**Les Cahiers du GERAD** ISSN: 0711–2440

# `PDENLPModels.jl`: An NLPModel API for optimization problems with PDE-constraints

T. Migot, D. Orban, A.S. Siqueira

G–2022–42

September 2022



La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2022
– Bibliothèque et Archives Canada, 2022

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2022
– Library and Archives Canada, 2022

**GERAD** HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

**Tél. : 514 340-6053**
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

# PDENLPModels.jl: An NLPModel API for optimization problems with PDE-constraints


**Tangi Migot** [a]

**Dominique Orban** [a]

**Abel S. Siqueira** [b]

[a] *GERAD and Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal (Qc), Canada, H3T 2A7*

[b] *Netherlands eScience Center, Amsterdam, Netherlands*

tangi.migot@polymtl.ca
dominique.orban@gerad.ca
abel.s.siqueira@gmail.com



**September 2022**
**Les Cahiers du GERAD**
**G–2022–42**

**Abstract :** This paper presents `PDENLPModels.jl` a new Julia package for modeling and discretizing optimization problems with mixed algebraic and partial differential equations in the constraints.

**Keywords:** Julia, nonlinear optimization, nonlinear programming, PDE-constrained optimization, optimal control

**Résumé :** Cet article présente le package Julia `PDENLPModels.jl` qui permet de modéliser et discrétiser des problèmes d'optimisation avec des contraintes algébriques et des contraintes sous forme d'équations aux dérivées partielles.

**Mots clés :** Julia, optimization non-linéaire, programmation non-linéaire, optimisation avec contraintes d'EDP, contrôle optimal

**Acknowledgements:** Tangi Migot is supported by IVADO and the Canada First Research Excellence Fund / Apogée, and Dominique Orban is partially supported by an NSERC Discovery Grant.

# 1 Summary

`PDENLPModels.jl` is a Julia (Bezanson et al., 2017) package for modeling and discretizing optimization problems with mixed algebraic and partial differential equations (PDE) in the constraints. The general form of the problems over some domain $\Omega \subset \mathbb{R}^d$ is

$$\underset{y,u,\theta}{\text{minimize}} \int_\Omega J(y,u,\theta)d\Omega \quad \text{subject to} \quad \begin{array}{ll} e(y,u,\theta) = 0, & \text{(governing PDE on } \Omega) \\ l_{yu} \leq (y,u) \leq u_{yu}, & \text{(functional bound constraints)} \\ l_\theta \leq \theta \leq u_\theta, & \text{(bound constraints)} \end{array}$$

where $y : \Omega \to \mathcal{Y}$ is the state, $u : \Omega \to \mathcal{U}$ is the control, and $\theta \in \mathbb{R}^k$ are algebraic variables. $J : \mathcal{Y} \times \mathcal{U} \times \mathbb{R}^k \to \mathbb{R}$ and $e : \mathcal{Y} \times \mathcal{U} \times \mathbb{R}^k \to \mathcal{C}$ are smooth mappings. $(\mathcal{Y}, \|\cdot\|_\mathcal{Y})$, $(\mathcal{U}, \|\cdot\|_\mathcal{U})$, and $(\mathcal{C}, \|\cdot\|_\mathcal{C})$ are real Banach spaces, $l_\theta, u_\theta \in \mathbb{R}^k$ are bounds on $\theta$, and $l_{yu}, u_{yu} : \Omega \to \mathcal{Y} \times \mathcal{U}$ are functional bounds on the controls and states.

After discretization of the domain $\Omega$, the integral, and the derivatives, the resulting problem is a nonlinear optimization problem of the form

$$\underset{x \in \mathbb{R}^{N_y+N_u+N_\theta}}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0, \quad l \leq x \leq u,$$

where $l, u \in \mathbb{R}^{N_y+N_u+N_\theta}$, $f : \mathbb{R}^{N_y} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_\theta} \to \mathbb{R}$ and $c : \mathbb{R}^{N_y} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_\theta} \to \mathbb{R}^{N_y}$.

The first difficulty in modeling such a challenging problem is to access a discretization of the domain and have the possibility to evaluate derivatives of $f$ and $c$. Fortunately, several packages allow the user to define the domain, meshes, function spaces, and finite-element families to approximate unknowns and model functionals and sets of PDEs in a weak form. The main ones are `FEniCS.jl`, a wrapper for the FEniCS library (Logg et al., 2012), `Ferrite.jl` (Carlsson et al., 2021), `FinEtools.jl` (Krysl, 2021), `JuliaFEM.jl` (Aho et al., 2018, 2019), and `Gridap.jl` (Badia and Verdugo, 2020; Verdugo and Badia, 2022). In `PDENLPModels.jl`, we focus on the latter as it is exclusively written in Julia and supports a variety of discretizations and meshing possibilities. Additionally, `Gridap.jl` has an expressive API allowing to model complex PDEs with few lines of code, and to write the underlying weak form with a syntax almost one-to-one with mathematical notation.

`PDENLPModels.jl` exports the `GridapPDENLPModel` type, which uses `Gridap.jl` for the discretization of the functional spaces by finite-elements. The resulting model is an instance of an `AbstractNLPModel`, as defined in `NLPModels.jl` (Orban et al., 2020a), and provides access to objective and constraint function values, to their first and second derivatives, and to any information that a solver might request from a model. As such, `PDENLPModels.jl` offers an interface between generic PDE-constrained optimization problems and cutting-edge optimization solvers such as `Artelys Knitro` (Byrd et al., 2006) via `NLPModelsKnitro.jl` (Orban et al., 2020c), `Ipopt` (Wächter and Biegler, 2006) via `NLPModelsIpopt.jl` (Orban et al., 2020b), `DCISolver.jl` (Migot et al., 2022), `Percival.jl` (dos Santos and Siqueira, 2020), and any solver accepting an `AbstractNLPModel` as input, see JuliaSmoothOptimizers (JSO) (Migot et al., 2021).

The following example shows how to solve a Poisson control problem with Dirichlet boundary conditions using `DCISolver.jl`:

$$\underset{y,u}{\text{minimize}} \int_{(-1,1)^2} \frac{1}{2}\|y_d - y\|^2 + \frac{\alpha}{2}\|u\|^2 d\Omega \quad \text{subject to} \quad \begin{array}{ll} \Delta y - u - h = 0, & \text{on } \Omega. \\ y = 0, & \text{on } \partial\Omega, \end{array}$$

for some given functions $y_d, h : (-1,1)^2 \to \mathbb{R}$, and $\alpha > 0$.

```julia
using DCISolver, Gridap, PDENLPModels
Ω = (-1, 1, -1, 1) # Cartesian discretization of Ω=(-1,1)^2 in 100^2 squares.
model = CartesianDiscreteModel(Ω, (100, 100))
fe_y = ReferenceFE(lagrangian, Float64, 2) # Finite-elements for the state
Xpde = TestFESpace(model, fe_y; dirichlet_tags = "boundary")
Ypde = TrialFESpace(Xpde, x -> 0.0) # y is 0 over ∂Ω
fe_u = ReferenceFE(lagrangian, Float64, 1) # Finite-elements for the control
Xcon = TestFESpace(model, fe_u)
Ycon = TrialFESpace(Xcon)
dΩ = Measure(Triangulation(model), 1) # Gridap's integration machinery
# Define the objective function f
yd(x) = -x[1]^2
f(y, u) = ∫(0.5 * (yd - y) * (yd - y) + 0.5 * 1e-2 * u * u) * dΩ
# Define the constraint operator in weak form
h(x) = -sin(7π / 8 * x[1]) * sin(7π / 8 * x[2])
c(y, u, v) = ∫(∇(v) ⊙ ∇(y) - v * u - v * h) * dΩ
# Define an initial guess for the discretized problem
x0 = zeros(num_free_dofs(Ypde) + num_free_dofs(Ycon))
# Build a GridapPDENLPModel, which implements the NLPModel API.
name = "Control elastic membrane"
nlp = GridapPDENLPModel(x0, f, dΩ, Ypde, Ycon, Xpde, Xcon, c, name = name)
dci(nlp, verbose = 1) # solve the problem with DCI
```

## 2 Statement of need

For PDEs, there are five main ways to discretize functions and their derivatives:

- Finite-difference methods: functions are represented on a grid, e.g., `DiffEqOperators.jl` (Rackauckas and Nie, 2017) or Trixi.jl (Schlottke-Lakemper et al., 2020);
- Finite-volume methods: functions are represented by a discretization of their integral;
- Spectral methods: functions are expanded in a global basis, e.g., `FFTW.jl` (Frigo and Johnson, 2005) and `ApproxFun.jl` (Olver and Townsend, 2014);
- Physics-informed neural networks: functions are represented by neural networks, e.g., `NeuralPDE.jl` (Zubov et al., 2021);
- Finite-element methods: functions are expanded in a local basis.

With finite-elements discretization, it is easy to increase the order of the elements or locally refine the mesh so that the physical fields can be approximated accurately. Another advantage is that you can straightforwardly combine different kinds of approximation functions, leading to mixed formulations. Finally, curved or irregular geometries of the domain are handled in a natural way.

Outside of Julia, there exist libraries handling finite-elements methods such as `deal.II` (Bangerth et al., 2007), `FEniCS` (Logg et al., 2012), `PETSc` (Balay et al., 2021), and `FreeFEM++` (Hecht, 2012). There exists a Julia wrapper to `FEniCS` (Rackauckas and Nie, 2017) and `PETSc` (Crean et al., 2021). However, interfaces to low-level libraries have limitations that pure Julia implementations do not have, including the ability to generate models with various arithmetic types.

Julia's JIT compiler is attractive for the design of efficient scientific computing software, and, in particular, mathematical optimization (Lubin and Dunning, 2015), and has become a natural choice for developing new modeling tools. There are other packages available in Julia for optimization problems with PDE in the constraints. `jInv.jl` (Ruthotto et al., 2017) and `ADCME.jl` (Xu and Darve, 2020) focus on inverse problems. `DifferentialEquations.jl` (Rackauckas and Nie, 2017) is a suite for numerically solving differential equations written in Julia, which includes features for parameter estimation and Bayesian analysis. `InfiniteOpt.jl` (Pulsipher et al., 2022) provides a general mathematical abstraction to express and solve infinite-dimensional optimization problems including with PDEs in the constraints handled by finite-differences. `TopOpt.jl` (Huang and Tarek, 2021) is a package for topology optimization. However, to the best of our knowledge, there are no packages with the generality of `PDENLPModels.jl`.

Optimization problems with PDEs in the constraints have been in the spotlight in recent years as challenging and highly structured. The great divide between optimization libraries and PDE libraries makes it difficult for optimization research to benefit from testing on a large base of PDE-constrained problems and PDE libraries to benefit from the latest advances in optimization. `PDENLPModels.jl` fills this gap by providing generic discretized models that can be solved by any solver from JuliaSmoothOptimizers.

## References


Jukka Aho, Tero Frondelius, Ovainola, Arilaakk, Tony Kelman, Vlad Stoian, The Gitter Badger, and Marja Rapo. Juliafem/juliafem.jl: Julia v1 compatible release, 2018. URL https://github.com/JuliaFEM/JuliaFEM.jl.

Jukka Aho, Antti-Jussi Vuotikka, and Tero Frondelius. Introduction to juliafem an open-source fem solver. Rakenteiden Mekaniikka, 52:148–159, 09 2019. doi: 10.23998/rm.75103.

Santiago Badia and Francesc Verdugo. Gridap: An extensible finite element toolbox in Julia. Journal of Open Source Software, 5(52):2520, 2020. doi: 10.21105/joss.02520.

Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.15, Argonne National Laboratory, 2021. URL https://www.mcs.anl.gov/petsc.

Wolfgang Bangerth, Ralf Hartmann, and Guido Kanschat. deal. ii—a general-purpose object-oriented finite element library. ACM Transactions on Mathematical Software (TOMS), 33(4):24–es, 2007. doi: 10.1145/1268776.1268779. URL https://dealii.org/.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. SIAM Review, 59(1):65–98, 2017. doi: 10.1137/141000671.

Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. Knitro: An integrated package for nonlinear optimization, pages 35–59. Springer US, Boston, MA, 2006. doi: 10.1007/0-387-30065-1_4.

Kristoffer Carlsson, Fredrik Ekre, and Contributors. Ferrite.jl, 3 2021. URL https://github.com/Ferrite-FEM/Ferrite.jl.

Jared Crean, Jeremy E. Kozdon, Katharine Hyatt, and Contributors. PETSc.jl, 2021. URL https://github.com/JuliaParallel/PETSc.jl.

Egmara Antunes dos Santos and Abel Soares Siqueira. Percival.jl: an augmented lagrangian method, July 2020. URL https://github.com/JuliaSmoothOptimizers/Percival.jl.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. Proceedings of the IEEE, 93 (2):216–231, 2005. doi: 10.1109/JPROC.2004.840301. Special issue on “Program Generation, Optimization, and Platform Adaptation”.

Frédéric Hecht. New development in freefem++. Journal of numerical mathematics, 20(3-4):251–266, 2012. doi: 10.1515/jnum-2012-0013. URL https://wiki2.org/en/FreeFem%2B%2B.

Yijiang Huang and Mohamed Tarek. Topopt.jl: Truss and continuum topology optimization, interactive visualization, automatic differentiation and more. In Proceedings of the 14th World Congress of Structural and Multidisciplinary Optimization, 2021. URL https://github.com/JuliaTopOpt/TopOpt.jl.

Petr Krysl. FinEtools.jl, 12 2021. URL https://github.com/PetrKryslUCSD/FinEtools.jl.

Anders Logg, Kent-Andre Mardal, and Garth Wells. Automated solution of differential equations by the finite element method: The FEniCS book, volume 84. Springer Science & Business Media, 2012. doi: 10.1007/978-3-642-23099-8. URL https://fenicsproject.org/.

Miles Lubin and Iain Dunning. Computing in operations research using Julia. INFORMS Journal on Computing, 27(2):238–248, 2015. doi: 10.1287/ijoc.2014.0623.

T. Migot, D. Orban, and A. S. Siqueira. The JuliaSmoothOptimizers ecosystem for linear and nonlinear optimization, 2021. URL https://juliasmoothoptimizers.github.io/.

T. Migot, Dominique Orban, and Abel Soares Siqueira. Dcisolver.jl: A Julia solver for nonlinear optimization using dynamic control of infeasibility. Journal of Open Source Software, 7(70):3991, 2022. doi: 10.21105/joss.03991.

Sheehan Olver and Alex Townsend. A practical framework for infinite-dimensional linear algebra. In Proceedings of the 1st Workshop for High Performance Technical Computing in Dynamic Languages – HPTCDL '14. IEEE, 2014. doi: 10.1109/HPTCDL.2014.10.

D. Orban, A. S. Siqueira, and contributors. NLPModels.jl: Data structures for optimization models, July 2020a. URL https://github.com/JuliaSmoothOptimizers/NLPModels.jl.

D. Orban, A. S. Siqueira, and contributors. NLPModelsIpopt.jl: A thin IPOPT wrapper for NLPModels, July 2020b. URL https://github.com/JuliaSmoothOptimizers/NLPModelsIpopt.jl.

D. Orban, A. S. Siqueira, and contributors. NLPModelsKnitro.jl: A thin KNITRO wrapper for NLPModels, July 2020c. URL https://github.com/JuliaSmoothOptimizers/NLPModelsKnitro.jl.

Joshua L. Pulsipher, Weiqi Zhang, Tyler J. Hongisto, and Victor M. Zavala. A unifying modeling abstraction for infinite-dimensional optimization. Computers & Chemical Engineering, 156, 2022. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2021.107567. URL https://www.sciencedirect.com/science/article/pii/S0098135421003458.

Christopher Rackauckas and Qing Nie. DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software, 5(1), 2017. doi: 10.5334/jors.151.

Lars Ruthotto, Eran Treister, and Eldad Haber. jInv–a flexible Julia package for PDE parameter estimation. SIAM Journal on Scientific Computing, 39(5):S702–S722, 2017. doi: 10.1137/16M1081063.

Michael Schlottke-Lakemper, Gregor J Gassner, Hendrik Ranocha, and Andrew R Winters. Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia, 08 2020.

Francesc Verdugo and Santiago Badia. The software design of Gridap: a finite element package based on the Julia JIT compiler. Computer Physics Communications, page 108341, 2022. doi: 10.1016/j.cpc.2022.108341.

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106(1):25–57, 2006. doi: 10.1007/s10107-004-0559-y.

Kailai Xu and Eric Darve. Adcme: Learning spatially-varying physical fields using deep neural networks. arXiv preprint arXiv:2011.11955, 2020. URL https://github.com/kailaix/ADCME.jl.

Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, et al. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations. arXiv preprint arXiv:2107.09443, 2021. URL https://github.com/SciML/NeuralPDE.jl.