ExaModels and MadNLP: Open-Source Software Infrastructure for Accelerated Nonlinear Optimization on GPUs

Sungho Shin[†], Francois Pacaud[‡], and Mihai Anitescu[†]

[†]Mathematics and Computer Science Division, Argonne National Laboratory
[‡]Centre Automatique et Systèmes, Mines Paris - PSL, Paris, France

We are pleased to submit our nominations for the COIN-OR cup, highlighting our contributions of ExaModels and MadNLP in the domain of computational infrastructure for operations research, in the particular area of nonlinear optimization. Our open-source software packages represent a significant advance in addressing the challenges of efficiently solving large-scale nonlinear optimziation problems by harnessing the capabilities of modern GPU hardware. In this document, we highlight the key challenges and opportunities we are facing for the development of optimization software in 2023, especially in the context of the advent of accelerated computing. We also explain how ExaModels and MadNLP jointly address the key technical challanges and harness the benefits that modern accelerated computing has to offer. We also highlight several performance improvement we have achieved in terms of solving large-scale, real-world nonolinear optimziation problems. Finally, we conclude by discussing how our development sets the stage for the next-generation nonlinear optimization solver development and opens up new possibilities in various applications.

Challenges and Significance The computational landscape of today presents us with immense challenges and potential opportunities, especially with Graphics Processing Units (GPUs) at our disposal. Notably, NVIDIA GPUs have enabled great stride in the scalabiltiy of deep learning and achieved remarkable success in building large-scale AI models. Also, in the public doamin, most of the computational power for the next generation exascale HPC systems, such as Frontier and Aurora, come from GPU accelerators. While the GPUs have enabled great success in different scientific computing areas, such as machine learning and high-fidelity simulations, utilizing GPUs within nonlinear programming (NLP) algorithms has been hindered by challenges like sparse automatic differentiation (AD) and sparse linear solver routines, which do not seamlessly translate to GPU architecture. The key issue is that the conventional sparse AD and matrix factorization algoirthms are built for CPUs, and they heavily rely on the serial computation.

While GPU computation can trivially accelerate several parts of the optimization process—especially various internal computations within the optimization solver—the sluggish data transfer between host and device memory hampers the ad-hoc implementation of GPU accelerations (Fig. 1). To fully leverage the potential offered by modern GPU hardware, it becomes imperative to have a comprehensive computational framework for optimization on GPUs. That is, we need an AD/algebraic modeling framework, sparse linear solvers, and NLP solvers that can operate entirely on the GPU. Specifically, for the best performance, both the problem data and the solver's intermediate computational data must be exclusively resident within the device memory, with the majority of operations executed on the GPU. This necessitates the development of *a comprehensive nonlinear optimization solution framework*, which performs all the necessary computations—automatic differentiation, nonlinear optimization, and linear algebra—exclusively on GPUs. ExaModels and MadNLP rise to these challenges, carving out a pathway to fully harness the power of GPUs for operaitons research and nonlinear optimization.

Developed Software Tools We have developed a comprehesnive nonlinear optimization framework by implementing our algebraic modeling/automatic differentiation tool ExaModels and nonlinear optimization solver MadNLP, while linear algebra computation is currently performed by the external cuSOLVER library.

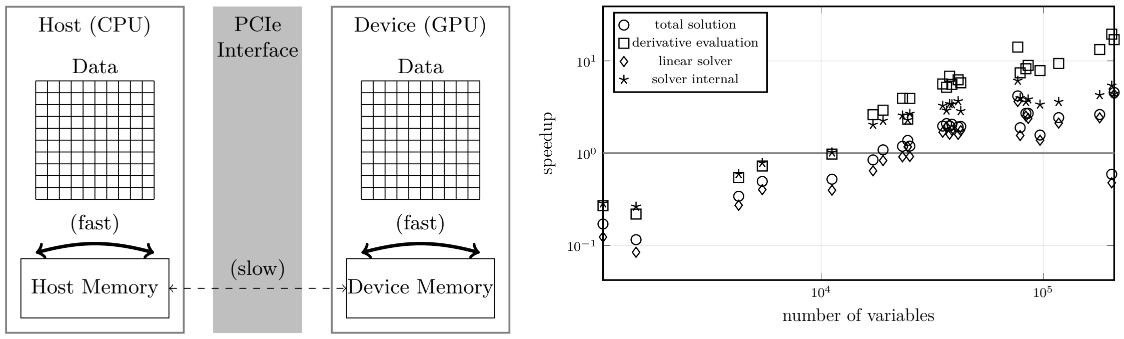

Figure 1: A schematic of host (CPU) and device (GPU) memory structure (left) and speedup achieved for AC optimal power flow problems (right).

ExaModels: SIMD Abstraction for NLPs: ExaModels pioneers the implementation of a groundbreaking single-instruction, multiple-data (SIMD) abstraction for NLPs. This revolutionary approach unlocks the potential for efficient parallel automatic differentiation on GPUs. The essence of this innovation lies in preserving the parallelizable structure within model equations, transforming derivative evaluations into streamlined operations. By enabling efficient computation of derivatives on GPUs, ExaModels offers an exceptional performance boost for power systems analysis.

The optimization models are most of the times implemented by the researchers and practitioners who do not necessarily understand the operations ExaModels provides a new platform for scalable nonlinear optimization. Our numerical results suggest that the conventional algebraic modeling systems are limited in terms of efficiently computing the derivates of the model equations (Table), and it does not allow performing the operations on GPUs (Table 2). ExaModels provide a framework that allows the user to conveniently inform the AD backend the parallelizable structure. Writing the models in this way will enable the GPU compatibility

MadNLP: Condensed-Space IPM with Inequality Relaxation: MadNLP introduces an equally transformative approach through its condensed-space interior-point method (IPM) with an inequality relaxation strategy. This strategy deftly overcomes the challenges posed by sparse matrix factorization on GPUs. The relaxation of equality constraints and condensation of the Karush-Kuhn-Tucker (KKT) system establishes positive definiteness, paving the way for the utilization of highly efficient linear solvers. MadNLP thus unveils a remarkable solution to traditionally intractable problems in GPU-based optimization.

While MadNLP has started as a port of Ipopt on Julia Language, now marking as the fourth year since its initial development, MadNLP has become a lot more versatile solver than the previously developed nonlinear optimization solvers. The key features include (1) being able to solve dense optimzation problems, (2) ability to handle different forms of KKT systems, (3) being able to handle diverse array data types (most notably, device arrays), and (4) exploit various Hessian approximation strategies (BFGS variants and limited-memory BFGS methods).

Performance Highlights The significance of these contributions is not merely theoretical. Empirical results speak volumes about the impact of ExaModels and MadNLP. When applied to AC OPF problems, the performance enhancements achieved are nothing short of extraordinary. The GPU-based solutions attain speedups of over 20 times compared to CPU-based counterparts, showcasing the profound potential of these software packages in power systems analysis. Their prowess surpasses even established tools interfaced with CPUs, marking a paradigm shift in computational efficiency.

[?]

Portability While NVIDIA seems to be most capable in computing power and mature in terms of the implementation of the low-level operations (especially, the CUDA libraries), we aim to support all the other

Case	nvars	ncons	**MadNLP + ExaModels (NVIDIA GPU)**				**MadNLP + ExaModels (CPU)**				**Ipopt + AMPL (CPU)**			**Ipopt + JuMP (CPU)**		
			iter	deriv.†	lin.†	total†	iter	deriv.†	lin.†	total†	iter	deriv.‡	total‡	iter	deriv.‡	total‡
4619_goc	42.5k	66.3k	54	0.06	2.59	2.97	46	0.32	4.54	5.78	48	5.49	11.02	46	10.04	15.48
10000_goc	76.8k	112.4k	56	0.06	2.63	3.11	77	0.89	9.54	13.00	74	14.02	24.18	74	25.13	36.46
8387_pegase	78.7k	118.7k	64	0.12	6.08	6.87	70	0.89	9.44	12.96	69	14.23	23.55	69	26.40	36.74
9591_goc	83.6k	130.6k	69	0.11	6.84	7.70	65	0.92	17.20	20.82	64	14.96	35.70	62	28.71	49.75
9241_pegase	85.6k	130.8k	60	0.10	4.35	5.15	63	0.89	10.34	13.91	61	14.09	24.33	61	25.98	37.19
10480_goc	96.8k	150.9k	70	0.13	13.19	14.26	66	1.05	18.19	22.40	64	16.93	38.04	63	33.53	56.04
13659_pegase	117.4k	170.6k	63	0.12	6.10	7.15	58	1.08	12.91	17.35	64	19.70	35.66	64	35.45	52.99
19402_goc	179.6k	281.7k	79	0.17	21.47	23.28	70	2.25	51.82	61.06	70	36.50	95.34	70	68.12	127.29
24464_goc	203.4k	313.6k	63	0.11	69.32	70.63	58	2.22	33.03	41.71	58	33.50	70.15	58	62.04	102.17
30000_goc	208.6k	307.8k	162	0.33	18.42	22.05	136	5.68	80.01	100.25	180	101.98	249.81	126	135.11	209.45

†Wall time (sec) measured by Julia. ‡CPU time (sec) reported by Ipopt.

Table 1: Numerical Performance of ExaModels and MadNLP

		CPU (single)	CPU (multi)	NVIDIA	AMD	Intel	Apple
Modeling Platforms	**AMPL**	✓	✗	✗	✗	✗	✗
	JuMP	✗	✓	✗	✗	✗	✗
	ExaModels	✓	✓	✓	✓	✓	✗
Solvers	**Ipopt**	✓	✗	✗	✗	✗	✗
	MadNLP	✓	✗	✓	✗	✗	✗

Table 2: The current status on the portability of the nonlinear optimization frameworks.

alternative architectures. This is because quite a few next-generaiton exascale high-performance computing architectures will be based on AMD and Intel accelerator devices. Also, as an open-source software, we aim to be the most generic software tool, not the most efficient one. By supporting diverse platforms, we hope to spark the attention of the chip makers in the operations research area and encourage better support for the software infrastructure for implmenting operations research software.

Table 2 summarizes the current state of the portability of our tools and other existing open-source software tools. Notably, ExaModels.jl already support all major architectures, whereas MadNLP.jl only support NVIDIA GPUs. Nonetheless, we have clear path forward in supporting all the major accelerator architectures, including NVIDIA, AMD, and Intel GPUs, while Apple GPUs are somewhat limited due to their inability to perform double precision operations. This is due to the fact that all of our GPU kernels are implemented via KernelAbstractions.jl, a julia package enabling portable implementation of GPU kernels. In this way, the GPU kernels for diverse accelerator architectures can be run based on a sam source code. Using this strategy, recently ExaModels.jl has demonstrated its capabiltiy to run on 4 different accelerator architectures (multi-thread CPUs and NVIDIA, AMD, and Intel GPUs). In the coming years, we plan to exand MadNLP's support for various accelerator architectures.

However, at the same time, it is absolutely crucial to have efficient and reliable, and preferrably also portable sparse linear sovler routines. Currently, we rely on CUSOLVER library for solving the linear systems. While vendor-implmented libraries are expected to have the best performance via high level of optimization, it would be beneficial for the community to have the open-source portable linear solvers. By showcasing the capability of solving high-stake nonlinear optimziation problems on GPUs, we can attract the attention of the numerical linear algebra community so that we can encourage the development of capable open-source sparse linear solvers. 1

Impacts in Various Applications *Energy Infrastructures.* The resillience We have recently showcased the capability of MadNLP with multiple GPUs for solving extremely-large scale security-constrained AC OPF problems.

1

Machine Learning Surrogate Models.

Closing Remarks ExaModels and MadNLP embody the spirit of innovation and progress that the COIN-OR cup seeks to recognize. By addressing critical challenges and introducing transformative strategies, these packages extend the frontiers of computational infrastructure for operations research. We are confident that their exceptional contributions will resonate within the operations research community and beyond, redefining the future of power systems analysis and optimization. It is with great enthusiasm that we nominate ExaModels and MadNLP for the COIN-OR cup, recognizing their exceptional impact on the field.