Develop of Clad Tutorials for CMS/HEP

Applicant: Austėja Jurgaitytė
Mentor: David Lange
Project Duration: 5 weeks

Project Context and Description

Particle tracking is an important part of the processing and analysis of data received from particle detectors, such as the Compact Muon Solenoid (CMS). Tracking is the step that determines the momentum of charged particles escaping from the collision point. It identifies individual particles by reconstructing their trajectories from points where charged particle "hits" were measured by the detector and interpreting them. [1] Due to the Lorentz force, charged particles move in a helical motion when affected by the magnetic field (neglecting other effects due to material interactions, etc). This means we can figure out a specific particle trajectory through the detector by fitting a helix function to data points in such a way that the distance from the data points and the helix would be minimized. In mathematical terms, we need to find optimal helix parameters by minimizing a loss function composed of the least squared distances, thus giving the best estimation of these parameters.

For this purpose we can use Clad - an automatic differentiation tool for C++, [2] to efficiently minimize the loss function. Automatic differentiation is a set of techniques used to evaluate derivatives of functions programmatically. It calculates function derivatives exactly (up to numerical precision limits) and is not confined to closed-form expressions. [3] Automatic differentiation makes use of the chain rule and intermediate variables, calculated using primitive arithmetic operations and elementary functions, present in every computer calculation. [4] There are two main modes of automatic differentiation: forward and reverse accumulations. Forward accumulation (Clad's Forward mode) traverses the chain rule from inside to outside, evaluating the intermediate functions and their derivatives with respect to one independent variable and returns a single numeric output, or in Clad's case, a function [4] [2]. Reverse accumulation (Reverse mode) computes the gradient of a function during the reverse pass, after function evaluation, using the stored dependencies of the expression. [5]

Goals

In this project, we will focus on creating a Clad based demonstration of finding the best fit helix parameters given a set of data points. Given the short timescale of the project, the complexity of the resulting tutorial will depend on complexities encountered during the research. At the same time, during the project, demos and reproducers of various Clad functionalities will be built and tested. Another goal of this project will be to contribute Clad code fixing the missing functionalities that we find along the way, such as adding support for the atan2 function differentiation.

Project timeline

- Study the code of Clad and understand how to use it and its usage limitations. (week 1-2)
- Create a simplified helix class that Clad can differentiate successfully. (week 2)
- Create an objective function that takes a set of data points and the code that finds the helix that best fits the minimum distances to these points with Clad. (week 3-4)
- Find, and fix or report, missing functionalities and bugs. (week 4-5)

References

[1] CERN CMS Experiment Website, Tracking [https://cmsexperiment.web.cern.ch/detector/identifying-tracks]

[2] Clad Github [https://github.com/vgvassilev/clad]

[3] Compiler Research website, Automatic differentiation [https://compiler-research.org/automatic_differentiation]

[4] Wiki Website on Automatic Differentiation [https://en.wikipedia.org/wiki/Automatic_differentiation]

[5] MathWorks website, Automatic Differentiation Background [https://ch.mathworks.com/help/optim/ug/autodiff-background.html]