Julia Smooth Optimizers
JSO-compliant solvers, news, and current challenges
@Julia Opt Days Paris 2023

Abel Soares Siqueira
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

- ~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
Co-host with PyData Eindhoven 2023

JuliaHub
(Tim Besnard)
CUDA.jl
LLVM.jl
oneAPI.jl
JuliaSmoothOptimizers
60+ repositories
ConformalPredictors.jl
Laplacian.jl
CounterfactualExplanations.jl
Deltaview.jl
Wflow.jl
BEAST.jl
CxxWrap.jl
(Bart Janssens)
Pluto.jl
(Fons van der Plas)

JuliaHUB
(Joris Kraak)

Julia in the region
Eindhoven
*many packages are international ;)

KiteSimulators.jl
KiteViewers.jl
KiteModels.jl
Kroki Kroki.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 production with more than 150 private packages
FOSDEM - Julia dev room?

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

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

\[ \begin{align*}
    \text{min} & \quad f(x) \\
    \text{s. to} & \quad c_L \leq c(x) \leq c_U \\
    & \quad \ell \leq x \leq u
\end{align*} \]

\[ f(x) \]
\[ c(x) \]
\[ \nabla f(x) \]
\[ J(x) = \left[ \nabla c_i(x)^T \right]_i \]
\[ \nabla^2 f(x) \]
\[ \nabla^2 c_i(x) \]
<table>
<thead>
<tr>
<th>Stage</th>
<th>Input</th>
<th>Method/Linear Algebra</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototyping</td>
<td>By hand</td>
<td>Dense matrices</td>
<td>Finds a solution?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Backslash (A \ b)</td>
<td></td>
</tr>
<tr>
<td>Theoretical paper</td>
<td>Small selection</td>
<td>-</td>
<td>Comparison to hand-coded alternative</td>
</tr>
<tr>
<td>Computational paper</td>
<td>Collection of problems</td>
<td>Sparse factorization Matrix-free methods</td>
<td>Comparison to established solvers</td>
</tr>
<tr>
<td>Maintainable package</td>
<td>Modeling language?</td>
<td>Customized model?</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>CI</td>
</tr>
</tbody>
</table>
JSO-compliant solver - streamlined solver development

AbstractNLPModel

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

Your method

• JSOSolverSkeleton.jl
• Sparse Factorizations
• Matrix-free

GenericExecutionStats

• status
• solution
• eval counters
• solver_specific Dict
Our 3 ecosystems
Model ecosystem - NLPModels.jl
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
using ADNLPMoodels, NLPModels

nlp = ADNLPMoodel(
    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
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
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)
Linear Algebra ecosystem

- SuiteSparseMatrixCollection
- SparseMatricesCOO
  - QRMumps
  - OperatorScaling
  - MUMPS
- Krylov
  - HSL
  - BasicLU
  - PROPACK
- LinearOperators
- LDLFactorizations
  - LimitedLDLFactorizations
- AMD
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`
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)

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)

([50.00990099009915, 50.01980198019815, 50.029702970297166, ..., 50.970297029703104, 50.980198019802124, 50.99009900990112], SimpleStats
niter: 2
solved: true
inconsistent: false
residuals: []
Aresiduals: []
κ₂(A): []
timer: 81.56μs
status: solution good enough given atol and rtol)
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
end
@info obj(nlp, x)
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)
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=false)
        title!("Iter $(stats.iter)")
        frame(anim)
    end
    output = trunk(nlp, callback=cb, max_iter=100)
    gif(anim, "callback.gif", fps=5)
end
wrapper()
using CUTEst, SolverBenchmark, BenchmarkPro

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")
What’s new / future plans
JSOSuite.jl

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

```plaintext
using JSOSuite

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

- More core developers
- Maintaining 50+ packages
  - COPIERTemplate.jl
  - Breakage.yml
  - Jira?
Tutorials and how-to guides

This is a curated list of tutorials.

- Introduction to SolverBenchmark
  In this tutorial we illustrate the main uses of 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.

- Introduction to JSOSolvers
  This package provides optimization solvers curated by the JuliaSmoothOptimizers organization.

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.
Thank you

Like and subscribe

https://abelsiqueira.com
abel.s.siqueira@gmail.com