# Develop of Clad Tutorials for CMS/HEP

Austėja Jurgaitytė
Mentor: David Lange

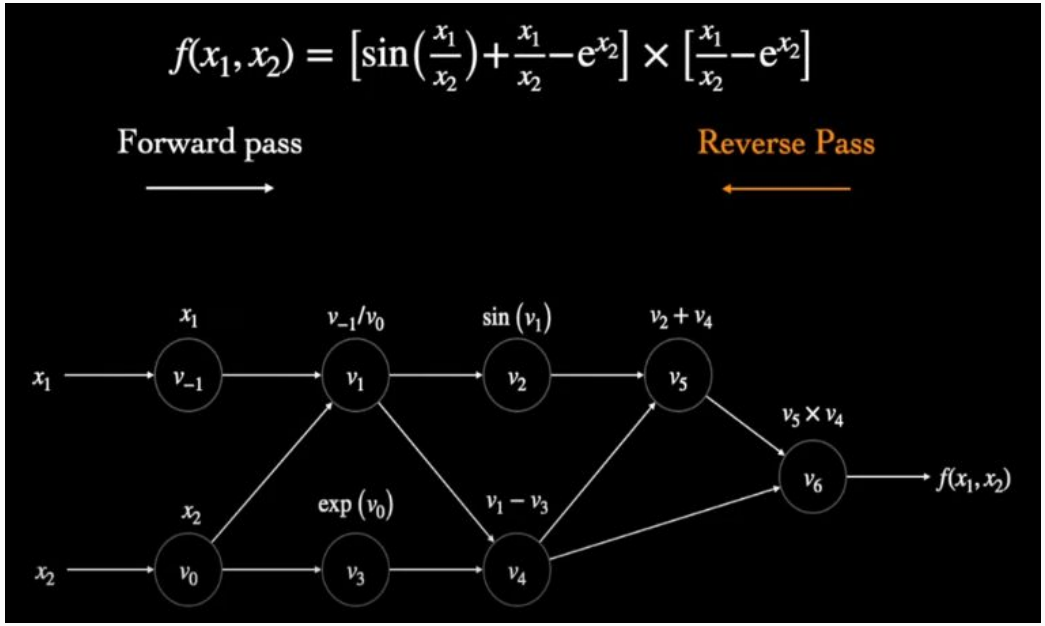
# Automatic differentiation (AD)

- A collection of methods used to compute derivatives of functions programmatically.
- Calculates the derivatives of functions precisely (up to the limits of numerical precision).
- Uses the chain rule and intermediate variables calculated using elementary arithmetic operations and elementary functions found in every computer calculation. [1]

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

[1]

# Types of AD


$f(x_1, x_2) = \left[\sin\left(\frac{x_1}{x_2}\right) + \frac{x_1}{x_2} - e^{x_2}\right] \times \left[\frac{x_1}{x_2} - e^{x_2}\right]$
Forward pass
Reverse Pass
$x_1$
$v_{-1}/v_0$
$\sin(v_1)$
$v_2 + v_4$
$x_1$
$v_{-1}$
$v_1$
$v_2$
$v_5$
$v_5 \times v_4$
$v_6$
$f(x_1, x_2)$
$x_2$
$\exp(v_0)$
$v_1 - v_3$
$x_2$
$v_0$
$v_3$
$v_4$

[2]

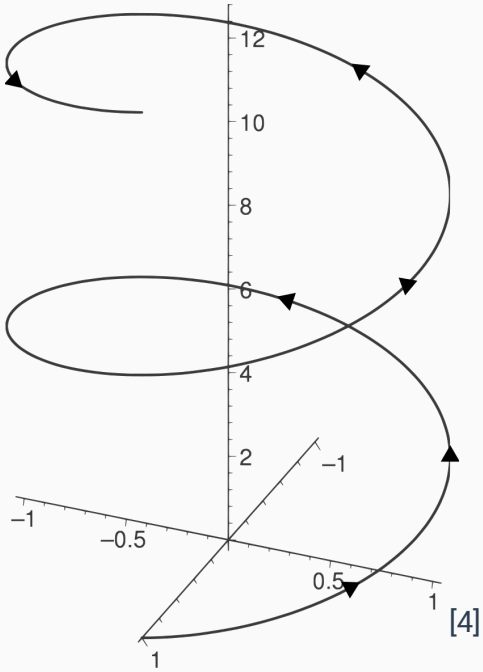
# Clad

Clad is a plugin for Clang compiler that enables automatic differentiation for for C++. When provided with a C++ function, Clad automatically generates code that computes the derivatives of that function.[3]


[3]

# Project goals

- Creating a Clad based demonstration of finding the best fit helix parameters given a set of data points.
- Contribute to Clad code fixing the missing functionalities that we find along the way.



[4]

# The plan for the main tutorial

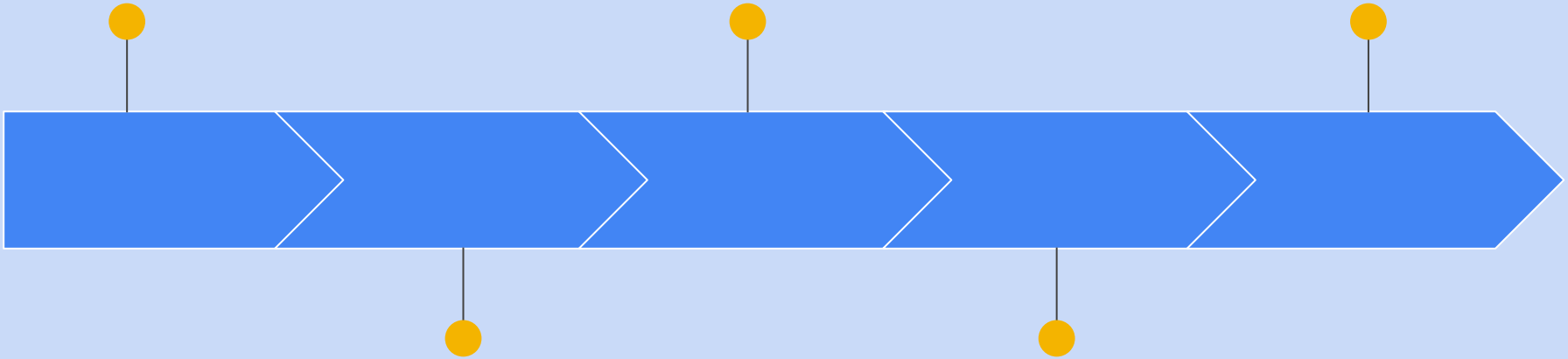
Simple function to describe a point on the helix

Function that generates points for a helix with noise

Calculation of distance from a helix to a point

Finding helix parameters with the Levenberg-Marquardt algorithm

Graph, showing the fitted spiral



# Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm combines two optimization methods: gradient descent and Gauss-Newton.

Its behaviour changes based on how close the current coefficients are to the optimal value.

The equation that dictates how to update the parameters in the Levenberg-Marquardt algorithm is this:

$$(J^T W J + \lambda I)\, h_{lm} = J^T W (y - \hat{y})^{[5]}$$

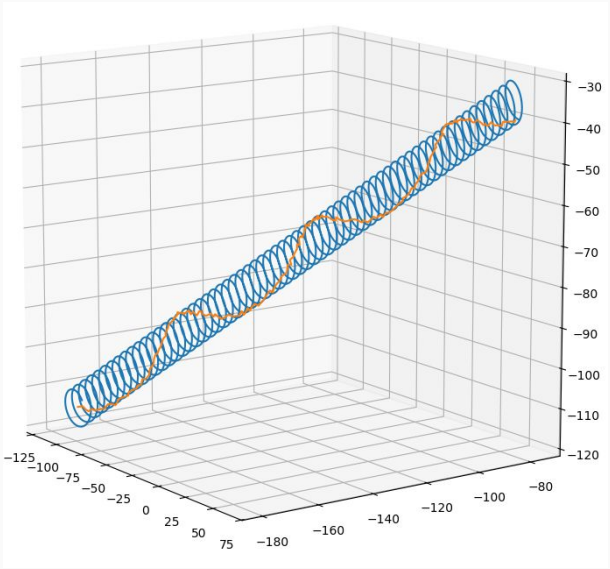
# Distance to point calculations

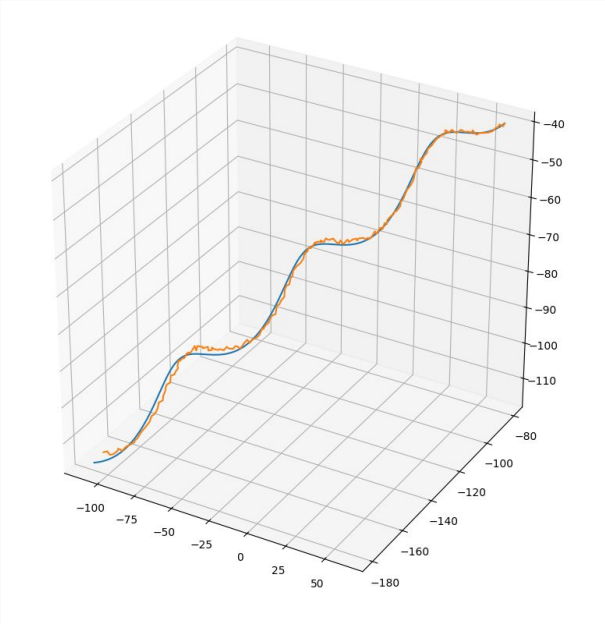
- To find the closest distance of a point to a helix, we do some scaling so that our helix is now defined by $(\cos t, \sin t, \hbar t)$.
- For a given point $P$(i,j,k), let $Q$ be the closest point on the helix. The line segment connecting $P$ and $Q$ must be perpendicular to the helix's tangent line at $Q$, which is just $(-\sin t, \cos t, \hbar)$:

$$-(\cos t - \mathrm{i})\sin t + (\sin t - \mathrm{j})\cos t + (\hbar t - \mathrm{k})\hbar = 0$$

- This simplifies to $A\sin(t+B)+Ct+D=0$ for some constants $A,B,C,D$.[6]
- To find the solution, I perform a binary search.

# Graphs


−30
−40
−50
−60
−70
−80
−90
−100
−110
−120
−125
−100
−75
−50
−25
0
25
50
75
−180
−160
−140
−120
−100
−80


−40
−50
−60
−70
−80
−90
−100
−110
−100
−75
−50
−25
0
25
50
−80
−100
−120
−140
−160
−180

# Gradient Descent

- Perhaps a better way to showcase Clad as it is more simple (but not necessarily a better way to approximate a helix)
- the implementation found in fitter.h gets stuck in a local minimum that is very far off from the actual expected results.

# What I learned

- I gained knowledge about automatic differentiation and Clad.
- Learned more about C++.
- Refreshed my knowledge about various fitting methods.
- Got experience working with a new mentor.
- Got a taste of what it’s like to work with a team.

# References


[1] Automatic Differentiation Wikipedia page, [https://en.wikipedia.org/wiki/Automatic_differentiation]

[2] „What is Automatic Differentiation?“, [https://www.youtube.com/watch?v=wG_nF1awSSY]

[3] Clad GitHub, [https://github.com/vgvassilev/clad]

[4] Helix Wikipedia page, [https://en.wikipedia.org/wiki/Helix]

[5] The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems, https://people.duke.edu/~hpgavin/lm.pdf

[6] Shortest distance between a point and a helix, [https://math.stackexchange.com/questions/13341/shortest-distance-between-a-point-and-a-helix]

**Project GitHub** [https://github.com/compiler-research/helix-example]