Title: Lecture - Numerical Methods, PHYS 777

Speakers: Erik Schnetter, Dustin Lang

Collection/Series: Numerical Methods (Core), PHYS 777-, January 6 - February 5, 2025

Subject: Other

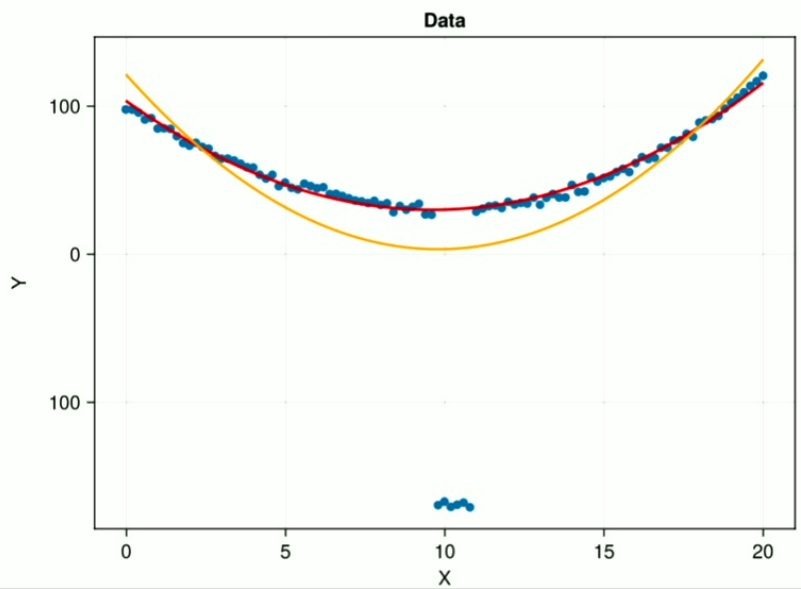
Date: January 16, 2025 - 10:15 AM

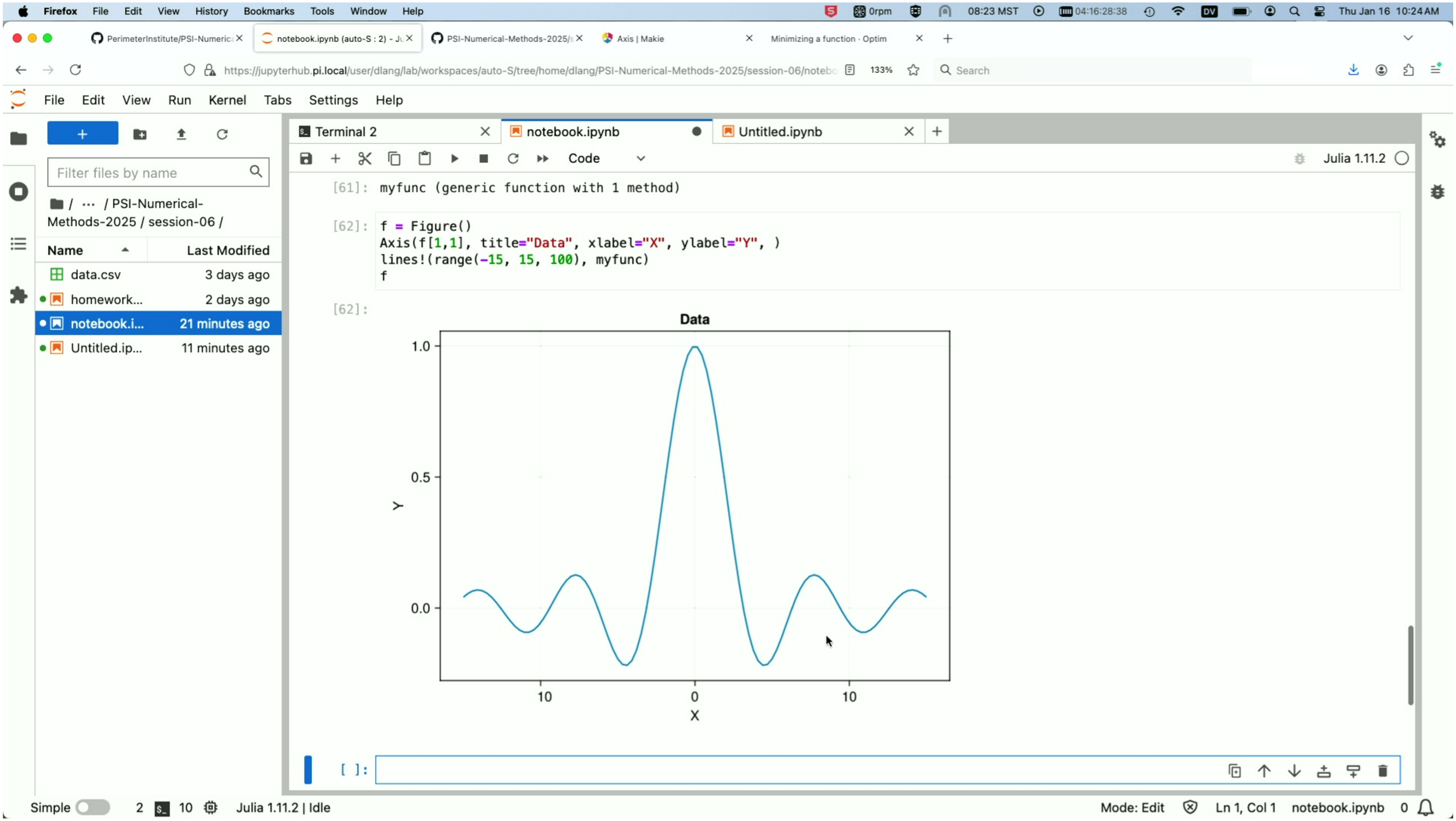
URL: https://pirsa.org/25010057

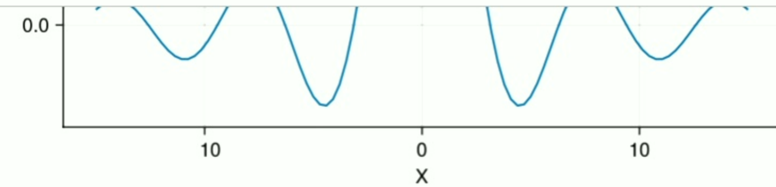
```
[58]: yy2 = @. c2[1] + c2[2] * xx + c2[3] * xx^2;
```

```
[60]: f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
scatter!(data2.x, data2.y)
lines!(xx, yy, color=:red, linewidth=2)
lines!(xx, yy2, color=:orange, linewidth=2)
#ylims!(0, 150)
f
```

[60]:




Firefox File Edit View History Bookmarks Tools Window Help
Thu Jan 16 10:24 AM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-S/tree/home/dlang/PSI-Numerical-Methods-2025/session-06/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 2
notebook.ipynb
Untitled.ipynb
Filter files by name
/ … / PSI-Numerical-Methods-2025 / session-06 /
Name
Last Modified
data.csv 3 days ago
homework... 2 days ago
notebook.i... 21 minutes ago
Untitled.ip... 11 minutes ago
Code
Julia 1.11.2
[61]: myfunc (generic function with 1 method)
[62]: f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y",)
lines!(range(-15, 15, 100), myfunc)
f
[62]:
Data
1.0
0.5
0.0
Y
10
0
10
X
[ ]:
Simple
Julia 1.11.2 | Idle
Mode: Edit
Ln 1, Col 1
notebook.ipynb



[63]:
```
using Optim
```

[ ]:
```
function mywrapper(x)
    # assum
```

[64]:
```
opt = optimize(myfunc, 1.)
```

```
MethodError: no method matching optimize(::typeof(myfunc), ::Float64)
The function `optimize` exists, but no method is defined for this combination of argument types.

Closest candidates are:
  optimize(::Any, ::T, ::T, ::Brent; rel_tol, abs_tol, iterations, time_limit, store_trace, show_trace, show_warnings, callback, show_every, extended_trace) where T<:AbstractFloat
   @ Optim ~/.julia/packages/Optim/fBdaz/src/univariate/solvers/brent.jl:23
  optimize(::Any, ::T, ::T, ::GoldenSection; rel_tol, abs_tol, iterations, time_limit, store_trace, show_trace, show_warnings, callback, show_every, extended_trace, nargs...) where T<:AbstractFloat
   @ Optim ~/.julia/packages/Optim/fBdaz/src/univariate/solvers/golden_section.jl:21
  optimize(::Any, ::Real, ::Real, ::Union{Brent, GoldenSection}; kwargs...)
   @ Optim ~/.julia/packages/Optim/fBdaz/src/univariate/optimize/interface.jl:46
  ...


Stacktrace:
 [1] top-level scope
   @ In[64]:1
```

```
[63]: using Optim
```

```
[65]: function mywrapper(x)
          # assume that "x" is an vector of length 1
          r = myfunc(x[1])
          return r
      end
```

```
[65]: mywrapper (generic function with 1 method)
```

```
[66]: opt = optimize(mywrapper, [1.])
```

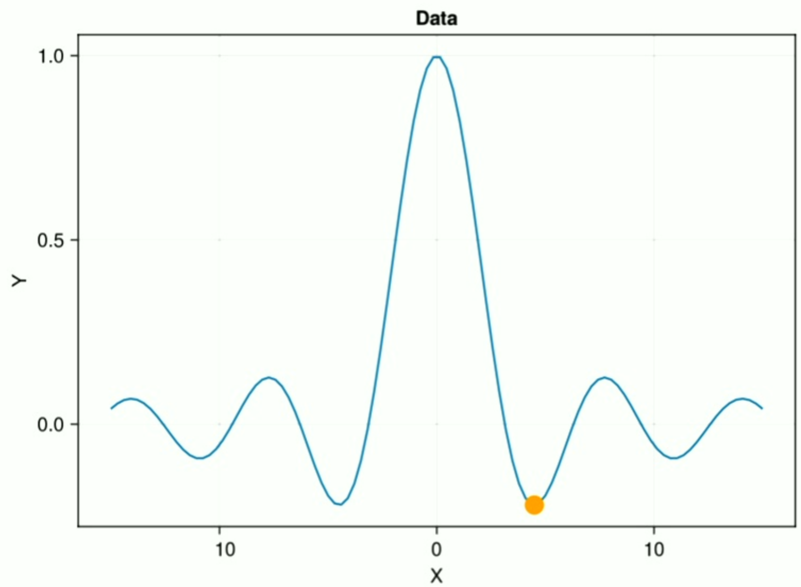
```
[66]:  * Status: success

       * Candidate solution
          Final objective value:     -2.172336e-01

       * Found with
          Algorithm:     Nelder-Mead

       * Convergence measures
          √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

       * Work counters
          Seconds run:   0  (vs limit Inf)
          Iterations:    11
          f(x) calls:    25
```

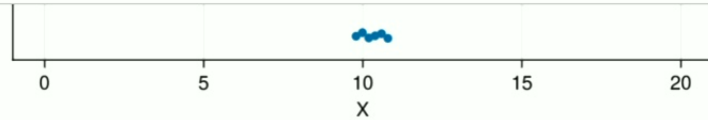
```
[ ]:
```

[69]: 4.493505859374998

[70]:
```julia
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
lines!(range(-15, 15, 100), myfunc)
scatter!(x_min, myfunc(x_min), color=:orange, markersize=20)
f
```

[70]:





[77]:
```julia
function myfunc(x)
    if x == 0
        return 1
    end
    return sin.(x) ./ x
end
```

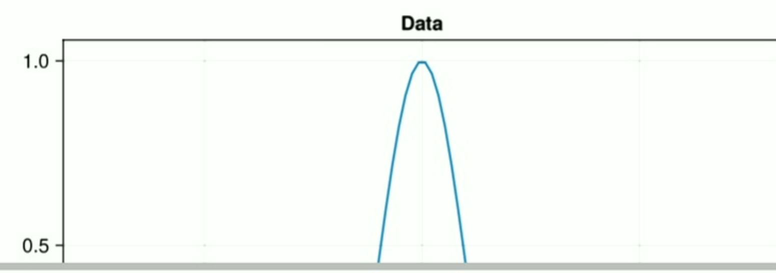
[77]: myfunc (generic function with 1 method)

[76]:
```julia
myfunc([1.])
```

[76]: 0.8414709848078965

[62]:
```julia
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
lines!(range(-15, 15, 100), myfunc)
f
```

[62]:



```
[63]: using Optim
```

```
[65]: function mywrapper(x)
          # assume that "x" is an vector of length 1
          r = myfunc(x[1])
          return r
      end
```

```
[65]: mywrapper (generic function with 1 method)
```

```
[71]: opt = optimize(myfunc, [1.])
```

```
MethodError: no method matching sin(::Vector{Float64})
The function `sin` exists, but no method is defined for this combination of argument types.

Closest candidates are:
  sin(::BigFloat)
   @ Base mpfr.jl:860
  sin(::IrrationalConstants.Twoπ)
   @ IrrationalConstants ~/.julia/packages/IrrationalConstants/vp5v4/src/trigonometric.jl:8
  sin(::Float16)
   @ Base math.jl:1511
  ...


Stacktrace:
 [1] myfunc(x::Vector{Float64})
   @ Main ./In[61]:5
 [2] value!!(obj::NonDifferentiable{Float64, Vector{Float64}}, x::Vector{Float64})
   @ NLSolversBase ~/.julia/packages/NLSolversBase/kavn7/src/interface.jl:9
 [3] initial_state(method::NelderMead{Optim.AffineSimplexer, Optim.AdaptiveParameters}, options::Optim.Options{Float64, Nothing},
d::NonDifferentiable{Float64, Vector{Float64}}, initial_x::Vector{Float64})
```

File Edit View Run Kernel Tabs Settings Help

Terminal 2 | notebook.ipynb | Untitled.ipynb

/ … / PSI-Numerical-Methods-2025 / session-06 /

Name	Last Modified
data.csv	3 days ago
homework…	2 days ago
notebook.i…	21 minutes ago
Untitled.ip…	11 minutes ago

Julia 1.11.2

[63]:
```
using Optim
```

[79]:
```
function mywrapper(x)
    # assume that "x" is an vector of length 1
    r = myfunc(x)
    return r[1]
end
```

[79]: mywrapper (generic function with 1 method)

[81]:
```
#opt = optimize(mywrapper, [1.])
```

[81]:
```
 * Status: success

 * Candidate solution
    Final objective value:     -2.172336e-01

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    11
    f(x) calls:    25
```

[69]:
```
x_min = Optim.minimizer(opt)[1]
```

[69]: 4.493505859374998

[70]:
```
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
lines!(range(-15, 15, 100), myfunc)
```

Simple 2 10 Julia 1.11.2 | Idle Mode: Edit Ln 1, Col 33 notebook.ipynb 0

[63]:
```julia
using Optim
```

[79]:
```julia
all_vals = []
function mywrapper(x)
    # assume that "x" is an vector of length 1
    r = myfunc(x)
    return r[1]
end
```

[79]:
```
mywrapper (generic function with 1 method)
```

[82]:
```julia
opt = optimize(mywrapper, [1.])

#opt = optimize(x -> myfunc(x)[1], [1.])
```

[82]:
```
 * Status: success

 * Candidate solution
    Final objective value:     -2.172336e-01

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    11
    f(x) calls:    25
```

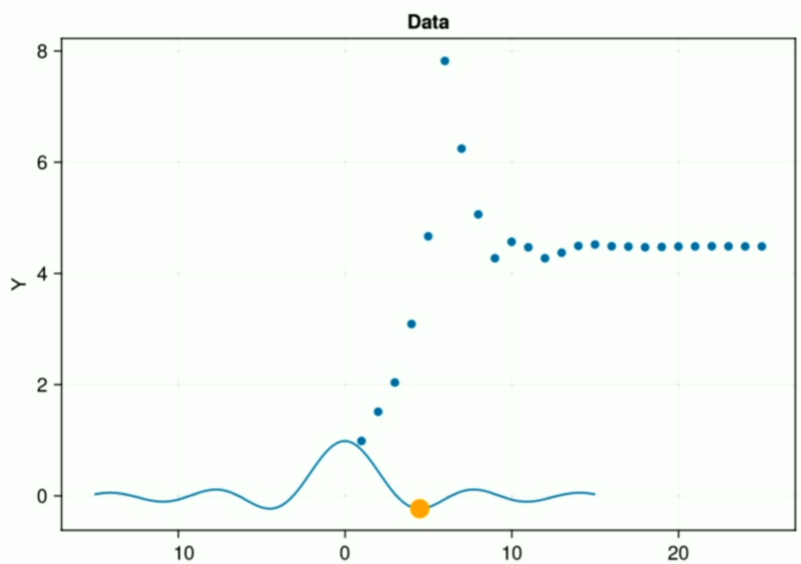
[69]:
```julia
x_min = Optim.minimizer(opt)[1]
```

[69]:
```
4.493505859374998
```

```
[69]: x_min = Optim.minimizer(opt)[1]

[69]: 4.493505859374998
```

```
[86]: f = Figure()
      Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
      lines!(range(-15, 15, 100), myfunc)
      scatter!(x_min, myfunc(x_min), color=:orange, markersize=20)
      scatter!(all_vals, m)
      f
```

[86]:



```
[90]: opt = optimize(mywrapper, [1.], LBFGS())

      #opt = optimize(x -> myfunc(x)[1], [1.])
```

```
[90]:  * Status: success

 * Candidate solution
    Final objective value:     -2.768973e-02

 * Found with
    Algorithm:     L-BFGS

 * Convergence measures
    |x - x'|               = 1.30e-03 ≰ 0.0e+00
    |x - x'|/|x'|          = 3.59e-05 ≰ 0.0e+00
    |f(x) - f(x')|         = 2.32e-08 ≰ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 8.40e-07 ≰ 0.0e+00
    |g(x)|                 = 9.52e-13 ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    4
    f(x) calls:    12
    ∇f(x) calls:   12
```

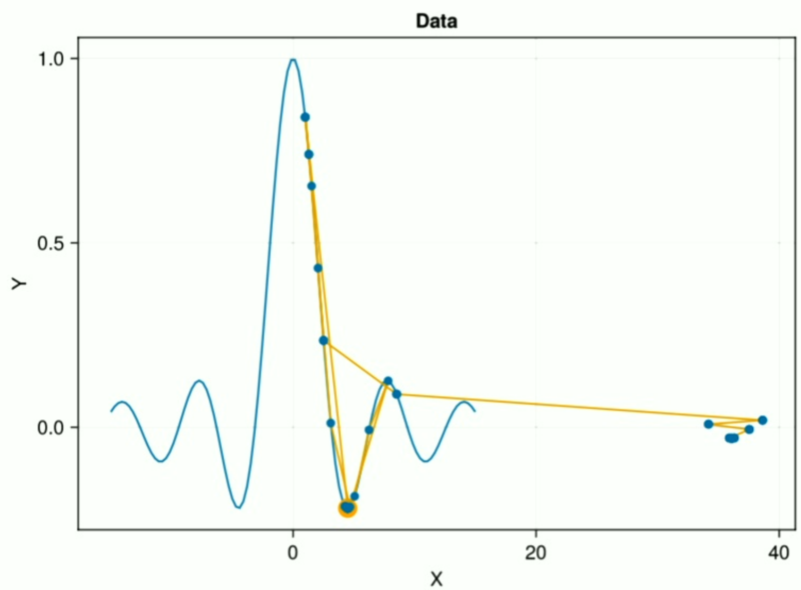
```
[85]: all_vals
```

```
[85]: 25-element Vector{Any}:
  1.0
  1.525
  2.05
  3.0999999999999996
  4.674999999999999
  7.8249999999999975
  6.249999999999998
  5.068749999999999
```

```
[69]: 4.493505859374998
```

```
[91]: f = Figure()
      Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
      lines!(range(-15, 15, 100), myfunc)
      scatter!(x_min, myfunc(x_min), color=:orange, markersize=20)
      lines!(all_vals, myfunc(all_vals))
      scatter!(all_vals, myfunc(all_vals))
      f
```

[91]:



Box Constrained Optimization

A primal interior-point algorithm for simple "box" constraints (lower and upper bounds) is available. Reusing our Rosenbrock example from above, boxed minimization is performed as follows:

```
lower = [1.25, -2.1]
upper = [Inf, Inf]
initial_x = [2.0, 2.0]
inner_optimizer = GradientDescent()
results = optimize(f, g!, lower, upper, initial_x, Fminbox(inner_optimizer))
```

This performs optimization with a barrier penalty, successively scaling down the barrier coefficient and using the chosen `inner_optimizer` (`GradientDescent()` above) for convergence at each step. To change algorithm specific options, such as the line search algorithm, specify it directly in the `inner_optimizer` constructor:

```
lower = [1.25, -2.1]
upper = [Inf, Inf]
initial_x = [2.0, 2.0]
# requires using LineSearches
inner_optimizer = GradientDescent(linesearch=LineSearches.BackTracking(order=3))
results = optimize(f, g!, lower, upper, initial_x, Fminbox(inner_optimizer))
```

This algorithm uses diagonal preconditioning to improve the accuracy, and hence is a good example of how to use `ConjugateGradient` or `LBFGS` with preconditioning. Other methods will currently not use preconditioning. Only the box constraints are used. If you can analytically compute the diagonal of the Hessian of your objective function, you may want to consider writing your own preconditioner.

There are two iterations parameters: an outer iterations parameter used to control `Fminbox` and an inner iterations parameter used to control the inner optimizer. For example, the following restricts the optimization to 2 major iterations

```
results = optimize(f, g!, lower, upper, initial_x, Fminbox(GradientDescent()), Optim.Options(ou
```

In contrast, the following sets the maximum number of iterations for each `GradientDescent()` optimization to 2

Firefox File Edit View History Bookmarks Tools Window Help 08:46 MST Thu Jan 16 10:46 AM

https://julianlsolvers.github.io/Optim.jl/stable/user/minimization/

Optim

Search docs (Ctrl + /)

Version v1.10.0

iterations

```
results = optimize(f, g!, lower, upper, initial_x, Fminbox(GradientDescent()), Optim.Options(ou
```

In contrast, the following sets the maximum number of iterations for each `GradientDescent()` optimization to 2

```
results = optimize(f, g!, lower, upper, initial_x, Fminbox(GradientDescent()), Optim.Options(it
```

Using second order information

When the Hessian of the objective function is available it is possible to use the primal-dual algorithm implemented in `IPNewton`. The interface is similar

```
results = optimize(f, lower, upper, initial_x, IPNewton())
results = optimize(f, g!, lower, upper, initial_x, IPNewton())
results = optimize(f, g!, h!, lower, upper, initial_x, IPNewton())
```

Minimizing a univariate function on a bounded interval

Minimization of univariate functions without derivatives is available through the `optimize` interface:

```
optimize(f, lower, upper, method; kwargs...)
```

Notice the lack of initial `x`. A specific example is the following quadratic function.

```
julia> f_univariate(x) = 2x^2+3x+1
f_univariate (generic function with 1 method)

julia> optimize(f_univariate, -2.0, 1.0)
Results of Optimization Algorithm
 * Algorithm: Brent's Method
 * Search Interval: [-2.000000, 1.000000]
 * Minimizer: -7.500000e-01
 * Minimum: -1.250000e-01
```

https://julianlsolvers.github.io/Optim.jl/stable/user/minimization/#Minimizing-a-univariate-function-on-a-bounded-interval

```
[92]: all_vals = []
function mywrapper(x)
    # assume that "x" is an vector of length 1
    push!(all_vals, x[1])
    r = myfunc(x)
    return r[1]
end
```

[92]: mywrapper (generic function with 1 method)

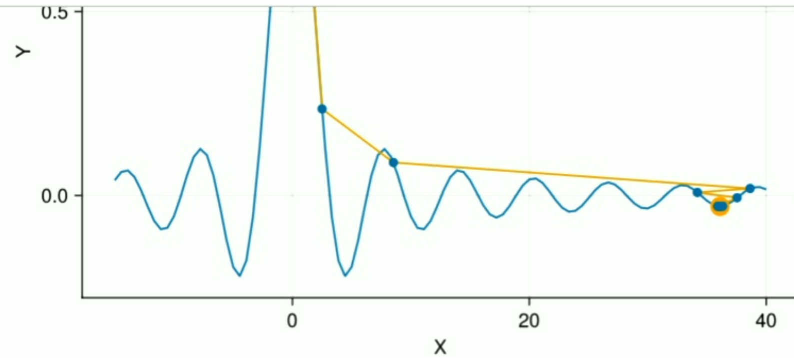
```
[104…: all_vals = []
opt = optimize(mywrapper, [0.], [5.], LBFGS())
#opt = optimize(myfunc, 0., 5.)#, LBFGS())

#opt = optimize(x -> myfunc(x)[1], [1.])
```

```
[104]: Results of Optimization Algorithm
 * Status: success
 * Algorithm: Brent's Method
 * Search Interval: [0.000000, 5.000000]
 * Minimizer: 4.493409e+00
 * Minimum: -2.172336e-01
 * Iterations: 14
 * Convergence: max(|x - x_upper|, |x - x_lower|) <= 2*(1.5e-08*|x|+2.2e-16): true
 * Objective Function Calls: 15
```

```
[105]: x_min = Optim.minimizer(opt)[1]
```

[105]: 4.493409457799146

```
[106]: f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
lines!(range(-15, 40, 100), myfunc)
scatter!(x_min, myfunc(x_min), color=:orange, markersize=20)
lines!(all_vals, myfunc(all_vals))
scatter!(all_vals, myfunc(all_vals))
f
```




```
[ ]: function objective(c)
         y_pred = ....
         residuals = y_pred - data2.y
         return sum(residuals^2)
     end
```

```
#opt = optimize(x -> myfunc(x)[1], [1.])
```

[107]:

```
 * Status: success

 * Candidate solution
    Final objective value:     -2.768973e-02

 * Found with
    Algorithm:     L-BFGS

 * Convergence measures
    |x - x'|               = 1.30e-03 ≰ 0.0e+00
    |x - x'|/|x'|          = 3.59e-05 ≰ 0.0e+00
    |f(x) - f(x')|         = 2.32e-08 ≰ 0.0e+00
    |f(x) - f(x')|/|f(x')| = 8.40e-07 ≰ 0.0e+00
    |g(x)|                 = 9.52e-13 ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    4
    f(x) calls:    12
    ∇f(x) calls:   12
```

[108]:

```
x_min = Optim.minimizer(opt)[1]
```

[108]: 36.10062224485131

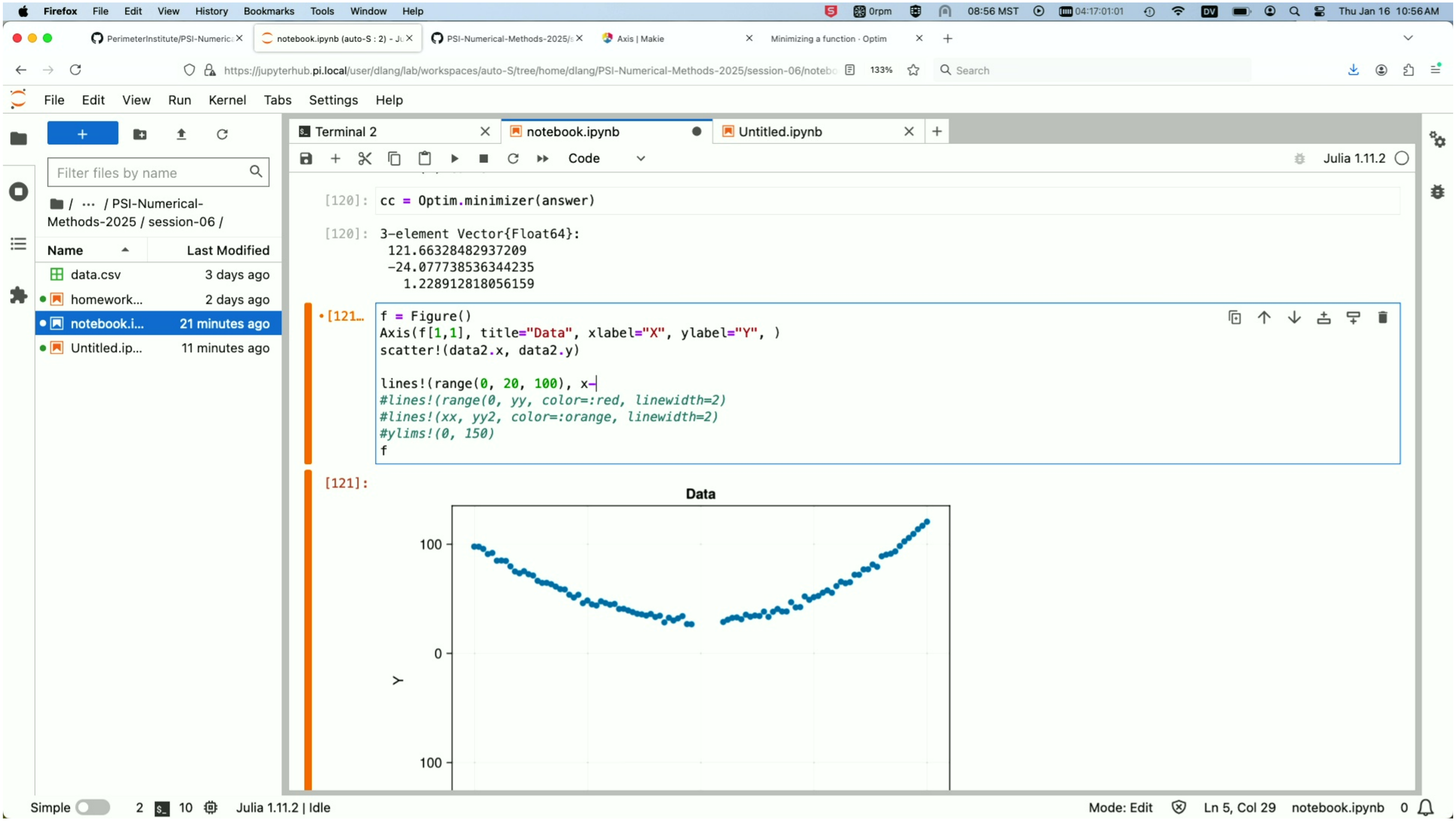
[109]:

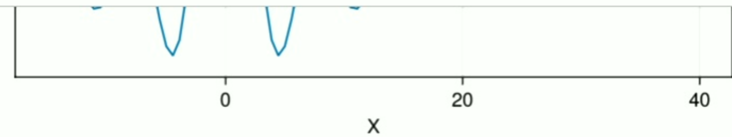
```
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
lines!(range(-15, 40, 100), myfunc)
scatter!(x_min, myfunc(x_min), color=:orange, markersize=20)
lines!(all_vals, myfunc(all_vals))
scatter!(all_vals, myfunc(all_vals))
f
```

[109]:

Data

```
[113]: function objective(c)
           # c is a vector of polynomial terms; assume quadratic
           y_pred = @. c[1] + c[2] * data2.x + c[3] * data2.x ^ 2
           residuals = y_pred .- data2.y
           return sum(residuals.^2)
       end

[113]: objective (generic function with 1 method)

[116]: objective([50., 0., 0.])

[116]: 351120.20491319767

[119]: answer = optimize(objective, [0., 0., 0.])

[119]:  * Status: success

        * Candidate solution
           Final objective value:     2.072481e+05

        * Found with
           Algorithm:     Nelder-Mead

        * Convergence measures
           √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

        * Work counters
           Seconds run:   0  (vs limit Inf)
           Iterations:    226
           f(x) calls:    422

[120]: cc = Optim.minimizer(answer)

[120]: 3-element Vector{Float64}:
        121.66328482937209
```


Firefox File Edit View History Bookmarks Tools Window Help
Thu Jan 16 10:56 AM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-S/tree/home/dlang/PSI-Numerical-Methods-2025/session-06/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 2
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ··· / PSI-Numerical-Methods-2025 / session-06 /
Name
Last Modified
data.csv 3 days ago
homework... 2 days ago
notebook.i... 21 minutes ago
Untitled.ip... 11 minutes ago
[120]: cc = Optim.minimizer(answer)
[120]: 3-element Vector{Float64}:
121.66328482937209
-24.077738536344235
1.228912818056159
[121…
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y",)
scatter!(data2.x, data2.y)
lines!(range(0, 20, 100), x-
#lines!(range(0, yy, color=:red, linewidth=2)
#lines!(xx, yy2, color=:orange, linewidth=2)
#ylims!(0, 150)
f
[121]:
Data
100
0
Y
100
Simple
2
10
Julia 1.11.2 | Idle
Mode: Edit
Ln 5, Col 29
notebook.ipynb
0




```
[113]: function objective(c)
           # c is a vector of polynomial terms; assume quadratic
           y_pred = @. c[1] + c[2] * data2.x + c[3] * data2.x ^ 2
           residuals = y_pred .- data2.y
           return sum(residuals.^2)
       end
```

```
[113]: objective (generic function with 1 method)
```

```
[116]: objective([50., 0., 0.])
```

```
[116]: 351120.20491319767
```

```
[119]: answer = optimize(objective, [0., 0., 0.])
```

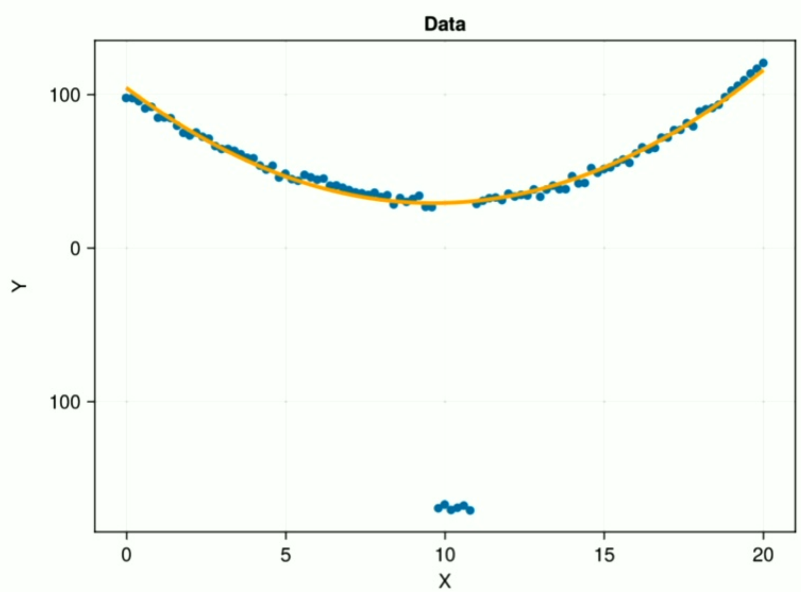
```
[119]:  * Status: success

        * Candidate solution
           Final objective value:     2.072481e+05

        * Found with
           Algorithm:     Nelder-Mead

        * Convergence measures
           √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

        * Work counters
           Seconds run:   0  (vs limit Inf)
           Iterations:    226
```

```
-15.567251576925173
  0.8071295925166654
```

[131]:

```
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
scatter!(data2.x, data2.y)

lines!(range(0, 20, 100), x -> cc[1] + cc[2]*x + cc[3]*x^2, color=:orange, linewidth=3)
f
```

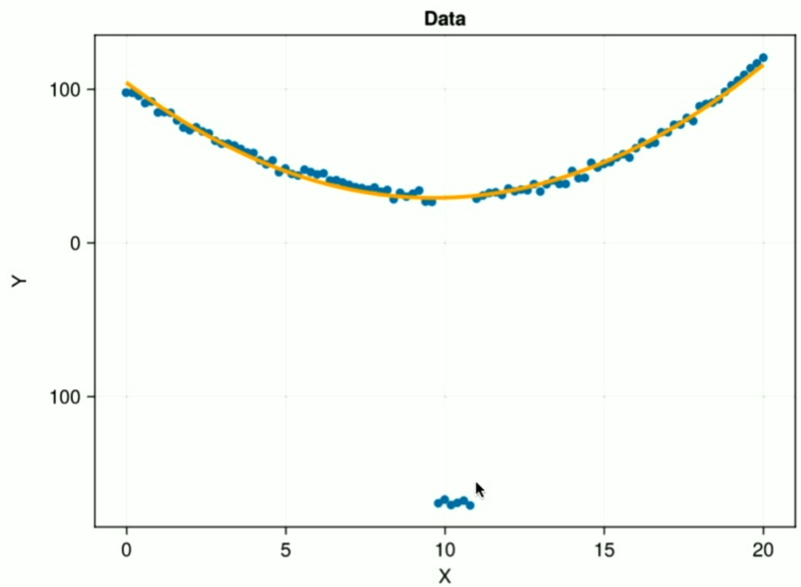
[131]:



```
[126]: function objective(c)
           # c is a vector of polynomial terms; assume quadratic
           y_pred = @. c[1] + c[2] * data2.x + c[3] * data2.x ^ 2
           residuals = y_pred .- data2.y
           #return sum(residuals.^2)
           return sum(abs.(residuals))
       end

[126]: objective (generic function with 1 method)

[127]: objective([50., 0., 0.])

[127]: 3254.911138994189

[128]: answer = optimize(objective, [0., 0., 0.])

[128]:  * Status: success

        * Candidate solution
           Final objective value:     1.414429e+03

        * Found with
           Algorithm:     Nelder-Mead

        * Convergence measures
           √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

        * Work counters
           Seconds run:   0  (vs limit Inf)
           Iterations:    263
           f(x) calls:    488

[129]: cc = Optim.minimizer(answer)

[129]: 3-element Vector{Float64}:
```

```
[131]: f = Figure()
       Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
       scatter!(data2.x, data2.y)

       lines!(range(0, 20, 100), x -> cc[1] + cc[2]*x + cc[3]*x^2, color=:orange, linewidth=3)
       f
```

[131]:



Firefox File Edit View History Bookmarks Tools Window Help

https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-S/tree/home/dlang/PSI-Numerical-Methods-2025/session-06/notebo

File Edit View Run Kernel Tabs Settings Help

Terminal 2 | notebook.ipynb | Untitled.ipynb

/ ··· / PSI-Numerical-Methods-2025 / session-06 /

Name	Last Modified
data.csv	3 days ago
homework...	2 days ago
notebook.i...	21 minutes ago
Untitled.ip...	11 minutes ago

[135]:
```
? min
```

```
search: min sin in x_min @main Pkg.pin Main Cint argmin bind sign asin sind sinh
```

[135]:
```
min(x, y, ...)
```

Return the minimum of the arguments, with respect to `isless`. If any of the arguments is `missing`, return `missing`. See also the `minimum` function to take the minimum element from a collection.

Examples

```
julia> min(2, 5, 1)
1

julia> min(4, missing, 6)
missing
```

```
min(x::BareInterval, y::BareInterval)
min(x::Interval, y::Interval)
```

Implement the `min` function of the IEEE Standard 1788-2015 (Table 9.1).

[137]:
```
min.([1., 7,], [2., 4.])
```

[137]:
```
2-element Vector{Float64}:
 1.0
 4.0
```

[126...
```
function objective(c)
    # c is a vector of polynomial terms; assume quadratic
    y_pred = @. c[1] + c[2] * data2.x + c[3] * data2.x ^ 2
    residuals = y_pred .- data2.y
    #return sum(residuals.^2)
    #return sum(abs.(residuals))
```

Simple 2 10 Julia 1.11.2 | Idle Mode: Command Ln 8, Col 36 notebook.ipynb 0

File Edit View Run Kernel Tabs Settings Help

Terminal 2 | notebook.ipynb | Untitled.ipynb

/ … / PSI-Numerical-Methods-2025 / session-06 /

Name	Last Modified
data.csv	3 days ago
homework...	2 days ago
notebook.i...	21 minutes ago
Untitled.ip...	11 minutes ago

```
[142]: answer = optimize(objective, [0., 0., 0.])

[142]:  * Status: success

 * Candidate solution
    Final objective value:     1.010000e+03

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    0
    f(x) calls:    5

[143]: cc_clipped = Optim.minimizer(answer)

[143]: 3-element Vector{Float64}:
 0.0
 0.0
 0.0

[129]: cc = Optim.minimizer(answer)

[129]: 3-element Vector{Float64}:
 104.6183144244942
 -15.567251576925173
   0.8071295925166654

[133]: typeof(answer)

[133]: Optim.MultivariateOptimizationResults{NelderMead{Optim.AffineSimplexer, Optim.AdaptiveParameters}, Vector{Float64}, Float64, Float64
, Vector{OptimizationState{Float64, NelderMead{Optim.AffineSimplexer, Optim.AdaptiveParameters}}}, Bool, @NamedTuple{f_limit_reache
d::Bool, g_limit_reached::Bool, h_limit_reached::Bool, time_limit::Bool, callback::Bool, f_increased::Bool}}
```

Simple 2 10 Julia 1.11.2 | Idle   Mode: Edit   Ln 1, Col 37   notebook.ipynb 0

```
 Seconds run:   0  (vs limit Inf)
 Iterations:    186
 f(x) calls:    346
```

[145]:
```
cc_clipped = Optim.minimizer(answer)
```

[145]:
```
3-element Vector{Float64}:
 103.9163841317835
 -15.340773952011103
   0.7968943076145193
```

[129]:
```
cc = Optim.minimizer(answer)
```

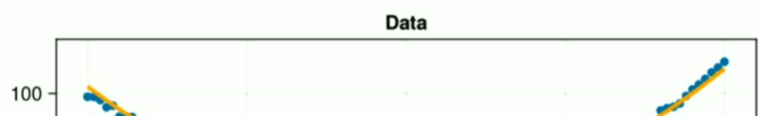
[129]:
```
3-element Vector{Float64}:
 104.6183144244942
 -15.567251576925173
   0.8071295925166654
```

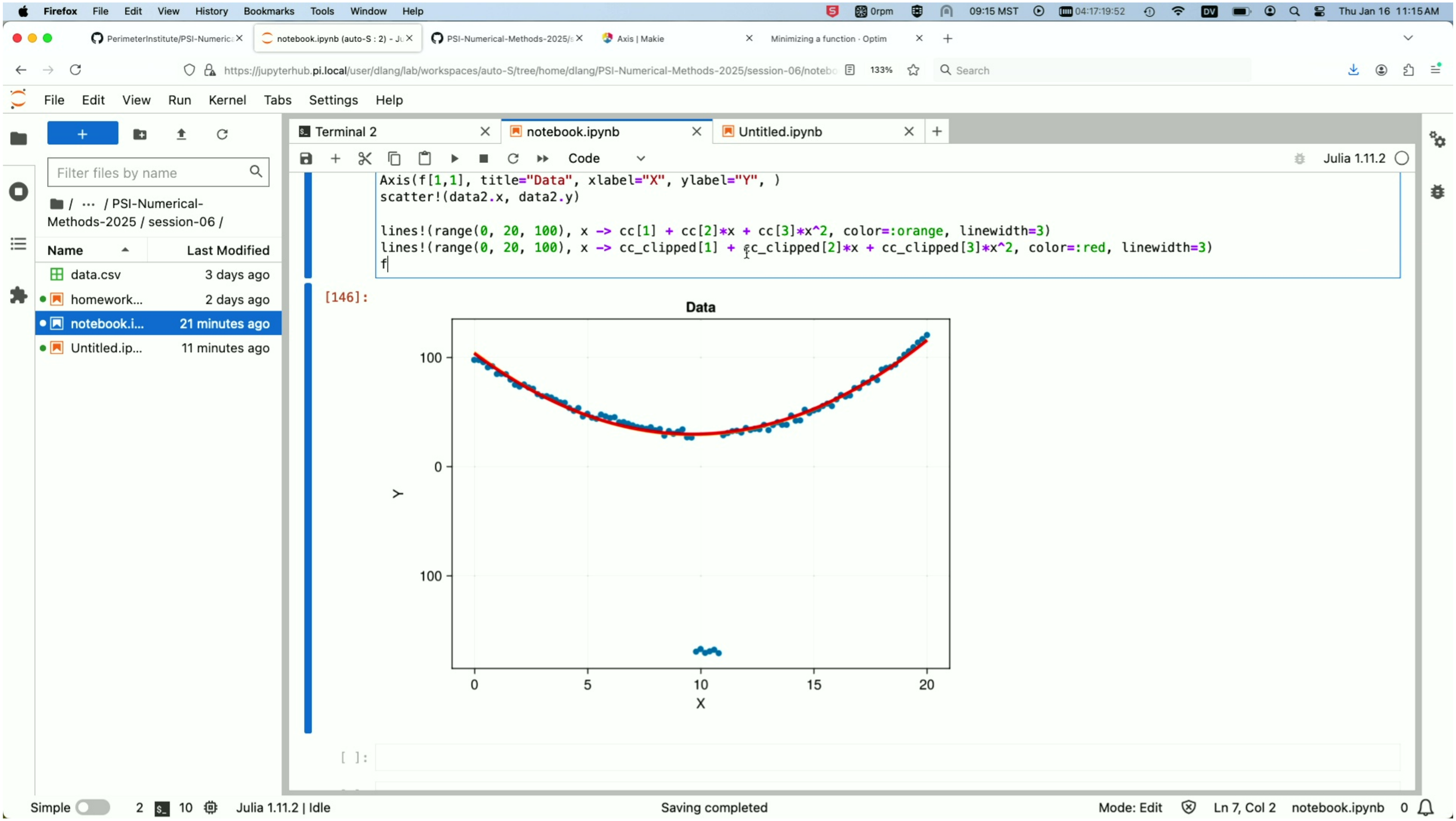
[133]:
```
typeof(answer)
```

[133]:
```
Optim.MultivariateOptimizationResults{NelderMead{Optim.AffineSimplexer, Optim.AdaptiveParameters}, Vector{Float64}, Float64, Float64
, Vector{OptimizationState{Float64, NelderMead{Optim.AffineSimplexer, Optim.AdaptiveParameters}}}, Bool, @NamedTuple{f_limit_reache
d::Bool, g_limit_reached::Bool, h_limit_reached::Bool, time_limit::Bool, callback::Bool, f_increased::Bool}}
```

• [131…
```
f = Figure()
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y", )
scatter!(data2.x, data2.y)

lines!(range(0, 20, 100), x -> cc[1] + cc[2]*x + cc[3]*x^2, color=:orange, linewidth=3)
lines!(range(0, 20, 100), x -> cc_clipped[1] + cc_clipped[2]*x + cc[3]*x^2, color=:orange, linewidth=3)
f
```

[131]:




Firefox File Edit View History Bookmarks Tools Window Help
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-S/tree/home/dlang/PSI-Numerical-Methods-2025/session-06/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 2
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ … / PSI-Numerical-Methods-2025 / session-06 /
Name
Last Modified
data.csv 3 days ago
homework... 2 days ago
notebook.i... 21 minutes ago
Untitled.ip... 11 minutes ago
Axis(f[1,1], title="Data", xlabel="X", ylabel="Y",)
scatter!(data2.x, data2.y)
lines!(range(0, 20, 100), x -> cc[1] + cc[2]*x + cc[3]*x^2, color=:orange, linewidth=3)
lines!(range(0, 20, 100), x -> cc_clipped[1] + cc_clipped[2]*x + cc_clipped[3]*x^2, color=:red, linewidth=3)
f
[146]:
Data
100
0
100
Y
0
5
10
15
20
X
[ ]:
Simple
Julia 1.11.2 | Idle
Saving completed
Mode: Edit
Ln 7, Col 2
notebook.ipynb