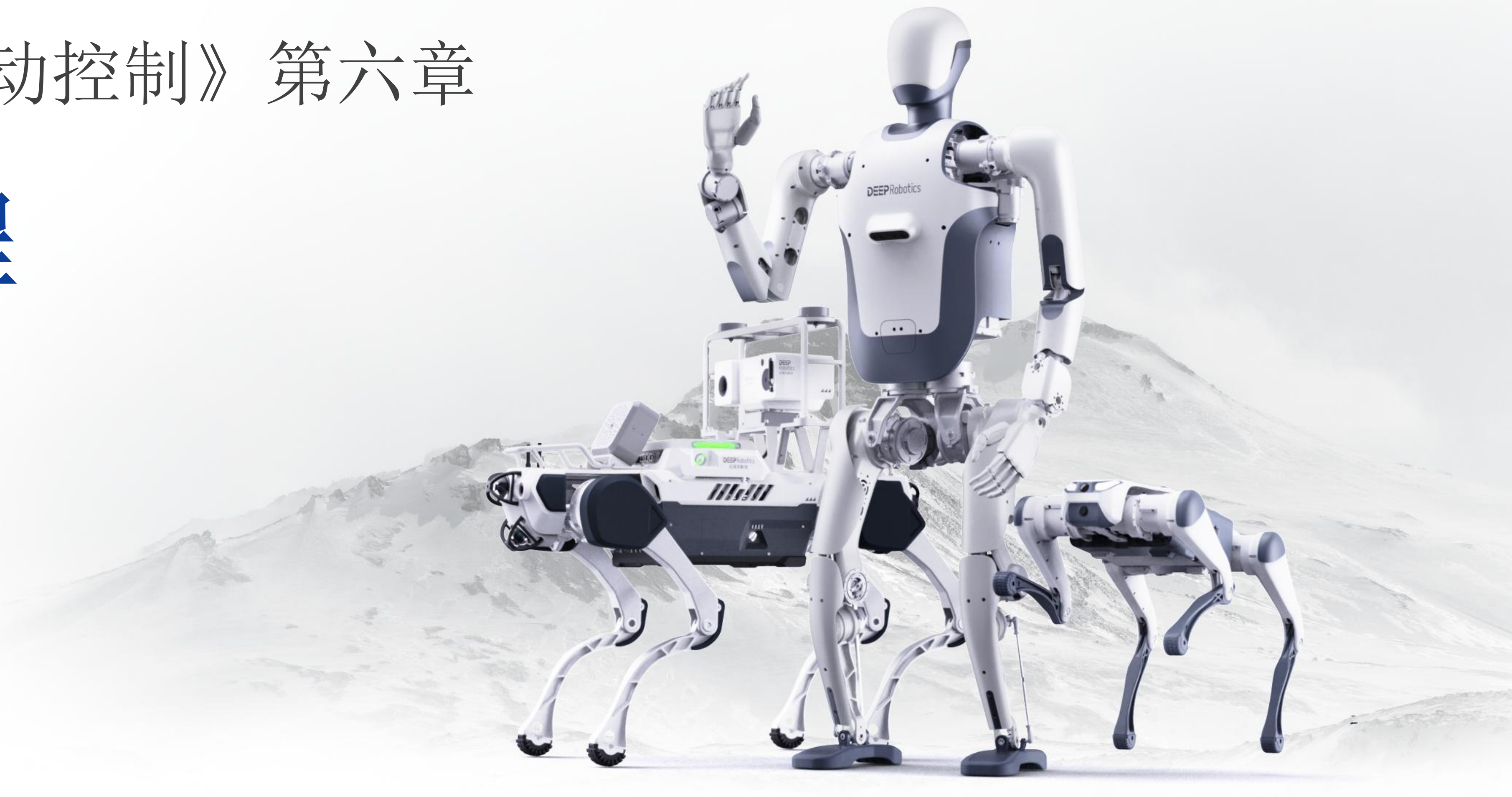
深蓝学院 shenlanxueyuan.com × 云深处科技 DEEP Robotics

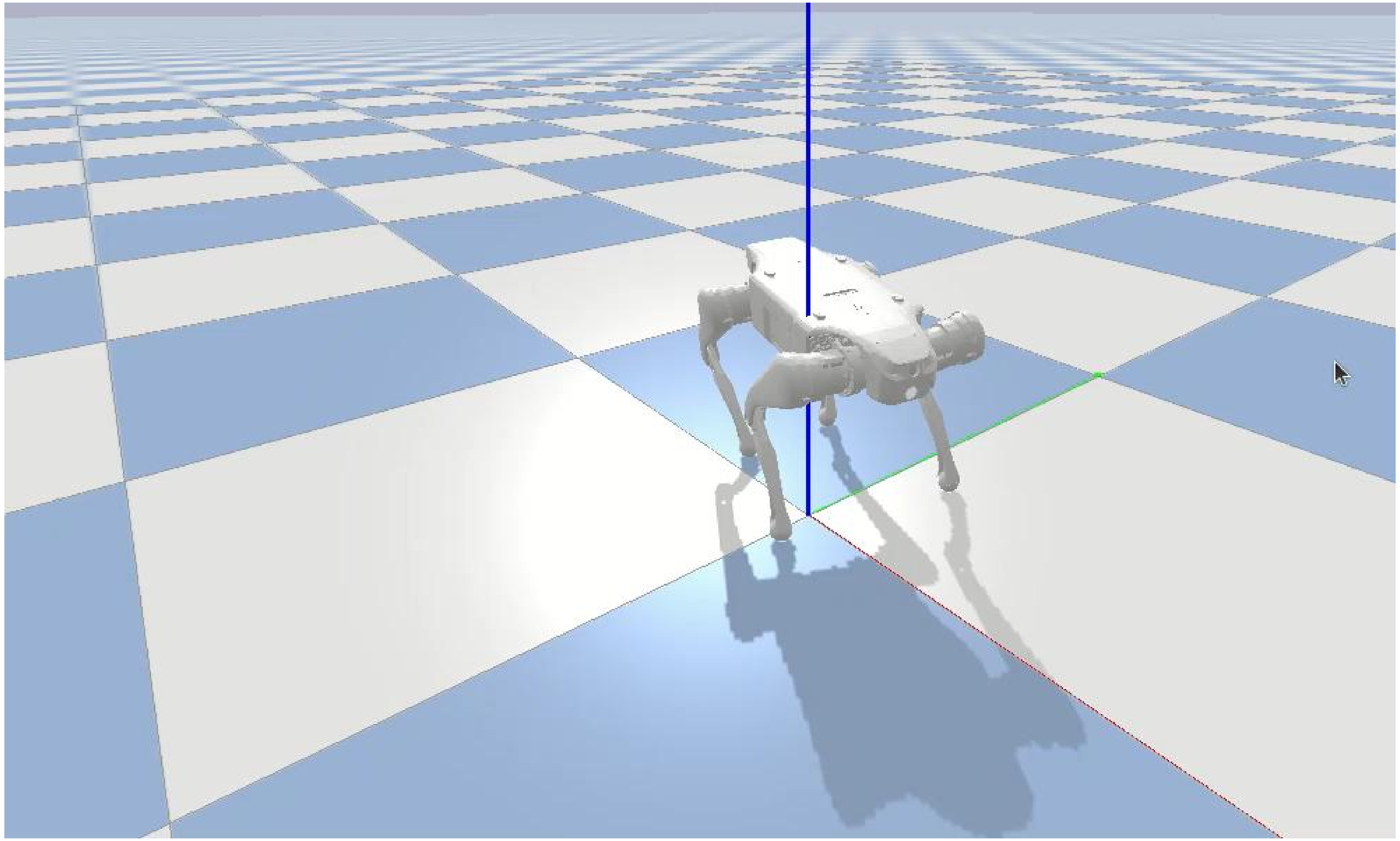
《足式机器人运动控制》第六章

动力学方程



动力学

- 仿真环境中是如何预测机器人的运动的？



目录

动力学介绍

动力学方程（Equation of Motion, EOM）

机器人动力学： 描述了当系统产生力并施加于系统时物体为什么会移动。

对于许多固定基机器人的应用，我们需要找到一个多刚体动力学模型，其公式如下：

$$M(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau + J_c(q)^T F_c$$

由以下部分组成：

$M(q) \in \mathbb{R}^{n_q \times n_q}$，广义空间的质量矩阵（对称矩阵）

$q, \dot{q}, \ddot{q} \in \mathbb{R}^{n_q}$，广义位置、速度和加速度矢量

$b(q, \dot{q}) \in \mathbb{R}^{n_q}$，科氏力和离心力项

$g(q) \in \mathbb{R}^{n_q}$，重力项

$\tau \in \mathbb{R}^{n_q}$，关节力矩向量

$F_c \in \mathbb{R}^{n_c}$，外部笛卡尔力（例如来自接触）

$J_c(q) \in \mathbb{R}^{n_c \times n_q}$，与外力相对应的几何雅可比矩阵

动力学介绍

浮动基动力学方程

浮动基动力学方程的公式如下：

$$M(q)\dot{u} + b(q,u) + g(q) = S^T\tau + J_{ext}^T F_{ext}$$

由以下部分组成：

- $M(q) \in \mathbb{R}^{n_q \times n_q}$，广义空间的质量矩阵（正交）
- $q \in \mathbb{R}^{n_q}$，广义位置
- $u \in \mathbb{R}^{n_q}$，广义速度
- $\dot{u} \in \mathbb{R}^{n_q}$，广义加速度
- $b(q,u) \in \mathbb{R}^{n_q}$，科氏力和离心力
- $g(q) \in \mathbb{R}^{n_q}$，重力项
- $S \in \mathbb{R}^{n_\tau \times n_q}$，驱动关节选择矩阵
- $\tau \in \mathbb{R}^{n_\tau}$，关节力矩向量
- $F_{ext} \in \mathbb{R}^{n_c}$，外力作用
- $J_{ext} \in \mathbb{R}^{n_c \times n_q}$，外力作用位置的（几何）雅可比矩阵

驱动关节坐标q_j和非驱动基座坐标q_b，分别对应速度$u_j = \dot{q}_j \in \mathbb{R}^{n_j}$和$u_b \in \mathbb{R}^{n_j}$。

选择矩阵S根据右侧公式选择驱动关节：$u_j = Su = S\begin{pmatrix} u_b \\ u_j \end{pmatrix} = \begin{bmatrix} 0_{6\times 6} & \mathbb{I}_{6\times n_j} \end{bmatrix}\begin{pmatrix} u_b \\ u_j \end{pmatrix}$

处理接触力（例如腿式机器人）时，使用替代符号F_c表示机器人对其环境施加的力：$M(q)\dot{u} + b(q,u) + g(q) + J_c^T F_c = S^T\tau$

动力学介绍

拉格朗日动力学方程

方程形式：

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau$$

$$L \ = T - V$$

$L(q, \dot{q})$是拉格朗日量，定义为动能减去势能

T为系统的总动能

V为系统的总势能

q是广义坐标系位置

$\dot{q}$是广义坐标系速度

τ对应的广义力

拉格朗日方程是一种基于能量的动力学方程，对于同一种机器人平台来说，各种方式计算出的动力学方程应该是一致的。

动力学介绍

拉格朗日动力学方程

方程形式：

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}-\frac{\partial L}{\partial q}=\tau$$

方程形式：对于机器人平台来说，第i根连杆的动能k_i可以表示为：

$$k_i=\frac{1}{2}m_i v_c^T v_c+\frac{1}{2}\omega_i^T I_i^C \omega_i$$

式中第一项是连杆线速度产生的动能，第二项是连杆角速度产生的动能，整个机器人系统的动能则是各个连杆的动能之和：

$$T=\sum_{i=0}^{N} k_i$$

只考虑第i个连杆的势能（只考虑重力势能）：

$$u_i=-m_i g^T p_i^0$$

总势能是各个连杆势能之和：

$$V=\sum_{i=0}^{N} u_i$$

动力学介绍

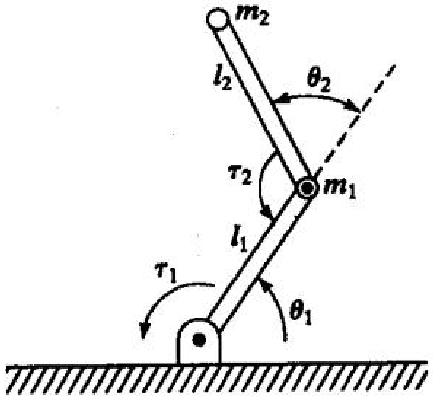
拉格朗日动力学方程

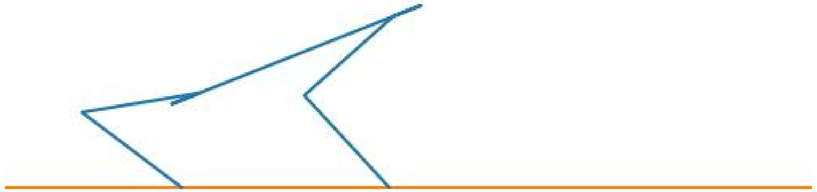
方程形式：

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial T}{\partial q} + \frac{\partial V}{\partial q} = \tau \quad \Longleftrightarrow \quad M\ddot{q} + C(q,\dot{q}) = \tau$$

适用于平面机器人或者自由度较少的情况：

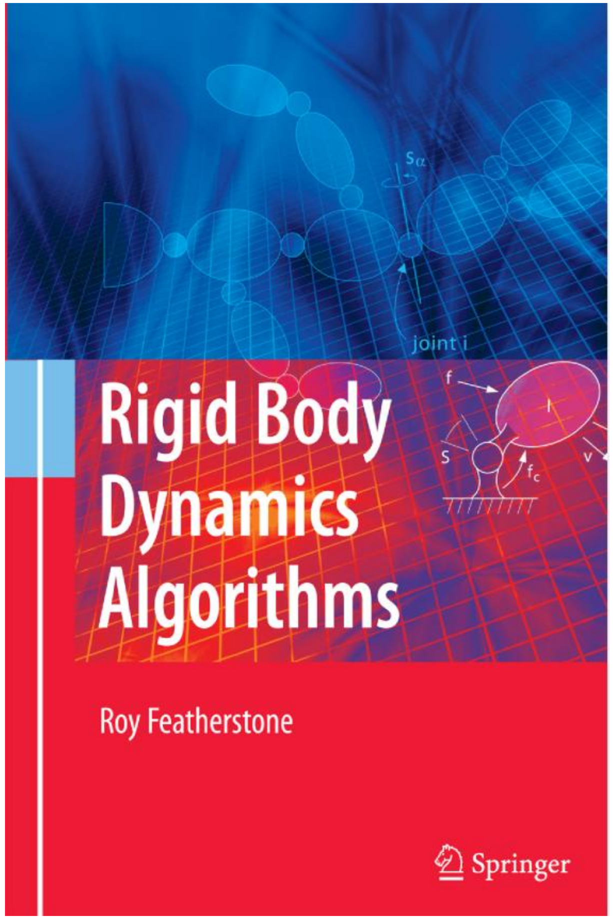




目录

Spatial Vector

Rigid Body Dynamics Algorithms



《Rigid Body Dynamics Algorithms》

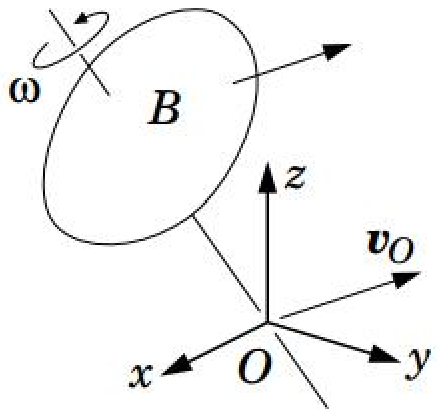
Authors: Roy Featherstone

1. A comprehensive collection of the best rigid-body dynamics algorithms
2. Use of spatial (6D) vectors to greatly reduce the volume of algebra, to simplify the treatment of the subject, and to simplify the computer code that implements the algorithms
3. Algorithms expressed both mathematically and in pseudocode for easy translation into computer programs

Spatial Vector

Spatial Velocity

概念：空间速度（spatialvelocity）在机器人学和动力学中指刚体在空间中的整体运动，包括其线速度和角速度。是用于描述刚体或物体在三维空间中瞬时运动状态的一个六维矢量。

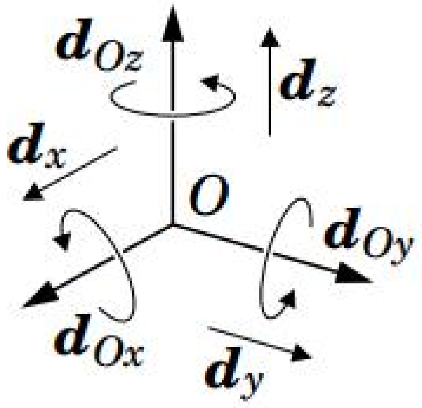


存在刚体B，在空间中的任何位置选择一个不动点O。

O，B的速度可以由一对3D向量指定：当前与O重合的物体固定点的线速度v_o和角速度向量ω。

$$\mathcal{V}_P = v_O + \omega \times \overrightarrow{OP}$$

$$\omega = \omega_x i + \omega_y j + \omega_z k, v_O = v_{O_x} i + v_{O_y} j + v_{O_z} k$$



定义一个M^6的基为：$D_O = \{d_{Ox}, d_{Oy}, d_{Oz}, d_x, d_y, d_z\} \subset M^6$

$$\hat{\mathcal{V}} = \omega_x \boldsymbol{d}_{Ox} + \omega_y \boldsymbol{d}_{Oy} + \omega_z \boldsymbol{d}_{Oz} + v_{Ox} \boldsymbol{d}_x + v_{Oy} \boldsymbol{d}_y + v_{Oz} \boldsymbol{d}_z$$

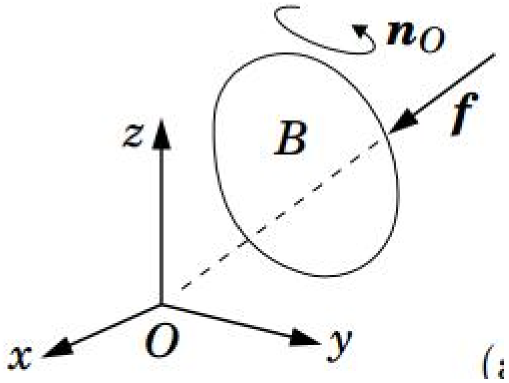
$\hat{\boldsymbol{v}}$在D_O基下的速度可以表示为：

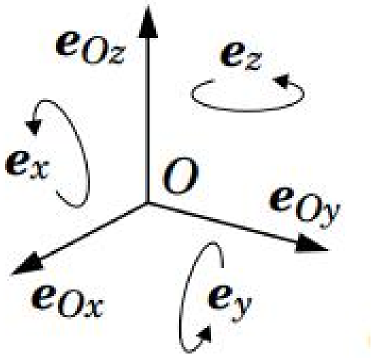
$$\underline{\hat{\mathcal{V}}}_O = \begin{bmatrix} \omega_x & \omega_y & \omega_z & v_{O_x} & v_{O_y} & v_{O_z} \end{bmatrix}^T = [\underline{\boldsymbol{\omega}} \quad \underline{\boldsymbol{v}}_O]^T$$

Spatial Vector

Spatial Force

概念：空间力（Spatial Force）是描述刚体在三维空间中受力状态的一个六维向量，包含了物体在某一参考点上的力和力矩两部分。它和空间速度类似，都是将平动和转动的分量结合起来。





存在刚体B，在空间中的任何位置选择一个不动点O。作用在刚体B上的最一般的力包括沿通过O的线作用的线性力f和力偶n_O，n_O等于对O的总矩。

则： $n_P = n_O + f \times \overrightarrow{OP}$

$$n_O = n_{Ox}i + n_{Oy}j + n_{Oz}k\ ,\ f = f_x i + f_y j + f_z k$$

定义一个F^6的基为：$\mathcal{E}_O = \{e_x, e_y, e_z, e_{Ox}, e_{Oy}, e_{Oz}\} \subset F^6$

$$\hat{\mathcal{F}} = n_{Ox}e_x + n_{Oy}e_y + n_{Oz}e_z + f_x e_{Ox} + f_y e_{Oy} + f_z e_{Oz}$$

$\hat{f}$在$\mathcal{E}_O$基下的速度可以表示为：

$$\underline{\hat{\mathcal{F}}}_O = \begin{bmatrix} n_{O_x} & n_{O_y} & n_{O_z} & f_x & f_y & f_z \end{bmatrix}^T = [\underline{n}_O \quad \underline{f}]^T$$

Spatial Vector

一些运算

X表示motion vector的坐标系转换，X^*表示force vector的坐标系转换：

$$X^* = X^{-T}$$

$\underline{m} \in M^6, \underline{f} \in F^6$, 对于所有的$\underline{m}$，$\underline{f}$需要满足：

$$\underline{m}^T \underline{f} = (X\underline{m})^T (X^* \underline{f})$$

motion vector的转换：

$$^{B}\boldsymbol{X}_A = \underbrace{\begin{bmatrix} \boldsymbol{E} & 0 \\ 0 & \boldsymbol{E} \end{bmatrix}}_{\text{Rotation}} \underbrace{\begin{bmatrix} 1 & 0 \\ -\boldsymbol{r}\times & 1 \end{bmatrix}}_{\text{Translation}} = \begin{bmatrix} \boldsymbol{E} & 0 \\ -\boldsymbol{E}\boldsymbol{r}\times & \boldsymbol{E} \end{bmatrix}$$

force vector的转换：

$$^{B}\boldsymbol{X}_A^* = \begin{bmatrix} \boldsymbol{E} & 0 \\ 0 & \boldsymbol{E} \end{bmatrix} \begin{bmatrix} 1 & -\boldsymbol{r}\times \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{E} & -\boldsymbol{E}\boldsymbol{r}\times \\ 0 & \boldsymbol{E} \end{bmatrix}$$

Spatial Vector

一些运算

$r \in E^3$ $$r \xRightarrow{[\omega \times]} \dot{r}$$

在motion vector和force vector中，$\hat{m} \in M^6, \hat{f} \in F^6$：

$$\dot{\hat{m}} = \hat{v} \times \hat{m}$$

$$\dot{\hat{f}} = \hat{v} \times^* \hat{f}$$

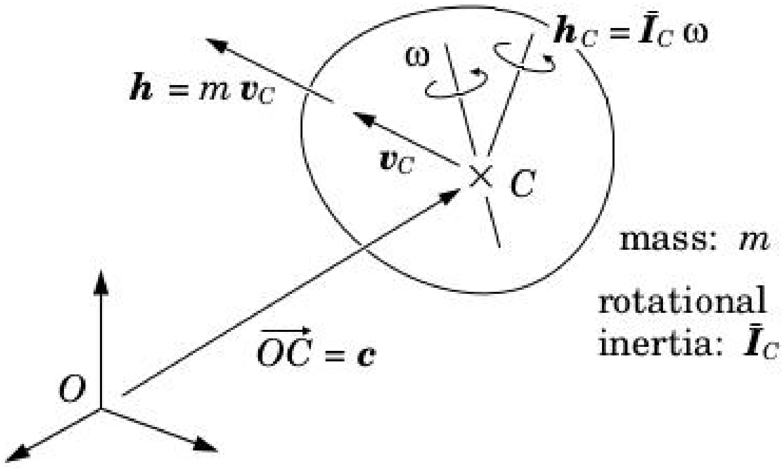
spatial cross products:

$$\hat{v}_O \times = \begin{bmatrix} \omega \\ v_O \end{bmatrix} \times = \begin{bmatrix} \omega \times & 0 \\ v_O \times & \omega \times \end{bmatrix}$$

$$\hat{\boldsymbol{v}}_O \times^* = \begin{bmatrix} \omega \\ \boldsymbol{v}_O \end{bmatrix} \times^* = \begin{bmatrix} \omega \times & \boldsymbol{v}_O \times \\ 0 & \boldsymbol{\omega} \times \end{bmatrix} = -(\hat{\boldsymbol{v}}_O \times)^T$$

Spatial Vector

一些运算



spatial inertia tensor:

$$I_C = \begin{bmatrix} \bar{I}_C & 0 \\ 0 & m1 \end{bmatrix}$$

$$\boldsymbol{h}_O = \begin{bmatrix} 1 & \boldsymbol{c}\times \\ 0 & 1 \end{bmatrix} \boldsymbol{I}_C \boldsymbol{v}_C$$

$$= \begin{bmatrix} 1 & \boldsymbol{c}\times \\ 0 & 1 \end{bmatrix} \boldsymbol{I}_C \begin{bmatrix} 1 & 0 \\ \boldsymbol{c}\times^{\mathrm{T}} & 1 \end{bmatrix} \boldsymbol{v}_O$$

$$\boldsymbol{I}_O = \begin{bmatrix} \bar{I}_C + m\boldsymbol{c}\times\boldsymbol{c}\times\times^{T} & m\boldsymbol{c}\times \\ m\boldsymbol{c}\times T & m1 \end{bmatrix}$$

目录

动力学方程计算

逆向动力学

逆动力学是在刚体系统中寻找产生给定加速度所需力的问题。用于在给定运动轨迹（位置、速度和加速度）条件下，计算为实现该运动而需要施加的力矩或力。

动力学方程的形式表示：

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \; - J_{ext}^{T}(q)F_{ext}$$

$$\tau = ID(model,\ q,\ \dot{q},\ \ddot{q},\ F_{ext})$$

正向动力学

正向动力学是寻找刚体系统对给定施加力的加速度的问题。用于在已知关节力矩（或驱动力）及环境作用力的情况下，计算机器人系统的加速度、速度和位置变化。

动力学方程的形式表示：

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \;=\; \tau + J_{ext}^{T}(q)F_{ext}$$

$$C(q,\dot{q})\dot{q} + G(q) = ID(model,\ q,\ \dot{q},\ 0,\ F_{ext})$$

$$\ddot{q} = M^{-1}(\tau + J_{ext}^{T}(q)F_{ext} - ID(model,\ q,\ \dot{q},\ 0,\ F_{ext}))$$

动力学方程计算

正向运动学

$$^{0}v_0 = 0$$

for $i = 1$ **to** N_B **do**

$$[X_J, v_J] = \text{jcalc}(\text{jtype}(i), q_i, \dot{q}_i)$$

$$^{i}X_{\lambda(i)} = X_J X_T(i)$$

if $\lambda(i) \neq 0$ **then**

$$^{i}X_0 = {}^{i}X_{\lambda(i)}{}^{\lambda(i)}X_0$$

end

$$^{0}v_i = {}^{0}v_{\lambda(i)} + {}^{0}X_i v_J$$

end

```python
def jointCalc(self, jointType, q):
    Xj = ca.DM.eye(6)
    S = ca.DM.zeros(6, 1)
    if jointType == 'RotX':
        Xj = self.rot(q, self.rotX)
        S[0] = 1
    elif jointType == 'RotY':
        Xj = self.rot(q, self.rotY)
        S[1] = 1
    elif jointType == 'RotZ':
        Xj = self.rot(q, self.rotZ)
        S[2] = 1
    else:
        print("Not revolute joint")
    return Xj, S
```

```python
def rot(self, theta, func):
    zero = ca.DM.zeros(3, 3)
    X = ca.vertcat(ca.horzcat(func(theta).T, zero),
                   ca.horzcat(zero, func(theta).T))
    return X
```

可以求得各个关节坐标系的转换关系

动力学方程计算

浮动基动力学计算

$$\begin{bmatrix} I_0^c & F \\ F^{\mathrm{T}} & H \end{bmatrix} \begin{bmatrix} a_0 \\ \ddot{q} \end{bmatrix} + \begin{bmatrix} p_0^c \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ \tau \end{bmatrix}$$

对于旋转关节：

$$v_{\mathrm{J}} = S(q)\dot{q}$$

$$c_J = 0$$

Calculate C and p_0^c:

$a_0^{vp} = -\,{}^0a_g$
for $i = 1$ to N_B do
  $[X_J, S_i, v_J, c_J] = \text{jcale}(\text{jtype}(i), q_i, \dot{q}_i)$
  ${}^iX_{\lambda(i)} = X_J X_T(i)$
  if $\lambda(i) \neq 0$ then
    ${}^iX_0 = {}^iX_{\lambda(i)}\,{}^{\lambda(i)}X_0$
  end
  $v_i = {}^iX_{\lambda(i)} v_{\lambda(i)} + v_J$
  $a_i^{vp} = {}^iX_{\lambda(i)} a_{\lambda(i)}^{vp} + c_J + v_i \times v_J$
  $f_i I_i a_i^{vp} + v_i \times{*}\, I_i v_i - {}^0f_i^x$
end
$f_0 = I_0 a_0^{vp} + v_0 \times{*}\, I_0 v_0 - {}^0f_0^x$
for $i = N_B$ to 1 do
  $C_i = S_i^T f_i$
  $f_{\lambda(i)} = f_{\lambda(i)} + {}^{\lambda(i)}X_i^* f_i$
end
$p_0^c = f_0$

Recursive Newton-Euler algorithm

Calculate H, F and I_0^c:

$H = 0$
for $i = 1$ to N_B do
  $I_i^c = I_i$
end
for $i = N_B$ to 1 do
  $I_{\lambda(i)}^c = I_{\lambda(i)}^c + {}^{\lambda(i)}X_i^* I_i^c\,{}^iX_{\lambda(i)}$
  $F_i = I_i^c S_i$
  $H_{ii} = S_i^T F_i$
  $j = i$
  while $\lambda(j) \neq 0$ do
    $F_i = {}^{\lambda(j)}X_j^* F_i$
    $j = \lambda(j)$
    $H_{ij} = F_i^T S_j$
    $H_{ji} = H_{ij}^T$
  end
  $F_i = {}^0X_j^* F_i$
end

Composite-rigid-body algorithm

动力学方程计算

开源动力学方程库

- Pinocchio

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

  支持C++和python，应用比较广泛

- RBDL

  https://github.com/rbdl/rbdl

  直接读取URDF文件，代码量比较少，现在也支持python调用

- Frost

  https://github.com/ayonga/frost-dev

- drake

深蓝学院
shenlanxueyuan.com

感谢聆听！

ThanksforListening