Pinocchio C++ 库

刚体动力学算法及其解析导数的快速灵活实现

Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Flore
Olivier Stasse 和 Nicolas Mansard

2026 年 1 月 5 日

摘要

我们介绍了 Pinocchio，一个实现刚体动力学算法及其解析导数的开源软件框架。Pinocchio 不仅包含机器人学中使用的标准算法（例如，正向和逆向动力学），还提供了对机器人控制、规划和仿真至关重要的附加功能。在本文中，我们描述了这些功能，并详细说明了使 Pinocchio 高效的编程模式和设计。我们评估了与 RBDL（机器人学社区内广泛传播的另一个框架）的性能对比。我们还展示了嵌入在 Pinocchio 中的源代码生成如何优于其他最先进的方法。

1 引言

刚体动力学是机器人学中非常有用的工具。尽管该理论可以追溯到 18 世纪 [1]，但当前的算法最近已被重新审视 [2]。它们允许以高效的方式计算刚体系统的逆向和正向动力学。这两个函数对于机器人系统的控制和仿真确实都是基础的。能够快速准确地计算它们对于正确规划和控制复杂系统（如人形机器人或四足机器人）的运动至关重要。

从 80 年代末开始，第一批实现主要基于代码生成：元程序首先生成专门用于特定机器人模型的源代码，该模型作为输入给出。然后将源代码编译成目标代码，其中模型参数是硬编码的。许多开源和闭源框架都遵循这一理念：SD/Fast [3]、ROBOTRAN [4]、HuMAnS [5]、Symoro [6]、METAPOD

[7]、RobCoGen [8] 等等。这种方法具有允许简化数学表达式的优点，从而避免不必要的内存分配或使用临时变量。但与此同时，这种方法缺乏灵活性：如果机器人模型发生任何轻微修改，代码必须从头重新生成。

随着更多计算资源在台式计算机上可用，另一种范式已经出现，即一次性生成一个编译库，能够在运行时加载机器人模型的描述文件（运动链、质量分布等）。许多最近的框架实现了这一范式：RBDL [9]、OpenHRP [10]、SymBody [11]、RigidBodyDynamics.jl [12]、Drake [13]、Bullet [14]、DART [15] 等。第二种方法更加通用，例如允许在运行时修改模型的动态属性或向模型添加新功能。但这意味着计算效率的损失，尽管其中一些框架的执行时间接近代码生成 [7]、[9]。

在本文中，我们介绍了一个名为 Pinocchio 的新刚体动力学框架。与所有其他现有框架不同，Pinocchio 同时遵循上述两种范式。Pinocchio 是一个能够在运行时加载任何机器人模型的动态库。效率与其他动态框架 [7]、[9] 相当，几乎匹配代码生成框架。它能够生成特定于机器人的源代码（也在运行时）。在这种情况下，它优于任何其他现有框架。

在第二节中，我们给出了框架的全局概述。代码实现的详细信息在第三节中提供。第四节给出了开始使用 Pinocchio 的入门教程。第五节报告了 Pinocchio 与类似框架的性能对比。第六节提供了基于或使用 Pinocchio 的项目摘要。最后，第七节总结了本文。

作者隶属于 Gepetto 团队，机器人学系，系统分析与架构实验室，CNRS 和图卢兹大学，图卢兹，法国。
通讯作者：Justin Carpentier (justin.carpentier@inria.fr)

2 Pinocchio 的主要特性

Pinocchio 出于效率原因用 C++ 编写，并使用 Eigen 库 [16] 进行线性代数例程。它附带 Python 绑定，便于代码原型设计。在本节的其余部分，我们介绍 Pinocchio 中实现的主要功能。

2.1 空间代数

空间代数 [2] 是刚体动力学中常用的数学符号，用于表示和操作物理量，如速度、加速度和力。Pinocchio 基于这种数学符号。提供了专用类来表示 3D 欧几里得空间中的坐标变换（命名为 SE3）、空间运动向量（Motion）、空间力向量（Force）和空间惯性（Inertia）。结合可用的方法，这使 Pinocchio 成为一个用于空间代数计算的高效软件库。

2.2 模型和数据

Pinocchio 的一个基本范式是模型和数据之间的严格分离。模型是指机器人的物理描述，包括定义其结构的运动学和可能的惯性参数。这些信息由专用类保存，一旦创建，就永远不会被 Pinocchio 的算法修改。数据是指作为计算结果的所有值。数据根据系统的关节配置、速度等而变化。例如，它包含每个连杆的速度和加速度。它还存储中间计算和算法的最终结果，以防止动态内存分配。通过这种分离，Pinocchio 中的所有算法都遵循签名：

```
algorithm(model, data, arg1, arg2, ...)
```

其中 arg1, arg2, ... 是函数的参数（例如配置或速度向量）。保持模型和数据分离可以在同一机器人上执行多个不同任务时减少内存占用，特别是在涉及并行计算时。每个进程可以使用自己的数据对象，同时共享相同的模型对象。模型对象在 Pinocchio 的算法中永远不会改变这一事实增强了代码的可预测性。

可以使用 C++ API 创建模型，或从外部文件加载，该文件可以是 URDF、Lua（遵循 RBDL 标准）或 Python。

2.3 支持的运动学模型

在模型中，机器人被表示为运动树，包含所有关节的集合、关于它们连接性的信息，以及可选地与每个连杆相关的惯性量。在 Pinocchio 中，关节可以有一个或多个自由度，它属于以下类别之一：旋转关节，围绕固定轴旋转，可以是 X、Y、Z 之一或自定义轴；移动关节，沿任何固定轴平移，与旋转情况相同；球关节，在 3D 空间中的自由旋转；平移关节，用于 3D 空

间中的自由平移；平面关节，用于 2D 空间中的自由运动；自由浮动关节，用于 3D 空间中的自由运动。平面和自由浮动关节旨在用作移动机器人（人形机器人、自动驾驶车辆或操作规划中的对象）运动树的基础。更复杂的关节可以通过复合关节的概念作为普通关节的集合来创建。

2.4 处理李群几何

每种类型的关节都有其特定的配置空间和切空间。例如，旋转关节的配置空间和切空间都是实轴 R，而对于球关节，配置空间对应于 3 维旋转矩阵的集合，其切空间是 3 维实向量 R3 的空间。某些配置空间可能不像向量空间那样表现，但必须配备相应的积分（exp）和微分（log）算子。Pinocchio 实现了所有这些特定的积分和微分算子。

2.5 几何模型

除了运动学模型，Pinocchio 还定义了几何模型，即附加到运动树的体积。该模型可用于显示机器人和计算与碰撞相关的量。与运动学模型一样，固定量（体积的位置和形状）存储在 GeometricModel 对象中，而关联算法使用的缓冲区和量定义在 GeometricData 对象中。体积使用 FCL 库 [17] 表示。机器人的主体附加到每个关节，而环境的障碍物在世界坐标系中定义。基于 FCL 方法实现了运动树的碰撞和距离算法。

注：Pinocchio 实际上使用 FCL 0.3.1 版本的分支。

2.6 主要算法

基本算法的实现，包括本节列出的所有算法，都是递归的。递归公式允许软件避免重复计算并利用运动树引起的稀疏性。对于动力学算法，我们主要从 [2] 中汲取灵感，并进行了轻微改进。

2.6.1 正向运动学

Pinocchio 实现了直到二阶的直接运动学计算。当给定机器人配置时，执行前向传递以计算每个关节的空间位置并将它们存储为坐标变换。如果

给定速度，它还计算每个关节的空间速度（在局部坐标系中表示），加速度也是如此。

2.6.2 运动学雅可比

每个关节的空间雅可比可以通过单次前向传递轻松计算，可以在局部坐标系或世界坐标系中表示。

2.6.3 逆向动力学

递归牛顿-欧拉算法（RNEA）[18] 计算逆向动力学：给定期望的机器人配置、速度和加速度，计算并存储执行此运动所需的扭矩。该算法首先执行前向传递（相当于二阶运动学）。然后它执行后向传递，计算沿结构传递的力螺旋并提取获得计算的连杆运动所需的关节扭矩。使用适当的输入，该算法还可以用于计算动态模型的特定项，例如重力效应。

2.6.4 关节空间惯性矩阵

复合刚体算法（CRBA）[19] 用于计算机器人的关节空间惯性矩阵。我们对原始算法进行了一些轻微修改，提高了计算效率。

2.6.5 正向动力学

关节体算法（ABA）[20] 计算无约束正向动力学：给定机器人配置、速度、扭矩和外部力，计算产生的关节加速度。

2.6.6 其他算法

除了上述算法外，还提供了其他方法，最值得注意的是约束正向动力学、脉冲动力学、关节空间惯性的逆 [21] 和质心动力学。

2.7 解析导数

除了提出标准的正向和逆向动力学算法外，Pinocchio 还提供了它们的解析导数的有效实现 [22]。这些导数在全身轨迹优化或更广泛的数值最优

控制的背景下至关重要。据我们所知，Pinocchio 是第一个原生实现此功能的刚体框架。

2.8 自动微分和源代码生成

除了解析导数，Pinocchio 还支持自动微分。这是通过整个 C++ 代码的完全标量模板化以及使用任何进行自动微分的外部库来实现的：ADOL-C [23]、CasADi [24]、CppAD [25] 等。重要的是要记住，这些自动导数通常比解析导数慢得多。

Pinocchio 的另一个独特但核心特性是它能够在编译时和运行时生成代码。这是通过使用另一个名为 CppADCodeGen 的外部工具箱实现的，该工具箱构建在 CppAD [25] 之上。从任何使用 Pinocchio 的函数，CppADCodeGen 能够动态生成各种语言的代码：C、LaTeX 等，并对数学表达式进行一些简化。通过此过程，可以生成针对特定机器人模型的代码，并在 Pinocchio 外部使用。

3 什么使 Pinocchio 快速

在本节中，我们详细说明了我们在 Pinocchio 中实现的编程范式，以使框架既高效又通用。

3.1 处理稀疏性

每个关节根据定义将两个主体之间的运动限制为某些特定的运动方向。这特别意味着每个关节可以配备其特定的算子和状态表示，以实现最小的内存占用和计算次数。例如，旋转关节变换由单个标量值表示，即围绕其轴的旋转，而球关节变换编码为旋转矩阵。对于表征关节状态的其他空间量，如空间速度或加速度、关节约束等，可以做出类似的观察。因此，Pinocchio 中的每个关节都配备了其自己的空间量的稀疏描述。结合空间算子的重载，这使我们能够在计算级别充分利用每个关节固有的稀疏性。

3.2 静态多态性

多态性的概念使我们能够充分利用关节引起的稀疏性。Pinocchio 类广泛使用继承和多态性。例如，所有不同的关节模型都实现为 JointModelBase 的子类，它定义了所有关节的通用 API。然后在每个关节模型类中专门化方法和数据结构。

在 Pinocchio 中，我们选择通过静态多态性来实现此行为，与动态多态性（在 C++ 中通过虚方法实现多态性的传统方式）形成对比。在动态多态性中，当调用方法时，在运行时推断对象类，然后执行适当的方法。这具有破坏现代 CPU 预测机制的主要缺点。相反，在静态多态性中，直接在编译时选择适当的函数。采用这种范式在许多方面提高了效率。首先，避免了动态多态性典型的双重重定向，以及运行时类推断。其次，由于类信息在编译时已知，编译器被允许优化代码以提高效率。

静态多态性通过所谓的奇异递归模板模式（CRTP）实现，这是我们框架的核心。这也是 Eigen 库采用的设计模式，极大地促进了其性能和通用性。我们现在通过一个简单的例子介绍 CRTP 的概念，该例子描述了 Pinocchio 中关节类的架构。根据这种设计模式，定义所有关节的通用方法和属性的关节模型基类 JointModelBase 由其子类模板化：

```
template<typename Derived>
struct JointModelBase<Derived> {
    void calc(q,v) {
        // 调用 Derived 类的 calc 方法
        static_cast<Derived*>(this)->calc();
    }
};
```

然后所有关节模型类继承此基类，如下所示：

```
struct JointModelRevolute
    : JointModelBase<JointModelRevolute> {
    void calc(q,v) {
        // 执行特定计算
    }
```

```
};
```

通过这种方式，JointModelBase 用于定义所有关节模型特定算子的原型，然后在子类中实现。反过来，这允许开发人员通过简单地编写模板函数来编写适用于所有子类的通用代码。例如，Pinocchio 中的所有算法都编写为在模型中包含的关节模型上执行的步骤序列。单个步骤可以实现为：

```
template<typename Derived>
void step(JointModelBase<Derived> joint, arg1, ...) {
    joint.calc(arg1,arg2); // 调用 Derived::calc
    ...
}
```

其中函数 step 调用在 JointModelBase<Derived> 中定义的 calc 方法，该方法直接重定向到执行计算的 Derived 类的 calc 方法。由于 Derived 的值在编译时已知，现代编译器能够消除这种间接级别并直接调用 Derived 类中的方法。对于在 step 内使用的所有其他类似实现的方法也会发生同样的情况。然后为每个关节类编译不同版本的 step，每个版本静态链接到关节特定方法并避免使用动态重定向。

这种增强的性能伴随着灵活性的损失。从概念上讲，当使用这种编码范式时，两个不同的派生类不会继承自同一个基类。例如，JointModelBase<JointModelRevolute> 是 JointModelRevolute 的基类，而 JointModelBase<JointModelPlanar> 是 JointModelPlanar 的基类。这意味着 JointModelRevolute 和 JointModelPlanar 不能转换为同一个父类，就像动态多态性通常所做的那样。因此，既不可能创建 JointModelBase 对象的向量，也不可能在编译时具有未知类型的关节。

为了恢复动态多态性允许的灵活性，我们诉诸变体的概念。变体是一个类，可用于表示有限预定类型集中的任何类型。最重要的是，变体保留有关所表示类型的运行时信息。在 Pinocchio 中，我们定义了一个名为 JointModelVariant 的变体，能够表示任何关节。关节在 Pinocchio 模型中存储为变体对象的集合。通过这种方式，当执行算法时，根据每个关节的类型在运行时选择适当的步骤。然而，如上所述，每个单独的步骤都针对手头的特定关节类型进行了完全优化。

由于 CRTP 和变体范式的结合，代码灵活性得以恢复，同时对于每个算法，由于类重定向导致的总体运行时开销被减少到最小。

4 开始使用 Pinocchio

Pinocchio [26] 目前支持大多数 Linux 发行版和 Mac OS X，计划很快发布 Windows 版本。该项目在 BSD-2-Clause 许可下完全开源，目前托管在以下 GitHub 存储库中：

```
https://github.com/stack-of-tasks/pinocchio
```

4.1 文档和教程

Pinocchio 的文档在包内以及 GitHub 项目页面上提供；Python 接口已完全记录，而 C++ 文档仍在进行中。基准测试和单元测试都在包中提供。Pinocchio 的教程可以在专门的 GitHub 存储库中找到：

```
github.com/stack-of-tasks/pinocchio-tutorials
```

教程主要是 Python 的，特别适合作为机器人学课程教育者的支持。

4.2 安装

Pinocchio 版本由 robotpkg 管理，这是一个为机器人软件量身定制的包管理器，在大多数 Unix 和 BSD 平台上可用。安装过程的详细信息可以在 GitHub 项目页面上找到。在这里，我们只提供一个概述。

在 Ubuntu 上，robotpkg 管理的包的软件二进制文件可通过 robotpkg apt 存储库直接获得。将存储库添加到可用源列表后，Pinocchio、其 Python 绑定以及所有必需的依赖项可以通过以下命令简单安装：

```
apt-get install robotpkg-py27-pinocchio
```

教程包括有关安装过程的详细信息，以及自动执行所有安装步骤的脚本。

在 Mac OS X 上，支持通过 Homebrew 包管理器安装 Pinocchio。只需发出以下命令：

```
brew tap gepetto/homebrew-gepetto
brew install pinocchio
```

另一个选项是直接从源代码安装框架,可以从 Pinocchio 的官方 GitHub 存储库下载。

4.3 小例子：在 ATLAS 机器人上运行 RNEA

在下文中，我们演示了如何使用 Pinocchio 在 Python 中计算人形机器人 Atlas 在随机输入值上的逆向动力学。

```
import pinocchio as se3, numpy as np
root = se3.JointModelFreeFlyer()
model = se3.buildModelFromUrdf('atlas.urdf',root)
data = model.createData()
qmax = np.matrix(np.full([model.nq, 1], np.pi))
q = se3.randomConfiguration(model,-qmax,qmax)
v = np.matrix(np.random.rand(model.nv,1))
a = np.matrix(np.random.rand(model.nv,1))
tau = se3.rnea(model,data,q,v,a)
```

5 结果

5.1 设置

我们首先评估 Pinocchio 在 8 个不同机器人模型上的性能，以及主要使用的算法，即 RNEA（逆向动力学）、ABA（正向动力学）和 CRBA（质量矩阵）。Pinocchio 的性能还与 RBDL [9]（另一个流行且高效的 C++ 框架）进行了比较。每个测试用例在配备 Intel Core I7 CPU @2.4 GHz 的标准笔记本电脑上使用随机输入值运行 100,000 次。

测试使用 7 自由度机械臂 KUKA LWR 4+ [27]、简单的 22 自由度腿式人形模型、人形机器人 Nao (www.softbankrobotics.com/emea/en/robots/nao)、Poppy (www.poppy-project.org)、Atlas (www.bostondynamics.com/atlas) 和

TALOS [28]、四足机器人 HyQ [29] 以及移动机械臂 TIAGo (www.tiago.pal-robotics.com) 运行。

图 1: 图 1：Pinocchio 的 RNEA、ABA 和 CRBA 在各种机器人上的性能。

图 2: 图 2：Pinocchio 与 RBDL 在 LWR 机器人（顶部）和 ATLAS 机器人（底部）模型上的性能比较。

图 3: 图 3：Pinocchio 生成的代码的性能（与相同算法的动态版本相比）。

5.2 结果

绝对性能在图 1 中绘制在 8 个机器人模型上。Pinocchio 在机械臂机器人上评估动力学需要约 1 µs，在腿式机器人上需要约 3 µs。与 RBDL 的性能对比在图 2 中报告了 2 个代表性模型。Pinocchio RNEA 与 RBDL 相似，但其 ABA 和 CRBA 实现优于 RBDL。这第一组结果是使用模型的动态加载获得的。

然后我们将这些分数与为给定输入模型生成专用源代码时的性能进行比较，如图 3 所示。代码生成将计算时间除以 3（对于复杂机器人，如人形机器人或四足机器人）到 8（对于更简单的机器人，如 LWR 机械臂）。执行时间的差异可能是由于缓存效应和预测的必要性。因此，可以在略多于 1µs 的时间内评估复杂腿式机器人的动力学。理论和实践之间可以提出另一个重要点。确实，从理论角度来看，已知 ABA 在操作次数方面比 RNEA 具有更大的复杂度 [2]。然而，在实践中，由于代码生成，RNEA、ABA 以及 CRBA 似乎具有相似的计算成本。

最后，Pinocchio 还提供了 RNEA 和 ABA 等导数的实现。我们在图 4 中报告了这些算法在动态加载模型时的性能。

图 4: 图 4：RNEA 和 ABA 的导数性能，针对 4 个机器人模型，与有限差分评估相比。

6 框架传播

Pinocchio 在 [30] 中以及后来在 [31]、[32] 中被用于生成 HRP-2 机器人的全身运动。这个具有多个接触点的爬楼梯实验已在真实硬件上运行约 100 次，成功率约为 80%。由于其通用性，它目前用于人形机器人 Pyrene (TALOS-01) [28] 以执行运动学和动力学任务。它还在 [33] 中被用于在不到 1 ms 的时间内执行 150 次 RNEA 运行以实现动态滤波器。这是提高反应式步行模式生成器能力的关键点。Pinocchio 已被用于生成全身最优控制，既用于协同设计方法 [34]，也用于快速步行运动 [35]。它还用于在人形路径规划器（HPP）软件 [36] 中对各种机器人执行操作规划：UR5、PR-2 和 Romeo。它最近还在 HPP 框架内用于规划 HRP-2 上动态可行的接触序列 [37]、[38]。

7 结论

我们介绍了 Pinocchio，一个实现刚体动力学算法及其解析导数的新、快速且灵活的框架。Pinocchio 在动态使用时表现出与所有其他类似库相当或更好的性能。在代码生成的支持下，差距进一步扩大，其中时间性能优于最先进的方法。

作为下一个里程碑，我们计划在 Pinocchio 中集成传动模型（齿轮、滑轮、肌腱等）和驱动模型（气动肌肉、生物肌肉、电机等），以便例如在运动动力学方程中考虑它们的物理效应。这将使 Pinocchio 不仅能够处理机器人系统，还能够处理生物系统的仿真，这对于理解例如拟人运动的基础至关重要 [39]。

8 致谢

我们首先感谢 Eigen、CppAD 和 CppADCodeGen 的所有开发人员，他们提供了高效、高质量的开源框架。我们还感谢所有直接或间接参与 Pinocchio 库开发的所有贡献者。这项工作得到了欧洲研究理事会（ERC）通过 Actanthrope 项目（资助 340050）和欧盟委员会通过 MEMMO 和 RoboCom++ 项目的支持。

参考文献

[1] J.-L. Lagrange, Mécanique analytique. Académie Royale des Sciences, 1788.

[2] R. Featherstone, Rigid Body Dynamics Algorithms. Springer, 2008.

[3] M. G. Hollars, D. E. Rosenthal, and M. A. Sherman, "SD/FAST user's manual," 1991.

[4] N. Docquier, A. Poncelet, and P. Fisette, "Robotran: a powerful symbolic generator of multibody models," Mechanical Sciences, vol. 4, no. 1, pp. 199–219, 2013.

[5] P.-B. Wieber, F. Billet, L. Boissieux, and R. Pissard-Gibollet, "The HuMAnS toolbox, a homogenous framework for motion capture, analysis and simulation," in International Symposium on the 3D Analysis of Human Movement, 2006.

[6] W. Khalil, A. Vijayalingam, B. Khomutenko, I. Mukhanov, P. Lemoine, and G. Ecorchard, "Opensymoro: An open-source software package for symbolic modelling of robots," in IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1206–1211, 2014.

[7] M. Naveau, J. Carpentier, S. Barthelemy, O. Stasse, and P. Souères, "METAPOD — Template META-PrOgramming applied to dynamics:

CoP-CoM trajectories filtering," in IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2014.

[8] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on Domain Specific Languages," Journal of Software Engineering for Robotics (JOSER), vol. 7, no. 1, pp. 36–54, 2016.

[9] M. L. Felis, "RBDL: an efficient rigid-body dynamics library using recursive algorithms," Autonomous Robots, 2017.

[10] F. Kanehiro, H. Hirukawa, and S. Kajita, "Openhrp: Open architecture humanoid robotics platform," The International Journal of Robotics Research, vol. 23, no. 2, pp. 155–165, 2004.

[11] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: multibody dynamics for biomedical research," Procedia Iutam, vol. 2, pp. 241–261, 2011.

[12] T. Koolen and contributors, "RigidBodyDynamics.jl," 2016.

[13] R. Tedrake and the Drake Development Team, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016.

[14] E. Coumans, "Bullet Physics Simulation," in ACM SIGGRAPH 2015 Courses, 2015.

[15] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," The Journal of Open Source Software, vol. 3, no. 22, p. 500, 2018.

[16] G. Guennebaud, B. Jacob, et al., "Eigen v3," 2010.

[17] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 3859–3866, IEEE, 2012.

[18] J. Y. Luh, M. W. Walker, and R. P. Paul, "On-line computational scheme for mechanical manipulators," Journal of Dynamic Systems, Measurement, and Control, vol. 102, no. 2, pp. 69–76, 1980.

[19] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," Journal of Dynamic Systems, Measurement, and Control, 1982.

[20] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," The International Journal of Robotics Research, 1983.

[21] J. Carpentier, "Analytical inverse of the joint space inertia matrix," tech. rep., Laboratoire d'Analyse et d'Architecture des Systèmes, 2018.

[22] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in Robotics: Science and Systems, 2018.

[23] A. Griewank, D. Juedes, and J. Utke, "ADOL-C: a package for the automatic differentiation of algorithms written in C/C++," ACM Transactions on Mathematical Software (TOMS), vol. 22, no. 2, pp. 131–167, 1996.

[24] J. Andersson, J. Åkesson, and M. Diehl, "CasADi: A symbolic package for automatic differentiation and optimal control," in Recent advances in algorithmic differentiation, pp. 297–307, Springer, 2012.

[25] B. Bell, "CppAD: a package for C++ algorithmic differentiation," 2005–2018.

[26] J. Carpentier, F. Valenza, N. Mansard, et al., "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," 2015–2018.

[27] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR lightweight robot arm - a new reference

platform for robotics research and manufacturing," in 41st International Symposium on Robotics, June 2010.

[28] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. D. Prete, P. Souères, N. Mansard, F. Lamiraux, J. . Laumond, L. Marchionni, H. Tome, and F. Ferro, "TALOS: A new humanoid research platform targeted for industrial applications," in Int. Conf. on Humanoid Robotics (Humanoids), 2017.

[29] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 225, no. 6, pp. 831–849, 2011.

[30] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in IEEE/RAS Int. Conf. on Robotics and Automation (ICRA), 2016.

[31] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning feasibility constraints for multi-contact locomotion of legged robots," in Robotics: Science and Systems, p. 9p, 2017.

[32] J. Carpentier and N. Mansard, "Multi-contact locomotion of legged robots," IEEE Transactions on Robotics (In Press), 2018.

[33] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," IEEE/RAS Robotics and Automation Letters, vol. 2, 2017.

[34] G. Saurel, J. Carpentier, N. Mansard, and J.-P. Laumond, "A simulation framework for simultaneous design and control of passivity based

walkers," in IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2016.

[35] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics," in 2018 International Conference on Humanoid Robots, (Beijing, China), Nov. 2018.

[36] J. Mirabel, S. Tonneau, P. Fernbach, A. Sepp al a, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016.

[37] P. Fernbach, S. Tonneau, and M. Taïx, "CROC: Convex Resolution Of Centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem," in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2018.

[38] P. Fernbach, S. Tonneau, J. Carpentier, O. Stasse, and M. Taïx, "C-CROC: Continuous and Convex Resolution of Centroidal dynamic trajectories for legged robots in multi-contact scenarios." working paper or preprint, Oct. 2018.

[39] J. Carpentier, Computational foundations of anthropomorphic locomotion. PhD thesis, Université Toulouse 3 Paul Sabatier, 2017.