Les Cahiers du GERAD ISSN: 0711–2440

DCISolver.jl: A Julia solver for nonlinear optimization using dynamic control of infeasibility

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

G–2021–67

December 2021

La collection *Les Cahiers du GERAD* est constituée des travaux de recherche menés par nos membres. La plupart de ces documents de travail a été soumis à des revues avec comité de révision. Lorsqu'un document est accepté et publié, le pdf original est retiré si c'est nécessaire et un lien vers l'article publié est ajouté.

Citation suggérée : T. Migot, D. Orban, A.S. Siqueira (Décembre 2021). DCISolver.jl: A Julia solver for nonlinear optimization using dynamic control of infeasibility, Rapport technique, Les Cahiers du GERAD G– 2021–67, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2021-67) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

The series *Les Cahiers du GERAD* consists of working papers carried out by our members. Most of these pre-prints have been submitted to peer-reviewed journals. When accepted and published, if necessary, the original pdf is removed and a link to the published article is added.

Suggested citation: T. Migot, D. Orban, A.S. Siqueira (December 2021). DCISolver.jl: A Julia solver for nonlinear optimization using dynamic control of infeasibility, Technical report, Les Cahiers du GERAD G–2021–67, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (https://www.gerad.ca/en/papers/G-2021-67) to update your reference data, if it has been published in a scientific journal.

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, 2021
– Bibliothèque et Archives Canada, 2021

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, 2021
– Library and Archives Canada, 2021

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

DCISolver.jl: A Julia solver for nonlinear optimization using dynamic control of infeasibility

Tangi Migot [a]

Dominique Orban [a]

Abel S. Siqueira [b]

[a] *GERAD & 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

December 2021
Les Cahiers du GERAD
G–2021–67

Copyright © 2021 GERAD, Migot, Orban, Siqueira

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs. Les auteurs conservent leur droit d'auteur et leurs droits moraux sur leurs publications et les utilisateurs s'engagent à reconnaître et respecter les exigences légales associées à ces droits. Ainsi, les utilisateurs:

- Peuvent télécharger et imprimer une copie de toute publication du portail public aux fins d'étude ou de recherche privée;
- Ne peuvent pas distribuer le matériel ou l'utiliser pour une activité à but lucratif ou pour un gain commercial;
- Peuvent distribuer gratuitement l'URL identifiant la publication.

Si vous pensez que ce document enfreint le droit d'auteur, contactez-nous en fournissant des détails. Nous supprimerons immédiatement l'accès au travail et enquêterons sur votre demande.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*. Copyright and moral rights for the publications are retained by the authors and the users must commit themselves to recognize and abide the legal requirements associated with these rights. Thus, users:

- May download and print one copy of any publication from the public portal for the purpose of private study or research;
- May not further distribute the material or use it for any profit-making activity or commercial gain;
- May freely distribute the URL identifying the publication.

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Abstract : This paper presents `DCISolver.jl` a new Julia package implementating the Dynamic Control of Infeasibility method (DCI), introduced by Bielschowsky & Gomes (2008), for solving equality-constrained nonlinear optimization problems.

Keywords : Julia, numerical optimization, nonlinear optimization, nonlinear programming

Résumé : Cet article présente le package Julia `DCISolver.jl` qui implémente la méthode *Dynamic Control of Infeasibility*, introduite par Bielschowsky & Gomes (2008), pour les problèmes d'optimisation non-linéaire avec contraintes d'égalités.

Mots clés : Julia, optimisation numérique, optimization non-linéaire, programmation non-linéaire

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

`DCISolver.jl` is a new Julia implementation of the Dynamic Control of Infeasibility method (DCI), introduced by Bielschowsky & Gomes [2], for solving the equality-constrained nonlinear optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable. DCI is an iterative method that aims to compute a local minimum of (1) using first and second-order derivatives. Our initial motivation for developing `DCISolver.jl` is to solve PDE-constrained optimization problems, many of which have equality constraints only.

Each DCI iteration is a two-step process. A tangential step first approximately minimizes a quadratic model subject to linearized constraints within a trust region. A normal step then recenters feasibility by way of a trust cylinder, which is the set of points such that $\|h(x)\| \leq \rho$, where $\rho > 0$. The idea of trust cylinders is to control infeasibility, contrary to penalty methods, which encourage feasibility by penalizing infeasibility. Each time the trust cylinder is violated during the tangential step, the normal step brings infeasibility back within prescribed limits. The radius ρ of the trust cylinder decreases with the iterations, so a feasible and optimal point results in the limit. For details and theoretical convergence, we refer the reader to the original paper [2].

`DCISolver.jl` is built upon the JuliaSmoothOptimizers (JSO) tools [11]. JSO is an academic organization containing a collection of Julia packages for nonlinear optimization software development, testing, and benchmarking. It provides tools for building models, accessing problems repositories, and solving subproblems. `DCISolver.jl` takes as input an `AbstractNLPModel`, JSO's general model API defined in `NLPModels.jl` [17], a flexible data type to evaluate objective and constraints, their derivatives, and to provide any information that a solver might request from a model. The user can hand-code derivatives, use automatic differentiation, or use JSO-interfaces to classical mathematical optimization modeling languages such as AMPL [8], CUTEst [9], or JuMP [7].

Internally, `DCISolver.jl` combines cutting-edge numerical linear algebra solvers. The normal step relies heavily on iterative methods for linear algebra from `Krylov.jl` [14], which provides more than 25 implementations of standard and novel Krylov methods, and they all can be used with Nvidia GPU via CUDA.jl [1]. The tangential step is computed using the sparse factorization of a symmetric and quasi-definite matrix via `LDLFactorizations.jl` [15], or the well-known Fortran code `MA57` [6] from the HSL (2007), via `HSL.jl` [16].

One of the significant advantages of our implementation is that the normal step is factorization-free, i.e., it uses second-order information via Hessian-vector products but does not need access to the Hessian as an explicit matrix. This makes `DCISolver.jl` a valuable asset for large-scale problems, for instance to solve PDE-constrained optimization problems [12]. In the current implementation, the tangential step requires the explicit hessian, but removing that restriction is the subject of ongoing research, as is the treatment of inequality constraints.

2 Statement of need

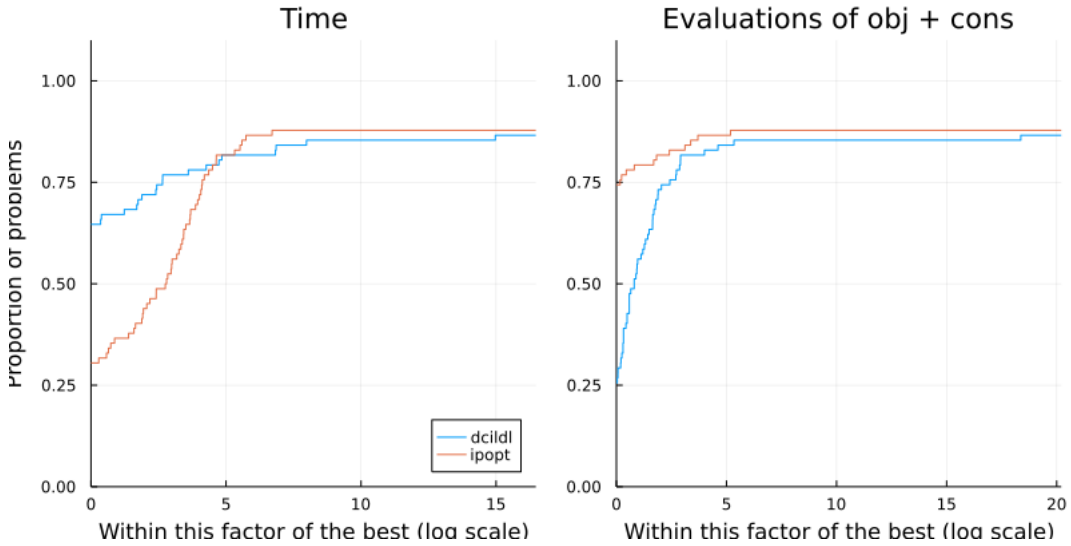
Julia's JIT compiler is attractive for the design of efficient scientific computing software, and, in particular, mathematical optimization [10], and has become a natural choice for developing new solvers.

There already exist ways to solve (1) in Julia. If (1) is amenable to being modeled in `JuMP` [7], the model may be passed to state-of-the-art solvers, implemented in low-level compiled languages, via wrappers thanks to Julia's native interoperability with such languages. However, interfaces to low-level languages have limitations that pure Julia implementations do not have, including the ability to apply solvers with various arithmetic types. `Optim.jl` [13] implements a factorization-based pure

Julia primal-dual interior-point method for problems with both equality and inequality constraints modeled after Artlelys Knitro [3] and Ipopt [21]. `Percival.jl` [5] is a factorization-free pure Julia implementation of an augmented Lagrangian method for problems with both equality and inequality constraints based on bound-constrained subproblems.

To the best of our knowledge, there is no available maintained open-source implementation of DCI in existence. The original authors did not make their implementation public, and the other known implementation is `dcicpp` [20], extending the original method to inequalities in the Ph.D. thesis by Siqueira [19], and it has had no updates in the last 5 years. Hence, we offer an interesting alternative to augmented Lagrangian and interior-point methods in the form of an evolving, research level yet stable solver.

`DCISolver.jl` can solve large-scale problems and can be benchmarked easily against other JSO-compliant solvers using `SolverBenchmark.jl` [18]. We include below performance profiles [4] of `DCISolver.jl` against Ipopt on 82 problems from CUTEst [9] with up to 10 000 variables and 10 000 constraints. Ipopt solved 72 problems (88%) successfully, which is one more than DCI. Without explaining performance profiles in full detail, the plot on the left shows that Ipopt is the fastest on 20 of the problems (28%), while DCI is the fastest on 51 of the problems (72%) among the 71 problems solved by both solvers. The plot on the right shows that Ipopt used fewer evaluations of objective and constraint functions on 50 of the problems (70%), DCI used fewer evaluations on 17 of the problems (24%), and there was a tie in the number of evaluations on 4 problems (6%). Overall, this performance profile is very encouraging for such a young implementation. The package's documentation includes more extensive benchmarks on classical test sets showing that `DCISolver.jl` is also competitive with Artelys Knitro.



Références

[1] Tim Besard, Christophe Foket, and Bjorn De Sutter. Effective extensible programming : Unleashing Julia on GPUs. IEEE Transactions on Parallel and Distributed Systems, 2018.

[2] Roberto H. Bielschowsky and Francisco A.M. Gomes. Dynamic control of infeasibility in equality constrained optimization. SIAM Journal on Optimization, 19(3) :1299–1325, 2008.

[3] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. Knitro : An integrated package for nonlinear optimization. In Large-scale nonlinear optimization, pages 35–59. Springer, 2006.

[4] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. Mathematical programming, 91(2) :201–213, 2002.

[5] Egmara Antunes dos Santos and Abel Soares Siqueira. Percival.jl : an augmented lagrangian method, July 2020.

[6] I. S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. ACM Transactions on Mathematical Software, 30(2) :118–144, 2004.

[7] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP : A modeling language for mathematical optimization. SIAM Review, 59(2) :295–320, 2017.

[8] Robert Fourer, David M. Gay, and Brian W. Kernighan. AMPL. A modeling language for mathematical programming. 2nd ed., Brooks/Cole, Pacific Grove, CA, 2003.

[9] Nicholas I. Gould, Dominique Orban, and Philippe L. Toint. CUTEst : A constrained and unconstrained testing environment with safe threads for mathematical optimization. Computational Optimization and Applications, 60(3) :545–557, 2015.

[10] Miles Lubin and Iain Dunning. Computing in operations research using Julia. INFORMS Journal on Computing, 27(2) :238–248, 2015.

[11] T. Migot, D. Orban, and A. S. Siqueira. The JuliaSmoothOptimizers ecosystem for linear and nonlinear optimization, 2021.

[12] T. Migot, D. Orban, and A. S. Siqueira. PDENLPModels.jl : A nlpmodel api for optimization problems with pde-constraints, July 2021.

[13] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. Optim : A mathematical optimization package for Julia. Journal of Open Source Software, 3(24) :615, 2018.

[14] A. Montoison, D. Orban, and contributors. Krylov.jl : A Julia basket of hand-picked Krylov methods, June 2020.

[15] D. Orban and contributors. LDLFactorizations.jl : Factorization of symmetric matrices, June 2020.

[16] D. Orban and contributors. HSL.jl : A julia interface to the HSL mathematical software library, March 2021.

[17] D. Orban, A. S. Siqueira, and contributors. NLPModels.jl : Data structures for optimization models, July 2020.

[18] D. Orban, A. S. Siqueira, and contributors. SolverBenchmark.jl : Benchmark tools for solvers, December 2020.

[19] Abel Soares Siqueira. Controle dinâmico de infactibilidade para programação não linear. PhD thesis, Universidade Estadual de Campinas, 2013.

[20] Abel Soares Siqueira. dcicpp : Dynamic control of infeasibility, 2016.

[21] 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.