UNIVERSITY OF CAMBRIDGE

C1 Research Computing - Coursework Assignment

Raunaq Rai (rsr45@cam.ac.uk)
Department of Physics, University of Cambridge

18 December, 2024

1 Introduction

This report details the development and implementation of a package designed for automatic differentiation using dual numbers. The package, named `dual_autodiff`, allows users to compute derivatives efficiently while also supporting various mathematical operations such as trigonometric, logarithmic, and exponential functions.

The main feature of `dual_autodiff` is its ability to leverage dual numbers for calculating derivatives in a computationally efficient and user-friendly manner. Additionally, a Cython-optimized version of the package, `dual_autodiff_x`, provides enhanced performance for intensive computations.

This document covers the structure of the project, the implementation details, and the mathematical principles underpinning the use of dual numbers for differentiation.

2 Theory of Dual Numbers and Their Use in Automatic Differentiation

2.1 Dual Numbers

Dual numbers can be defined as truncated Taylor series of the form:

$$x = v + \dot{v}\epsilon,$$

where $v, \dot{v} \in \mathbb{R}$, and ϵ is a nilpotent number such that $\epsilon^2 = 0$ and $\epsilon \neq 0$. In this representation: - v is referred to as the *primal value.* - $\dot{v}$ represents the *derivative* or *tangent value.*

As discussed by Baydin et al. [1], arithmetic operations with dual numbers are designed to mirror symbolic differentiation rules. For example:

$$(x_1 + \dot{x}_1\epsilon) + (x_2 + \dot{x}_2\epsilon) = (x_1 + x_2) + (\dot{x}_1 + \dot{x}_2)\epsilon,$$

$$(x_1 + \dot{x}_1\epsilon)(x_2 + \dot{x}_2\epsilon) = x_1x_2 + (x_1\dot{x}_2 + \dot{x}_1x_2)\epsilon.$$

These operations align naturally with differentiation principles, where the coefficients of ϵ represent the derivatives of the associated functions.

3 Historical Context of Automatic Differentiation

The foundations of automatic differentiation (AD) were laid in 1964 when R.E. Wengert published his influential paper, *A Simple Automatic Derivative Evaluation Program* [2]. Wengert's work introduced a systematic approach for computing derivatives of complex functions numerically, eliminating the need for labor-intensive manual derivations of analytical expressions.

Wengert's method was motivated by the challenges engineers and scientists faced when working with intricate mathematical models. Analytical differentiation was often cumbersome, especially for large systems of equations. Numerical differentiation, though simpler, was prone to errors due to finite precision. Wengert addressed these issues by proposing a method that broke down complex functions into a series of simpler, intermediate steps.

The key innovation in Wengert's method was the use of intermediate variables to represent elementary operations. By systematically tracking these variables and their contributions, Wengert demonstrated how derivatives could be computed as a natural by-product of evaluating the function itself. This approach provided a practical and efficient solution, making it easier to compute both total and partial derivatives for arbitrary algebraic functions.

3.1 Legacy and Impact

Wengert's method marked the beginning of automatic differentiation as a distinct field. His approach simplified derivative computation, enabling engineers and researchers to handle complex systems with ease. Over time, these principles evolved into the modern AD tools used today, which play a crucial role in fields such as machine learning, optimization, and computational physics.

3.2 Automatic Differentiation with Dual Numbers

Automatic differentiation can be achieved by evaluating functions using dual numbers. For a function $f(x)$, substituting $x = v + \dot{v}\epsilon$ yields:

$$f(x) = f(v + \dot{v}\epsilon) = f(v) + f'(v)\dot{v}\epsilon,$$

where: - $f(v)$ is the function value. - $f'(v)\dot{v}$ represents the derivative scaled by $\dot{v}$.

To compute the derivative of f, the input x is represented as $x = v + \epsilon$, where $\dot{v} = 1$. This ensures that the derivative of f at v is embedded in the coefficient of ϵ in the result.

As Wengert's early work [2] and Baydin et al. [1] explain, the chain rule naturally follows for compositions of functions:

$$f(g(v + \dot{v}\epsilon)) = f(g(v)) + f'(g(v))g'(v)\dot{v}\epsilon.$$

This demonstrates that the derivative of the composition is embedded in the coefficient of ϵ.

3.3 Practical Implementation

In practice, dual numbers are used to evaluate functions, allowing simultaneous computation of function values and derivatives. A real number x is represented as a dual number

$x = v + \dot{v}\epsilon$, where:

- v is the value of the function input.
- $\dot{v}$ is initialized to 1 to compute the derivative with respect to x.

For example, to compute the derivative of $\sin(x)$ at $x = 1$, substitute $x = 1 + \epsilon$:

$$\sin(1 + \epsilon) = \sin(1) + \cos(1)\epsilon.$$

Here:

- $\sin(1)$ is the function value at $x = 1$.
- $\cos(1)$, the coefficient of ϵ, represents the derivative of $\sin(x)$ at $x = 1$.

This process demonstrates how dual numbers encode both the function value and its derivative in a single representation, eliminating the need for separate derivative computations.

3.4 Applications in Programming

Dual numbers enable seamless automatic differentiation through operator overloading. By augmenting basic arithmetic and mathematical operations to respect dual arithmetic rules, derivatives can be computed alongside function evaluations. This transparency allows: - Simultaneous computation of function values and derivatives. - Integration into arbitrary program constructs, as dual numbers can exist within any data structure.

The `dual_autodiff` package uses dual numbers to implement forward-mode AD, enabling efficient computation of derivatives for scientific and numerical tasks. Through operator overloading, users can compute derivatives by writing functions in a natural, mathematical syntax.

References

[1] Atılım Güneş Baydin et al. "Automatic Differentiation in Machine Learning: A Survey". In: *Journal of Machine Learning Research* 18 (2018). Submitted 8/17; Published 4/18, pp. 1–43. URL: `https://jmlr.org/papers/v18/17-468.html`.

[2] R. E. Wengert. "A Simple Automatic Derivative Evaluation Program". In: *Communications of the ACM* 7.8 (1964), pp. 463–464. URL: `https://dl.acm.org/doi/10.1145/355586.364791`.