D. Automatic Differentiation

In Chapter 5, we explored numerical differentiation through finite difference methods. These methods approximate derivatives by evaluating the function at different points and calculating differences. While straightforward to implement, they suffer from two main limitations:

1. **Truncation error**: As we've seen, the error scales with some power of the step size h
2. **Round-off error**: As h gets extremely small, floating-point arithmetic leads to precision loss

Automatic differentiation (AD) is a different approach that computes derivatives exactly (to machine precision) without relying on finite differences. AD leverages the chain rule and the fact that all computer programs, no matter how complex, ultimately break down into elementary operations (addition, multiplication, sin, exp, etc.) whose derivatives are known.

Let's introduce the concept of a **computation graph**, which is fundamental to understanding automatic differentiation.

A computation graph represents a mathematical function as a directed graph where:

- **Nodes** represent variables (inputs, outputs, or intermediate values)
- **Edges** represent dependencies between variables
- Each node performs a simple operation with known derivatives

---

Example D.1. Consider the function $f(x, y) = x^2 y + \sin(xy)$. We can break this down into elementary operations and visualize it as a computation graph:

This computation graph for $f(x, y) = x^2 y + \sin(xy)$ at $(x, y) = (2, 3)$ shows:

1. Input nodes: $x = 2$, $y = 3$
2. Intermediate computations:
    - $v_1 = x^2 = 4$
    - $v_3 = xy = 6$
    - $v_2 = v_1 y = 12$

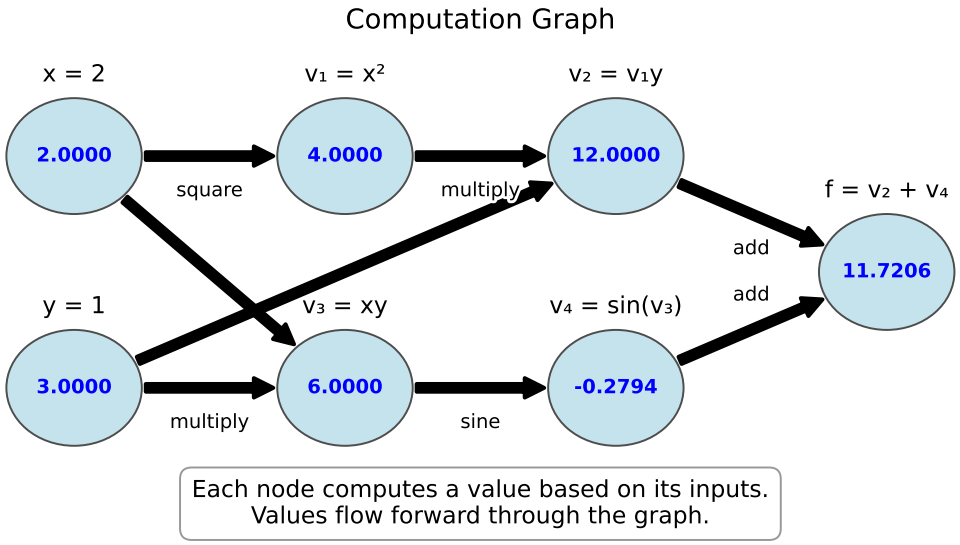

Figure D.1.: Computation graph for f(x,y) = $x^2y + \sin(xy)$ at (x,y) = (2,1)

- $v_4 = \sin(v_3) = \sin(6) \approx -0.2794$

3. Output node: $f = v_2 + v_4 = 12 + (-0.2794) \approx 11.7206$

For each operation in the computation graph, we know:

1. How to compute the function value (forward evaluation)
2. How to compute the derivative of the operation with respect to its inputs

The computation graph is the foundation for automatic differentiation:

- **Forward mode AD**: Derivatives flow through the graph in the same direction as function evaluation
- **Reverse mode AD**: Function values flow forward through the graph, derivatives flow backward

This will become clearer in sections Section D.1 and Section D.2.

Exercise D.1. 1. Draw the computation graph for the function $f(x) = x^2 \sin(x)$.

2. For the function $h(x, y, z) = xy + yz + zx$, identify:
   - The input nodes
   - The intermediate nodes and their operations
   - The output node

3. Explain why breaking a complex function into a computation graph of elementary operations is useful for derivative computation.

D.1. Forward Mode AD

In forward mode automatic differentiation, we track both the values of variables and their derivatives with respect to the input variables. This allows us to build the derivatives as we compute the function value.

Example D.2. Consider a simple function $f(x) = x^2 \cdot \sin(x)$. We can compute both the value and the derivative at $x = 2$ as follows:

1. Initialize: $x = 2$, $\frac{dx}{dx} = 1$ (The derivative of x with respect to itself is 1)
2. Compute $u = x^2$:
   - Value: $u = 2^2 = 4$
   - Derivative: $\frac{du}{dx} = \frac{d(x^2)}{dx} \cdot \frac{dx}{dx} = 2x \cdot 1 = 2 \cdot 2 = 4$
3. Compute $v = \sin(x)$:
   - Value: $v = \sin(2) \approx 0.9093$
   - Derivative: $\frac{dv}{dx} = \frac{d\sin(x)}{dx} \cdot \frac{dx}{dx} = \cos(x) \cdot 1 = \cos(2) \approx -0.4161$
4. Compute $f = u \cdot v$:
   - Value: $f = 4 \cdot 0.9093 \approx 3.6372$
   - Derivative: $\frac{df}{dx} = \frac{d(u \cdot v)}{dx} = \frac{du}{dx} \cdot v + u \cdot \frac{dv}{dx} = 4 \cdot 0.9093 + 4 \cdot (-0.4161) \approx 1.9728$

This is **forward mode** automatic differentiation.

The figure below illustrates the computational graph for $f(x) = x^2 \cdot \sin(x)$ with forward mode AD. The blue values show the function evaluation, while the red values show the derivative calculation:

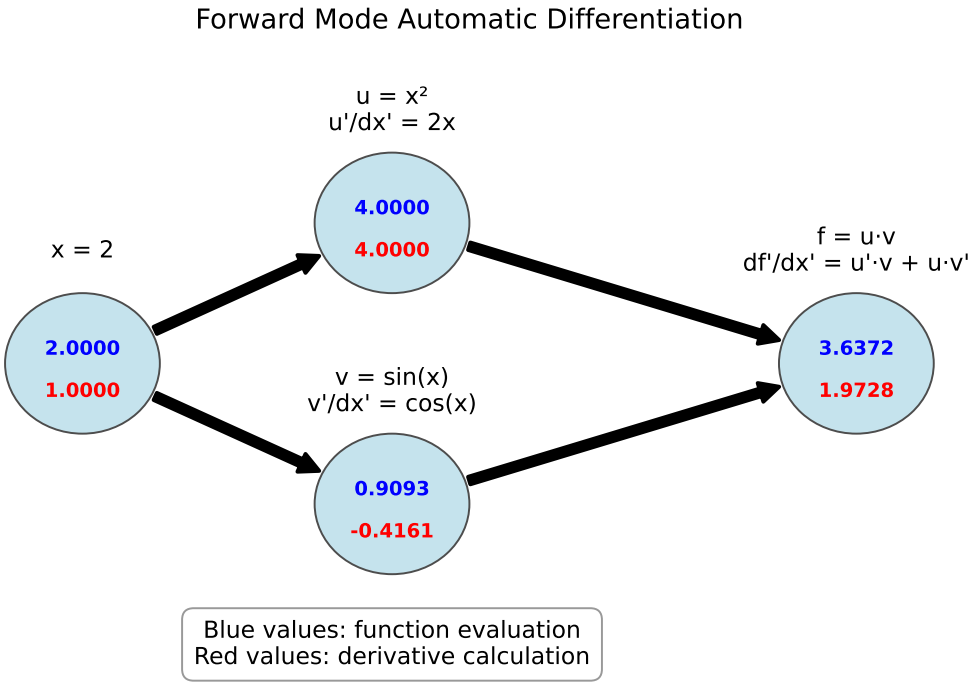

Figure D.2.: Forward mode automatic differentiation for f(x) = x² · sin(x) at x = 2

Let's formalize the forward mode automatic differentiation process by creating a systematic approach. We'll use the concept of a "dual number" that carries both a value and its derivative.

We will represent each variable as a pair $\begin{pmatrix} v \\ d \end{pmatrix}$ where v is the variable's value and d is its derivative with respect to the input we're differentiating against.

Consider these basic operations and their dual number representations:

1. Addition: $\begin{pmatrix} a \\ a' \end{pmatrix} + \begin{pmatrix} b \\ b' \end{pmatrix} = \begin{pmatrix} a+b \\ a'+b' \end{pmatrix}$
2. Multiplication: $\begin{pmatrix} a \\ a' \end{pmatrix} \cdot \begin{pmatrix} b \\ b' \end{pmatrix} = \begin{pmatrix} a \cdot b \\ a' \cdot b + a \cdot b' \end{pmatrix}$
3. Division: $\begin{pmatrix} a \\ a' \end{pmatrix} / \begin{pmatrix} b \\ b' \end{pmatrix} = \begin{pmatrix} a/b \\ (a' \cdot b - a \cdot b')/b^2 \end{pmatrix}$
4. Power: $\begin{pmatrix} a \\ a' \end{pmatrix}^n = \begin{pmatrix} a^n \\ n \cdot a^{n-1} \cdot a' \end{pmatrix}$
5. Sine: $\sin \begin{pmatrix} a \\ a' \end{pmatrix} = \begin{pmatrix} \sin(a) \\ \cos(a) \cdot a' \end{pmatrix}$

6. Exponential: $e^{\begin{pmatrix} a \\ a' \end{pmatrix}} = \begin{pmatrix} e^a \\ e^a \cdot a' \end{pmatrix}$

With this notation the calculations from Example D.2 can be written as:

$$\begin{pmatrix} x \\ x' \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$
$$\begin{pmatrix} u \\ u' \end{pmatrix} = \begin{pmatrix} x \\ x' \end{pmatrix}^2 = \begin{pmatrix} x^2 \\ 2xx' \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$
$$\begin{pmatrix} v \\ v' \end{pmatrix} = \sin \begin{pmatrix} x \\ x' \end{pmatrix} = \begin{pmatrix} \sin(x) \\ \cos(x)x' \end{pmatrix} = \begin{pmatrix} 0.9093 \\ -0.4161 \end{pmatrix}$$
$$\begin{pmatrix} f \\ f' \end{pmatrix} = \begin{pmatrix} u \\ u' \end{pmatrix} \cdot \begin{pmatrix} v \\ v' \end{pmatrix} = \begin{pmatrix} u \cdot v \\ u' \cdot v + u \cdot v' \end{pmatrix} = \begin{pmatrix} 3.6372 \\ 1.9728 \end{pmatrix}$$

Exercise D.2. Compute the derivative of using the dual number approach:

b. $f(x) = \frac{x^2+1}{2x-3}$ at $x = 2$

Verify your result by computing the derivative analytically and comparing.

Next let us teach Python how to do this.

```
(np.float64(3.637189707302727), np.float64(1.9726023611141572))
```

```
(np.float64(3.637189707302727), np.float64(1.9726023611141572))
```

For multivariate functions, we can compute one directional derivative at a time.
For a function $f(x, y)$, to compute $\frac{\partial f}{\partial x}$, we initialize:

- $x = \begin{pmatrix} x \\ 1 \end{pmatrix}$ (value and derivative of x with respect to x)
- $y = \begin{pmatrix} y \\ 0 \end{pmatrix}$ (value and derivative of y with respect to x)

And to compute $\frac{\partial f}{\partial y}$, we initialize:

- $x = \begin{pmatrix} x \\ 0 \end{pmatrix}$ (value and derivative of x with respect to y)
- $y = \begin{pmatrix} y \\ 1 \end{pmatrix}$ (value and derivative of y with respect to y)

Exercise D.3. 1. For $f(x, y) = x^2 y + \sin(xy)$, compute both $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at $(x, y) = (2, 1)$ using forward mode AD with the dual number approach.

2. For $f(x, y, z) = x^2 y + y \sin(z) + z \cos(x)$, compute $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, and $\frac{\partial f}{\partial z}$ at $(x, y, z) = (1, 1, 1)$.
3. If a function has n inputs and we want all partial derivatives, how many forward mode passes do we need? What implications does this have for functions with many inputs?

D.2. Reverse Mode AD

Now let's explore **reverse mode** automatic differentiation, which is more efficient for functions with many inputs and few outputs. Reverse mode AD first computes the function value and then propagates derivatives backward from the output to the inputs.

For this exercise, we'll introduce the concept of **adjoints** or **accumulated gradients**. The adjoint of a variable v is denoted $\overline{v}$ and represents $\frac{\partial f}{\partial v}$ where f is the final output.

Example D.3. Let's trace through the reverse mode process for $f(x, y) = x^2 y + \sin(xy)$ at $(x, y) = (2, 3)$:

1. Define intermediate variables in the computation graph as in Figure D.1:
    - $v_1 = x^2 = 4$
    - $v_2 = v_1 \cdot y = 4 \cdot 3 = 12$
    - $v_3 = x \cdot y = 2 \cdot 3 = 6$
    - $v_4 = \sin(v_3) = \sin(6) \approx -0.2794$
    - $f = v_2 + v_4 = 12 + (-0.2794) \approx 11.7206$ (final output)
2. Initialize the adjoint of the output: $\overline{f} = 1$
3. Propagate adjoints backward using the chain rule:
    - $\overline{v_4} = \overline{f} \cdot \frac{\partial f}{\partial v_4} = 1 \cdot 1 = 1$

- $\overline{v_3} = \overline{v_4} \cdot \frac{\partial v_4}{\partial v_3} = 1 \cdot \cos(v_3) = \cos(6) \approx 0.9602$
- $\overline{v_2} = \overline{v_5} \cdot \frac{\partial f}{\partial v_2} = 1 \cdot 1 = 1$
- $\overline{v_1} = \overline{v_2} \cdot \frac{\partial v_2}{\partial v_1} = 1 \cdot y = 3$
- $\overline{x} = \overline{v_1} \cdot \frac{\partial v_1}{\partial x} + \overline{v_3} \cdot \frac{\partial v_3}{\partial y} = 3 \cdot 2x + 0.9602 \cdot y = 3 \cdot 2 \cdot 2 + 0.9602 \cdot 3 \approx 14.8806$
- $\overline{y} = \overline{v_2} \cdot \frac{\partial v_2}{\partial y} + \overline{v_3} \cdot \frac{\partial v_3}{\partial y} = 1 \cdot v_1 + 0.9602 \cdot x = 1 \cdot 4 + 0.9602 \cdot 2 \approx 5.9204$

4. The final results are $\frac{\partial f}{\partial x} = \overline{x} \approx 14.8806$ and $\frac{\partial f}{\partial y} = \overline{y} \approx 5.9204$

Figure D.3 illustrates the reverse mode AD computation graph for this example.

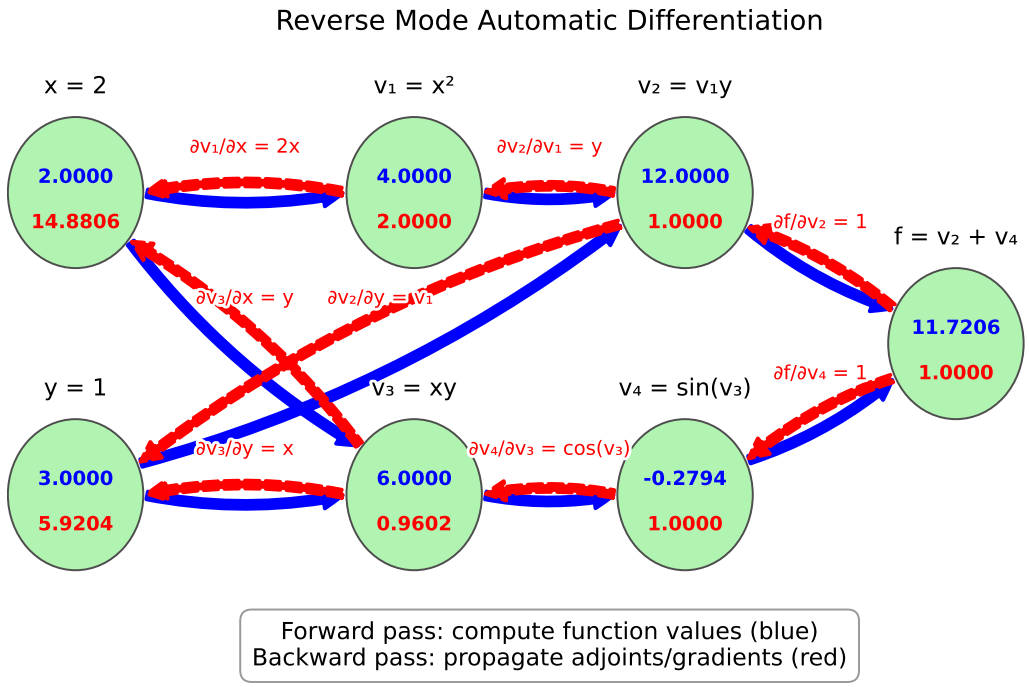

Figure D.3.: Reverse mode automatic differentiation for f(x,y) = x^2y + sin(xy) at (x,y) = (2,1)

Exercise D.4. Trace through the reverse mode process for:

a. $f(x, y) = e^{xy}$ at $(x, y) = (0, 1)$.

b. $f(x) = xy^2 \cos(xy^2)$ at $(x, y) = (\pi, 3)$.

c. $f(x, y, z) = x^2 + y^2 + z^2 + xy + yz + xz$ at $(x, y, z) = (1, 1, 1)$.

Let's now look at how to use JAX, a library that provides automatic differentiation in Python. JAX is designed to be simple to use while providing powerful capabilities. Unlike our finite-difference methods that had truncation errors, JAX computes the derivative exactly (to machine precision).

Example D.4. Here's a simple example using JAX to compute derivatives:

```
import jax
import jax.numpy as jnp

# Define a function
def f(x):
    return jnp.sin(x) * (1 - x)

# Compute the derivative function
df = jax.grad(f)

# Evaluate at x = 1
print(f"f(1) = {f(1)}")
print(f"f'(1) = {df(1.0)}")
```

Two things to note:

- We had to use the jax versions of NumPy functions, like `jnp.sin` instead of `np.sin`.
- We had to pass a float to the `df` function, rather than an integer. So we had to write `df(1.0)` instead of `df(1)`.

Exercise D.5. Use JAX to compute the derivatives of:

a. $f(x) = x^3 - 2x^2 + 4x - 7$

b. $f(x) = \sin(x)\exp(x)$

c. $f(x) = \frac{x^2+1}{2x-3}$

Check that you get the same results as in Exercise D.2.

Example D.5. For multivariate functions, JAX allows us to compute partial derivatives:

```
def g(x, y):
    return x**2 * y + jnp.sin(x * y)

# Compute partial with respect to first argument (x)
dg_dx = jax.grad(g, argnums=0)
```

```
# Compute partial with respect to second argument (y)
dg_dy = jax.grad(g, argnums=1)

# Evaluate at (2, 1)
print(f" g/ x at (2,1) = {dg_dx(2.0, 1.0)}")
print(f" g/ y at (2,1) = {dg_dy(2.0, 1.0)}")
```

Exercise D.6. Use this approach to compute the partial derivatives of:

a. $f(x, y) = x \exp(y + x) + y \sin(xy)$

b. $f(x, y, z) = x^2 y + y \sin(z) + z \cos(x)$

JAX can also compute higher-order derivatives:

```
# Second derivative
d2f = jax.grad(jax.grad(f))
print(f"f''(1) = {d2f(1.0)}")

# Or more concisely
d2f = jax.hessian(f)
print(f"f''(1) = {d2f(1.0)}")
```

Exercise D.7. Compare the accuracy and efficiency of the differentiation methods we've studied:

1. For the function $f(x) = \sin(x)(1 - x)$, compute the derivative at $x = 1$ using:

    a. Forward difference with step sizes $h = 0.1, 0.01, 0.001$

    b. Central difference with step sizes $h = 0.1, 0.01, 0.001$

    c. JAX automatic differentiation

    Create a table giving the value of the derivative and the absolute error for each of these 7 cases.

D.3. Algorithm Summaries

Exercise D.8. Explain how to define arithmetic operations on dual numbers and how to use them to compute the derivative of a function with forward mode automatic differentiation.

Exercise D.9. Given a computation graph of a function, explain how to accumulate the gradients in a backwards pass through the computation graph. What is the advantage of this method over forward mode automatic differentiation?
