Julia Smooth Optimizers

JSO-compliant solvers, news, and current challenges

@Julia Opt Days Paris 2023

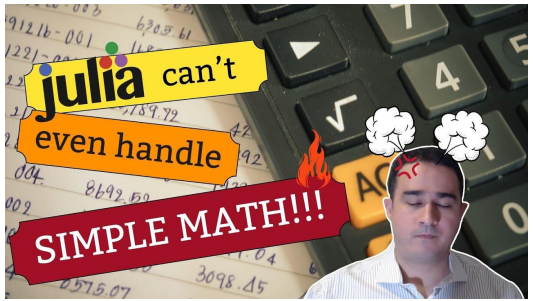
Abel Soares Siqueira

netherlands eScience center

netherlands eScience center

Brief Personal Introduction

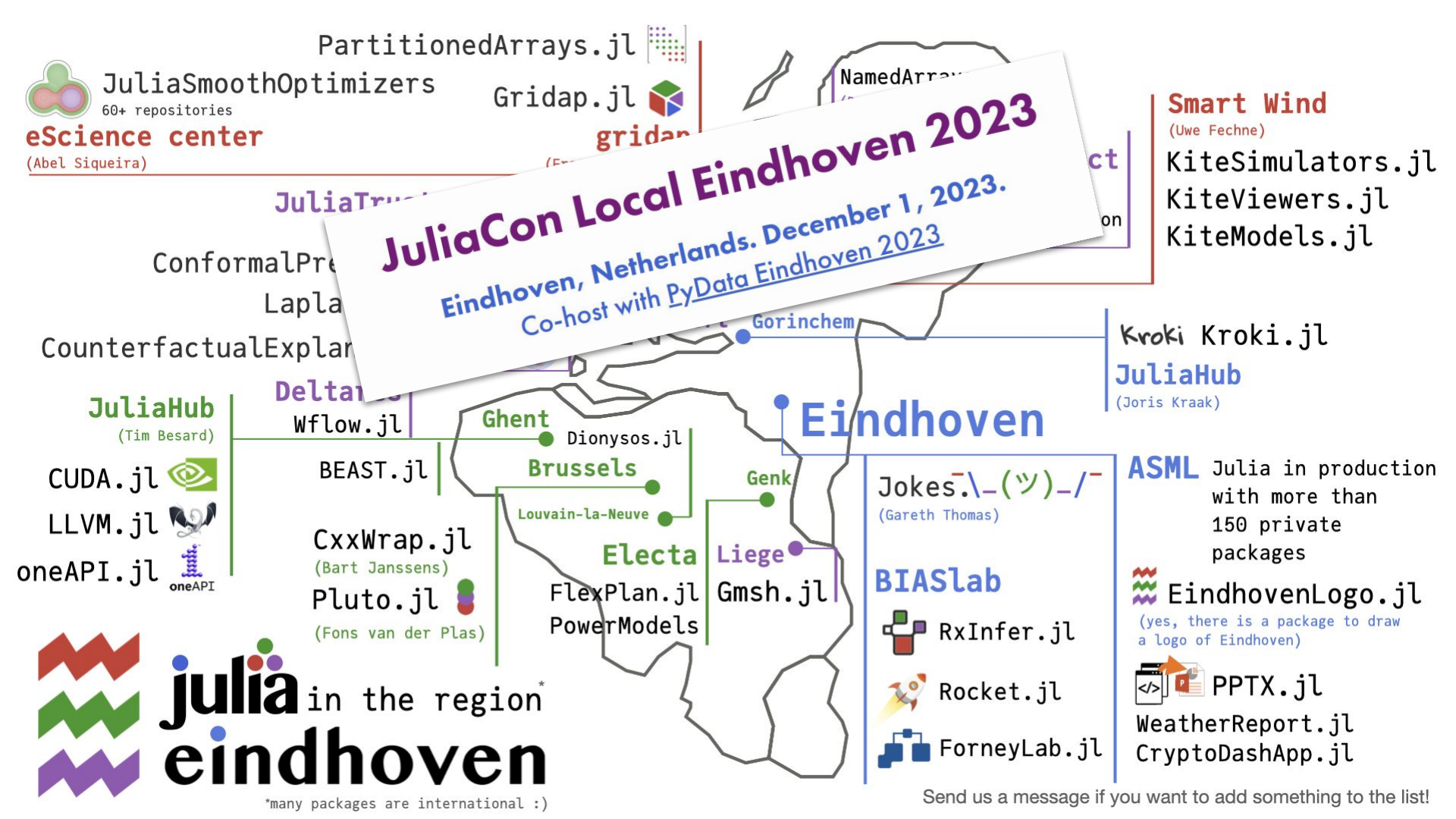
- Brazilian, Former professor (2014-2021)
- Sr Research Software Engineer - since 2021 @ Netherlands eScience Center
- YouTuber - since 2020
  - @AbelSiqueira
- https://abelsiqueira.com



netherlands eScience center

Netherlands eScience Center

- ~80 Research Software Engineers
- Funded by gov., provide in-kind support to researchers in NL
- Various kinds of projects
  - Typescript + Vue + Quasar, bit of C++, Python (Graph NN), Julia/JuMP for energy models
- Hiring again soon


JuliaCon Local Eindhoven 2023
Eindhoven, Netherlands. December 1, 2023.
Co-host with PyData Eindhoven 2023
PartitionedArrays.jl
JuliaSmoothOptimizers
60+ repositories
Gridap.jl
eScience center
(Abel Siqueira)
Smart Wind
(Uwe Fechne)
KiteSimulators.jl
KiteViewers.jl
KiteModels.jl
ConformalPre
Lapla
CounterfactualExplar
Gorinchem
Kroki Kroki.jl
JuliaHub
(Joris Kraak)
JuliaHub
(Tim Besard)
CUDA.jl
LLVM.jl
oneAPI.jl
oneAPI
Wflow.jl
Ghent
Dionysos.jl
BEAST.jl
Brussels
Louvain-la-Neuve
CxxWrap.jl
(Bart Janssens)
Pluto.jl
(Fons van der Plas)
Eindhoven
Genk
Electa
Liege
FlexPlan.jl
PowerModels
Gmsh.jl
Jokes.¯_(ツ)_/¯
(Gareth Thomas)
BIASlab
RxInfer.jl
Rocket.jl
ForneyLab.jl
ASML Julia in production with more than 150 private packages
EindhovenLogo.jl
(yes, there is a package to draw a logo of Eindhoven)
PPTX.jl
WeatherReport.jl
CryptoDashApp.jl
julia in the region*
eindhoven
*many packages are international :)
Send us a message if you want to add something to the list!

netherlands eScience center

FOSDEM - Julia dev room?

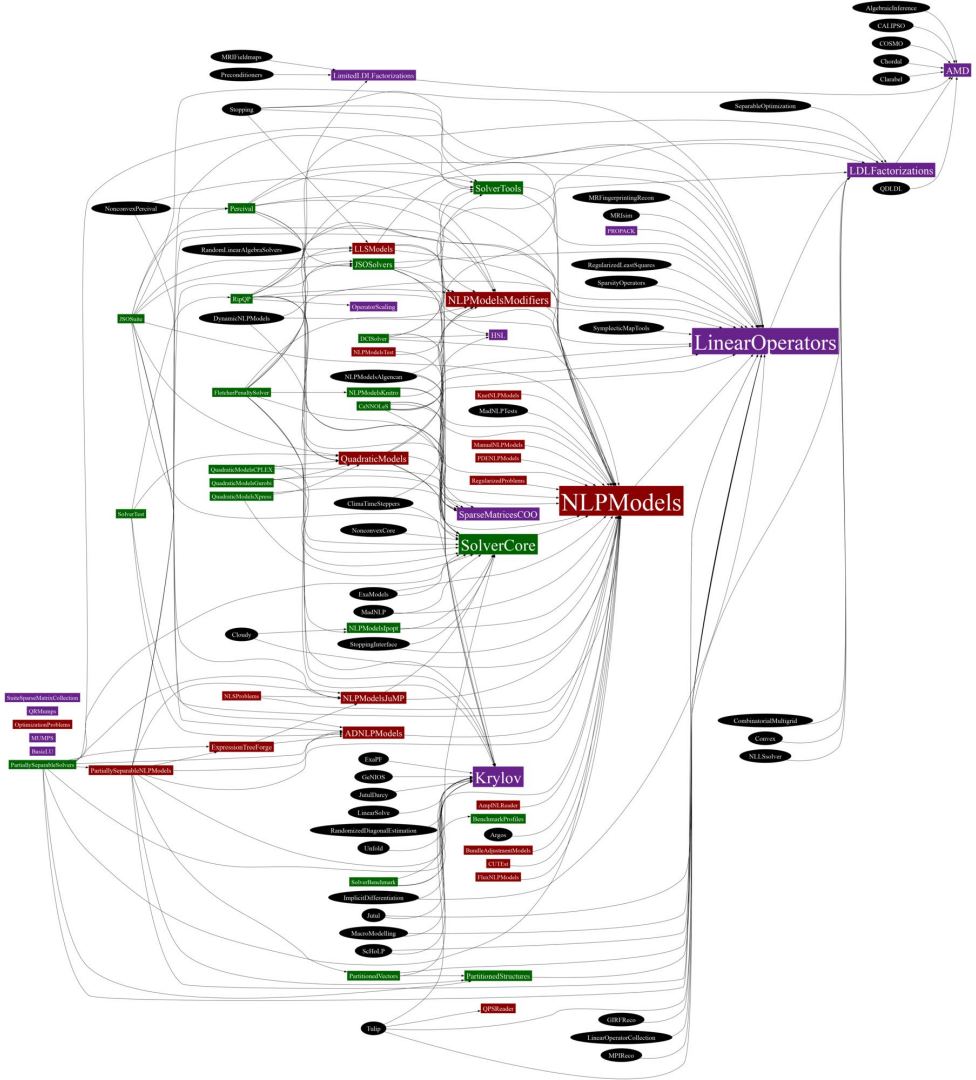
- Still applying for it
- Luca Ferranti (@lucaferranti on Slack)

netherlands eScience center

Julia Smooth Optimizers

- Over 50 pkg, 1000 stars, 75 contributors, 10k commits
- Started in **2015** with CUTEst.jl
- Research oriented
- Optimization and Linear Algebra

netherlands eScience center



Nonlinear optimization

$$
\begin{aligned}
\min \quad & f(x) \\
\text{s. to} \quad & c_L \leq c(x) \leq c_U \\
& \ell \leq x \leq u
\end{aligned}
$$

$$f(x)$$

$$c(x)$$

$$\nabla f(x)$$

$$J(x) = [\nabla c_i(x)^T]_i$$

$$\nabla^2 f(x)$$

$$\nabla^2 c_i(x)$$

Stage	Input	Method/Linear Algebra	Validation
Prototyping	By hand	Dense matrices Backslash (A \ b)	Finds a solution?
Theoretical paper	Small selection	-	Comparison to hand-coded alternative
Computational paper	Collection of problems	Sparse factorization Matrix-free methods	Comparison to established solvers
Maintainable package	Modeling language? Customized model?	-	CI

netherlands eScience center

JSO-compliant solver - streamlined solver development

`AbstractNLPModel` → **Your method** → `GenericExecutionStats`

- Single mandatory input
- API from NLPModels.jl
- Over 10 implementations

- JSOSolverSkeleton.jl
- Sparse Factorizations
- Matrix-free

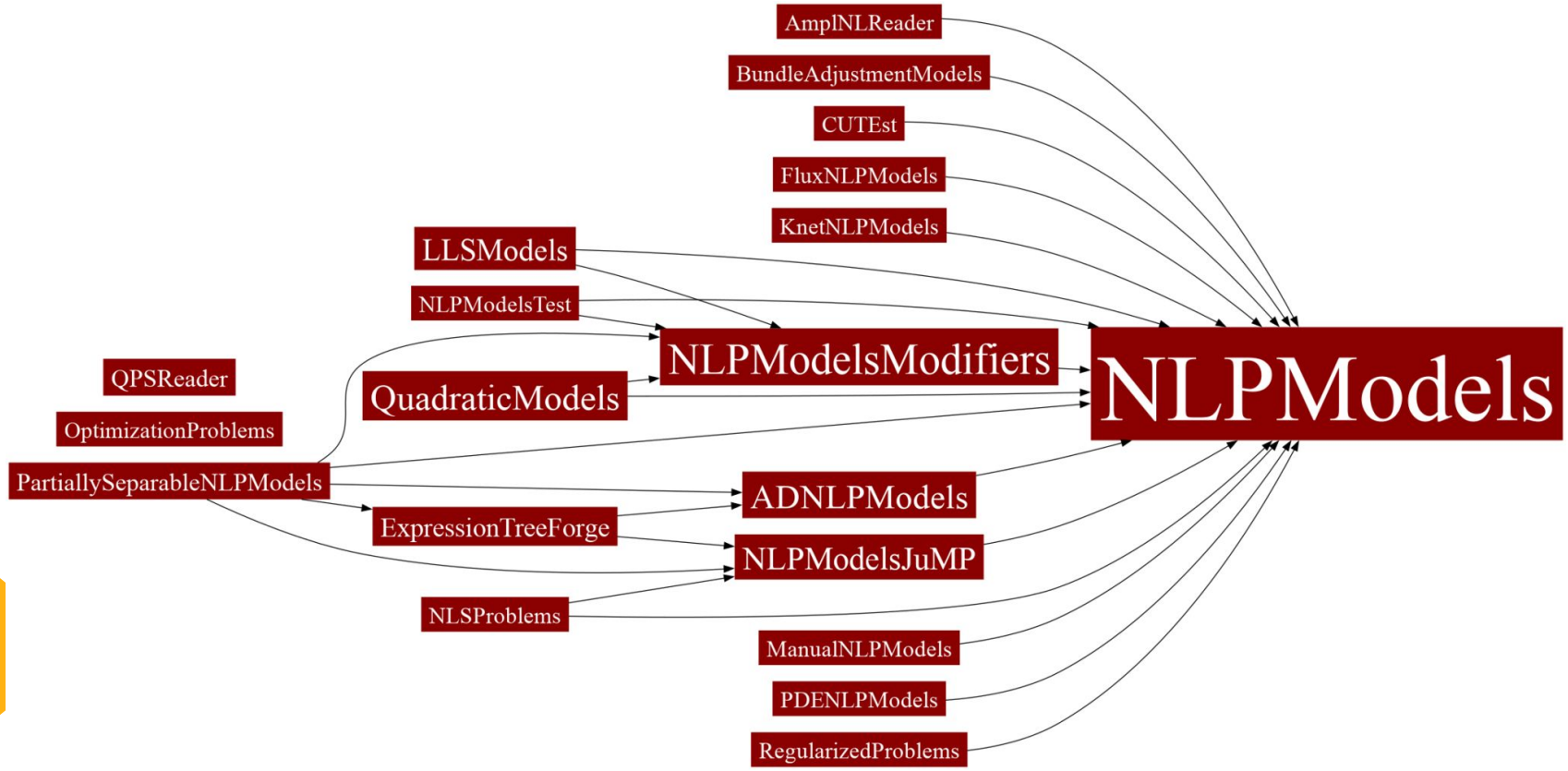
- status
- solution
- eval counters
- solver_specific Dict

netherlands eScience center

Our 3 ecosystems

netherlands eScience center

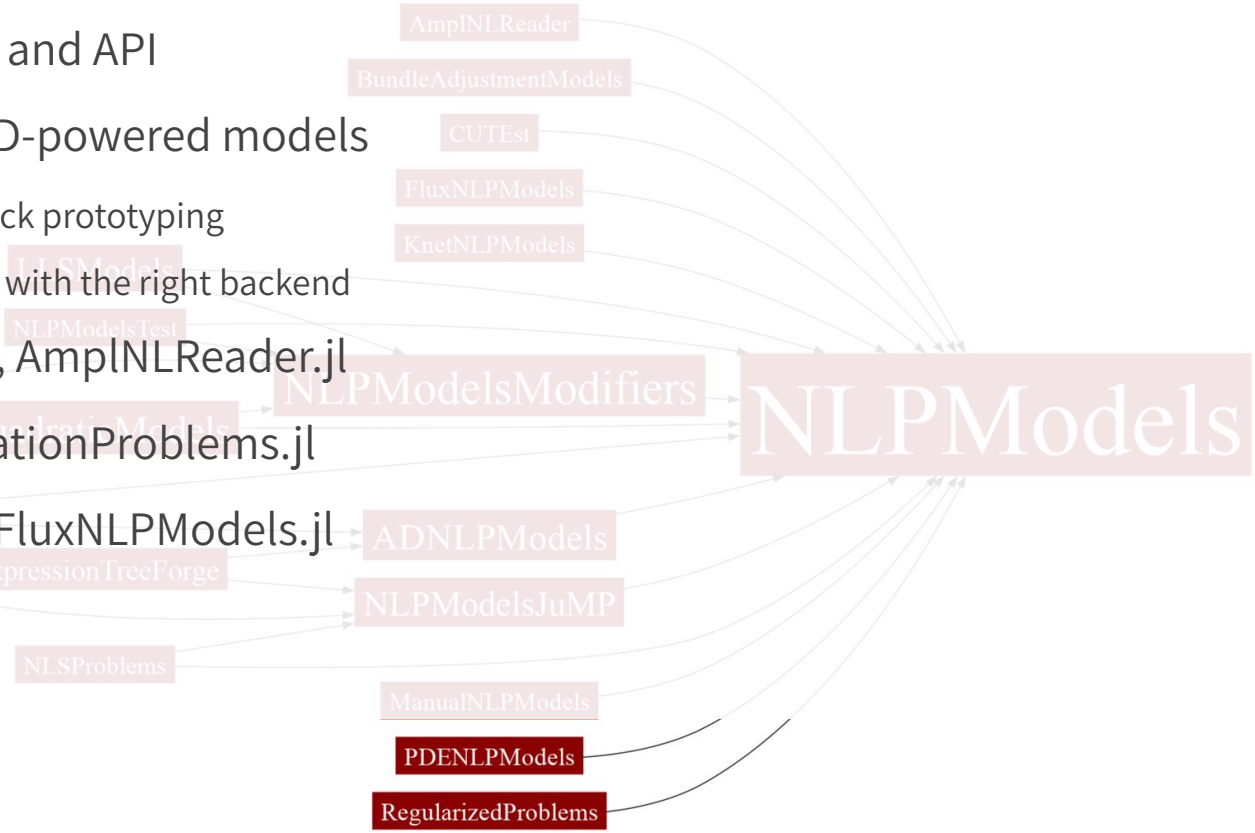
Model ecosystem - NLPModels.jl



netherlands eScience center

Model ecosystem - NLPModels.jl

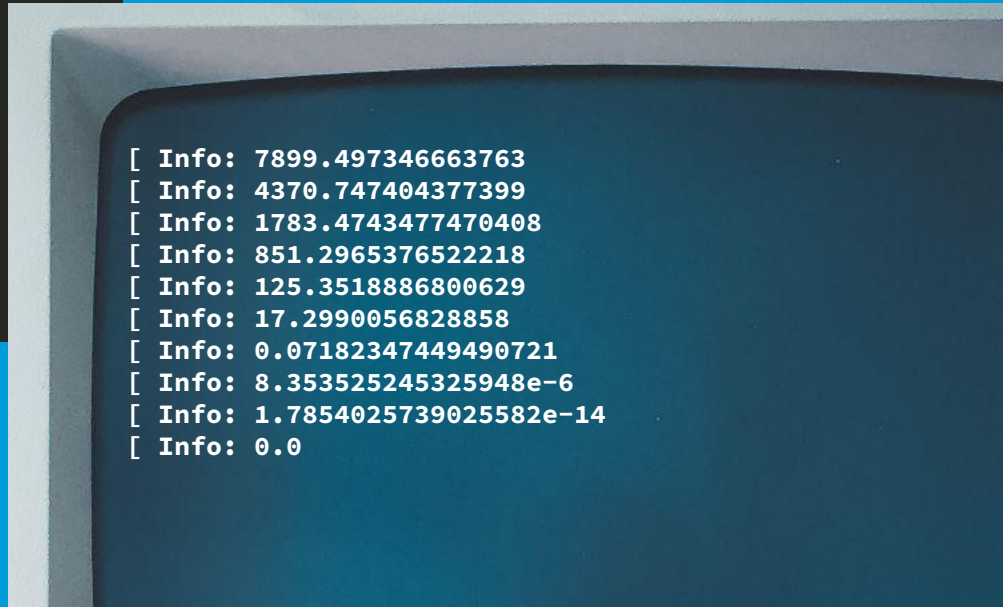
- AbstractNLPModel and API
- ADNLPModels.jl: AD-powered models
  - Simple API for quick prototyping
  - Sparsity available with the right backend
- NLPModelsJuMP.jl, AmplNLReader.jl
- CUTEst.jl, OptimizationProblems.jl
- PDENLPModels.jl, FluxNLPModels.jl
- NLPModelsTest.jl



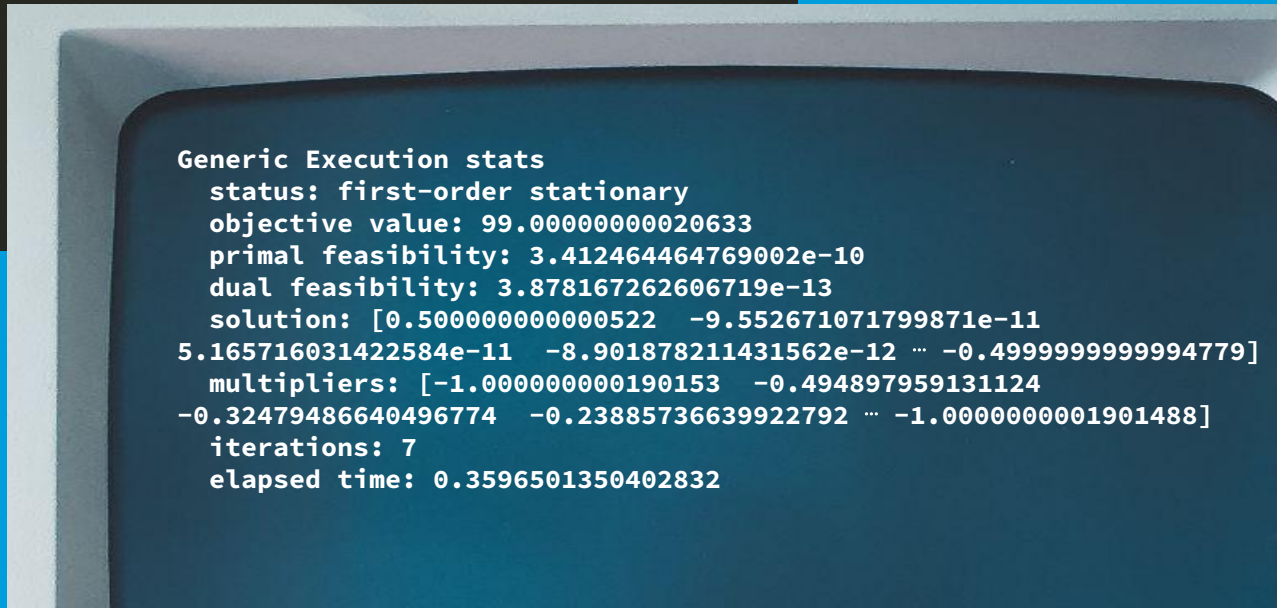
```julia
using ADNLPModels, NLPModels

nlp = ADNLPModel(
  x -> (x[1] - 1)^2 + 4 * (x[2] - x[1]^2)^2, # objective
  [-1.2; 1.0], # starting point
)

x = copy(nlp.meta.x0)
@info x
for i = 1:10
  x .-= hess(nlp, x) \ grad(nlp, x)
  @info x
end
```

```
[ Info: [-1.2, 1.0]
[ Info: [-0.7132743362831855, 0.27185840707964526]
[ Info: [-0.12151367194279328, -0.33541511139156144]
[ Info: [0.17350928196578896, -0.05693307240466178]
[ Info: [0.6607383400906399, 0.19918299898450476]
[ Info: [0.7777599336711202, 0.5912164610599902]
[ Info: [0.9780569732145443, 0.9164765388037358]
[ Info: [0.9946685075749813, 0.9890894968876326]
[ Info: [0.9999882564172207, 0.9999482132446086]
[ Info: [0.9999999973418803, 0.9999999945459112]
[ Info: [1.0, 1.0]
```

```julia
using Symbolics

n = 5000
nlp = ADNLPModel(
  x -> sum(
    (x[2i-1] - 1)^2 + 4 * (x[2i] - x[2i-1]^2)^2
    for i = 1:div(n, 2)
  ),
  [i % 2 == 0 ? 1.0 : -1.2 for i = 1:n],
  hessian_backend=ADNLPModels.SparseADHessian,
)

x = copy(nlp.meta.x0)
for i = 1:10
  x .-= hess(nlp, x) \ grad(nlp, x)
  @info obj(nlp, x)
end
```



```julia
using JuMP, NLPModelsJuMP, Percival

n = 100
jmp = Model()
@variable(jmp, x[1:n])
@variable(jmp, y[1:n])
@objective(jmp, Min,
  sum((x[i] - 1)^2 for i = 1:n) + sum(y[i]^2 for i = 1:n)
)
@constraint(jmp, [i = 1:n, j = i+1:n], x[i] + x[j] == y[i] - y[j])

nlp = MathOptNLPModel(jmp)

output = percival(nlp)
println(output)
```



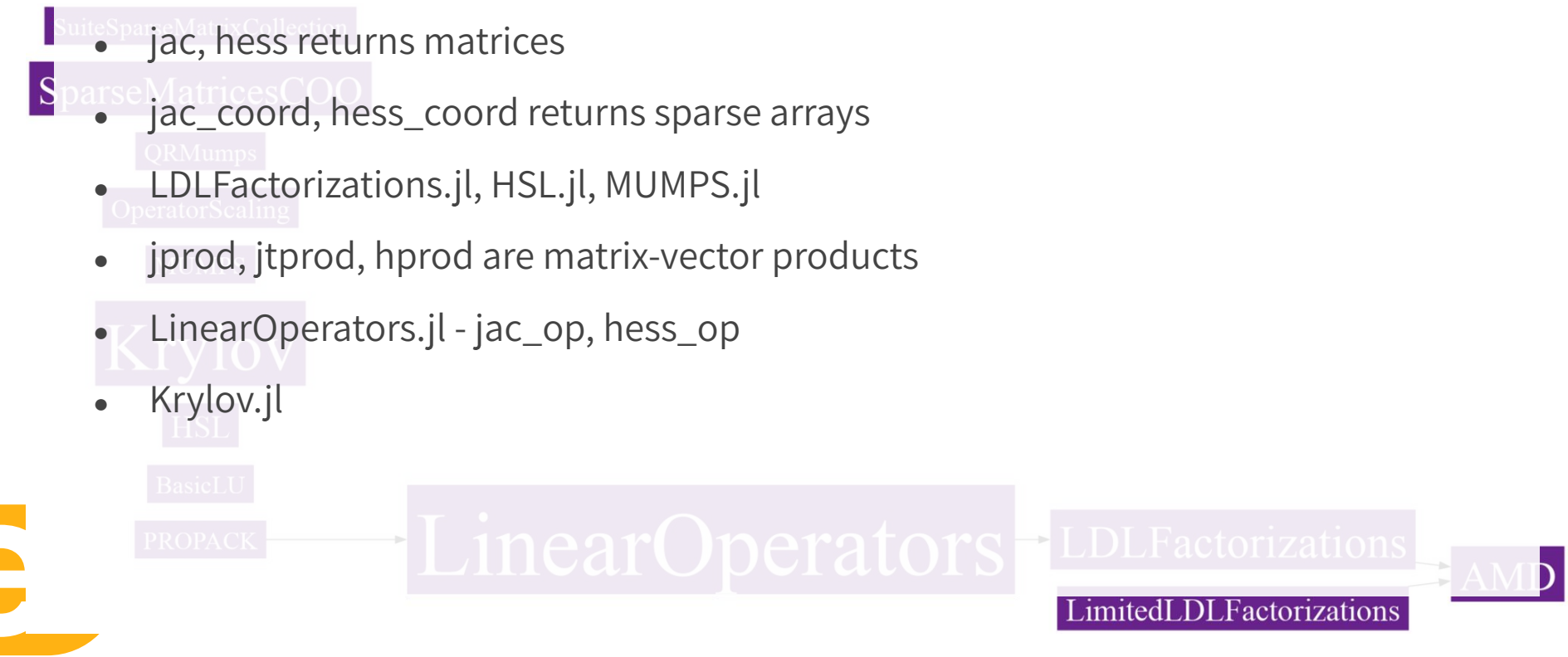
netherlands eScience center

Linear Algebra ecosystem

netherlands eScience center

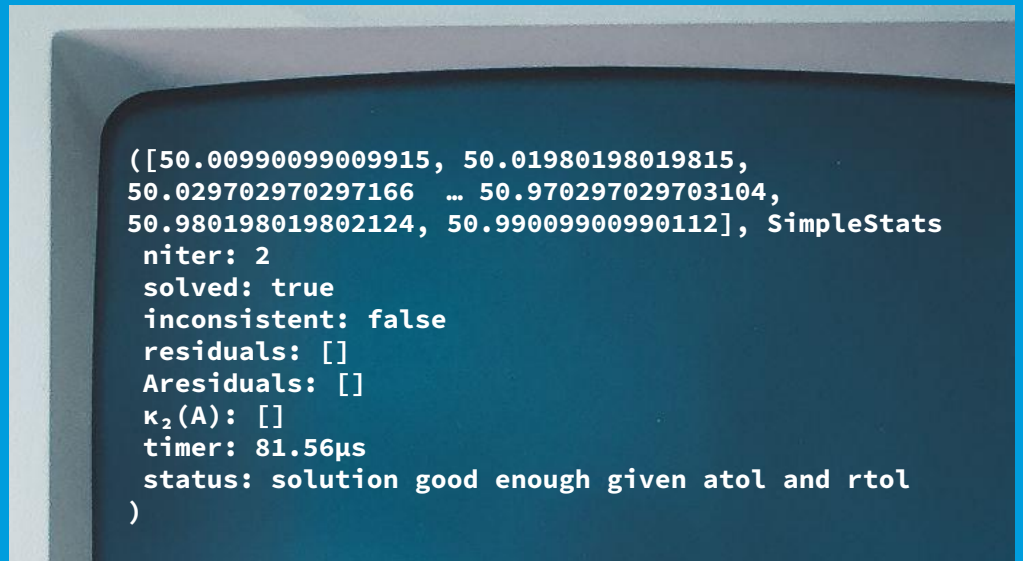
Linear Algebra ecosystem

- jac, hess returns matrices
- jac_coord, hess_coord returns sparse arrays
- LDLFactorizations.jl, HSL.jl, MUMPS.jl
- jprod, jtprod, hprod are matrix-vector products
- LinearOperators.jl - jac_op, hess_op
- Krylov.jl

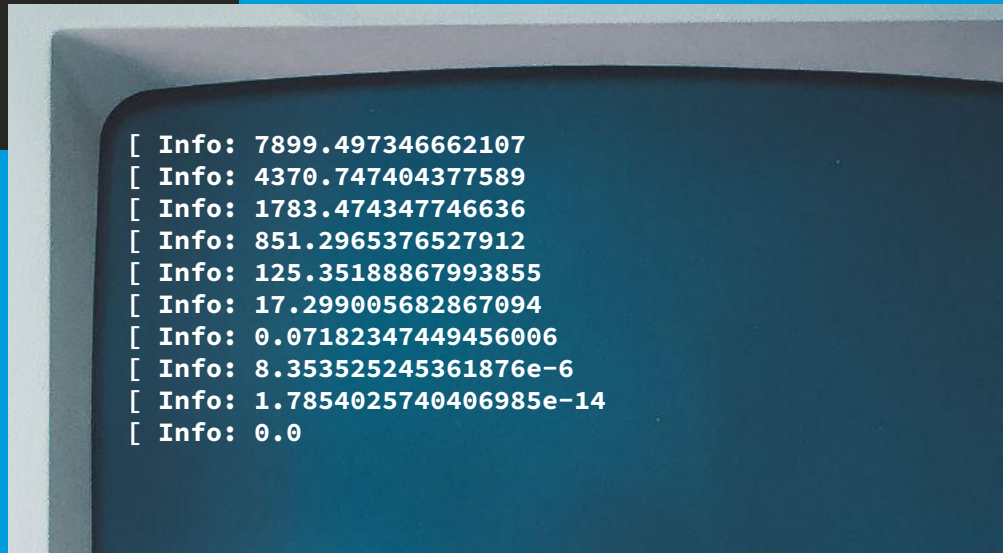


```julia
using Krylov

n = 100
A = [i == j ? 1.0n : -1.0 for i = 1:n, j = 1:n]
b = [1.0i for i = 1:n]

x, cg_stats = Krylov.cg(A, b)
```

```julia
using LinearOperators, LinearAlgebra

function Av(res, v)
  res .= (n + 1) * v .- sum(v)
  return res
end
A = LinearOperator(Float64, n, n, true, true, Av)
y, cg_stats = Krylov.cg(A, b)
```

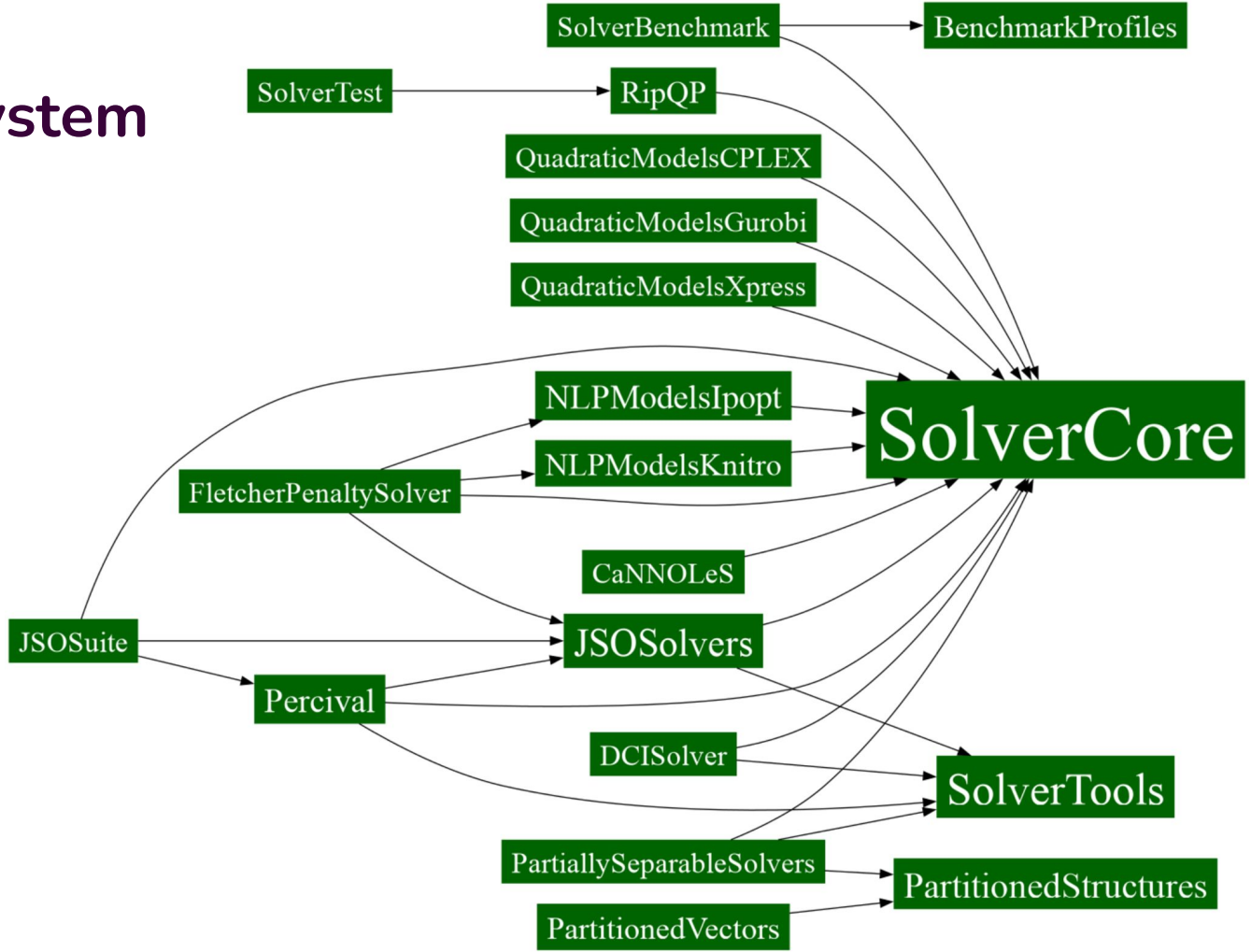


```julia
n = 5000
nlp = ADNLPModel(
  x -> sum(
    (x[2i-1] - 1)^2 + 4 * (x[2i] - x[2i-1]^2)^2
    for i = 1:div(n, 2)
  ),
  [i % 2 == 0 ? 1.0 : -1.2 for i = 1:n],
)

x = copy(nlp.meta.x0)
for i = 1:10
  d, cg_stats = Krylov.cg(hess_op(nlp, x), grad(nlp, x))
  x .-= d
  @info obj(nlp, x)
end
```



netherlands eScience center

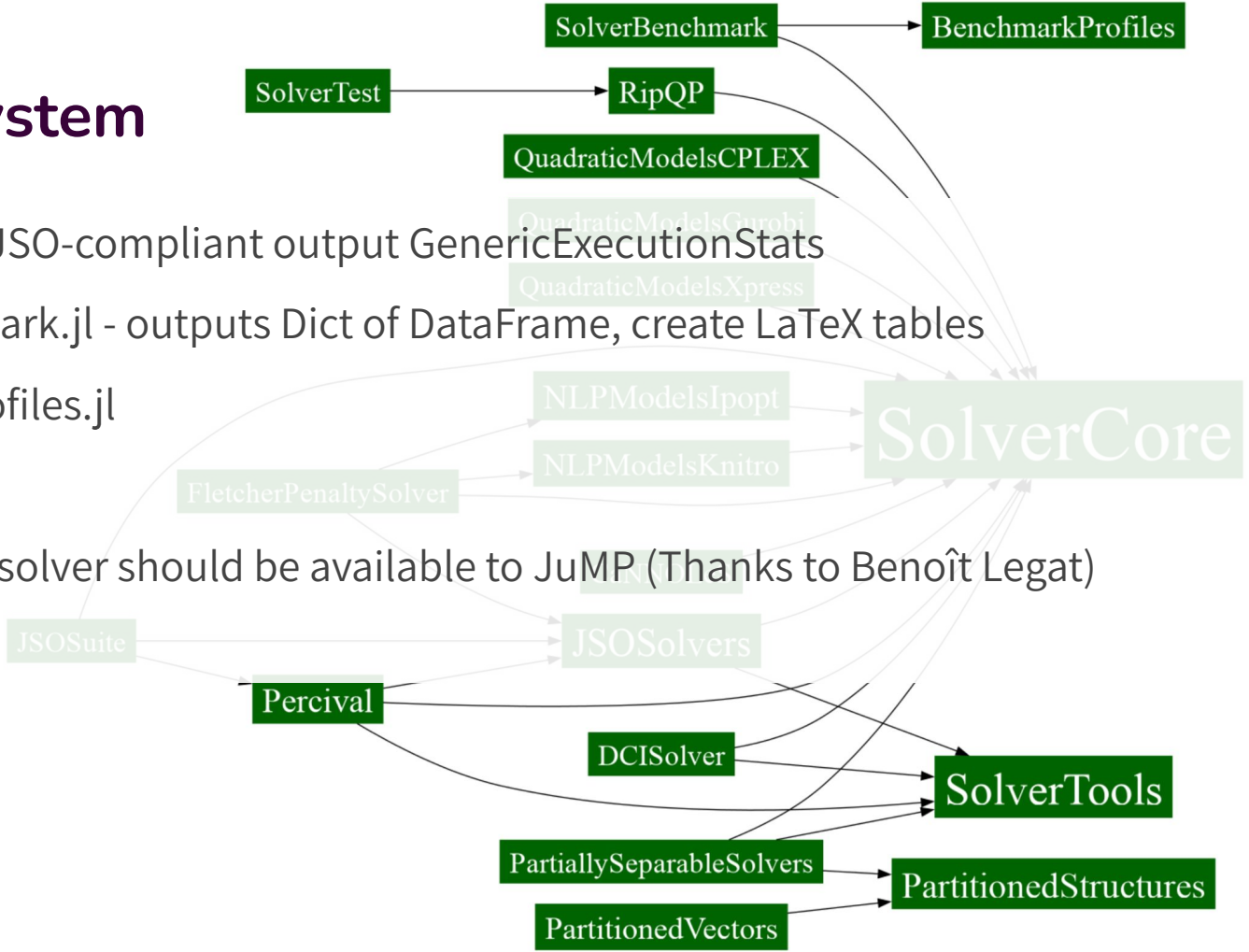
Solver ecosystem



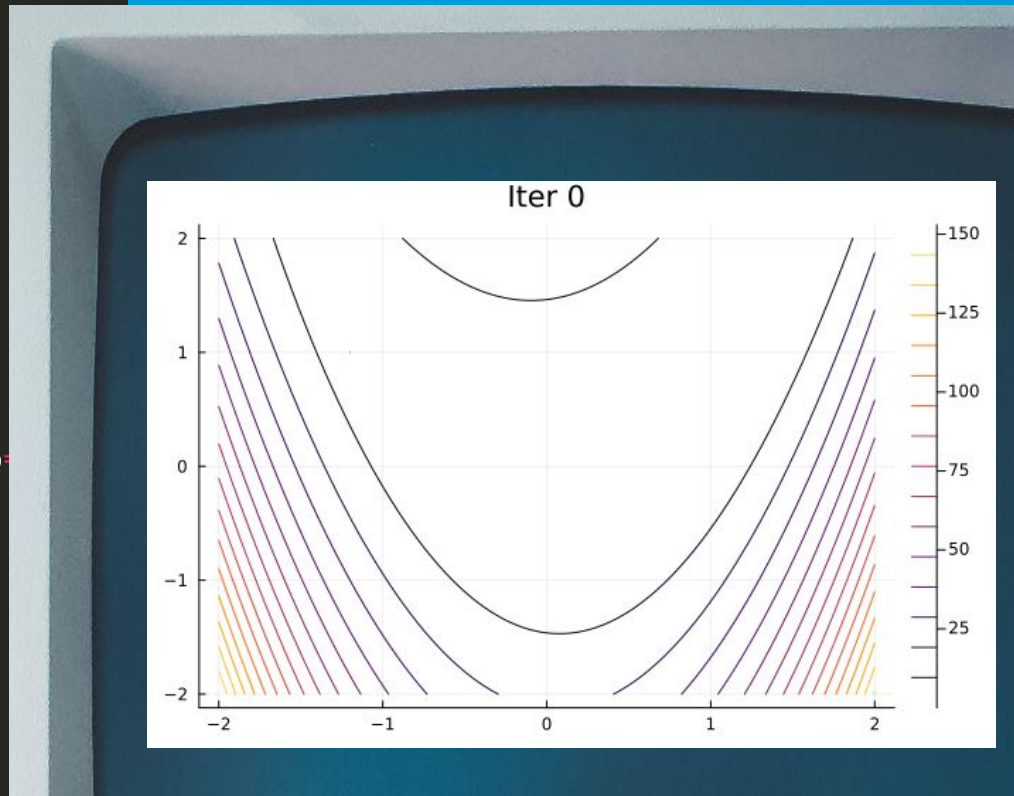
netherlands eScience center

Solver ecosystem

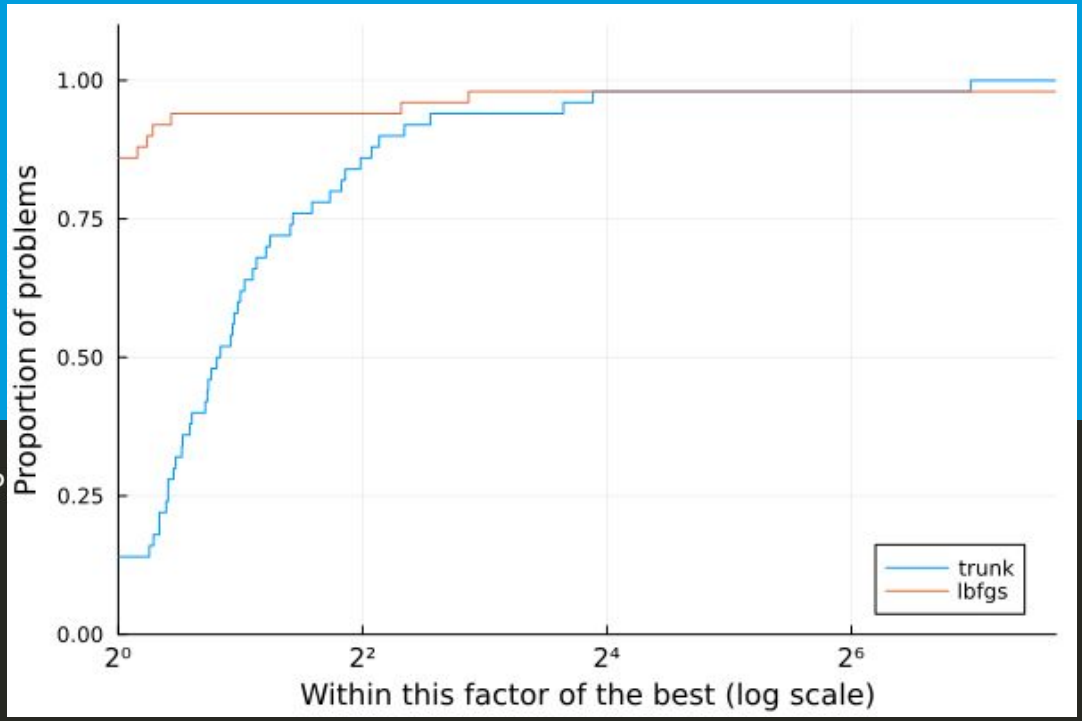
- SolverCore.jl: JSO-compliant output GenericExecutionStats
- SolverBenchmark.jl - outputs Dict of DataFrame, create LaTeX tables
- BenchmarkProfiles.jl
- SolverTest.jl
- Recent: These solver should be available to JuMP (Thanks to Benoît Legat)



```julia
using Plots, JSOSolvers, ADNLPModels, NLPModels

function wrapper()
  anim = Animation()
  nlp = ADNLPModel(
    x -> (x[1] - 1)^2 + 4 * (x[2] - x[1]^2)^2,
    [-1.2; 1.0],
  )
  X = zeros(2, 1)
  X[:, 1] .= nlp.meta.x0
  cb = (nlp, solver, stats) -> begin
    X = [X solver.x]
    contour(
      range(-2, 2, length=100),
      range(-2, 2, length=100),
      (x, y) -> obj(nlp, [x; y]),
    )
    plot!(X[1, :], X[2, :], c=:red, l=:arrow, lab
    title!("Iter $(stats.iter)")
    frame(anim)
  end
  output = trunk(nlp, callback=cb, max_iter=100)
  gif(anim, "callback.gif", fps=5)
end
wrapper()
```



netherlands eScience center




```julia
using CUTEst, SolverBenchmark, BenchmarkP

solvers = Dict(
  :lbfgs => lbfgs,
  :trunk => trunk,
)
problem_names = CUTEst.select(max_var=2, max_con=0, only_free_var=true)
problems = (CUTEstModel(p) for p in problem_names)
df_per_solver = with_logger(NullLogger()) do
  bmark_solvers(solvers, problems)
end
performance_profile(df_per_solver, df -> (df.status .!= :first_order) * Inf + df.elapsed_time)
png("perprof")
```

netherlands eScience center

What's new / future plans

netherlands eScience center

JSOSuite.jl

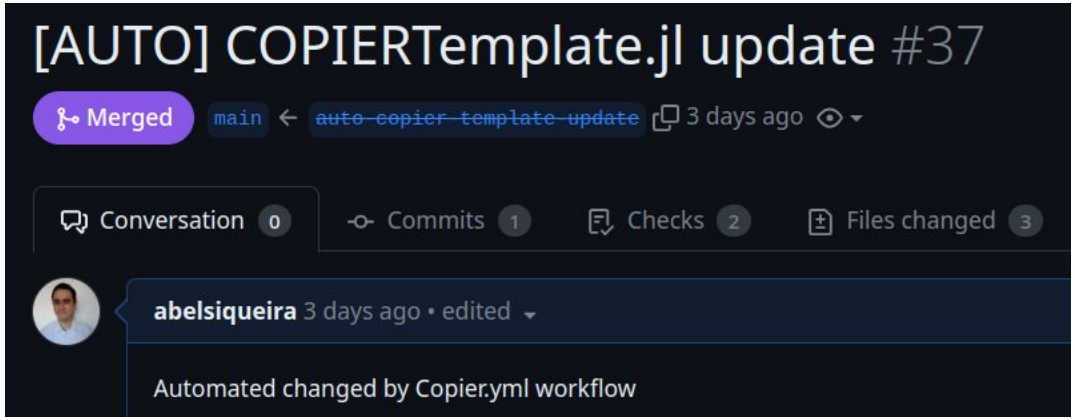
- User-friendly interface
- Select solver amongst loaded
- Wider community
- You can still use JSO solvers through Nonconvex.jl

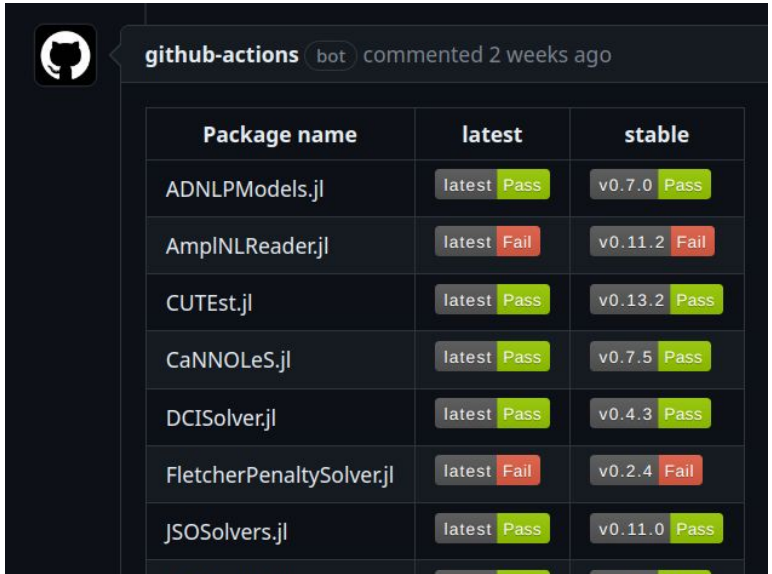
```
using JSOSuite

output = minimize(
  x -> (x[1] - 1)^2 + 4 * (x[2] - x[1]^2)^2,
  [-1.2; 1.0],
)
```

netherlands eScience center

Challenges

- More core developers
- Maintaining 50+ packages
  - **COPIERTemplate.jl**
  - **Breakage.yml**
  - Jira?





netherlands eScience center

https://jso.dev

- News and tutorials
- Comm. channels

Tutorials and how-to guides

This is a curated list of tutorials.

Show tags

Introduction to SolverBenchmark

In this tutorial we illustrate the main uses of SolverBenchmark.

solver benchmark profile latex SolverBenchmark

Creating an ADNLPModels backend that supports multiple precisions

One of the main strengths of Julia for scientific computing is its native usage of arbitrary precision arithmetic, the same optimization models and solvers.

models automatic differentiation multi-precision tests ADNLPModels JSOSolvers NLPModels OptimizationProblems

Introduction to JSOSolvers

This package provides optimization solvers curated by the JuliaSmoothOptimizers organization.

solvers nlpmodels least-squares nlsmodels test set ADNLPModels JSOSolvers SolverCore

Adjustment in the Large repository

Talk to us

You can find us in the following channels:

- JuliaLang Slack in the `#math-optimization`, `#linear-algebra`, or `#smooth-optimizers`.
- Julia Zulip in the `#math-optimization`.
- GitHub Discussions.
- Abel Siqueira's YouTube.

netherlands eScience center

Thank you

https://abelsiqueira.com

abel.s.siqueira@gmail.com

Like and subscribe