Parallel Dynamics Computation

Jia Pan

https://sites.google.com/site/panjia/

Robot dynamics

- Inverse dynamics $F = ma$
- Forward dynamics $a = F/m$

Inverse dynamics

- Given all joint positions q_i, velocities $\dot{q}_i$, and accelerations $\ddot{q}_i$, and end-effector external force F
- Compute all joint torques τ_i
- $\tau_i = ID(q_i, \dot{q}_i, \ddot{q}_i, F)$

Forward dynamics

- Given all joint positions q_i, velocities $\dot{q}_i$, and torques τ_i, and end-effector external force F
- Compute all joint accelerations $\ddot{q}_i$
- $\ddot{q}_i = FD(q_i, \dot{q}_i, \tau_i, F)$

Inverse dynamics & motion planning

- Given a planned or desired trajectory of a robot
- Inverse dynamics compute the joint torques required to achieve / execute the trajectory

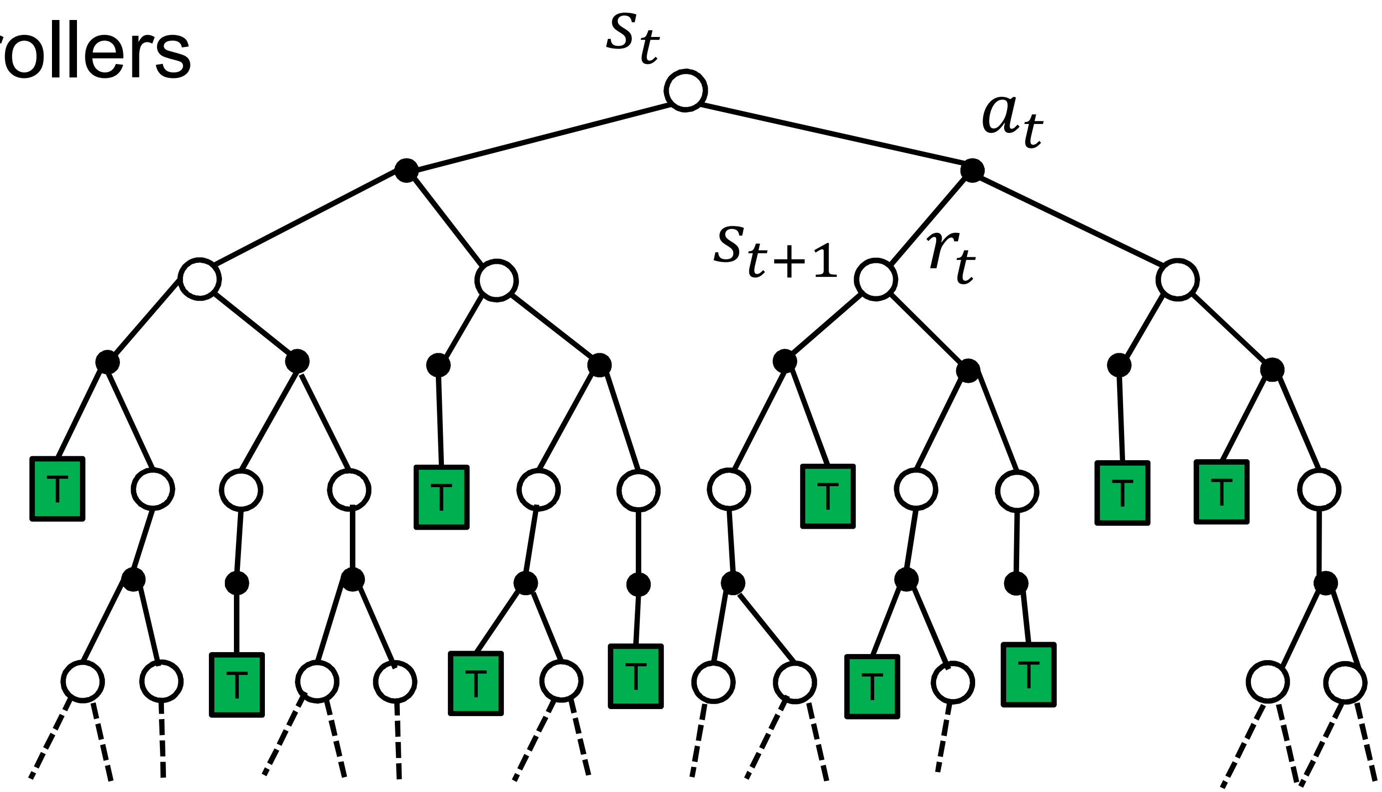
Forward dynamics & motion simulation

- Given a sequence of torque control commands at different time steps $\tau_i^0, \tau_i^1, \ldots, \tau_i^T$
- $\ddot{q}_i^t = FD(q_i^t, \dot{q}_i^t, \tau_i^t, F)$
- Solve the ODE $\frac{d}{dt}\begin{pmatrix} q_i^t \\ \dot{q}_i^t \end{pmatrix} = \begin{pmatrix} \dot{q}_i^t \\ \ddot{q}_i^t \end{pmatrix}$

Forward dynamics & optimal control

- Control optimization (model-based reinforcement learning) needs many forward dynamics simulations
  - Automatically construct many candidate controllers, evaluate their performance in simulation, and use the data to construct better controllers
  - Controller evaluation needs huge number of rollout calls
  - More than 2000,000,000 evaluations

Evaluating a single controller or multiple controllers

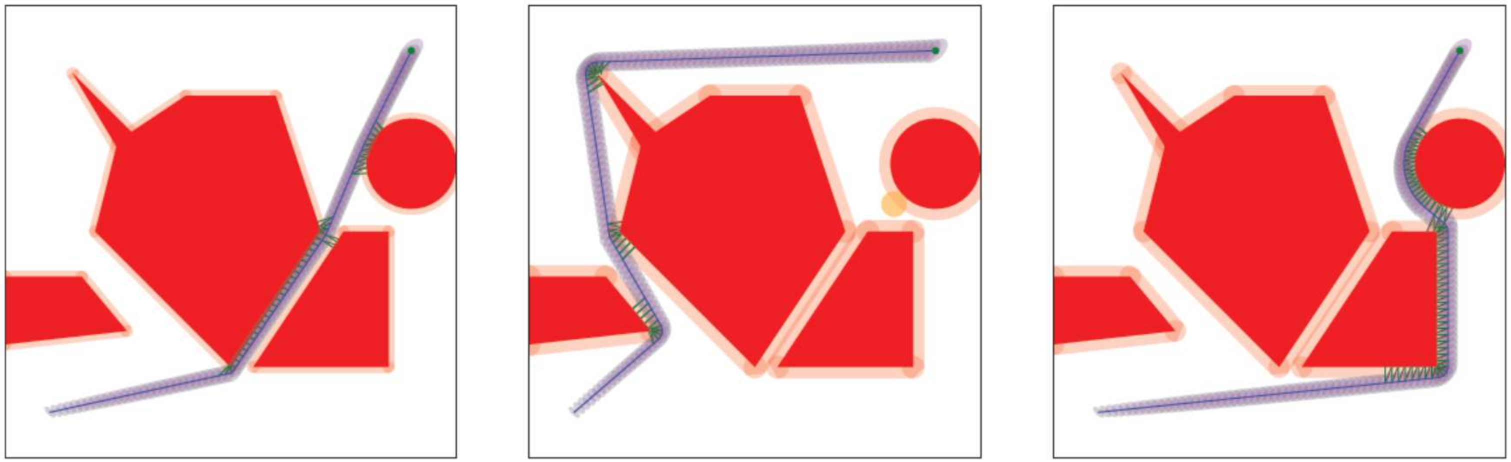


Forward dynamics & model-free RL approaches

- Testing RL algorithms on real-robots is expensive.
- A fast and correct simulator can save a lot of development time.

Forward dynamics & uncertainty-aware planning

- Systems have noise in actuation and localization
- Efficient forward dynamics can be used to estimate trajectory uncertainty

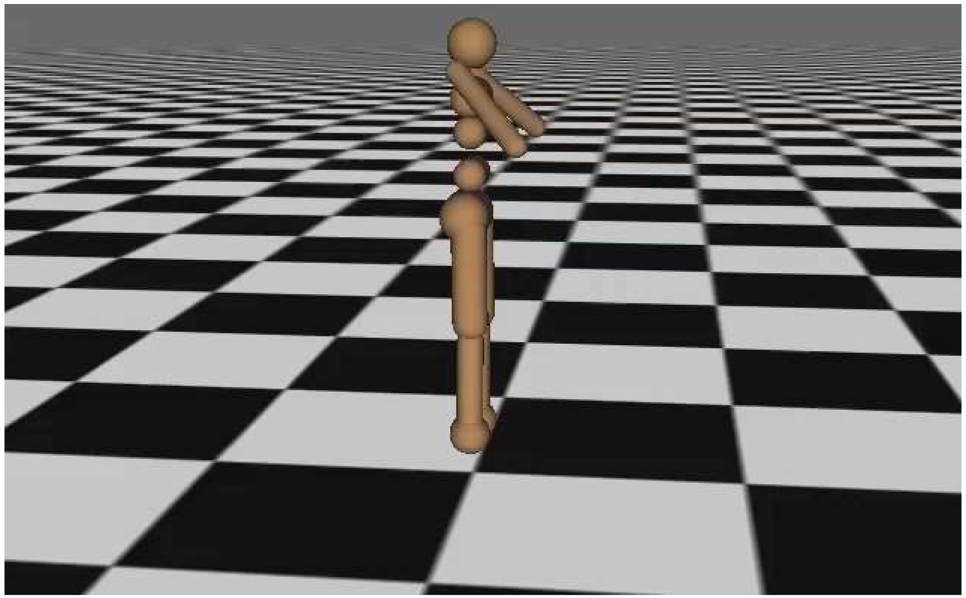


Key challenges for dynamics computations (I)

- Dynamics computation for a system with many links
  - Cloth simulation, rope simulation, soft robots

Key challenges for dynamics computations (II)

- Dynamics computation for a same system with many different instances of joint configurations
  - Deep reinforcement learning



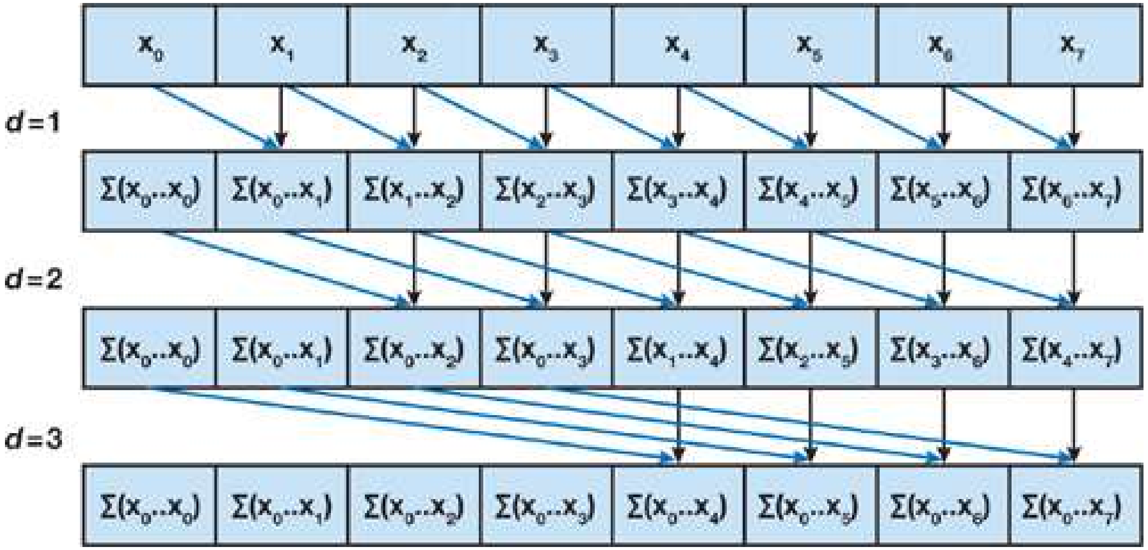
Our goal

- Provide very efficient dynamics solvers for both many-link and many-instance cases
- Based on GPU’s parallel scan operations

Parallel prefix sum (scan)

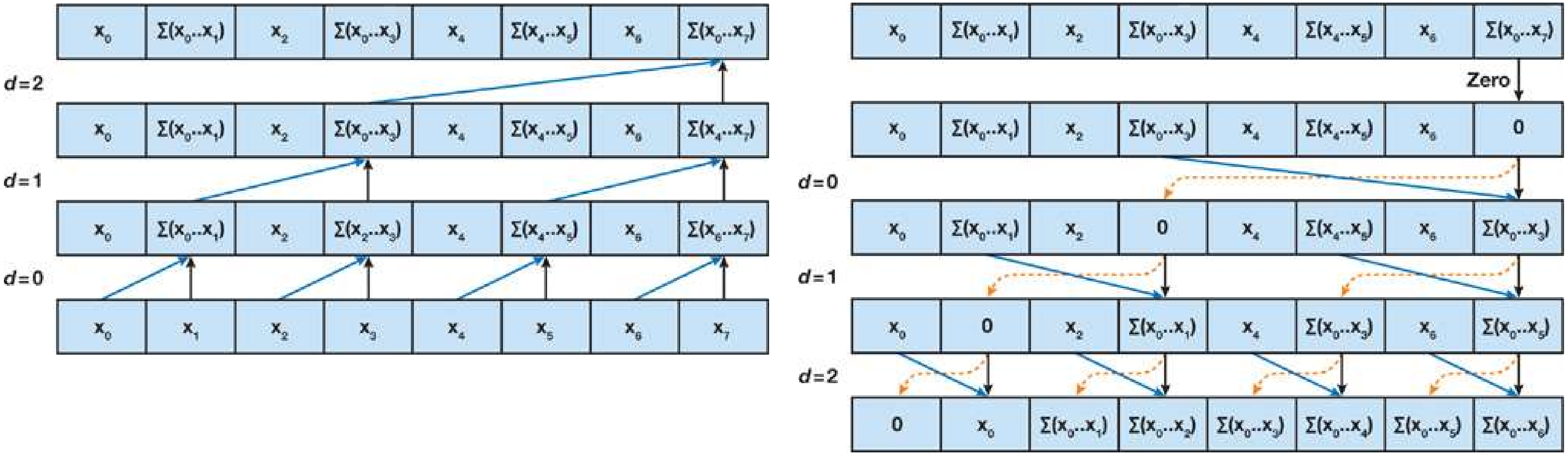
- Given an array $A = [a_0, a_1, \ldots, a_{n-1}]$and a binary associative operator $\oplus$
- $scan(A) = [a_0, a_0 \oplus a_1, \ldots, a_0 \oplus a_1 \oplus a_2 \ldots \oplus a_{n-1}]$
- Example:
  - if $\oplus$ is addition, then scan on the set $[3,1,7,0,4,1,6,3]$
  - Returns the set $[3,4,11,11,15,16,22]$

Naïve scan algorithm



- $n \log n$ works – not work efficient
- $\log n$ steps – step efficient

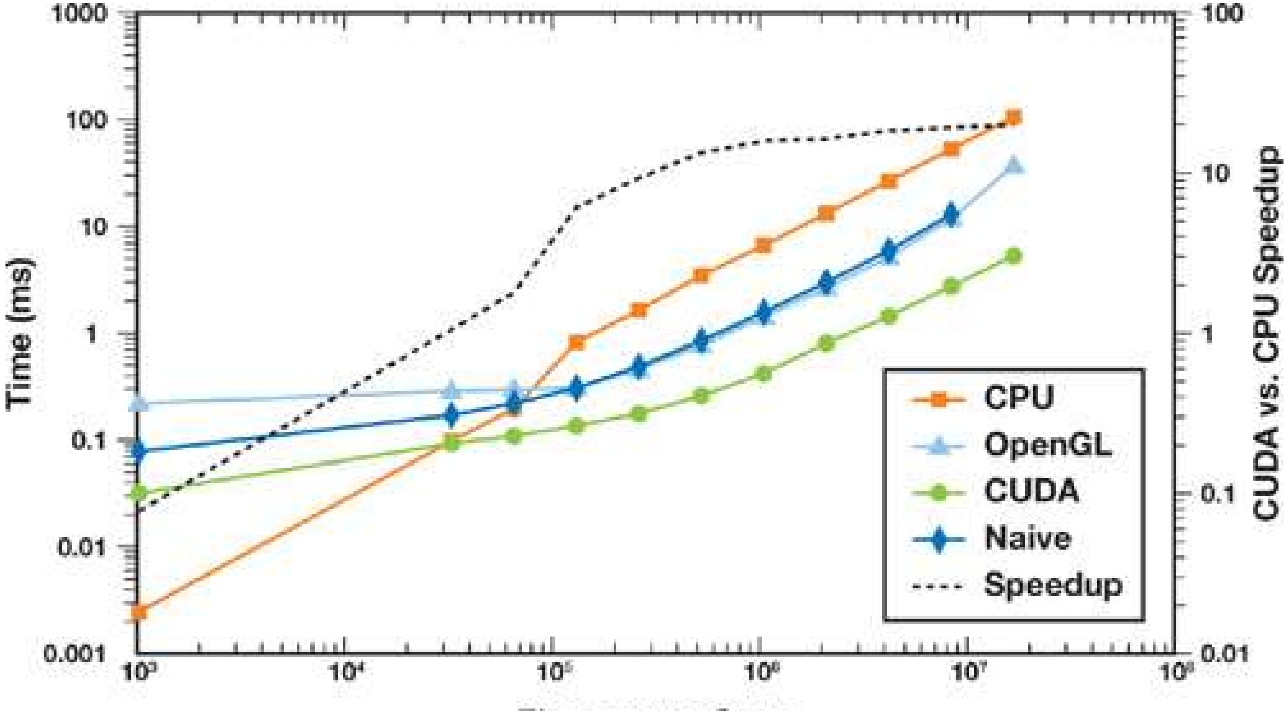
Blelloch algorithm



- n works – work efficient
- $2n \log n$ steps – not step efficient

State-of-the-art parallel scans

- Millions of scans in 10 ms



Dynamics and Scan

- Inverse dynamics can be implemented as two scans
- Forward dynamics can also be (almost) casted as a sequence of scans
- Our method uses Lie-group based generalized coordinate – accurate and fast

Inverse dynamics (two scans)

for $i = 1, \ldots, n$:

$$\begin{cases} f_{i-1,i} = M_i e^{S_i q_i} \\ V_i = \mathrm{Ad}_{f_{i-1,i}^{-1}}(V_{i-1}) + S_i \dot{q}_i \\ \dot{V}_i = S_i \ddot{q}_i + \mathrm{Ad}_{f_{i-1,i}^{-1}}(\dot{V}_{i-1}) - \mathrm{ad}_{S_i \dot{q}_i} \mathrm{Ad}_{f_{i-1,i}^{-1}}(V_{i\text{-}} \end{cases}$$

for $i = n, \ldots, 1$:

$$\begin{cases} F_i = \mathrm{Ad}^T_{f_{i,i+1}^{-1}}(F_{i+1}) + J_i \dot{V}_i - \mathrm{ad}^T_{V_i}(J_i V_i) \\ \tau_i = S_i^T F_i. \end{cases}$$

Forward dynamics (joint-space inertia algo)

$$\begin{cases} \tau = M(q)\ddot{q} + \tau^{\text{bias}} \\ \tau^{\text{bias}} := \text{ID}(q, \dot{q}, 0, V_0, \dot{V}_0, F_{n+1}) \end{cases}$$

- The inertia matrix is computed by call n different instances of inverse dynamics
- Eventually $\ddot{q} = M^{-1}(q)\hat{\tau} := M^{-1}(q)(\tau - \tau^{\text{bias}})$
- The matrix inversion is $O(n^3)$

Parallel JSI

- Use parallel scans and parallel matrix inversion when the link number is not too many (< 100)
- When link number is > 100, the GPU inversion becomes the bottleneck
- Eventually infeasible for more links or more instances

Forward dynamics (articulated body inertia algo)

- The first step is to compute the "inertia matrix" in a recursive way

for $i = n, \ldots, 1$:

$$\hat{J}_i = J_i + \mathrm{Ad}^T_{f^{-1}_{i,i+1}} \hat{J}_{i+1} \mathrm{Ad}_{f^{-1}_{i,i+1}} - \frac{\mathrm{Ad}^T_{f^{-1}_{i,i+1}} \hat{J}_{i+1} S_{i+1} S^T_{i+1} \hat{J}_{i+1} \mathrm{Ad}_{f^{-1}_{i,i+1}}}{S^T_{i+1} \hat{J}_{i+1} S_{i+1}}$$

- We find it is "impossible" to be parallelized
- But it can compute the inverse inertia at $O(n)$

Forward dynamics (articulated body inertia algo)

Backward recursion
(for $i = n, \dots, 1$)

$$\begin{cases} \hat{z}_i = Y_{i,i+1}\hat{z}_{i+1} + \Pi_{i,i+1}\hat{\tau}_{i+1} \\ c_i = \hat{\tau}_i - S_i^T \hat{z}_i \\ \hat{c}_i = \Omega_i^{-1} c_i := (S_i^T \hat{J}_i S_i)^{-1} c_i \end{cases}$$

Forward recursion
(for $i = 1, \dots, n$)

$$\begin{cases} \lambda_i = Y_{i-1,i}^T \lambda_{i-1} + S_i \hat{c}_i \\ \ddot{q}_i = \hat{c}_i - \Pi_{i-1,i}^T \lambda_{i-1} \end{cases}$$

$$Y_{i,i+1} := \mathrm{Ad}_{f_{i,i+1}^{-1}}^T \left(I - \frac{\hat{J}_{i+1} S_{i+1} S_{i+1}^T}{S_{i+1}^T \hat{J}_{i+1} S_{i+1}} \right) \qquad \Pi_{i,i+1} := \frac{\mathrm{Ad}_{f_{i,i+1}^{-1}}^T \hat{J}_{i+1} S_{i+1}}{S_{i+1}^T \hat{J}_{i+1} S_{i+1}}$$

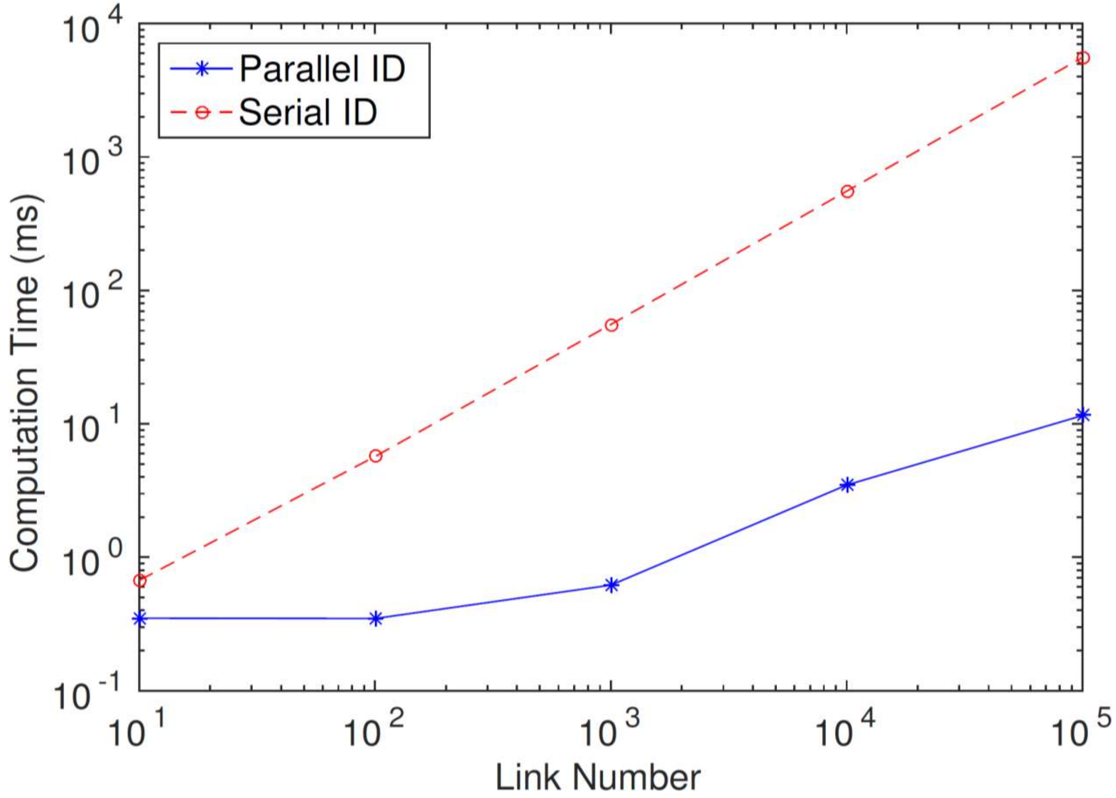
Parallel ABI

- The first recursion is implemented on CPU in the asynchronous manner
- The other parts of the forward dynamics are implemented as scans on GPU
- A hybrid CPU-GPU algorithm

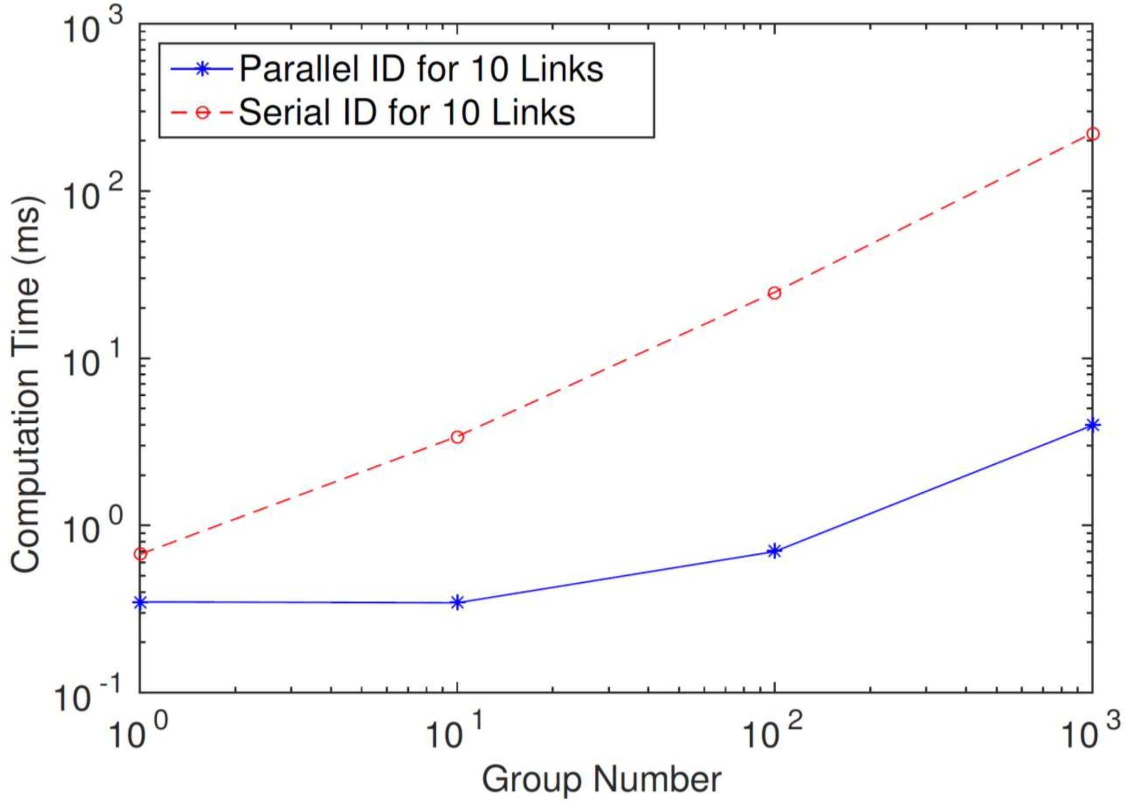
Scan-based parallel dynamics calculator

- Can compute dynamics for robots with many links in parallel
- Can compute dynamics for many robots in parallel
- Compare with multi-threading CPU implementations

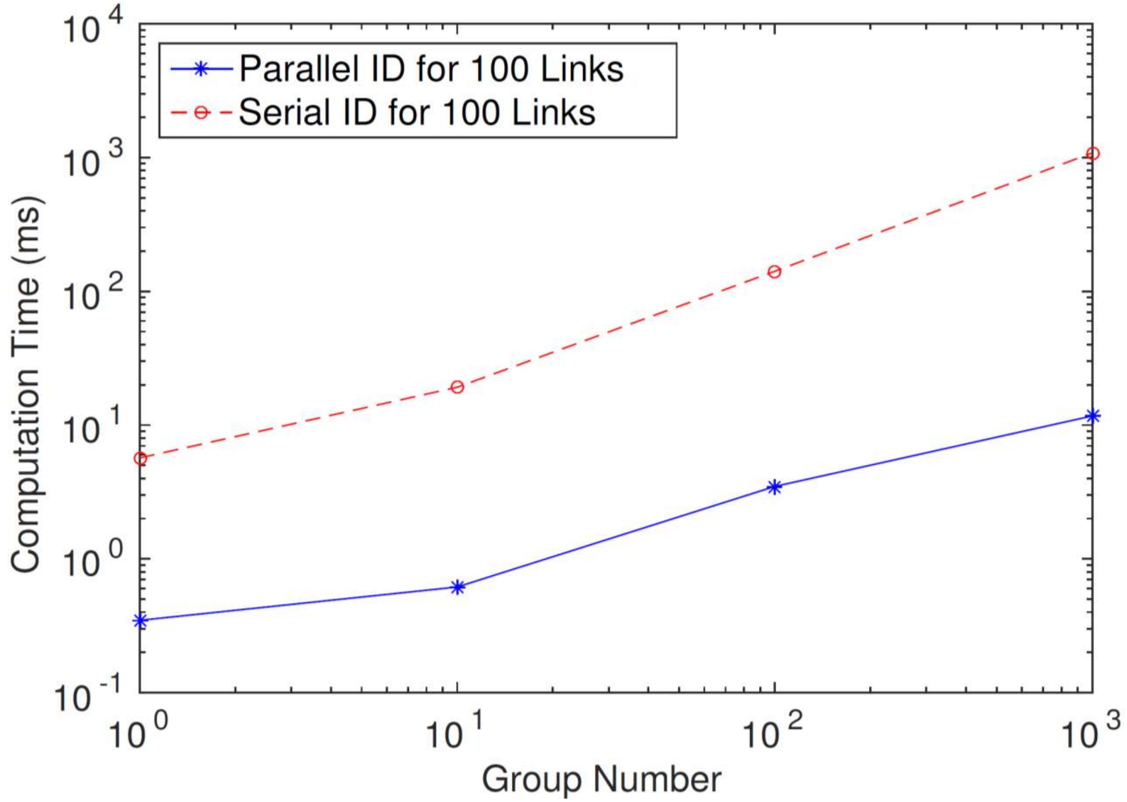
Inversed dynamics for many links



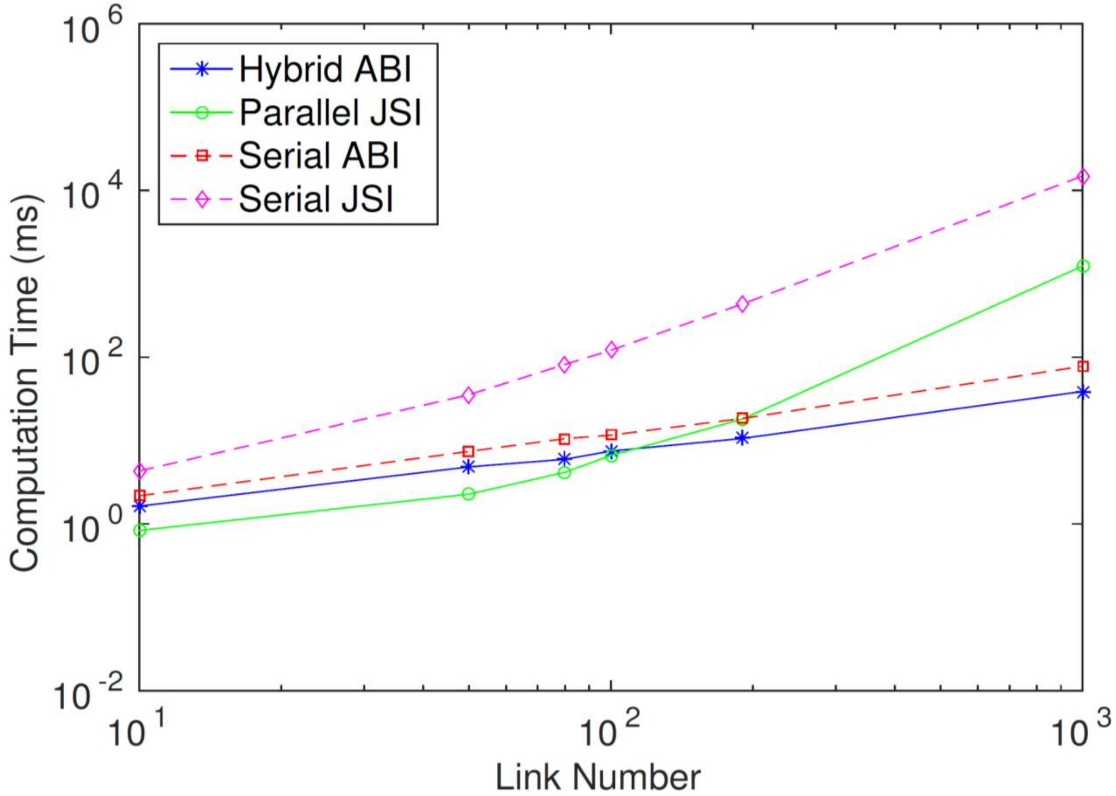
Inverse dynamics for many robots (10 links)



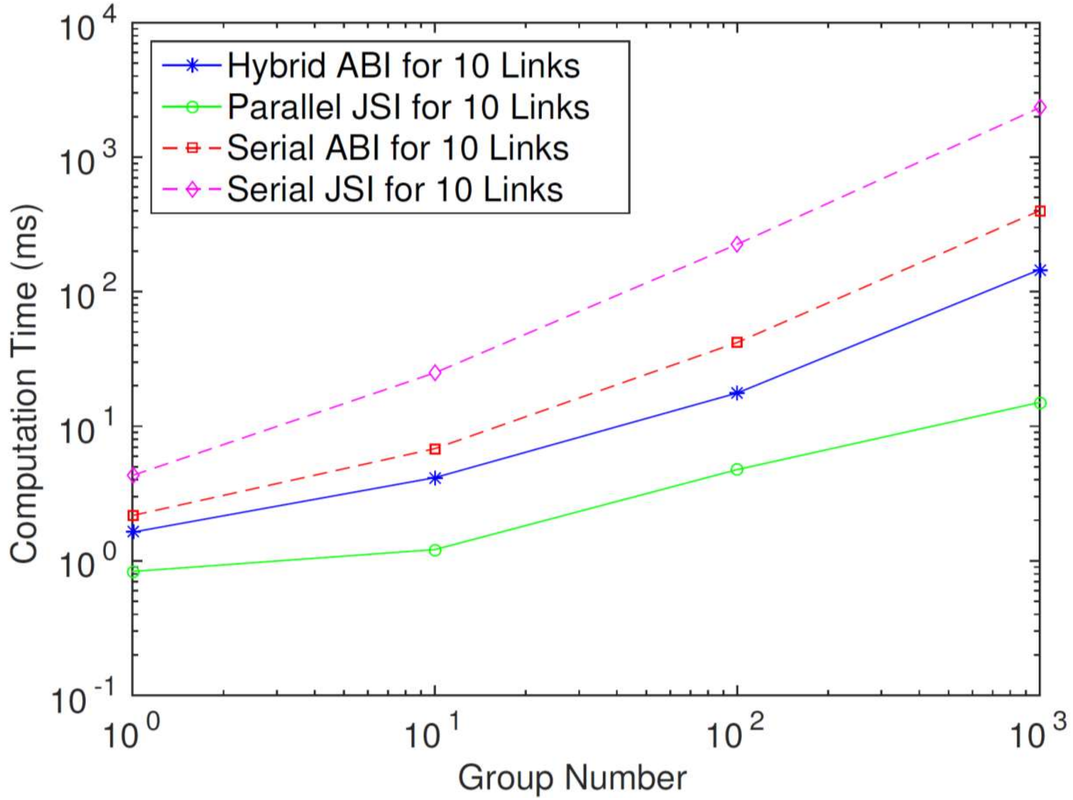
Inverse dynamics for many robots (100 links)



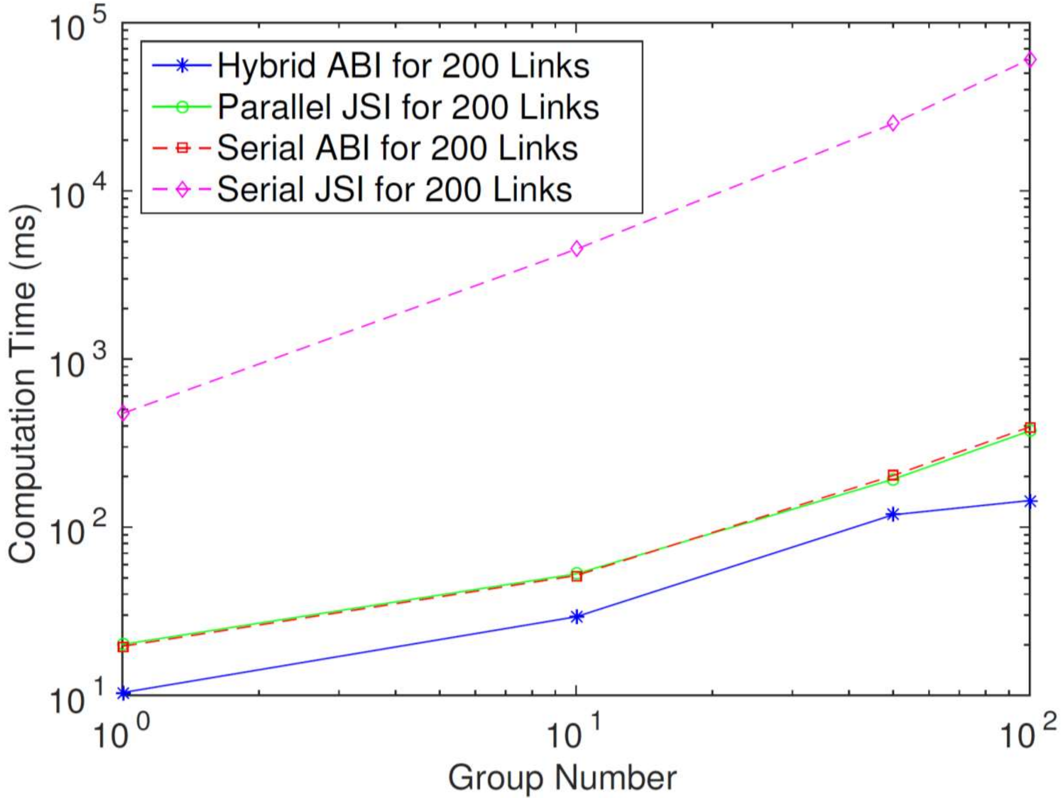
Forward dynamics for many links



Forward dynamics for many robots (10 links)



Forward dynamics for many robots (100 links)



Conclusion

- A very fast GPU-based parallel inverse dynamics and forward dynamics package
- Will release as an open-source package soon
- If you have any good applications about this method, please contact me
- If you can find a way to make ABI fully parallelized on GPU, please contact me

Future work

- Implement the entire dynamics simulator (gradient, Hessian, variational integrator, etc..)
- Take care of the contact dynamics