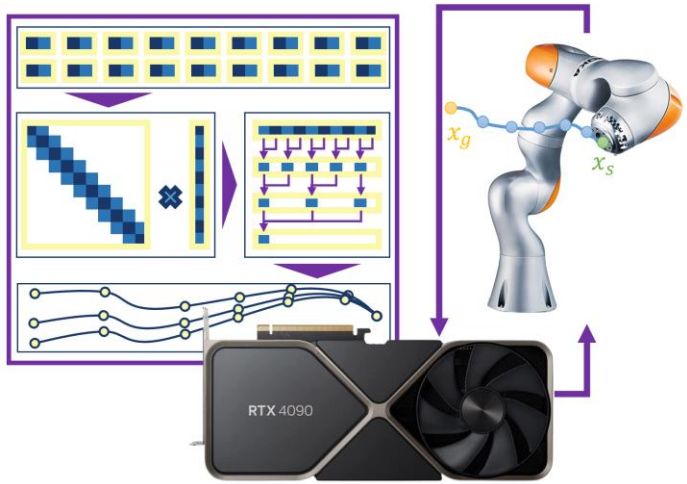
ICRA2024
May 13-17, Yokohama, Japan

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.
*These authors contributed equally to this work.

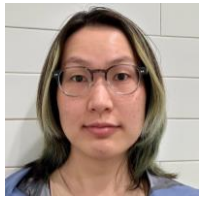
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU



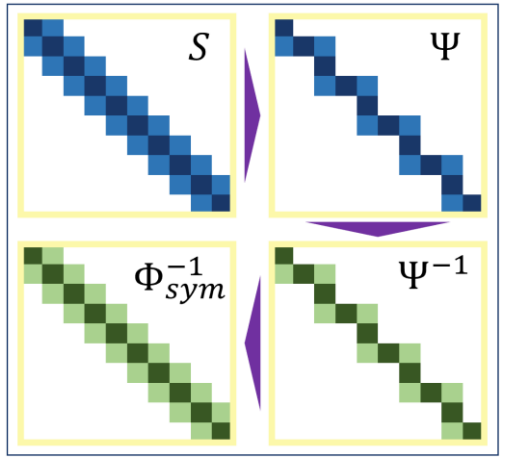

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



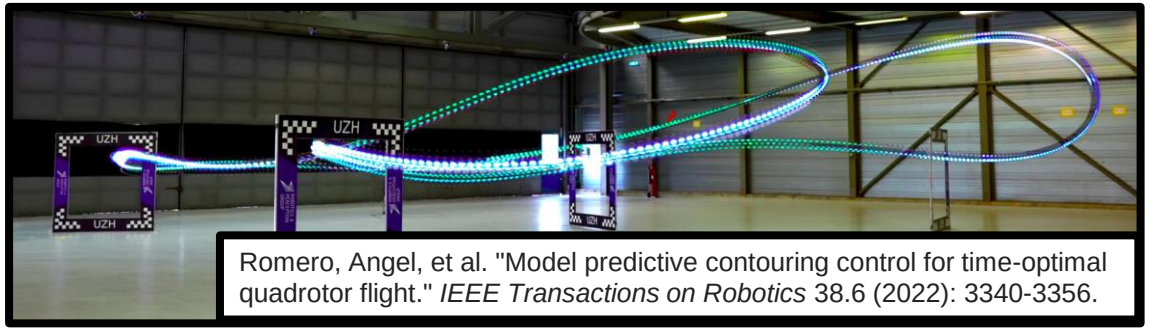
Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization



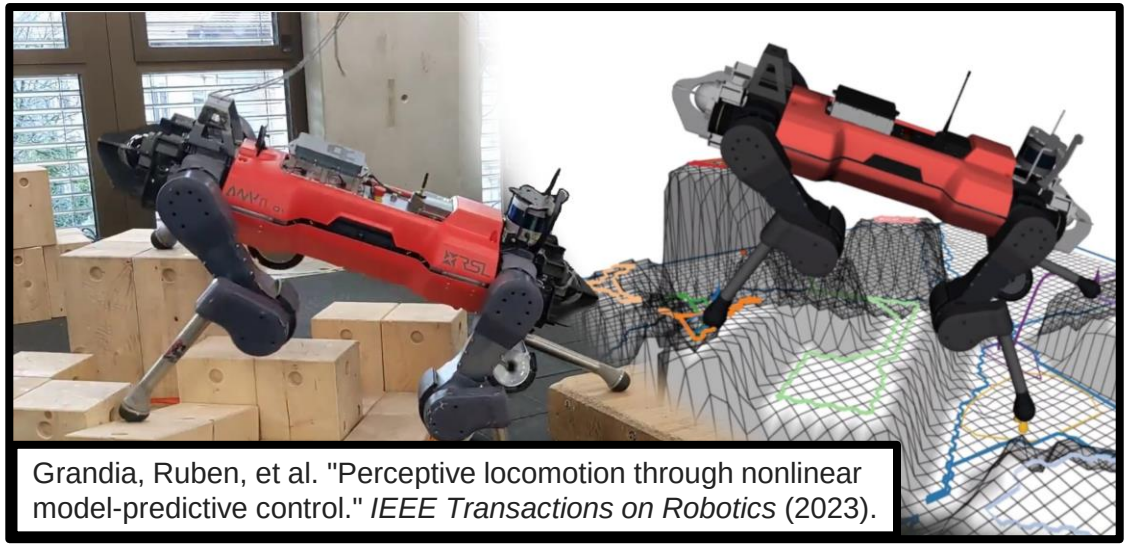

Xueyi Bu[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



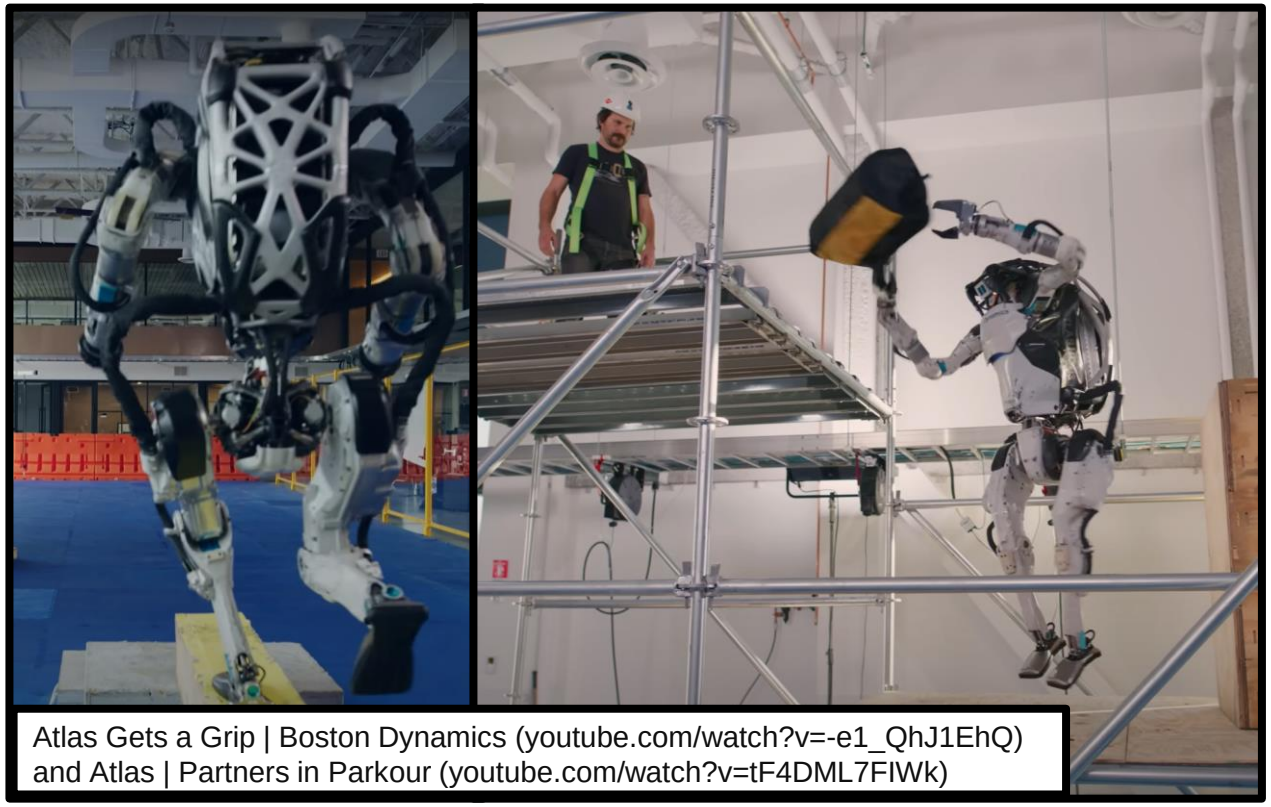
Model Predictive Control (MPC): State-of-the-Art Robot Motion Planning and Control



Romero, Angel, et al. "Model predictive contouring control for time-optimal quadrotor flight." *IEEE Transactions on Robotics* 38.6 (2022): 3340-3356.



Grandia, Ruben, et al. "Perceptive locomotion through nonlinear model-predictive control." *IEEE Transactions on Robotics* (2023).



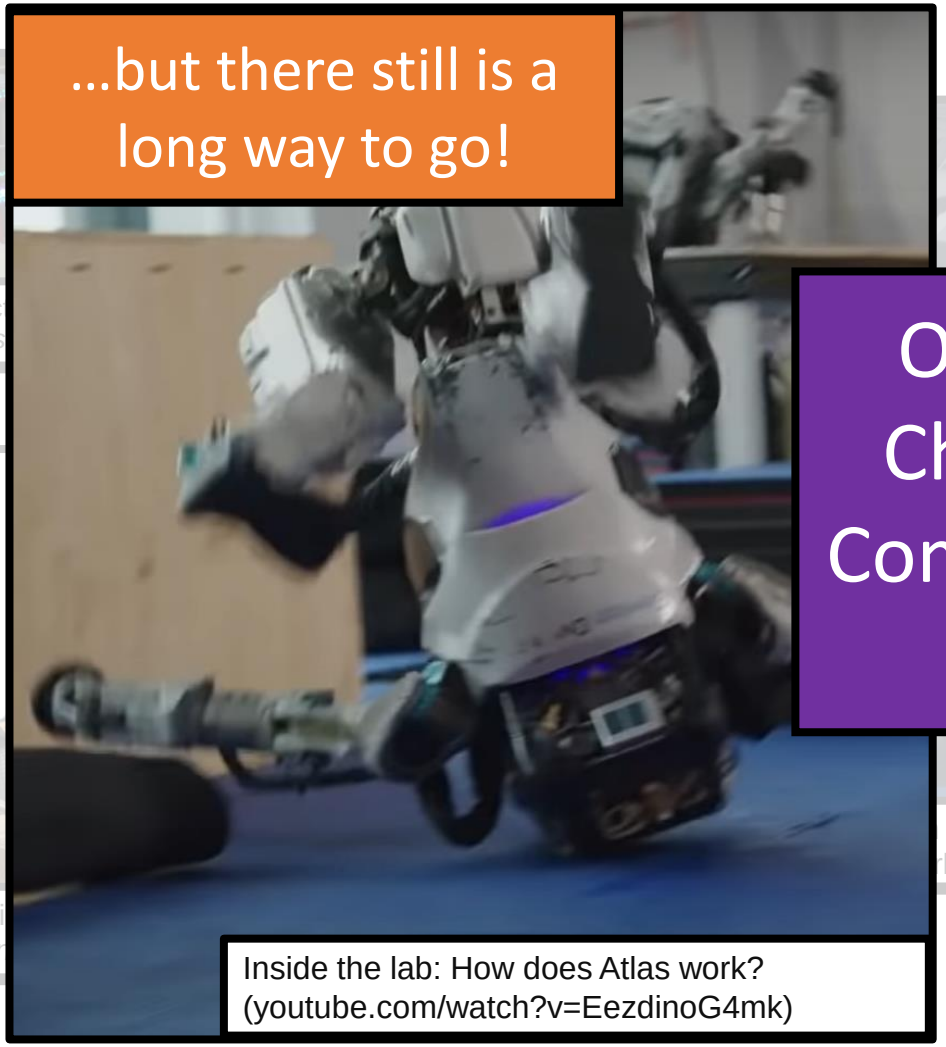
Atlas Gets a Grip | Boston Dynamics (youtube.com/watch?v=-e1_QhJ1EhQ) and Atlas | Partners in Parkour (youtube.com/watch?v=tF4DML7FIWk)

Model Predictive Control (MPC): State-of-the-Art Robot Motion Planning and Control

Romero, Angel, et al. "Model predic
quadrotor flight." *IEEE Transactions*

Grandia, Ruben, et al. "Perceptive locomoti
model-predictive control." *IEEE Transactio*

…but there still is a long way to go!

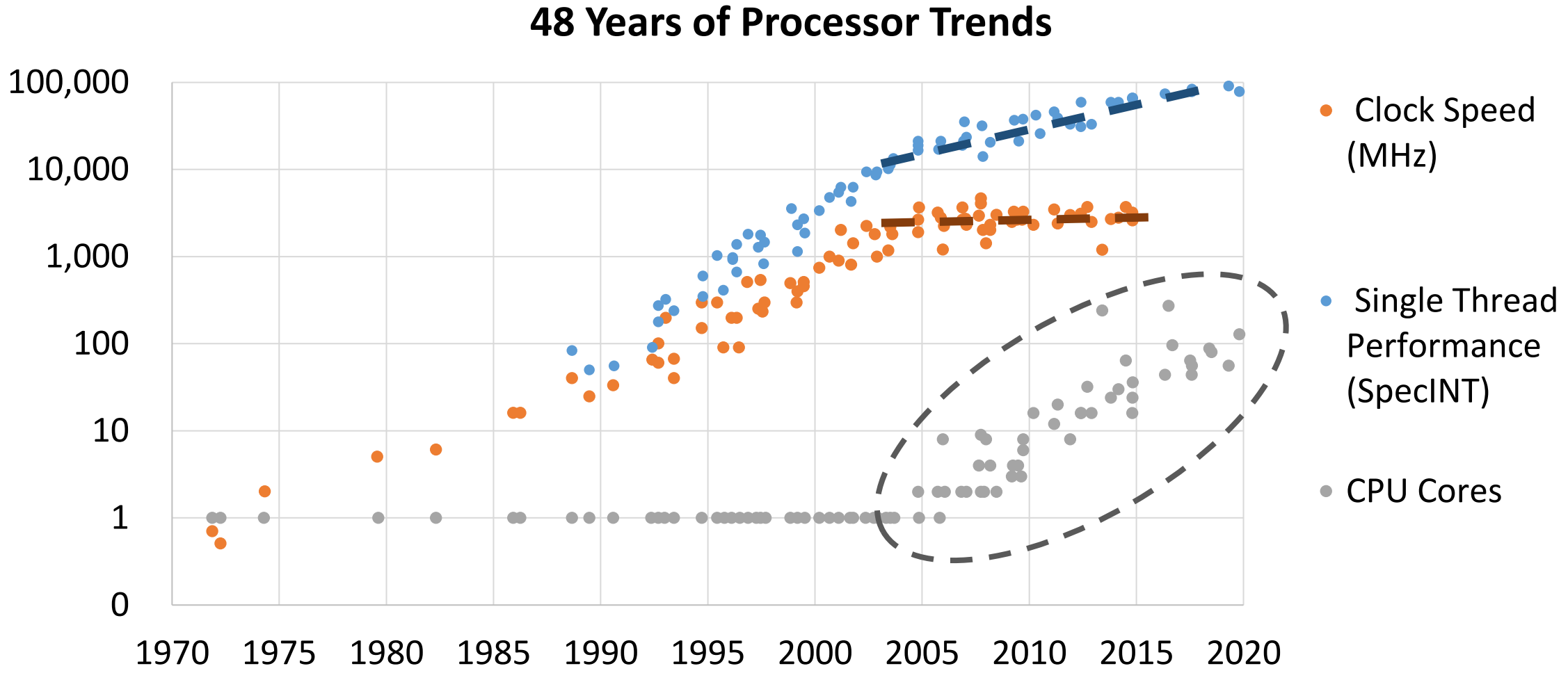


Inside the lab: How does Atlas work? (youtube.com/watch?v=EezdinoG4mk)

One Major Challenge – Computational Speed!

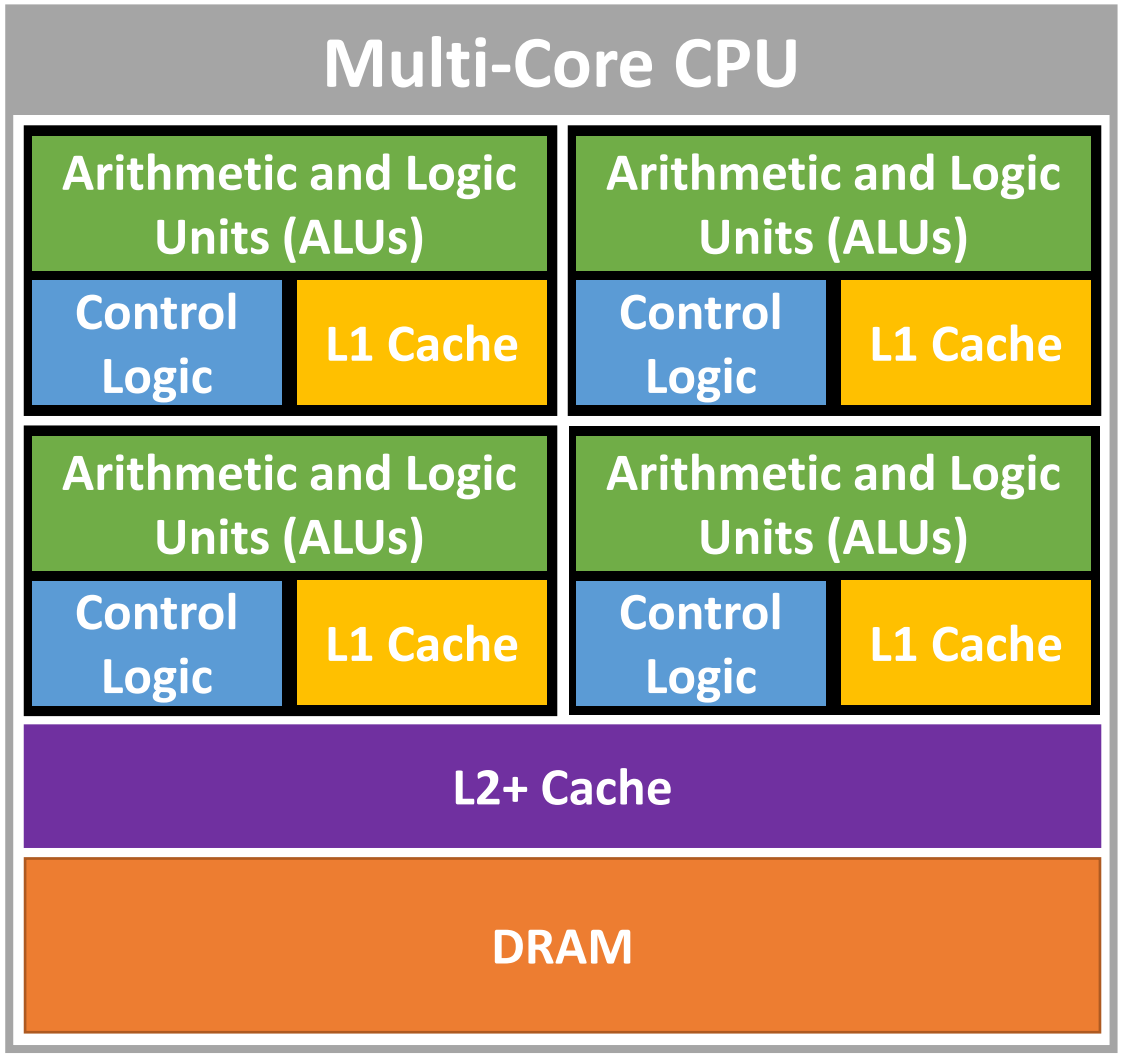
Dynamics (youtube.com/watch?v=-e1_QhJ1EhQ)
kour (youtube.com/watch?v=tF4DML7FIWk)

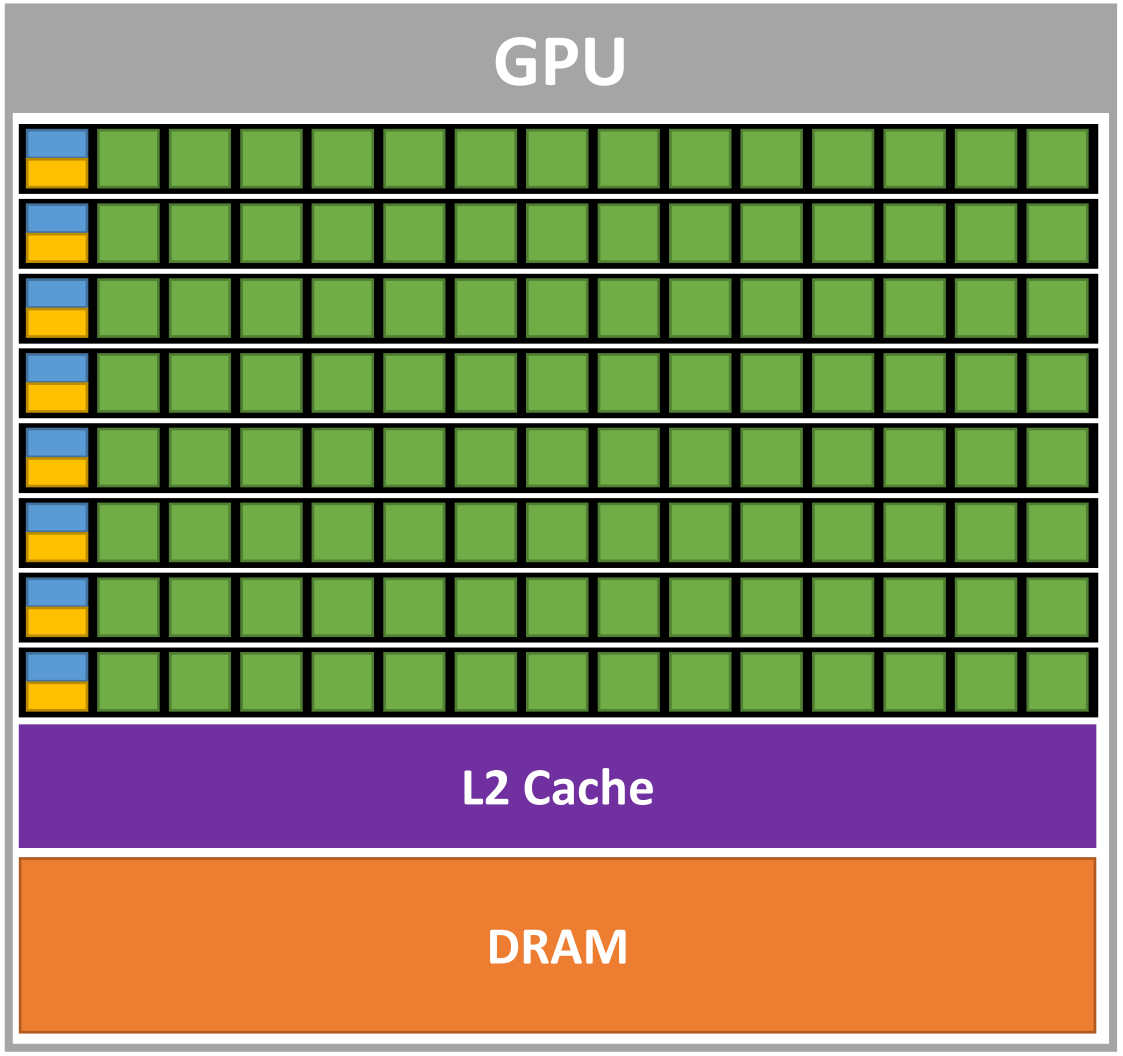
CPUs are not getting faster and CPU design benefits are decreasing in effect resulting in a need for parallelism!



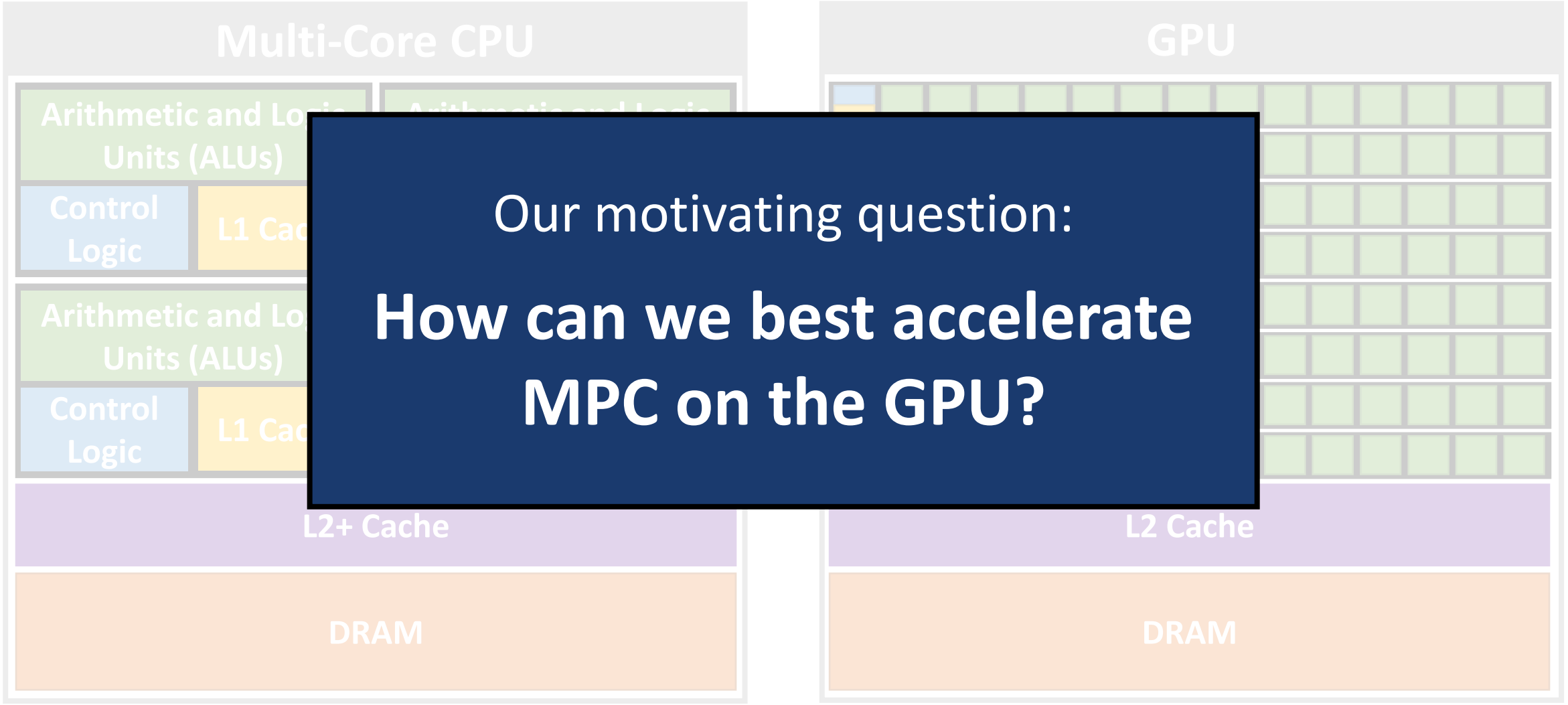
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp https://github.com/karlrupp/microprocessor-trend-data

GPUs offer increased computational parallelism and are natural choice to explore to accelerate MPC





GPUs offer increased computational parallelism and are natural choice to explore to accelerate MPC



Our motivating question:

How can we best accelerate MPC on the GPU?

Model Predictive Control (MPC): Via Successive Convex Optimization

Direct Transcription

$$\min_{x_0,u_0 \ldots x_{N-1},u_{N-1},x_N} l_f(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$

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

$g(x_k, u_k) \geq 0 \ \forall k$

Minimize a separable cost across time

Obey Physics

Other Constraints

Position/Velocity/Torque Limits
Collision Avoidance
Waypoint Goals (x_s, x_g)

MPC: Control through continuous re-planning often via Trajectory Optimization

Model Predictive Control (MPC): Via Successive Convex Optimization

Successive Convex Optimization

While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem

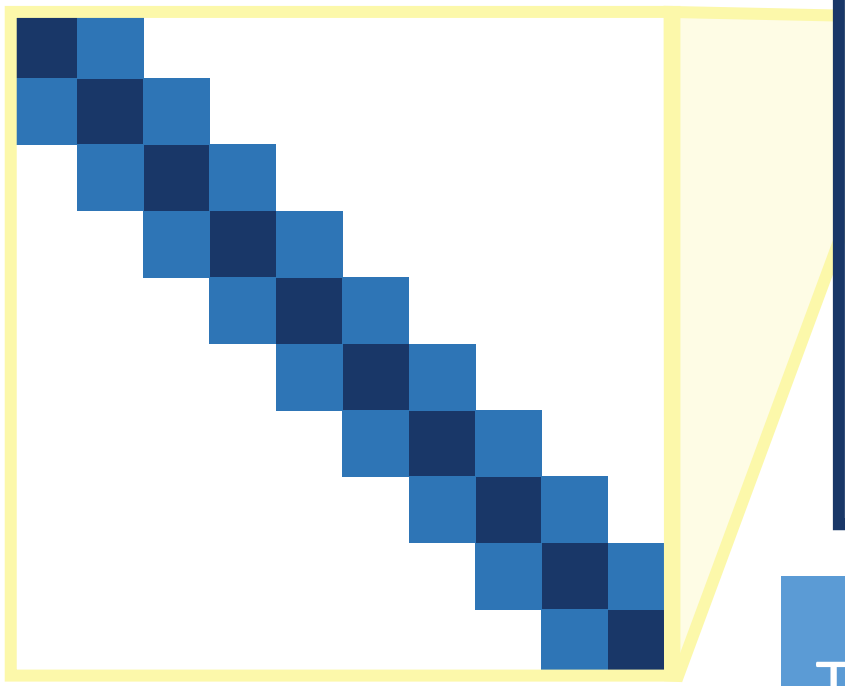
Solve the KKT System

$$\begin{bmatrix} G & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} -\delta Z \\ \lambda \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix}$$

Iterative Methods like the **Preconditioned Conjugate Gradient** Algorithm are GPU-Parallel Friendly

But they require a **symmetric system and preconditioner**!

Solving the KKT System via the Schur Complement



$$S\lambda^* = \gamma$$

$$S = -CG^{-1}C^T$$

$$\gamma = c - CG^{-1}g$$

$$\delta Z^* = G^{-1}\left(g - C^T\lambda^*\right)$$

Schur Complement

$$\begin{bmatrix} G & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} -\delta Z \\ \lambda \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix}$$

KKT System

The remaining trick is then to find a good ***parallel*** preconditioner

$$P \approx A$$

$$Ax = b$$

$$P^{-1}Ax = P^{-1}b$$

The trick is then to find a good *parallel* preconditioner

Jacobi or Block-Jacobi

Alternating/Overlapping Block

Can we do better?

1. Introduction and Motivation
2. Background and Related Work
3. Parallel Preconditioners
4. **The (Symmetric) Stair Preconditioner**
5. Experiments and Results

Polynomial Splitting Preconditioners

$$S = \Psi - P$$

$$S^{-1} \approx \Phi^{-1} = (I + \Psi^{-1}P + (\Psi^{-1}P)^2 \dots)\Psi^{-1}$$

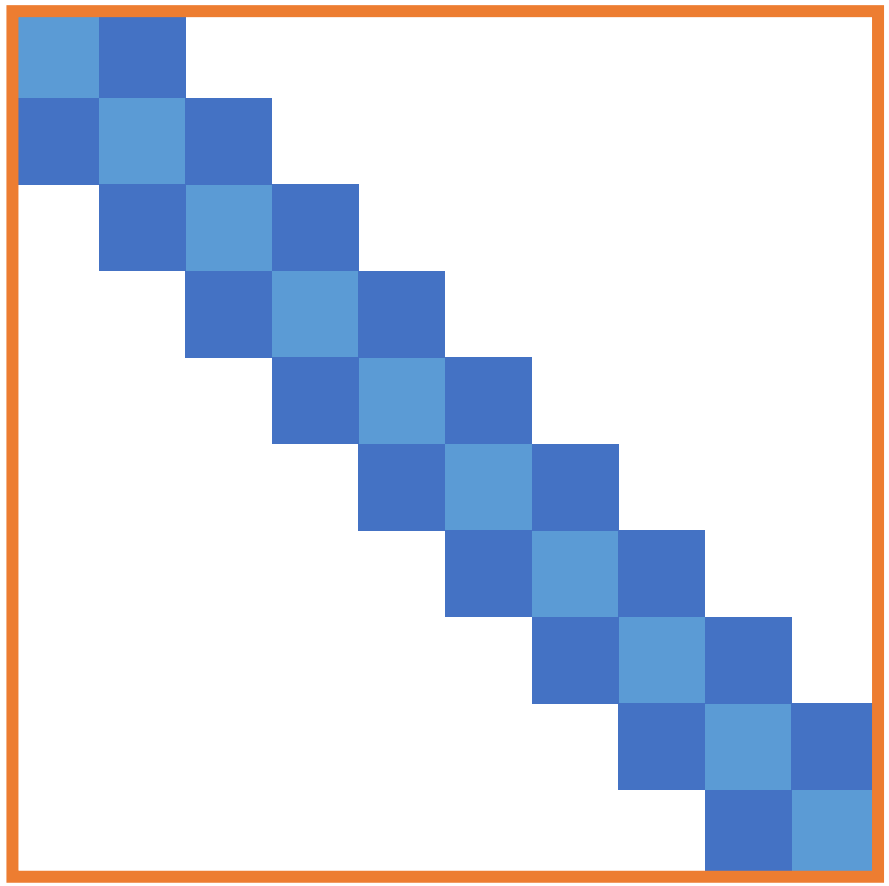
Tradeoff between accuracy and computational complexity & sparsity

Key Requirement:

$$\rho(\Psi^{-1}P) < 1$$

A block tri-diagonal **stair** preconditioner

$$S = \Psi - P \qquad S^{-1} \approx \Phi^{-1} = (I + \Psi^{-1}P + (\Psi^{-1}P)^2 \dots)\Psi^{-1}$$



Schur Complement is Block Tri-Diagonal and Symmetric!

$$S = \begin{bmatrix} D_1 & O_1 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & O_2^T & D_3 \end{bmatrix}$$

A block tri-diagonal **stair** preconditioner

$$S = \Psi - P \qquad S^{-1} \approx \Phi^{-1} = (I + \Psi^{-1}P + (\Psi^{-1}P)^2 \dots)\Psi^{-1}$$

Left and Right Stair Splitting

$$S = \begin{bmatrix} D_1 & O_1 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & O_2^T & D_3 \end{bmatrix}$$

$$\Psi_l = \begin{bmatrix} D_1 & 0 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & 0 & D_3 \end{bmatrix} \qquad P_l = -\begin{bmatrix} 0 & O_1 & 0 \\ 0 & 0 & 0 \\ 0 & O_2^T & 0 \end{bmatrix}$$

$$\Psi_r = \begin{bmatrix} D_1 & O_1 & 0 \\ 0 & D_2 & 0 \\ 0 & O_2^T & D_3 \end{bmatrix} \qquad P_r = -\begin{bmatrix} 0 & 0 & 0 \\ O_1^T & 0 & O_2 \\ 0 & 0 & 0 \end{bmatrix}$$

[52] Li, Hou-Biao, et al. "On some new approximate factorization methods for block tridiagonal matrices suitable for vector and parallel processors."
[53] Li, Hou-Biao, et al. "Chebyshev-type methods and preconditioning techniques."
[54] Lu, Hao. "Stair matrices and their generalizations with applications to iterative methods I: A generalization of the successive overrelaxation method."

A block tri-diagonal **stair** preconditioner

$$S = \Psi - P \qquad S^{-1} \approx \Phi^{-1} = (I + \Psi^{-1}P + (\Psi^{-1}P)^2 \dots)\Psi^{-1}$$

Sparse Analytical Inverse

$$\Psi_l = \begin{bmatrix} D_1 & 0 & 0 \\ O_1^T & D_2 & O_2 \\ 0 & 0 & D_3 \end{bmatrix} \qquad \Psi_l^{-1} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ -D_2^{-1}O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1}O_2 D_3^{-1} \\ 0 & 0 & D_3^{-1} \end{bmatrix}$$

$$\Psi_r = \begin{bmatrix} D_1 & O_1 & 0 \\ 0 & D_2 & 0 \\ 0 & O_2^T & D_3 \end{bmatrix} \qquad \Psi_r^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1 D_2^{-1} & 0 \\ 0 & D_2^{-1} & 0 \\ 0 & -D_3^{-1}O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix}.$$

[52] Li, Hou-Biao, et al. "On some new approximate factorization methods for block tridiagonal matrices suitable for vector and parallel processors."
[53] Li, Hou-Biao, et al. "Chebyshev-type methods and preconditioning techniques."
[54] Lu, Hao. "Stair matrices and their generalizations with applications to iterative methods I: A generalization of the successive overrelaxation method."

A block tri-diagonal **stair** preconditioner

Benefits

- Sparse
- Parallel-Friendly
- $\rho(\Psi^{-1}P) < 1$

Challenges

- Not Symmetric so PCG will ***not*** work!

$$S^{-1} \approx \Phi^{-1} = \Psi^{-1}$$

Is this good for **PCG** inside MPC?

$$\Psi_l^{-1} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ -D_2^{-1}O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1}O_2 D_3^{-1} \\ 0 & 0 & D_3^{-1} \end{bmatrix}$$

$$\Psi_r^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1 D_2^{-1} & 0 \\ 0 & D_2^{-1} & 0 \\ 0 & -D_3^{-1}O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix}.$$

[52] Li, Hou-Biao, et al. "On some new approximate factorization methods for block tridiagonal matrices suitable for vector and parallel processors."
[53] Li, Hou-Biao, et al. "Chebyshev-type methods and preconditioning techniques."
[54] Lu, Hao. "Stair matrices and their generalizations with applications to iterative methods I: A generalization of the successive overrelaxation method."

A symmetric block tri-diagonal stair-based preconditioner

$$\Phi_{add}^{-1} = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})$$

Symmetric!

Transpose!

$$\Psi_l^{-1} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ -D_2^{-1}O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1}O_2 D_3^{-1} \\ 0 & 0 & D_3^{-1} \end{bmatrix}$$

$$\Psi_r^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1 D_2^{-1} & 0 \\ 0 & D_2^{-1} & 0 \\ 0 & -D_3^{-1}O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix}.$$

[52], [53], [54]

A symmetric block tri-diagonal stair-based preconditioner

$$\Phi_{add}^{-1} = \frac{1}{2}(\Psi_l^{-1} + \Psi_r^{-1})$$

Symmetric!

Benefits

- Sparse
- Parallel-Friendly
- Symmetric Positive Definite
- $\rho(\Phi_{add}^{-1} P) < 1$

Challenges

$$\lambda(\Phi_{add}^{-1} S) \in \left(0, \frac{9}{8}\right]$$

PCG ***not*** guaranteed to converge!

[52], [53], [54]

An improved symmetric block tri-diagonal stair-based preconditioner

$$\Phi_{sym}^{-1} = \Psi_l^{-1} + \Psi_r^{-1} - D^{-1}$$

$$\Psi_l^{-1} = \begin{bmatrix} D_1^{-1} & 0 & 0 \\ -D_2^{-1} O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1} O_2 D_3^{-1} \\ 0 & 0 & D_3^{-1} \end{bmatrix}$$

T T

$$\Phi_{sym}^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1} O_1 D_2^{-1} & 0 \\ -D_2^{-1} O_1^T D_1^{-1} & D_2^{-1} & -D_2^{-1} O_2 D_3^{-1} \\ 0 & -D_3^{-1} O_2^T D_2^{-1} & D_3^{-1} \end{bmatrix}$$

[52], [53], [54]

An improved symmetric block tri-diagonal stair-based preconditioner

$$\Phi_{sym}^{-1} = \begin{bmatrix} D_1^{-1} & -D_1^{-1}O_1D_2^{-1} & 0 \\ -D_2^{-1}O_1^TD_1^{-1} & D_2^{-1} & -D_2^{-1}O_2D_3^{-1} \\ 0 & -D_3^{-1}O_2^TD_2^{-1} & D_3^{-1} \end{bmatrix}$$

Benefits

- Sparse
- Parallel-Friendly
- Symmetric Positive Definite
- $\rho(\Phi_{sym}^{-1}P) < 1$

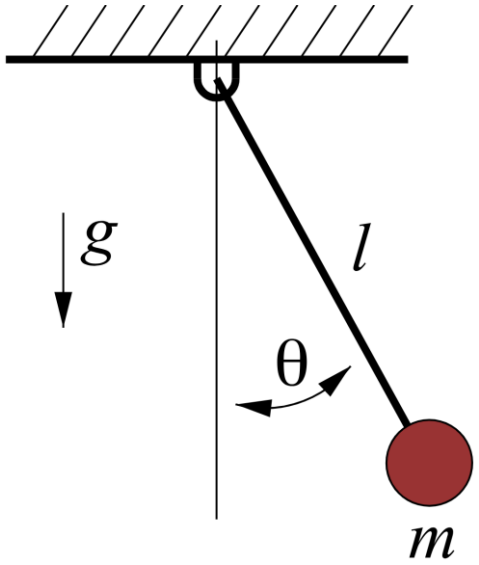
$$\lambda(\Phi_{sym}^{-1}S) \in (0, 1]$$

[52], [53], [54]

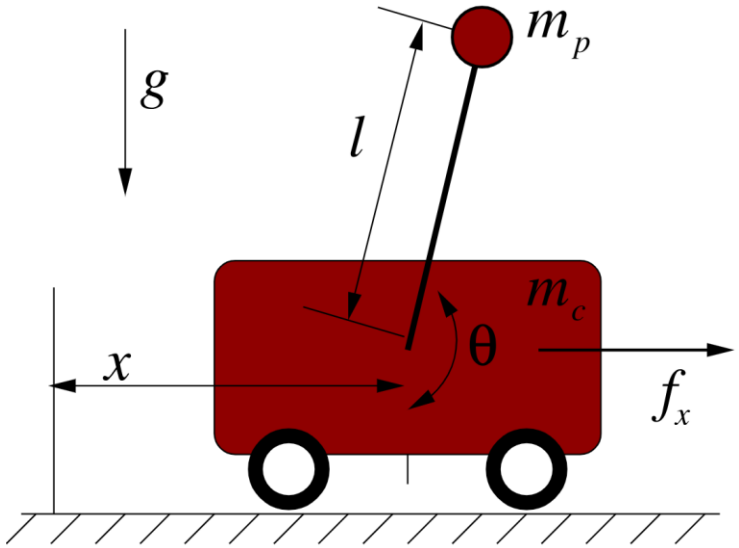
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU

1. Introduction and Motivation
2. Background and Related Work
3. Parallel Preconditioners
4. The (Symmetric) Stair Preconditioner
5. **Experiments and Results**

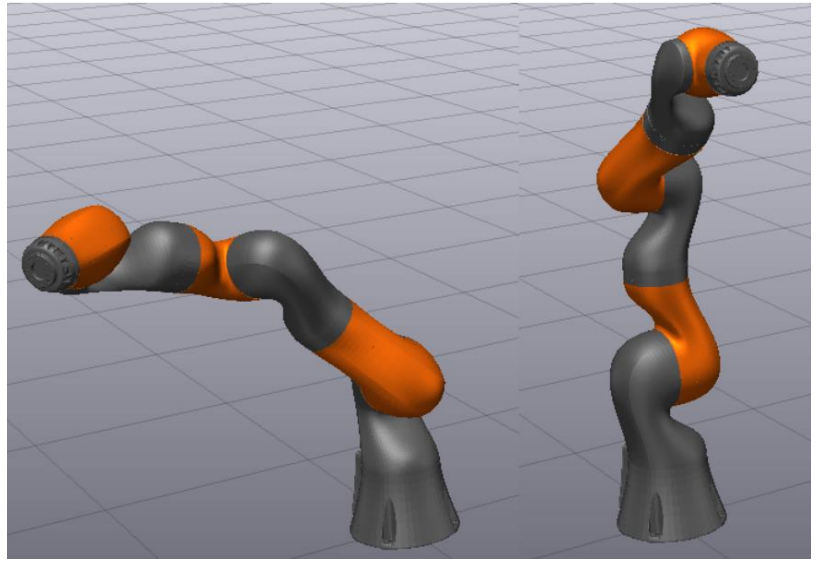
We tested the impact of our preconditioner on representative trajectory optimization problems



Pendulum



Cart Pole



Manipulator

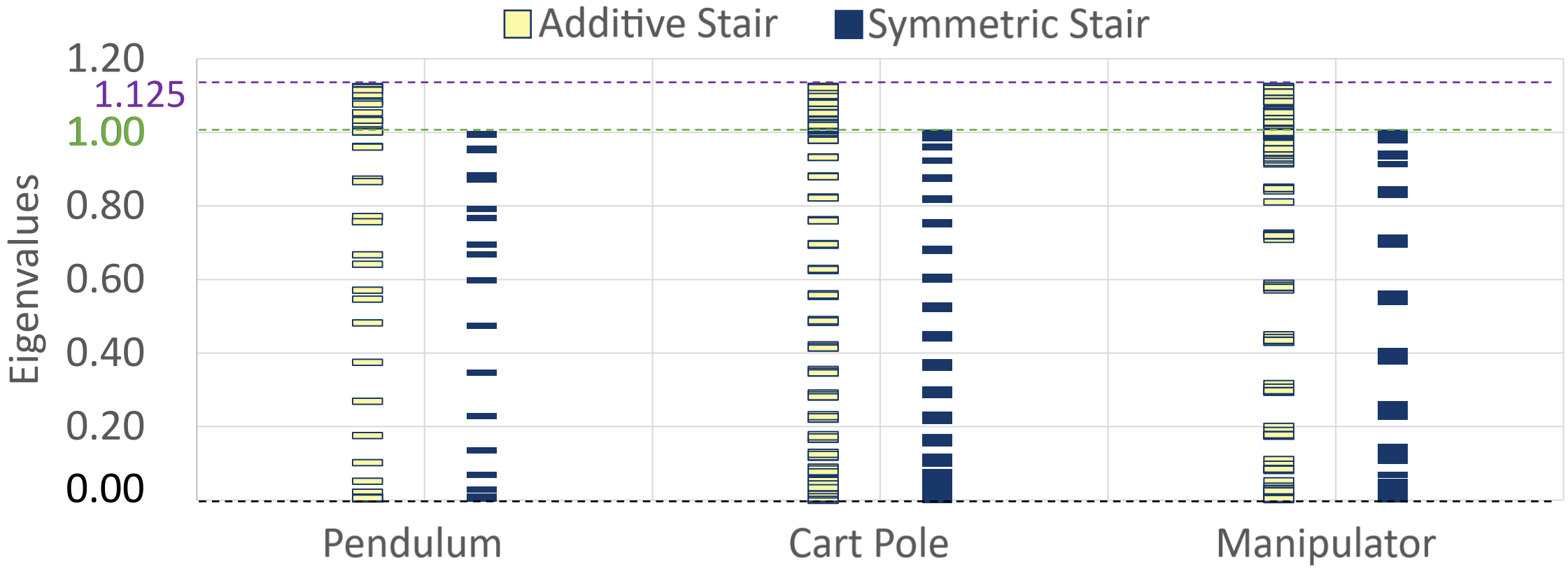
github.com/A2R-Lab/SymStair

Image Sources: Drake Simulator and underactuated.csail.mit.edu

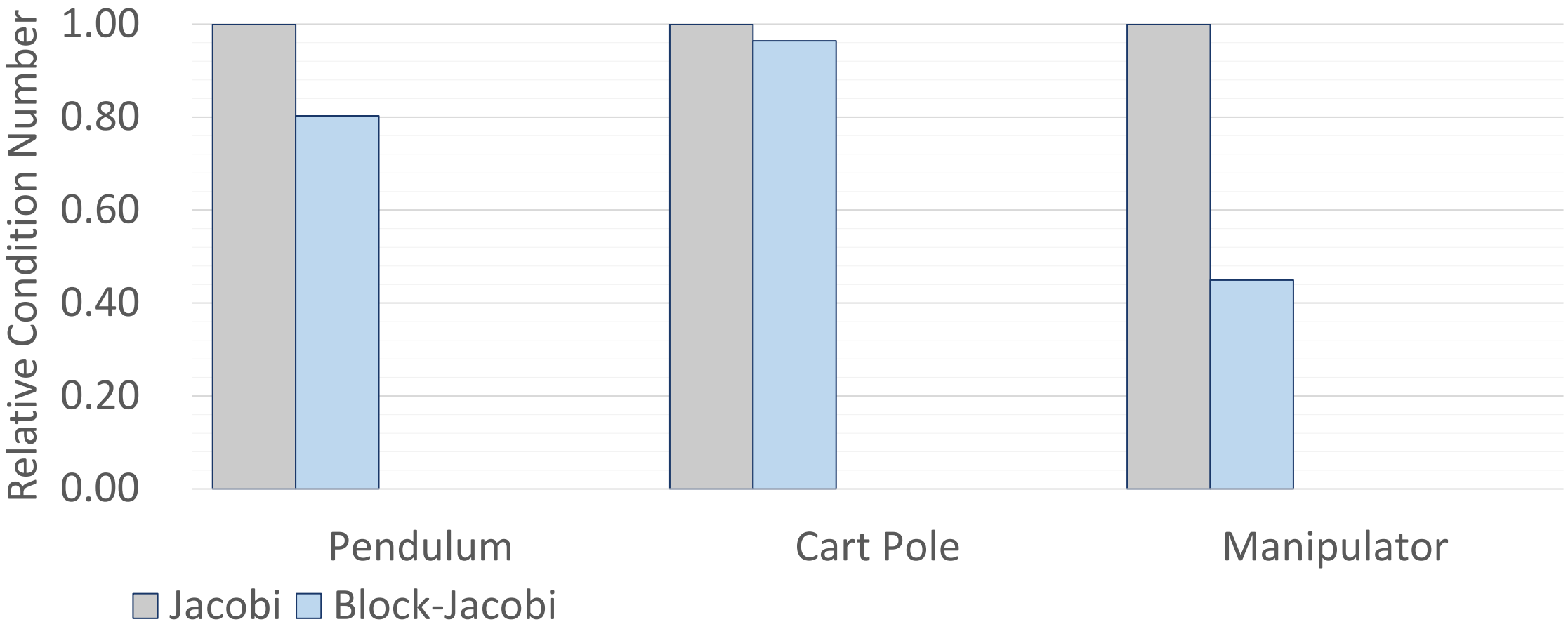
Numerical analysis confirms the theoretical **eigenvalue distribution** for trajectory optimization …

$$\lambda(\Phi_{add}^{-1}S) \in \left(0, \frac{9}{8}\right]$$

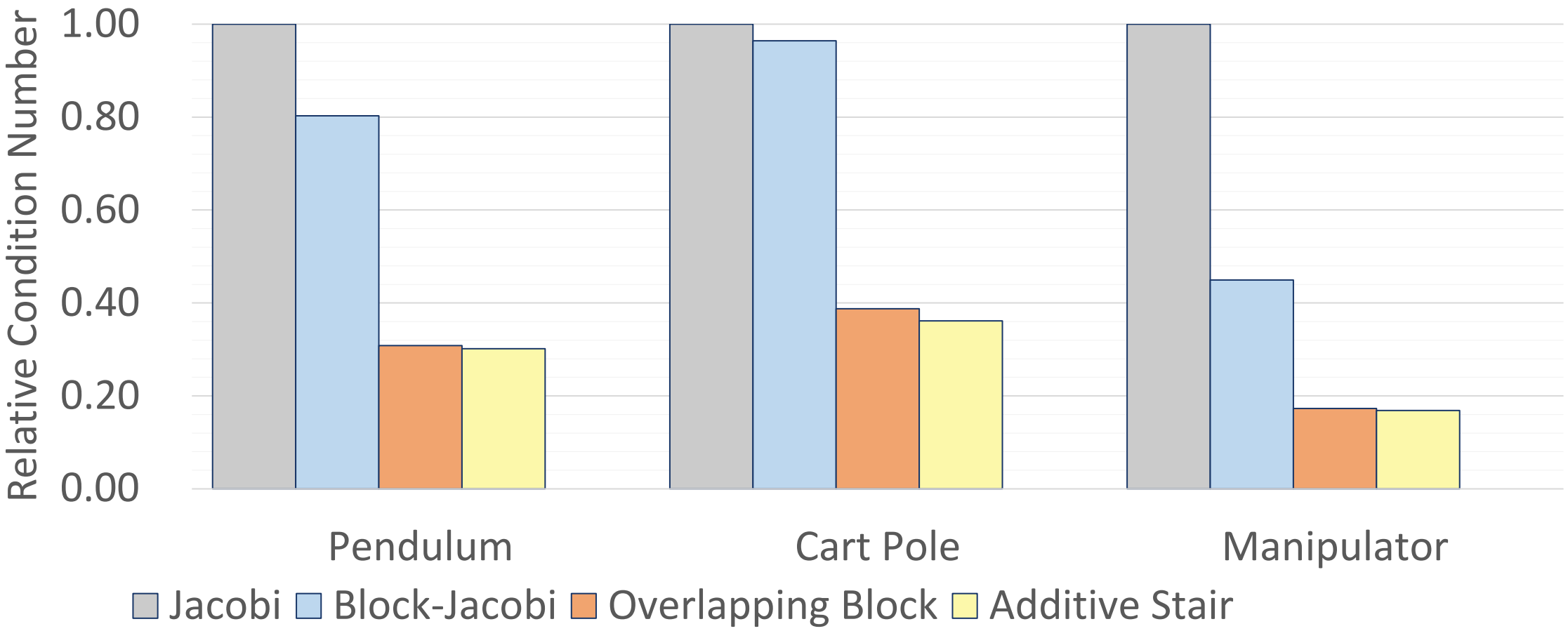
$$\lambda(\Phi_{sym}^{-1}S) \in (0, 1]$$



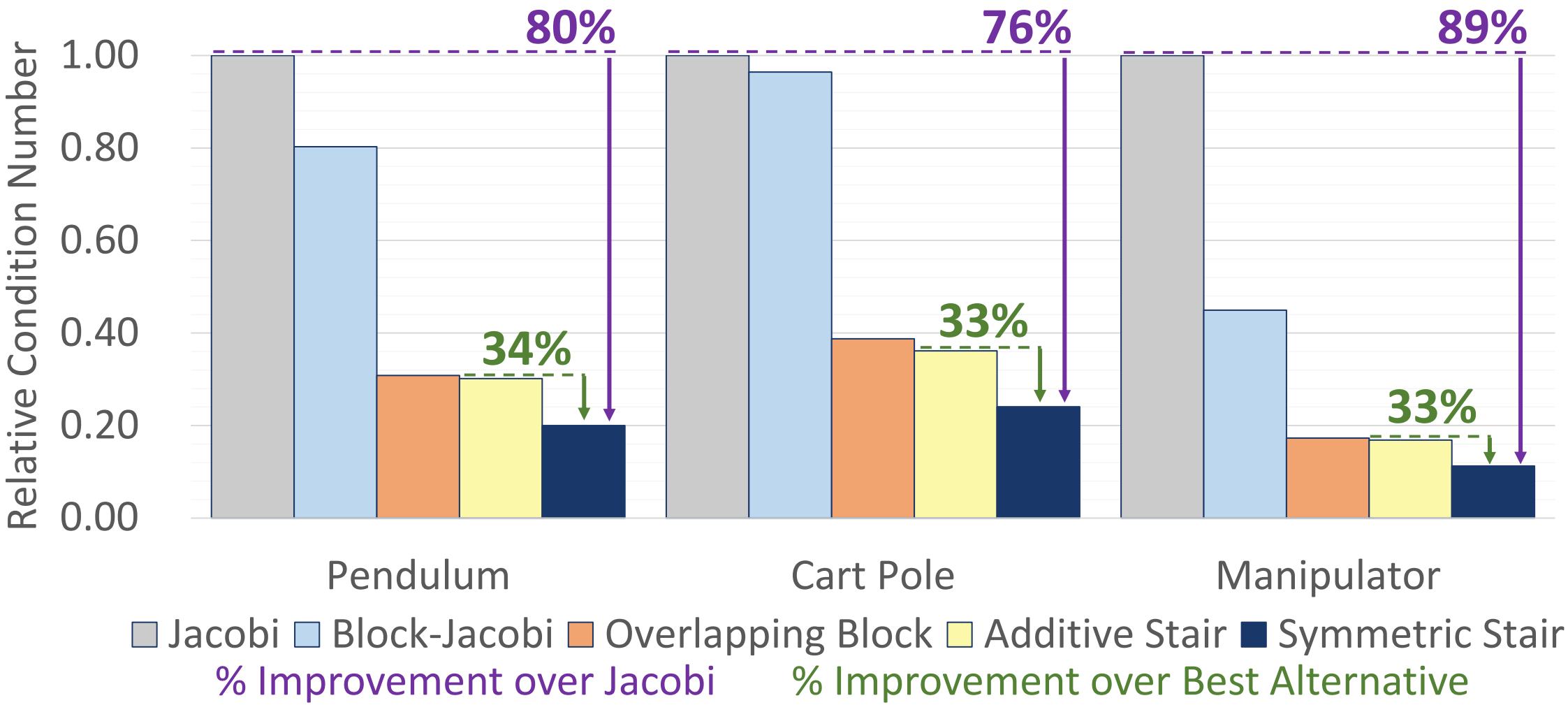
…resulting in a drastically **reduced condition number**…



…resulting in a drastically reduced condition number…

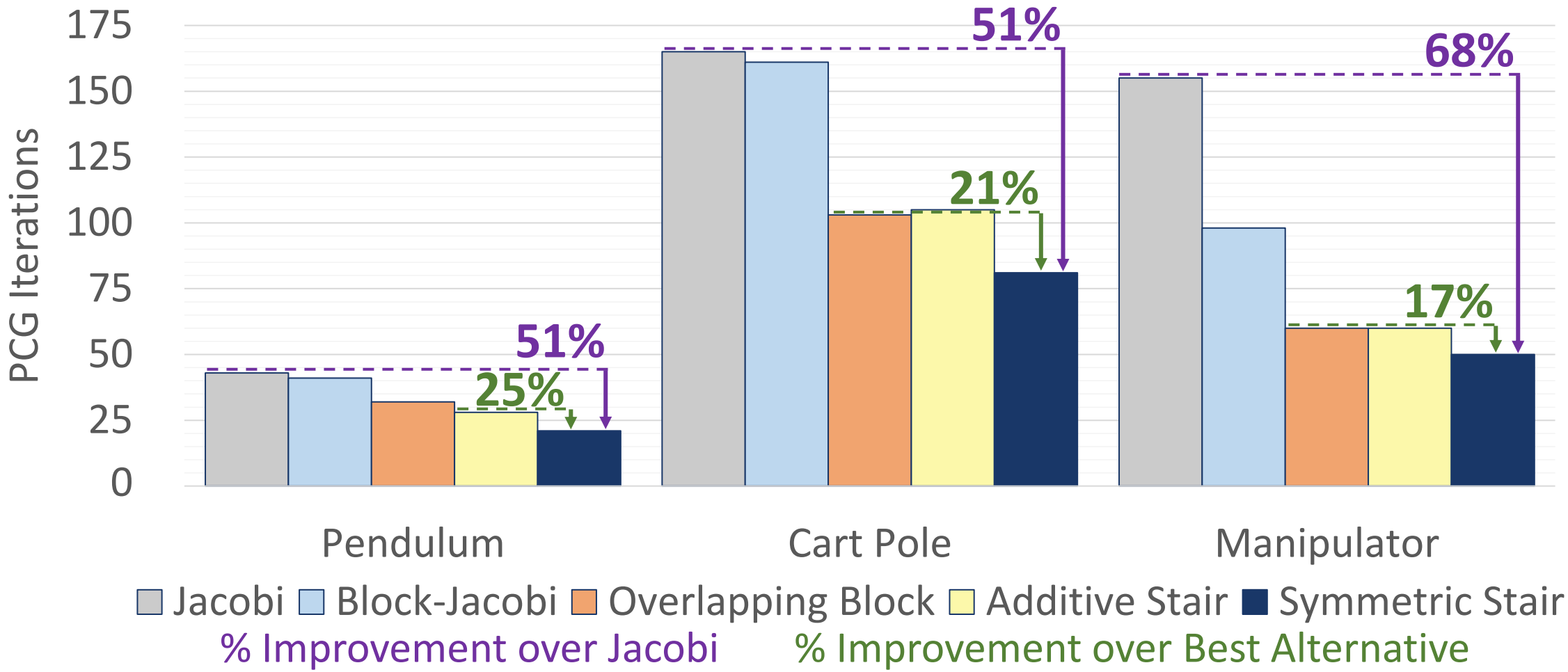


…resulting in a drastically reduced condition number…

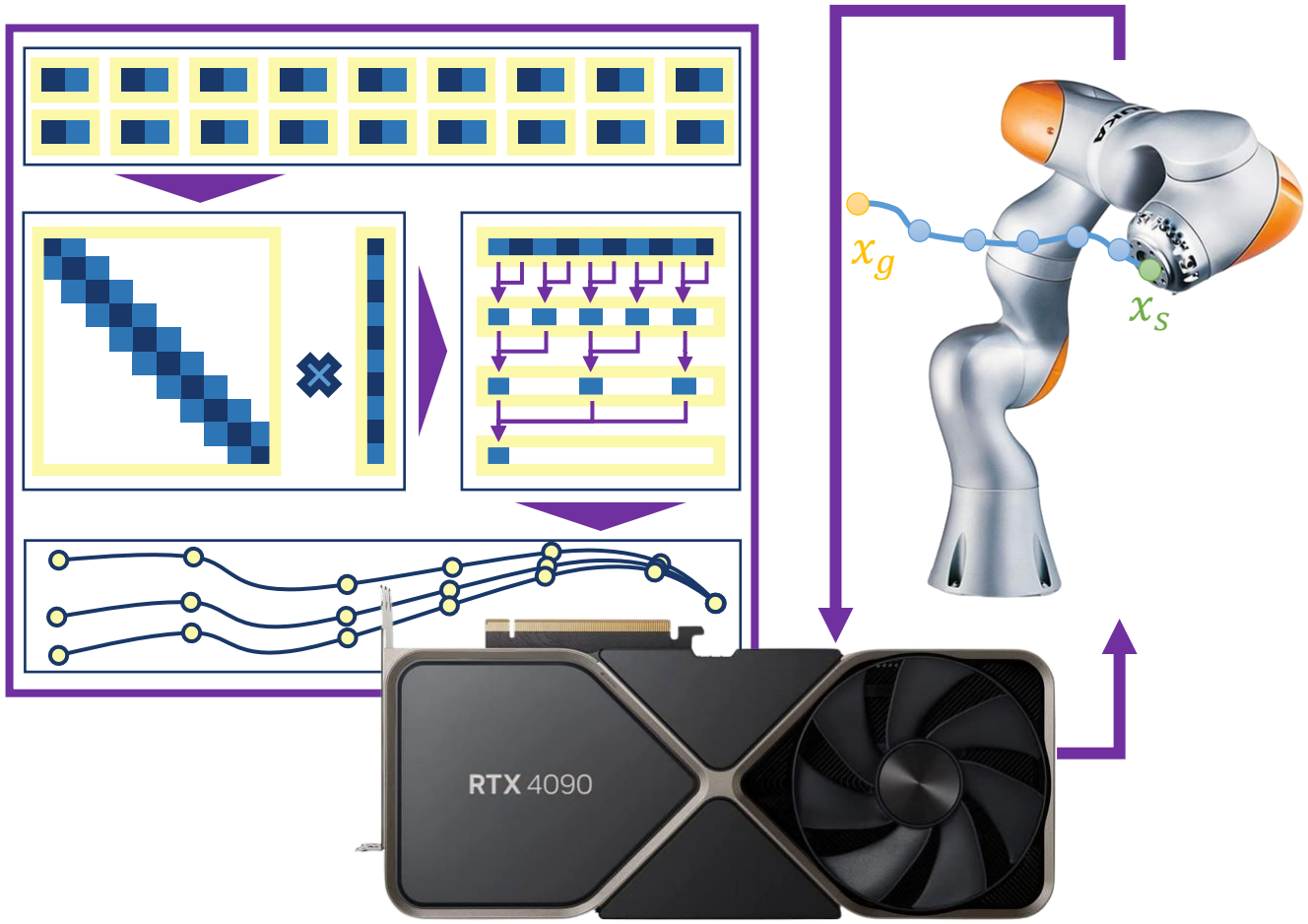


…and critically **fewer iterations to convergence** on representative trajectory optimization problems

…and critically **fewer iterations to convergence** on representative trajectory optimization problems



How can we leverage this to accelerate end-to-end robotic planning and control systems?



Stay for the next paper!

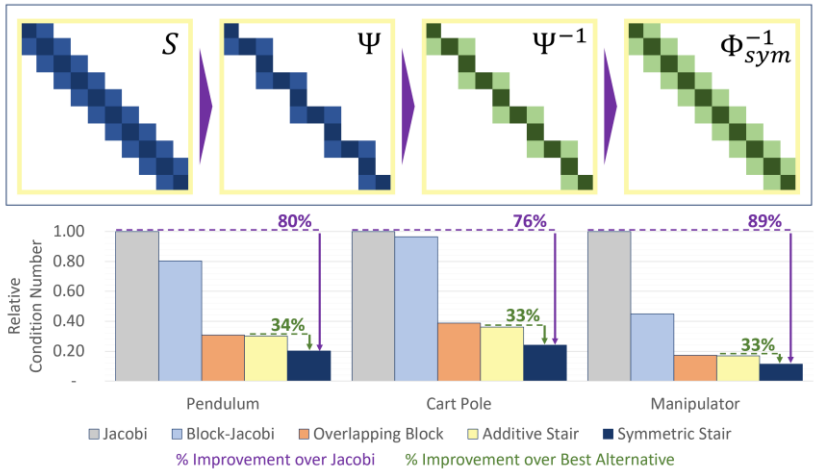
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU

Emre Adabag[1], Miloni Atal* [1], William Gerard* [1], Brian Plancher [2]
1: School of Engineering and Applied Science, Columbia University 2: Barnard College, Columbia University

Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization

1. The symmetric stair preconditioner is parallel friendly and has advantageous theoretical properties (resulting spectral radius ≤ 1)

2. This translates to improved condition number and **iterations to convergence** for iterative linear system solvers in the context of trajectory optimization



The Barnard Accessible and Accelerated Robotics Lab

Xueyi Bu[1], Brian Plancher[2]

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

bplancher@barnard.edu a2r-lab.org

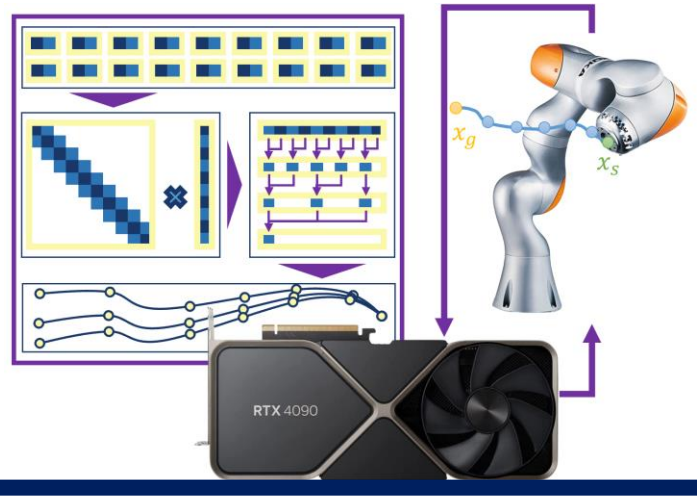
ICRA2024
May 13-17, Yokohama, Japan

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.
*These authors contributed equally to this work.

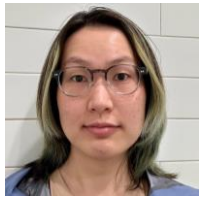
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU



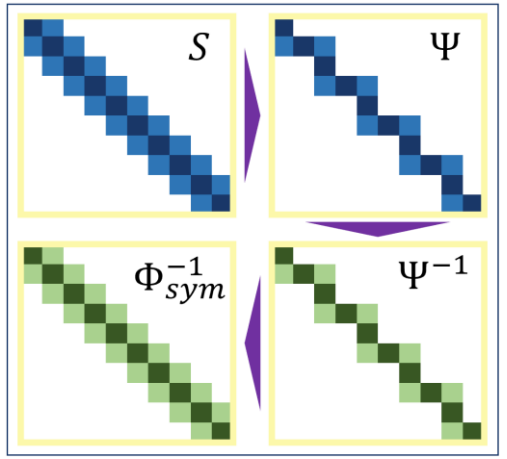

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization




Xueyi Bu[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



ICRA2024
May 13-17, Yokohama, Japan

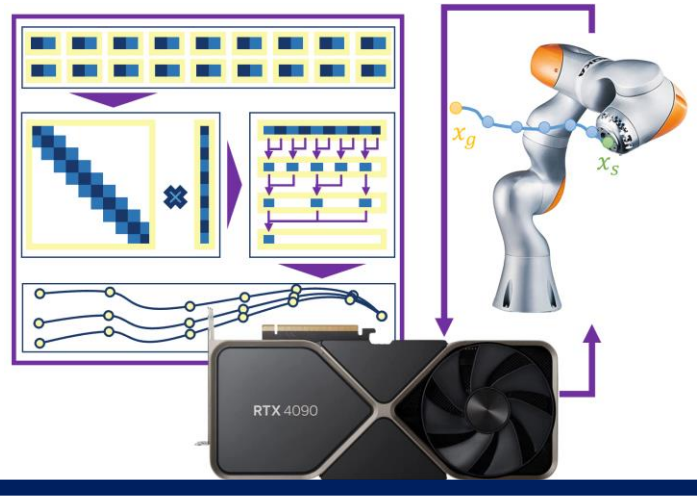
The Barnard Accessible and Accelerated Robotics Lab

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.
*These authors contributed equally to this work.

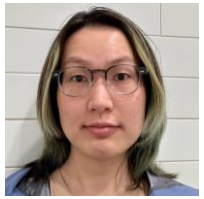
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU



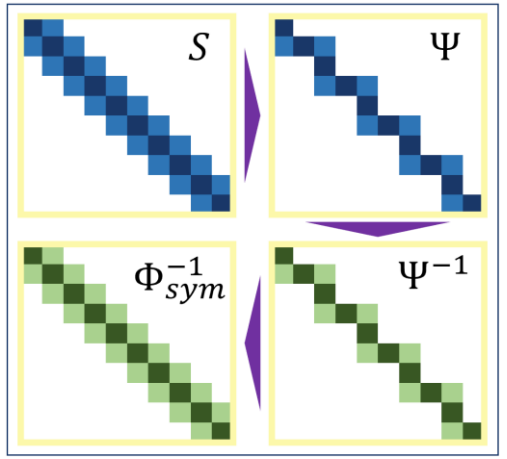

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization




Xueyi Bu[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



ICRA2024
May 13-17, Yokohama, Japan

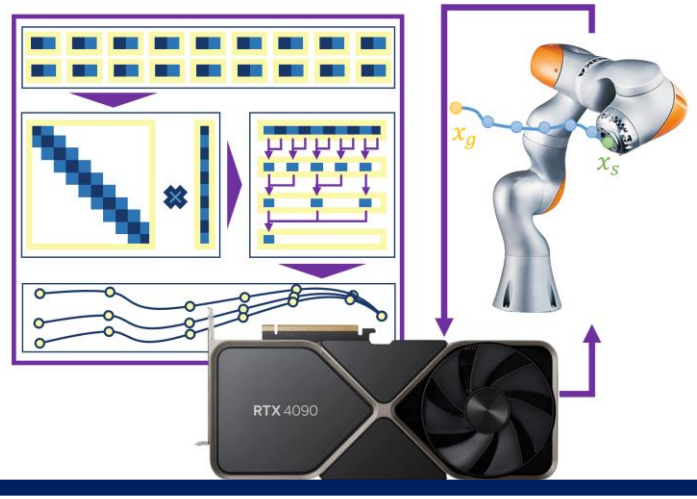
The Barnard Accessible and Accelerated Robotics Lab

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.
*These authors contributed equally to this work.

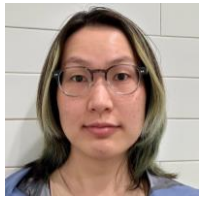
MPCGPU: Real-Time Nonlinear Model Predictive Control through **Preconditioned Conjugate Gradient** on the GPU



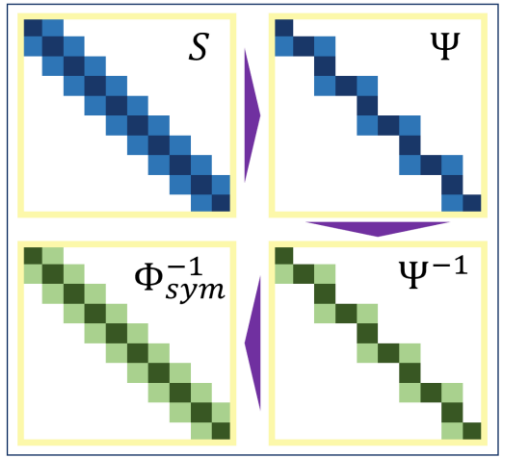

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization




Xueyi Bu[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



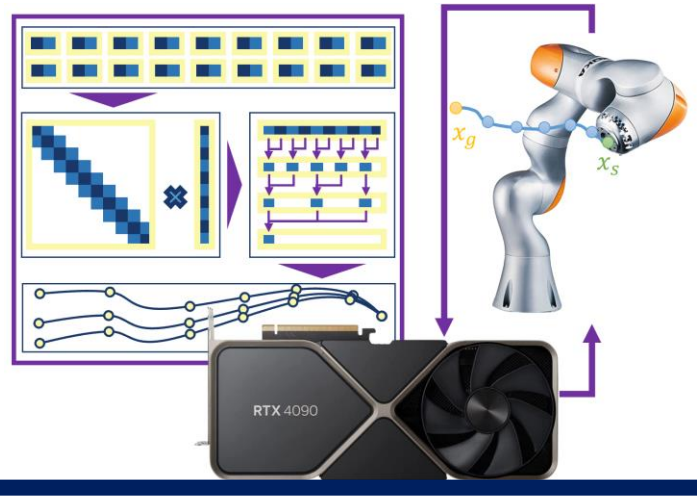
ICRA2024
May 13-17, Yokohama, Japan

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.
*These authors contributed equally to this work.

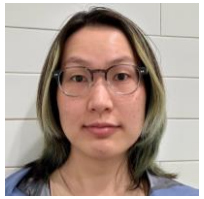
MPC**GPU**: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient **on the GPU**



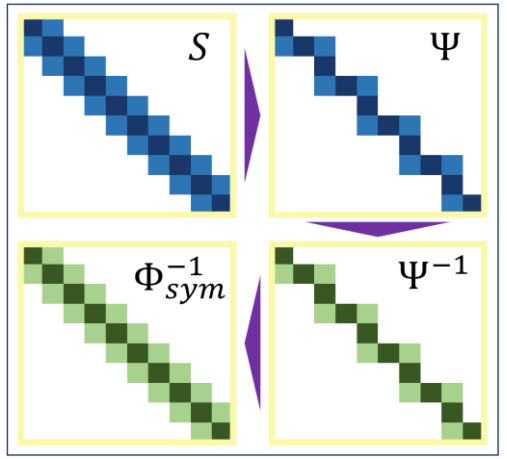

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



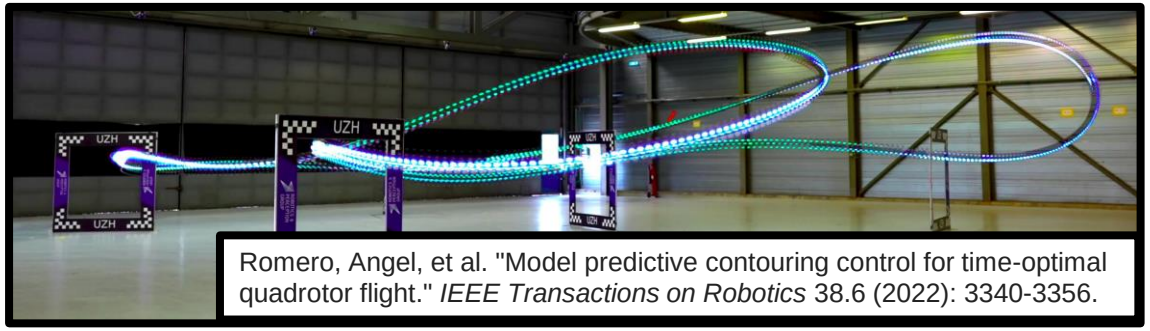
Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization



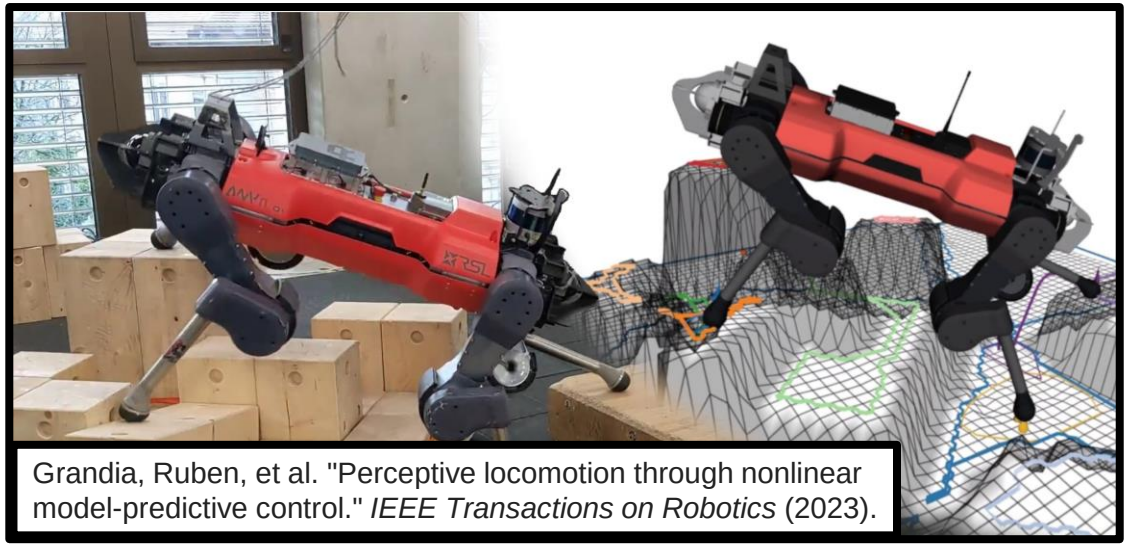

Xueyi Bu[1], Brian Plancher[2]
1: School of Engineering and Applied Science, Columbia University
2: Barnard College, Columbia University



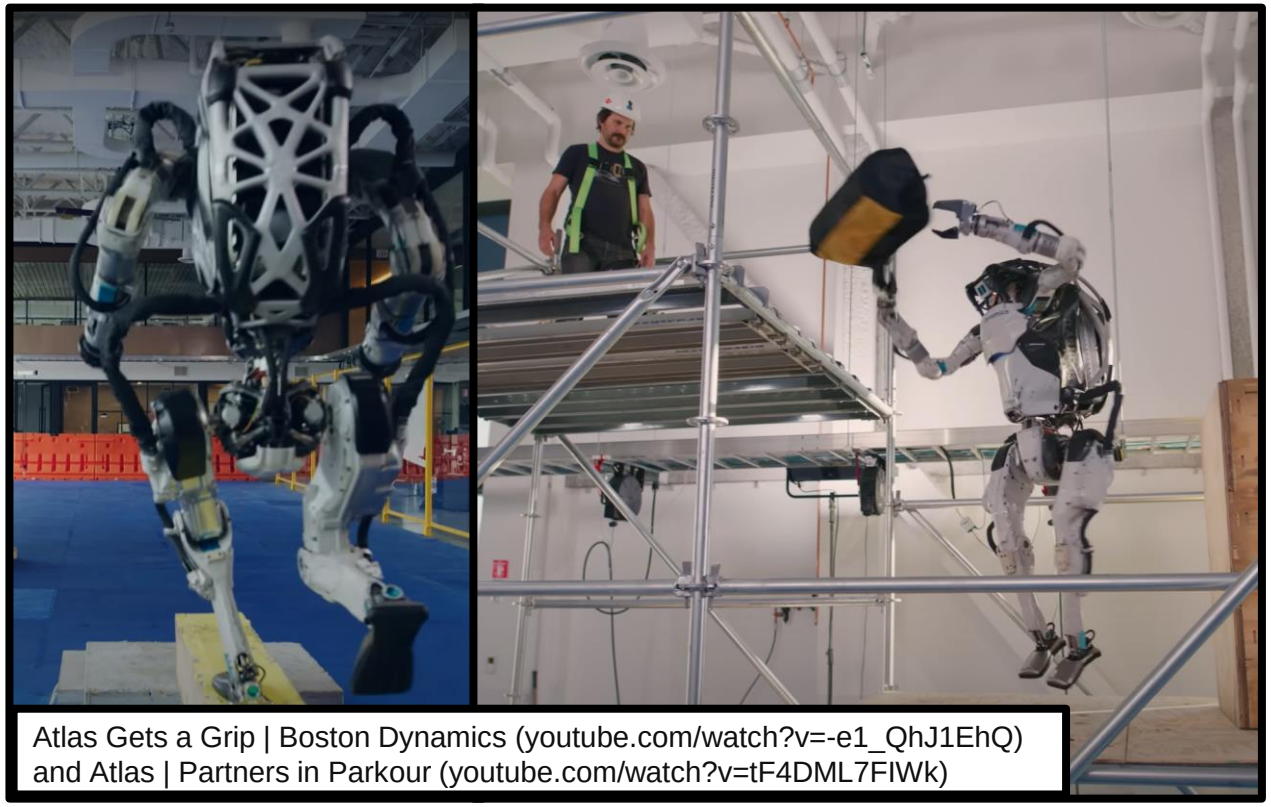
Model Predictive Control (MPC): State-of-the-Art Robot Motion Planning and Control



Romero, Angel, et al. "Model predictive contouring control for time-optimal quadrotor flight." *IEEE Transactions on Robotics* 38.6 (2022): 3340-3356.



Grandia, Ruben, et al. "Perceptive locomotion through nonlinear model-predictive control." *IEEE Transactions on Robotics* (2023).



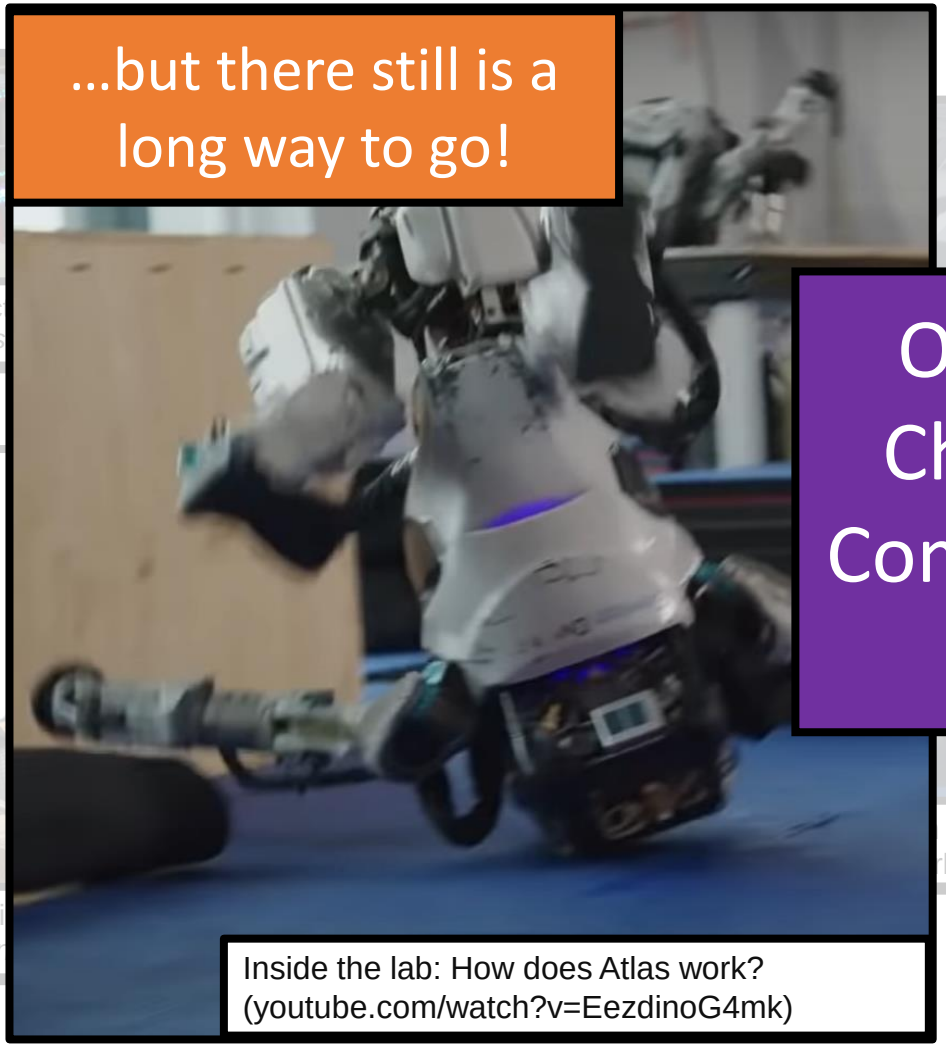
Atlas Gets a Grip | Boston Dynamics (youtube.com/watch?v=-e1_QhJ1EhQ) and Atlas | Partners in Parkour (youtube.com/watch?v=tF4DML7FIWk)

Model Predictive Control (MPC): State-of-the-Art Robot Motion Planning and Control

Romero, Angel, et al. "Model predic
quadrotor flight." *IEEE Transactions*

Grandia, Ruben, et al. "Perceptive locomoti
model-predictive control." *IEEE Transactio*

…but there still is a long way to go!

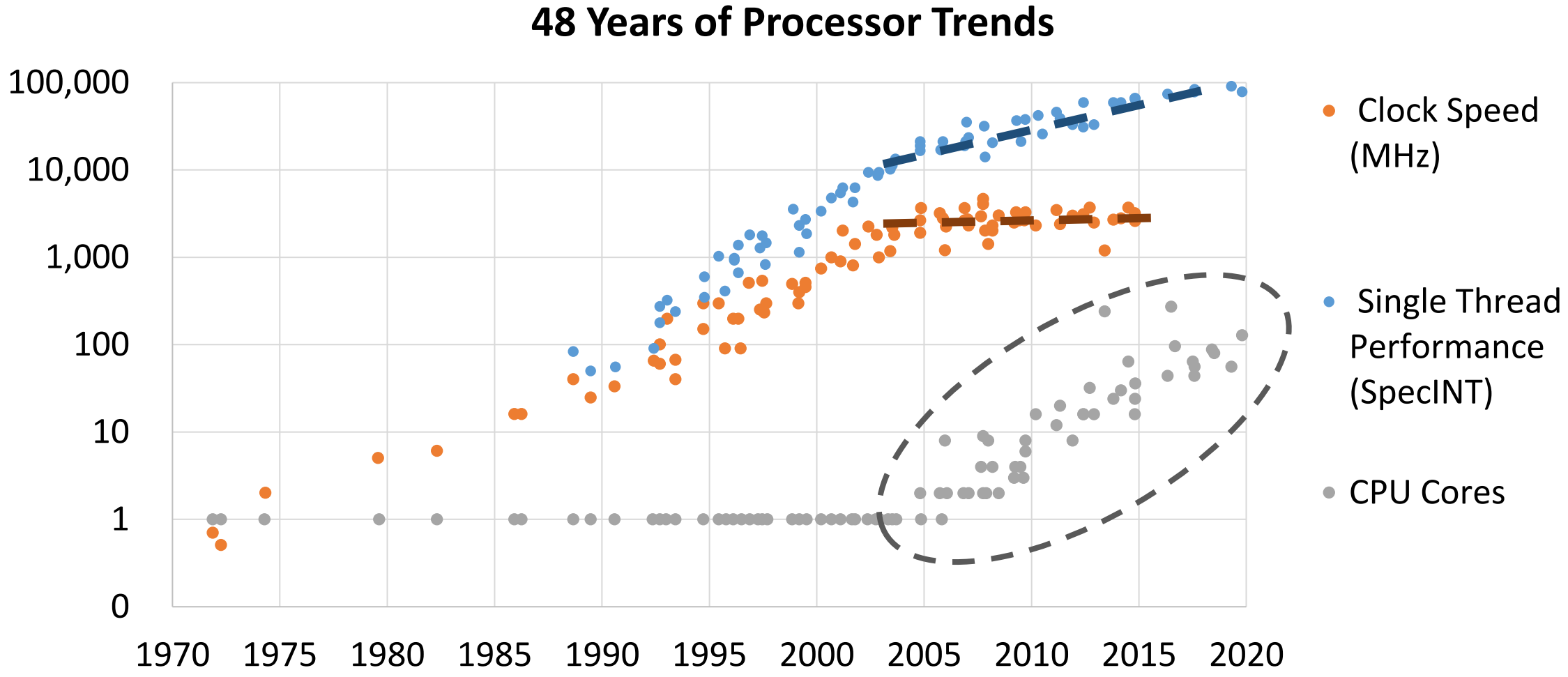


Inside the lab: How does Atlas work? (youtube.com/watch?v=EezdinoG4mk)

One Major Challenge – Computational Speed!

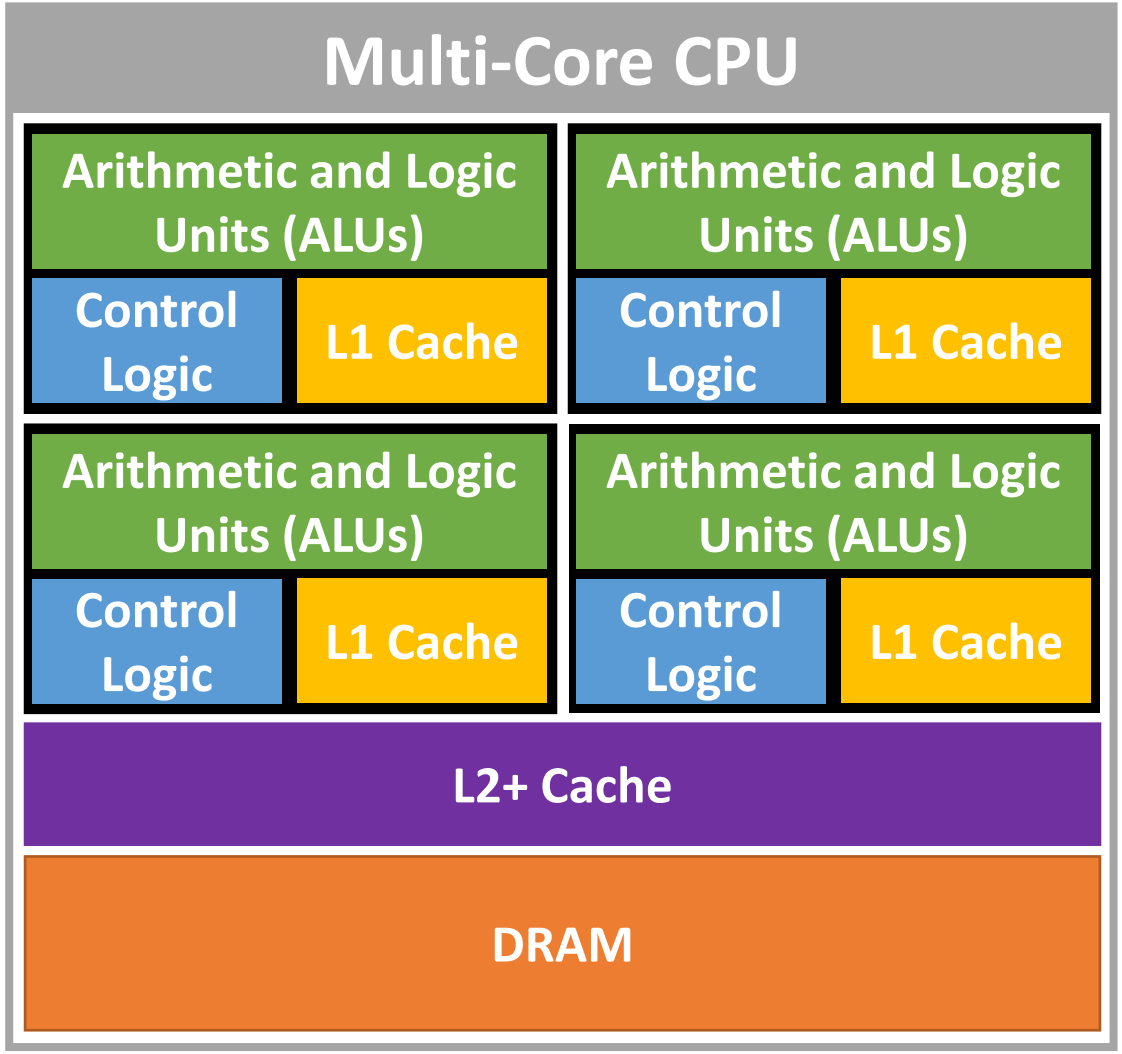
Dynamics (youtube.com/watch?v=-e1_QhJ1EhQ)
kour (youtube.com/watch?v=tF4DML7FIWk)

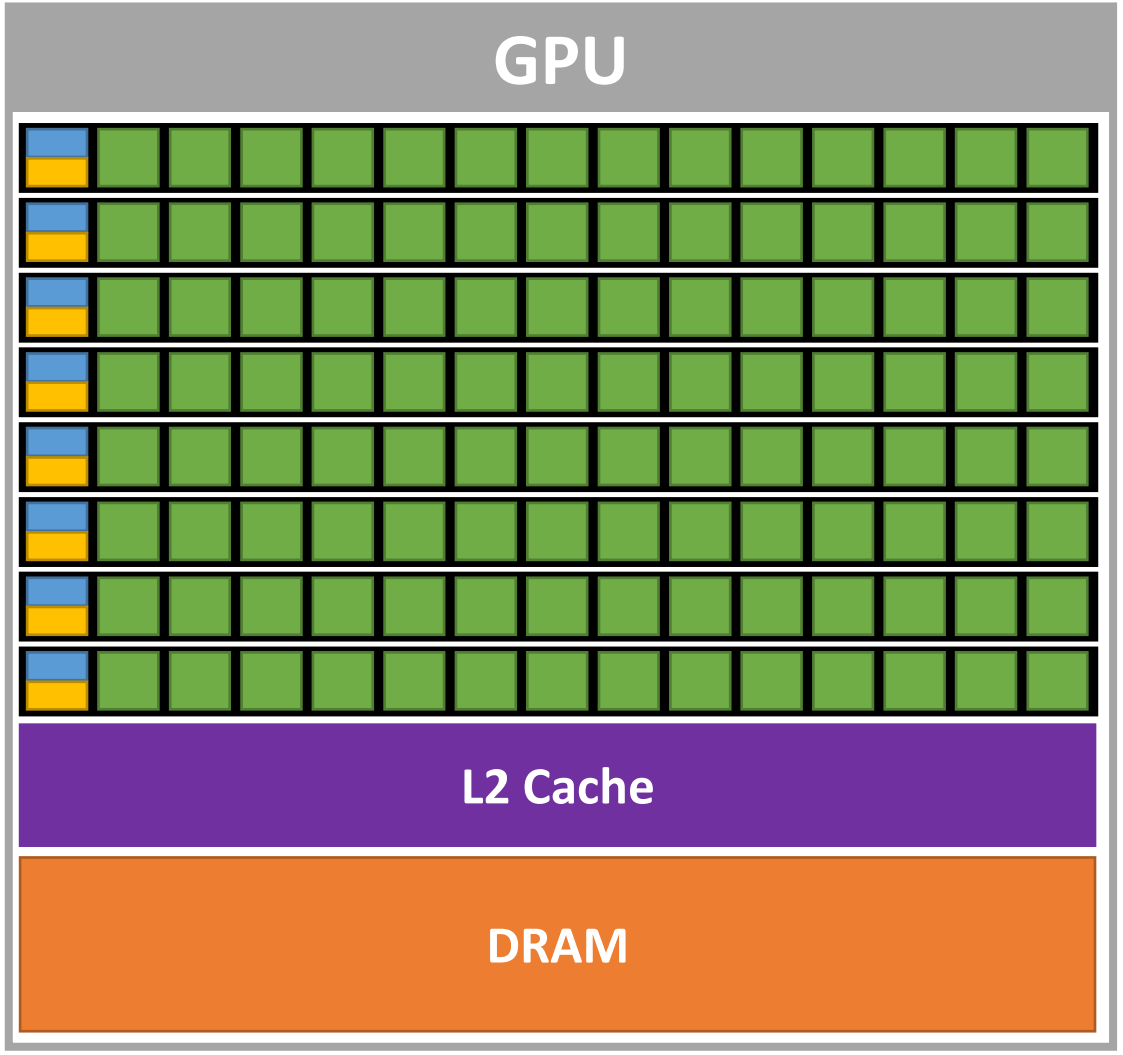
...resulting in a need to leverage parallelism!



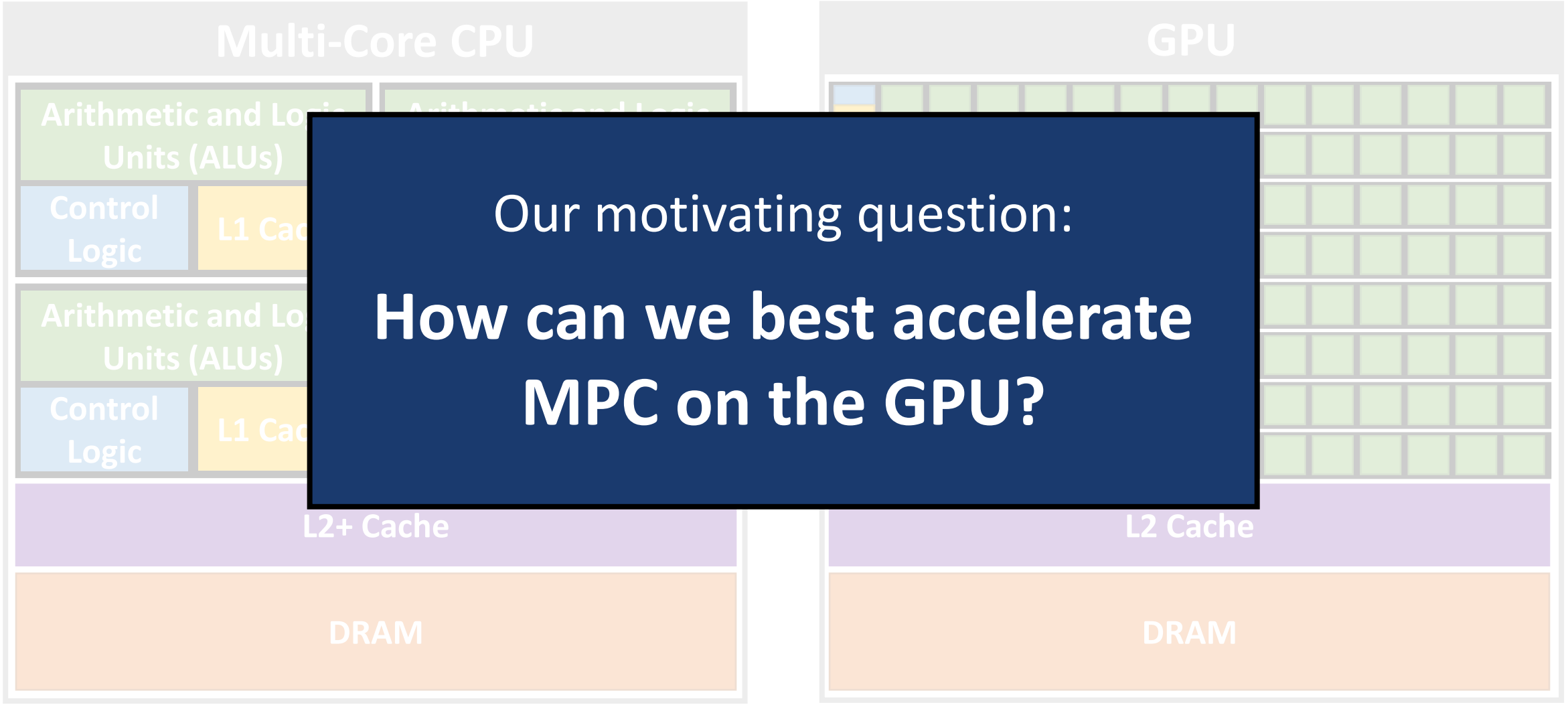
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp https://github.com/karlrupp/microprocessor-trend-data

GPUs offer increased computational parallelism and are natural choice to explore to accelerate MPC





GPUs offer increased computational parallelism and are natural choice to explore to accelerate MPC



Our motivating question:

How can we best accelerate MPC on the GPU?

Model Predictive Control (MPC): Via Successive Convex Optimization

Fully Separable Across Time!

Successive Convex Optimization

While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem

Direct Transcription

$$\min_{x_0,u_0 \ldots x_{N-1},u_{N-1},x_N} l_f(x_N) + \sum_{k=0}^{N-1} l(x_k, u_k)$$

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

$g(x_k, u_k) \geq 0 \ \forall k$

Direct Transcription QP (Taylor expansion)

$$\min_{x_0,u_0 \ldots x_{N-1},u_{N-1},x_N} \left(x_k - x_g\right)^T Q_N \left(x_k - x_g\right) + \sum_{k=0}^{N-1} \left(x_k - x_g\right)^T Q \left(x_k - x_g\right) + u_k^T R u_k$$

subject to: $x_0 - x_s = 0$

$x_{k+1} - A_k x_k - B_k u_k = 0 \ \forall k \in [0, N)$

Model Predictive Control (MPC): Via Successive Convex Optimization

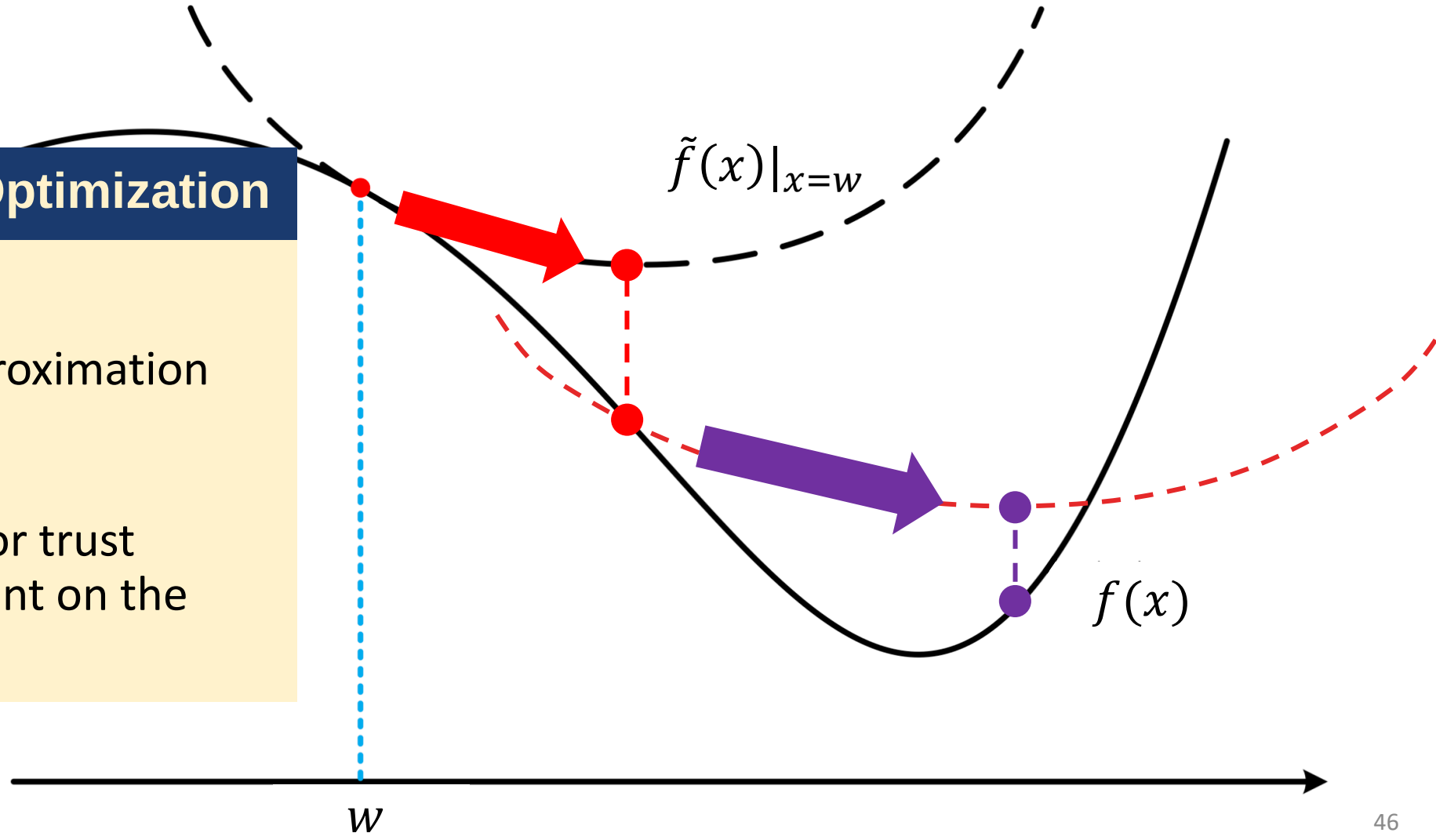
Successive Convex Optimization

While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem



Model Predictive Control (MPC): Via Successive Convex Optimization

Parallel Line Search Techniques

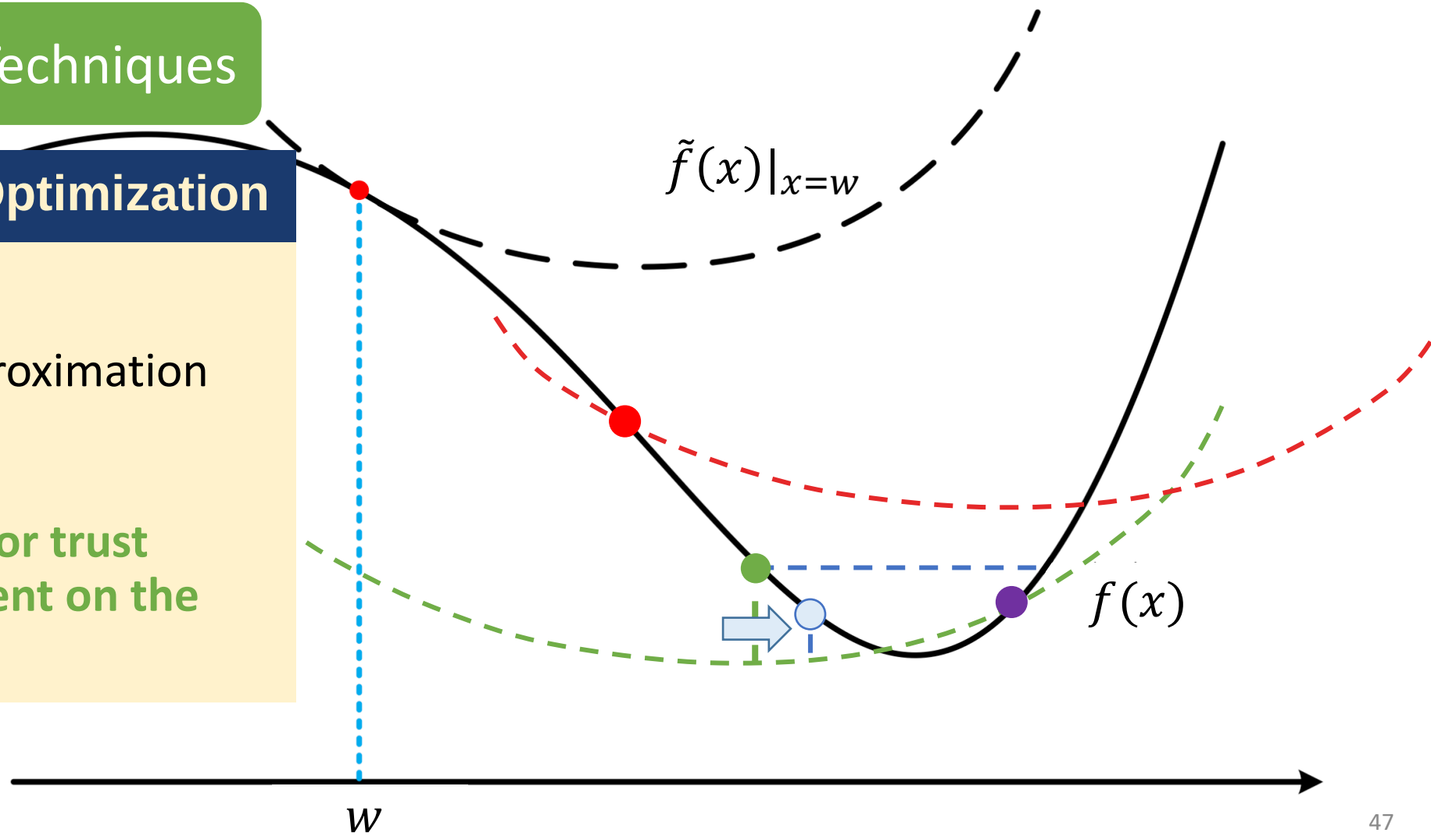
Successive Convex Optimization

While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem



1. Introduction and Motivation
2. Background and Related Work
3. **GBD-PCG** **A GPU-Optimized Linear System Solver for Trajectory Optimization Problems**
4. MPCGPU
5. Experiments and Results

Model Predictive Control (MPC): Via Successive Convex Optimization

Successive Convex Optimization

While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem

Solve the KKT System

$$\begin{bmatrix} G & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} -\delta Z \\ \lambda \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix}$$

Iterative Methods like the **Preconditioned Conjugate Gradient** Algorithm are Parallel Friendly

We can use the Symmetric Stair Preconditioner and the Schur Complement!

We can solve the KKT system with a GPU-parallel-friendly preconditioned iterative method

Preconditioned Conjugate Gradient

$r = \gamma - S\lambda$

$\tilde{r}, p = \Phi^{-1} r$

while $r < \epsilon$

$\alpha = \eta \,/\, p^T S p, \quad \eta = r^T \tilde{r}$

$\lambda \mathrel{+}= \alpha p, \quad r \mathrel{-}= \alpha S p$

$\tilde{r} = \Phi^{-1} r$

$\beta = r^T \tilde{r} \,/\, \eta$

$p = \tilde{r} + \beta p$

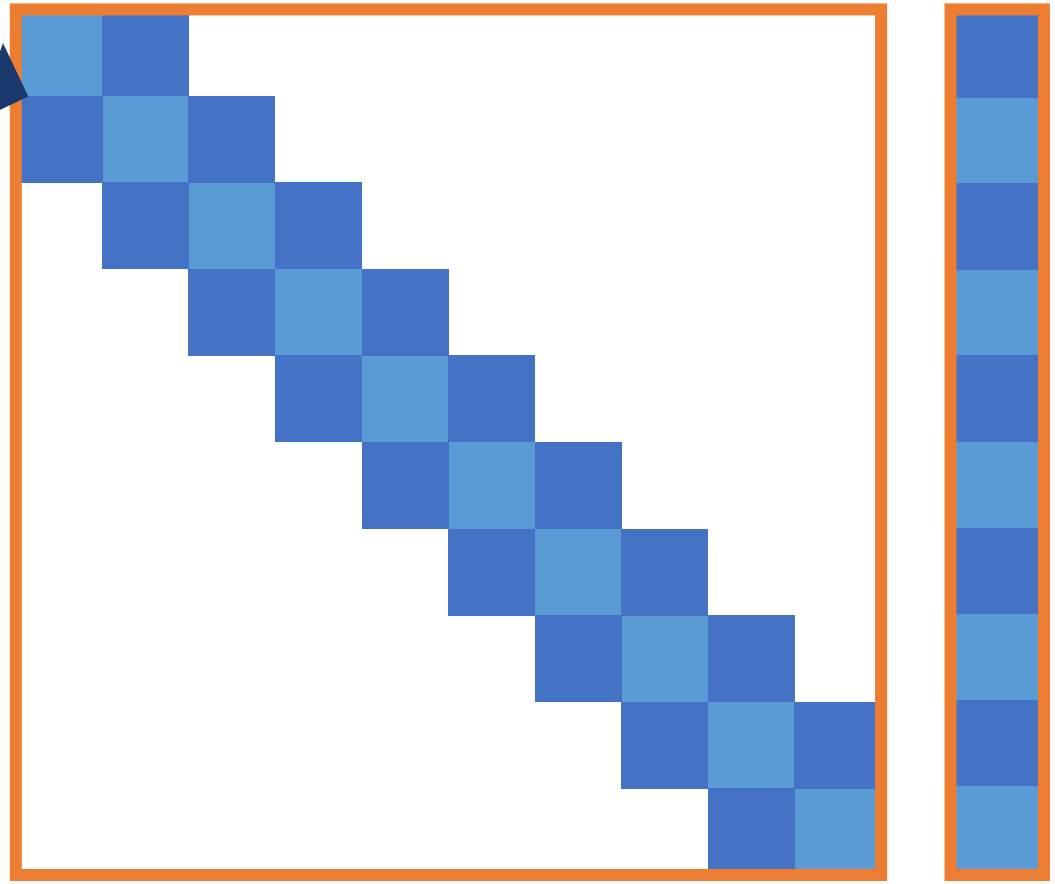
Computationally dominated by matrix-vector products!
(and reductions)

And we can refactor it to make it more parallel!

Algorithm 1: Preconditioned Conjugate Gradient (PCG)
$(S, \Phi^{-1}, \gamma, \lambda, \epsilon) \rightarrow \lambda^*$

$r = \gamma - S\lambda$
$\tilde{r}, p = \Phi^{-1} r$
$\eta = r^T \tilde{r}$
for iter $i = 1$: max_iter **do**
  $\alpha = \eta / (p^T S p)$
  $r = r - \alpha S p$
  $\lambda = \lambda + \alpha p$
  $\tilde{r} = \Phi^{-1} r$
  $\eta' = r^T \tilde{r}$
  if $\eta' < \epsilon$ **then** return λ
  $\beta = \eta' / \eta$
  $p = \tilde{r} + \beta p$
  $\eta = \eta'$
return λ

Each **Block-Row** is a mostly independent operation and you only ever have to copy to your neighbors!



And we can refactor it to make it more parallel!

Algorithm 1: Preconditioned Conjugate Gradient (PCG)
$(S, \Phi^{-1}, \gamma, \lambda, \epsilon) \rightarrow \lambda^*$

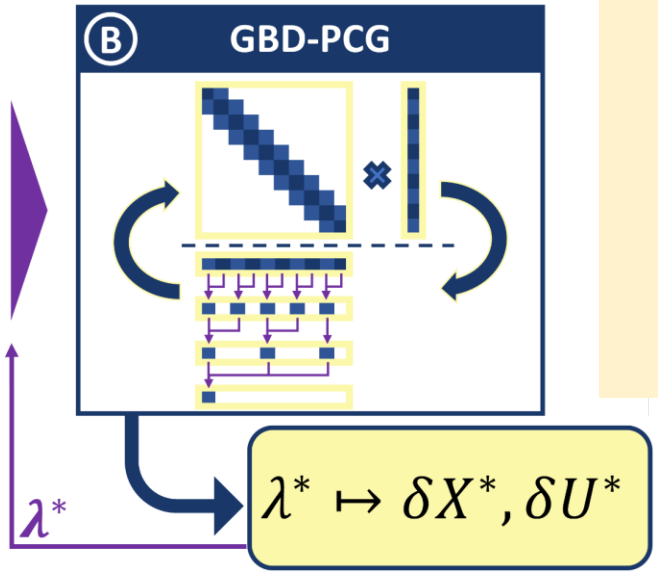
1: $r = \gamma - S\lambda$
2: $\tilde{r}, p = \Phi^{-1} r$
3: $\eta = r^T \tilde{r}$
4: **for** iter $i = 1$: max_iter **do**
5: $\quad \alpha = \eta / (p^T S p)$
6: $\quad r = r - \alpha S p$
7: $\quad \lambda = \lambda + \alpha p$
8: $\quad \tilde{r} = \Phi^{-1} r$
9: $\quad \eta' = r^T \tilde{r}$
10: $\quad$ **if** $\eta' < \epsilon$ **then** return λ
11: $\quad \beta = \eta' / \eta$
12: $\quad p = \tilde{r} + \beta p$
13: $\quad \eta = \eta'$
14: return λ

Initialization

Main Loop

17: **for** block $b = 0 : N$ **in parallel do**
18: $\quad$ Load r_{b-1}, r_{b+1}
19: $\quad \tilde{r}_b = \Phi_b^{-1} r_{b-1:b+1}$
20: $\quad \eta'_b = r_b^T \tilde{r}_b$
21: $\eta' =$ **ParallelReduce**(η'_b)

MPCGPU: A 3-Step Method for GPU-Parallel-Friendly Direct Trajectory Optimization



Successive Convex Optimization

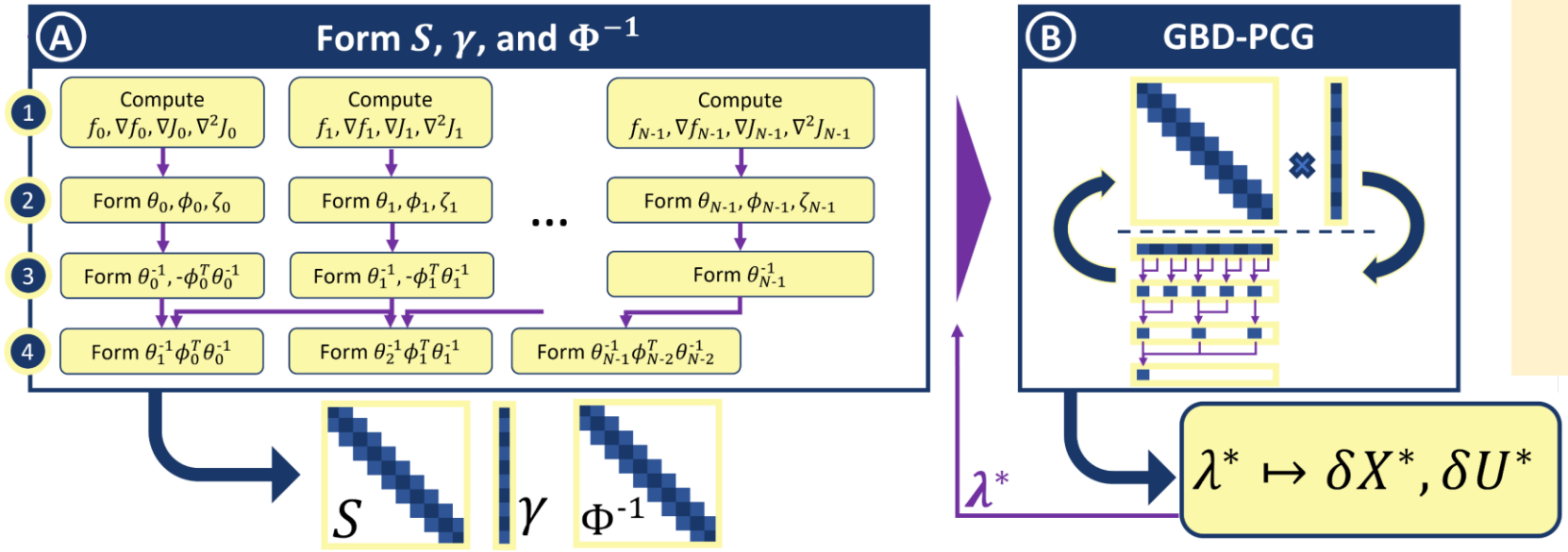
While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

3) Apply a line search (or trust region) to ensure descent on the original problem

MPCGPU: A 3-Step Method for GPU-Parallel-Friendly Direct Trajectory Optimization



Successive Convex Optimization

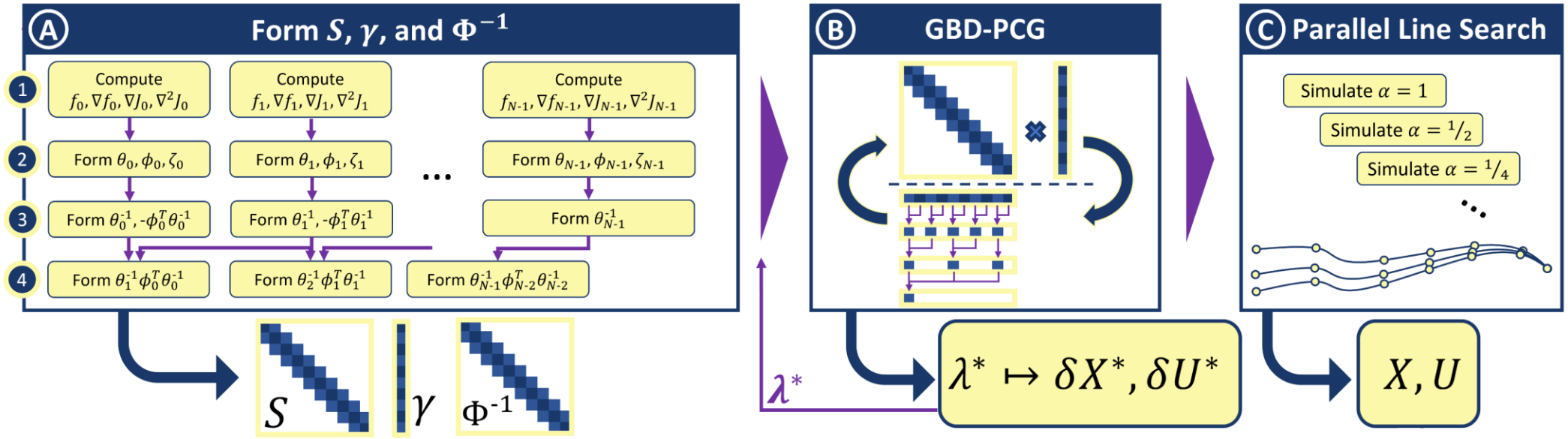
While (not converged):

1) Compute Taylor Approximation

2) Take Gradient Step

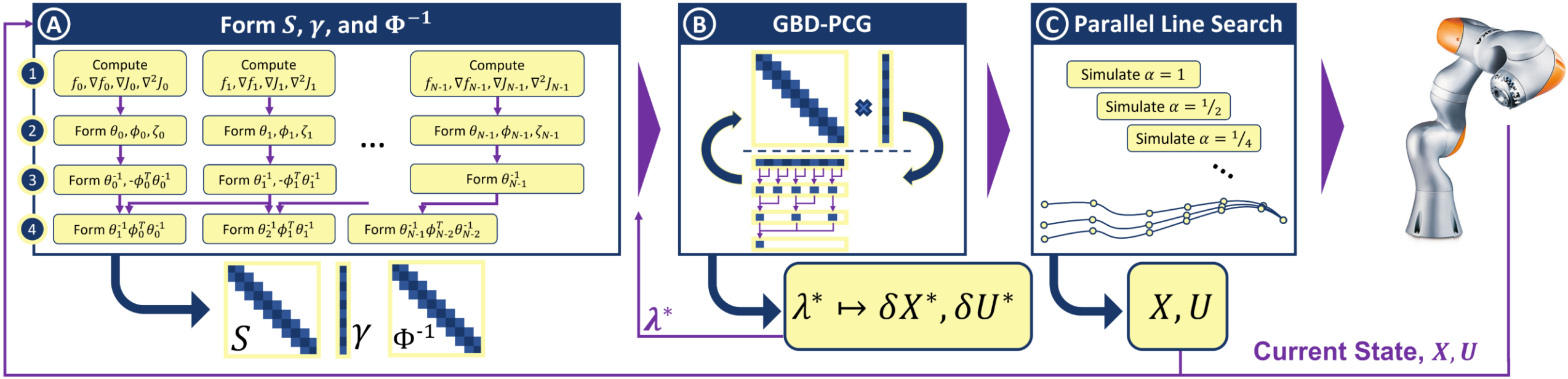
3) Apply a line search (or trust region) to ensure descent on the original problem

MPCGPU: A 3-Step Method for GPU-Parallel-Friendly Direct Trajectory Optimization



MPCGPU: A 3-Step Method for GPU-Parallel-Friendly Direct Trajectory Optimization

MPCGPU: Successive Convex Optimization Optimized for MPC on the GPU!



MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU

1. Introduction and Motivation
2. Background and Related Work
3. GBD-PCG
4. MPCGPU
5. **Experiments and Results**

We then tested MPCGPU on a 5-point pick-and-place circuit in simulation…

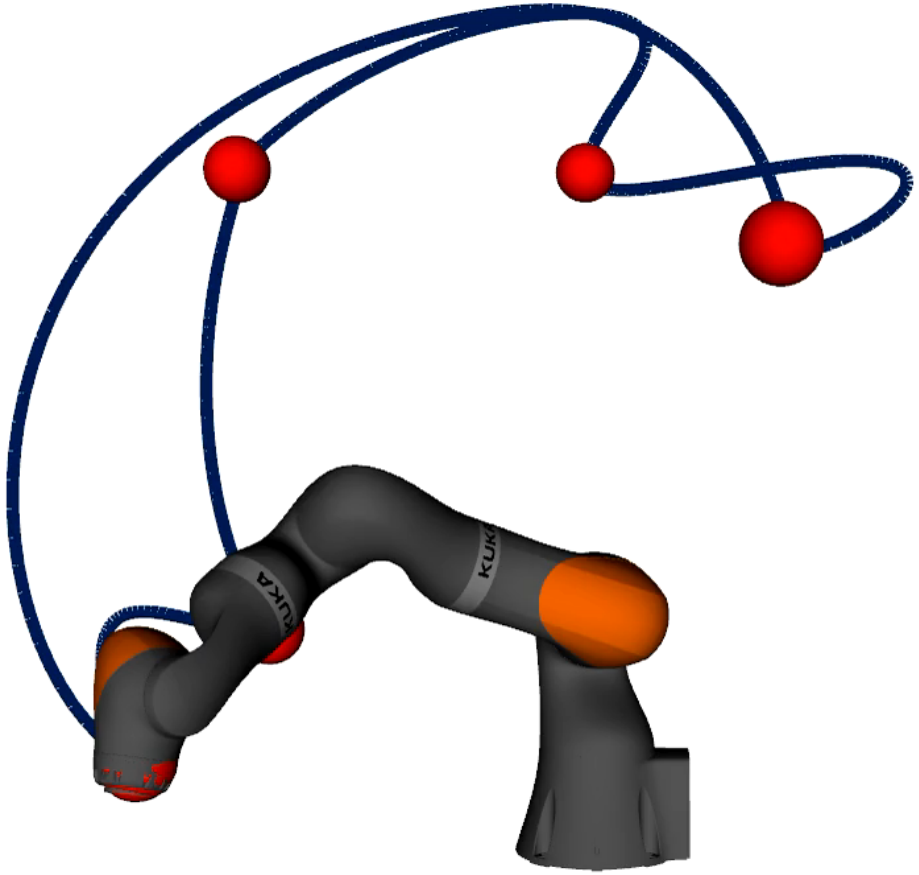
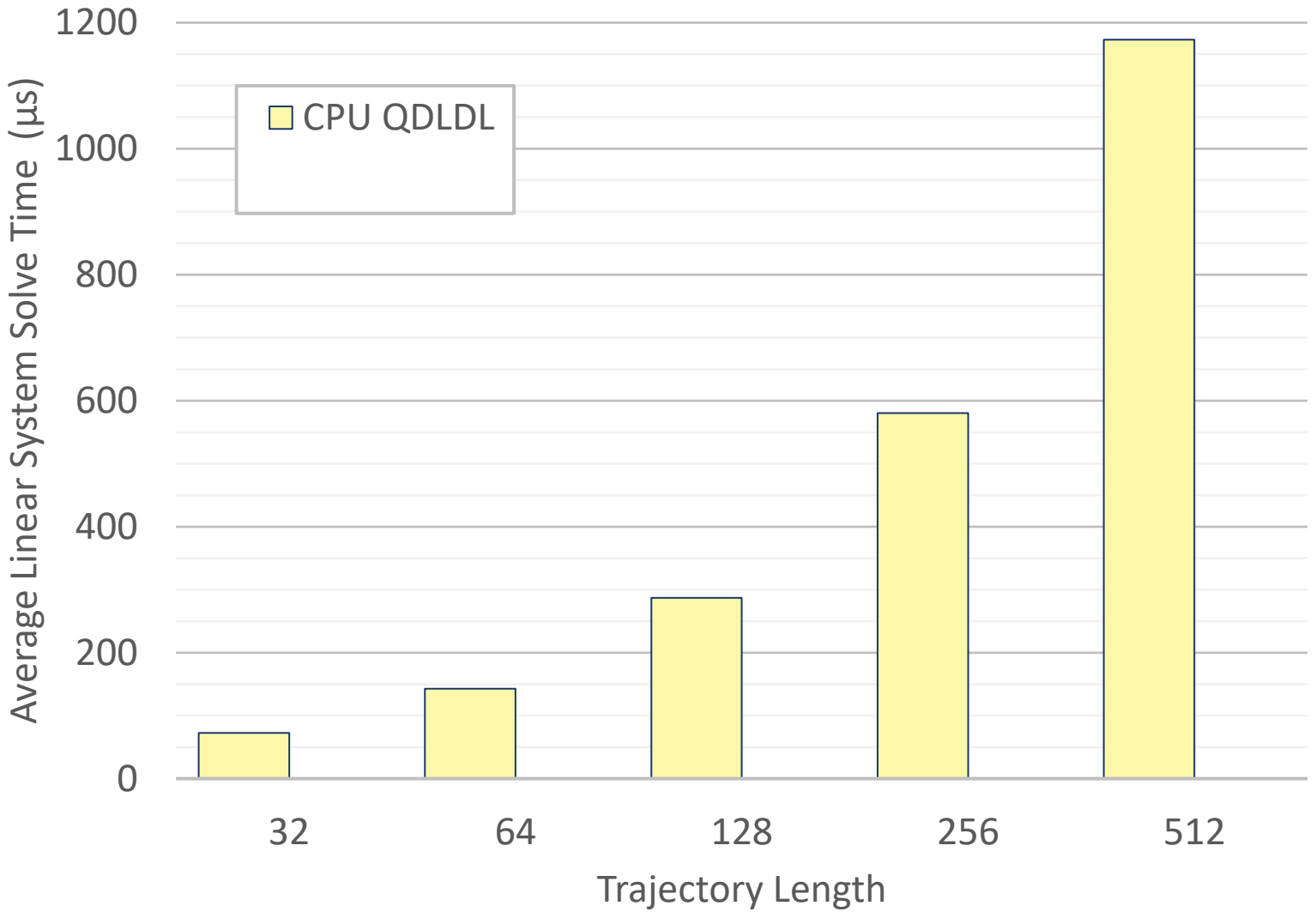


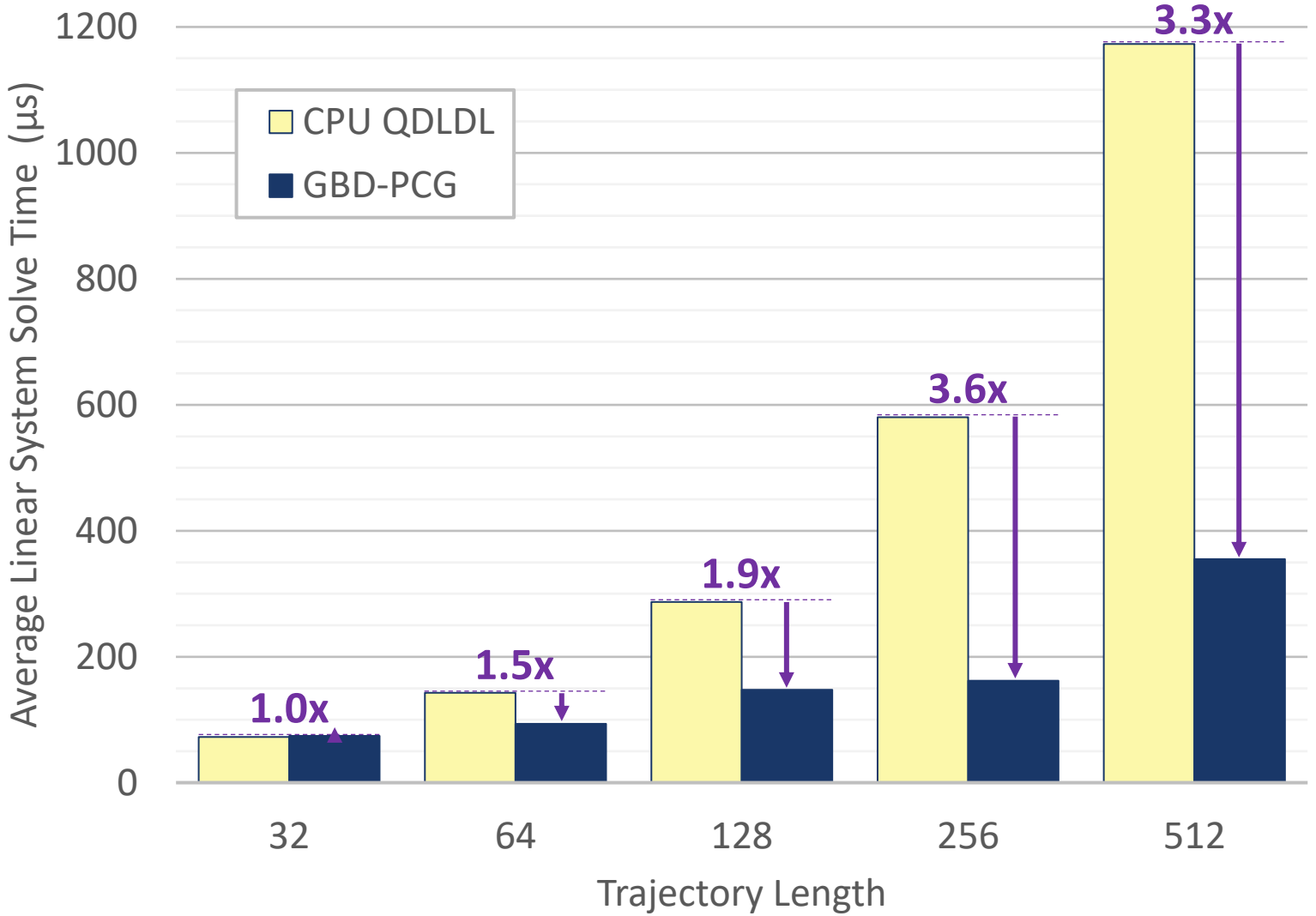
Image Source:
Crocoddyl Viewer

github.com/A2R-Lab/MPCGPU

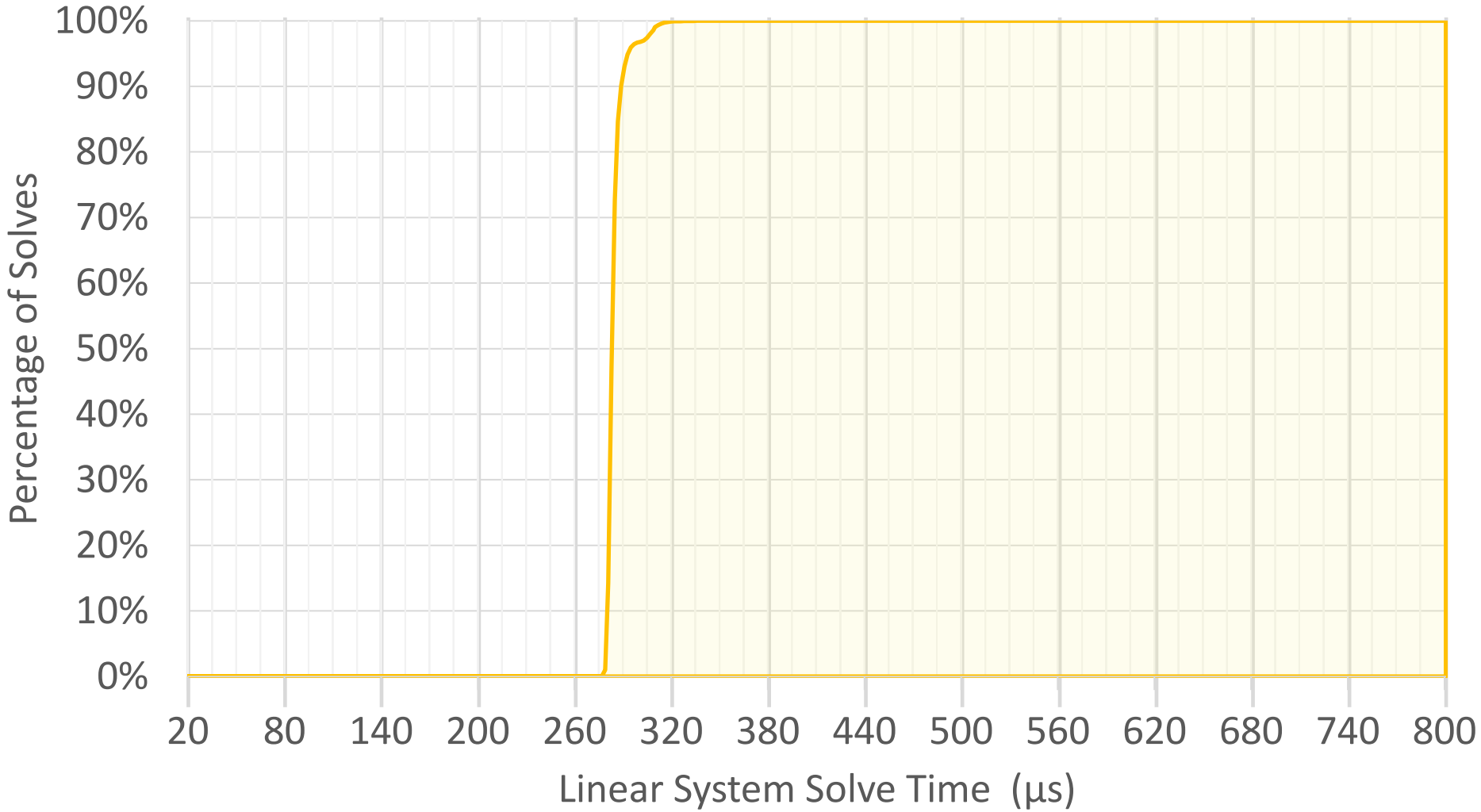
… resulting in up to a 3.6x average speedup in linear system solve time …



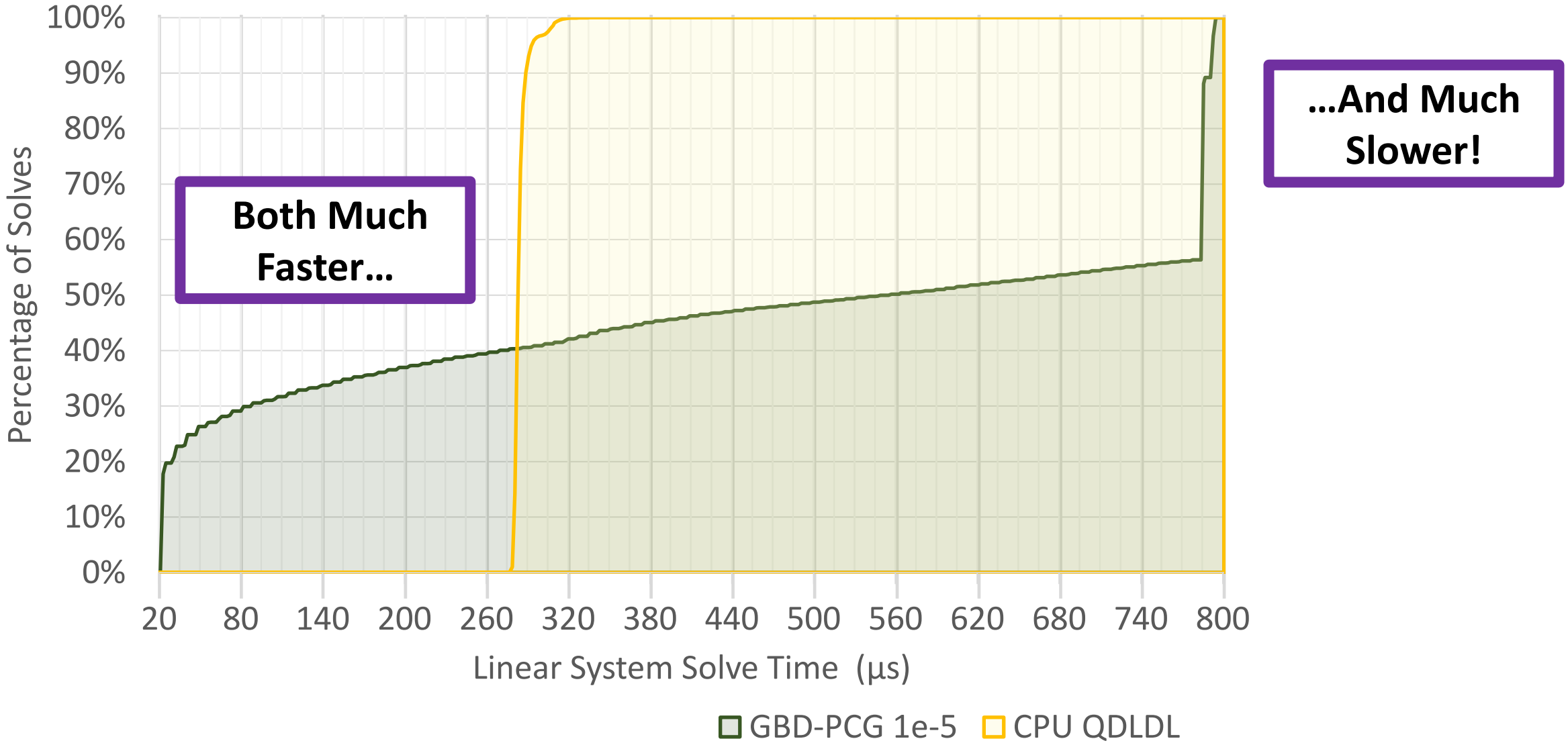
… resulting in up to a 3.6x average speedup in linear system solve time …



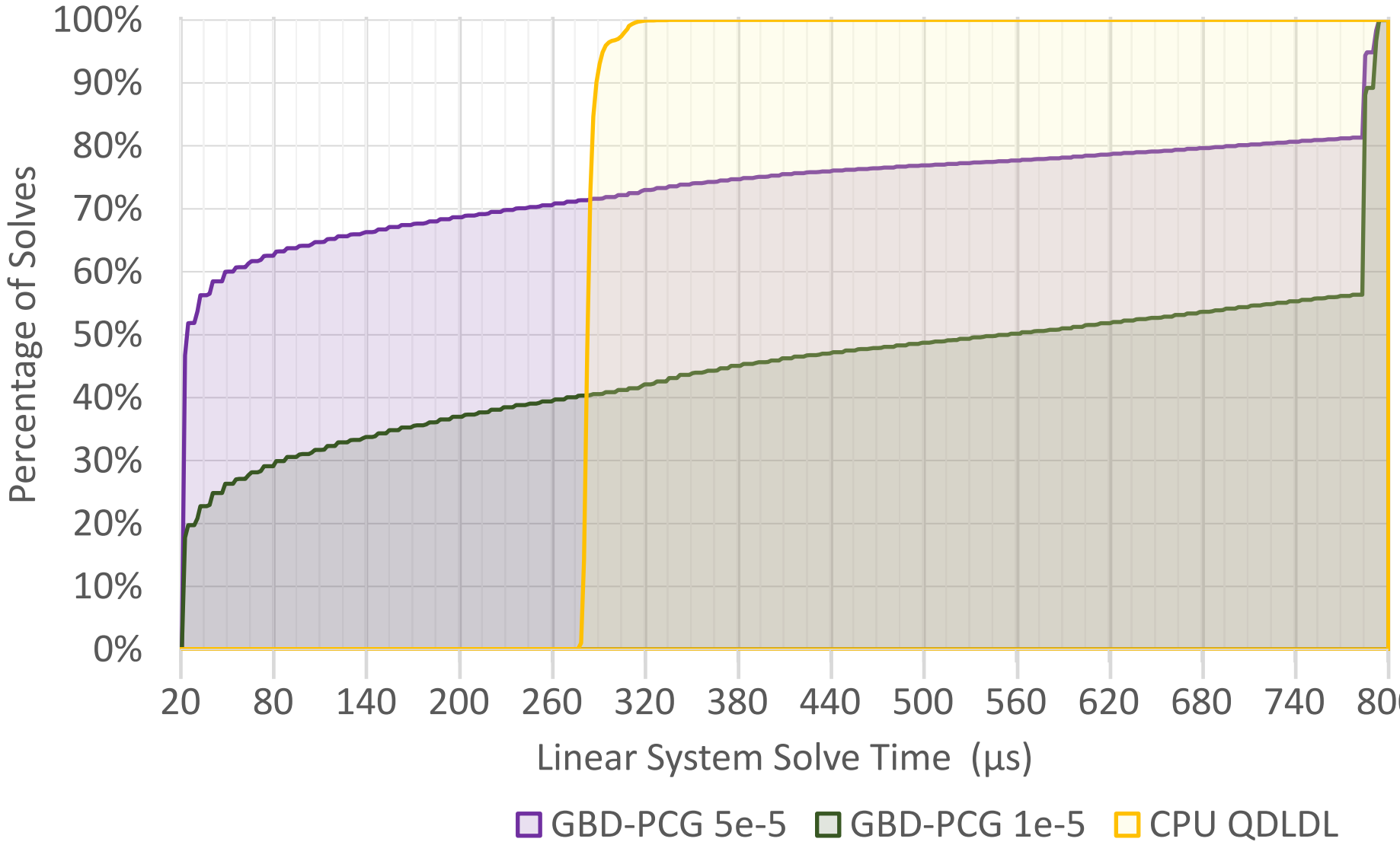
… and a best case order-of-magnitude speedup for the majority of solves!



… and a best case order-of-magnitude speedup for the majority of solves!

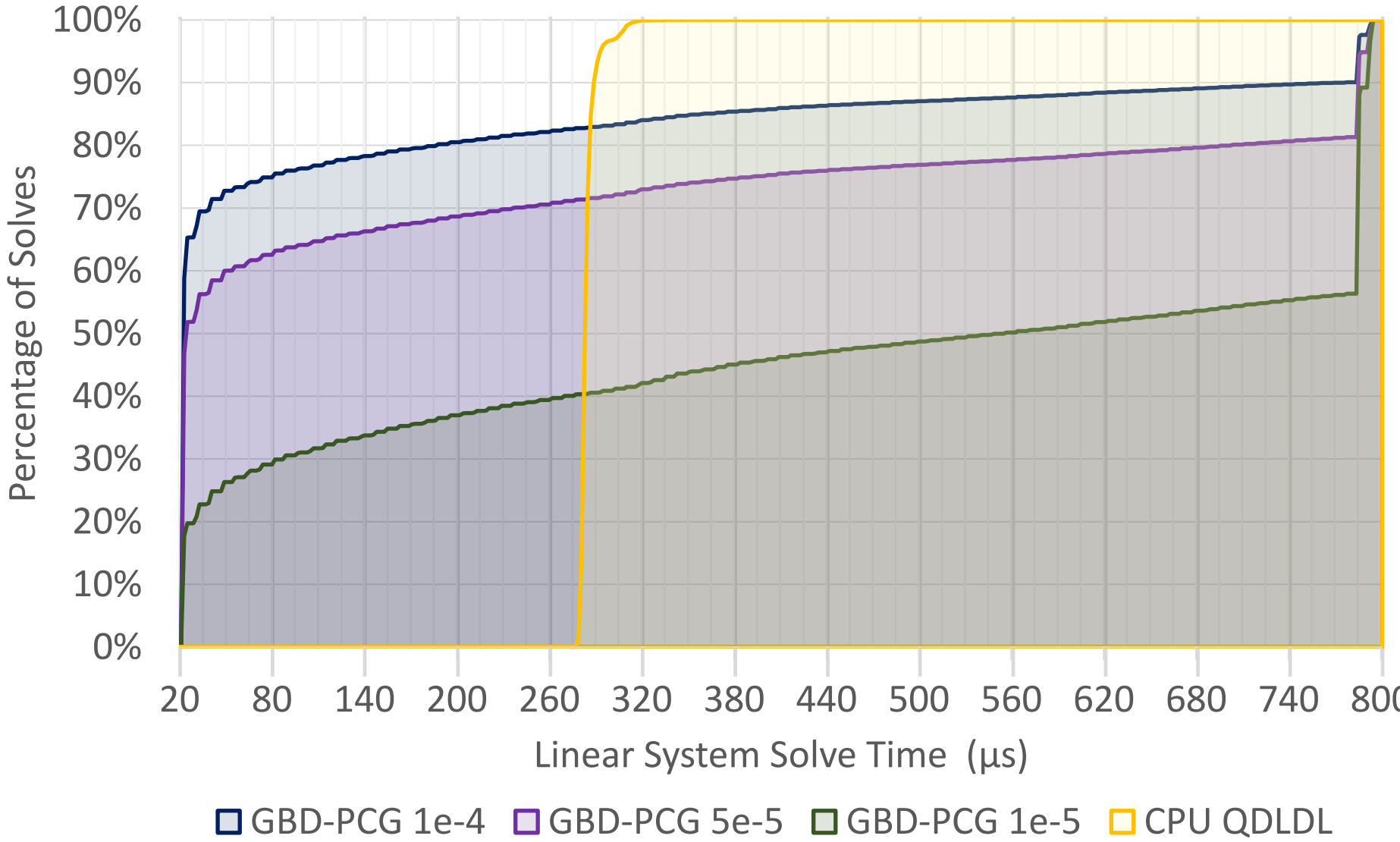


… and a best case order-of-magnitude speedup for the majority of solves!



A lower exit tolerance can greatly decrease solve time

… and a best case order-of-magnitude speedup for the majority of solves!



>65% of GBD-PCG solves are ≥10x faster than the fastest QDLDL solve

<10% of GBD-PCG solves are ≥2x slower, and the slowest is 2.5x slower, than the slowest QDLDL solve

MPCGPU scales to…

MPCGPU with GBD-PCG		Knot Points				
		32	64	128	256	512
Control Rate	250Hz	Number of **Loop Iterations** of Successive Convex Optimization				
	500Hz					
	1kHz					

MPCGPU scales to…

MPCGPU with GBD-PCG		Knot Points				
		32	64	128	256	512
Control Rate	250Hz	22.2	19.7	15.4	5.2	4.4
	500Hz	10.3	Number of **Loop Iterations** of Successive Convex Optimization			
	1kHz	4.9				

MPCGPU scales to 512 knot points at 1kHz and a per-iteration rate of 4kHz for 128 knot points at 500Hz

MPCGPU with GBD-PCG		Knot Points				
		32	64	128	256	512
Control Rate	250Hz	22.2	19.7	15.4	5.2	4.4
	500Hz	10.3	10.6	8.0	4.6	3.0
	1kHz	4.9	5.2	3.7	2.4	1.7

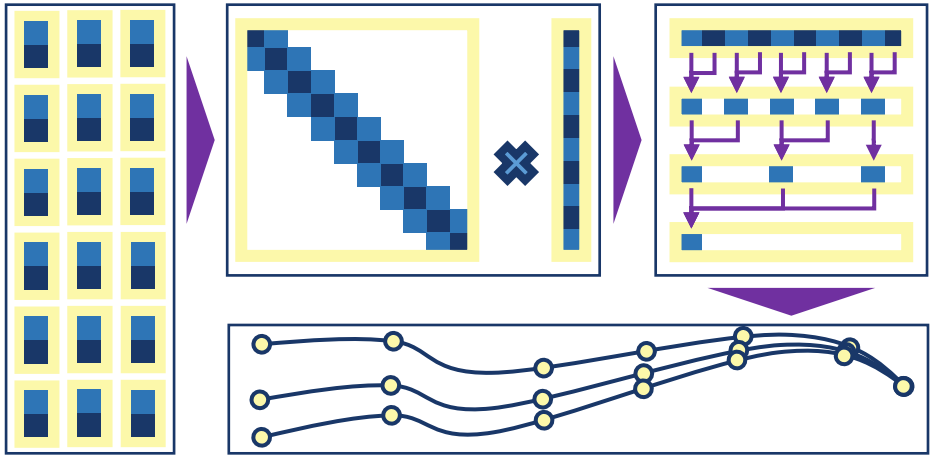
MPCGPU: Real-Time Nonlinear Model Predictive Control through Preconditioned Conjugate Gradient on the GPU

Symmetric Stair Preconditioning of Linear Systems for Parallel Trajectory Optimization

1. We leverage the improved conditioning of the symmetric stair preconditioner to build a fast parallel PCG solver which provides up to a 10x speedup for a majority of linear system solves and a 3.6x speedup on average

2. Through its GPU-first design MPCGPU scales to kilohertz control rates for NMPC with trajectories as long as 512 knot points

3. Warm-starting and Co-Design Matter!



The Barnard Accessible and Accelerated Robotics Lab

bplancher@barnard.edu a2r-lab.org

Emre Adabag[1], Miloni Atal*[1], William Gerard*[1], Brian Plancher[2] Xueyi Bu[1], Brian Plancher[2]

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

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. *These authors contributed equally to this work.