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 delighted to submit our nominations for the COIN-OR Cup, highlighting our contributions of **ExaModels** and **MadNLP** in the field of computational infrastructure for operations research, specifically in the area of nonlinear programming (NLP). Our open-source software packages represent a significant breakthrough in efficiently solving large-scale NLPs by leveraging the capabilities of modern GPU hardware. In this document, we address the primary challenges and opportunities for optimization software development in 2023, particularly within the context of accelerated computing. We elucidate how ExaModels and MadNLP collectively address technical obstacles and capitalize on the advantages offered by modern accelerated computing. Furthermore, we emphasize the remarkable performance enhancements achieved in solving real-world nonlinear optimization problems at scale. A noteworthy example of such progress is our recent accomplishment in accelerating alternating current (AC) optimal power flow (OPF) problems by up to ten times compared to state-of-the-art tools. Finally, we discuss how our development lays the foundation for the advancement in various topics in operations research and unlocks new possibilities.

Challenges and Opportunities In recent years, the scalability of deep learning has experienced significant advancements, primarily due to the utilization of NVIDIA GPUs, which have demonstrated remarkable success in constructing large-scale AI models. Additionally, in the public domain, GPU accelerators have provided a substantial portion of the computational power for next-generation leadership HPC systems such as Frontier and Aurora, achieving the milestone of one exaflop per second. While GPUs have proven highly effective in various scientific computing areas like machine learning and high-fidelity simulations, integrating them into constrained nonlinear programming (NLP) algorithms has encountered considerable challenges. Specifically, the seamless translation of sparse automatic differentiation (AD) and sparse linear solver routines to GPU architecture has been recognized to be particularly challenging. This is because conventional sparse AD and matrix factorization algorithms are designed for CPUs and heavily rely on serial computation, making them unsuitable for the single-instruction, multiple-data (SIMD) parallelism employed by GPU architectures (see Figure , left).

While GPU computation can accelerate several parts of the optimization process straightforwardly, the slow data transfer between host and device memory hinders the ad-hoc implementation of GPU accelerations (Figure , right). To fully exploit the potential offered by modern GPU hardware, it is crucial to develop a comprehensive computational framework for optimization on GPUs. This requires keeping the problem data and the solver's intermediate computational data reside exclusively within device memory, with the majority of operations executed on the GPU. Consequently, the development of a comprehensive NLP solution framework that performs all necessary computations, including AD, optimization, and linear algebra, exclusively on GPUs becomes imperative. To address these challenges, ExaModels and MadNLP offer a comprehensive solution framework for NLP. They enable AD and NLP solution exclusively on GPUs, while linear algebra computation is performed by the external library (e.g., cuSOLVER), allowing for the full utilization of GPUs.

ExaModels ExaModels is an algebraic modeling and AD system embedded in the Julia Language, specialized for the SIMD abstraction of nonlinear programs. This system is designed based on the repetitive

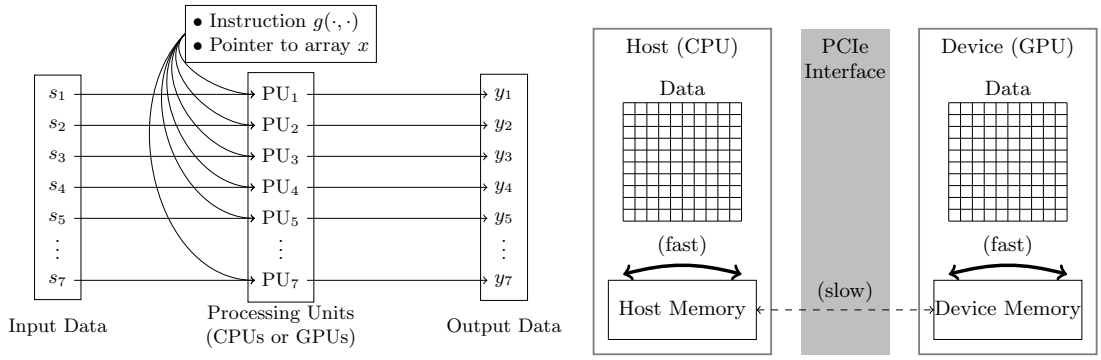

Figure 1: Schematics of host and device memory architectures (left) and SIMD parallelism (right).

structure commonly found in large-scale NLPs. For example, AC OPF problems, which may have millions of variables and constraints, can be expressed using just 15 computational patterns. By enforcing the creation of models through specifying repetitive patterns, ExaModels preserves the parallelizable structure within model equations and facilitates parallel automatic differentiation. Writing models in ExaModels automatically enables GPU compatibility, which is especially beneficial as optimization models are often implemented by domain experts who may not fully understand the internals of algebraic modeling systems and NLP solvers. This makes ExaModels a powerful modeling platform for scalable computations.

By enabling efficient computation of derivatives on GPUs, ExaModels provides exceptional performance, surpassing existing tools like AMPL (AD backend of Pyomo) and JuMP by more than two orders of magnitude. Our numerical results demonstrate that conventional algebraic modeling systems struggle to efficiently compute derivatives of model equations (Table), and they are incompatible with GPU operations (Table **??**). In contrast, ExaModels provides a convenient framework for users to inform the AD backend about the parallelizable structure, and in turn, efficient, parallelized AD can be performed in the backend.

MadNLP MadNLP is a versatile NLP solver capable of handling various problem data structures. By leveraging the modularity provided by the key features of the Julia Language, such as multiple dispatch and just-in-time compilation, MadNLP offers great flexibility in handling different types of data structures while retaining the sophistication of the algorithm. Importantly, with this approach, we can avoid re-implementing interior point solver every time for each data types, which in turn ensures that the operations performed are mathematically equivalent to those in the mature, extensively tested existing code base. MadNLP has started as a port of Ipopt (an open-source nonlinear optimziation solver running on CPUs) in the Julia Language, but over the past four years, it has expanded beyond its initial scope and now offers a range of capabilities not supported by state-of-the-art NLP solvers. These include solving dense optimization problems, handling different numerical precisions, working with different forms of KKT systems, managing diverse array data types (including GPU device arrays), and utilizing various Hessian approximation strategies, such as different variants of BFGS methods.

Recently, we have discovered that the condensed-space interior point method (IPM) with an inequality relaxation strategy is particularly effective for solving large-scale NLPs, like AC optimal power flow, up to moderate precisions. This strategy efficiently addresses the challenges posed by sparse matrix factorization on GPUs, as the condensed KKT systems can be factorized without numerical pivoting, which has previously hindered GPU utilization in the large-scale optimization regime. Furthermore, MadNLP offers diverse approaches for handling KKT systems with special structures, via reduction strategies (citation needed) and Schur complement decomposition strategies (citation needed).

Performance Highlights Our numerical results, summarized in Table (for detailed results, please refer to [**?**]), clearly demonstrate the significant potential of using GPUs. Specifically, when applied to AC OPF problems, which are one of the most important applications of NLP, we have achieved significant performance enhancements. The GPU-based solutions achieve speedups of over 10 times compared to CPU-

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 for solving AC OPF problems

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

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

based state-of-the-art tools. Notably, ExaModels enables derivative evaluations that are up to two orders of magnitude faster, and the use of the condensed-space IPM strategy in MadNLP greatly improves linear algebra computation speed. Furthermore, we have also explored the utilization of these methods in reduced-space OPF [**?**] and model predictive control (MPC) [**?**], where we have achieved similar degrees of speedup.

Portability While NVIDIA currently leads in terms of computing power and mature implementation of low-level operations, especially with CUDA libraries, we aim to provide support for alternative architectures as well. This is crucial because many next-generation exascale high-performance computing architectures will be based on AMD and Intel accelerator devices. Furthermore, by supporting diverse platforms, we hope to drive the attention of chip makers towards the operations research field and encourage better support for software infrastructure in implementing operations research software.

Table **??** provides an overview of the current state of portability for our tools and other existing open-source software tools. ExaModels.jl already supports all major architectures, while MadNLP.jl currently only supports NVIDIA GPUs. However, we have a clear roadmap to extend support to all major accelerator architectures, including NVIDIA, AMD, and Intel GPUs. The portability is enabled by KernelAbstractions.jl, a Julia package that facilitates the portable implementation of GPU kernels.

Expected Impacts in Applications

- *Energy Infrastructures*: Enhancing the scalability of optimization methods is crucial for improving the resilience of energy system operations. We have demonstrated the ability of MadNLP with multiple GPUs to solve large-scale security-constrained AC OPF problems effectively. This showcases the potential of ExaModels and MadNLP to enhance the scalability of NLP by utilizing GPU capabilities.
- *Model Predictive Control*: MPC has been widely used for optimization-based control in various settings, such as chemical processes, energy systems, and robotics. However, the effectiveness of MPC is often limited by the efficiency of optimization solvers, especially when short sampling times are required. Our results have shown that MadNLP running on GPUs can solve such problems with high efficiency [**?**]. Moving forward, we plan to implement the extension packages for control problems so that control community can take advantage of accelerated NLP frameworks.
- *Machine Learning Surrogate Models*: An exciting future direction is the integration of neural network surrogate models into constrained optimization algorithms. GPU acceleration plays a crucial role in

these applications, considering the complexity of real-world AI models. MadNLP offers the necessary capabilities through the native handling of device arrays as well as dense KKT systems with various quasi-Newton strategies. We intend to explore these problems further and provide extensions for surrogate modeling with GPU acceleration.

Closing Remarks ExaModels and MadNLP provide computational infrastructures for operations research that align with the era of accelerated computing, which we believe the COIN-OR cup aims to recognize. By addressing critical challenges and introducing new strategies, these packages significantly extend the boundaries of computational infrastructure for operations research. We are confident that these contributions will resonate within the operations research community and beyond, opening up new possibilities in applications such as energy systems, control, and machine learning. With great enthusiasm, we nominate ExaModels and MadNLP for the COIN-OR cup, in recognition of their impact on the field.