PR[AI]RIE

PaRis Artificial Intelligence Research InstitutE

Pinocchio

Efficient and versatile rigid body dynamics algorithms

Justin Carpentier

Researcher, INRIA and ENS, Paris

What is Pinocchio?

Pinocchio

Efficient and versatile rigid body dynamics algorithms

Pinocchio is an open-source and efficient framework implementing most common rigid body dynamics algorithms written in C++ and coming with Python bindings

github.com/stack-of-tasks/pinocchio

Pinocchio

Efficient and versatile rigid body dynamics algorithms

Pinocchio is an open-source and highly efficient framework for simulation, planning and control used in robotics, biomechanics, civil engineering, etc.

Resulting from a joint and fruitful collaboration between Willow and Gepetto (LAAS-CNRS), with an active roadmap:

- 2015 — classic dynamics algorithms
- 2018 — efficient analytical derivatives + contact dynamics
- 2021 — differentiable simulator + real to simulation learning
- tomorrow

Pinocchio — Rigid body dynamics for articulated systems

In brief:

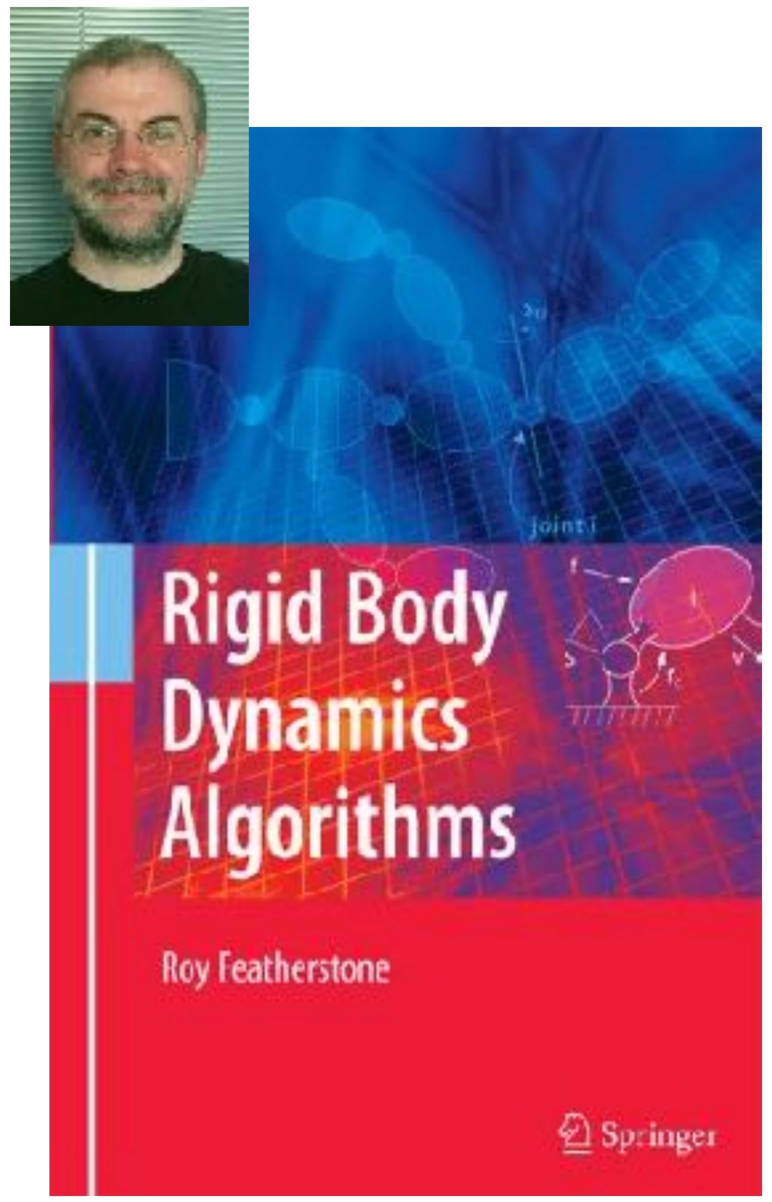
- C++ / Python
- BSD-2 license
- 5k+ commits
- 100k+ lines of code
- 4k downloads per day
- online documentation
- code generation CPU/GPU
- automatic differentiation
- deployed on major OS
- examples + tutorials

Worldwide community:

- 100+ academic labs
- 20+ universities for teaching robotics
- many robotic companies, among them:

A real influencer

The Rigid Body Dynamics Algorithms



Roy Featherstone

Goal: exploit at best the **sparsity** induced by the kinematic tree

The Articulated Body Algorithm

$$\ddot{q} = \textbf{ForwardDynamics}\left(q, \dot{q}, \tau, \lambda_c\right)$$

Simulation

Control

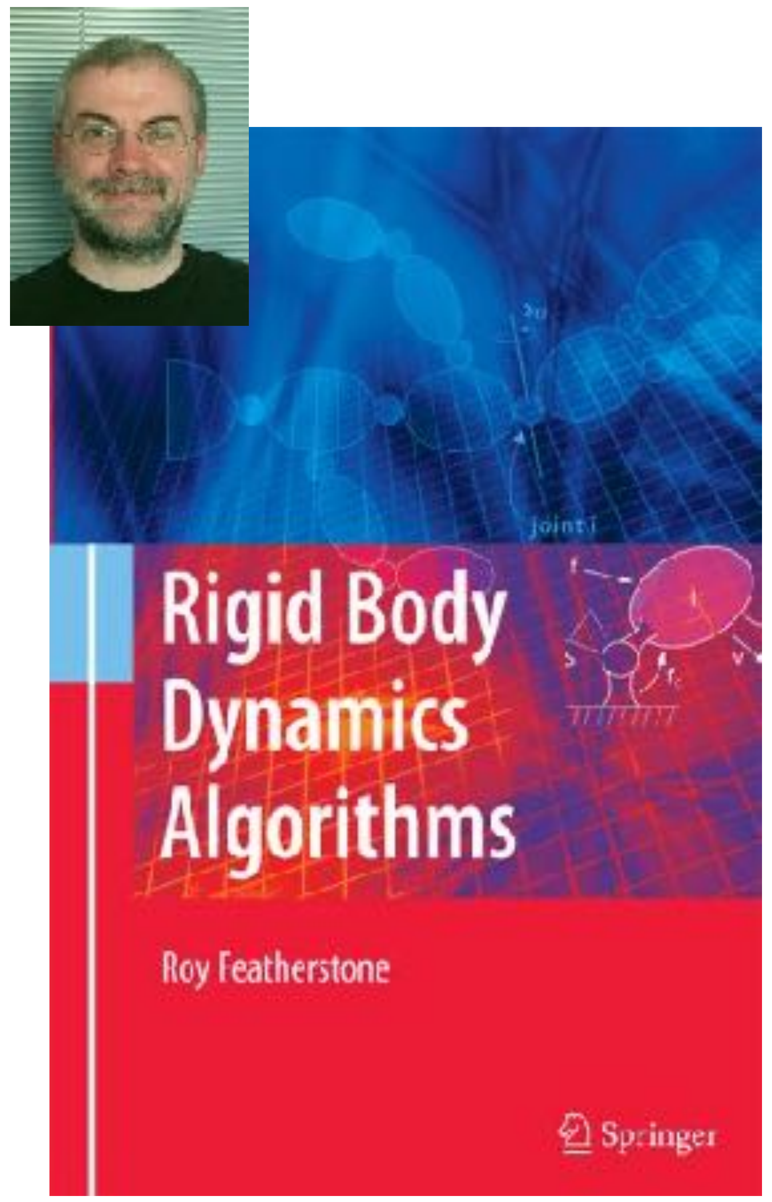
$$\tau = \textbf{InverseDynamics}\left(q, \dot{q}, \ddot{q}, \lambda_c\right)$$

The Recursive Newton-Euler Algorithm

$$M(q)\ddot{q} \; + \; C(q, \dot{q}) \; + \; G(q) \; = \; \tau \; + \; J_c^\top(q)\lambda_c$$

Mass Matrix	Coriolis centrifugal	Gravity	Motor torque	External forces

The Rigid Body Dynamics Algorithms



Roy Featherstone

Goal: exploit at best the **sparsity** induced by the kinematic tree

The Articulated Body Algorithm

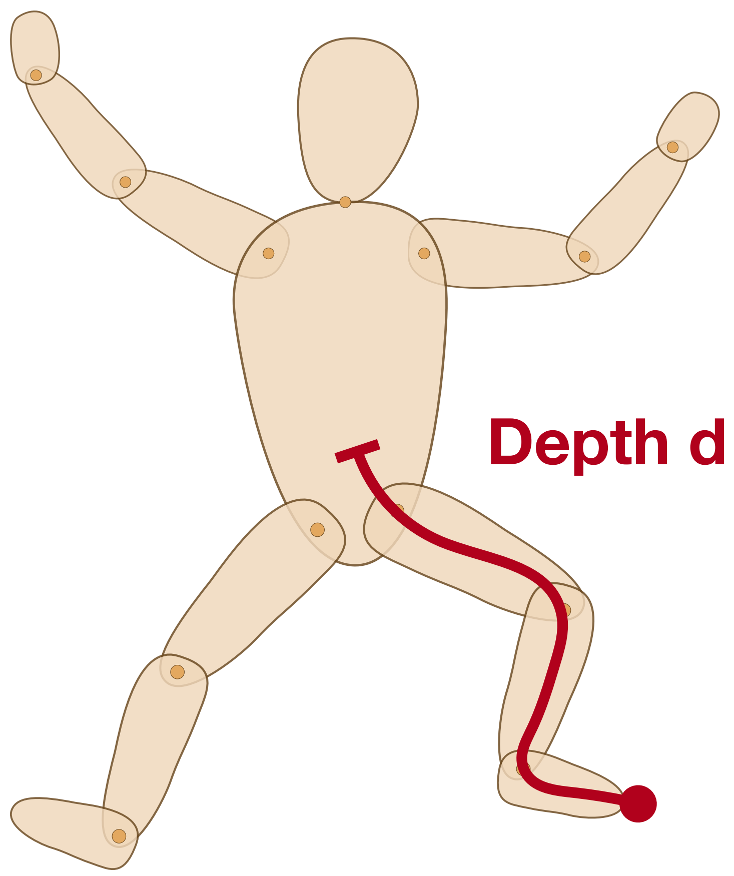
$$\ddot{q} = \textbf{ForwardDynamics}\left(q, \dot{q}, \tau, \lambda_c\right)$$

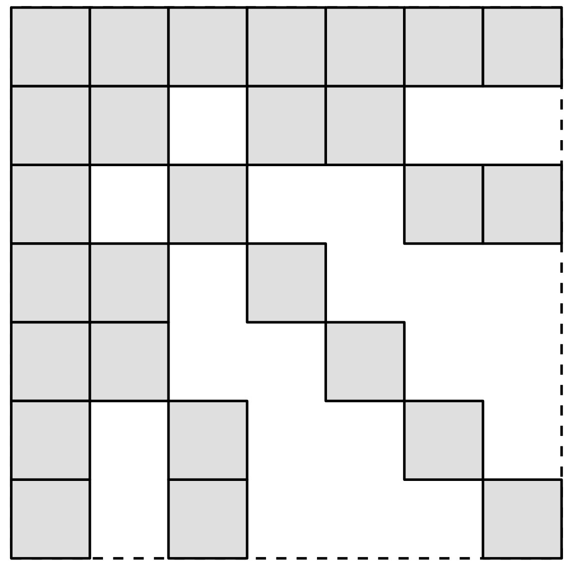
Simulation

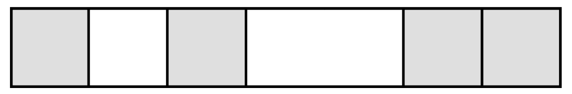
Control

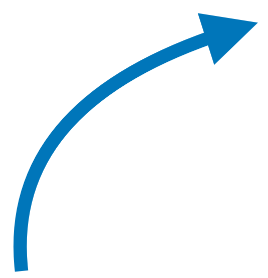
$$\tau = \textbf{InverseDynamics}\left(q, \dot{q}, \ddot{q}, \lambda_c\right)$$

The Recursive Newton-Euler Algorithm









$$M(q)\ddot{q} \;+\; C(q,\dot{q}) \;+\; G(q) \;=\; \tau \;+\; J_c^\top(q)\lambda_c$$

Mass Matrix — Coriolis centrifugal — Gravity — Motor torque — External forces

The main features of Pinocchio

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- <u>deals with Lie group geometry</u> [accurate]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- <u>deals with Lie group geometry</u> [accurate]
- <u>analytical derivatives</u> [online predictive control, reinforcement learning]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- <u>deals with Lie group geometry</u> [accurate]
- <u>analytical derivatives</u> [online predictive control, reinforcement learning]
- <u>automatic differentiation</u> [flexible]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- <u>deals with Lie group geometry</u> [accurate]
- <u>analytical derivatives</u> [online predictive control, reinforcement learning]
- <u>automatic differentiation</u> [flexible]
- <u>source code generation</u> [dedicated to each architecture]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- deals with Lie group geometry [accurate]
- analytical derivatives [online predictive control, reinforcement learning]
- automatic differentiation [flexible]
- source code generation [dedicated to each architecture]
- Python bindings [fast prototyping]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- deals with Lie group geometry [accurate]
- analytical derivatives [online predictive control, reinforcement learning]
- automatic differentiation [flexible]
- source code generation [dedicated to each architecture]
- Python bindings [fast prototyping]
- multi-thread friendly [fast]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- deals with Lie group geometry [accurate]
- analytical derivatives [online predictive control, reinforcement learning]
- automatic differentiation [flexible]
- source code generation [dedicated to each architecture]
- Python bindings [fast prototyping]
- multi-thread friendly [fast]
- collision detection with HPP-FCL [simulation]

The main features of Pinocchio

- supports a large number of joints (revolute, prismatic, free-flyer, etc.) [flexible]
- handles the complete sparsity via the Featherstone algorithms [fast]
- implements classic + advanced rigid body dynamics algorithms [versatile]
- deals with Lie group geometry [accurate]
- analytical derivatives [online predictive control, reinforcement learning]
- automatic differentiation [flexible]
- source code generation [dedicated to each architecture]
- Python bindings [fast prototyping]
- multi-thread friendly [fast]
- collision detection with HPP-FCL [simulation]
- reads robot model from URDF, SDF, etc. [compatibility]

The central paradigm

The key aspect is the explicit splitting between model and data:

```
algorithm<Scalar>(model, data, arg1, arg2, …)
```

full templatization — constant quantity — cached variables

The central paradigm

The key aspect is the explicit splitting between model and data:

```
algorithm<Scalar>(model, data, arg1, arg2, …)
```

- full templatization (Scalar)
- constant quantity (model)
- cached variables (data)

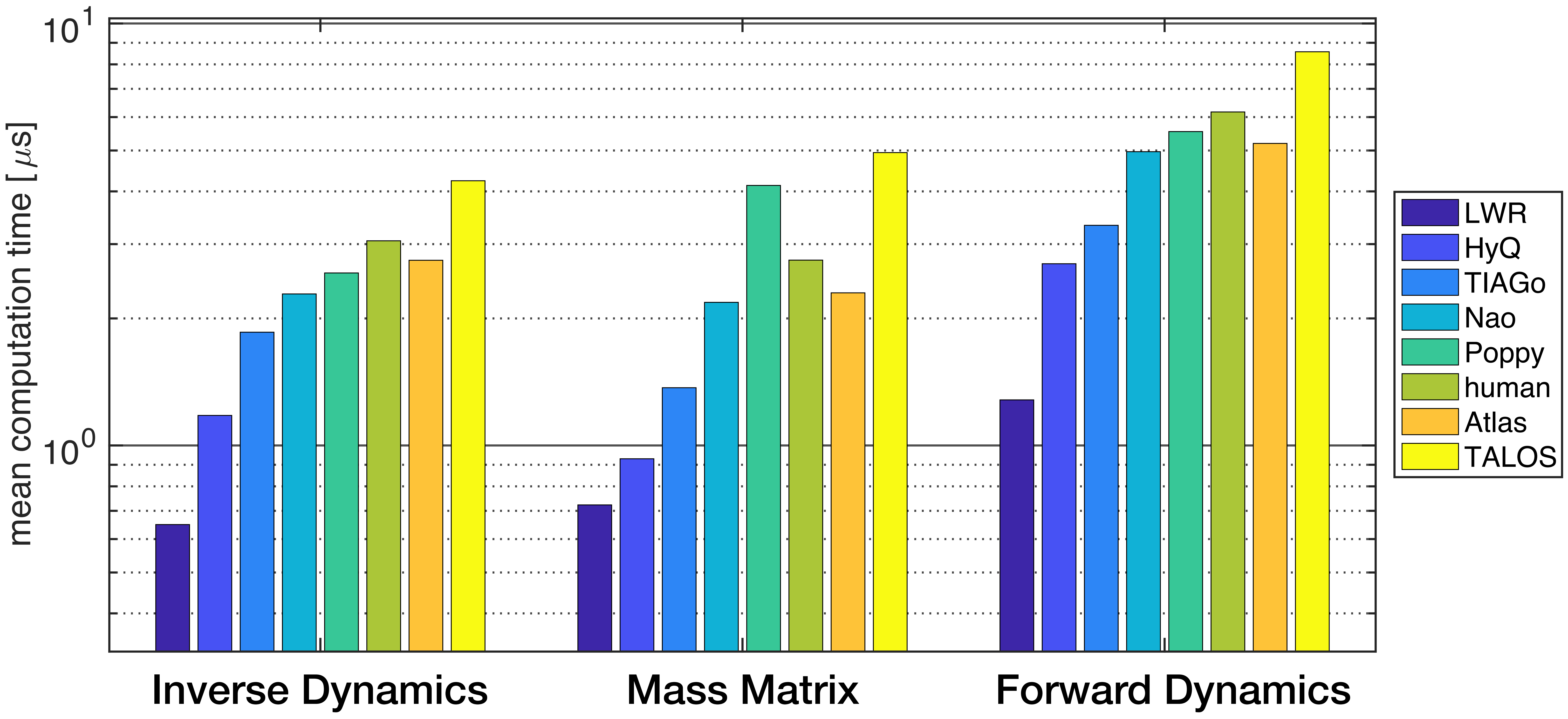
Main advantages

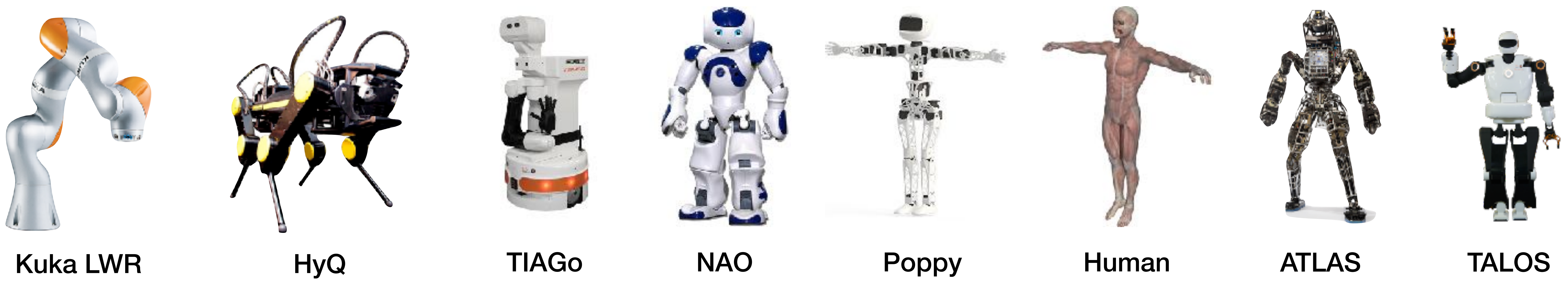
- the compiler guesses what is constant, what varies
- no online memory allocation
- good prediction/anticipation of the CPU
- algorithms are easier to write
- …


Pinocchio in action


Pinocchio in action

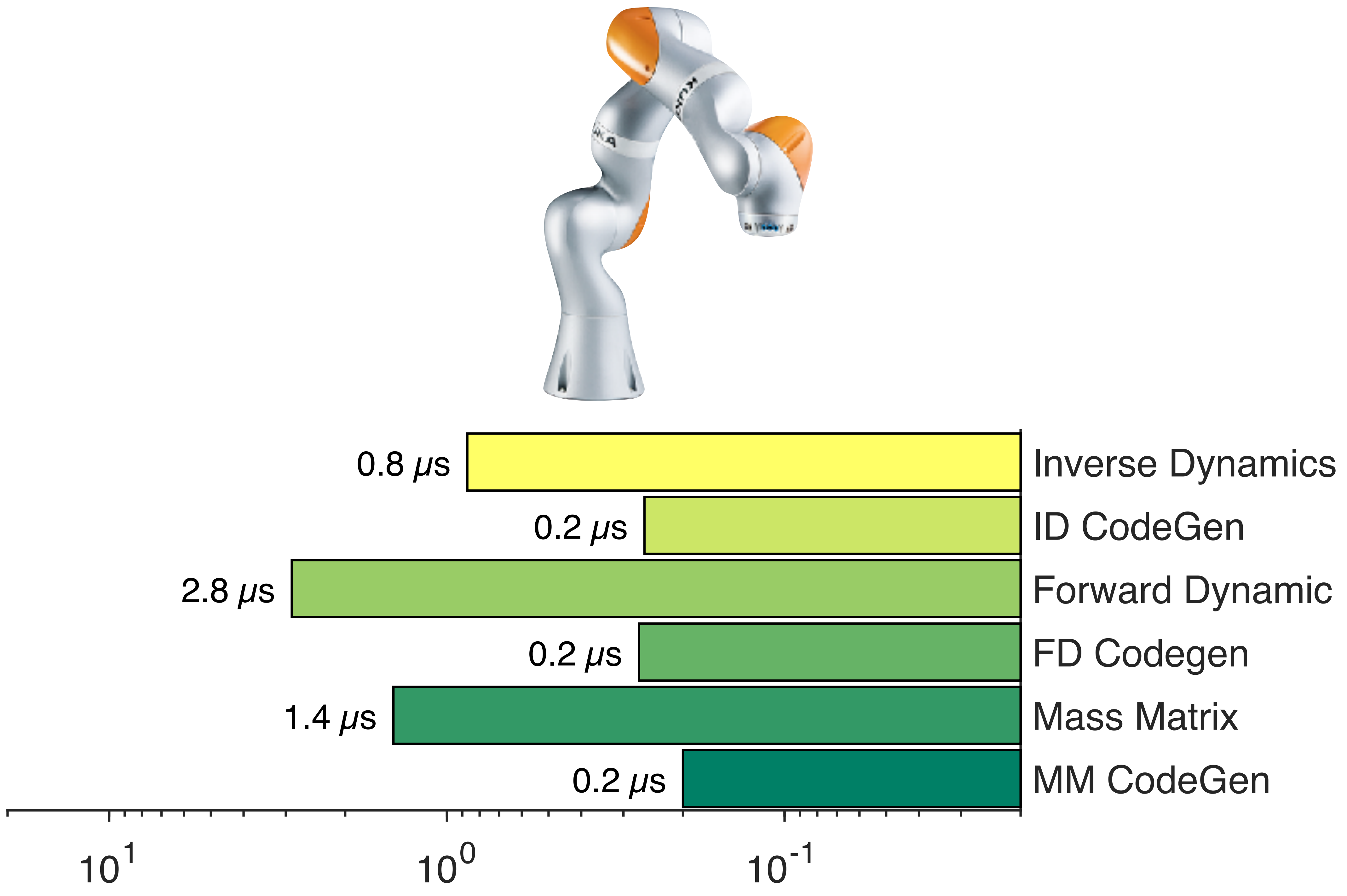
Benchmarks of basic algorithms

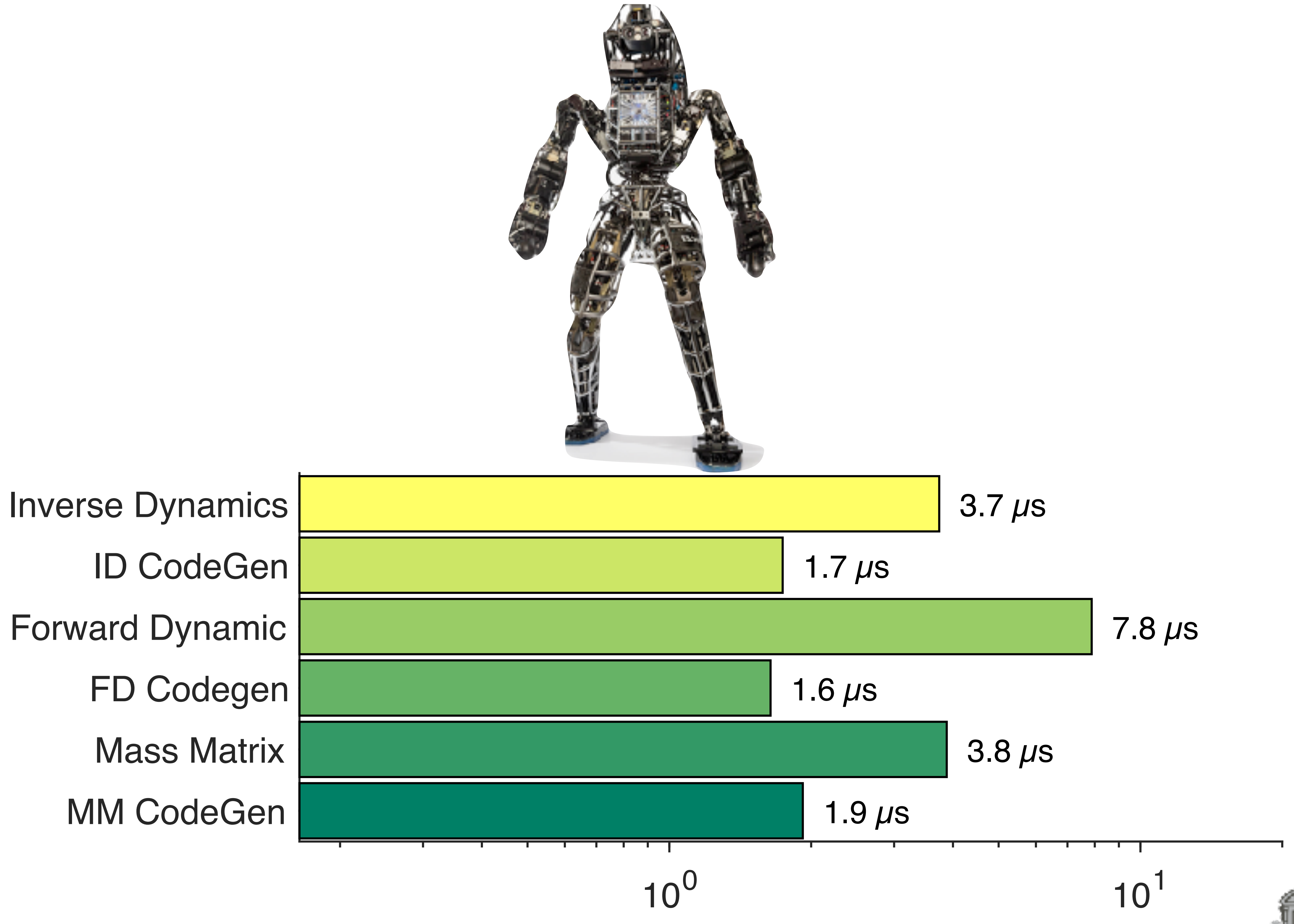




The source code generation

Pinocchio also supports source code generation:
you can compile on the fly (JIT paradigm) your code
for the best performances on your hardware





The upcoming features of Pinocchio

- GPU/FPGA **code source generation** (mostly for Model Predictive Control)
- handling **constrained** systems
- a dedicated **open-source** robotics simulator
- 100% **differentiable** simulator
- extending the support of biomechical systems (muscles)
- code generation of robot controllers
- features on demand
- extension of the collision/detection part
- …

Pinocchio on GitHub

README.md

Pinocchio

Efficient and versatile rigid body dynamics algorithms

License BSD 2-Clause | docs online | coverage 93.00% | downloads 568k | conda-forge v2.9.1 | pypi package 2.6.19

Pinocchio instantiates the state-of-the-art Rigid Body Algorithms for poly-articulated systems based on revisited Roy Featherstone's algorithms. Besides, **Pinocchio** provides the analytical derivatives of the main Rigid-Body Algorithms like the Recursive Newton-Euler Algorithm or the Articulated-Body Algorithm.

Pinocchio is first tailored for robotics applications, but it can be used in extra contexts (biomechanics, computer graphics, vision, etc.). It is built upon Eigen for linear algebra and FCL for collision detection. **Pinocchio** comes with a Python interface for fast code prototyping, directly accessible through Conda.

Pinocchio is now at the heart of various robotics software as Crocoddyl, an open-source and efficient Differential Dynamic Programming solver for robotics, the Stack-of-Tasks, an open-source and versatile hierarchical controller framework or the Humanoid Path Planner, an open-source software for Motion and Manipulation Planning.

If you want to learn more on **Pinocchio** internal behaviors and main features, we invite you to read the related paper and the online documentation.

If you want to directly dive into **Pinocchio**, only one single line is sufficient (assuming you have Conda):

conda install pinocchio -c conda-forge

or via pip (currently only available on Linux):

pip install pin

Table of contents

+ 39 contributors

Environments 1

github-pages Active

Languages

- C++ 92.2%
- Python 6.4%
- CMake 1.4%

Installing Pinocchio

github.com/stack-of-tasks/pinocchio

```
conda install pinocchio -c conda-forge
```

Installing Pinocchio

github.com/stack-of-tasks/pinocchio

```
conda install pinocchio -c olivier.roussel
```

Citing Pinocchio

```
@inproceedings{carpentier2019pinocchio,
   title={The Pinocchio C++ library -- A fast and flexible implementation of rigid body dynamics alg
   author={Carpentier, Justin and Saurel, Guilhem and Buondonno, Gabriele and Mirabel, Joseph and La
   booktitle={IEEE International Symposium on System Integrations (SII)},
   year={2019}
}
```

and the following one for the link to the GitHub codebase:

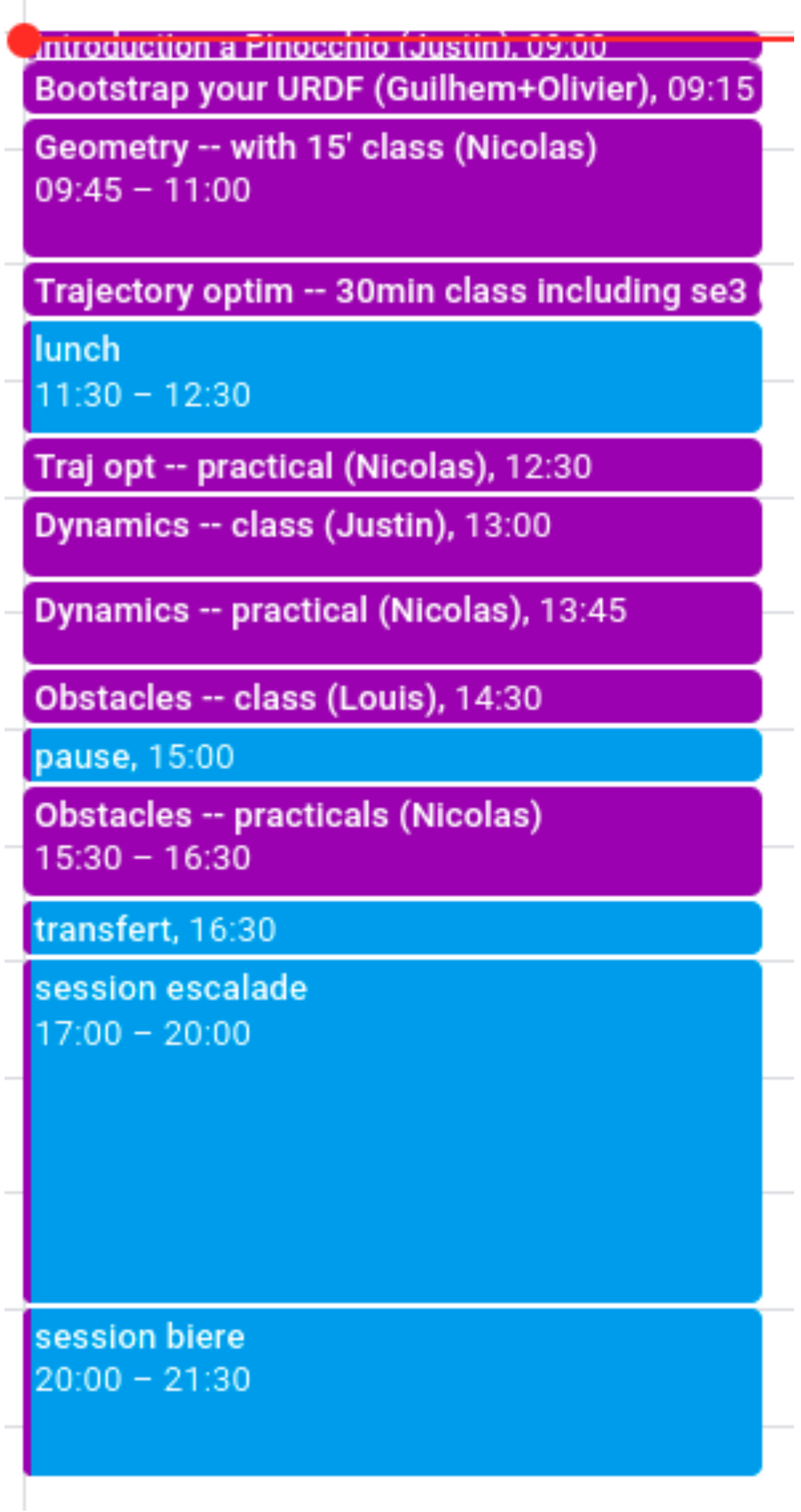
```
@misc{pinocchioweb,
   author = {Justin Carpentier and Florian Valenza and Nicolas Mansard and others},
   title = {Pinocchio: fast forward and inverse dynamics for poly-articulated systems},
   howpublished = {https://stack-of-tasks.github.io/pinocchio},
   year = {2015--2021}
}
```

The algorithms for the analytical derivatives of rigid-body dynamics algorithms are detailed here:

```
@inproceedings{carpentier2018analytical,
  title = {Analytical Derivatives of Rigid Body Dynamics Algorithms},
  author = {Carpentier, Justin and Mansard, Nicolas},
  booktitle = {Robotics: Science and Systems},
  year = {2018}
}
```

Contributing to Pinocchio

Planning of the day



Live discussions

matrix.to/#/#jnrh2023-tuto:laas.fr