keywords that let us specify whether a function, method or variable may be immediate. Such expressions can be possibly evaluated during the translation phase and in such case don’t make it into the final executable.

Why didn’t constexpr/consteval just work?

There were two main issues that stopped Clad from working with immediate functions:

- `CladFunction` stands in the middle of the user’s code and the generated derivatives, but it is not constexpr.
- constexpr/consteval expressions are evaluated and folded before Clad receives the AST, from which it generates derivatives.

Issue one: non-constexpr `CladFunction`



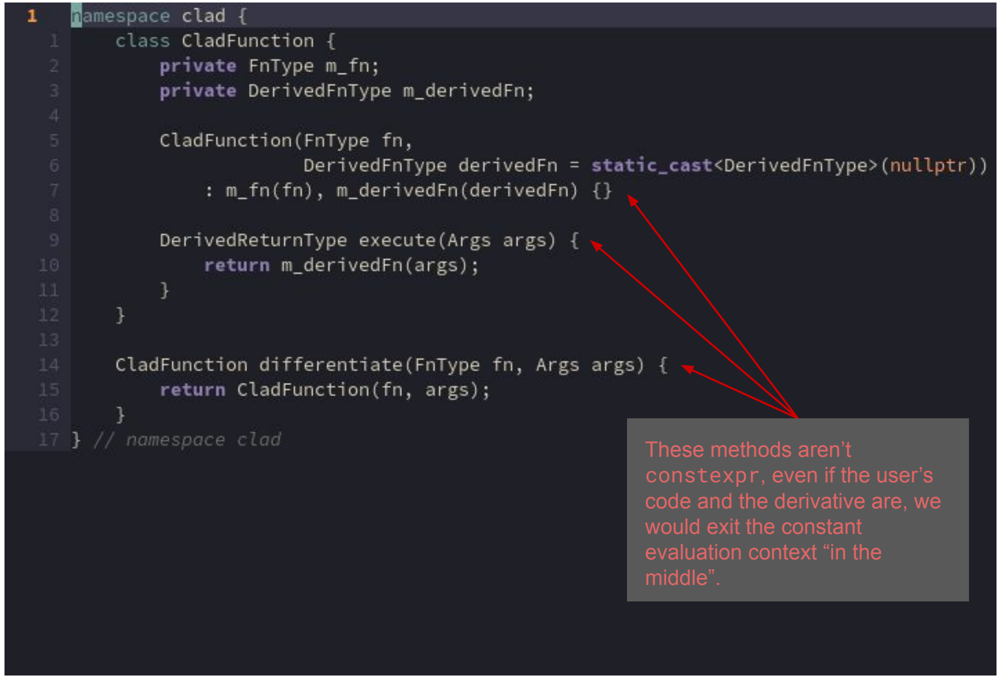
```cpp
#include "clad/Differentiator/Differentiator.h"

constexpr double fn(double x, double y) {
    return x * x + 2 * x * y + y * y;
}

int main() {
    auto d_fn = clad::differentiate(fn, "x");

    printf("%f\n", d_fn.execute(3, 4));
}
```

The `differentiate` and `execute` calls are not in the same constant evaluation context.



```cpp
namespace clad {
    class CladFunction {
        private FnType m_fn;
        private DerivedFnType m_derivedFn;

        CladFunction(FnType fn,
                     DerivedFnType derivedFn = static_cast<DerivedFnType>(nullptr))
            : m_fn(fn), m_derivedFn(derivedFn) {}

        DerivedReturnType execute(Args args) {
            return m_derivedFn(args);
        }
    }

    CladFunction differentiate(FnType fn, Args args) {
        return CladFunction(fn, args);
    }
} // namespace clad
```

These methods aren't `constexpr`, even if the user's code and the derivative are, we would exit the constant evaluation context "in the middle".

Issue one solution

1. Converted methods of `CladFunction` to all be constexpr.

2. Replaced usages of `printf` and `assert` with compile-time friendly alternatives.
   a. Checking if clad.so was loaded was previously done with an `assert`, but I replaced that with a macro, which works at compile-time.
   b. Most of the places that had `printf`’s could be replaced with `static_assert`’s.

Issue two: Clad doesn’t see constexpr functions

Clad works by registering callbacks with clang for the different events during AST generation, but then it delays processing, until the whole AST is available. This is done to support the cases, where the declaration and definition are not in the same top-level decl, but that’s not possible with `constexpr` anyways.

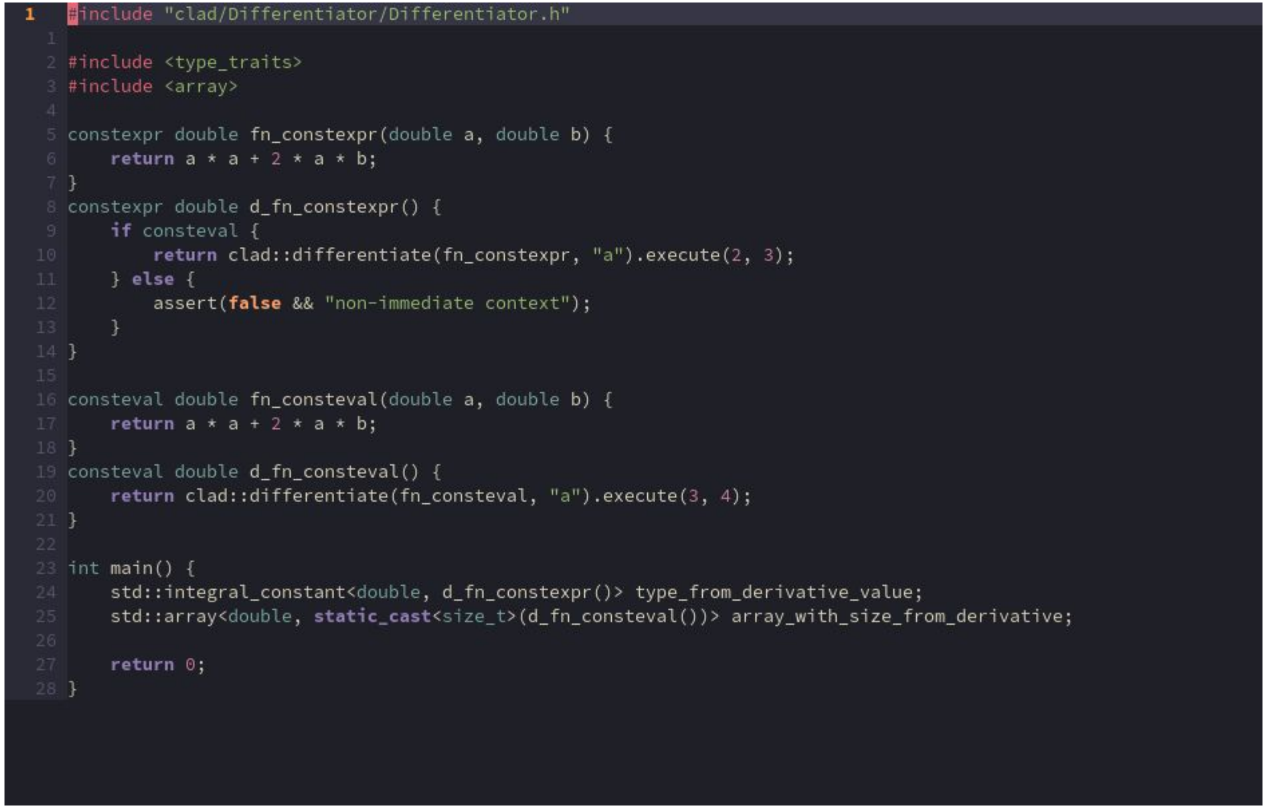
If we tried to run Clad with a `constexpr` function and `constexpr CladFunction` we would see that the `constexpr` function doesn’t even reach Clad, it’s already evaluated and folded.

Issue two solution

1. Figure out how the clang constant expression evaluation works.

2. Modify `HandleTopLevelDecl` to find the nodes in the differentiation graph which are for `constexpr` functions and process those requests sooner.

`constexpr` and `consteval` working



```cpp
#include "clad/Differentiator/Differentiator.h"

#include <type_traits>
#include <array>

constexpr double fn_constexpr(double a, double b) {
    return a * a + 2 * a * b;
}
constexpr double d_fn_constexpr() {
    if consteval {
        return clad::differentiate(fn_constexpr, "a").execute(2, 3);
    } else {
        assert(false && "non-immediate context");
    }
}

consteval double fn_consteval(double a, double b) {
    return a * a + 2 * a * b;
}
consteval double d_fn_consteval() {
    return clad::differentiate(fn_consteval, "a").execute(3, 4);
}

int main() {
    std::integral_constant<double, d_fn_constexpr()> type_from_derivative_value;
    std::array<double, static_cast<size_t>(d_fn_consteval())> array_with_size_from_derivative;

    return 0;
}
```

What’s left to solve

After the former changes constexpr and consteval functions work as expected, but there is still some work left to do:

1. `constexpr` clad::tape
2. Improve the interface for differentiating and executing in constant contexts.
3. Create nicers demos and some more docs.

Questions?