IMPROVING AUTOMATIC DIFFERENTIATION OF OBJECT-ORIENTED PARADIGMS USING CLAD

Daemond Zhang
MOTIVATION TO SUPPORT OOP IN CLAD

• Supporting object oriented paradigms in clad will allow users to compute derivatives to the algorithms in their projects seamlessly.

• Our goal is to finally enable users to realize clad's potential by making it compatible with Softsusy and Eigen libraries codebases.

• I will introduce to you what I plan to do during this summer.
ADD SUPPORT FOR DIFFERENTIATION OF SPECIAL MEMBER FUNCTIONS

```cpp
class Vector {
public:
    double x, y;
    Vector(double x, double y) : x(x), y(y) {}
    Vector(const Vector& v) : x(v.x), y(v.y) {}
};

Vector v1(1.0, 2.0);
Vector v2 = v1;  // Copy constructor is used here
```

- We can perform copy operations on class types similar to those on builtin types, such as "int y = x;" -> "class A y = x".
IMPROVE SUPPORT FOR DIFFERENTIATION OF OPERATOR OVERLOADS

- Currently, clad already supports member functions and the () operator in reverse mode.
- Other operator overloads are similar to this in that they involve an indirect call to member.

```cpp
class Vector {
public:
    double x, y;

    Vector(double x, double y) : x(x), y(y) {}

    Vector operator+(const Vector& rhs) const {
        return Vector(x + rhs.x, y + rhs.y);
    }

    Vector operator-(const Vector& rhs) const {
        return Vector(x - rhs.x, y - rhs.y);
    }

    Vector operator*(double scalar) const {
        return Vector(x * scalar, y * scalar);
    }
};
```
ADD SUPPORT FOR CUSTOM DERIVATIVES FOR SPECIAL MEMBER FUNCTIONS

class Matrix {
public:
    Matrix(size_t rows, size_t cols) { /* implementation omitted */ }
    Matrix(const Matrix& other) { /* copy constructor */ }
    Matrix operator*(const Matrix& other) const { /* matrix multiplication */ }
    Matrix transpose() const { /* transpose operation */ }
    // ... other member functions ...
};

double costFunction(const Matrix& m1, const Matrix& m2) {
    return (m1 * m2.transpose()).norm();
}
ADD SUPPORT FOR CUSTOM DERIVATIVES FOR SPECIAL MEMBER FUNCTIONS

```cpp
class Matrix {
public:
    // ... other functions as before ...

    // Custom derivatives for the special member functions:
    Matrix transpose_pushforward() const { /* custom derivative implementation */ }

    Matrix operator_mult_pushforward(const Matrix& other) const { /* custom derivative */ }

    // ... potentially other custom derivatives ...
};
```
RESEARCH WAYS TO IMPROVE CLAD OBJECT-ORIENTED DIFFERENTIABLE MODEL

```cpp
class Complex {
public:
  double real;
  double imag;

  Complex(double r, double i) : real(r), imag(i) {};

  double magnitude() const {
    return sqrt(real*real + imag*imag);
  }

  // More member functions...
};
```
RESEARCH WAYS TO IMPROVE CLAD OBJECT-ORIENTED DIFFERENTIABLE MODEL

class Complex {
    public:
        double real;
        [[clad::non_differentiable]] double imag; // Marked as non-differentiable

        Complex(double r, double i) : real(r), imag(i) {}

        [[clad::non_differentiable]] // This function won't be differentiated
        double magnitude() const {
            return sqrt(real*real + imag*imag);
        }

        // More member functions...
};

Pitch Deck
THANK YOU!