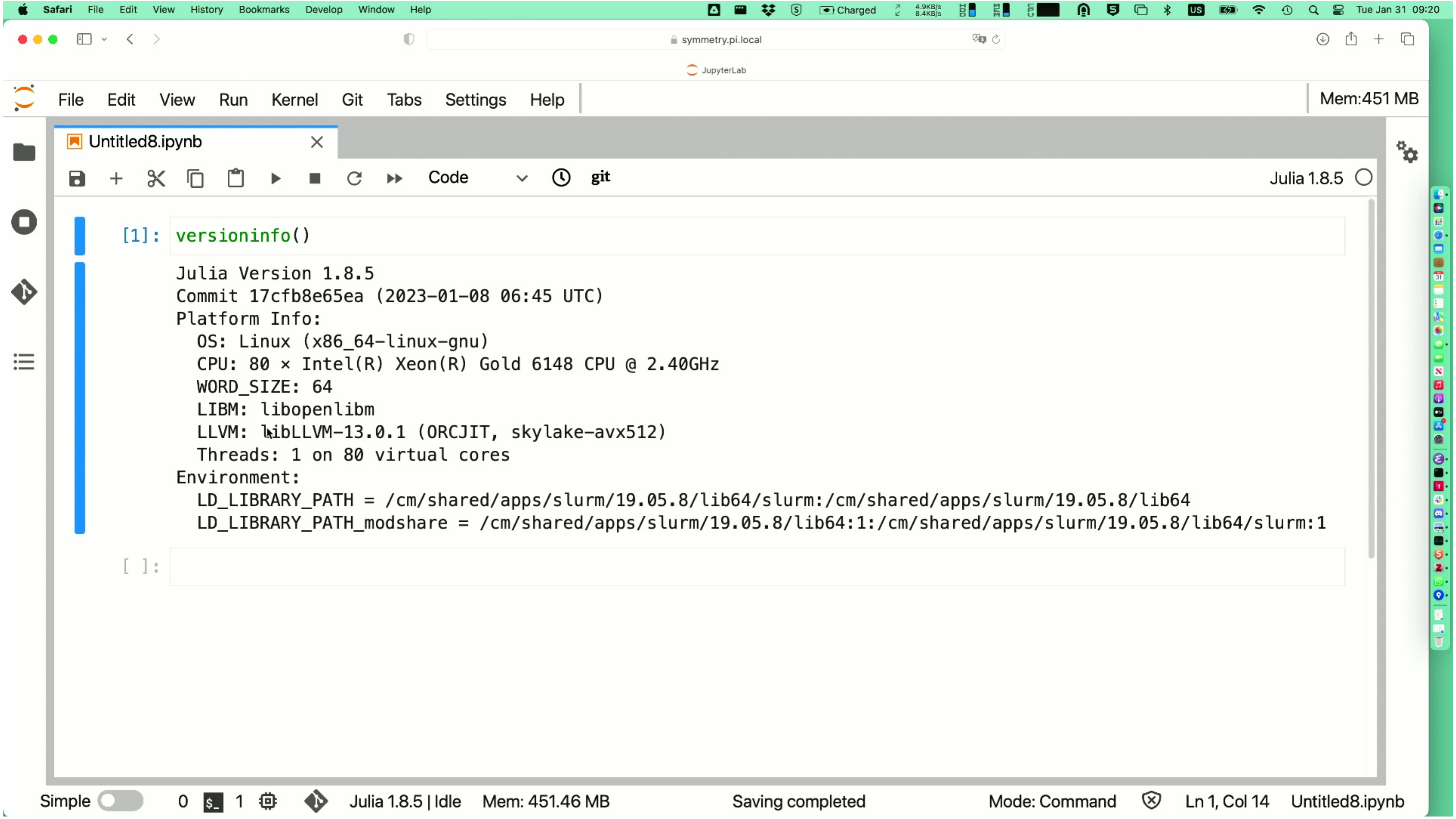
Title: Numerical Methods Lecture - 230131

Speakers: Erik Schnetter

Collection: Numerical Methods (2022/2023)

Date: January 31, 2023 - 9:15 AM

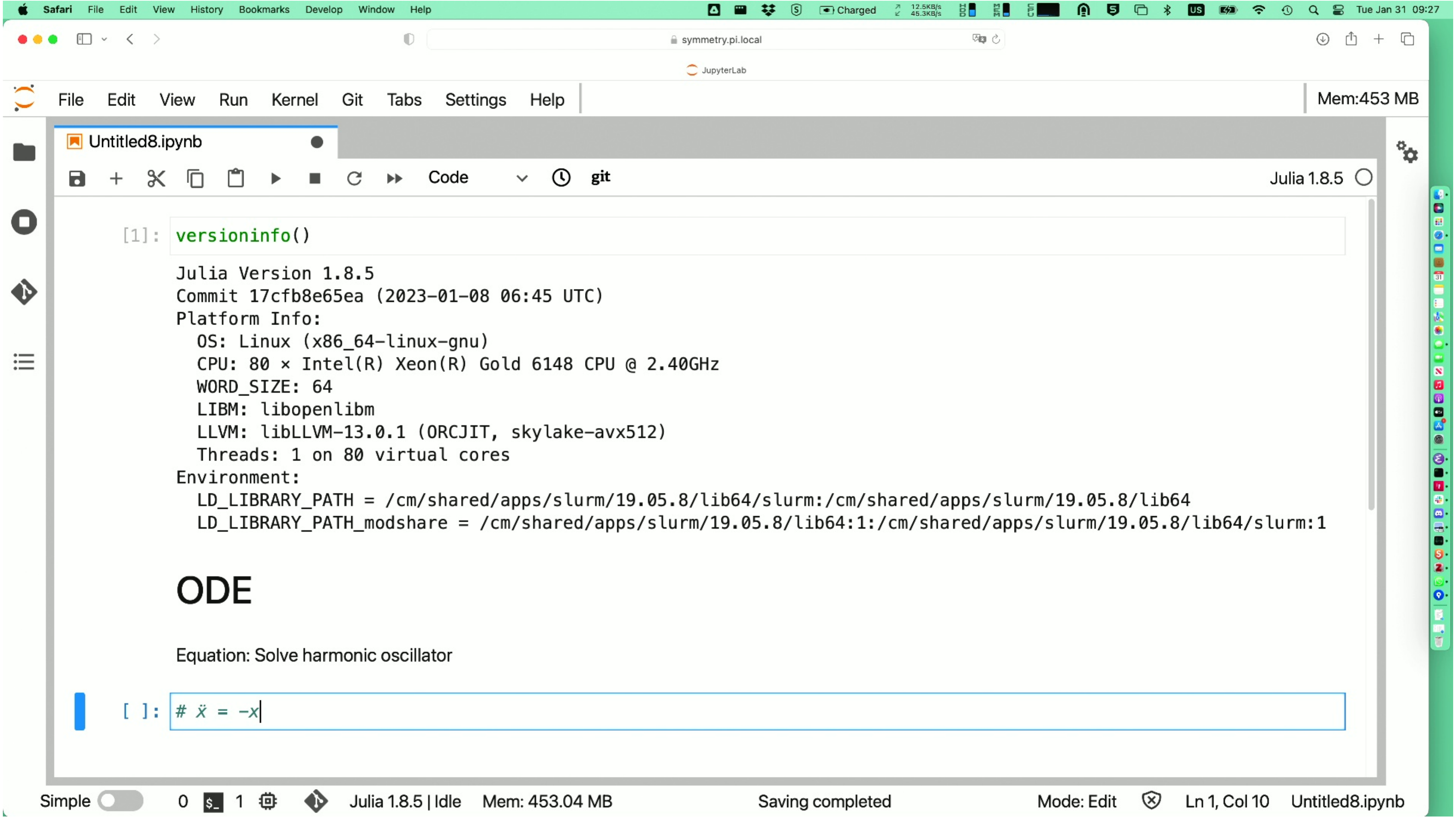
URL: https://pirsa.org/23010008


Safari File Edit View History Bookmarks Develop Window Help
Charged
Tue Jan 31 09:20
symmetry.pi.local
JupyterLab
File Edit View Run Kernel Git Tabs Settings Help
Mem:451 MB
Untitled8.ipynb
Code
git
Julia 1.8.5

```
[1]: versioninfo()
```

```
Julia Version 1.8.5
Commit 17cfb8e65ea (2023-01-08 06:45 UTC)
Platform Info:
  OS: Linux (x86_64-linux-gnu)
  CPU: 80 × Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-13.0.1 (ORCJIT, skylake-avx512)
  Threads: 1 on 80 virtual cores
Environment:
  LD_LIBRARY_PATH = /cm/shared/apps/slurm/19.05.8/lib64/slurm:/cm/shared/apps/slurm/19.05.8/lib64
  LD_LIBRARY_PATH_modshare = /cm/shared/apps/slurm/19.05.8/lib64:1:/cm/shared/apps/slurm/19.05.8/lib64/slurm:1
```

```
[ ]:
```

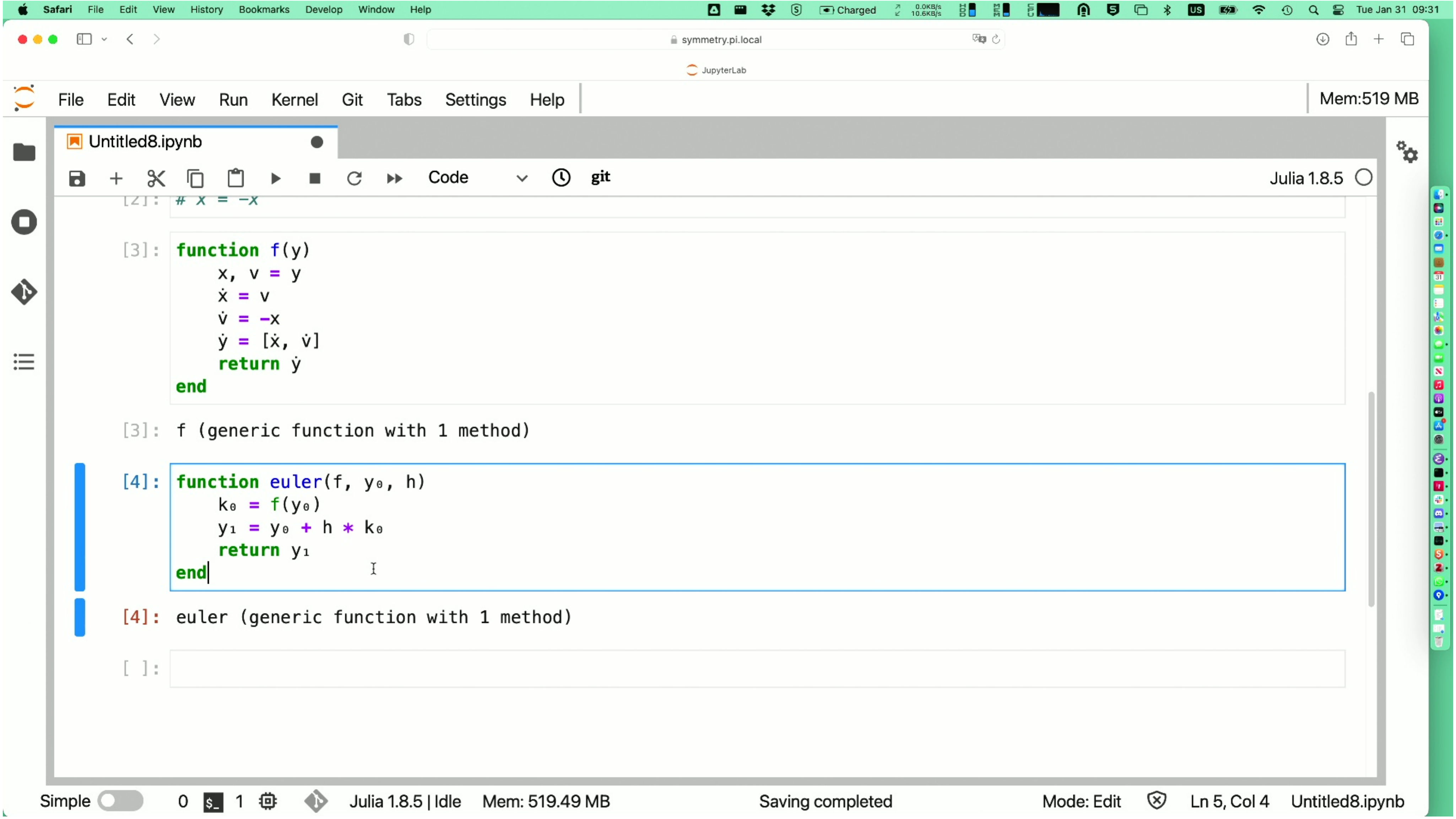


```
[1]: versioninfo()

Julia Version 1.8.5
Commit 17cfb8e65ea (2023-01-08 06:45 UTC)
Platform Info:
  OS: Linux (x86_64-linux-gnu)
  CPU: 80 × Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-13.0.1 (ORCJIT, skylake-avx512)
  Threads: 1 on 80 virtual cores
Environment:
  LD_LIBRARY_PATH = /cm/shared/apps/slurm/19.05.8/lib64/slurm:/cm/shared/apps/slurm/19.05.8/lib64
  LD_LIBRARY_PATH_modshare = /cm/shared/apps/slurm/19.05.8/lib64:1:/cm/shared/apps/slurm/19.05.8/lib64/slurm:1
```

ODE

Equation: Solve harmonic oscillator

```
[ ]: # ẍ = -x
```



```
[2]: # x = −x
```

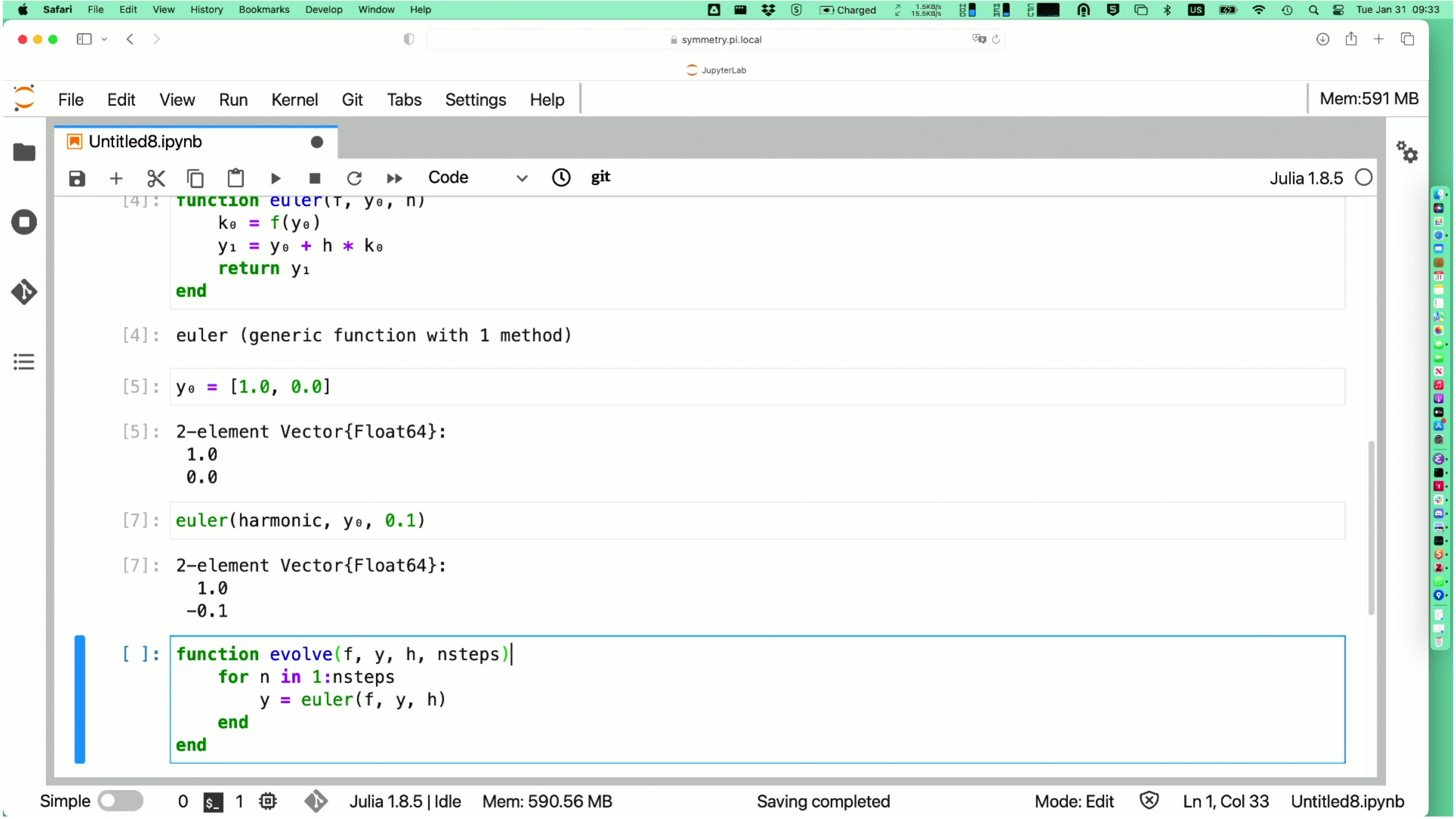
```
[3]: function f(y)
         x, v = y
         ẋ = v
         v̇ = −x
         ẏ = [ẋ, v̇]
         return ẏ
     end
```

```
[3]: f (generic function with 1 method)
```

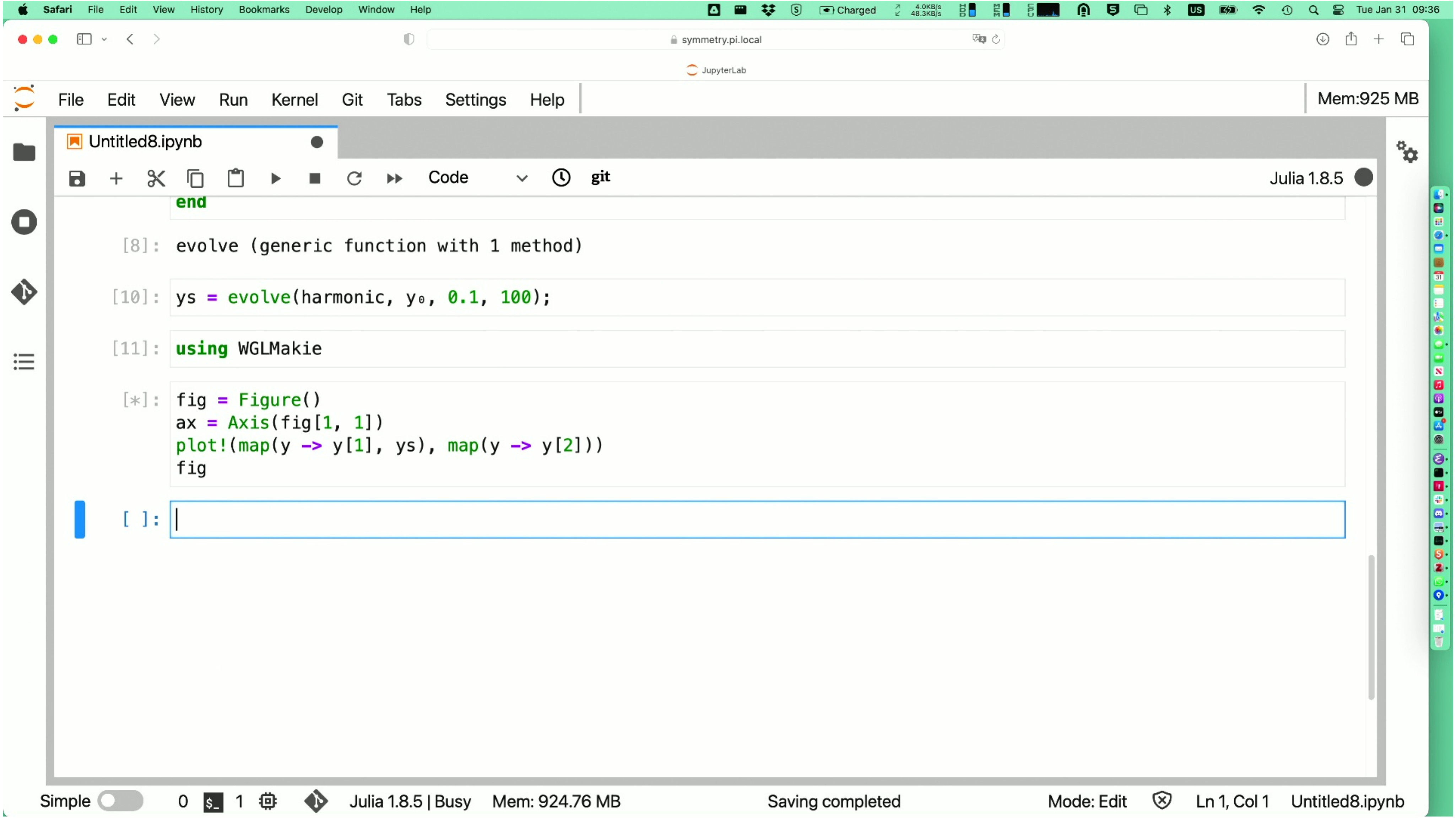
```
[4]: function euler(f, y₀, h)
         k₀ = f(y₀)
         y₁ = y₀ + h * k₀
         return y₁
     end
```

```
[4]: euler (generic function with 1 method)
```

```
[ ]:
```



```
[4]: function euler(f, y₀, h)
         k₀ = f(y₀)
         y₁ = y₀ + h * k₀
         return y₁
     end

[4]: euler (generic function with 1 method)

[5]: y₀ = [1.0, 0.0]

[5]: 2-element Vector{Float64}:
      1.0
      0.0

[7]: euler(harmonic, y₀, 0.1)

[7]: 2-element Vector{Float64}:
       1.0
      -0.1

[ ]: function evolve(f, y, h, nsteps)
         for n in 1:nsteps
             y = euler(f, y, h)
         end
     end
```


Safari File Edit View History Bookmarks Develop Window Help
Charged
Tue Jan 31 09:36
symmetry.pi.local
JupyterLab
File Edit View Run Kernel Git Tabs Settings Help
Mem:925 MB
Untitled8.ipynb
Code
git
Julia 1.8.5

```
    end
```

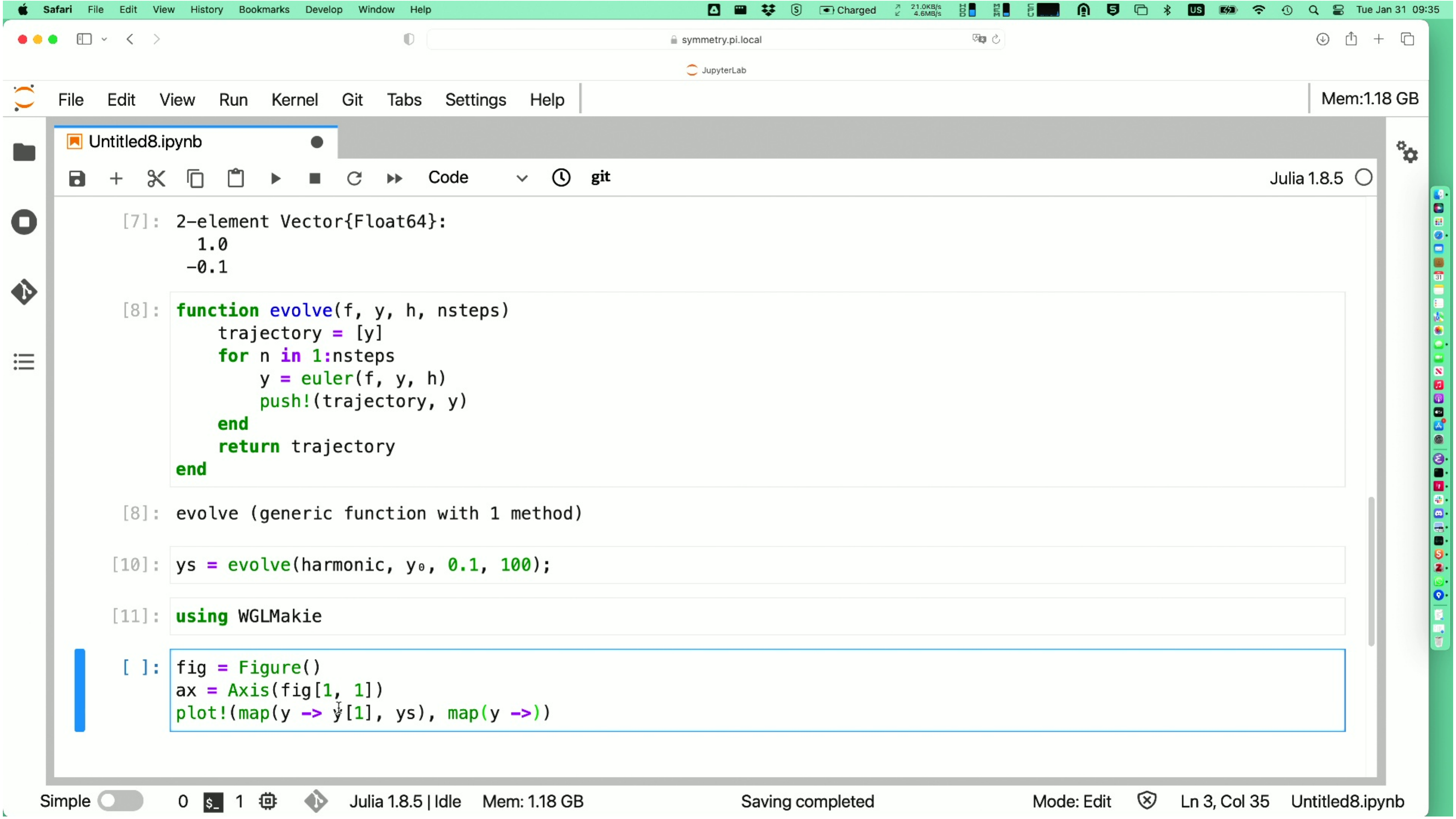
```
[8]: evolve (generic function with 1 method)
```

```
[10]: ys = evolve(harmonic, y₀, 0.1, 100);
```

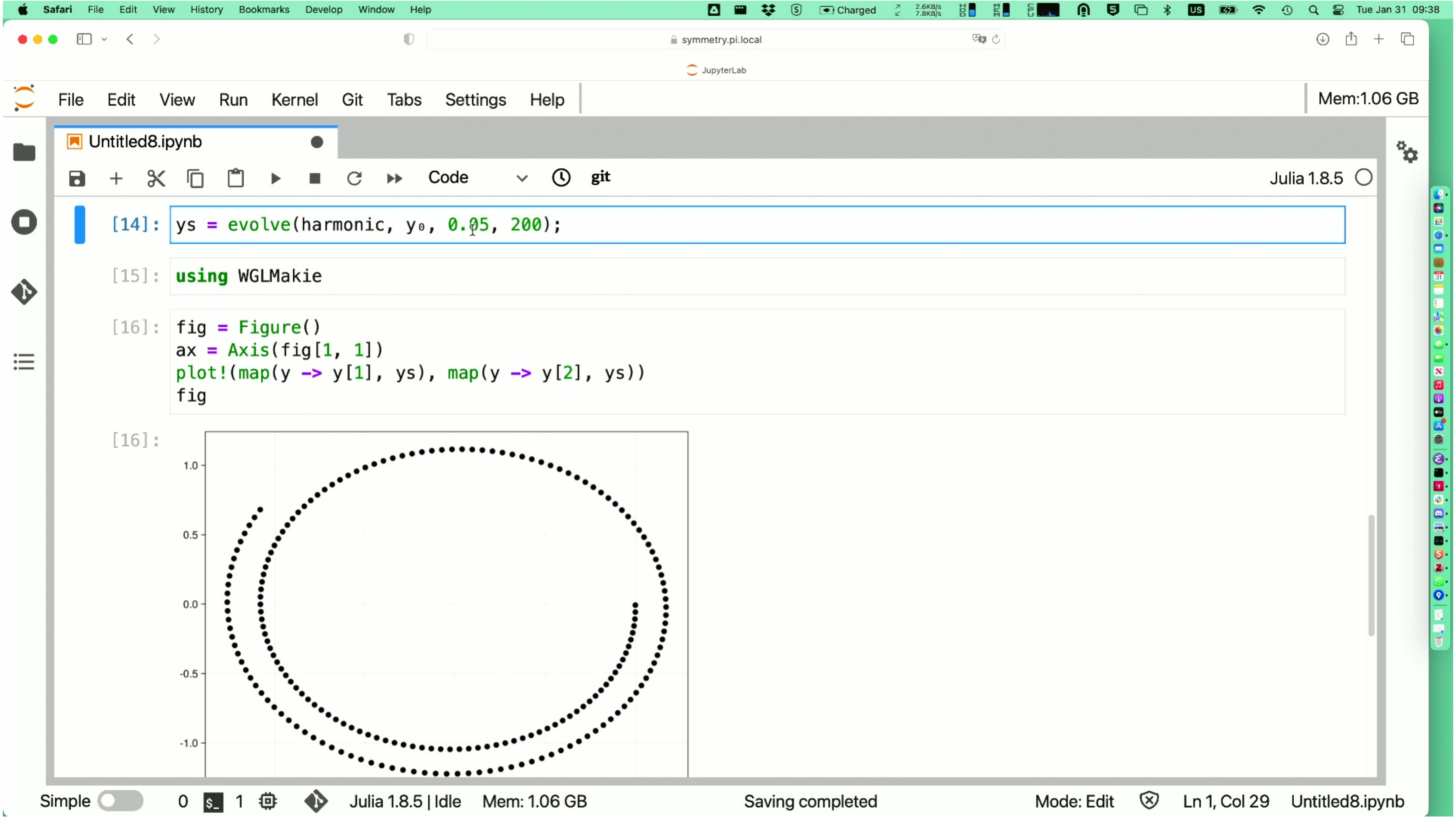
```
[11]: using WGLMakie
```

```
[*]: fig = Figure()
     ax = Axis(fig[1, 1])
     plot!(map(y -> y[1], ys), map(y -> y[2]))
     fig
```

```
[ ]:
```



```
[7]: 2-element Vector{Float64}:
       1.0
      -0.1

[8]: function evolve(f, y, h, nsteps)
         trajectory = [y]
         for n in 1:nsteps
             y = euler(f, y, h)
             push!(trajectory, y)
         end
         return trajectory
     end

[8]: evolve (generic function with 1 method)

[10]: ys = evolve(harmonic, y₀, 0.1, 100);

[11]: using WGLMakie

[ ]: fig = Figure()
     ax = Axis(fig[1, 1])
     plot!(map(y -> y[1], ys), map(y ->))
```


Safari File Edit View History Bookmarks Develop Window Help
Charged
Tue Jan 31 09:38
symmetry.pi.local
JupyterLab
File Edit View Run Kernel Git Tabs Settings Help
Mem:1.06 GB
Untitled8.ipynb
Code
git
Julia 1.8.5
1.0
0.5
0.0
-0.5
-1.0

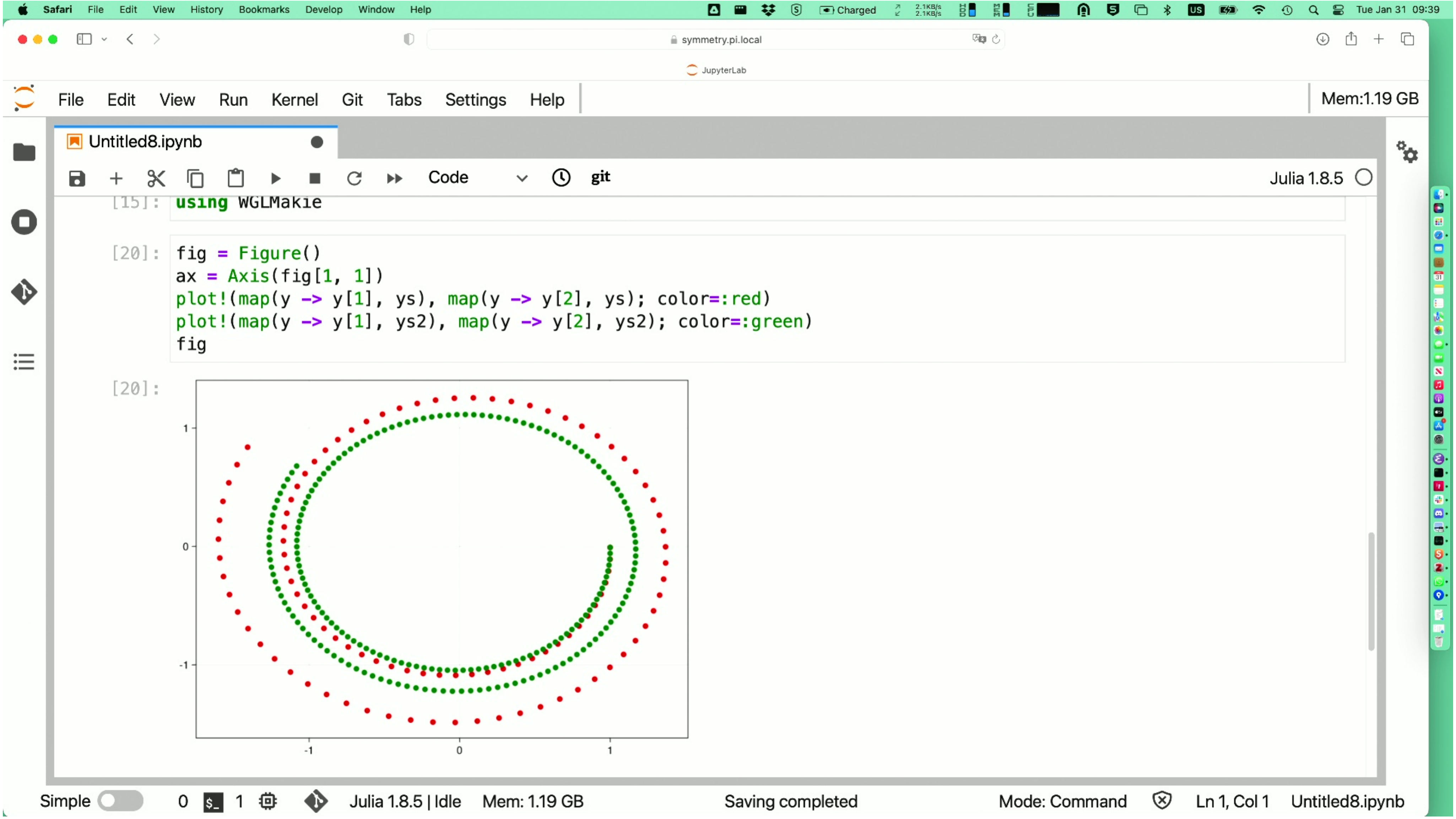
[14]:
```
ys = evolve(harmonic, y₀, 0.05, 200);
```

[15]:
```
using WGLMakie
```

[16]:
```
fig = Figure()
ax = Axis(fig[1, 1])
plot!(map(y -> y[1], ys), map(y -> y[2], ys))
fig
```

[16]:

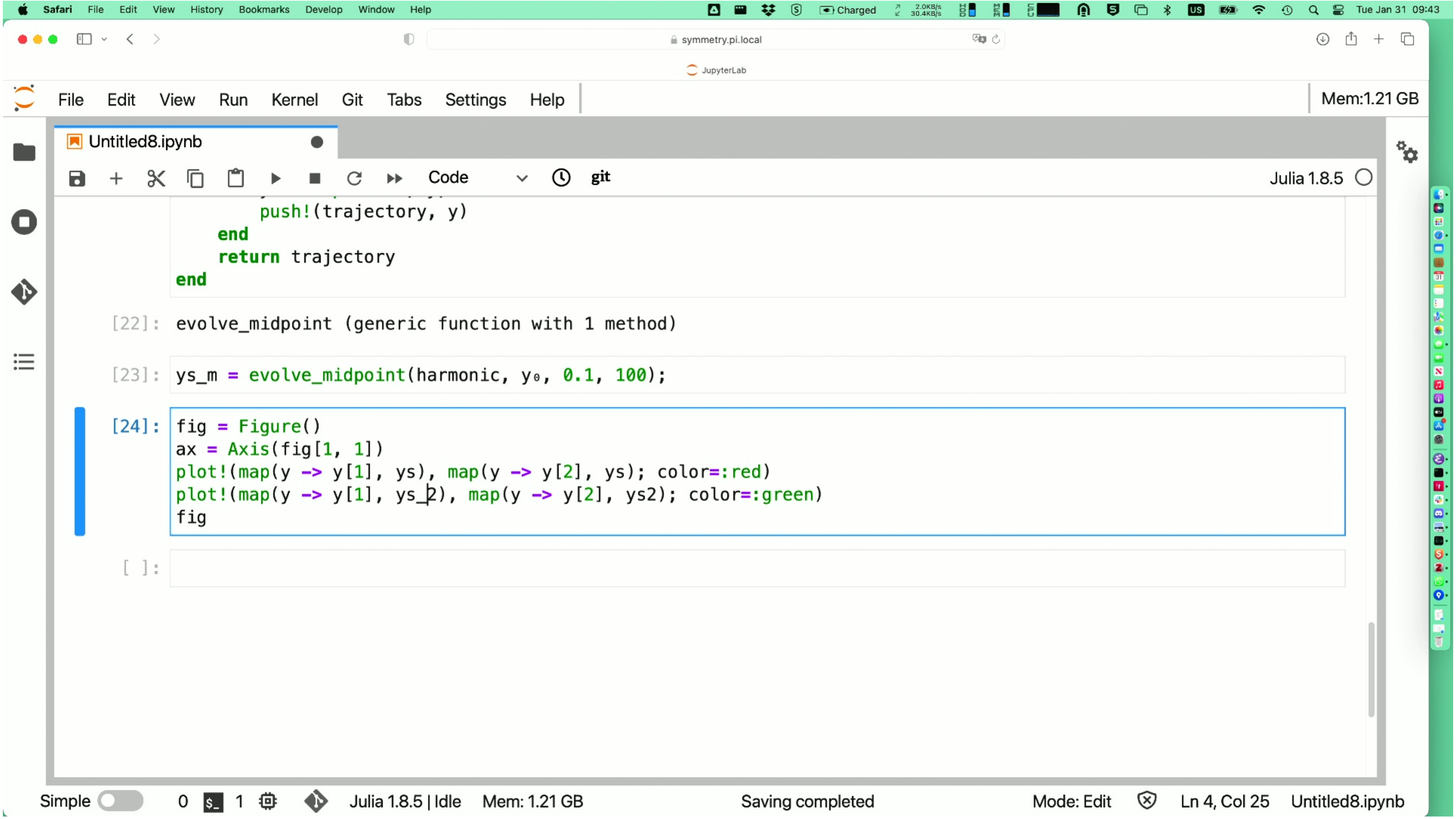
Simple 0 1 Julia 1.8.5 | Idle Mem: 1.06 GB
Saving completed
Mode: Edit Ln 1, Col 29 Untitled8.ipynb



```
[15]: using WGLMakie
```

```
[20]: fig = Figure()
      ax = Axis(fig[1, 1])
      plot!(map(y -> y[1], ys), map(y -> y[2], ys); color=:red)
      plot!(map(y -> y[1], ys2), map(y -> y[2], ys2); color=:green)
      fig
```

[20]:



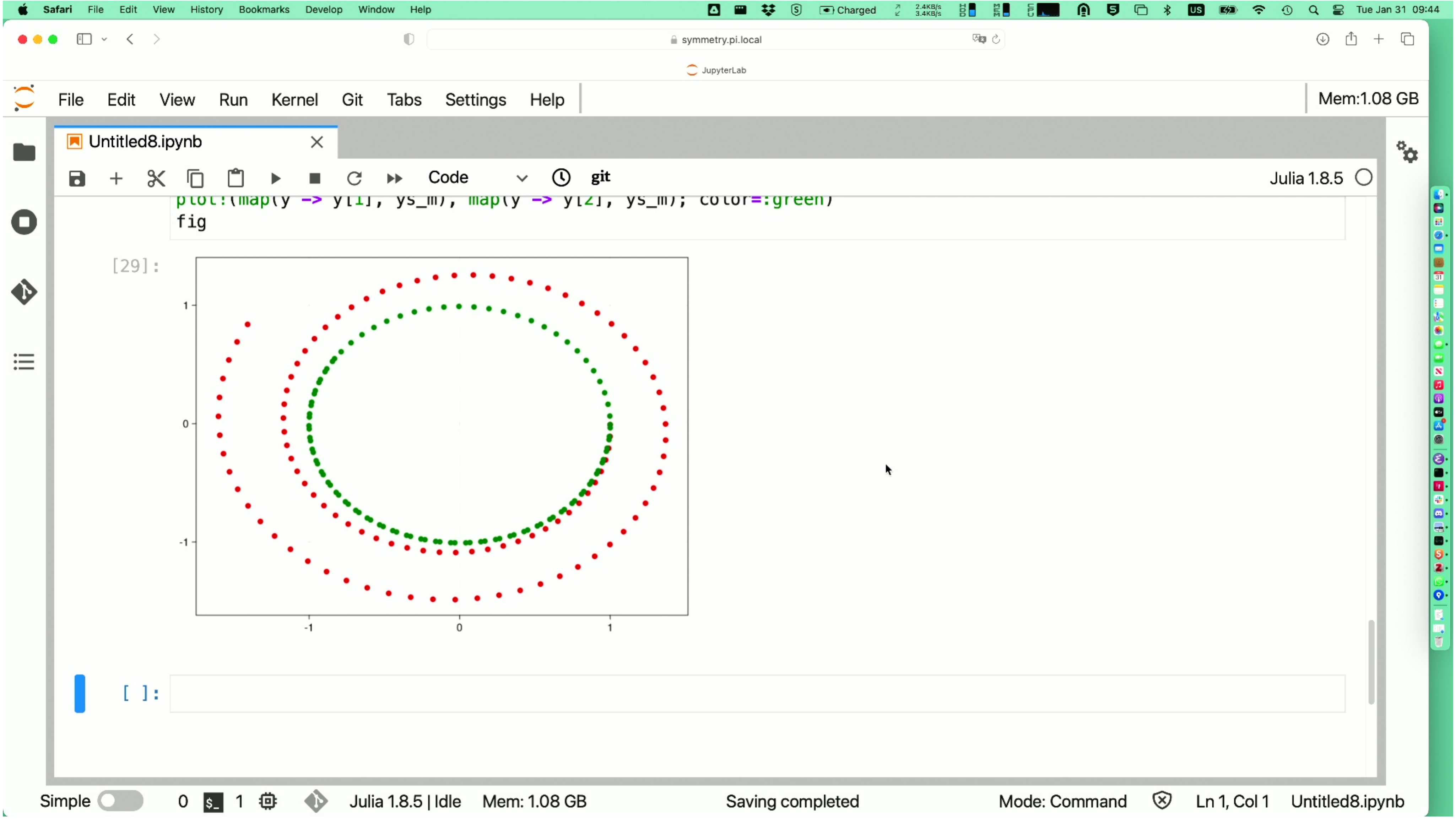
```
        push!(trajectory, y)
    end
    return trajectory
end
```

```
[22]: evolve_midpoint (generic function with 1 method)
```

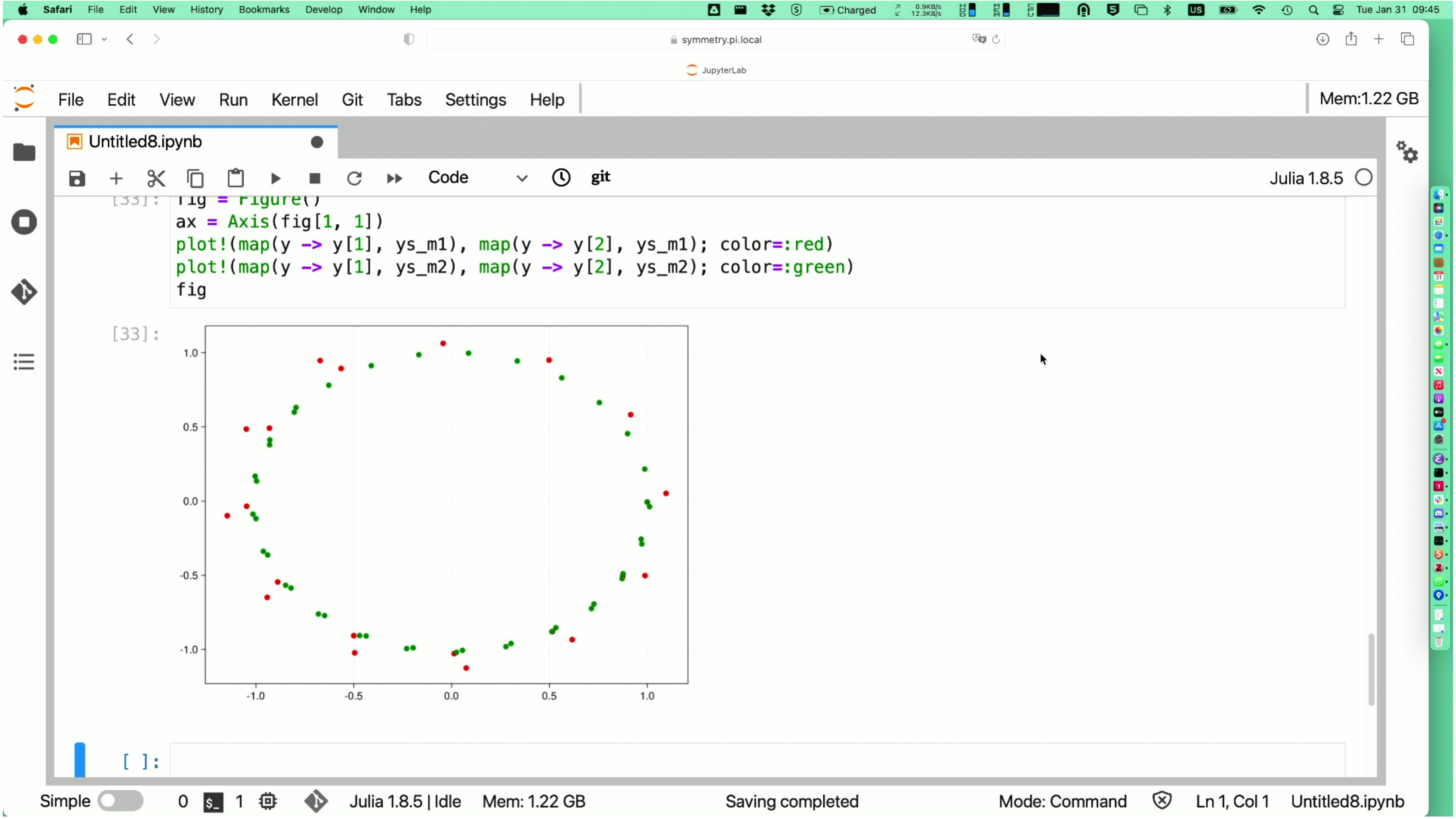
```
[23]: ys_m = evolve_midpoint(harmonic, y₀, 0.1, 100);
```

```
[24]: fig = Figure()
      ax = Axis(fig[1, 1])
      plot!(map(y -> y[1], ys), map(y -> y[2], ys); color=:red)
      plot!(map(y -> y[1], ys_2), map(y -> y[2], ys2); color=:green)
      fig
```

```
[ ]:
```



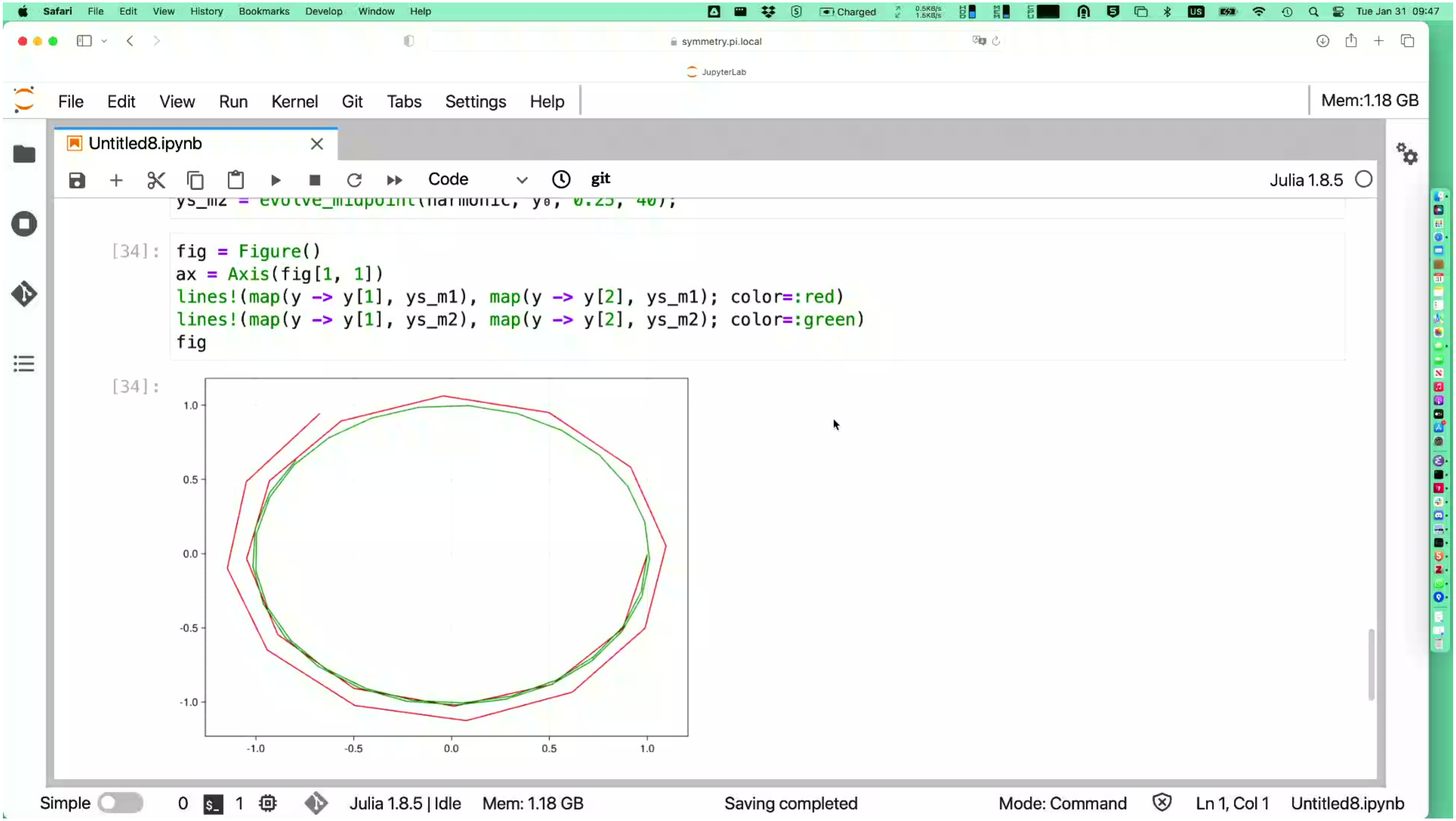
```
plot!(map(y -> y[1], ys_m), map(y -> y[2], ys_m); color=:green)
fig
```



```
[33]: fig = Figure()
      ax = Axis(fig[1, 1])
      plot!(map(y -> y[1], ys_m1), map(y -> y[2], ys_m1); color=:red)
      plot!(map(y -> y[1], ys_m2), map(y -> y[2], ys_m2); color=:green)
      fig
```

```
[33]:
```

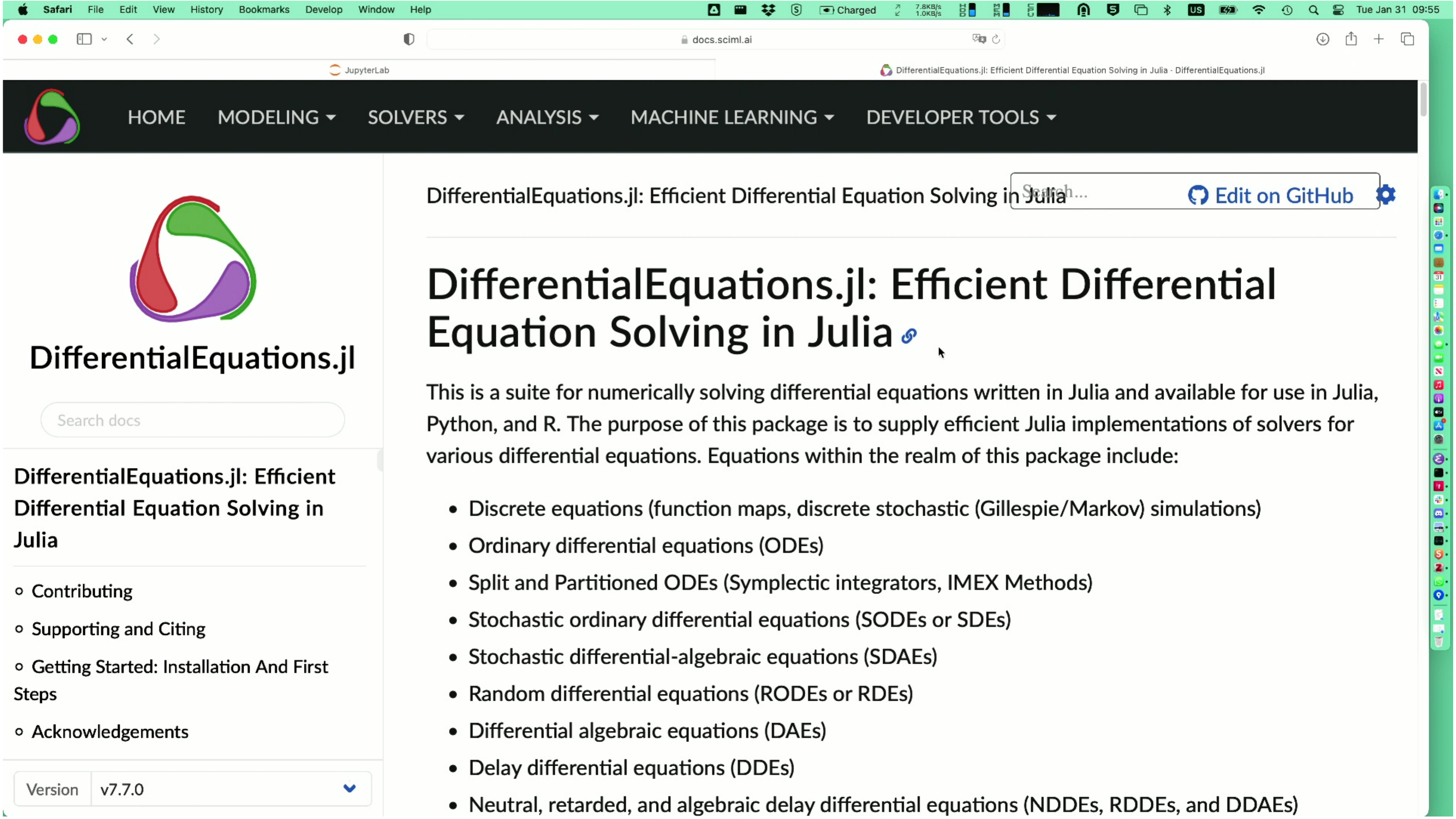
```
[ ]:
```



```
ys_m2 = evolve_midpoint(harmonic, y₀, 0.25, 40);
```

[34]:

```
fig = Figure()
ax = Axis(fig[1, 1])
lines!(map(y -> y[1], ys_m1), map(y -> y[2], ys_m1); color=:red)
lines!(map(y -> y[1], ys_m2), map(y -> y[2], ys_m2); color=:green)
fig
```



DifferentialEquations.jl: Efficient Differential Equation Solving in Julia

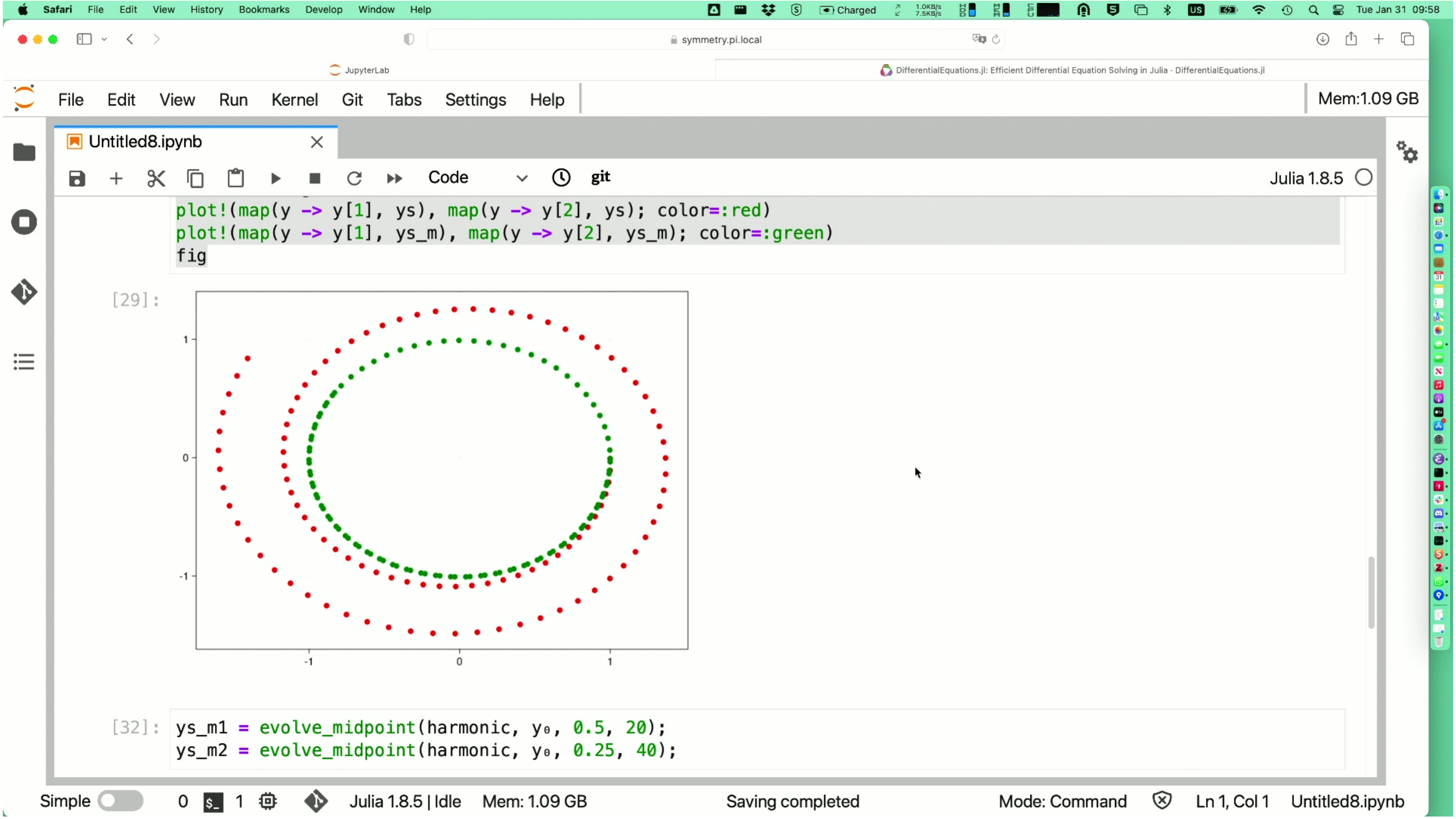
- Contributing
- Supporting and Citing
- Getting Started: Installation And First Steps
- Acknowledgements

DifferentialEquations.jl: Efficient Differential Equation Solving in Julia

DifferentialEquations.jl: Efficient Differential Equation Solving in Julia

This is a suite for numerically solving differential equations written in Julia and available for use in Julia, Python, and R. The purpose of this package is to supply efficient Julia implementations of solvers for various differential equations. Equations within the realm of this package include:

- Discrete equations (function maps, discrete stochastic (Gillespie/Markov) simulations)
- Ordinary differential equations (ODEs)
- Split and Partitioned ODEs (Symplectic integrators, IMEX Methods)
- Stochastic ordinary differential equations (SODEs or SDEs)
- Stochastic differential-algebraic equations (SDAEs)
- Random differential equations (RODEs or RDEs)
- Differential algebraic equations (DAEs)
- Delay differential equations (DDEs)
- Neutral, retarded, and algebraic delay differential equations (NDDEs, RDDEs, and DDAEs)



Safari File Edit View History Bookmarks Develop Window Help
Charged
Tue Jan 31 09:58
symmetry.pi.local
JupyterLab
DifferentialEquations.jl: Efficient Differential Equation Solving in Julia · DifferentialEquations.jl
File Edit View Run Kernel Git Tabs Settings Help
Mem:1.09 GB
Untitled8.ipynb
Code
git
Julia 1.8.5

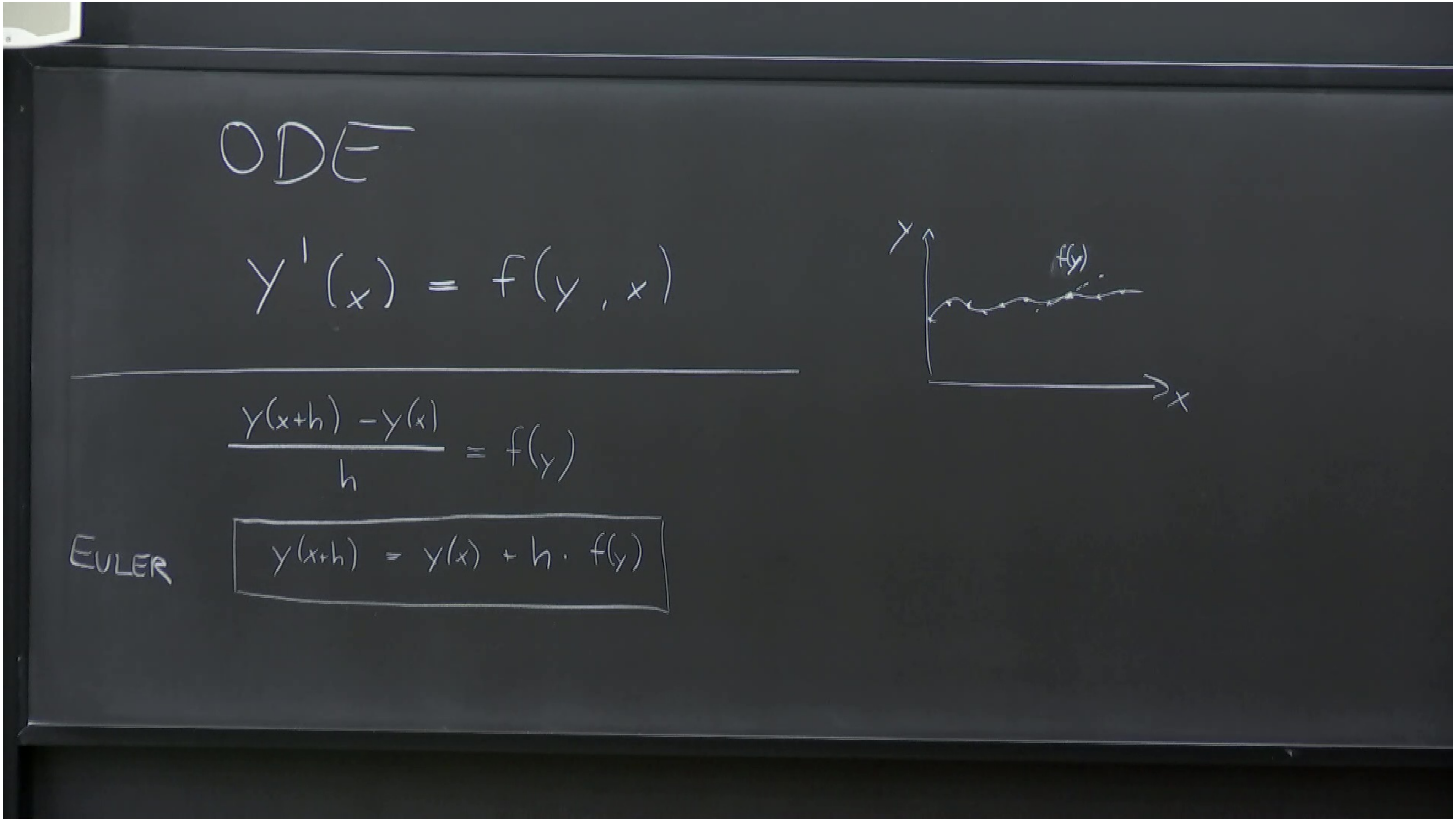
```
plot!(map(y -> y[1], ys), map(y -> y[2], ys); color=:red)
plot!(map(y -> y[1], ys_m), map(y -> y[2], ys_m); color=:green)
fig
```

[29]:

[32]:

```
ys_m1 = evolve_midpoint(harmonic, y₀, 0.5, 20);
ys_m2 = evolve_midpoint(harmonic, y₀, 0.25, 40);
```

Simple
0
1
Julia 1.8.5 | Idle
Mem: 1.09 GB
Saving completed
Mode: Command
Ln 1, Col 1
Untitled8.ipynb

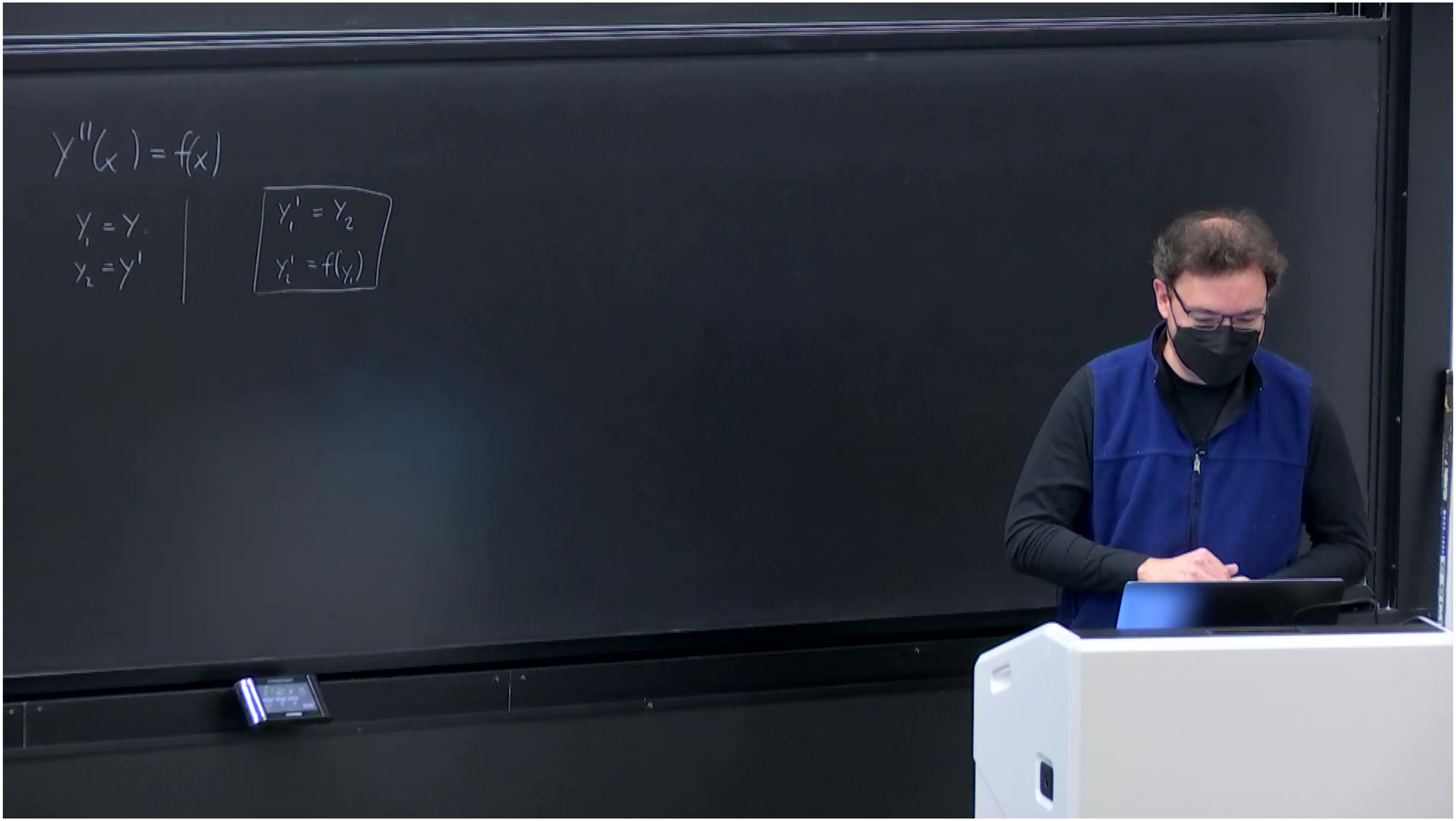


ODE

$$y'(x) = f(y, x)$$

$$\frac{y(x+h) - y(x)}{h} = f(y)$$

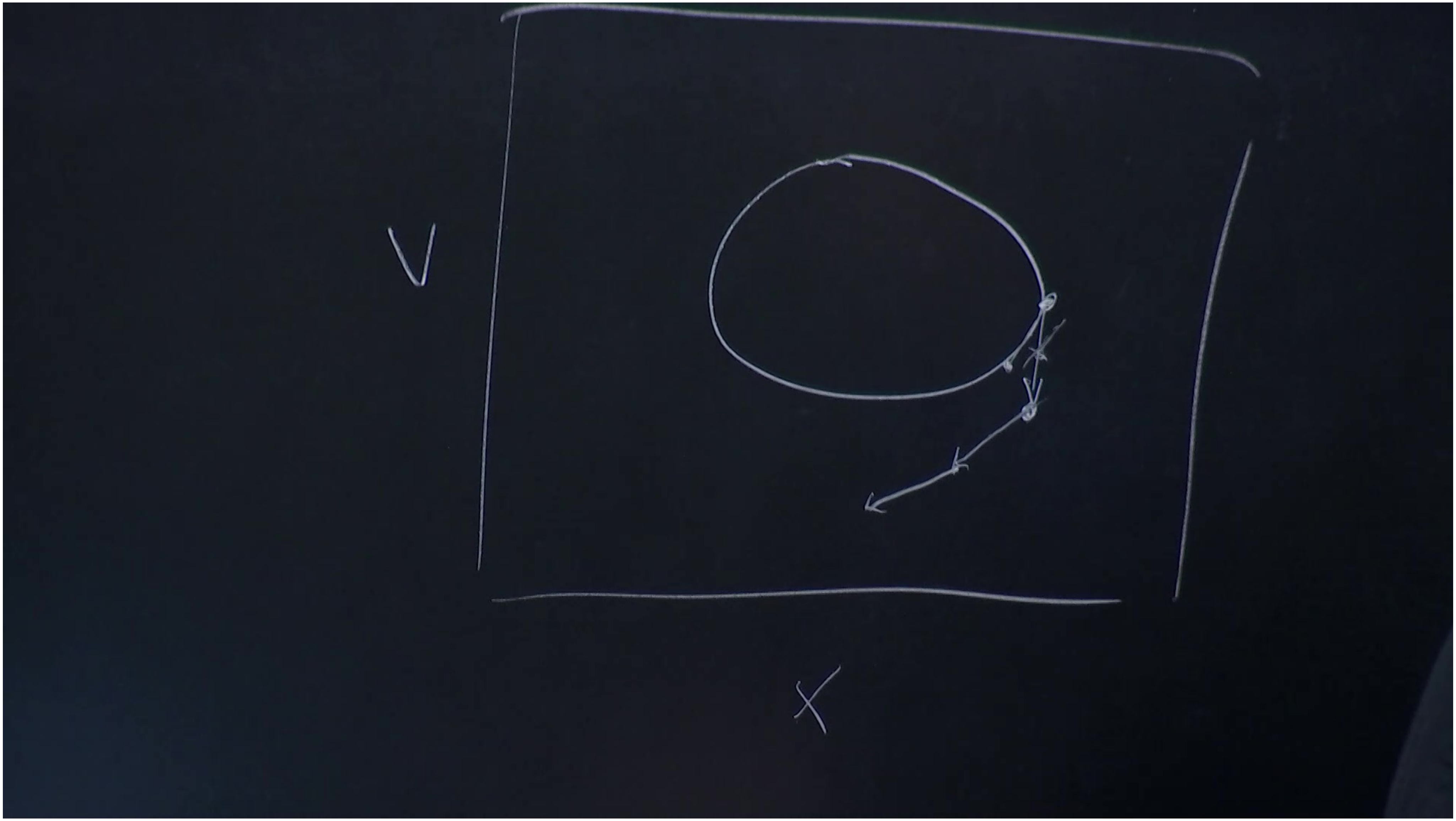
EULER $$\boxed{y(x+h) = y(x) + h \cdot f(y)}$$



$$y''(x) = f(x)$$

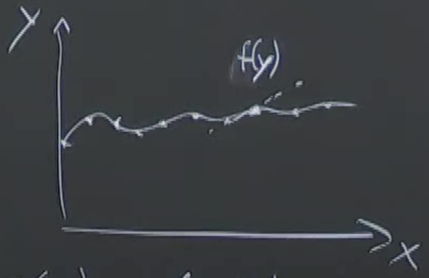
$$y_1 = y$$
$$y_2 = y'$$

$$\boxed{\begin{aligned} y_1' &= y_2 \\ y_2' &= f(y_1) \end{aligned}}$$


v
x

ODE

$$y'(x) = f(y,x)$$



$$\frac{y(x+h)-y(x)}{h} = f(y)$$

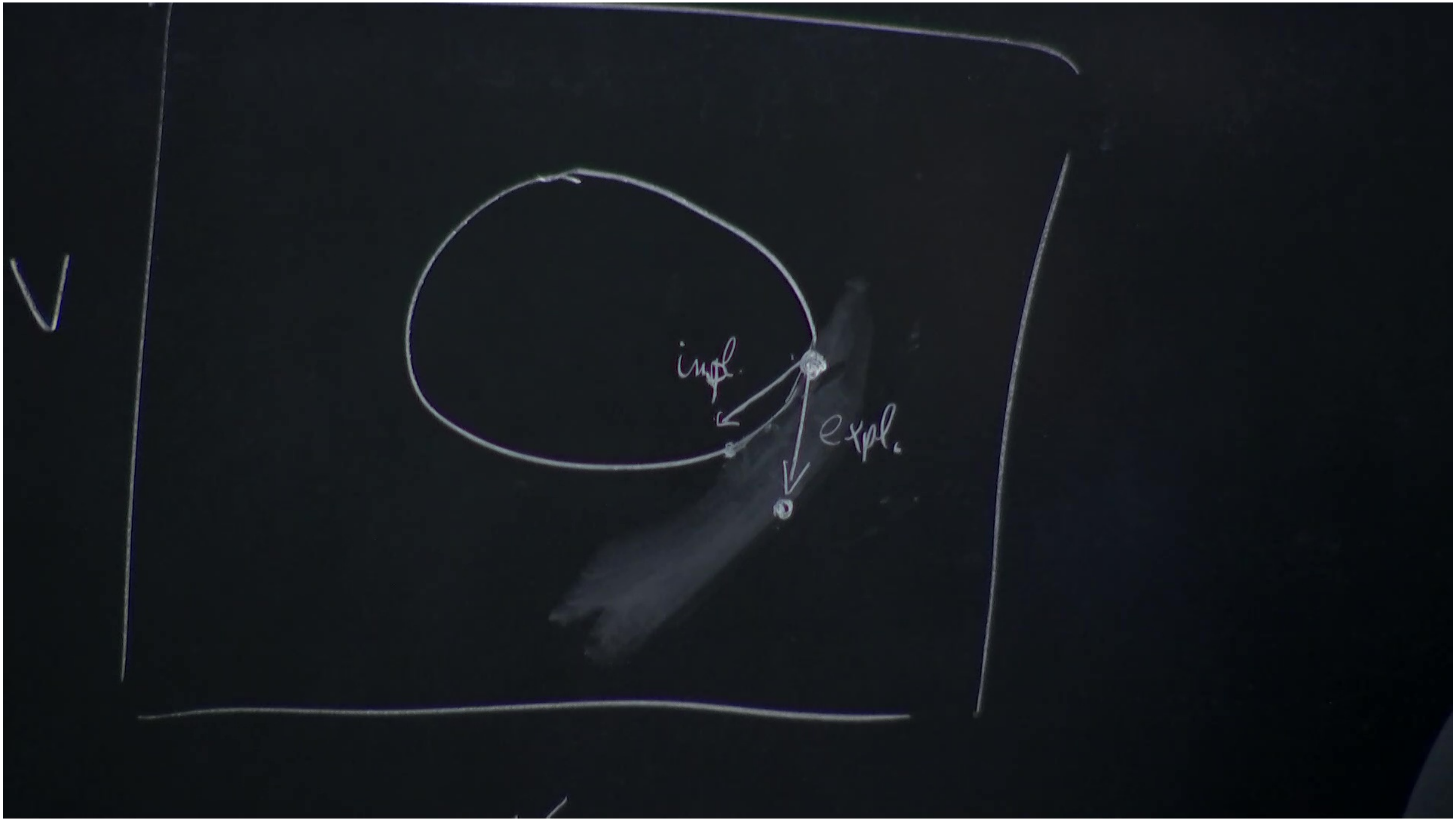
EULER $$\boxed{y(x+h) = y(x) + h\cdot f(y,x)}$$

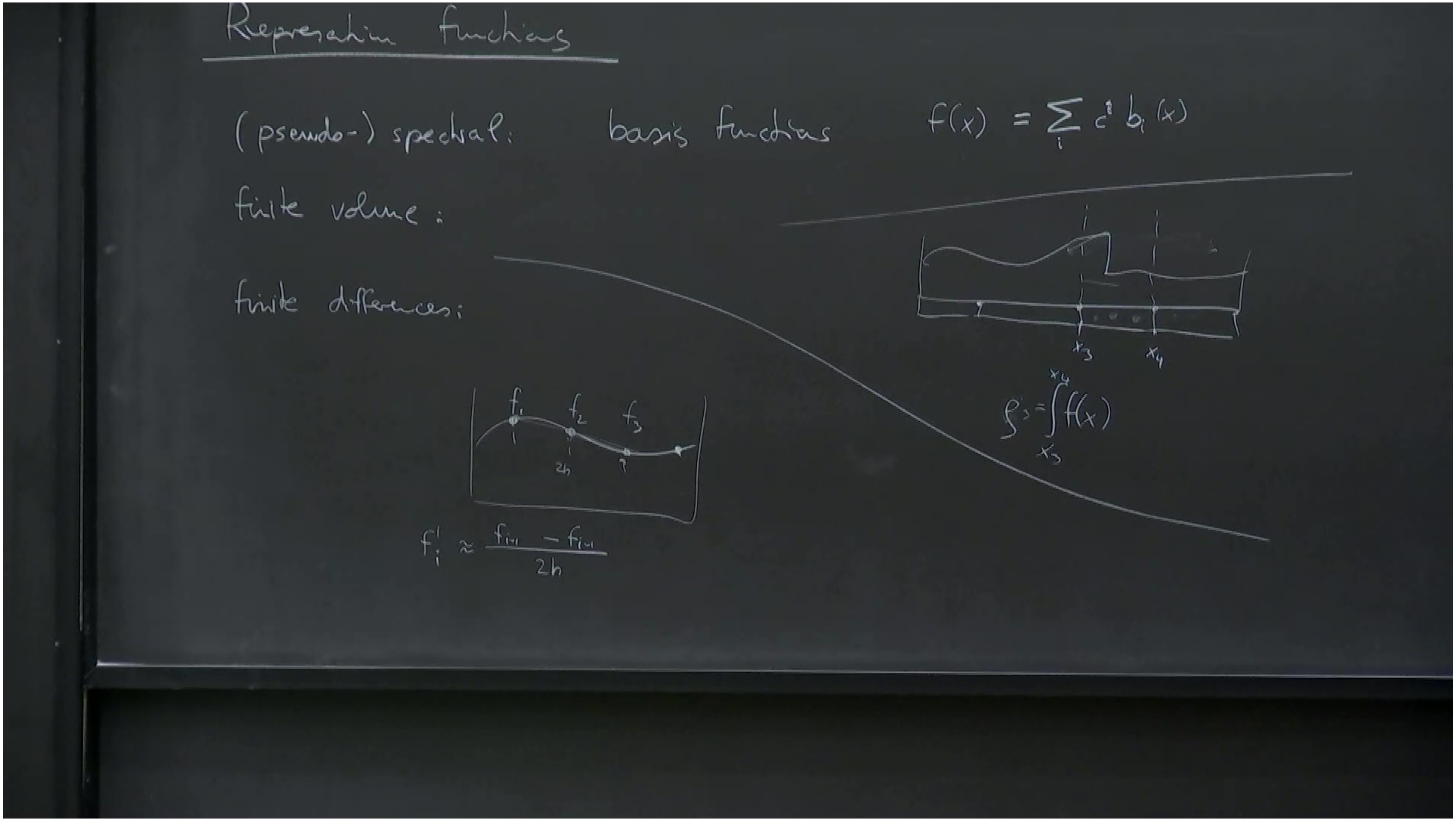
explicit

$$\frac{y(x)-y(x-h)}{h} = f(y,x)$$

$$\boxed{y(x+h) = y(x) + h\, f(y, x+h)}$$

implicit


V
impl.
expl.



Representation functions

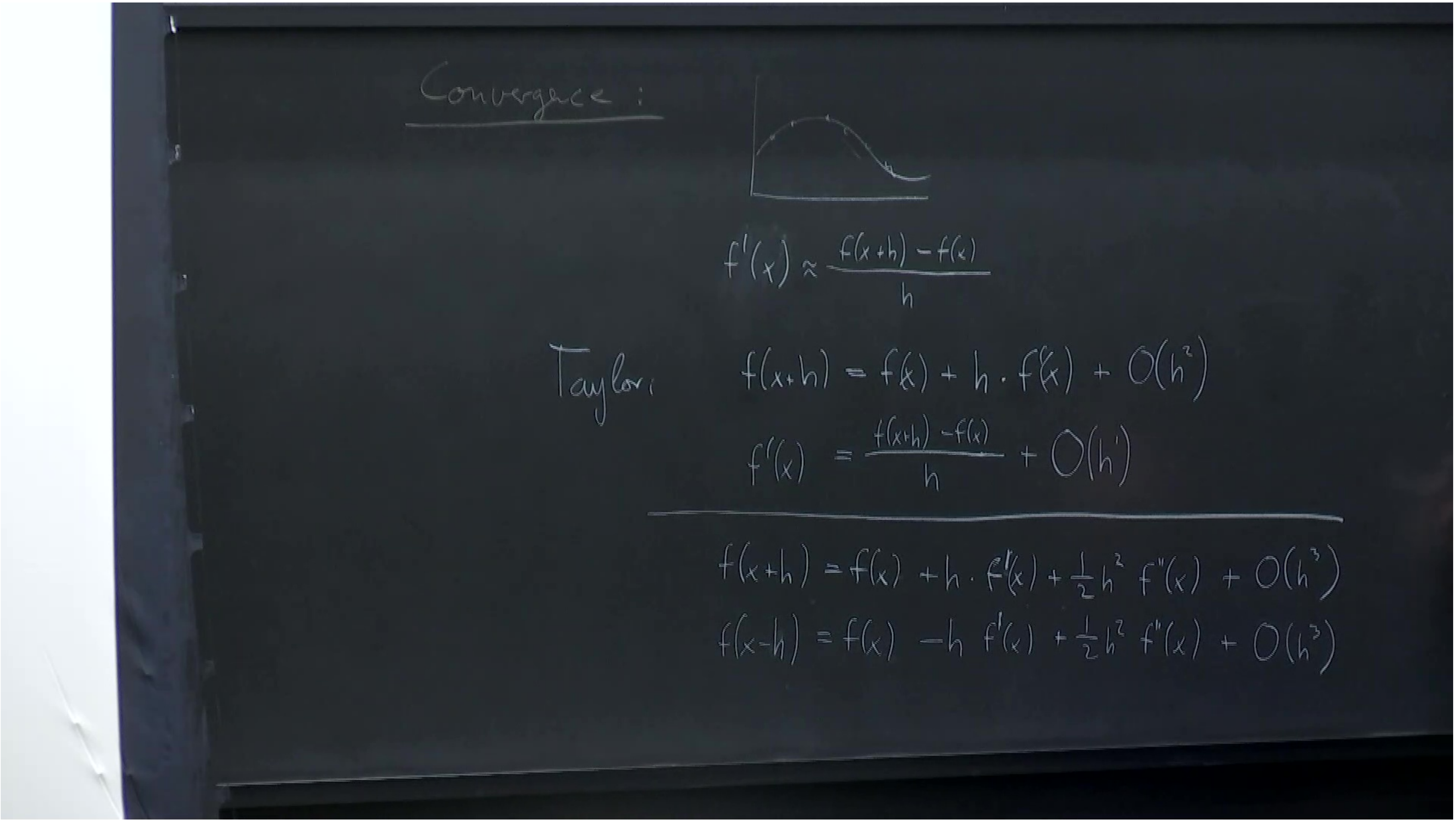
(pseudo-) spectral: basis functions $f(x) = \sum_i c^i b_i(x)$

finite volume:

finite differences:

$\rho_3 = \int_{x_3}^{x_4} f(x)$

$f'_i \approx \frac{f_{i+1} - f_{i-1}}{2h}$



Convergence :

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

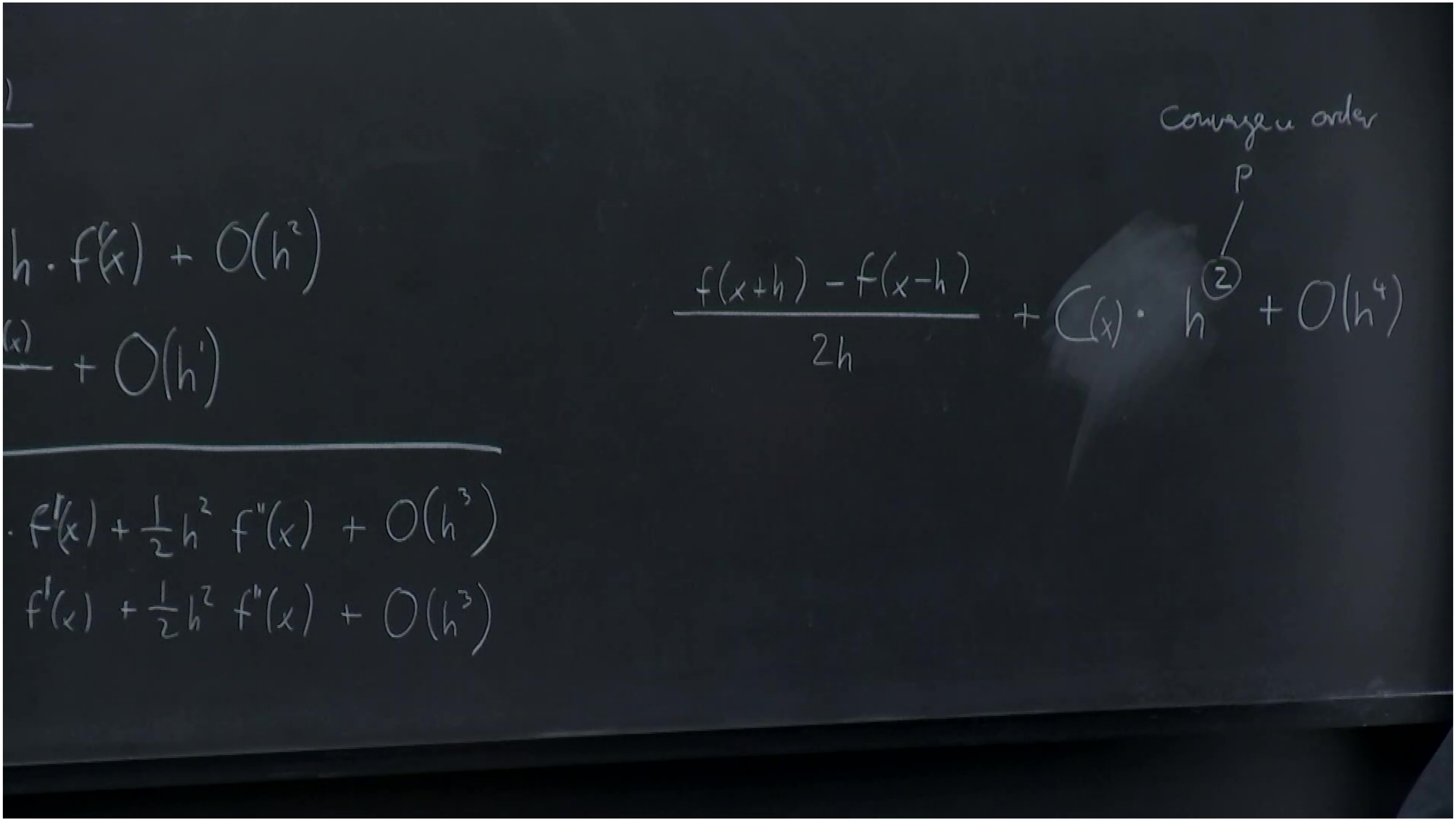
Taylor:

$$f(x+h) = f(x) + h \cdot f'(x) + O(h^2)$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h^1)$$

$$f(x+h) = f(x) + h \cdot f'(x) + \frac{1}{2} h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - h f'(x) + \frac{1}{2} h^2 f''(x) + O(h^3)$$



$)$

$h \cdot f'(x) + O(h^2)$

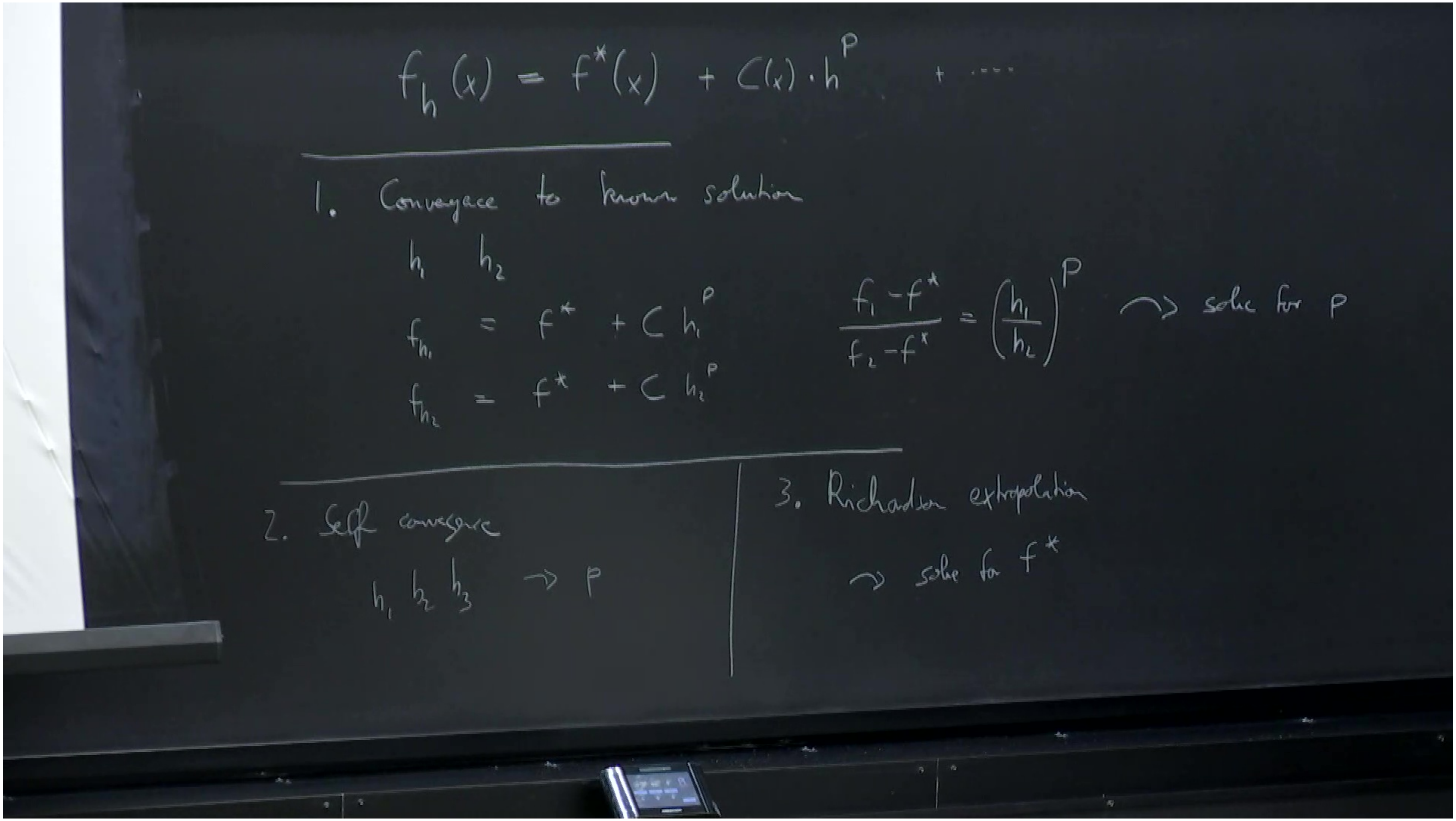
$\frac{(x)}{} + O(h^1)$

$\cdot f'(x) + \frac{1}{2}h^2 f''(x) + O(h^3)$

$f'(x) + \frac{1}{2}h^2 f''(x) + O(h^3)$

Convergence order

$$\frac{f(x+h)-f(x-h)}{2h} + C(x) \cdot h^{\overset{p}{\textcircled{2}}} + O(h^4)$$



$$f_h(x) = f^*(x) + C(x)\cdot h^p + \cdots$$

1. Convergace to known solution

$h_1 \quad h_2$

$f_{h_1} = f^* + C h_1^p$

$f_{h_2} = f^* + C h_2^p$

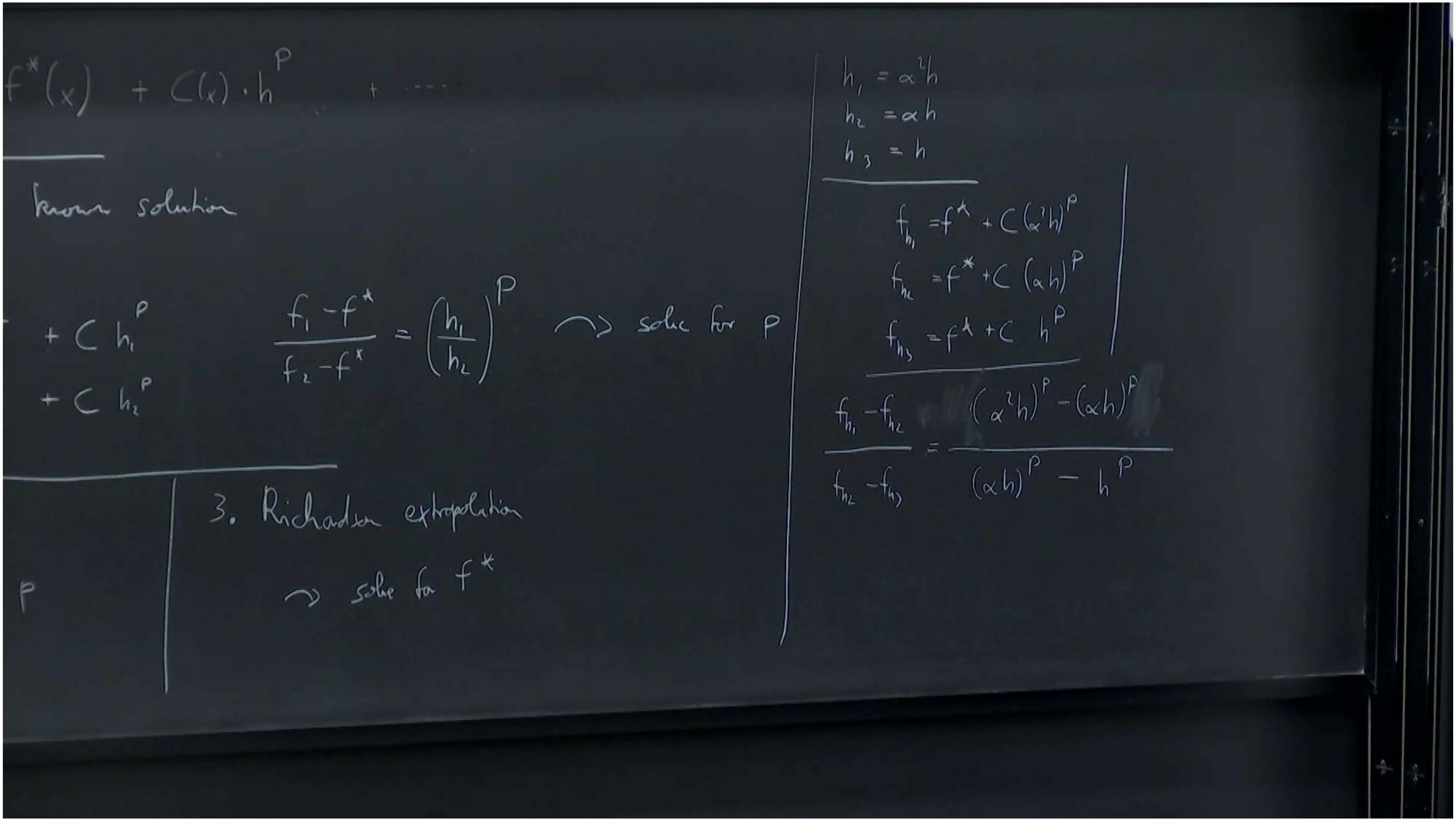
$$\frac{f_1 - f^*}{f_2 - f^*} = \left(\frac{h_1}{h_2}\right)^p \rightsquigarrow \text{solve for } p$$

2. Self convergence

$h_1\ h_2\ h_3 \rightsquigarrow p$

3. Richardson extrapolation

$\rightsquigarrow$ solve for f^*



$f^*(x) \ + C(x) \cdot h^p \quad + \ \cdots$

known solution

$+ C\, h_1^p$

$+ C\, h_2^p$

$$\frac{f_1 - f^*}{f_2 - f^*} = \left(\frac{h_1}{h_2}\right)^p \quad \rightsquigarrow \text{ solve for } p$$

p

3. Richardson extrapolation

$\rightsquigarrow$ solve for f^*

$h_1 = \alpha^2 h$

$h_2 = \alpha h$

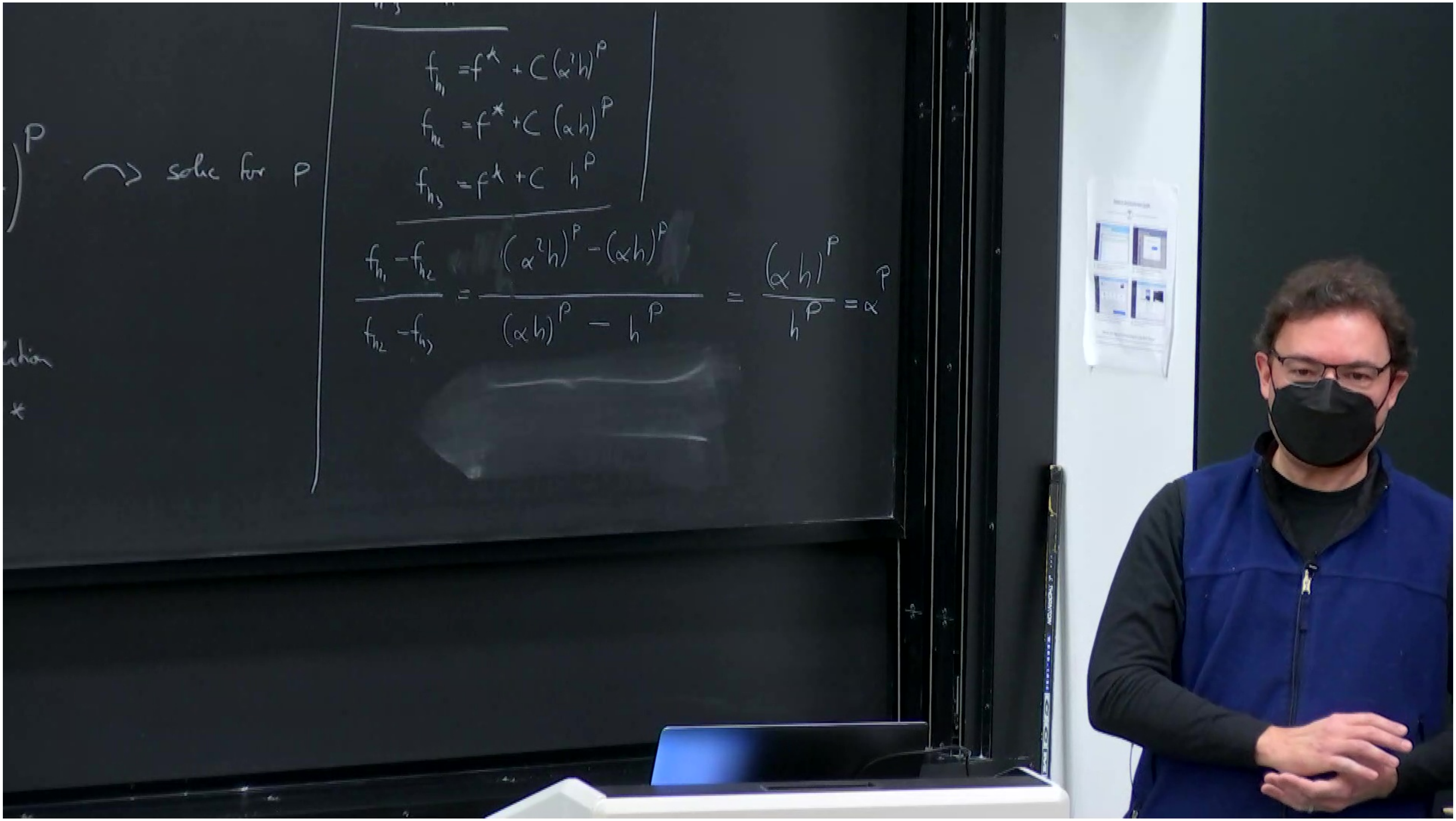
$h_3 = h$

$f_{h_1} = f^* + C(\alpha^2 h)^p$

$f_{h_2} = f^* + C\,(\alpha h)^p$

$f_{h_3} = f^* + C\ h^p$

$$\frac{f_{h_1} - f_{h_2}}{f_{h_2} - f_{h_3}} = \frac{(\alpha^2 h)^p - (\alpha h)^p}{(\alpha h)^p - h^p}$$



$)^p$ ⤳ solve for p

ation

$*$

$$f_{h_1} = f^* + C(\alpha^2 h)^p$$

$$f_{h_2} = f^* + C(\alpha h)^p$$

$$f_{h_3} = f^* + C\ h^p$$

$$\frac{f_{h_1} - f_{h_2}}{f_{h_2} - f_{h_3}} = \frac{(\alpha^2 h)^p - (\alpha h)^p}{(\alpha h)^p - h^p} = \frac{(\alpha h)^p}{h^p} = \alpha^p$$