Pitfalls of Automatic Differentiation

UCSD CSE 291 Differentiable Programming

Tzu-Mao Li

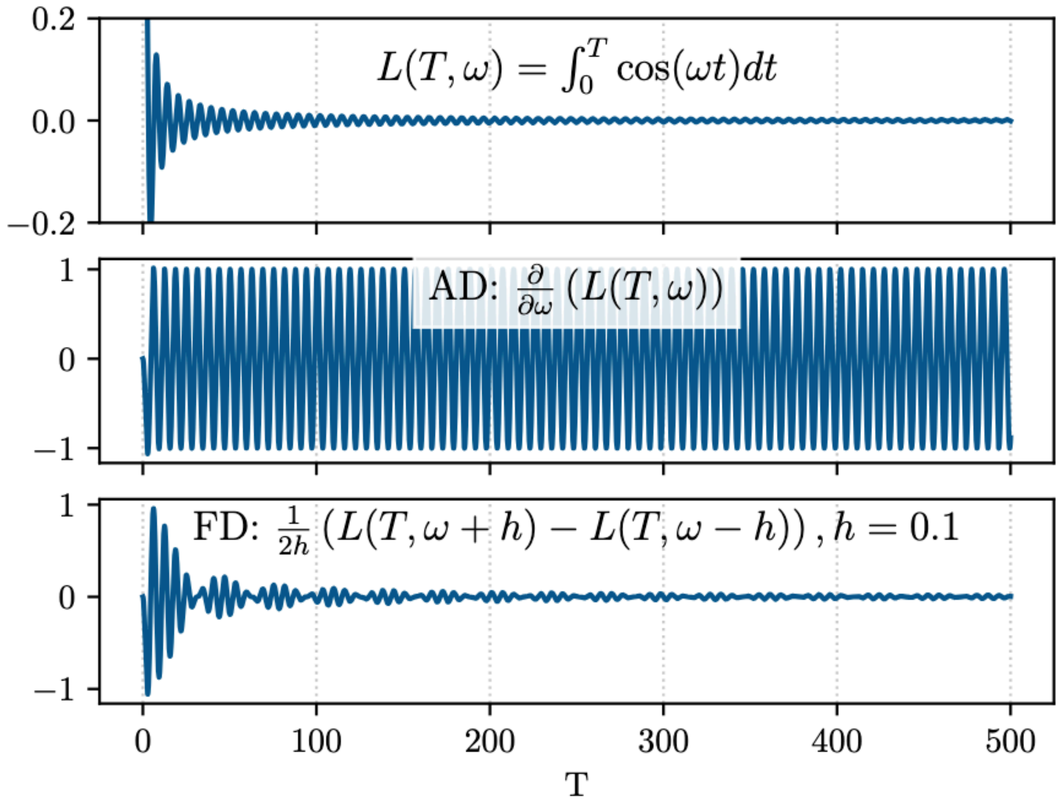
Today

Understanding Automatic Differentiation Pitfalls

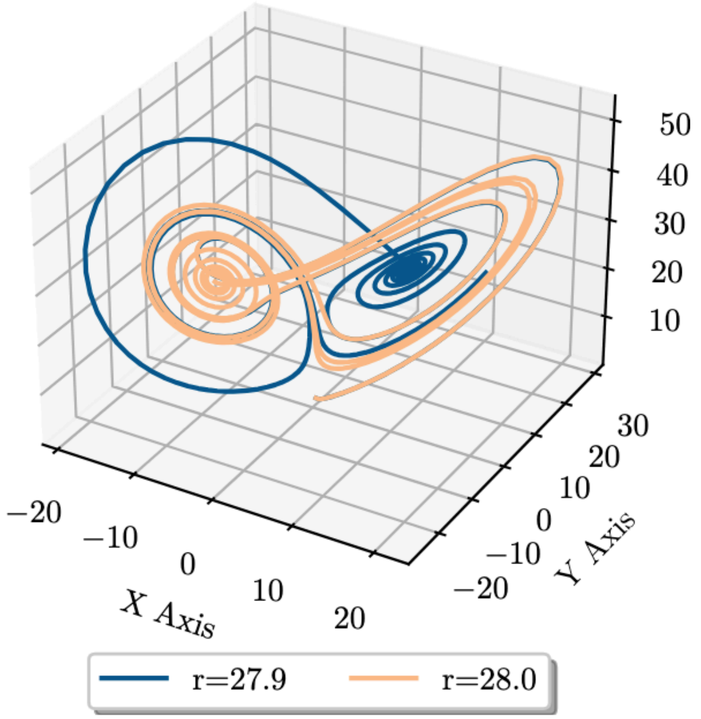
Jan Hückelheim[1] | Harshitha Menon[2] | William Moses[3] | Bruce Christianson[4] | Paul Hovland[1] | Laurent Hascoët[5]

Take home message

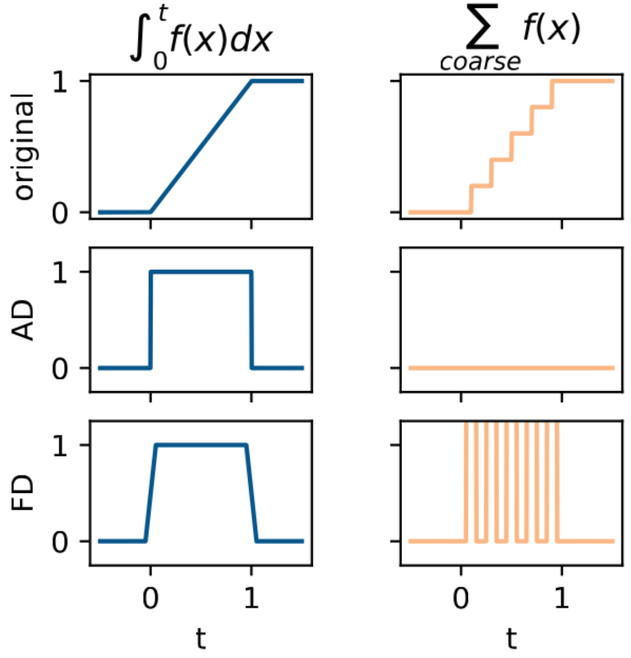
- Sometimes, you can't just autodiff your program and walk away



4 Pitfalls of AD



unexpected derivatives



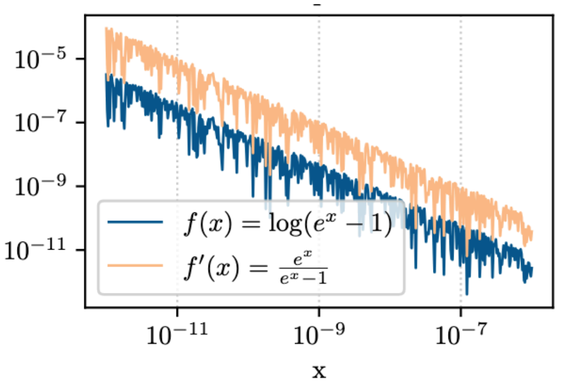
bad approximation

Code with fast path

```
def g(x):
  if x == 0:
    return 0
  else:
    return x
```

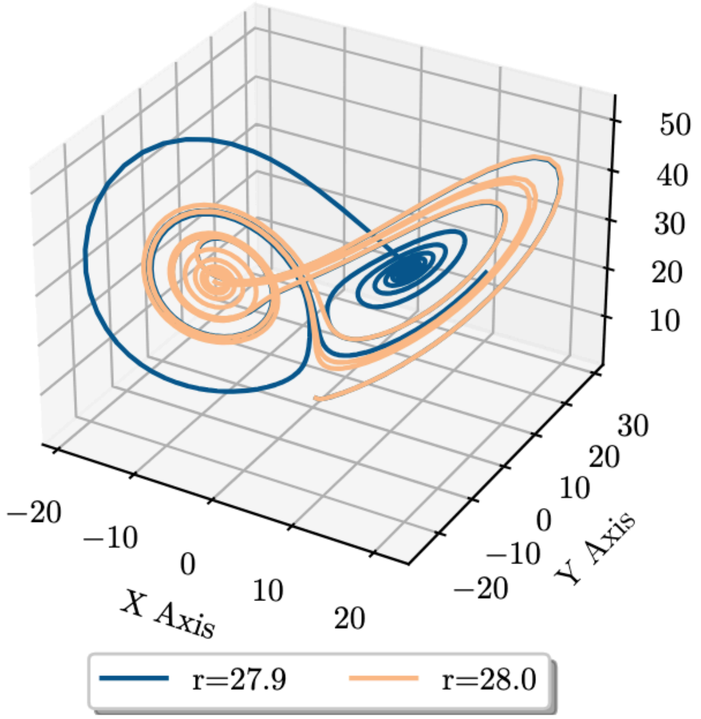
```
def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

branching



numerical accuracy

4 Pitfalls of AD



unexpected derivatives

bad approximation

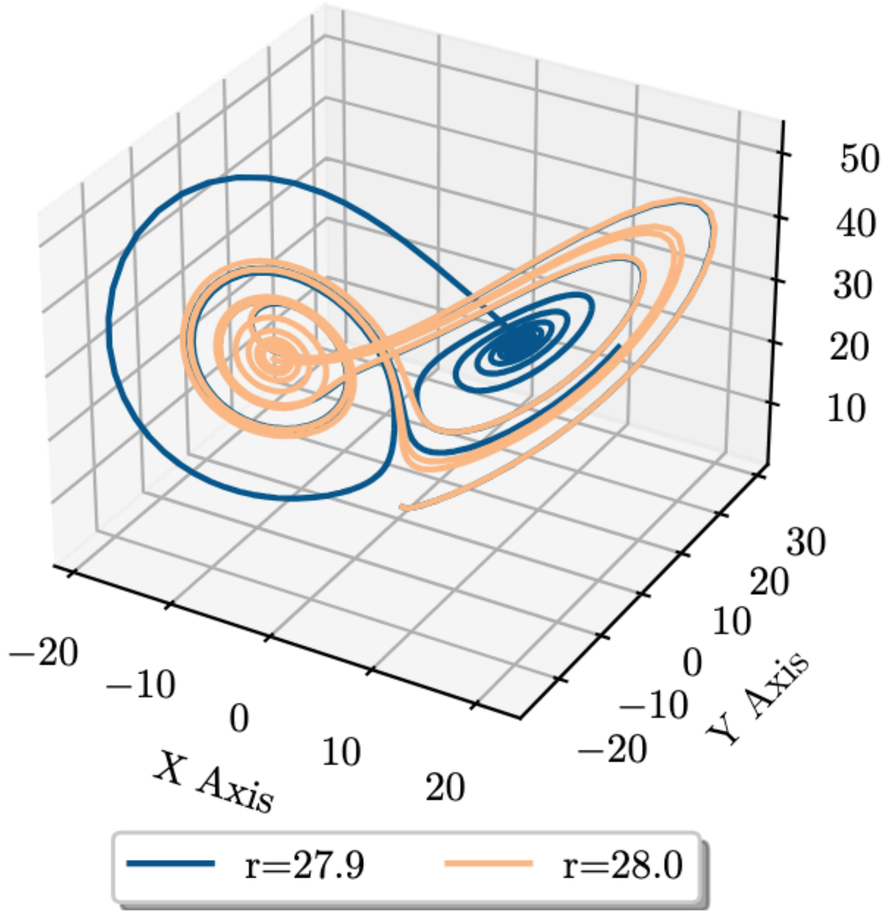
```
Code with fast path

def g(x):
  if x == 0:
    return 0
  else:
    return x

def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

branching

numerical accuracy

Chaos in physics can lead to ill-behaved derivatives

Lorenz attractor

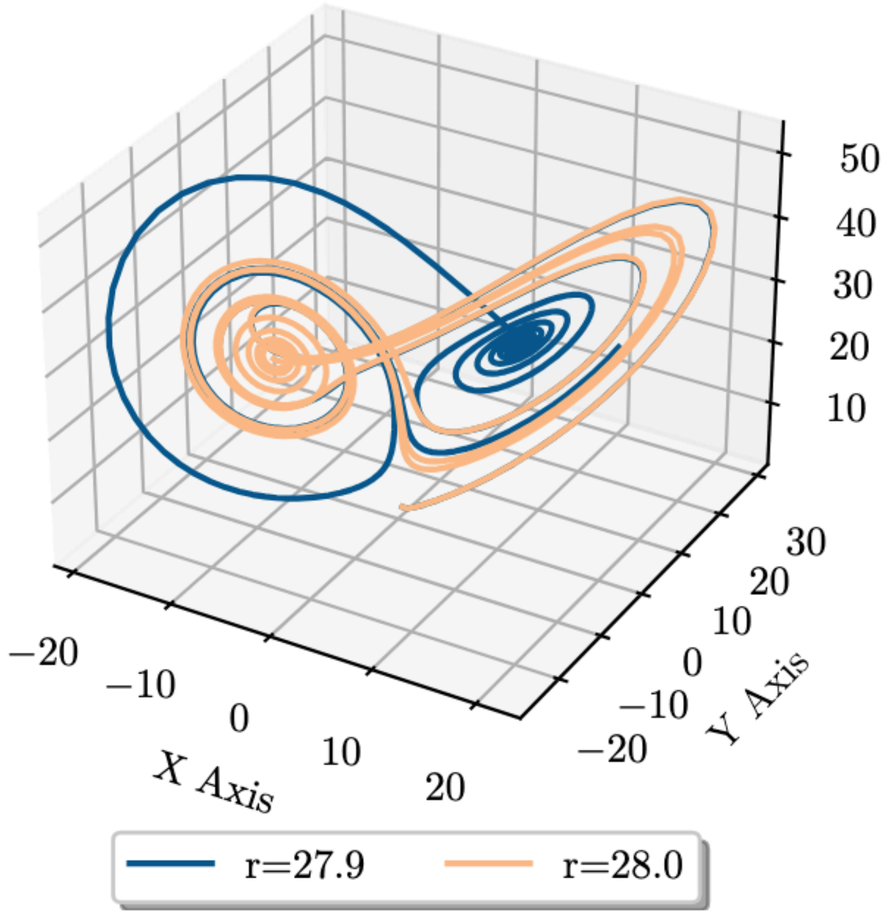


a slight change of r can completely changes the physical trajectory

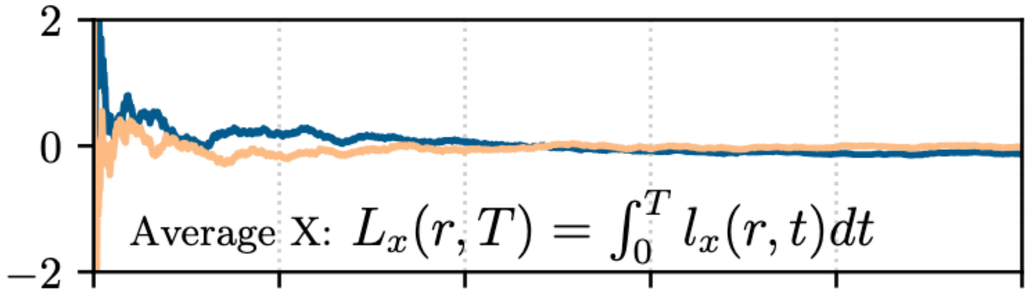
$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(r - z) - y$$
$$\dot{z} = xy - bz$$

Chaos in physics can lead to ill-behaved derivatives

Lorenz attractor

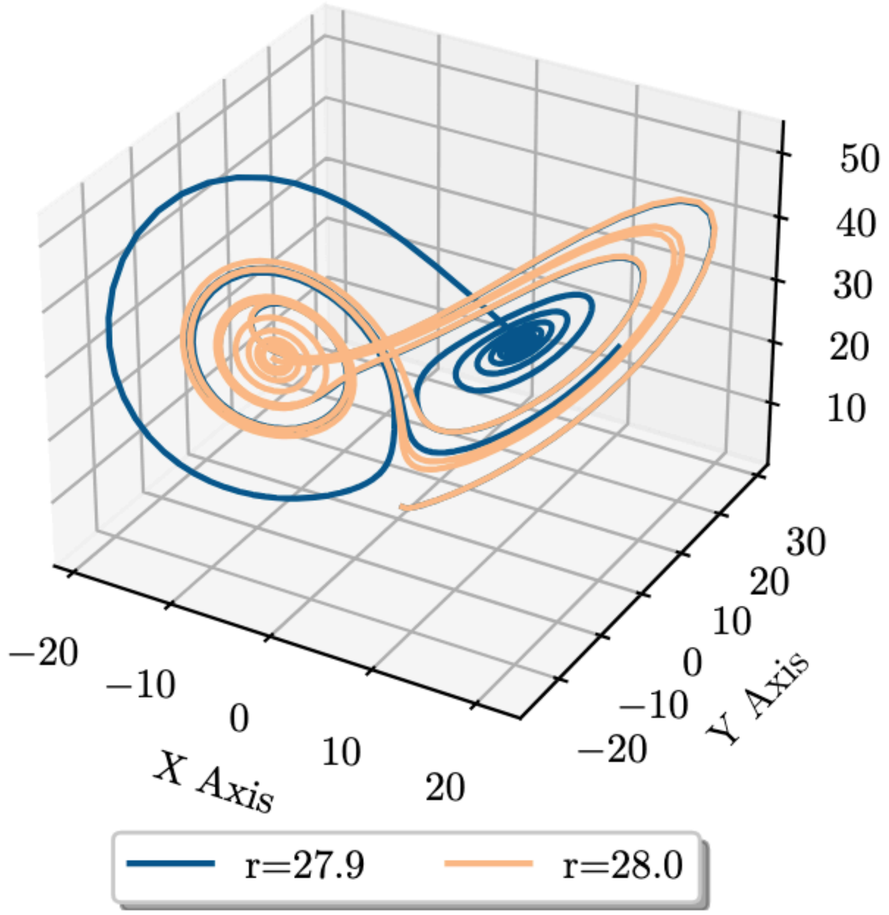


the average of X position over time T is differentiable with r

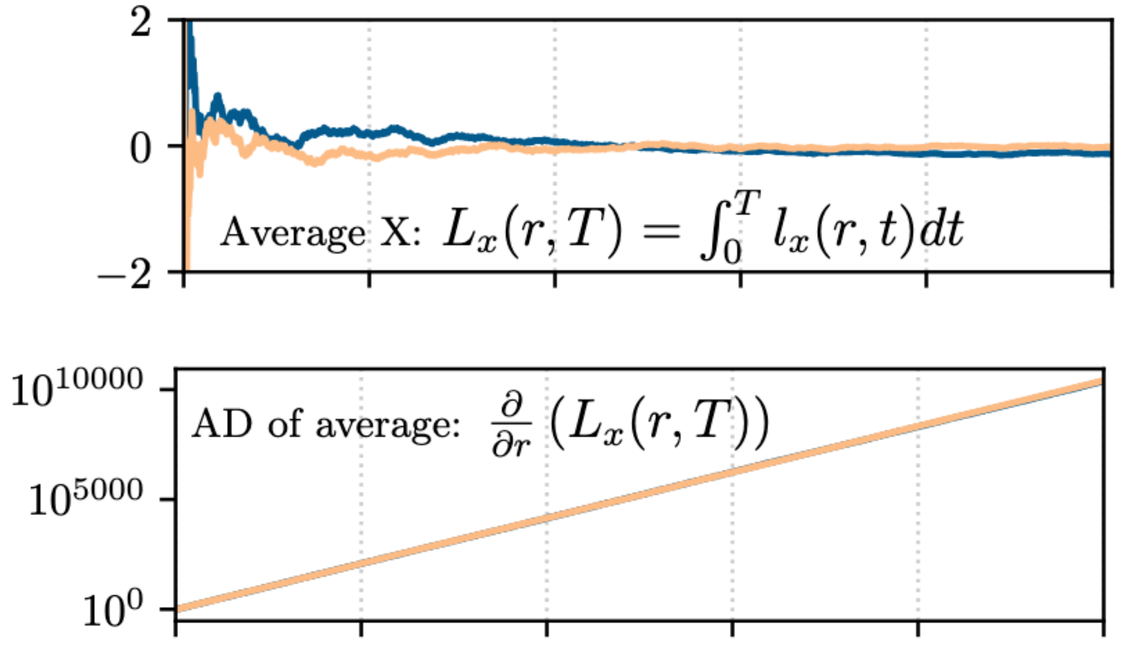


Chaos in physics can lead to ill-behaved derivatives

Lorenz attractor



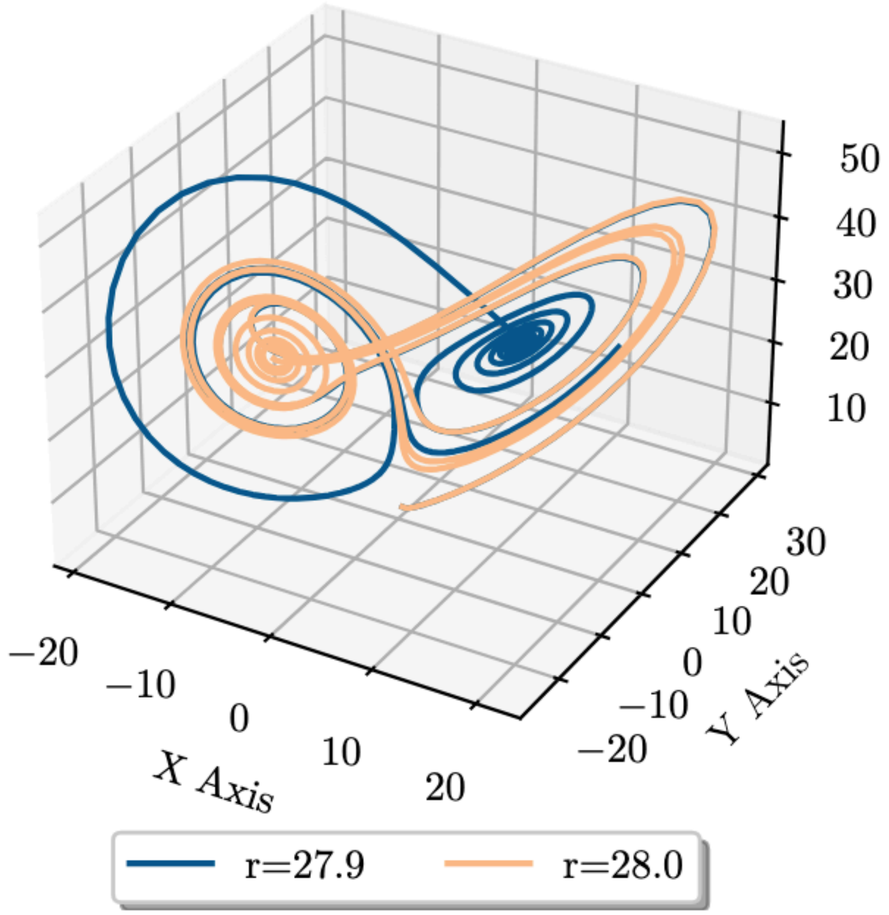
the average of X position over time T is differentiable with r



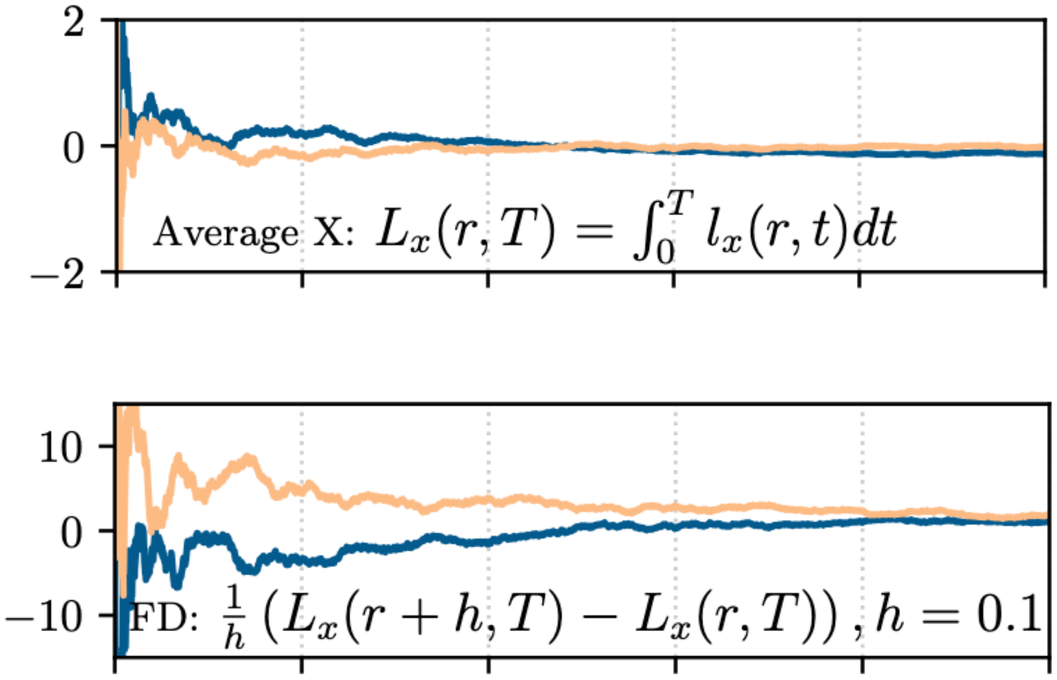
but the derivative w.r.t. r grows exponentially with T!!!

Chaos in physics can lead to ill-behaved derivatives

Lorenz attractor



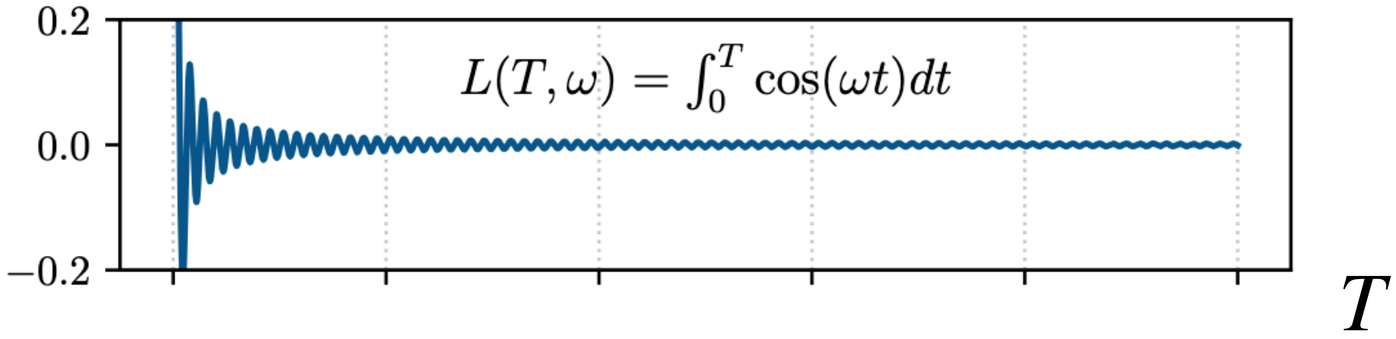
the average of X position over time T is differentiable with r



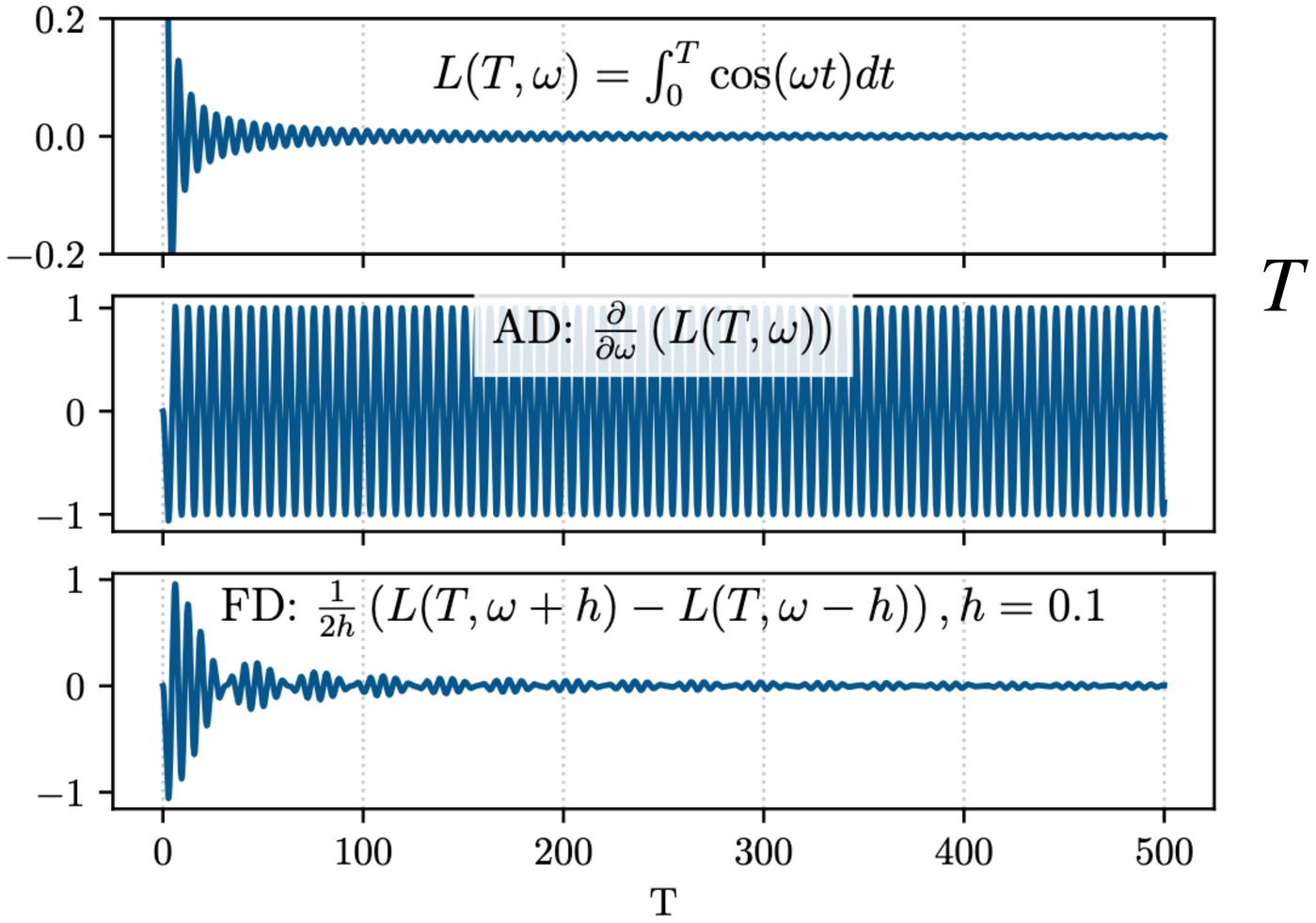
T

finite differences, on the other hand, is a lot more well behaved

Non-chaotic functions can also have similar behaviors



Non-chaotic functions can also have similar behaviors



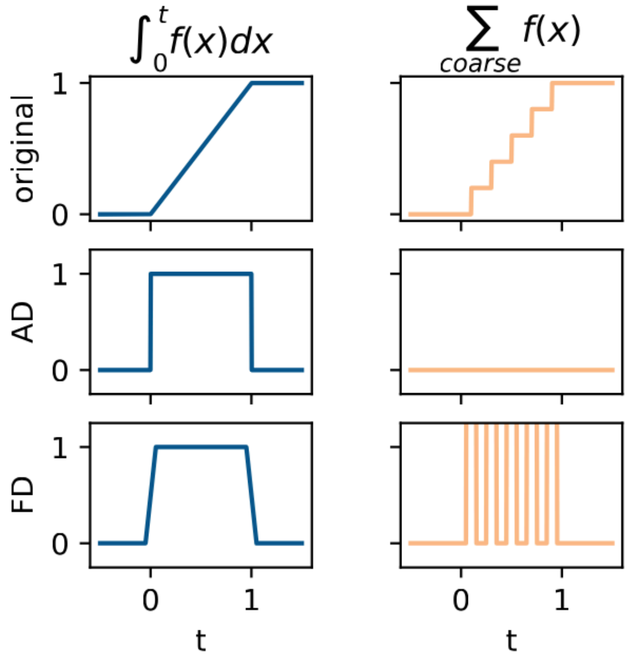
A possible remedy: tame the integrand using a smooth falloff

$$\int_0^T \cos(\omega t)\mathrm{d}t \longrightarrow \int_0^T e^{-\frac{t^2}{T^2\sigma^2}} \cos(\omega t)\mathrm{d}t$$

often called “windowing” or “apodization” in signal processing

4 Pitfalls of AD

unexpected derivatives



bad approximation

Code with fast path

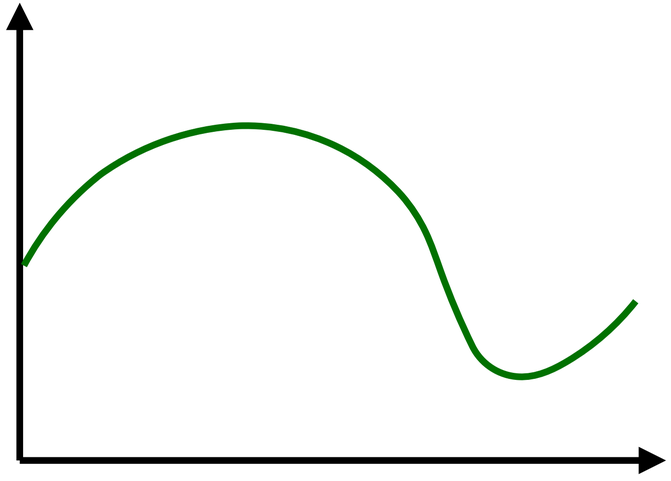
```
def g(x):
  if x == 0:
    return 0
  else:
    return x
```

```
def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

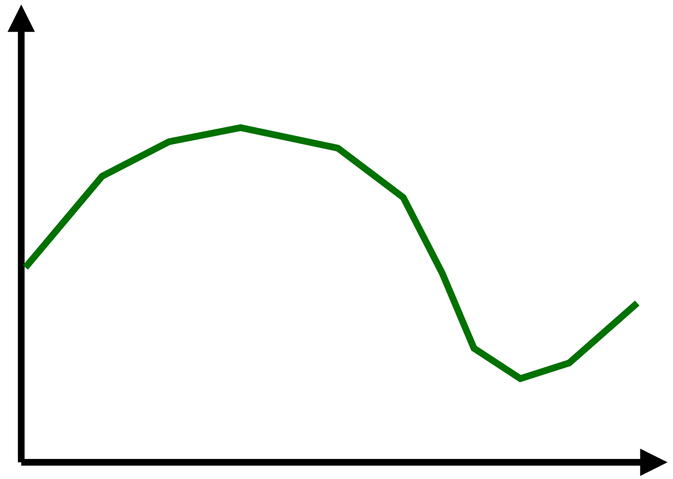
branching

numerical accuracy

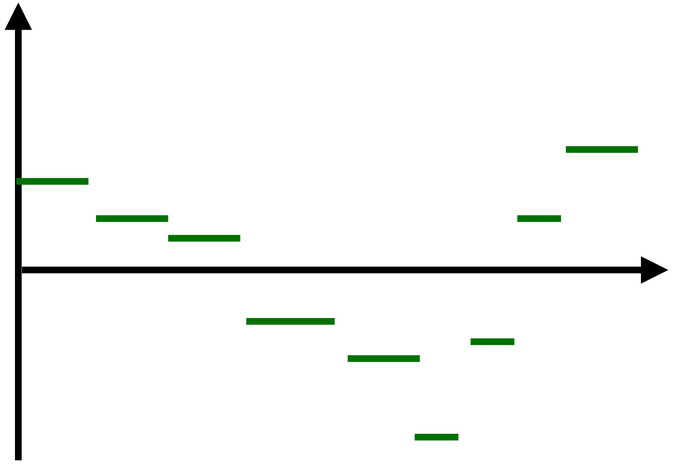
Differentiating an approximation of a function can lead to arbitrarily bad derivatives



a smooth primal function



a piecewise approximation



derivative of a piecewise approximation (bad!)

Differentiating an approximation of a function can lead to arbitrarily bad derivatives

- similar problems happen with integrals & root solving, we have talked about them in previous lectures

```
def g(t):
  sum = 0
  for i in range(N):
    if rand() < t:
      sum += 1
  return sum / N

def dg(t, dt):
  sum = 0
  dsum = 0
  for i in range(N):
    if rand() < t:
      sum += 1
      dsum += 0
  return dsum / N
```

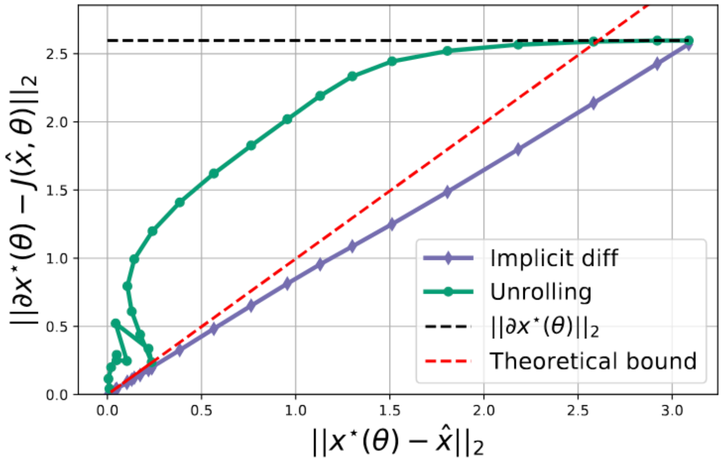
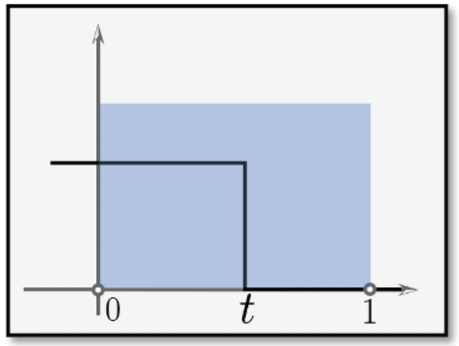


Figure 3: Jacobian estimate errors. Empirical error of implicit differentiation follows closely the theoretical upper bound. Unrolling achieves a much worse error for comparable iterate error.

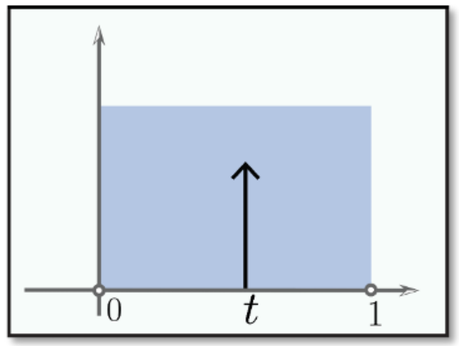
A potential remedy:
differentiate the primal function, then make the approximation

$$\int_0^1 [x < t]\,dx$$



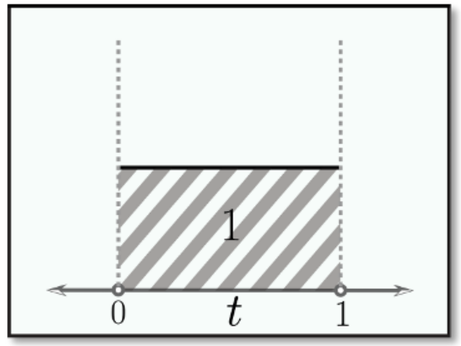
$$\frac{d}{dt}$$

$$\int_0^1 \delta(t - x)\,dx$$



$$\delta$$

$$[0 < t < 1]$$



4 Pitfalls of AD

unexpected derivatives

bad approximation

Code with fast path

```
def g(x):
  if x == 0:
    return 0
  else:
    return x
```

```
def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

branching

numerical accuracy

Branching can lead to undefined behavior

d/dx for x=0 in PyTorch

```
def sillyID_0(x):
  return max(0, x) - max(-x, 0)

def sillyID_1(x):
  return max(x, 0) - max(-x, 0)

def sillyID_2(x):
  return max(0, x) - max(0, -x)
```

example from

Distribution Theoretic Semantics for Non-Smooth Differentiable Programming

PEDRO H. AZEVEDO DE AMORIM, Cornell University, United States
CHRISTOPHER LAM, University of Illinois at Urbana-Champaign, United States

Branching can lead to undefined behavior

```
def sillyID_0(x):
  return max(0, x) - max(-x, 0)

def sillyID_1(x):
  return max(x, 0) - max(-x, 0)

def sillyID_2(x):
  return max(0, x) - max(0, -x)
```

d/dx for x=0 in PyTorch

1

0

2

example from

Distribution Theoretic Semantics for Non-Smooth Differentiable Programming

PEDRO H. AZEVEDO DE AMORIM, Cornell University, United States
CHRISTOPHER LAM, University of Illinois at Urbana-Champaign, United States

The undefined behavior only occurs in a measure zero set of the input space

hence, a practical hack is to simply perturb the input slightly

only behave weirdly at x=0

```
def sillyID_0(x):
  return max(0, x) - max(-x, 0)

def sillyID_1(x):
  return max(x, 0) - max(-x, 0)

def sillyID_2(x):
  return max(0, x) - max(0, -x)
```

see proof here

On Correctness of Automatic Differentiation for Non-Differentiable Functions

Wonyeol Lee
School of Computer Science, KAIST
wonyeol@kaist.ac.kr

Hangyeol Yu
School of Computer Science, KAIST
yhk1344@kaist.ac.kr

Xavier Rival
INRIA Paris, Département d'Informatique of ENS, and CNRS/PSL University
rival@di.ens.fr

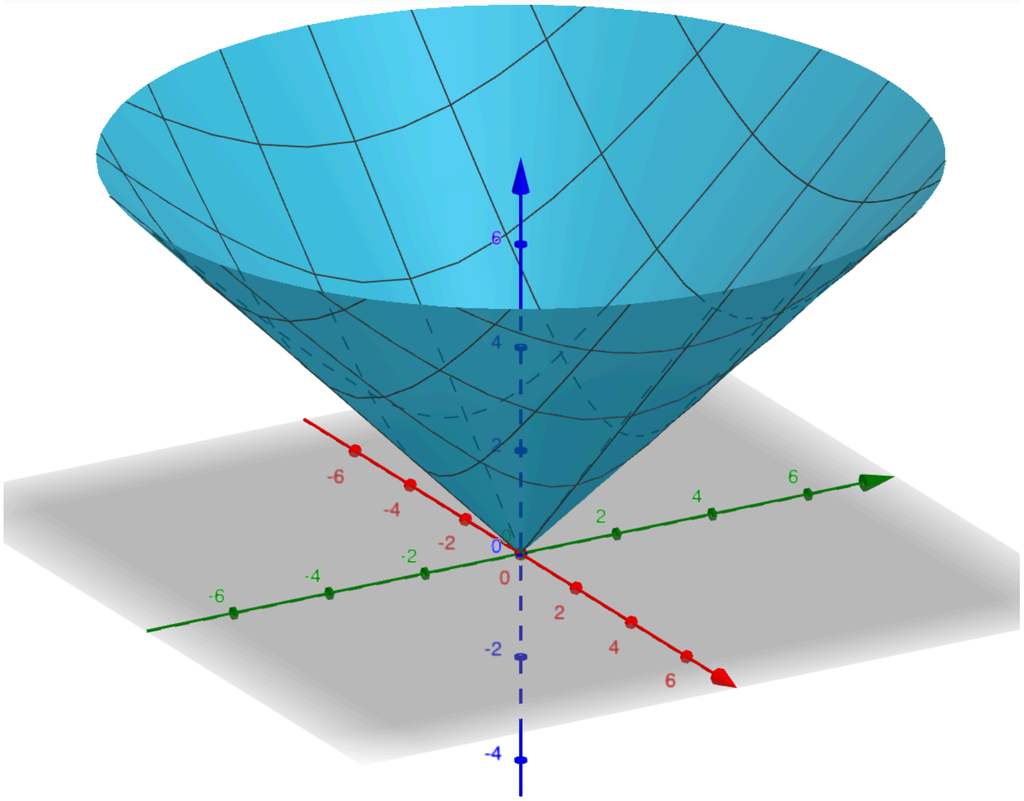
Hongseok Yang
School of Computer Science, KAIST
hongseok.yang@kaist.ac.kr

However, non-differentiability can lead to numerical issues in general

$$f(x, y) = \sqrt{x^2 + y^2}$$

$$Df(0,0)(dx, dy) = ?$$

most autodiff systems will throw NaN at you



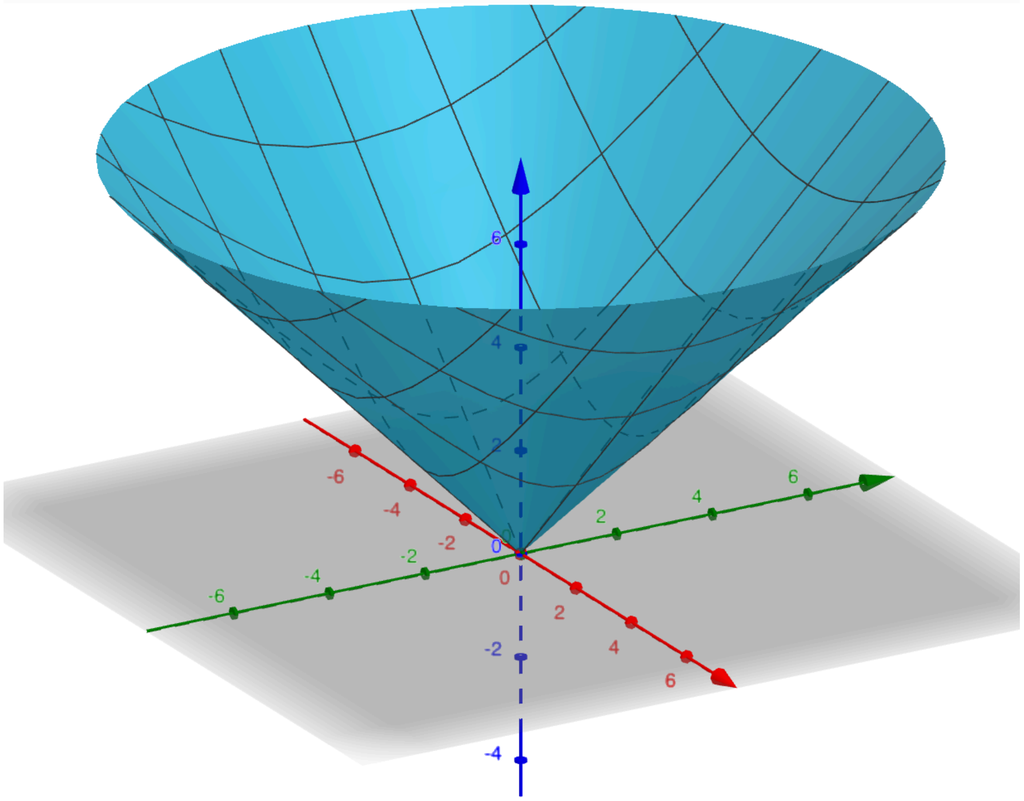
However, non-differentiability can lead to numerical issues in general

$$f(x, y) = \sqrt{x^2 + y^2}$$

$$Df(0,0)(dx, dy) = ?$$

most autodiff systems will throw NaN at you

$$Df(10^{-10},10^{-10})(dx, dy) = ?$$



Potential remedy: no definite answer yet!

- one random idea from me: maybe AD systems should issue warnings when an input is close to a non-differentiable point?

- any other ideas?

```
def sillyID_0(x):
  return max(0, x) - max(-x, 0)

def sillyID_1(x):
  return max(x, 0) - max(-x, 0)

def sillyID_2(x):
  return max(0, x) - max(0, -x)
```

detecting whether different programs are semantically equivalent is undecidable!

4 Pitfalls of AD

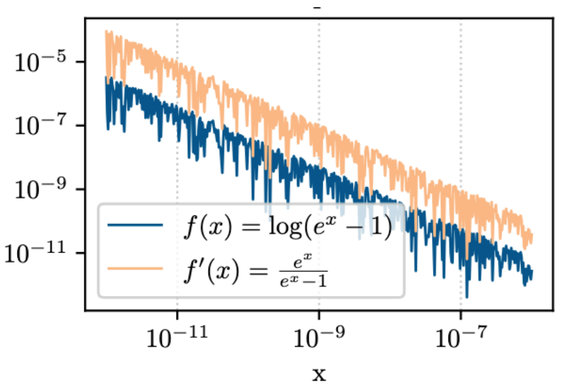
unexpected derivatives

bad approximation

Code with fast path

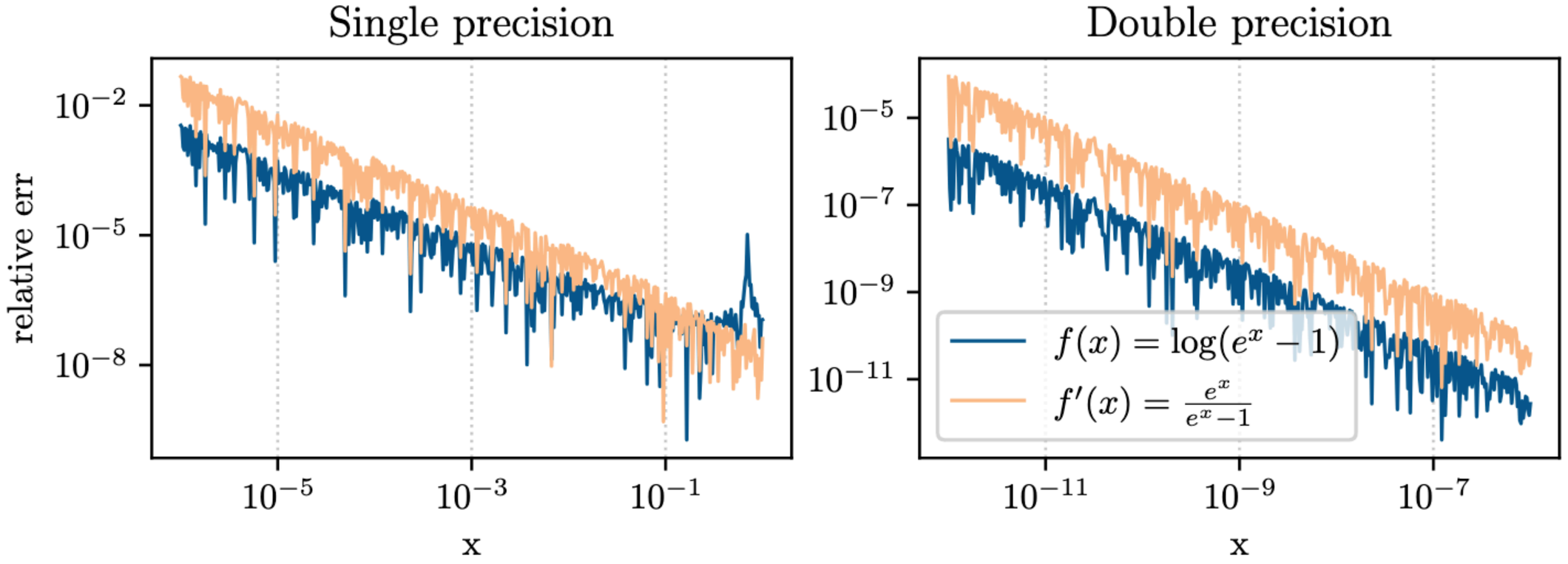
```
def g(x):
  if x == 0:
    return 0
  else:
    return x

def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

branching



numerical accuracy

The derivative operation can be less numerically accurate than primal ones



A potential remedy: Herbie

doesn't work for large scale programs but is excellent for small operations

Initial Program: 5.9% accurate, 1.0× speedup

Math

$$\frac{e^x}{e^x - 1}$$

Alternative 1: 99.1% accurate, 41.0× speedup

Math

$$\frac{1}{x} + 0.5$$

▶ Derivation

Alternative 2: 98.0% accurate, 68.3× speedup

Math

$$\frac{1}{x}$$

▶ Derivation

Alternative 3: 3.2% accurate, 205.0× speedup

Math

$$0.5$$

▶ Derivation

Automatically Improving Accuracy for Floating Point Expressions

Pavel Panchekha Alex Sachez-Stern James R. Wilcox Zachary Tatlock
University of Washington, USA
{pavpan,asnchstr,jrw12,ztatlock}@cs.washington.edu

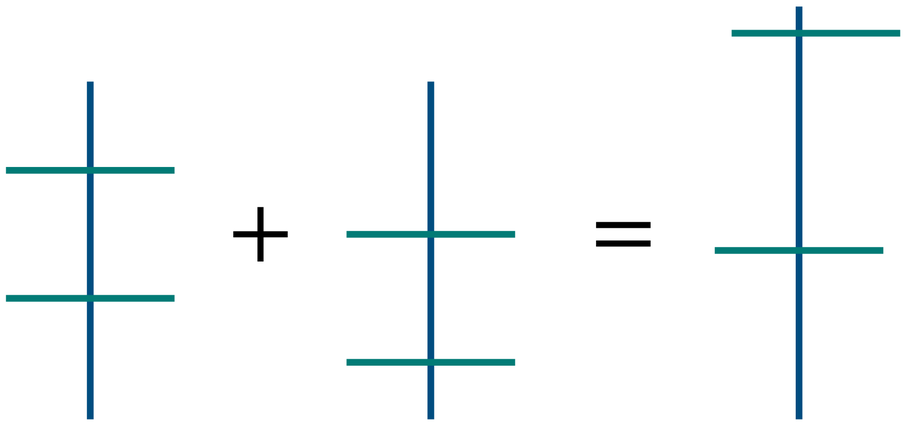
https://herbie.uwplse.org/

Another potential remedy: error bounds tracking through uncertainty propagation

$$a \oplus b = \text{round}(a + b) \in (a + b)(1 \pm \epsilon_{\text{m}}) = [(a + b)(1 - \epsilon_{\text{m}}), (a + b)(1 + \epsilon_{\text{m}})].$$

keep track of the lower bound and upper bound for each operation to track the floating point cancellation error

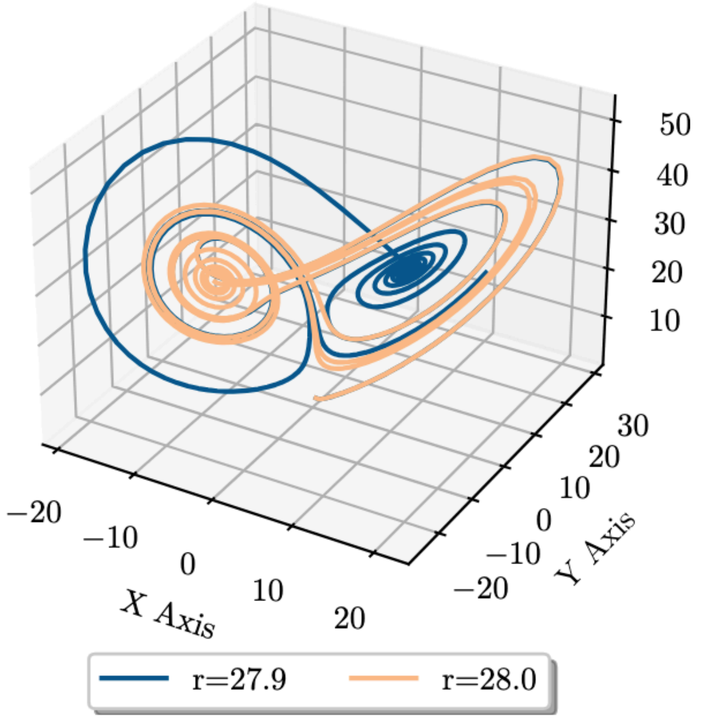
$$(((a \oplus b) \oplus c) \oplus d) \in ((((a + b)(1 \pm \epsilon_{\text{m}})) + c)(1 \pm \epsilon_{\text{m}}) + d)(1 \pm \epsilon_{\text{m}}) = (a + b)(1 \pm \epsilon_{\text{m}})^3 + c(1 \pm \epsilon_{\text{m}})^2 + d(1 \pm \epsilon_{\text{m}}).$$



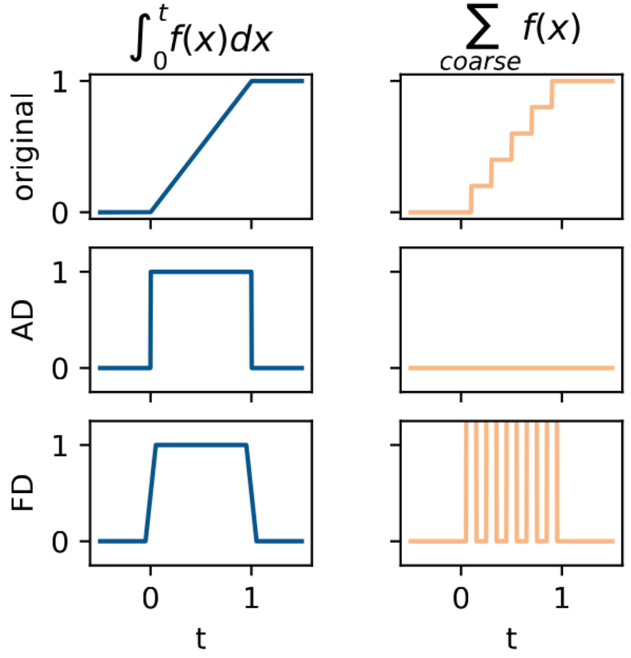
https://www.pbr-book.org/4ed/Shapes/Managing_Rounding_Error

https://dspace.mit.edu/bitstream/handle/1721.1/127465/1192966915-MIT.pdf?sequence=1&isAllowed=y

4 Pitfalls of AD



unexpected derivatives



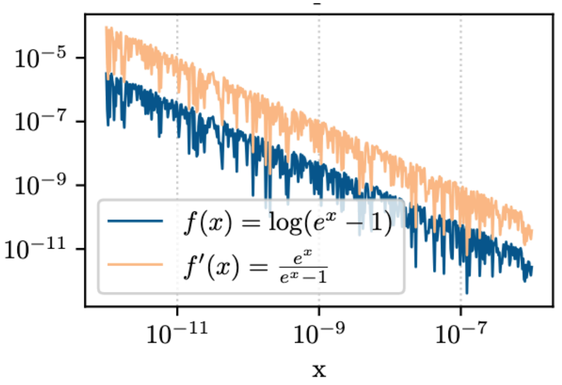
bad approximation

Code with fast path

```
def g(x):
  if x == 0:
    return 0
  else:
    return x
```

```
def g_d(x, xd=1.0):
  if x == 0:
    return 0 # wrong!
  else:
    return xd
```

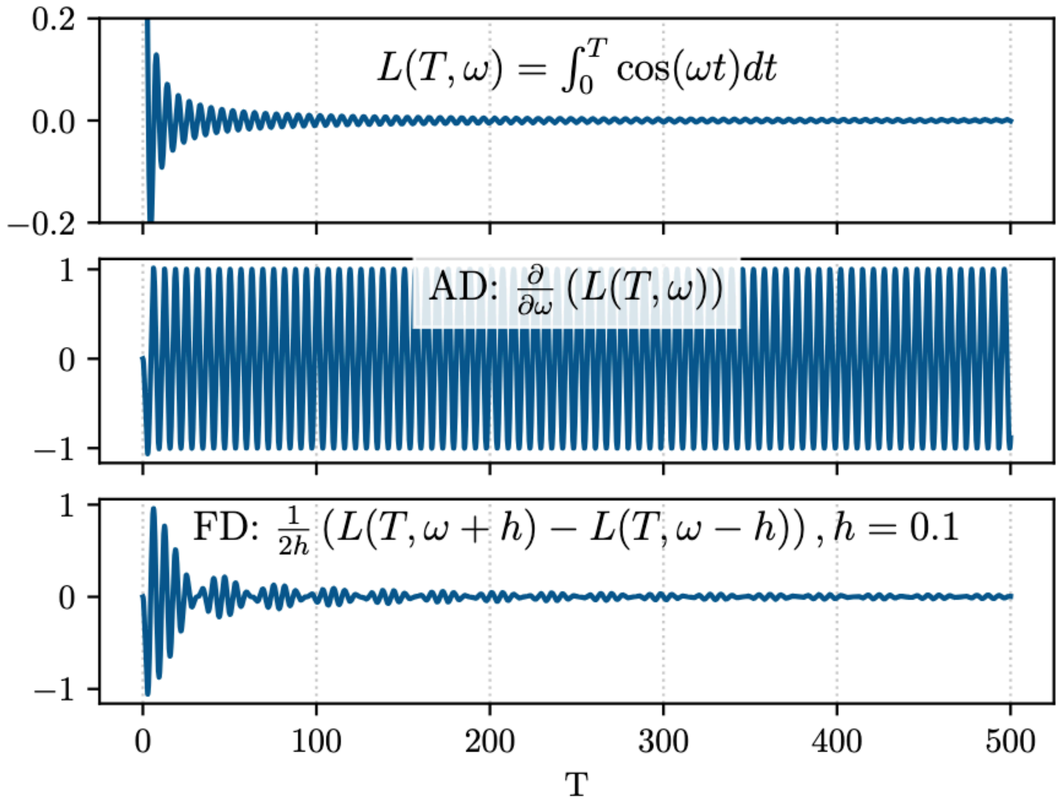
branching



numerical accuracy

Debugging strategies

- finite differences
  - though sometimes, finite differences are not to be trusted since they are also susceptible to numerical errors
- compare between forward and reverse modes
- in general, having language support for debugging AD issues is an open research problem

Take home message

- Sometimes, you can't just autodiff your program and walk away