The Journal of Open Source Software

RegularizedOptimization.jl: A Julia framework for regularized and nonsmooth optimization

Youssef Diouane [1], **Maxence Gollier** [1¶], **Mohamed Laghdaf Habiboullah** [1¶], **and Dominique Orban** [1]

1 GERAD and Department of Mathematics and Industrial Engineering, Polytechnique Montréal, QC, Canada ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 03 October 2025
Published: unpublished

Summary

RegularizedOptimization.jl is a Julia package that implements families of quadratic regularization and trust-region methods for solving the nonsmooth optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)+h(x) \quad \text{subject to} \quad c(x)=0, \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ and $c:\mathbb{R}^n\to\mathbb{R}^m$ are continuously differentiable, and $h:\mathbb{R}^n\to\mathbb{R}\cup\{+\infty\}$ is lower semi-continuous. The nonsmooth objective h can be a *regularizer*, such as a sparsity-inducing penalty, model simple constraints, such as x belonging to a simple convex set, or be a combination of both. All f, h and c can be nonconvex. RegularizedOptimization.jl provides a modular and extensible framework for solving (1), and developing novel solvers. Currently, the following solvers are implemented:

- **Trust-region solvers TR and TRDH** (Aravkin et al., 2022; Leconte & Orban, 2025)
- **Quadratic regularization solvers R2, R2DH and R2N** (Aravkin et al., 2022; Diouane, Habiboullah, et al., 2024)
- **Levenberg-Marquardt solvers LM and LMTR** (Aravkin et al., 2024) used when f is a least-squares residual.
- **Augmented Lagrangian solver AL** (De Marchi et al., 2023).

All solvers rely on first derivatives of f and c, and optionally on their second derivatives in the form of Hessian-vector products. If second derivatives are not available, quasi-Newton approximations can be used. In addition, the proximal mapping of the nonsmooth part h, or adequate models thereof, must be evaluated. At each iteration, a step is computed by solving a subproblem of the form (1) inexactly, in which f, h, and c are replaced with appropriate models about the current iterate. The solvers R2, R2DH and TRDH are particularly well suited to solve the subproblems, though they are general enough to solve (1). All solvers are implemented in place, so re-solves incur no allocations. To illustrate our claim of extensibility, a first version of the AL solver was implemented by an external contributor. Furthermore, a nonsmooth penalty approach, described in (Diouane, Gollier, et al., 2024) is currently being developed, that relies on the library to efficiently solve the subproblems.

Statement of need

Model-based framework for nonsmooth methods

In Julia, (1) can be solved using ProximalAlgorithms.jl, which implements splitting schemes and line-search–based methods (Stella et al., 2017; Themelis et al., 2018). Among others, the **PANOC** (Stella et al., 2017) solver takes a step along a direction d, which depends on the L-BFGS quasi-Newton approximation of f, followed by proximal steps on h.

By contrast, RegularizedOptimization.jl focuses on model-based trust-region and quadratic regularization methods, which typically require fewer evaluations of f and its gradient than first-order line search methods, at the expense of more evaluations of proximal operators (Aravkin et al., 2022). However, each proximal computation is inexpensive for numerous commonly used choices of h, such as separable penalties and bound constraints, so that the overall approach is efficient for large-scale problems.

RegularizedOptimization.jl provides an API to formulate optimization problems and apply different solvers. It integrates seamlessly with the JuliaSmoothOptimizers (Migot et al., 2021) ecosystem.

The smooth objective f can be defined via NLPModels.jl (Orban et al., 2020), which provides a standardized Julia API for representing nonlinear programming (NLP) problems. The nonsmooth term h can be modeled using ProximalOperators.jl.

Given f and h, the companion package RegularizedProblems.jl provides a way to pair them into a *Regularized Nonlinear Programming Model*

```
reg_nlp = RegularizedNLPModel(f, h)
```

They can also be paired into a *Regularized Nonlinear Least-Squares Model* if $f(x) = \frac{1}{2}\|F(x)\|^2$ for some residual $F : \mathbb{R}^n \to \mathbb{R}^m$, in the case of the **LM** and **LMTR** solvers.

```
reg_nls = RegularizedNLSModel(F, h)
```

RegularizedProblems.jl also provides a set of instances commonly used in data science and in nonsmooth optimization, where several choices of f can be paired with various regularizers. This design makes for a convenient source of problem instances for benchmarking the solvers in RegularizedOptimization.jl.

Support for both exact and approximate Hessian

In contrast with ProximalAlgorithms.jl, RegularizedOptimization.jl, methods such as **R2N** and **TR** methods support exact Hessians as well as several Hessian approximations of f. Hessian–vector products $v \mapsto Hv$ can be obtained via automatic differentiation through ADNLPModels.jl or implemented manually. Limited-memory and diagonal quasi-Newton approximations can be selected from LinearOperators.jl. This design allows solvers to exploit second-order information without explicitly forming dense or sparse Hessians, which is often expensive in time and memory, particularly at large scale.

Example

We illustrate the capabilities of RegularizedOptimization.jl on a Support Vector Machine (SVM) model with a $\ell^{1/2}_{1/2}$ penalty for image classification (Aravkin et al., 2024).

Below is a condensed example showing how to define and solve the problem, and perform a solve followed by a re-solve:

```
using LinearAlgebra, Random, ProximalOperators
using NLPModels, RegularizedProblems, RegularizedOptimization
using MLDatasets

Random.seed!(1234)
model, nls_model, _ = RegularizedProblems.svm_train_model()  # Build SVM model
f = LSR1Model(model)                                         # L-SR1 Hessian approximation
λ = 1.0                                                      # Regularization parameter
h = RootNormLhalf(λ)                                         # Nonsmooth term
reg_nlp = RegularizedNLPModel(f, h)                          # Regularized problem
```

The Journal of Open Source Software

```
solver = R2NSolver(reg_nlp)                                    # Choose solver
stats  = RegularizedExecutionStats(reg_nlp)
solve!(solver, reg_nlp, stats; atol=1e-4, rtol=1e-4, verbose=1, sub_kwargs=(max_iter=200,))
solve!(solver, reg_nlp, stats; atol=1e-5, rtol=1e-5, verbose=1, sub_kwargs=(max_iter=200,))
```

Numerical results

We compare **TR**, **R2N**, **LM** and **LMTR** from our library on the SVM problem.

The table reports the convergence status of each solver, the number of evaluations of f, the number of evaluations of ∇f, the number of proximal operator evaluations, the elapsed time and the final objective value. For TR and R2N, we use limited-memory SR1 Hessian approximations. The subproblem solver is **R2**.

Method	Status	$t(s)$	$\#f$	$\#\nabla f$	$\#prox$	Objective
TR	first_order	3.9349	347	291	4037	179.837
R2N	first_order	1.9511	185	101	27932	192.493
LM	first_order	19.7826	6	2876	1001	201.186
LMTR	first_order	12.4967	11	1614	432	188.274

For the **LM** and **LMTR** solvers, $\#\nabla f$ counts the number of Jacobian–vector and adjoint-Jacobian–vector products.

All methods successfully reduced the optimality measure below the specified tolerance of 10^{-4}, and thus converged to an approximate first-order stationary point. Note that the final objective values differ due to the nonconvexity of the problem.

R2N is the fastest in terms of time and number of gradient evaluations. However, it requires more proximal evaluations, but these are inexpensive. **LMTR** and **LM** require the fewest function evaluations, but incur many Jacobian–vector products, and are the slowest in terms of time.

Ongoing research aims to reduce the number of proximal evaluations.

Acknowledgements

The authors would like to thank A. De Marchi for the Augmented Lagrangian solver. M. L. Habiboullah is supported by an excellence FRQNT grant. Y. Diouane, M. Gollier and D. Orban are partially supported by an NSERC Discovery Grant.

References

Aravkin, A. Y., Baraldi, R., & Orban, D. (2022). A proximal quasi-Newton trust-region method for nonsmooth regularized optimization. *SIAM J. Optim.*, *32*(2), 900–929. https://doi.org/10.1137/21M1409536

Aravkin, A. Y., Baraldi, R., & Orban, D. (2024). A Levenberg–Marquardt method for nonsmooth regularized least squares. *SIAM J. Sci. Comput.*, *46*(4), A2557–A2581. https://doi.org/10.1137/22M1538971

De Marchi, A., Jia, X., Kanzow, C., & Mehlitz, P. (2023). Constrained composite optimization and augmented Lagrangian methods. *Math. Program.*, *201*(1), 863–896. https://doi.org/10.1007/s10107-022-01922-4

Diouane, Y., Gollier, M., & Orban, D. (2024). *A nonsmooth exact penalty method for equality-constrained optimization: Complexity and implementation* (Cahier G-2024-65). GERAD. https://doi.org/10.13140/RG.2.2.16095.47527

Diouane, Y., Habiboullah, M. L., & Orban, D. (2024). *A proximal modified quasi-Newton method for nonsmooth regularized optimization* (Cahier G-2024-64). GERAD. https://www.gerad.ca/fr/papers/G-2024-64

Leconte, G., & Orban, D. (2025). The indefinite proximal gradient method. *Comput. Optim. Appl.*, *91*(2), 861–903. https://doi.org/10.1007/s10589-024-00604-5

Migot, T., Orban, D., & Siqueira, A. S. (2021). *The JuliaSmoothOptimizers ecosystem for linear and nonlinear optimization*. https://doi.org/10.5281/zenodo.2655082

Orban, D., Siqueira, A. S., & contributors. (2020). *NLPModels.jl: Data structures for optimization models*. https://doi.org/10.5281/zenodo.2558627

Stella, L., Themelis, A., Sopasakis, P., & Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1939–1944. https://doi.org/10.1109/CDC.2017.8263933

Themelis, A., Stella, L., & Patrinos, P. (2018). Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line search algorithms. *SIAM J. Optim.*, *28*(3), 2274–2303. https://doi.org/10.1137/16M1080240