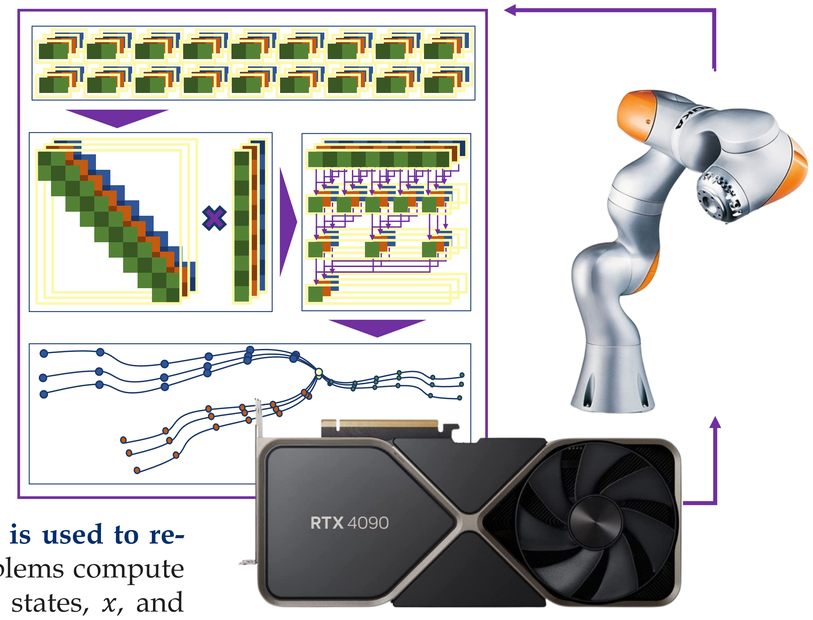
GATO: GPU-Accelerated and Batched Trajectory Optimization for Real-Time Nonlinear Model Predictive Control

Alexander Du[1], Brian Plancher [2]

1: Columbia University Fu Foundation School of Engineering and Applied Science 2: Barnard College, Columbia University

The Big Picture:

With the recent explosion in parallel computing capabilities, several recent efforts have leveraged **hardware acceleration on GPUs** to improve the computation performance of trajectory optimization, which is often nonlinear and demands **significant computation** to achieve optimality. However, most of these approaches are designed to accelerate a single solve or are optimized for very large batches of solves (e.g., >1000). In this work, we **build on our recent MPCGPU solver [1]**, which scales to kilohertz control rates for trajectories as long as 512 knot points, to **develop a batched solver for tens to hundreds of trajectory optimization solves**.



Model Predictive Control Background:

In most MPC formulations, a **trajectory optimization problem is used to re-optimize the robot's trajectory at each control step**. These problems compute a robot's optimal path through an environment as a series of states, x, and controls, u, by minimizing a cost function, $l(\cdot)$, subject to discrete time dynamics, $f(\cdot)$, and additional constraints, $g(\cdot)$, (e.g., obstacle avoidance, torque limits).

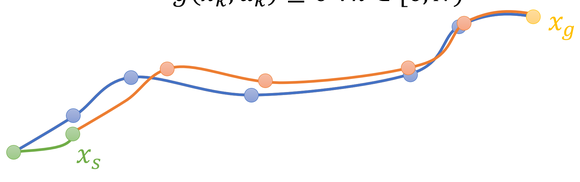
MPCGPU Solver:

MPCGPU leverages a **GPU-accelerated, block diagonal, preconditioned conjugate gradient solver [2]** (GBD-PCG) at its core, and solves the NMPC problem through **Sequential Quadratic Programming** in a three-step process:

$$\min_{X,U} \; l_f(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$

$$\text{subject to: } f(x_k, u_k) = x_{k+1} \; \forall k \in [0, N)$$

$$g(x_k, u_k) \leq 0 \; \forall k \in [0, N)$$



1) In **parallel**, compute **Taylor approximations** and form the Schur Complement and Symmetric Stair Preconditioner.
2) Use **GBD-PCG to compute δX*, δU*** efficiently by re-factoring the PCG algorithm to expose parallelism and exploiting sparsity in the trajectory optimization problem
3) Leverage a **parallel line search with a merit function** to form the final trajectory

After the trajectory is passed to the robot, this process is warm started with our last solution and run from the current state

GBD-PCG's advantage over the CPU-based QDLDL **scales with problem size, with up to a 3.6x average speedup** over QDLDL on the CPU and shows a **bi-modal solve time distribution** which is **usually 10x faster** than the uni-modal CPU and **at worst 2.5x slower**.

[1] E. Adabag, et al. "Mpcgpu: Real-time nonlinear model predictive control through preconditioned conjugate gradient on the gpu." IEEE International Conference on Robotics and Automation (ICRA). 2024.

[2] X. Bu and B. Plancher. "Symmetric stair preconditioning of linear systems for parallel trajectory optimization." IEEE International Conference on Robotics and Automation (ICRA). 2024.

Batched Implementation & Results

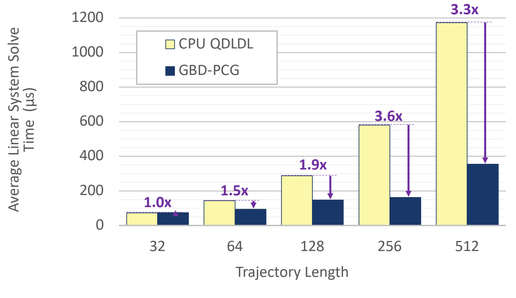
In addition to MPCGPU's optimizations, our design **parallelizes across multiple solves and boosts GPU utilization** through:

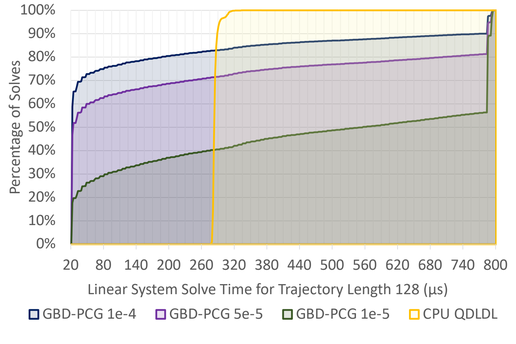
- Multi-dimensional threading and contiguous memory allocation
- **Improved PCG solver architecture** featuring improved launch configurations and more efficient reduction operations
- **Parallelization both across and within** trajectories and alpha values for merit function and line-search computations
- Fine-grained synchronization and concurrent heterogenous operations using **warp-level tiling** within kernel blocks

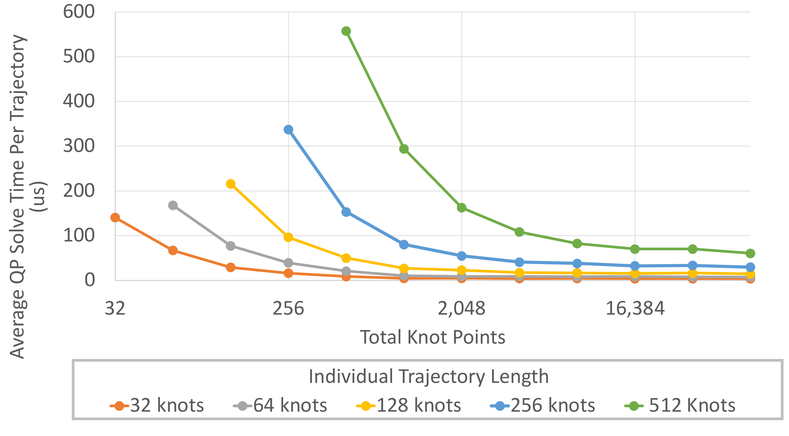
Across a range of batch sizes and trajectory, our batched approach shows **strong scalability** for our target batch sizes. E.g. 32-knot trajectories scale to batches of 32 with minimal overhead, and to batches of 128 with only a 3.9x increase in solve time.

Overall, we achieve **good scaling up to 8192 total knot points** (batch size times knots per solve), after which solve times scale linearly with further increases in size. This plateau is driven by **hardware resource limits** on our target RTX 4090 GPU. Improved future scalability should be possible on next-generation GPU hardware.

Relative Total Solve Time	Knot Points 32	64	128	256	512
Batch Size 1	1.0	1.2	1.5	2.4	4.0
16	1.0	1.2	2.6	4.7	9.4
32	1.1	2.1	4.0	8.7	16.0
64	2.3	4.1	7.6	14.9	32.0
128	3.9	7.5	14.3	30.5	55.4
256	8.4	16.1	30.1	54.3	108.1
512	14.9	28.5	53.6	105.8	211.1







Applications and Future Work

We are excited about applications of our batched solver in sampling-based planning under uncertainty, as well as opportunities for future work such as integrating with actor-critic RL to guide agent exploration and with branch-and-bound-based solvers for contact-implicit trajectory optimization.

This material is based upon work supported by the National Science Foundation (under Award 2246022). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations.