Starting to Learn
Automatic Differentiation:
A Case Study

H. Martin Bücker, Daniel Walther
Mathematics and Computer Science
Friedrich Schiller University Jena
Outline

- Toy problem
- Efficient implementation
- First attempt of AD transformation
- Second attempt of AD transformation
- Where is the trap?
Toy Problem: Differentiate \( \sin() \)

\[ n: \text{Number of samples} \]
An Inefficient Implementation

Function that generates simple sinusoidal signal

```matlab
function y = SimpleSinus(A, f0, fn, n)
    y = zeros(1, n);
    w = 2.0 * pi * f0 / fn;
    for i = 1 : n
        y(i) = A * sin(i * w);
    end
end
```

A: Amplitude
f0: Signal frequency
fn: Sampling frequency
n: Number of samples
Performance Optimization

- Fact: Usage of `sin()` is inefficient for generating signals

- Idea: mapping to discrete Laplace space by using Z-transform

\[ f(n\omega) = \sin(n\omega) \overset{Z}{\longrightarrow} \frac{z^{-1} \cdot \sin(\omega)}{1 - z^{-1} \cdot 2 \cos(\omega) + z^{-2}} \]
• Transformed function represents transfer function of oscillating IIR filter that generates sinusoidal signal of interest

\[
H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-1} \cdot \sin(\omega)}{1 - z^{-1} \cdot 2 \cos(\omega) + z^{-2}}
\]

• As difference equation:

\[
y[n] - y[n - 1]2 \cos(\omega) + y[n - 2] = x[n - 1] \sin(\omega)
\]

• Set input \(x[\cdot]\) to 0 and solve for \(y[n]\)

\[
y[n] = y[n - 1]2 \cos(\omega) - y[n - 2]
\]
An Efficient Implementation

Based on difference equation, avoid sin() in loop

```matlab
function y = OptimSinus(A, f0, fn, n)
    y = zeros(1, n);
    w = 2.0 * pi * f0 / fn;
    k = 2.0 * cos(w);
    % Set up initial values
    y2 = A * sin(-(2.0 * w));
    y1 = A * sin(-(1.0 * w));
    % Use only single multiplication
    % and single subtraction in loop body
    for i = 1 : n
        y0 = k * y1 - y2;
        y2 = y1;
        y1 = y0;
        y(i) = y0;
    end
end
```
A = 1.0; f0 = 20; fn = 20000; n = 5000;
y = OptimSinus(A, f0, fn, n);

Function returns a sinusoidal signal as desired

How can AD be used to compute cosine?
First Attempt of AD transformation

```matlab
function y = OptimSinus(A, f0, fn, n)
...
for i = 1 : n
    y0 = k * y1 - y2;
    y2 = y1;
    y1 = y0;
    y(i) = y0;
end
end
```

Time is somehow represented by \( n \)

Specify \( n \) as independent variable
Second Attempt of AD transformation

- Rather than in time domain, OptimSinus() works in frequency domain

- Specify sample frequency $fn$ as independent variable!

```matlab
A = 1.0; f0 = 20; fn = 20000; n = 5000;
y = OptimSinus(A,f0,fn,n);
```

Simple-to-Use interface of AD tool ADiMat:

```matlab
admOpt = admOptions('dependents',1,'independents',3);
[J,y] = admDiffFor(@OptimSinus,1,A,f0,fn,n,admOpt);
```
Numerical Results

- **original signal**
- **differentiated signal**

- “Cosine-like”
- Increasing amplitude
- Wrong sign
Where is the Trap?

Think about it:

• Original function \( f(n\omega) \) works in time domain
• Improved function works in frequency domain because it was transformed via Z-transform

• We are looking for: \( \frac{\partial f(n\omega)}{\partial n} \)

• But we get: \( \frac{\partial Z\{f(n\omega)\}}{\partial z} =: \frac{\partial F(z)}{\partial z} \neq \frac{\partial f(n\omega)}{\partial n} \)
Where is the Trap?

The derivative of a Z-transformed function satisfies

\[ nf(n\omega) \cdot - z \frac{\delta F(z)}{\delta z} \]

Because AD tool can not know rules of differentiation in Z-transformed domain, numerical results will never fulfill expectations!

AD tool developers: Take an AD user’s perspective
AD Tool: ADiMat

www.adimat.de

References


Thank you for your attention