Openloong开源社区线下交流会

基于MPC与WBC的人形机器人控制框架

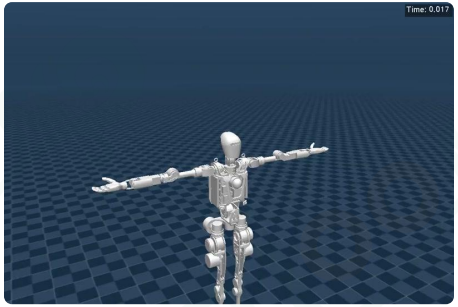
人形机器人（上海）有限公司　马雪艳

2024/10/22

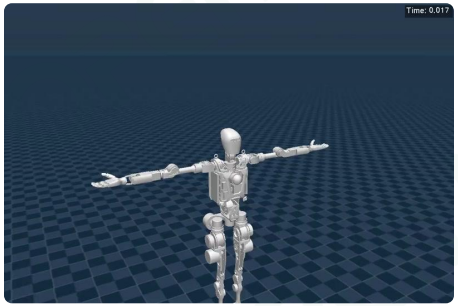
1. 控制框架概述

- Openloong开源仓库的动力学控制是基于MPC与WBC的人形控制算法，已在仿真与实物上部署并实现行走；
- 仓库地址

开放原子： https://atomgit.com/openloong/openloong-dyn-control.git
Github：https://github.com/loongOpen/openloong-dyn-control.git

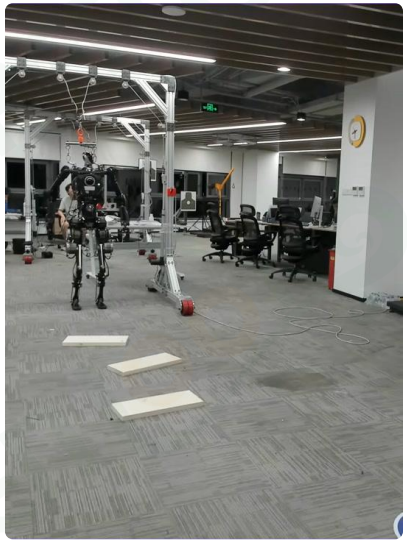


Mujoco仿真行走



Mujoco仿真盲踩障碍

实物行走

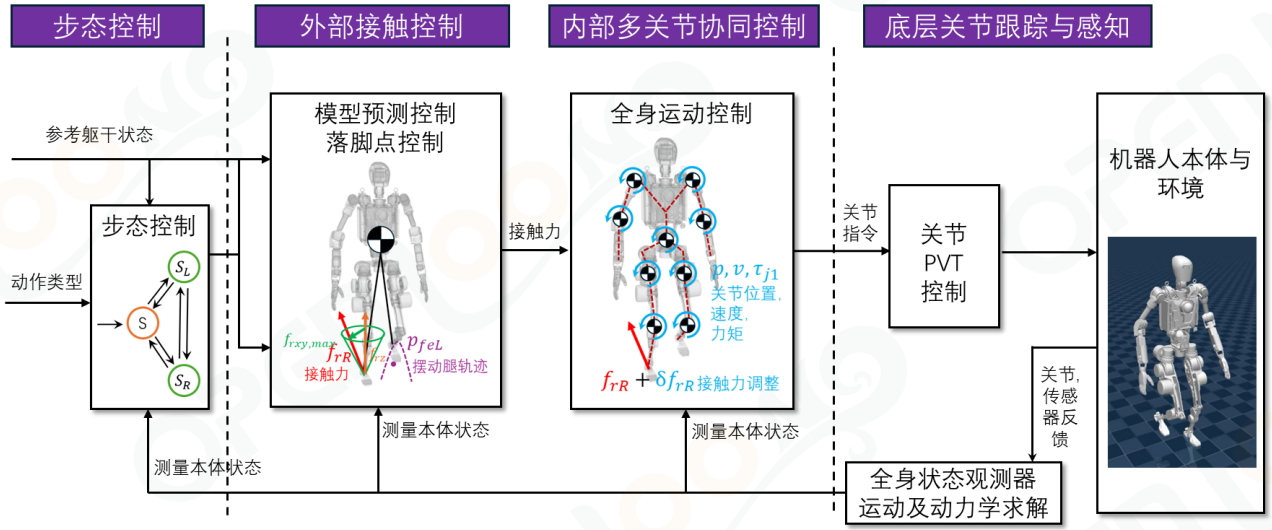


实物盲踩障碍

1 控制框架算法及开发流程概述

1. 控制框架概述

- Openloong开源仓库的动力学控制是基于MPC与WBC的人形控制算法，已在仿真与实物上部署并实现行走；
- 主要算法模块包括步态控制、**外部接触控制、内部多关节协同控制、底层关节跟踪与感知模块**；



外部接触控制

- 支撑腿：模型预测控制 MPC
- 摆动腿：固定参考轨迹

内部多关节协同控制

- 全身动力学控制（WBC）

底层关节跟踪与感知模块

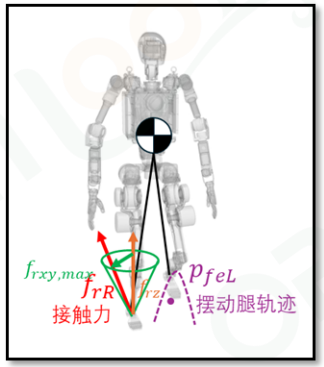
- PVT关节指令跟踪
- 基于pinocchio的运动学及动力学解算
- 全身状态观测器

2. 外部接触控制

- 基于单刚体模型，产生支撑腿的末端接触力及腾空腿的末端位置轨迹；
- 参考文献：

[1] D. Kim, J. D. Carlo, B. Katz, G. Bledt, S. Kim, Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control
[2] 卞泽坤, 王兴兴. 四足机器人控制算法: 建模、控制与实践[M]. 机械工业出版社, 2023

单刚体模型



$$\dot{x}(t) = A_c^{'}x(t) + B^{'}{}_c u(t) + C_c$$

其中：

$$A_c = \begin{bmatrix} 0_{3\times3}, 0_{3\times3}, R^T(\psi)), 0_{3\times3} \\ 0_{3\times3}, 0_{3\times3}, 0_{3\times3}, E_{3\times3} \\ 0_{3\times3}, 0_{3\times3}, 0_{3\times3}, 0_{3\times3} \\ 0_{3\times3}, 0_{3\times3}, 0_{3\times3}, 0_{3\times3} \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0_{3\times3}, 0_{3\times3}, 0_{3\times3}, 0_{3\times3} \\ 0_{3\times3}, 0_{3\times3}, 0_{3\times3}, 0_{3\times3} \\ I^{-1}[p_r]_\times, I^{-1}, I^{-1}[p_l]_\times, I^{-1} \\ E_{3\times3}/m, 0_{3\times3}, E_{3\times3}/m, 0_{3\times3} \end{bmatrix}$$

$$C_c = \begin{bmatrix} 0_{3\times1} \\ 0_{3\times1} \\ 0_{3\times1} \\ [0, 0, g]^T \end{bmatrix}$$

支撑腿控制（MPC）

目标为与期望误差尽量小、系统输入尽量小

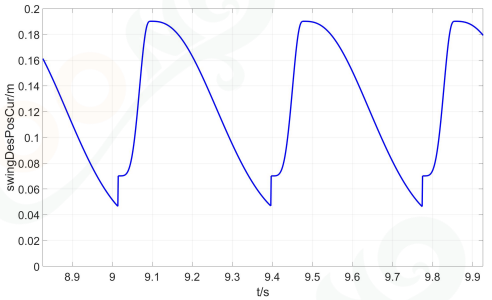
$$\min_{x,u} \sum_{i=0}^{k-1} (||x_{i+1} - x_{i+1\cdot ref}||_{Q_i} + ||u_i||_{R_i}$$

约束：

- z方向的半封闭约束
- 摩擦力约束
- 力矩约束-防止足底反转
- 力矩约束-防止足底扭转

腾空腿控制（基于速度）

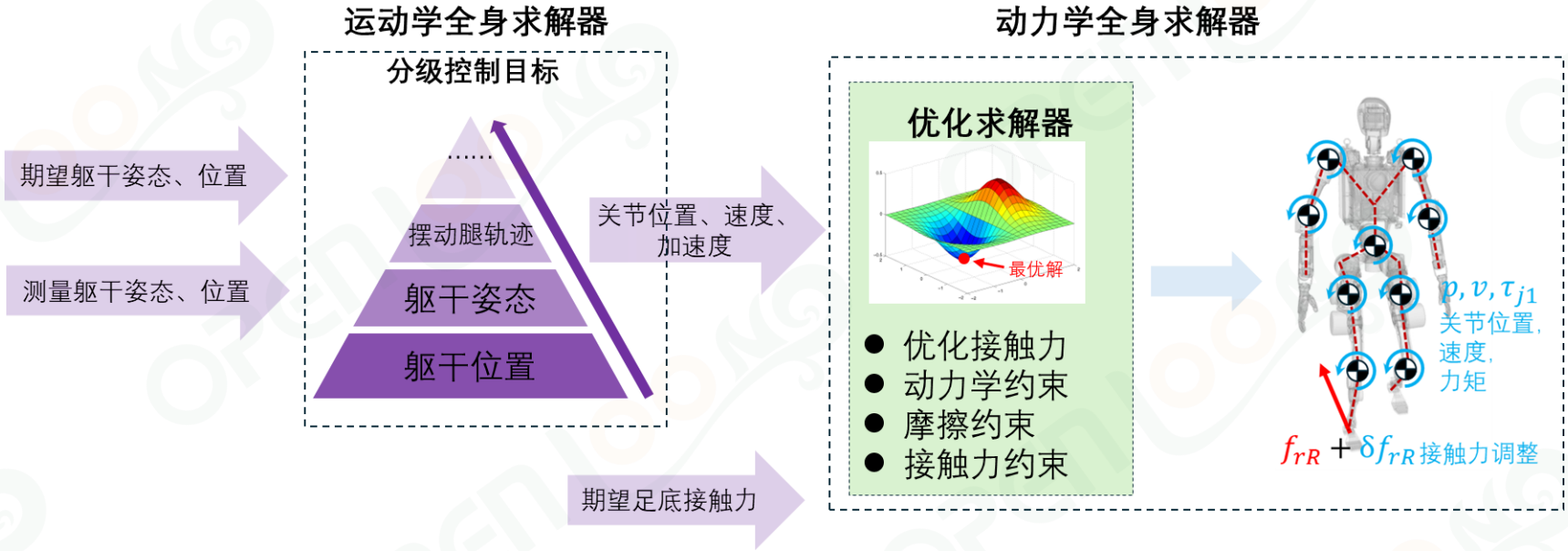
- 机身速度进行x y期望落脚点的计算；
- 应用贝塞尔曲线组合生成相位可调的z方向轨迹



3. 内部多关节协同控制

- 基于全身动力学模型，将外部接触控制生成的末端力矩速度位置参考映射到关节层面；
- 参考文献：

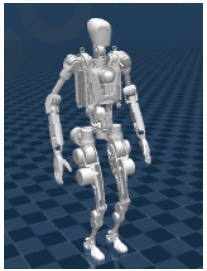
[1] D. Kim, J. D. Carlo, B. Katz, G. Bledt, S. Kim, Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control
[2] Kim D, Jorgensen S J, Lee J, et al. Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control. arXiv:1901.08100 (2020).



4. 开发流程概述

- **更换模型仿真开发流程**
  - **仿真环境搭建**
    - 导出xml模型
    - 模型PD的对应
  - **算法更改** 参考仓库文档tutorial.md

  - **仿真测试与调参**

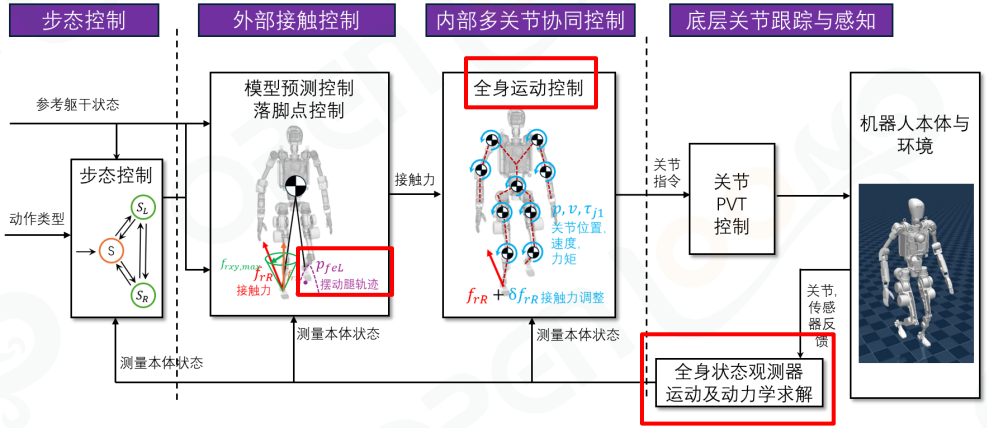


- **实物开发流程**
  - **单关节测试**
    - 电机电流扭矩标定
    - 关节PVT模式调试
  - **上电无动作测试**
    - 控制器实时框架
    - 遥控器测试
    - 电机、传感器数据采集测试
    - 数据记录测试
    - 多关节零点及限位标定
  - **悬空测试**
    - 关节刚度整定
    - 悬空动作进行运动学测试
  - **站立行走测试**
    - 站立测试
    - IMU零位
    - 单踏步测试
    - 状态估计离线调参
    - 算法迁移
    - 连续踏步、行走

2 Pinocchio动力学求解库的应用及逆运动学的求解

1. pinocchio概述

- https://stack-of-tasks.github.io/pinocchio/
- pinocchio是基于C++的开源动力学求解库，同时也提供了python的接口，可以根据输入的URDF模型进行机器人运动学及动力学的参数计算
- 相关安装步骤可参考官网，也可使用openloong-dyn-control中的third party
- 具体功能如下：
- 动力学方程参数的求解；
- 逆运动学解算；
- 正运动学求解；
- 雅可比矩阵求解。



- Pinocchio计算结果在控制算法中的应用
- 正逆运动学的求解
- WBC任务优先级的雅可比矩阵
- WBC QP求解中动力学约束
- 状态估计的足底接触力估计

2. 动力学方程参数的获取

- 人形机器人全身动力学模型如下：

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{f}_{ext}$$

人形机器人一般为浮动基模型，广义坐标q包含了质心位置、质心姿态、关节位置信息，

$\boldsymbol{q}$ = [global_base_position, global_base_quaternion (x, y, z, w), joint_positions]

$\dot{\boldsymbol{q}}$ = [local_base_velocity_linear, local_base_velocity_angular, joint_velocities]

值得注意的是，q中姿态采用四元数表示有4维，但dq中角速度只有3维，因此q比dq的维数多1。

	pinocchio::crab(model, data, q)	data.M
	pinocchio::computeCoriolisMatrix(model, data, q, dq)	data.C
	pinocchio::computeGeneralizedGravity(model, data, q)	data.g
I	pinocchio::ccrba(model, data, q, dq)	data.Ig
	pinocchio::ccrba(model, data, q, dq)	data.com[0]

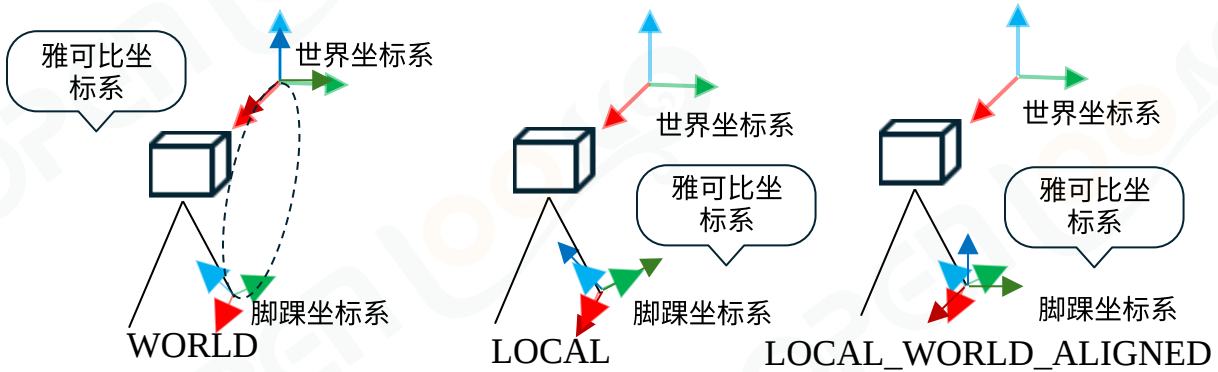
- 注意点：
- Pinocchio中默认的q关节顺序，根据urdf中joint的名字及串联关系确定的，先根据串联关系，然后根据其首字母进行排列，算法中的关节顺序最好与此对应；
- dq为本体坐标系下的速度。
- urdf关节类型要选revolute而不是continuous，不然会q的个数会多一个

3. 正运动学、雅可比矩阵的获取

pinocchio::forwardKinematics(model, data, q)	data.oMi[Joint_index].translation(), data.oMi[Joint_index].rotation()
pinocchio::computeJointJacobians(model, data, q)	getJointJacobian(model, data, Joint_index, pinocchio::LOCAL, J_r)

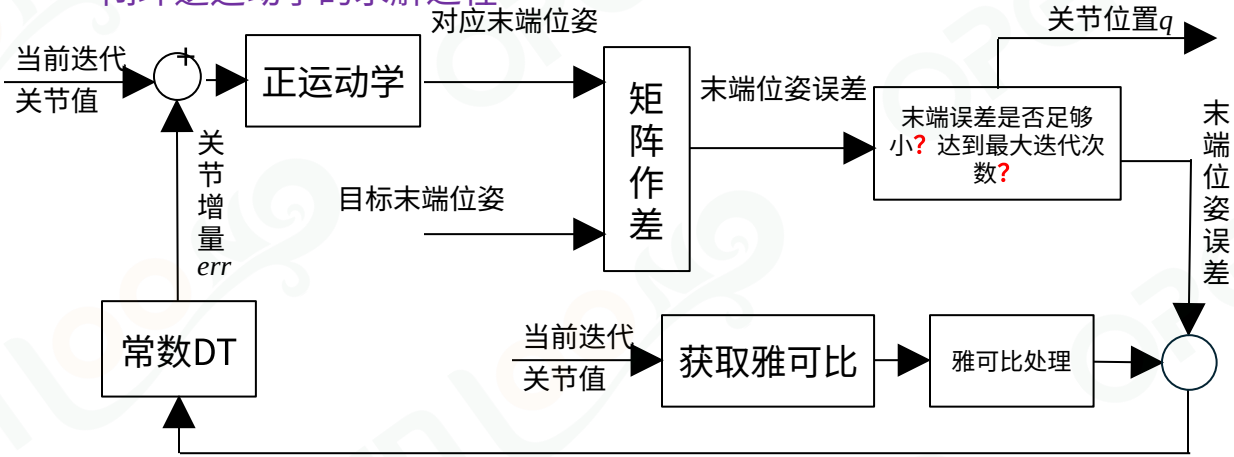
- 注意点：

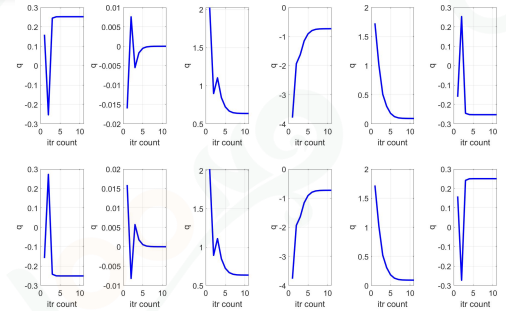
- 获取雅可比矩阵前，需先调用正运动学计算，如果有增加的坐标，在调用正运动学后也需要更新
- Pinocchio中有三种坐标形式，即LOCAL、WORLD、LOCAL_WORLD_ALIGNED,在求解雅可比矩阵时，使用不同的坐标形式，求取的雅可比矩阵计算出的末端速度不同：
  a. WORLD：坐标轴朝向与世界坐标系平行，原点位于机身最初的base处；
  b. LOCAL：与雅可比求解的末端坐标重合；
  c. LOCAL_WORLD_ALIGNED：坐标原点为雅可比求解的末端坐标原点，方向与世界坐标系同相。



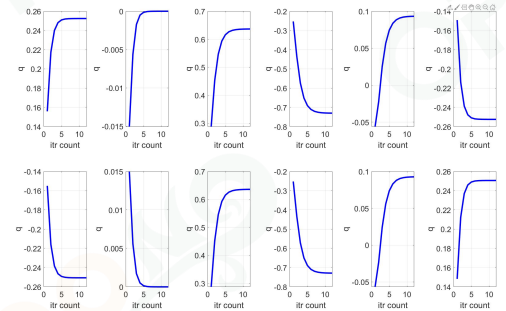
4. 逆运动学求解

- 闭环逆运动学的求解过程





雅可比直接求伪逆，迭代中各关节位置求解变化



雅可比处理后，迭代中各关节位置求解变化

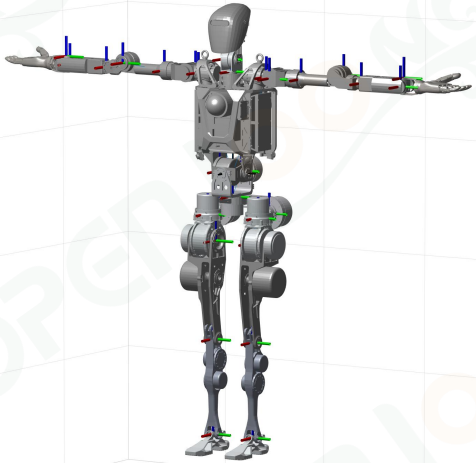
- 注意点：
- 获取关节增量的方式为:

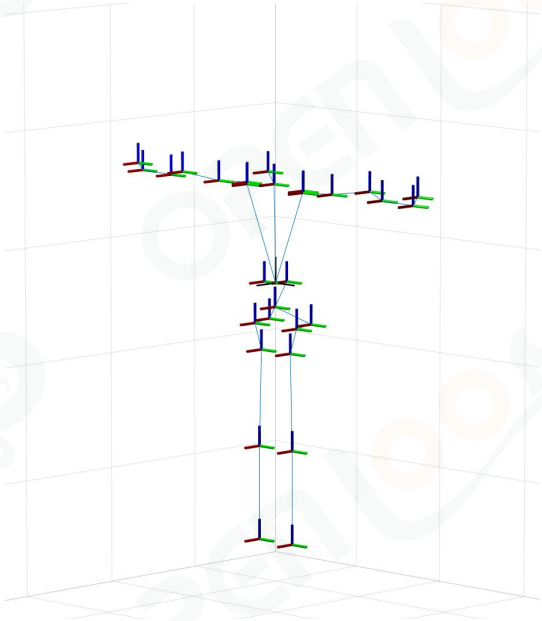
这是为了避免奇点问题，采用阻尼伪逆，可减少求解时的震荡，两种方式求解时，迭代过程中的关节角的变化如右图。

- DT起到类似增益作用，过大的关节增量err会导致超调，需要选取合适的值，令结果快速收敛并且不会超调

5. 实例讲解

- 青龙全身具备31个自由度，关节顺序为左侧手臂（7），右侧手臂（7），头（2），腰（3），左腿（6），右腿（6）





Thank you!

欢迎交流！