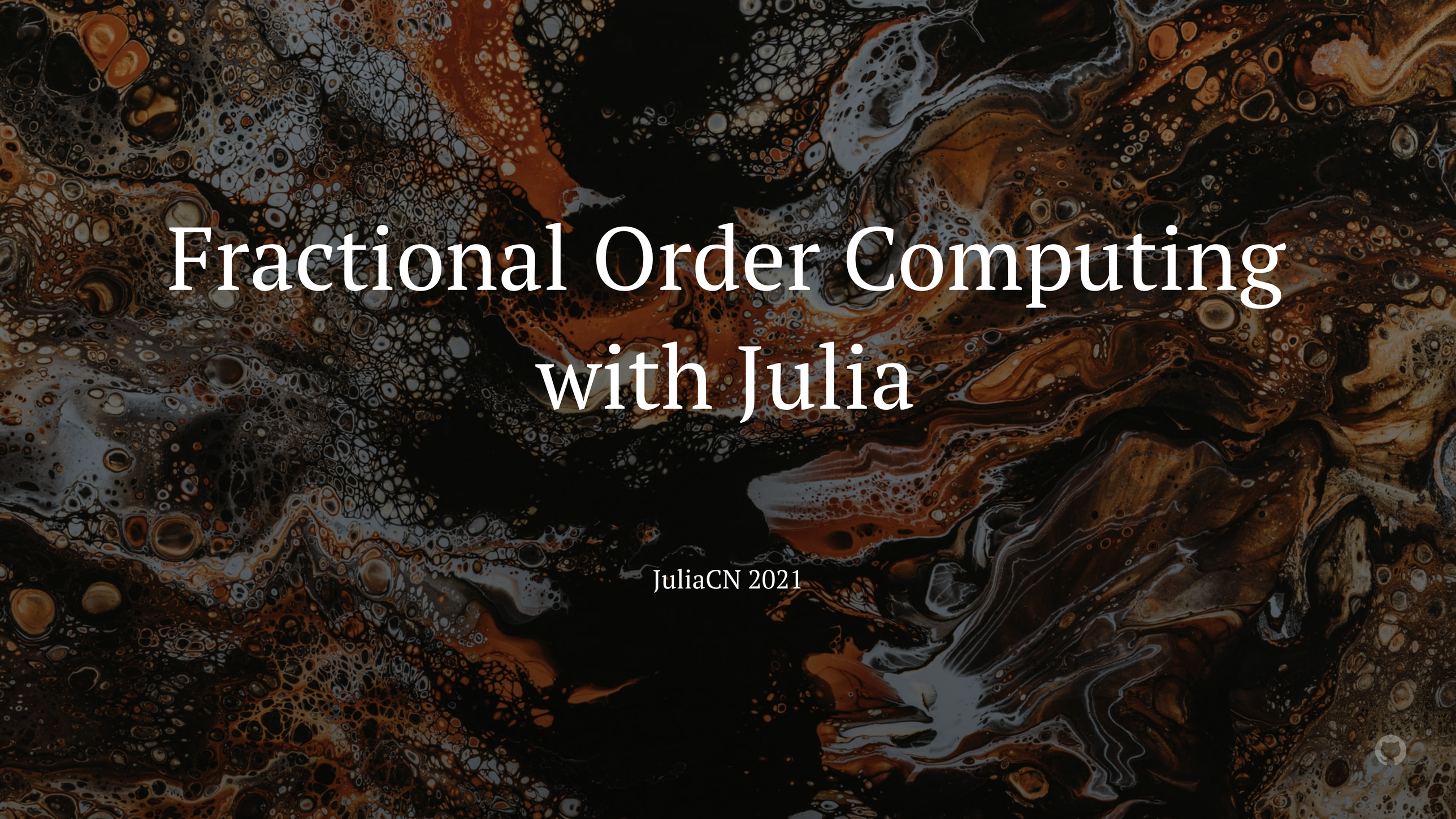


Fractional Order Computing with Julia

JuliaCN 2021

OutLine

1. SciFracX introduction
2. History of fractional calculus and fractional differential equations
3. Get hands on FractionalCalculus.jl
4. Get hands on FractionalDiffEq.jl
5. One more thing

What is SciFracX?

SciFracX is an orgnization aiming at deploying the advantages of Julia to fractional computing area.

Scientific Fractional Order Computing

- There are huge gap between theoretical researches and applications in fractional scientific computing area. SciFracX is aiming at filling the gap and bridge the theoretical and application.
- No deadline, no pushing, just a labor of love.

Fractional Calculus

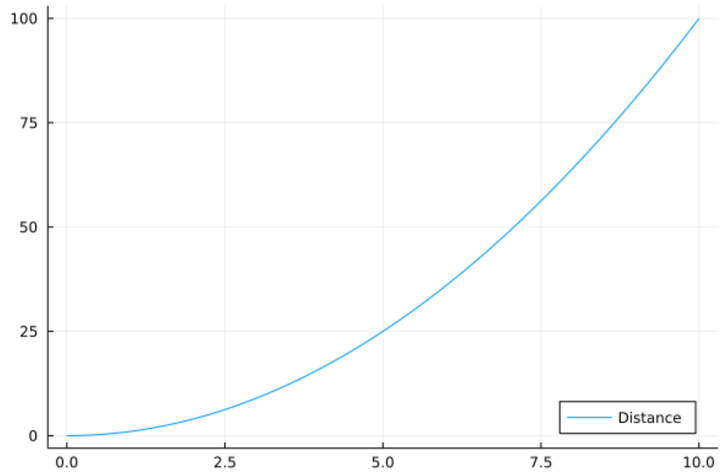
Classical Calculus

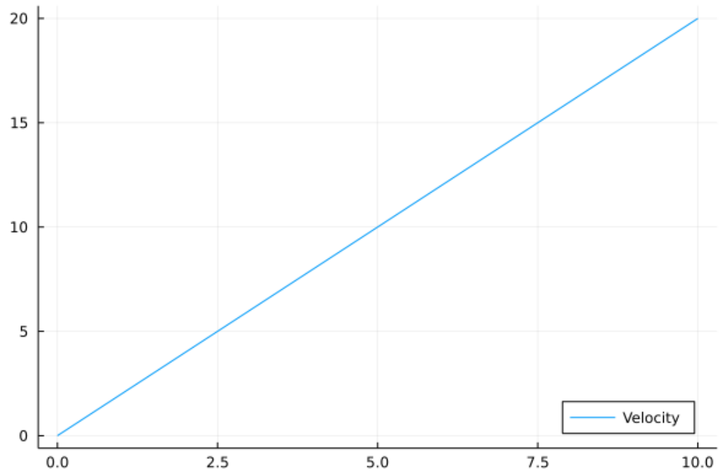
We already learned classical calculus

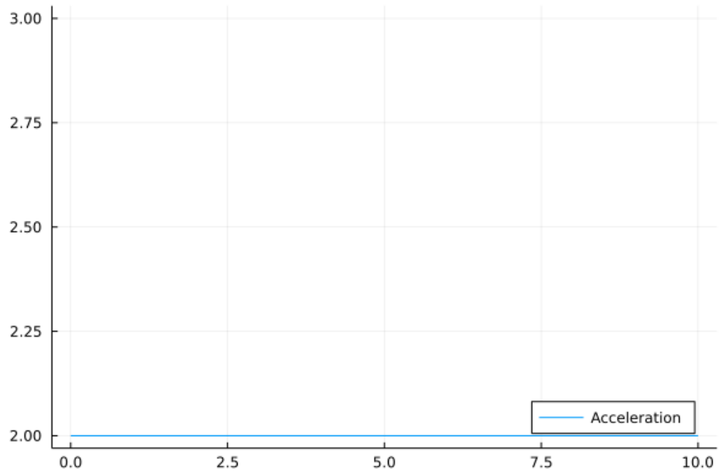
The derivative and integral has its geometric illustration and meaning.

$$\frac{d}{dt}v = a$$

$$\int_{t_1}^{t_2} v dt = s$$







Meaning

Then what is the meaning of semi-order derivative or a quarter integral?🤔

Let’s go back to three handreds years ago

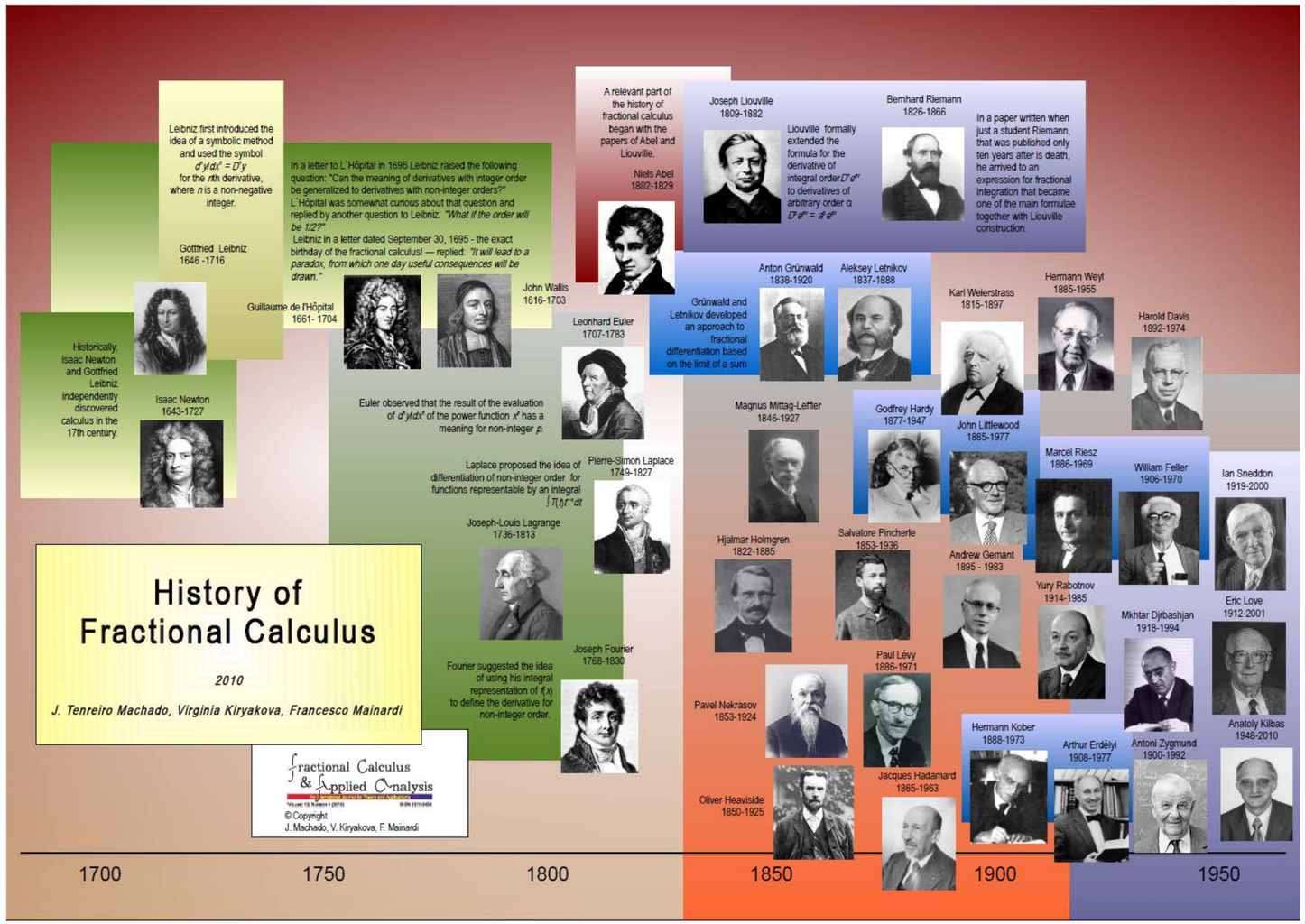
In a letter to L’Hôpital in 1695
Leibniz raised the question:
"Can the meaning of derivatives with integer order be generalized to derivatives with non-integer orders?"

L`Hôpital was curious about that question and replied by another question: "What if the order will be $\frac{1}{2}$?"
Leibniz in a letter dated September 30, 1695 — the exact birthday of the fractional calculus! — replied:

"It will lead to a paradox, from which one day useful consequences will be drawn."

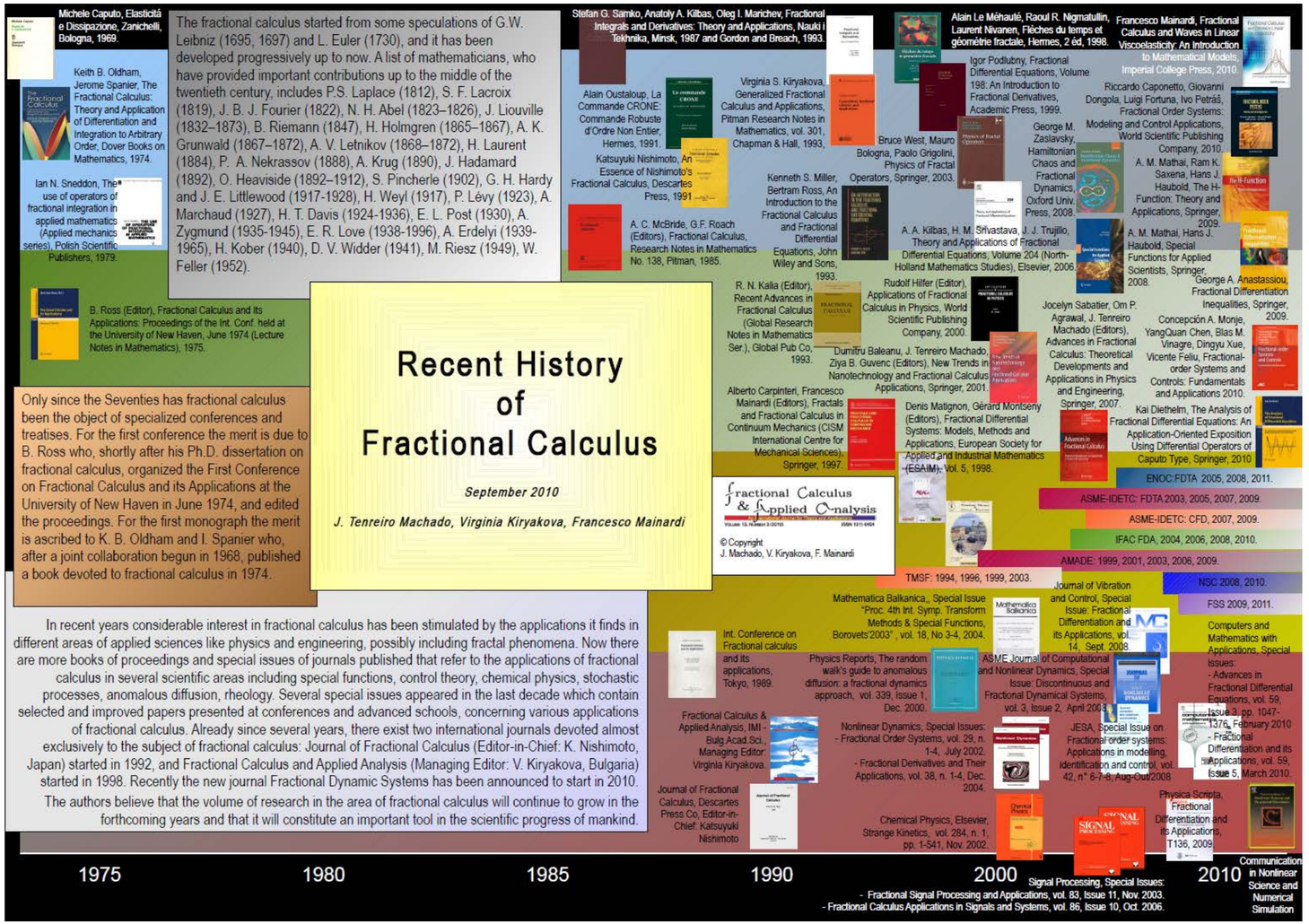
An outlook of Fractional Calculus

"Old history"



An outlook of Fractional Calculus

"Recent history"



Basic introductioon

Fractional Calculus here means arbitrary order in fact, **fractions**, **complex**, **arbitrary**

We first think about how the fractional derivative is being obtained.

Suppose we want the fractional derivative of a power function: $f(x) = x^k$

Using the basic calculus we have learned is enough:

The first derivative:

$$f'(x) = \frac{d}{dx} f(x) = kx^{k-1}$$

And the second:

$$f''(x) = \frac{d^2}{dx^2} f(x) = k^2 x^{k-2}$$

We can know the n-th derivative:

$$f^{(n)}(x) = \frac{d^n}{dx^n} x^k = \frac{k!}{(k-n)!} x^{k-n} = \frac{\Gamma(k+1)}{\Gamma(k-n+1)} x^{k-n}$$

So we replace $\mathbf{n}$ with $\frac{1}{2}$, we can get the semi-derivative of the power function:

$$f^{(\frac{1}{2})}(x) = \frac{\Gamma(2)}{\Gamma(\frac{3}{2})} x^{\frac{1}{2}} = \frac{2}{\sqrt{\pi}} x^{\frac{1}{2}}$$

Bingo 🎉 !! You get the semi-derivative of $f(x) = x$!

Quick Start

Let's get start!!

Install FractionalCalculus.jl in REPL package manager with entering `]`

```
Pkg> add FractionalCalculus
julia> using FractionalCalculus
```

Then if you want to calculuate the semi derivative of $f(x) = x^2$ at $x = 1$:

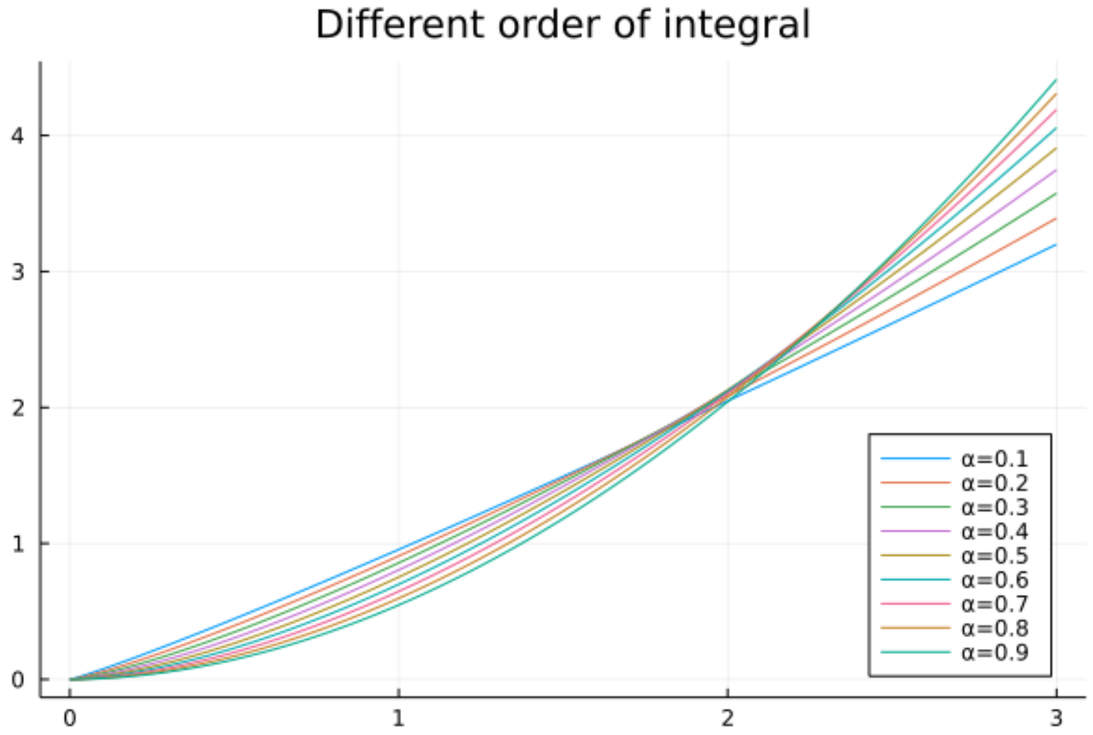
```
julia> fracdiff(x->x^2, 0.5, 1, 0.001, RLDiff_Approx())
1.5044908143658473
```

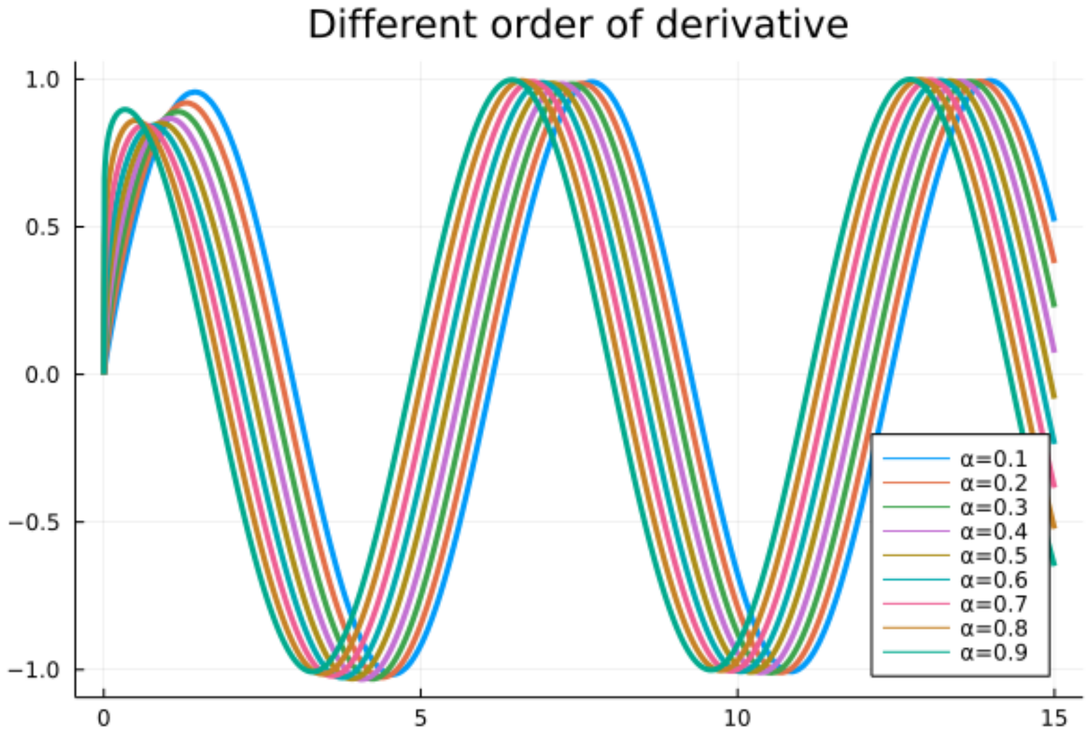
Or a quarter integral of $f(x) = \sin x$ ay $x = 1$? Not a problem!

```
julia> fracint(sin, 0.25, 3, 0.001, RLInt_Approx())
0.6197918550987523
```

Plots

Let’s plot different order integral and derivative:





Different sense of fractional derivative and integral

Thanks to the hard work of pioneers, there are many sense of fractional derivative and integral.

FractionalCalculus.jl has supports for many sense of fractional derivative and fractional integral:

Fractional derivative

- Caputo fractional derivative
- Grunwald-Letnikov fractional derivative
- Riemann-Liouville fractional derivative
- Riesz fractional derivative
- Hadamard fractional derivative

Fractional integral

- Riemann-Liouville fractional integral

Current Algorithms

Fractional Derivative

- Caputo_Piecewise
- Caputo_Diethelm
- GL_Multiplicative_Additive
- GL_Lagrange_Three_Point_Interp
- RLDiff_Approx
- RLDiff_Matrix
- Hadamard_LRect
- Hadamard_RRect
- Hadamard_Trap
- Riesz_Symmetric

Fractional Integral

- RLDiff_Approx
- RL_Piecewise
- RL_LinearInterp
- RLInt_Approx
- RLInt_Matrix
- RLInt_Simpson
- RLInt_Trapezoidal
- RLInt_Rectangular
- RLInt_Cubic_Spline_Interp

Fractional Differential Equations

Fractional Differential Equations

A generalization of differential equation

Fractional differential equation can be seen as the generalization of differential equations. In our daily life, models are usually better described in fractional differential equations.

Getting start!

Let's learn how to use **FractionalDiffEq.jl** to easily solve a FDE

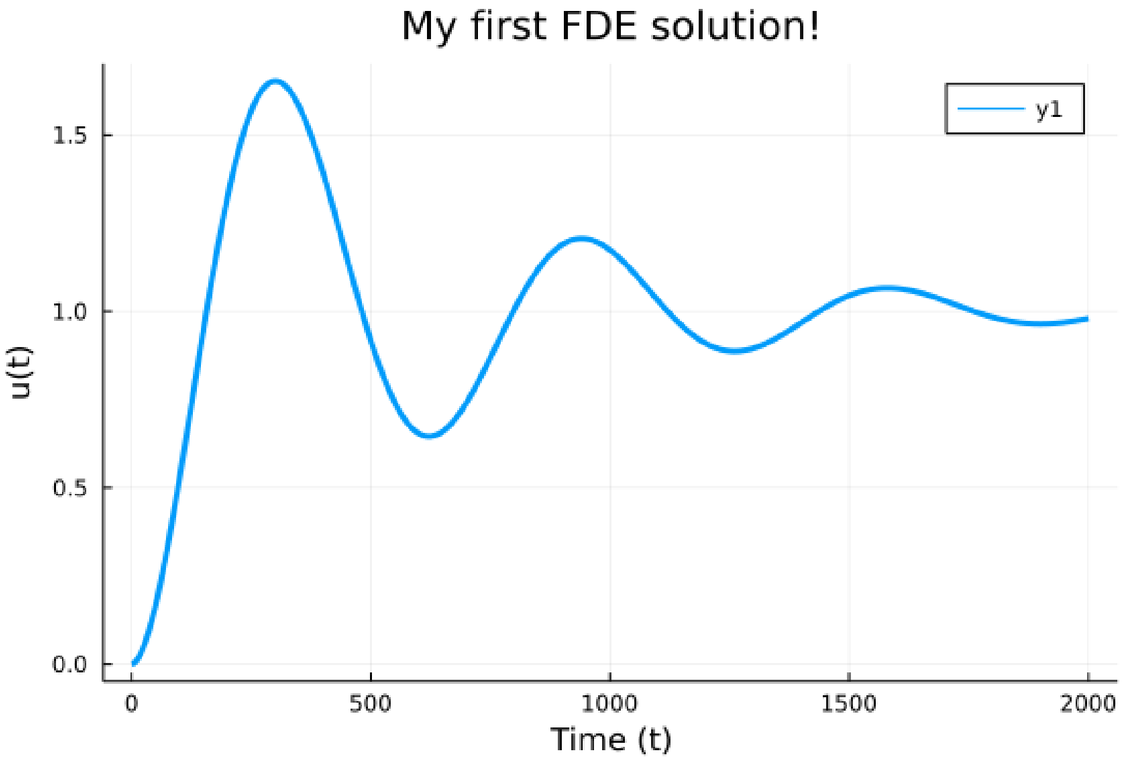
$$D^{1.8}y(x) + y(x) = 1$$

$$y(0) = 0$$

We can use the PECE(Predict-Evaluate-Correct-Evaluate) method:

```
using FractionalDiffEq, Plots, LaTeXStrings
s = "My first FDE solution"
fun(t, u)=1-u
prob = FODEProblem(fun, 1.8, 0, 5, 0.01)
sol = solve(prob, PECE())
plot(sol, linewidth=5, title="My first FDE solution!",
    xaxis="Time (t)", yaxis="u(t)")
```

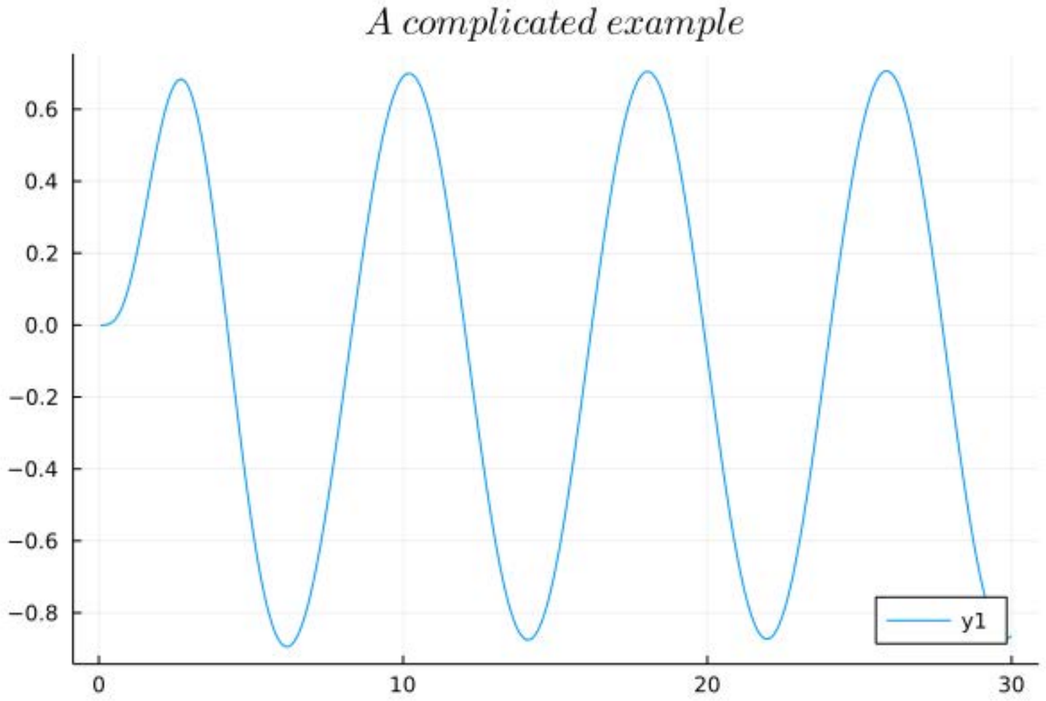
Then plot the solution!!



Or multi-term FDE? **No problem**!

$$y'''(t) + \frac{1}{16}{}_0^C D_t^{2.5} y(t) + \frac{4}{5}y''(t) + \frac{3}{2}y'(t) + \frac{1}{25}{}_0^C D_t^{0.5} y(t) + \frac{6}{5}y(t) = \frac{172}{125}\cos(\frac{4t}{5})$$

```
system = D(30, 3, h)+1/16*D(30, 2.5, h)+4/5*D(30, 2, h)+3/2*D(30, 1, h)+1/25*D(30, 0.5, h)+6/5*D(30, 1, h);
rightfun = 172/125*cos(4/5*x)
result = solve(system, rightfun, 3, 0.05, 30, FODEMatrixDiscrete())
```

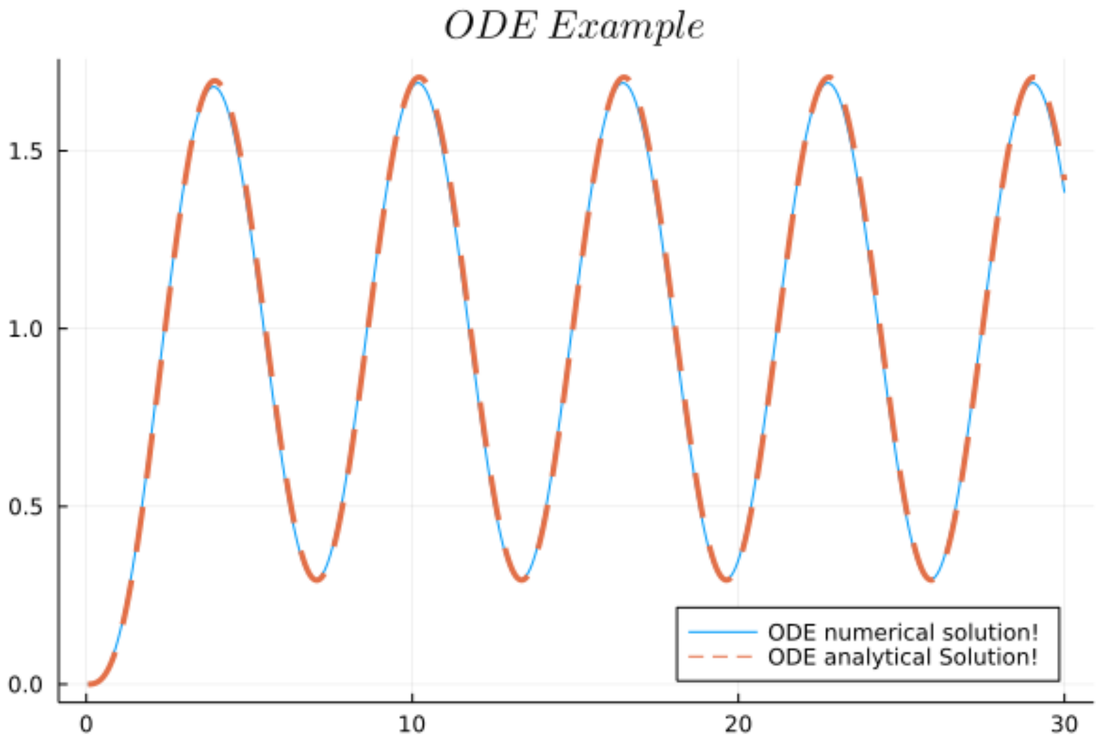


A complicated example

Or solve Ordinary Differential Equations? Just do it~

$$y''(x) + y'(x) = \sin(x)$$
$$y(0) = 0$$

```
using FractionalDiffEq
using Plots, LaTeXStrings

s="\$ODE\\ Example\$"

T = 30
h=0.05
tspan = collect(0.05:h:T)

f(x)=1/2*(-exp(-x)-sin(x)-cos(x)+2)
target=f.(tspan)

eq = D(600, 2, h)+D(600, 1, h)
rightfun(x) = sin(x)
result = solve(eq, rightfun, 2, h, T, FODEMatrixDiscrete())

plot(tspan, result, title=s, legend=:bottomright, label="OD

plot!(tspan, target, lw=3,ls=:dash,label="ODE Analytical Sc
```



Current status

As a young organization and all of these packages are just version **0.1.***, all of the APIs and algorithms are under heavy development, all kinds of contributions are sincerely welcomed.

I believe power of community is great, a group of people with the same interest working together for the same goal.(We prepared contribution gifts for contributors~~)

Road Map

- Fractional Partial Differential Equations
- More robust algorithms and better design

Welcome to join our community!!

One more thing!

How can we get derivative??

Finite Difference? $f'(x) = \frac{f(x+h)-f(x-h)}{2h}$ Round Off Error 😔

Symbolic Difference? Large Computation Cost 😔

Auto Differentiation? 😔

Complex Step Differentiation!!🙌

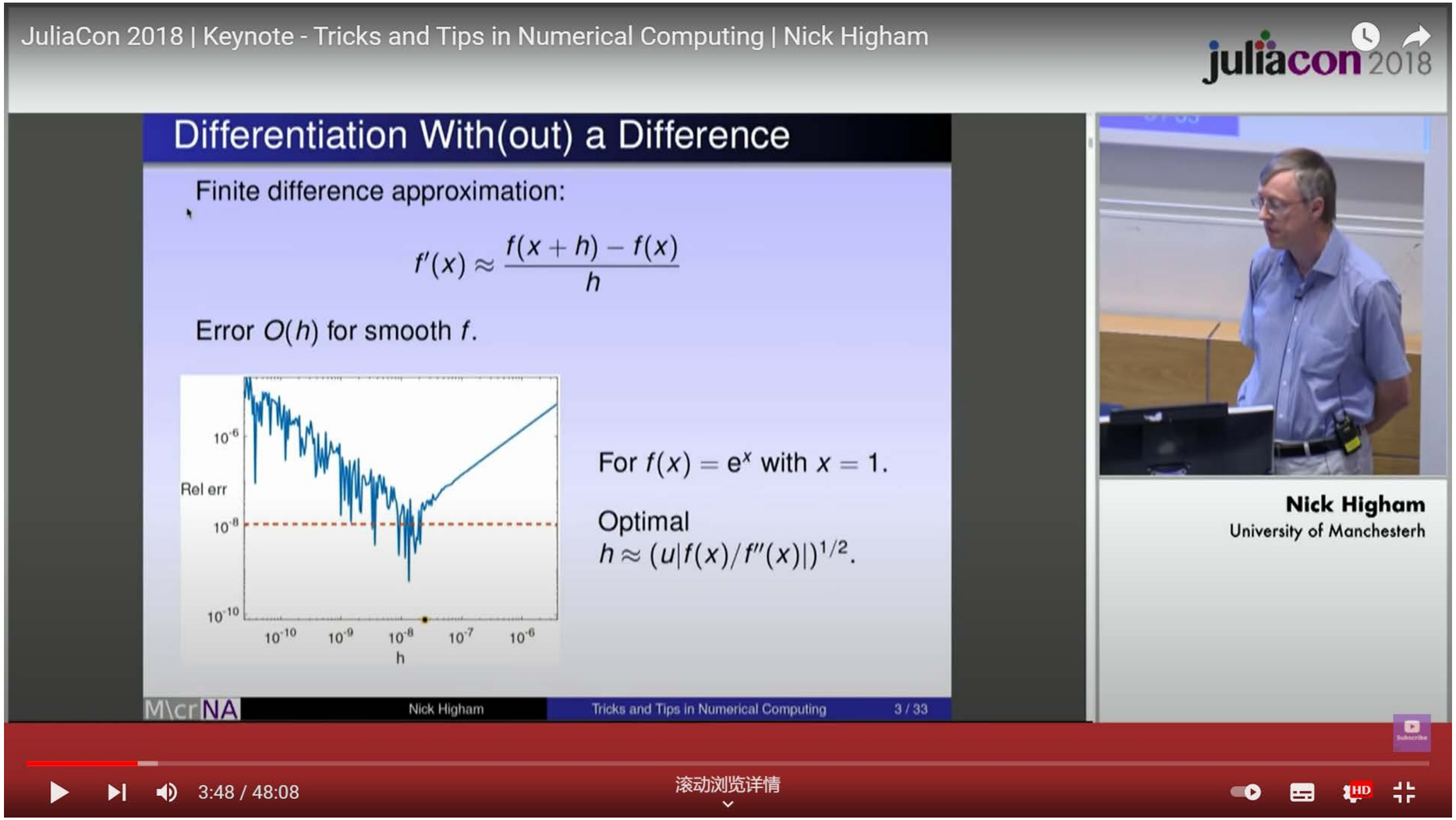
$$f'(x) = \frac{\mathcal{I}m(f(x+ih))}{h}$$

Very pleased to bring **ComplexDiff.jl** !!

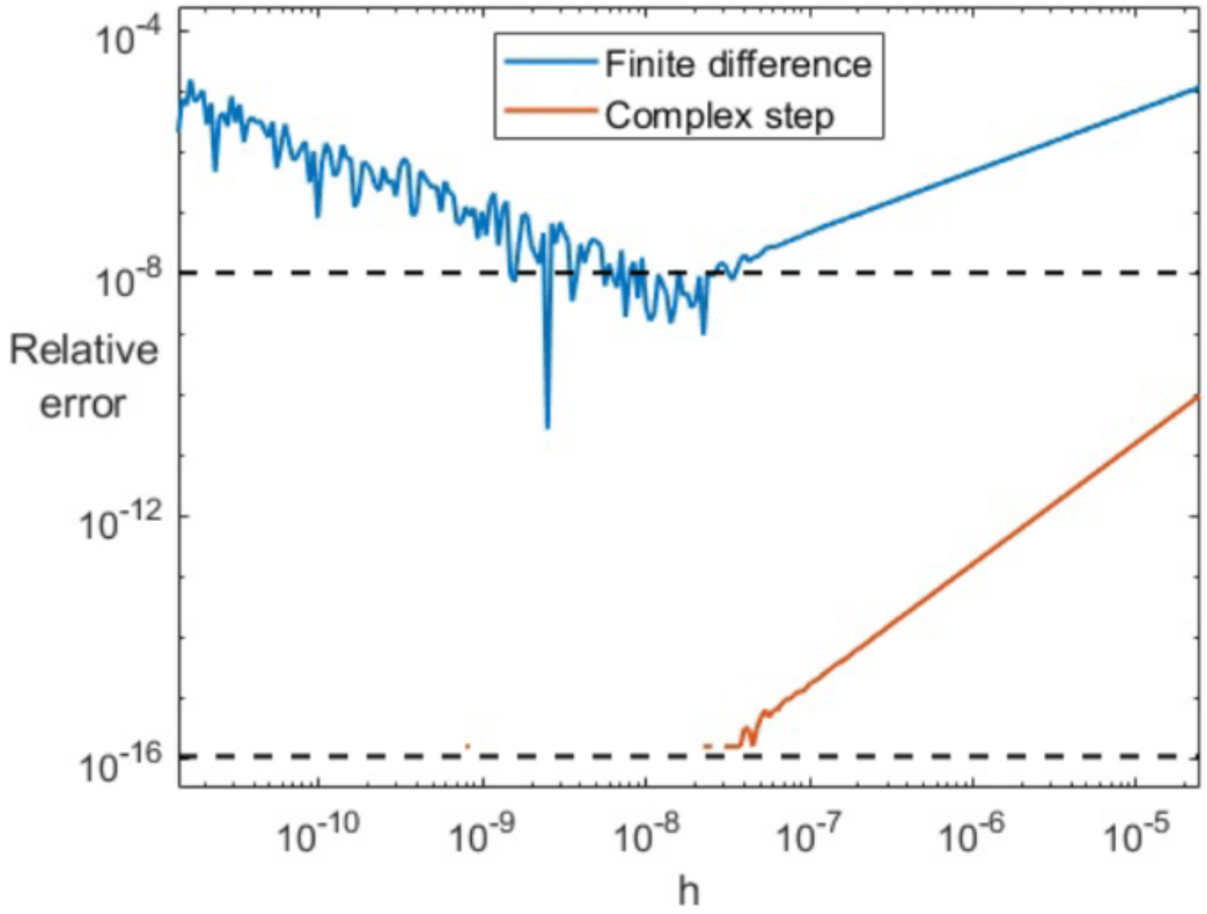
The complex step differentiation thoughts comes from **Cleve Moler**.

JuliaCon talk on Complex Step Differentiation

It is amazing!!!



Compasion with finite difference:



We can see the round off error in Complex Step Differentiation is getting smaller and smaller when **h** becomes extremely small, while the error of finite difference is getting bigger and bigger.

The Complex Step Differentiation method also appeared in the Deep Learning Book

If one has access to numerical computation on complex numbers, then there is a very efficient way to numerically estimate the gradient by using complex numbers as input to the function (Squire and Trapp, 1998). The method is based on the observation that

$$f(x + i\epsilon) = f(x) + i\epsilon f'(x) + O(\epsilon^2), \tag{11.8}$$

$$\text{real}(f(x + i\epsilon)) = f(x) + O(\epsilon^2), \quad \text{imag}(\frac{f(x + i\epsilon)}{\epsilon}) = f'(x) + O(\epsilon^2), \tag{11.9}$$

where $i = \sqrt{-1}$. Unlike in the real-valued case above, there is no cancellation effect because we take the difference between the value of f at different points. This allows the use of tiny values of ϵ, like $\epsilon = 10^{-150}$, which make the $O(\epsilon^2)$ error insignificant for all practical purposes.

With complex arithmetic computing built inside, Julia is absolutely a good choice!

Multi-Complex Step Differentiation

The generalization of **Single**complex to **Multi**complex

$$\mathbb{C}_n = \{\zeta_n = \zeta_{n-1,1} + \zeta_{n-1,2} \cdot i_n, \quad \zeta_{n-1,1}, \zeta_{n-1,2} \in \mathbb{C}_{n-1}\}$$

```
import Base: +, -, *, /
import Base: sin, cos, tan, cot
import Base: sinh, cosh, tanh

+(bi1::bicomplex, bi2::complex) = mat2bicomplex(mat(bi1) + mat(bi2))
-(bi1::bicomplex, bi2::complex) = mat2bicomplex(mat(bi1) - mat(bi2))
```

With fully utilizing multiple dispatch and operator overloading, we can achieve high order derivative computing.