Differentiation: the good, the bad, and the ugly

Lindon Roberts, University of Sydney (lindon.roberts@sydney.edu.au)

Sydney University Mathematical Society
1 May 2024

Differentiation

If $f : \mathbb{R} \to \mathbb{R}$, then its derivative is

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

provided the limit exists.

Differentiation

If $f : \mathbb{R} \to \mathbb{R}$, then its derivative is

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

provided the limit exists.

From high school calculus, you basically know enough to differentiate almost any function you might see in practice:

- Building blocks: polynomials, trig functions, exponents
- Combining blocks: chain rule, product rule, Fundamental Theorem of Calculus, etc.

(by contrast, there isn't a 'rote procedure' for integration)

Usage in Optimisation

My research is in mathematical optimisation: automatic procedures for finding maxima/minima of functions.

Usage in Optimisation

My research is in mathematical optimisation: automatic procedures for finding maxima/minima of functions. For example, maximise profit of a trading strategy, minimise drag for a racecar, minimise prediction errors of a machine learning model.

Usage in Optimisation

My research is in mathematical optimisation: automatic procedures for finding maxima/minima of functions. For example, maximise profit of a trading strategy, minimise drag for a racecar, minimise prediction errors of a machine learning model.

Theorem

If $f : \mathbb{R} \to \mathbb{R}$ is differentiable and has a maximum/minimum at x^*, then $f'(x^*) = 0$. (but not the converse!)

Usage in Optimisation

My research is in mathematical optimisation: automatic procedures for finding maxima/minima of functions. For example, maximise profit of a trading strategy, minimise drag for a racecar, minimise prediction errors of a machine learning model.

Theorem

If $f : \mathbb{R} \to \mathbb{R}$ is differentiable and has a maximum/minimum at x^*, then $f'(x^*) = 0$. (but not the converse!)

Derivatives are more useful than just this! Given a guess x for a minimum (for example), if $f'(x) > 0$ then we should try smaller values of x, but if $f'(x) < 0$ then we should try larger values of x.

This generalises to functions $f : \mathbb{R}^n \to \mathbb{R}$. Called gradient descent or steepest descent.

Outline

1. **The good**
2. The bad
3. The ugly
4. The good (again)

Black Boxes

If you have the mathematical form for $f(x)$, then the 'rote procedure' can give you $f'(x)$. But sometimes you don't have a simple equation!

- Do a real-world experiment or survey
- Solve a complicated differential equation (approximately!)
- My code for $f(x)$ calls some other package and I don't know what it does
- Mathematica gave me an answer but it was so long I keep making mistakes coding it up

Black Boxes

If you have the mathematical form for $f(x)$, then the 'rote procedure' can give you $f'(x)$. But sometimes you don't have a simple equation!

- Do a real-world experiment or survey
- Solve a complicated differential equation (approximately!)
- My code for $f(x)$ calls some other package and I don't know what it does
- Mathematica gave me an answer but it was so long I keep making mistakes coding it up

Question

If you can only compute $f(x)$, can you estimate $f'(x)$?

Finite Differencing

Question

If you can only compute $f(x)$, can you estimate $f'(x)$?

Yes! The most common approach is called finite differencing.

Finite Differencing

Question

If you can only compute $f(x)$, can you estimate $f'(x)$?

Yes! The most common approach is called finite differencing.

The simplest finite difference estimate is forward differencing:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

for some value $h \approx 0$.

Finite Differencing

The forward differencing approximation is

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

What is the error introduced by this approximation?

Finite Differencing

The forward differencing approximation is

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

What is the error introduced by this approximation?

Use a Taylor series with remainder:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi),$$

for some $\xi \in [x, x+h]$.

Finite Differencing

The forward differencing approximation is

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

What is the error introduced by this approximation?

Use a Taylor series with remainder:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi),$$

for some $\xi \in [x, x+h]$.

Rearrange to get

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}hf''(\xi),$$

so the error is (up to constants) of size h, usually written $\mathcal{O}(h)$.

Forward differencing has error of size $\mathcal{O}(h)$, but there are other approximations with better accuracy:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \mathcal{O}(h^2)$$

is central differencing, and if h is small (the whole point!) then $h^2 \ll h$.

Forward differencing has error of size $\mathcal{O}(h)$, but there are other approximations with better accuracy:

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + \mathcal{O}(h^2)$$

is central differencing, and if h is small (the whole point!) then $h^2 \ll h$.

So, whenever you need $f'(x)$ but can't compute it directly, just evaluate one of these approximations (and take h sufficiently small that you get the accuracy you want). Easy!

Outline

1. The good
2. **The bad**
3. The ugly
4. The good (again)

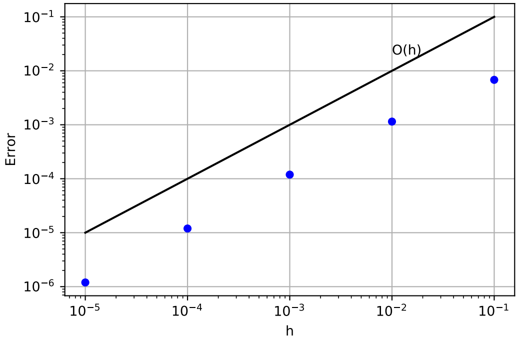
Finite Differencing In Practice

Let's write some Python code to test forward differencing. We will try to estimate $f'(1)$ for $f(x) = x\sin(x)$, with true answer $f'(1) = \sin(1) + \cos(1) \approx 1.382$:

```
import numpy as np
import matplotlib.pyplot as plt

f = lambda x: x * np.sin(x)          # function to differentiate, f(x)
x = 1.0                              # where to evaluate f'(x)
true_value = np.sin(1) + np.cos(1)  # true value f'(x)

for h in [0.1, 0.01, 0.001, 0.0001, 0.00001]:  # try different values of h
  deriv_est = (f(x+h) - f(x))/h                 # forward differencing approximation
  error = abs(deriv_est - true_value)           # error in forward differencing
  plt.loglog(h, error, 'b.')                    # plot error vs h

plt.show()  # show plot
```

Finite Differencing In Practice

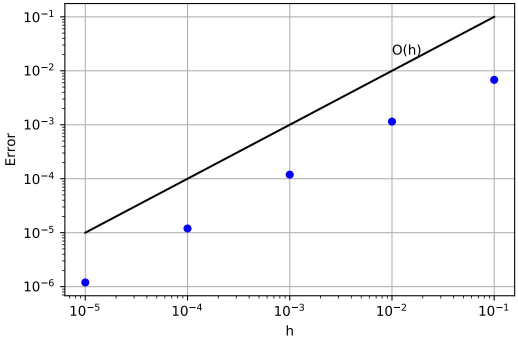
This produces the plot



So the error looks to be decreasing at a rate of $\mathcal{O}(h)$ as expected.

Finite Differencing In Practice

This produces the plot

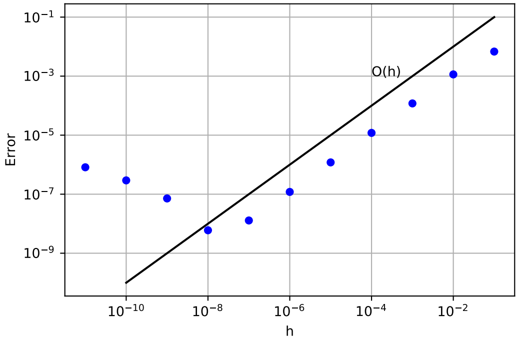


So the error looks to be decreasing at a rate of $\mathcal{O}(h)$ as expected.

Let's keep decreasing the value of h, to get better accuracy...

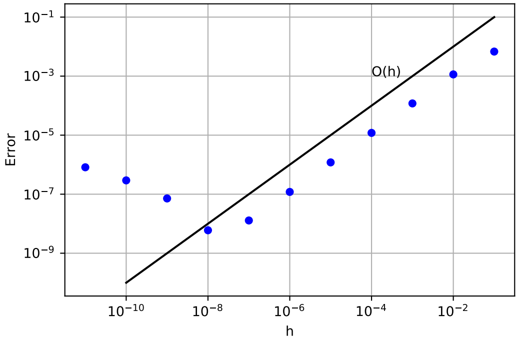
Finite Differencing In Practice

Taking $h = 10^{-1}, 10^{-2}, \ldots, 10^{-11}$ in the same code gives the plot:



Finite Differencing In Practice

Taking $h = 10^{-1}, 10^{-2}, \ldots, 10^{-11}$ in the same code gives the plot:



Something goes very wrong for small *h*!

Finite Differencing With Errors

Problem

We have a theorem that says the error decreases to zero like $\mathcal{O}(h)$, but that doesn't happen in practice. Is mathematics wrong?

Finite Differencing With Errors

Problem

We have a theorem that says the error decreases to zero like $\mathcal{O}(h)$, but that doesn't happen in practice. Is mathematics wrong?

Of course not! Something must mean we aren't satisfying the assumptions — what?

Finite Differencing With Errors

Problem

We have a theorem that says the error decreases to zero like $\mathcal{O}(h)$, but that doesn't happen in practice. Is mathematics wrong?

Of course not! Something must mean we aren't satisfying the assumptions — what?

Our theoretical analysis doesn't account for rounding errors.

Finite Differencing With Errors

Problem

We have a theorem that says the error decreases to zero like $\mathcal{O}(h)$, but that doesn't happen in practice. Is mathematics wrong?

Of course not! Something must mean we aren't satisfying the assumptions — what?

Our theoretical analysis doesn't account for rounding errors.

- Computers can't store all numbers in $\mathbb{R}$, they round them in scientific notation to approx. 15–16 significant figures (53 significant figures in binary)
- This is called 64-bit floating-point representation
- This leads to some weird arithmetic properties, e.g. loss of associativity $(a+b)+c \neq a+(b+c)$
- Simple example: try calculating $0.1+0.2-0.3$ in Python

Finite Differencing With Errors

We can model rounding errors as:

$$\text{Computed value} = \text{True value} \cdot (1 + \delta),$$

for some value $|\delta| \le \epsilon$, where $\epsilon = 2^{-52} \approx 2.22 \times 10^{-16}$ for 64-bit floating-point numbers.

Finite Differencing With Errors

We can model rounding errors as:

$$\text{Computed value} = \text{True value} \cdot (1 + \delta),$$

for some value $|\delta| \leq \epsilon$, where $\epsilon = 2^{-52} \approx 2.22 \times 10^{-16}$ for 64-bit floating-point numbers.

Incorporating this into finite differencing, our error now looks like:

$$\left| \frac{f(x+h)(1+\delta_1) - f(x)(1+\delta_2)}{h} - f'(x) \right|$$

Finite Differencing With Errors

We can model rounding errors as:

$$\text{Computed value} = \text{True value} \cdot (1+\delta),$$

for some value $|\delta| \leq \epsilon$, where $\epsilon = 2^{-52} \approx 2.22 \times 10^{-16}$ for 64-bit floating-point numbers.

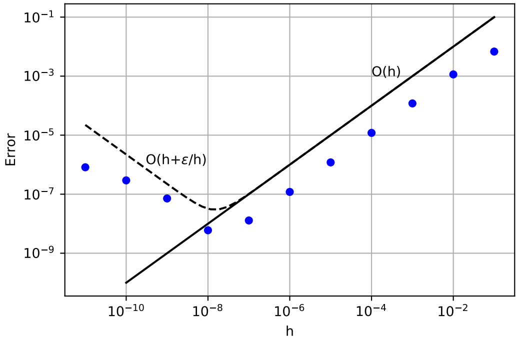
Incorporating this into finite differencing, our error now looks like:

$$\left| \frac{f(x+h)(1+\delta_1) - f(x)(1+\delta_2)}{h} - f'(x) \right|$$

From the triangle inequality, we get

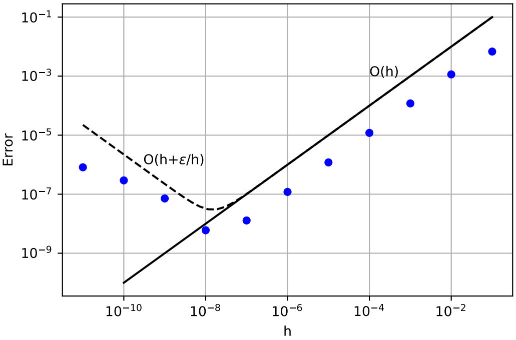
$$\text{error} \leq \underbrace{\left| \frac{f(x+h)-f(x)}{h} - f'(x) \right|}_{=\mathcal{O}(h)} + \frac{|f(x+h)| \cdot \overbrace{|\delta_1|}^{\leq \epsilon}}{h} + \frac{|f(x)| \cdot \overbrace{|\delta_2|}^{\leq \epsilon}}{h} = \mathcal{O}\left(h + \frac{\epsilon}{h}\right)$$

Same plot as before, but compare $\mathcal{O}(h)$ and $\mathcal{O}(h + \epsilon/h)$ error estimates:



Finite Differencing In Practice

Same plot as before, but compare $\mathcal{O}(h)$ and $\mathcal{O}(h+\epsilon/h)$ error estimates:



This explains what was going wrong! Note minimum error is $\mathcal{O}(\sqrt{\epsilon})$ at $h \approx \sqrt{\epsilon}$.

Outline

1. The good
2. The bad
3. **The ugly**
4. The good (again)

Bus Arrival Problem

Problem

People arrive at a bus stop randomly between times $t = 0$ and $t = 1$ (uniform distribution). A bus will leave at $t = 1$, and we want to schedule a second bus to leave at another time $t \in (0, 1)$. What other bus time minimises the average waiting time?

Bus Arrival Problem

Problem

People arrive at a bus stop randomly between times $t = 0$ and $t = 1$ (uniform distribution). A bus will leave at $t = 1$, and we want to schedule a second bus to leave at another time $t \in (0, 1)$. What other bus time minimises the average waiting time?

If the second bus leaves at time t, then a passenger arriving at time x has

$$\text{Waiting time}(x) = \begin{cases} t - x, & \text{if } x \leq t, \\ 1 - x, & \text{otherwise.} \end{cases}$$

Bus Arrival Problem

Problem

People arrive at a bus stop randomly between times $t = 0$ and $t = 1$ (uniform distribution). A bus will leave at $t = 1$, and we want to schedule a second bus to leave at another time $t \in (0, 1)$. What other bus time minimises the average waiting time?

If the second bus leaves at time t, then a passenger arriving at time x has

$$\text{Waiting time}(x) = \begin{cases} t - x, & \text{if } x \le t, \\ 1 - x, & \text{otherwise.} \end{cases}$$

So the average wait time for passengers $x \in \text{Uniform}([0, 1])$ is

$$\int_0^1 \text{Waiting time}(x)\, dx = \int_0^t t - x\, dx + \int_t^1 1 - x\, dx = t^2 - t + \frac{1}{2}$$

Bus Arrival Problem

Problem

People arrive at a bus stop randomly between times $t = 0$ and $t = 1$ (uniform distribution). A bus will leave at $t = 1$, and we want to schedule a second bus to leave at another time $t \in (0, 1)$. What other bus time minimises the average waiting time?

If the second bus leaves at time t, then a passenger arriving at time x has

$$\text{Waiting time}(x) = \begin{cases} t - x, & \text{if } x \le t, \\ 1 - x, & \text{otherwise.} \end{cases}$$

So the average wait time for passengers $x \in \text{Uniform}([0, 1])$ is

$$\int_0^1 \text{Waiting time}(x)\, dx = \int_0^t t - x\, dx + \int_t^1 1 - x\, dx = t^2 - t + \frac{1}{2}$$

Minimising over $t \in (0, 1)$ gives optimal choice $t^* = \frac{1}{2}$.

Bus Arrival Problem

It is very easy to construct much harder versions of this problem, such as locations of ambulance bases or bike sharing racks. With this in mind, what if we wanted to solve the bus problem computationally?

Bus Arrival Problem

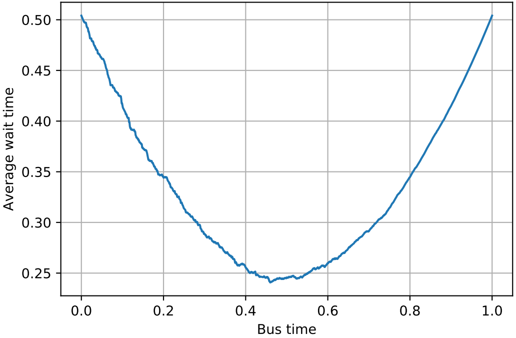
It is very easy to construct much harder versions of this problem, such as locations of ambulance bases or bike sharing racks. With this in mind, what if we wanted to solve the bus problem computationally?

Basic idea is simulation optimisation:

- Generate a large number of passenger arrival times using a random number generator
- For a given t, calculate the average waiting time for these passengers
- Minimise this quantity with respect to t

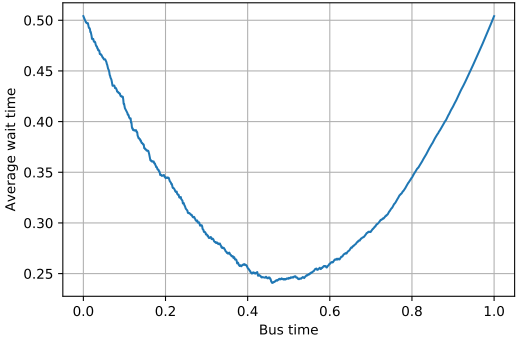
Bus Arrival Problem

Simulating 1000 arrival times and plotting the average waiting time w.r.t. t gives:



Bus Arrival Problem

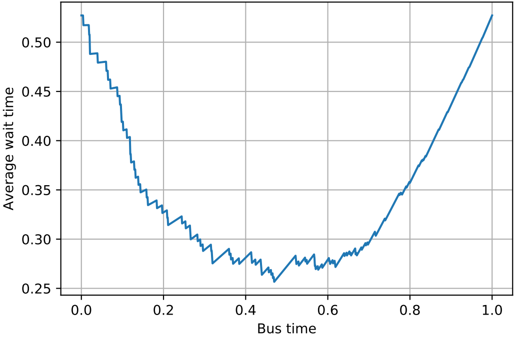
Simulating 1000 arrival times and plotting the average waiting time w.r.t. t gives:



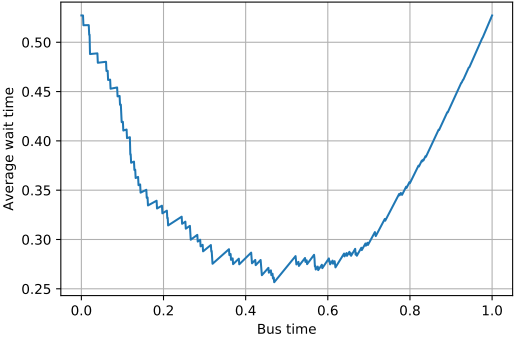
Overall, this looks good: it approximates $t^2 - t + \frac{1}{2}$ and has a minimum near $t^* = 1/2$

Bus Arrival Problem

Now simulate only 100 arrival times:



Now simulate only 100 arrival times:



Is this really a good approximation? Does sign$(f'(t))$ tell you whether f is increasing/decreasing?

Bus Arrival Problem

Let's think about the function $f(t)$ for a fixed set of arrival times $x_1 < x_2 < \cdots < x_N$:

Bus Arrival Problem

Let's think about the function $f(t)$ for a fixed set of arrival times $x_1 < x_2 < \cdots < x_N$:

- If $x_j < t < x_{j+1}$ then j passengers get first bus, $N - j$ get second bus, average wait time is

$$f(t) = \frac{1}{N}\left[\sum_{i=1}^{j} t - x_i + \sum_{i=j+1}^{N} 1 - x_i\right]$$

Bus Arrival Problem

Let's think about the function $f(t)$ for a fixed set of arrival times $x_1 < x_2 < \cdots < x_N$:

- If $x_j < t < x_{j+1}$ then j passengers get first bus, $N-j$ get second bus, average wait time is

$$f(t) = \frac{1}{N}\left[\sum_{i=1}^{j} t - x_i + \sum_{i=j+1}^{N} 1 - x_i\right]$$

so f is linear in t with

$$f'(t) = \frac{1}{N}\left[\sum_{i=1}^{j} 1 + \sum_{i=j+1}^{N} 0\right] = \frac{j}{N}$$

Bus Arrival Problem

Let's think about the function $f(t)$ for a fixed set of arrival times $x_1 < x_2 < \cdots < x_N$:

- If $x_j < t < x_{j+1}$ then j passengers get first bus, $N - j$ get second bus, average wait time is

$$f(t) = \frac{1}{N}\left[\sum_{i=1}^{j} t - x_i + \sum_{i=j+1}^{N} 1 - x_i\right]$$

so f is linear in t with

$$f'(t) = \frac{1}{N}\left[\sum_{i=1}^{j} 1 + \sum_{i=j+1}^{N} 0\right] = \frac{j}{N}$$

- If $t = x_j$, then passenger j moves from second bus to first bus (waits $1 - t$ less time), so $f(t)$ decreases by $\frac{1-t}{N}$

So the simulated average wait time f is piecewise linear with N discontinuities of size $\mathcal{O}(1/N)$.

Bus Arrival Problem

So the simulated average wait time f is piecewise linear with N discontinuities of size $\mathcal{O}(1/N)$.

More importantly for us, $f'(t)$ is strictly positive for all t (except the discontinuities).

Bus Arrival Problem

So the simulated average wait time f is piecewise linear with N discontinuities of size $\mathcal{O}(1/N)$.

More importantly for us, $f'(t)$ is strictly positive for all t (except the discontinuities).

So, we cannot use $f'(t)$ to help us minimise f ("f is strictly increasing"?!)

Bus Arrival Problem

So the simulated average wait time f is piecewise linear with N discontinuities of size $\mathcal{O}(1/N)$.

More importantly for us, $f'(t)$ is strictly positive for all t (except the discontinuities).

So, we cannot use $f'(t)$ to help us minimise f ("f is strictly increasing"?!)

Finite differencing doesn't help, this is a fundamental problem with the actual function we are trying to differentiate!

Outline

1. The good
2. The bad
3. The ugly
4. **The good (again)**

Noise-Aware Finite Differencing

Problem

How do you calculate useful derivatives for the bus problem?

Noise-Aware Finite Differencing

Problem

How do you calculate useful derivatives for the bus problem?

With rounding errors of size ϵ, we know finite differencing with $h \approx \sqrt{\epsilon}$ is the best choice.

Problem

How do you calculate useful derivatives for the bus problem?

With rounding errors of size ϵ, we know finite differencing with $h \approx \sqrt{\epsilon}$ is the best choice.

Using similar ideas, model the bus problem as a smooth function with noise:

$$\text{Computed value} = \text{True value} + \delta,$$

for some $|\delta| \leq \epsilon$. The finite differencing error now satisfies

$$\text{error} \leq \left| \frac{f(x+h) - f(x)}{h} - f'(x) \right| + \frac{|\delta_1|}{h} + \frac{|\delta_2|}{h} \lesssim \frac{1}{2} h |f''(x)| + \frac{2\epsilon}{h}$$

using the Taylor form for the finite differencing error, $f''(\xi) \approx f''(x)$ since $\xi \in [x, x+h]$.

Noise-Aware Finite Differencing

For the bus problem, we have finite differencing error

$$\text{error} \lesssim \frac{1}{2}h|f''(x)| + \frac{2\epsilon}{h},$$

so a more careful optimal choice of h is given by minimising the RHS to get

$$h^* = \frac{2\sqrt{\epsilon}}{\sqrt{|f''(x)|}}$$

For the bus problem, we have finite differencing error

$$\text{error} \lesssim \frac{1}{2}h|f''(x)| + \frac{2\epsilon}{h},$$

so a more careful optimal choice of h is given by minimising the RHS to get

$$h^* = \frac{2\sqrt{\epsilon}}{\sqrt{|f''(x)|}}$$

There are only two problems with this approach:

- How do we estimate ϵ?
- How do we estimate $f''(x)$? (the whole point of finding h is to be able to estimate derivatives!)

Noise-Aware Finite Differencing

Problem 1

How do we estimate ϵ?

Noise-Aware Finite Differencing

Problem 1

How do we estimate ϵ?

Perhaps surprisingly, we can use finite difference estimates!

Problem 1

How do we estimate ϵ?

Perhaps surprisingly, we can use finite difference estimates!

Theorem (Moré & Wild, 2011)

Assume the noise δ is i.i.d. for each input x (independent & identically distributed), and the true function is continuous. Then

$$\lim_{h\to 0} \frac{(k!)^2}{(2k)!} \cdot \mathbb{E}\left[(D_k f(x))^2\right] = \epsilon^2,$$

where $D_k f(x)$ is a finite difference approximation for $f^{(k)}(x)$.

Noise-Aware Finite Differencing

Problem 1

How do we estimate ϵ?

Perhaps surprisingly, we can use finite difference estimates!

Theorem (Moré & Wild, 2011)

Assume the noise δ is i.i.d. for each input x (independent & identically distributed), and the true function is continuous. Then

$$\lim_{h\to 0} \frac{(k!)^2}{(2k)!} \cdot \mathbb{E}\left[(D_k f(x))^2\right] = \epsilon^2,$$

where $D_k f(x)$ is a finite difference approximation for $f^{(k)}(x)$.

In practice, pick a not-too-small value of h and calculate the LHS for several values of k.

Noise-Aware Finite Differencing

Problem 2

How do we estimate $f''(x)$?

Noise-Aware Finite Differencing

Problem 2

How do we estimate $f''(x)$?

Theorem (Shi et al, 2022)

If h is chosen such that

$$\frac{|f(x+4h) - 4f(x+h) + 3f(x)|}{\epsilon} \in [L, U],$$

for $L > 1$ and $U > L + 2$, then it approximately holds that

$$h \in \left[\frac{\sqrt{L-1}}{\sqrt{3}} \cdot h^*, \frac{\sqrt{U+1}}{\sqrt{3}} \cdot h^*\right]$$

Noise-Aware Finite Differencing

Problem 2

How do we estimate $f''(x)$?

Theorem (Shi et al, 2022)

If h is chosen such that

$$\frac{|f(x+4h)-4f(x+h)+3f(x)|}{\epsilon} \in [L, U],$$

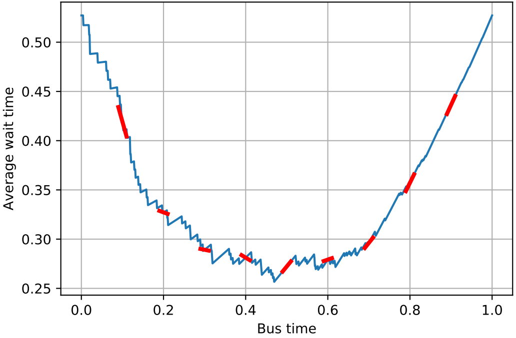
for $L > 1$ and $U > L+2$, then it approximately holds that

$$h \in \left[\frac{\sqrt{L-1}}{\sqrt{3}} \cdot h^*, \frac{\sqrt{U+1}}{\sqrt{3}} \cdot h^*\right]$$

e.g. If $L = 1.5$ and $U = 6$, get $h \in [0.40h^*, 1.53h^*]$. Finding such h is not too hard.

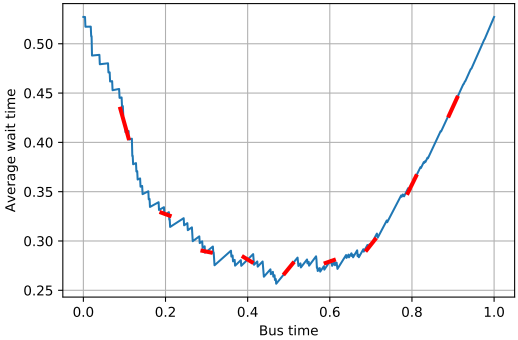
Bus Arrival Problem

Same simulation as before, but use noise-aware finite differencing to draw tangents:



Bus Arrival Problem

Same simulation as before, but use noise-aware finite differencing to draw tangents:



The tangent lines now give useful information about the large-scale behaviour of the function!

Complex Step Finite Differencing

Aside: moving into the complex plane can help with finite differencing!

Complex Step Finite Differencing

Aside: moving into the complex plane can help with finite differencing!

We have

$$\frac{f(x+h)-f(x)}{h} \approx f'(x) + \mathcal{O}(h)$$

unless h is too small that rounding errors matter (because of the subtraction in the numerator).

Complex Step Finite Differencing

Aside: moving into the complex plane can help with finite differencing!

We have

$$\frac{f(x+h)-f(x)}{h} \approx f'(x) + \mathcal{O}(h)$$

unless h is too small that rounding errors matter (because of the subtraction in the numerator).

Suppose we can extend f into the complex plane, $f : \mathbb{C} \to \mathbb{C}$ (e.g. analytic functions like polynomials/trig/exponentials via Taylor series).

Complex Step Finite Differencing

Aside: moving into the complex plane can help with finite differencing!

We have

$$\frac{f(x+h)-f(x)}{h} \approx f'(x) + \mathcal{O}(h)$$

unless h is too small that rounding errors matter (because of the subtraction in the numerator).

Suppose we can extend f into the complex plane, $f : \mathbb{C} \to \mathbb{C}$ (e.g. analytic functions like polynomials/trig/exponentials via Taylor series).

For real x, look at a complex step, $x + ih$ for small h ($i = \sqrt{-1}$).

Complex Step Finite Differencing

Using Taylor series,

$$f(x+ih) = f(x) + ihf'(x) + \frac{1}{2}i^2h^2f''(x) + \frac{1}{6}i^3h^3f'''(x) + \cdots$$

Complex Step Finite Differencing

Using Taylor series,

$$f(x+ih) = f(x) + ihf'(x) + \frac{1}{2}i^2h^2f''(x) + \frac{1}{6}i^3h^3f'''(x) + \cdots$$

Using $i^2 = -1$ and $i^3 = -i$,

$$f(x+ih) = f(x) + ihf'(x) - \frac{1}{2}h^2f''(x) - \frac{1}{6}ih^3f'''(x) + \cdots$$

Complex Step Finite Differencing

Using Taylor series,

$$f(x+ih) = f(x) + ihf'(x) + \frac{1}{2}i^2h^2f''(x) + \frac{1}{6}i^3h^3f'''(x) + \cdots$$

Using $i^2 = -1$ and $i^3 = -i$,

$$f(x+ih) = f(x) + ihf'(x) - \frac{1}{2}h^2f''(x) - \frac{1}{6}ih^3f'''(x) + \cdots$$

Now since $f(x)$, $f'(x)$, etc are all real, take imaginary parts of both sides:

$$\text{Im}[f(x+ih)] = hf'(x) - \frac{1}{6}h^3f'''(x) + \cdots$$

Complex Step Finite Differencing

Using Taylor series,

$$f(x+ih) = f(x) + ihf'(x) + \frac{1}{2}i^2h^2f''(x) + \frac{1}{6}i^3h^3f'''(x) + \cdots$$

Using $i^2 = -1$ and $i^3 = -i$,

$$f(x+ih) = f(x) + ihf'(x) - \frac{1}{2}h^2f''(x) - \frac{1}{6}ih^3f'''(x) + \cdots$$

Now since $f(x)$, $f'(x)$, etc are all real, take imaginary parts of both sides:

$$\text{Im}[f(x+ih)] = hf'(x) - \frac{1}{6}h^3f'''(x) + \cdots$$

This gives a new approximation

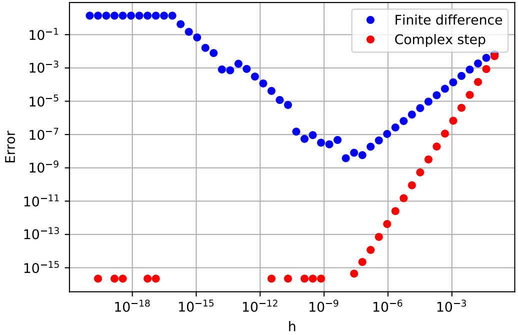
$$\frac{\text{Im}[f(x+ih)]}{h} \approx f'(x) + \mathcal{O}(h^2)$$

Complex Step Finite Differencing

Using complex steps gives a more accurate estimate ($\mathcal{O}(h^2)$ vs. $\mathcal{O}(h)$), and no subtraction means no rounding errors!

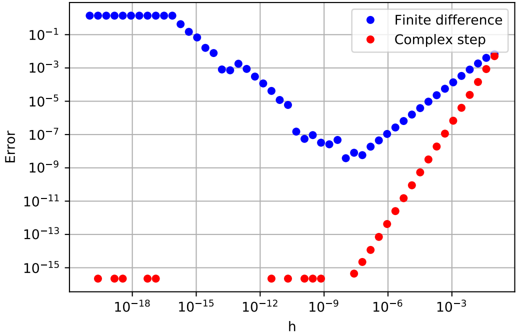
Complex Step Finite Differencing

Using complex steps gives a more accurate estimate ($\mathcal{O}(h^2)$ vs. $\mathcal{O}(h)$), and no subtraction means no rounding errors! For the same test example as before:



Complex Step Finite Differencing

Using complex steps gives a more accurate estimate ($\mathcal{O}(h^2)$ vs. $\mathcal{O}(h)$), and no subtraction means no rounding errors! For the same test example as before:



Complex step is so accurate that measuring the error is subject to rounding issues!

Complex Step Finite Differencing

Effectively, the software implementing complex functions does some symbolic differentiation for us!

Complex Step Finite Differencing

Effectively, the software implementing complex functions does some symbolic differentiation for us!

For example, if $f(x) = \sin(x)$, then (complex) trig identities give

$$f(x+ih) = \sin(x+ih) = \sin(x)\cosh(h) + i\cos(x)\sinh(h),$$

so using the Taylor series for $\sinh(h)$ gives

$$\frac{\mathrm{Im}(f(x+ih))}{h} = \cos(x)\cdot\frac{\sinh(h)}{h} = \cos(x)\cdot\frac{h+\frac{h^3}{6}+\mathcal{O}(h^5)}{h},$$

Complex Step Finite Differencing

Effectively, the software implementing complex functions does some symbolic differentiation for us!

For example, if $f(x) = \sin(x)$, then (complex) trig identities give

$$f(x+ih) = \sin(x+ih) = \sin(x)\cosh(h) + i\cos(x)\sinh(h),$$

so using the Taylor series for $\sinh(h)$ gives

$$\frac{\mathrm{Im}(f(x+ih))}{h} = \cos(x)\cdot\frac{\sinh(h)}{h} = \cos(x)\cdot\frac{h+\frac{h^3}{6}+\mathcal{O}(h^5)}{h},$$

The correct $\cos(x)$ term comes from how the complex sine function handles the real/imaginary parts of its arguments.

Complex Step Finite Differencing

Effectively, the software implementing complex functions does some symbolic differentiation for us!

For example, if $f(x) = \sin(x)$, then (complex) trig identities give

$$f(x+ih) = \sin(x+ih) = \sin(x)\cosh(h) + i\cos(x)\sinh(h),$$

so using the Taylor series for $\sinh(h)$ gives

$$\frac{\operatorname{Im}(f(x+ih))}{h} = \cos(x)\cdot\frac{\sinh(h)}{h} = \cos(x)\cdot\frac{h+\frac{h^3}{6}+\mathcal{O}(h^5)}{h},$$

The correct $\cos(x)$ term comes from how the complex sine function handles the real/imaginary parts of its arguments.

Using the low-level implementation of component functions to build up symbolic derivatives is the idea behind backpropagation, a key tool in machine learning.

Conclusions & Future Work

Conclusions

- Differentiation is surprisingly difficult!
- Finite differencing can give good approximations but care is needed
- Can automatically build up symbolic derivatives (complex step/backpropagation)

Other interesting topics

- How does backpropagation work?
- Derivatives of multidimensional functions or higher-order derivatives
- Generalisations of derivatives: functional analysis (Fréchet/Gateaux derivatives), convex analysis (subgradients), fractional calculus, ...
- How to use derivatives to optimise functions

Further Study

Can you study these topics in your degree?

- MATH2X70 — Optimisation and Financial Mathematics
- MATH3X76 — Mathematical Computing
- MATH4411 — Applied Computational Mathematics

References i

R. L. Burden and J. D. Faires, *Numerical Analysis*, Springer Series in Operations Research and Financial Engineering, Brooks/Cole, Boston, 9th ed ed., 2011.

S. Kim, R. Pasupathy, and S. G. Henderson, *A guide to sample average approximation*, in Handbook of Simulation Optimization, M. C. Fu, ed., vol. 216 of International Series in Operations Research & Management Science, Springer New York, New York, NY, 2015.

J. J. Moré and S. M. Wild, *Estimating computational noise*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1292–1314.

J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2nd ed ed., 2006.

H.-J. M. Shi, Y. Xie, M. Q. Xuan, and J. Nocedal, *Adaptive finite-difference interval estimation for noisy derivative-free optimization*, SIAM Journal on Scientific Computing, 44 (2022), pp. A2302–A2321.

W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM Review, 40 (1998), pp. 110–112.