The Journal of Open Source Software

Enlsip.jl: A Julia optimization package to solve constrained nonlinear least-squares problems

Pierre Borie [1], **Alain Marcotte**[2], **Fabian Bastin** [1], **and Stéphane Dellacherie**[2,3]

1 Department of Computer Science and Operations Research, University of Montreal, Montreal, QC, Canada **2** Unit of Inflow and Load Forecasting, Hydro-Québec, Montreal, QC, Canada **3** Department of Computer Science, UQÀM, Montreal, QC, Canada

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @tmigot
- @odunbar

Submitted: 31 October 2023
Published: unpublished

Summary

Easy Nonlinear Least Squares Inequality Program (ENLSIP[1]) is the name of an optimization algorithm and an open-source Fortran77 library developed and released in 1988. It implements a nonlinear least squares under nonlinear constraints solver.

This type of problems is mathematically formulated as:

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2 \tag{1a}$$

$$\text{s.t.} \quad c_i(x) = 0, \quad 1 \le i \le q \tag{1b}$$

$$c_i(x) \ge 0, \quad q+1 \le i \le \ell, \tag{1c}$$

where the multi-functions r_i, often denoted as the residuals, and constraints c_i are two-times differentiable functions. Integers n, m, q and l are the dimensions of the problem.

The ENLISP solver incorporates a Gauss-Newton type method developed by Lindström & Wedin (1988). This method also uses an active set strategy to handle the constraints (see Nocedal & Wright, 2006, Chapter 16).

Statement of need

The ENLSIP Fortran77 library has been successfully used for decades by Hydro-Québec, the main electricity supplier for the province of Quebec in Canada, to calibrate its short-term electricity demand forecast models, which are coded in Fortran90. Since Hydro-Québec is transitioning from Fortran77 to Julia (Bezanson et al., 2017) and because its systems are used in a highly critical context, the primary goal of this transition is to ensure that the replacing Julia version reproduces the results given by the original Fortran77 version. The conversion of the above-mentioned ENLSIP library to Julia is a crucial part of this process.

Nonlinear least squares arise in a variety of model calibration scenarios. Formulation (1) is particularly relevant in contexts where additional constraints, such as those related to physical models, need to be satisfied. This is due to the high-risk nature of Hydro-Québec's forecasting operations.

Comparison of results and performance on operational Hydro-Québec optimization problems have been conducted using a Julia-Fortran interface and they have shown very good concordance results. We additionaly compared numerical results on nonlinear programming test problems

[1]The source code is available at https://plato.asu.edu/sub/nonlsq.html

(Hock & Schittkowski, 1980; Lukšan & Vlček, 1999) to ascertain whether the two versions could yield significantly disparate outcomes or distinct solutions. On the tested problems, we observed no differences in convergence behavior. Furthermore, the obtained solutions did not differ from a predetermined tolerance, the same one we previously employed to ensure the results of our Julia version were consistent with the requirements of Hydro-Québec. This has led us to consider that the current version of our implementation can be published as the Julia package `Enlsip.jl`.

From Fortran77 to Julia

Our first motivation to convert `ENLSIP` in Julia was to improve reliability, readability and ease of maintenance of the original code. Also, linear algebra tools in Julia, based on OpenBLAS, benefit from improved implementations than those of the algorithm by Lindström & Wedin (1988), based on MINPACK. Furthermore, this language is highly convenient for optimization, offering various interface tools such as `JuMP` (Dunning et al., 2017) or `NLPModels` (Orban et al., 2020), to model optimization problems. Although these libraries are not currently used in our package, they are under consideration for future developments. Finally, after conducting some comparison tests on Hydro-Québec operational context problems, we observed that performances of our Julia version were similar to, if not better than, the Fortran77 version in terms of computation time.

Other nonlinear least-squares packages

Several existing Julia packages can be used to solve nonlinear least-squares problems, such as NL2sol.jl, NLS_Solver.jl or CaNNOLeS.jl. However, they do not entirely cover the formulation stated in (1). Indeed, the first two are designed for unconstrained (Dennis Jr et al., 1981) or bound constrained problems and the last one is designed for equality constrained problems (Orban & Siqueira, 2020).

Although this algorithm may not benefit from state-of-the-art least-squares and nonlinear optimization improvements, its use remains relevant. Indeed, its application remains very general, covering nonlinearity and non-convexity of the residuals and constraints. Compared to other categories, like the unconstrained case (as discussed in Dennis Jr & Schnabel, 1996, Chapter 10), this specific class of least-squares problems with general constraints is, to the best of our knowledge, rarely addressed in the literature.

Usage

`Enlsip.jl` can be dowloaded from the Julia package manager by running the following command into the REPL:

```
using Pkg
Pkg.add("Enlsip")
```

Our package provides a basic interface for modeling optimization problems like in (1), by passing the residuals, constraints functions and dimensions of the problem. This is accomplished by creating an instance of our `CnlsModel` structure. Users can also provide functions to compute Jacobian matrices of residuals and constraints, or they can let the algorithm compute them numerically using forward differences.

We then demonstrate how to use the package by modeling and solving a small dimensions test problem (Hock & Schittkowski, 1980, problem 65).

```
using Enlsip

# Dimensions of the problem
```

```julia
n = 3 # number of parameters
m = 3 # number of residuals
l = 1 # number of nonlinear inequality constraints

# Residuals and Jacobian matrix associated
r(x::Vector) = [x[1] - x[2]; (x[1]+x[2]-10.0) / 3.0; x[3]-5.0]

jac_r(x::Vector) = [1. -1. 0; 1/3 1/3 0.; 0. 0. 1.]

# Constraints (one nonlinear inequality and box constraints)
c(x::Vector) = [48.0 - x[1]^2-x[2]^2-x[3]^2]
jac_c(x::Vector) = [ -2x[1] -2x[2] -2x[3]]
x_l = [-4.5, -4.5, -5.0]
x_u = [4.5, 4.5, 5.0]

# Starting point
x0 = [-5.0, 5.0, 0.0]

# Instantiate the  model associated with the problem
model = Enlsip.CnlsModel(r, n, m; jacobian_residuals=jac_r, starting_point=x0,
      ineq_constraints = c, jacobian_ineqcons=jac_c, nb_ineqcons = l,
      x_low=x_l, x_upp=x_u)
```

Once a model has been instantiated, the solver function can be called. If wanted by the user, some details about the iterations can be shown.

```julia
# Call of the `solve!` function
Enlsip.solve!(model)
```

Additional information on how to use the package and examples with test problems from the literature can be found in the online documentation[2].

Acknowledgements

The research has been supported by MITACS grants IT25656, IT28724, and IT36208. We would also like to mention that the release of `Enlsip.jl` results from a close collaboration between the University of Montreal and the Unit of Inflow and Load Forecasting of Hydro-Québec. The work of Fabian Bastin is supported by the Natural Sciences and Engineering Research Council of Canada [Discovery Grant 2022-04400].

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Dennis Jr, J. E., Gay, D. M., & Walsh, R. E. (1981). An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, *7*(3), 348–368. https://doi.org/10.1145/355958.355965

Dennis Jr, J. E., & Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, Philadelphia, PA, USA. https://doi.org/10.1137/1.9781611971200

Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, *59*(2), 295–320. https://doi.org/10.1137/15M1020575

[2] https://uncertainlab.github.io/Enlsip.jl

Hock, W., & Schittkowski, K. (1980). *Test examples for nonlinear programming codes* (Second edition, Vol. 187). Springer, Berlin, Heidelberg, Germany. https://doi.org/10.1007/978-3-642-48320-2

Lindström, P., & Wedin, P. Å. (1988). Gauss-Newton based algorithms for constrained nonlinear least squares problems. *Technical Report S-901 87 from the Institute of Information Processing, University of Umeå, Sweden*.

Lukšan, L., & Vlček, J. (1999). Sparse and partially separable test problems for unconstrained and equality constrained optimization. *Technical Report 767*. http://hdl.handle.net/11104/0123965

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (Second edition). Springer, New York, NY, USA. https://doi.org/10.1007/978-0-387-40065-5

Orban, D., & Siqueira, A. S. (2020). A regularization method for constrained nonlinear least squares. *Computational Optimization and Applications*, *76*(3), 961–989. https://doi.org/10.1007/s10589-020-00201-2

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