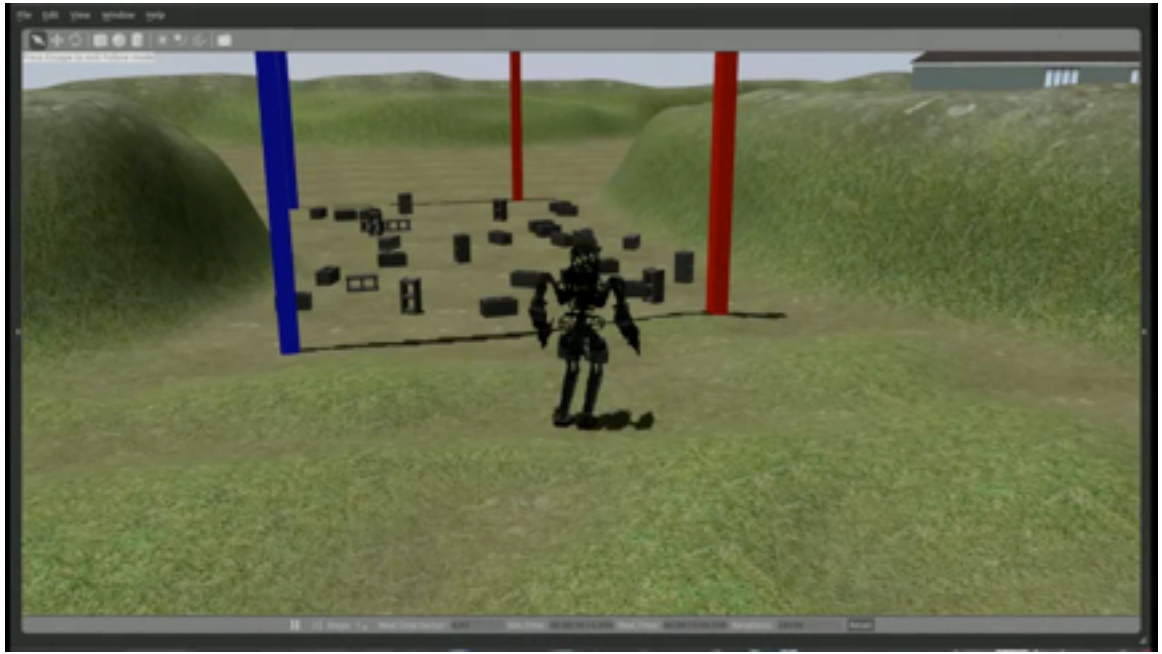
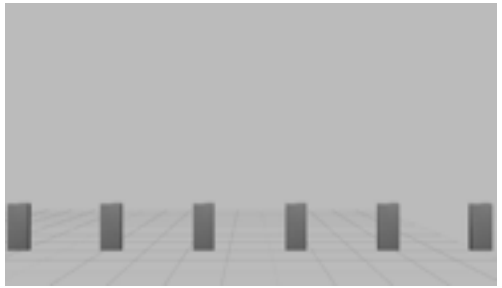
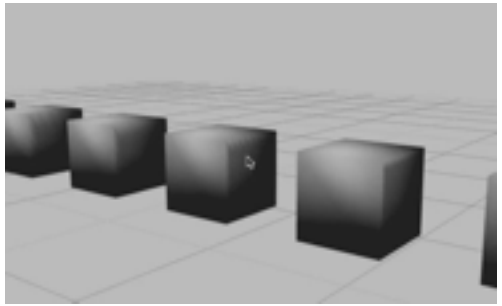
Simple benchmarks for speed and accuracy of rigid body dynamic simulators

Steven Peters (scpeters), John Hsu (hsu)







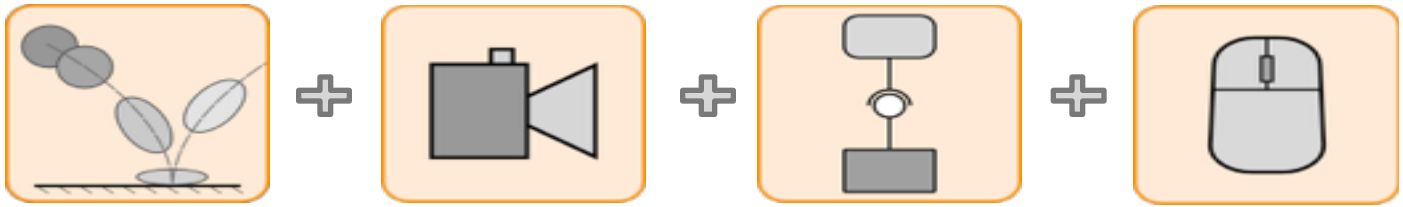
Outline

- Overview of the Open Source Gazebo Simulator
- Robot walking simulation speed
- Components of a benchmark
- Analysis of simple benchmark
- Software details
- Future work

Gazebo Simulator: Overview and Purpose

Goal: Best possible substitute for physical robot

Architecture:



Physics + Sensors + Interfaces + GUI

Use cases:

- Design and testing of robot components and control
- Software testing and verification
- Competitions

gazebosim.org

Open Source Physics Engines in Gazebo

Open Dynamics Engine (ODE)	bitbucket.org/odedevs/ode Robotics, gaming
Bullet	github.com/bulletphysics/bullet3 Gaming, animation, Sony, AMD, Google
Simbody	github.com/simbody/simbody Biomechanics, Stanford
DART	github.com/dartsim/dart Robotics, animation, Georgia Tech

Open Source Physics Engines in Gazebo

Easy to switch between physics engines (gazebo 3.0+)

Command line option:

```
gazebo -e {bullet|dart|ode|simbody}
```

Attribute in sdf world file:

```
<world><physics type="simbody" />...
```

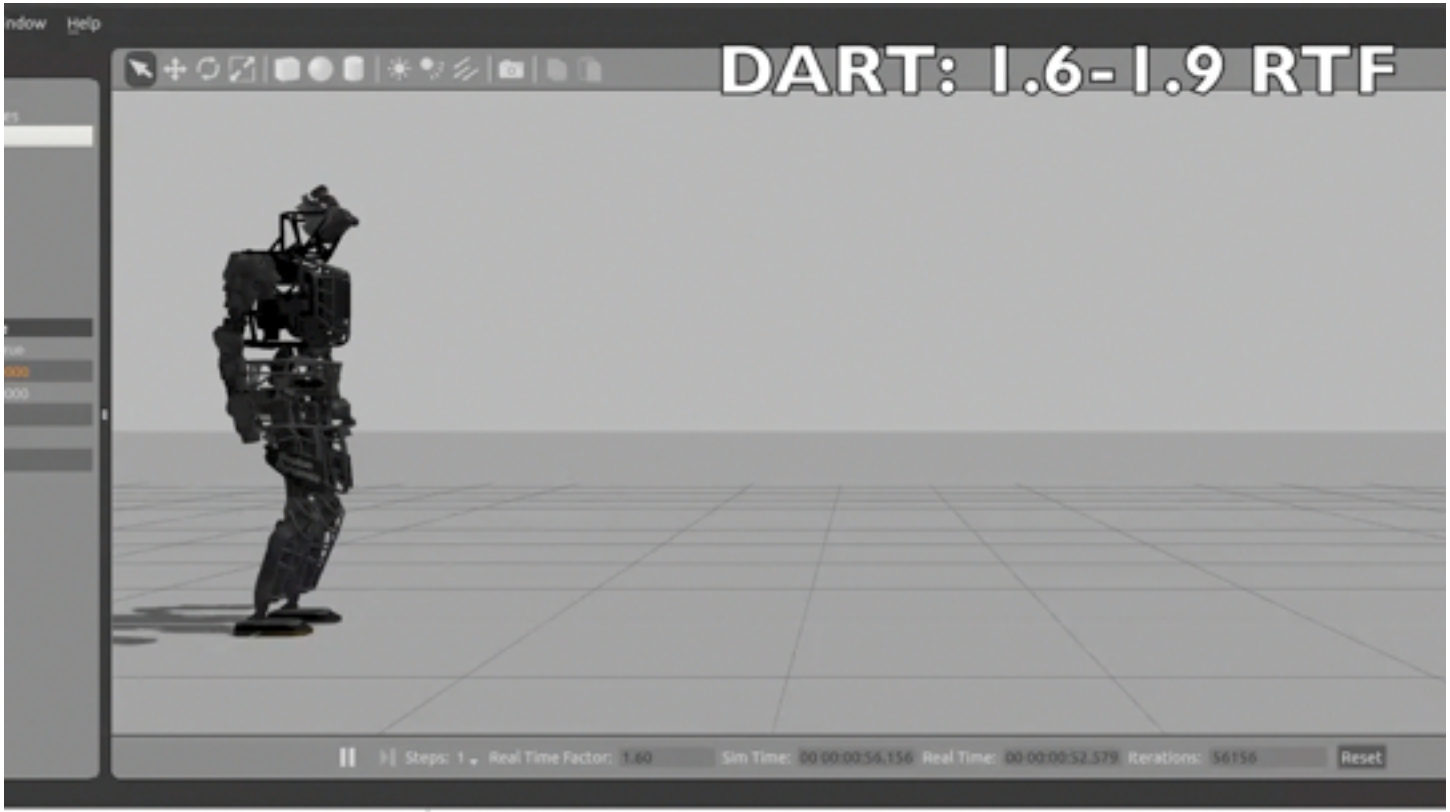
gazebo5+ include support for Bullet, ODE, and Simbody

(Dart requires building from source)

from packages.ros.org:

```
sudo apt-get install ros-jade-gazebo-ros-pkgs
```

Robotic walking task: speed comparison

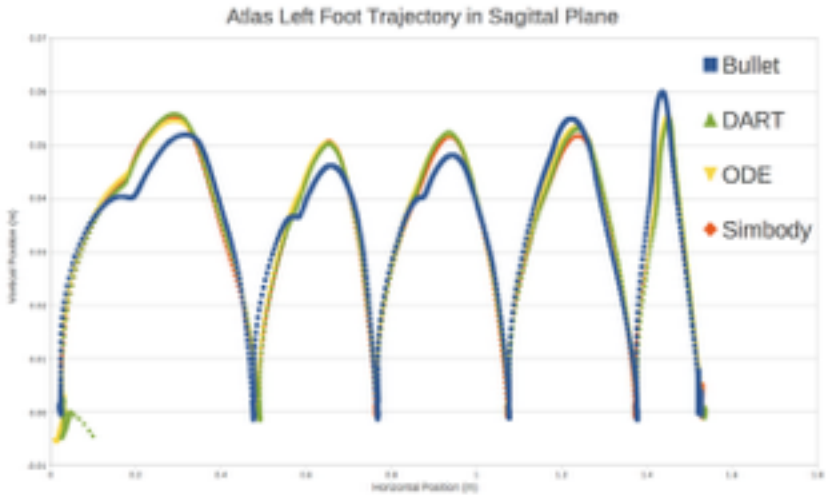


https://vimeo.com/105584932

Robotic walking task: analysis

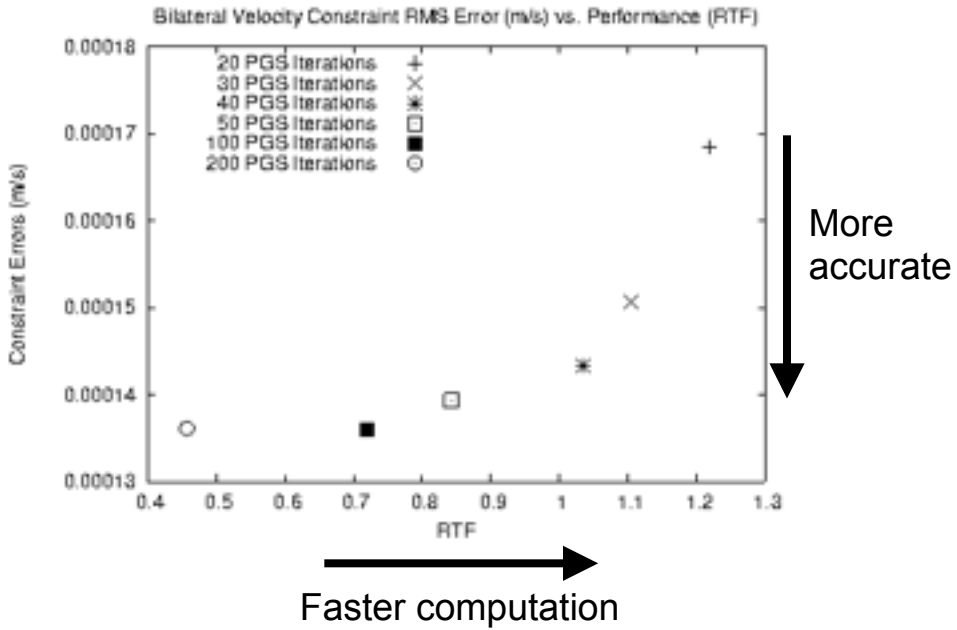
Trajectories look similar

Hard to say more without validation



Open Dynamics Engine parameter study

Iterations vs error vs speed



Components of a benchmark

Scenario

- Model: dx/dt = f(x, t)
- Initial conditions: x(t0)
- Expected behavior: y = h(x,t) for t in [t0,tf]

Parameters

- May be different for each physics engine
- Time step size, number of objects, solver iterations

Performance metrics

- Accuracy
- Computational speed

Reasons to use simple benchmarks

Make it easier to define accuracy metrics

analytical solutions, conservation laws

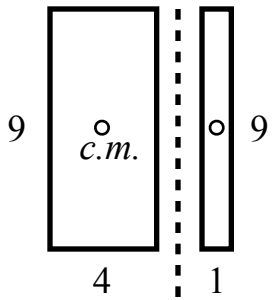
Isolate effects of solver parameters

simplifies parameters sensitivity analysis

Boxes: free-floating rigid bodies

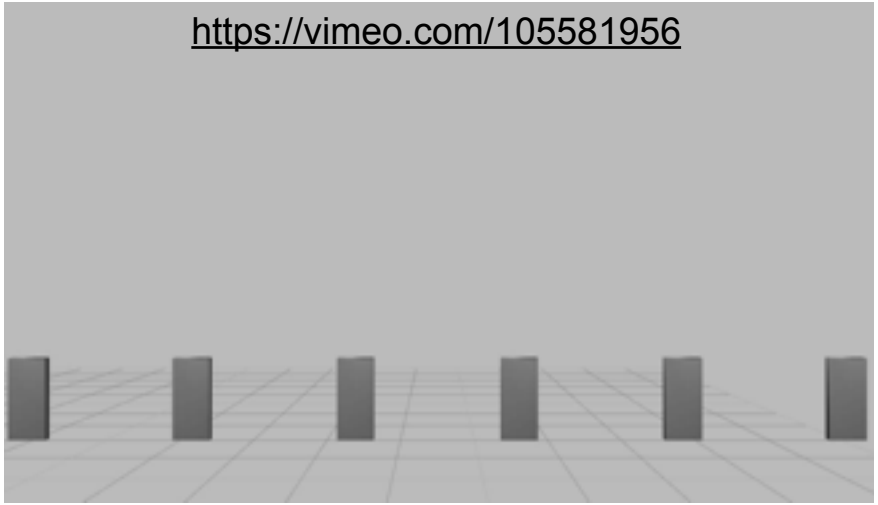
Model: boxes 1x4x9

- Free-floating
- Constant gravity field



Initial conditions:

- Largest angular velocity about axis of size 4 (leads to tumbling)



Expected behavior:

- Parabolic trajectory of center of mass ($c.m.$)
- Angular momentum conserved in world frame

Parameters:

- Solver time step size
- Number of boxes

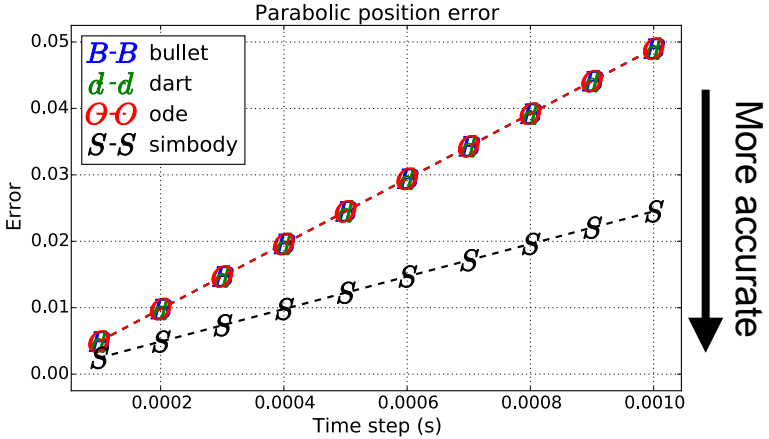
Analysis: parabolic position error

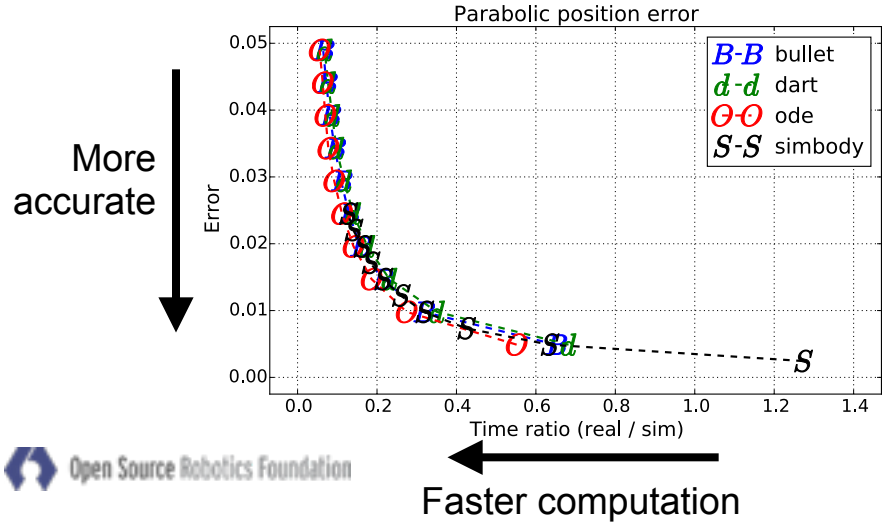
All using semi-explicit Euler (1st order)

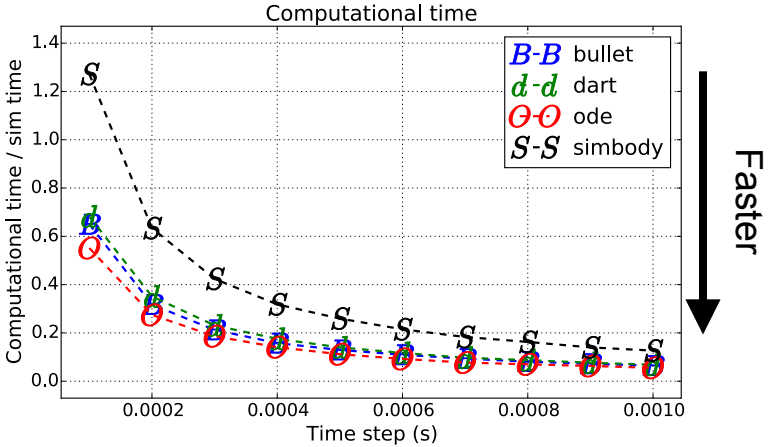
Simbody takes extra half-step to estimate error

Error proportional to time step size:

Accuracy vs. computational time is comparable





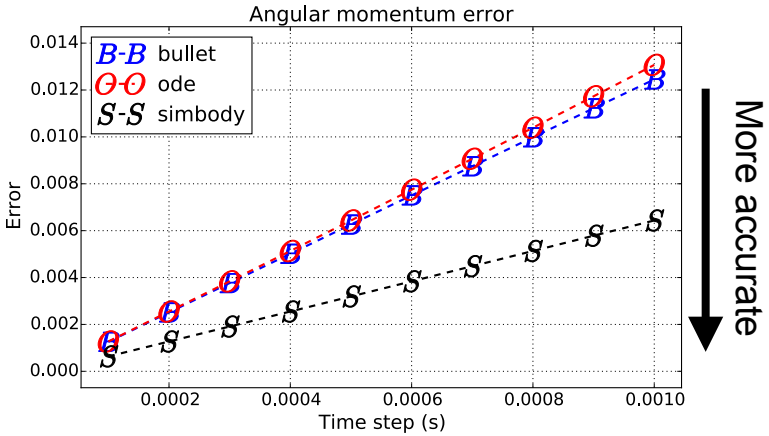


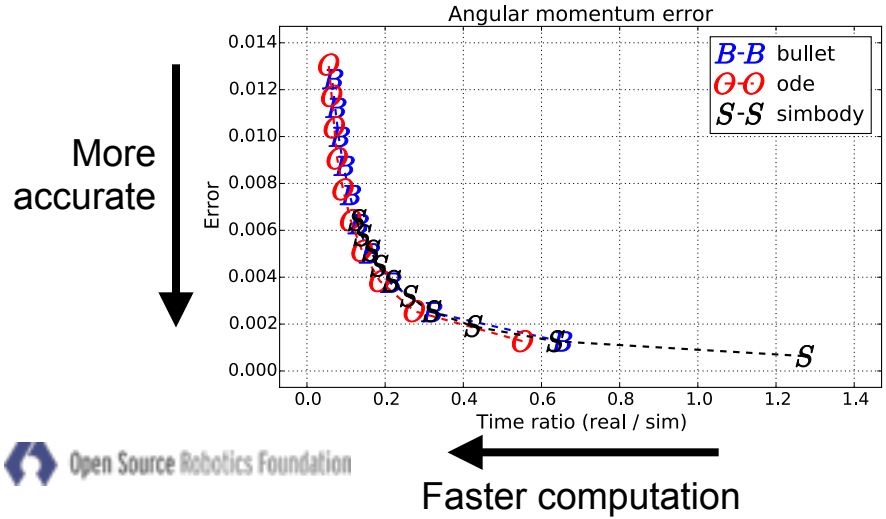
Open Source Robotics Foundation

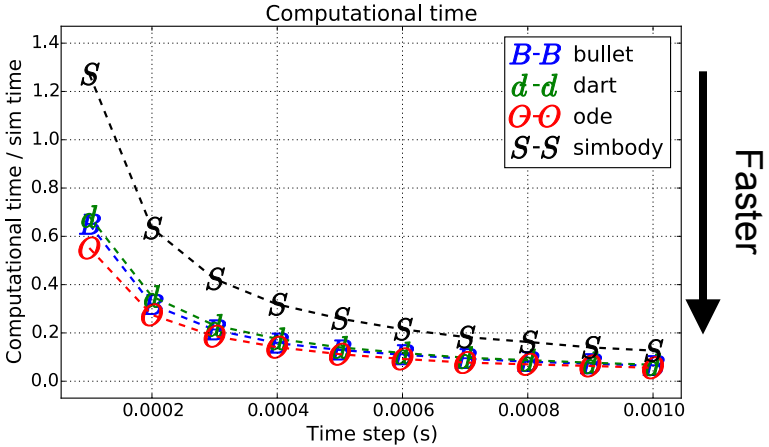
Analysis: angular momentum error

Results are similar to parabolic position error

DART not shown since it currently has a bug I found while running this benchmark https://github.com/dartsim/dart/issues/424







Open Source Robotics Foundation

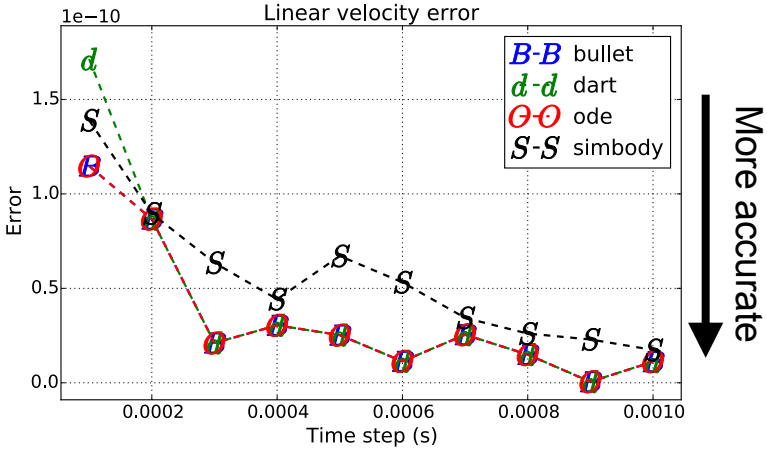
Analysis: linear velocity error

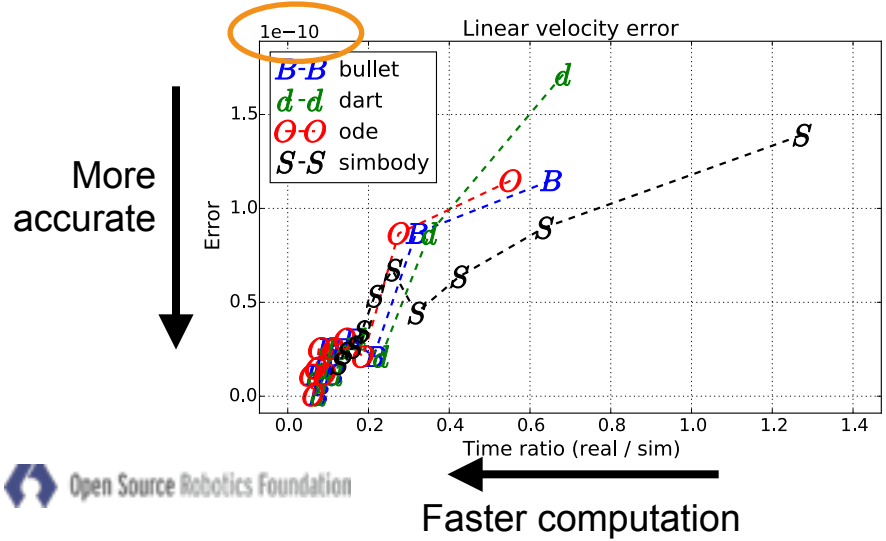
Integration scheme should not contribute error

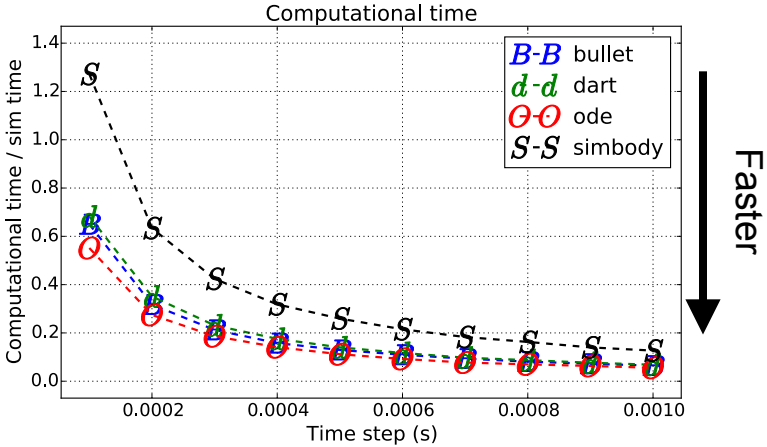
Error is due to floating point rounding errors

Smaller time-step means more floating point calculations

Usually not dominant source of error







Open Source Robotics Foundation

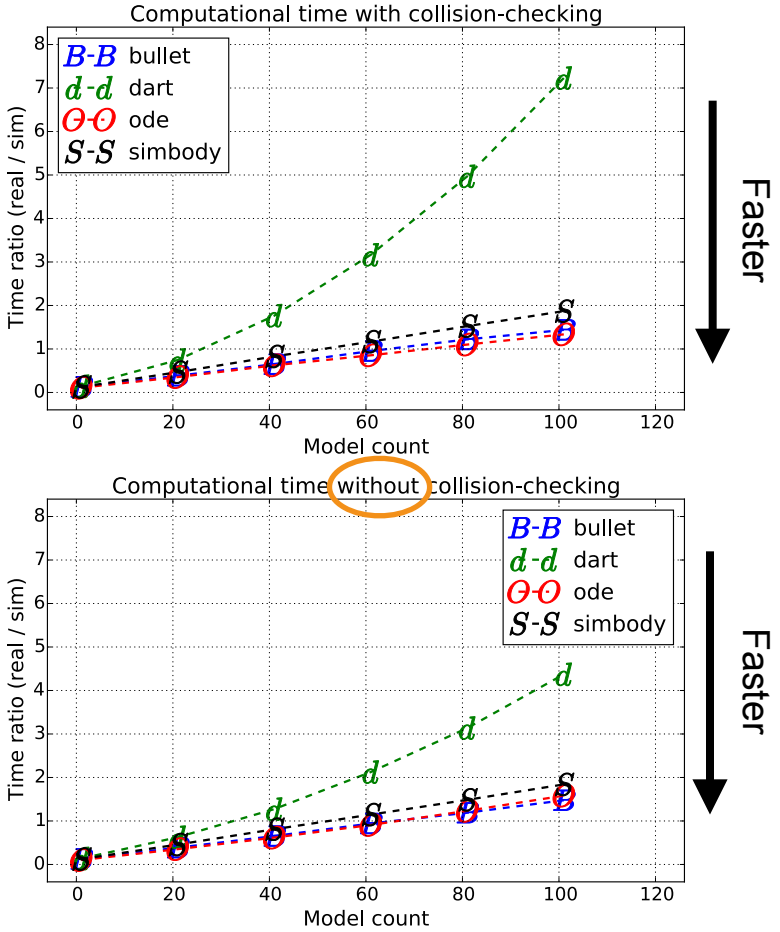
Analysis: computational time

Scaling of computational time with multiple boxes (1 - 101)

Timestep chosen to give equal accuracy (larger timestep for simbody)

DART appears to have more super-linear sensitivity to number of boxes

Disabling collision checker helps since it doesn't use a broadphase, but only partially



Software components

"Boxes" benchmark written in **C++**

- Model creation (geometry, inertias, initial conditions)
- Time-stepping
- Computation of performance metrics (full logs not saved by default)

Cmake `find_package(gazebo 6)` to link against gazebo version 6

GoogleTest runs each parameter option as a separate test case

- Full factorial combination of up to 9 parameters at once (that's too many)
- Exports data as junit XML and time-stamped **CSV**

Documentation (with **LaTeX**) and plots in **iPython notebook**

Hosted on **Github**: helps with collaboration

http://github.com/scpeters/benchmark

Food for thought

Run timing benchmarks on same hardware

Making it easy to reproduce results (compile, configure, etc)

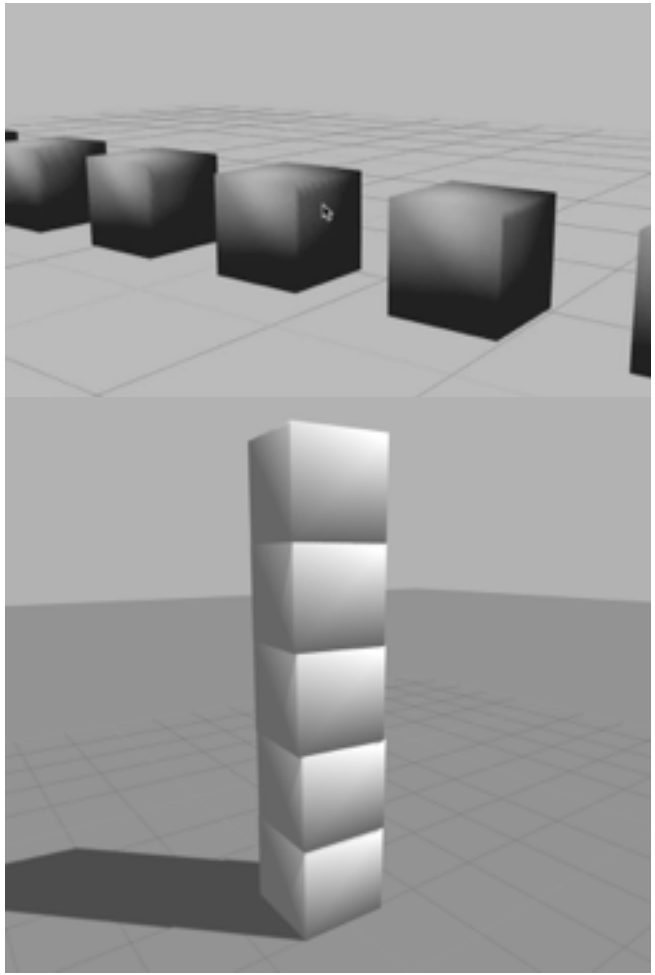
Online collaboration like github?

Future work

More scenarios with articulation and contact

Software improvements

- Ease adding parameters to Gazebo API
- Logging and debugging of single test cases
- Generic physics API for Gazebo



OSRF Sponsors