1 Introduction

This report discusses development of the package designed to implement automatic differentiation using dual numbers in Python.

2 Repository

2.1 Structure

Good practice suggests the following directory structure:

```
dual_autodiff
├── report
│   └── ...
├── dual_autodiff
│   ├── __init__.py
│   └── dual.py
├── tests
│   └── test_dual.py
├── docs
│   └── ...
├── build
│   └── ...
├── source
│   └── ...
├── pyproject.toml
├── setup.py
├── README.md
├── Makefile
└── LICENSE
```

2.2 Project configuration

The following is the contents of the `pyproject.toml` file.

Listing 1: pyproject.toml

```
[build-system]
requires = ["setuptools", "wheel", "setuptools_scm"]
build-backend = "setuptools.build_meta"

[project]
name = "dual_autodiff"
version = "0.0.1"
description = "A Python package for automatic
   differentiation using dual numbers."
readme = "README.md"
```

```
requires-python = ">=3.10"
license = { file = "LICENSE" }
authors = [
    { name = "Laura Just Fung", email = "lj441@cam.ac.uk"
       }
]

dependencies = [
    "numpy",
    "mpmath"
]

[tool.setuptools_scm]
write_to = "dual_autodiff/version.py"
version_scheme = "post-release"
local_scheme = "no-local-version"

[tool.setuptools.packages.find]
where = ["."]

[project.urls]
Homepage = "https://github.com/ljf441/dual_autodiff"
Issues = "https://github.com/ljf441/dual_autodiff/issues"
```

3 Implementation

3.1 Dual class

The `Dual` class is defined and initialised like so:

Listing 2: Dual class

```
import mpmath as mp
import numdifftools as nd
    """
            dual (int, float): dual part of dual number.
        """
        self.real = real
        self.dual = dual
```

A dual number can be initialised as follows:

Cell 02

```
# initialise dual number
x = df.Dual(2, 1)
print(f"x.real = {x.real}, x.dual = {x.dual}")
```

```
x.real = 2, x.dual = 1
```

And it produces the following output when printed:

Cell 03

```
# printing dual number
print(x)
```

```
Dual(real=2, dual=1)
```

3.2 Arithmetic operations

Arithmetic operations include: addition, subtraction, multiplication, and division. The `Dual` class allows for these arithmetic operations performed over dual numbers as well as real numbers. An example is as follows:

Cell 04

```
# basic arithmetic operations

x = df.Dual(2, 1)
y = df.Dual(3, 2)

print(f"x + y = {x + y}")
print(f"x - y = {x - y}")
print(f"x * y = {x * y}")
print(f"x / y = {x / y}")
```

```
x + y = Dual(real=5, dual=3)
x - y = Dual(real=-1, dual=-1)
x * y = Dual(real=6, dual=7)
x / y = Dual(real=0.6666666666666666, dual=-0.25)
```

3.3 Trigonometric operations

Trigonometric operations include: sine, cosine, and tangent. The `Dual` class allows for these trigonometric operations performed over dual numbers. An example is as follows:

Cell 05

```
# trigonometric operations

print(f"sin(x) = {x.sin()}")
print(f"cos(x) = {x.cos()}")
print(f"tan(x) = {x.tan()}")
```

```
sin(x) = Dual(real=0.9092974268256817, dual=-0.4161468365471424)
cos(x) = Dual(real=-0.4161468365471424, dual=-0.9092974268256817)
tan(x) = Dual(real=-2.185039863261519, dual=5.774399204041917)
```

3.4 Other operations

Other implemented operations that the `Dual` class allows for over dual numbers are: natural logarithm, exponential, power. An example is as follows:

Cell 06

```
# the natural logarithm and exponential

print(f"log(x) = {x.log()}")
print(f"exp(x) = {x.exp()}")
```

```
log(x) = Dual(real=0.6931471805599453, dual=0.5)
exp(x) = Dual(real=7.38905609893065, dual=7.38905609893065)
```

Cell 07

```
# powers

print(f"x**y = {x**y}")
```

```
x**y = Dual(real=8, dual=35.090354888959126)
```

4 Package

The package is installable with `pip install -e .` from the root project folder. The package is importable with `import dual_autodf as df`.

5 Differentiation

Consider the function $f(x) = \log \sin x + x^2 \cos x$ where $x = 1.5$. The derivative can be easily calculated using `dual_autodiff`:

Cell 08

```
# automatic differentiation

#initialise x = 1.5, with x.dual = 1 to allow for differentiation
x = df.Dual(1.5, 1)

function = x.sin().log() + x**2 * x.cos() #f(x) = log(sin(x)) + x^2 * cos(x)

print(f"log(sin(x)) + x^2 cos(x) = {function.real}")
print(f"d/dx(log(sin(x)) + x^2 cos(x)) = {function.dual}")
```

```
log(sin(x)) + x^2 cos(x) = 0.15665054756073515
d/dx(log(sin(x)) + x^2 cos(x)) = -1.9612372705533612
```

6 Test suite