机器人逆动力学（Robot Inverse Dynamics）

任 程

逆动力学问题是指：已知某一时刻机器人各关节的位置 q，关节速度 $\dot{q}$ 及关节加速度 $\ddot{q}$，求此时施加在机器人各杆件上的驱动力（力矩）$\boldsymbol{\tau}$。

逆动力学问题在机器人控制与计算机动画领域都有广泛的应用。例如当给出期望的机器人运动状态时，我们可以通过逆动力学解算来分析其力矩是否可以由作动系统实现。在计算机动画领域，可以利用优化算法求解力矩消耗最小的动画过程（如文献[1]）来得到一个自然的动画。另外，逆动力学也常作为正动力学的一个子部分来求解正动力学（正动力学指已知力和力矩，求系统状态）。

逆动力学可以利用牛顿欧拉(Newton-Euler)方程来求解，也可以利用拉格朗日(Lagrange)方程来求解（二者的等价性与区别读者可以参看文献[2]中的 2.3 节）。本文旨在讲解如何基于牛顿欧拉(Newton-Euler)方程来求解机器人逆动力学，其算法被称为“迭代牛顿欧拉算法(Recursive Newton-Euler Algorithm)”。

1. 预备知识

在介绍“迭代牛顿欧拉算法(Recursive Newton-Euler Algorithm)”之前，让我们先看一下什么是牛顿欧拉方程：

$$\begin{aligned}\sum \mathbf{f}(t) &= m\mathbf{a}(t) \\ \sum \boldsymbol{\tau}(t) &= \mathrm{I}\,\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \mathrm{I}\,\boldsymbol{\omega}(t)\end{aligned} \tag{1.1}$$

其中 $\mathbf{a}(t)$ 表示线加速度，$\dot{\boldsymbol{\omega}}(t)$ 表示角加速度（角速度的导数），等式左边的求和符号表示公式中应该使用合力与合力矩。关于如何得出牛顿欧拉方程，请参看我的前一篇文章：《刚体动力学》
http://www.cnblogs.com/ArenAK/archive/2010/06/07/1753427.html

2. 迭代牛顿欧拉算法

下面我们说明什么是“迭代牛顿欧拉算法(Recursive Newton-Euler Algorithm)”，即它如何从系统状态求解出力矩，之后在下一节本文将说明为什么要进行“迭代”计算，从而导出此算法的复杂度。

迭代牛顿欧拉算法包含两个迭代过程：

（1）**第一个迭代过程**由机器人运动系统的根节点开始，逐步向各个叶子节点进行迭代。在此迭代过程中各个杆件的运动速度（线速度 $\mathbf{v}$、角速度 $\boldsymbol{\omega}$）、运动加速度（线加速度 $\dot{\mathbf{v}}$、角加速度 $\dot{\boldsymbol{\omega}}$）被依次求出，从父节点向子节点迭代的直观原因是父节点会带动子节点运动，因此子节点的速度/加速度要在父节点的速度/加速度求出后迭代求出：

$$\boldsymbol{\omega}_i = {}^{i}\mathbf{E}_{i-1}\boldsymbol{\omega}_{i-1} + \mathbf{z}\dot{q}_i \tag{1.2}$$

$$\mathbf{v}_i = {}^{i}\mathbf{E}_{i-1}(\mathbf{v}_{i-1} + \boldsymbol{\omega}_{i-1} \times {}^{i-1}\mathbf{r}_i) \tag{1.3}$$

$$\dot{\boldsymbol{\omega}}_i = {}^{i}\mathbf{E}_{i-1}\dot{\boldsymbol{\omega}}_{i-1} + \mathbf{z}\ddot{q}_i + ({}^{i}\mathbf{E}_{i-1}\boldsymbol{\omega}_{i-1}) \times \mathbf{z}\dot{q}_i \tag{1.4}$$

$$\dot{\mathbf{v}}_i = {}^{i}\mathbf{E}_{i-1}(\dot{\mathbf{v}}_{i-1} + \dot{\boldsymbol{\omega}}_{i-1} \times {}^{i-1}\mathbf{r}_i + \boldsymbol{\omega}_{i-1} \times \boldsymbol{\omega}_{i-1} \times {}^{i-1}\mathbf{r}_i) \tag{1.5}$$

其中速度（线速度 $\mathbf{v}_i$、角速度 $\boldsymbol{\omega}_i$）和加速度（线加速度 $\dot{\mathbf{v}}_i$、角加速度 $\dot{\boldsymbol{\omega}}_i$）都表示在杆件 i 的局部坐标系 F_i 中；

线速度 $\dot{\mathbf{v}}_i$ 和线加速度 $\dot{\boldsymbol{\omega}}_i$ 为杆件 i 局部坐标系原点的运动速度（表示在局部坐标系 F_i 中）；

$\mathbf{z}$ 为杆件 i 的运动轴（平移轴或旋转轴）在坐标系 F_i 中的方向；

$\dot{q}_i$，$\ddot{q}_i$ 为测量得到的杆件 i 旋转的速度和加速度数值（是标量）；

${}^{i}\mathbf{E}_{i-1}$ 为由坐标系 F_{i-1} 到坐标系 F_i 的旋转矩阵；

${}^{i-1}\mathbf{r}_i$ 为坐标系 F_i 的原点在坐标系 F_{i-1} 中的位置。

同时由于各个杆件所受的合力、合力矩只与加速度有关，因此在此迭代过程中各杆件所受的合力、合力矩也可以被求出：

$$\mathbf{F}_i = m_i(\dot{\mathbf{v}}_i + \dot{\boldsymbol{\omega}}_i \times \mathbf{c}_i + \boldsymbol{\omega}_i \times \boldsymbol{\omega}_i \times \mathbf{c}_i) \tag{1.6}$$

$$\mathbf{N}_i = \mathbf{I}_i^{cm}\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \mathbf{I}_i^{cm}\boldsymbol{\omega}_i \tag{1.7}$$

其中 $\mathbf{F}_i$、$\mathbf{N}_i$ 分别为杆件 i 的质心所受到合力、合力矩。

m_i 为杆件 i 的质量，$\mathbf{c}_i$ 为杆件 i 的质心在坐标系 F_i 中的坐标，$\mathbf{I}_i^{cm}$ 为坐标系 F_i 中以质心为参考点杆件 i 的惯量阵。

（2）**第二个迭代过程**的方向由叶子节点开始，逐步向根节点进行迭代，目的是求出各个杆件之间的相互作用力。每个杆件受到的力（力矩）包括父杆件作用于本杆件的力（力矩），子杆件作用于本杆件的力（力矩），以及重力。对于最末端

的叶子杆件，不存在子杆件的作用力（即下式中$\mathbf{f}_{n+1}=\mathbf{0}$， $\mathbf{n}_{n+1}=\mathbf{0}$），因此可以从子杆件开始算起，逐步推导出各个杆件之间的相互作用力（力矩）。

$$\mathbf{f}_i=\mathbf{F}_i+{}^{i}\mathbf{E}_{i+1}\mathbf{f}_{i+1} \tag{1.8}$$

$$\mathbf{n}_i=\mathbf{N}_i+c_i\times\mathbf{F}_i+{}^{i}\mathbf{E}_{i+1}\mathbf{n}_{i+1}+{}^{i}r_{i+1}\times{}^{i}\mathbf{E}_{i+1}\mathbf{f}_{i+1} \tag{1.9}$$

其中$\mathbf{f}_i$、$\mathbf{n}_i$分别为杆件i经由关节i而受到的力、力矩。

最后可求出杆件在某个轴向上所受到的力矩的大小（标量）：

$$\tau_i=\mathbf{z}^T\mathbf{n}_i \tag{1.10}$$

3. 迭代计算的优越性

现在我们已经知道了如何利用迭代牛顿欧拉算法进行计算，我们再来看看为何要进行“迭代”计算。实际上，不进行迭代也可以计算出结果，然而不进行迭代将会存在过多的冗余计算，大大影响计算效率。我们将以速度的计算为例来说明，为了叙述简单，我们设各个关节只有一个自由度，各个速度量表示在一个公共坐标系（而不是各个杆件的局部坐标系）下。因此利用递归方法杆件i的速度表达式为：

$$\mathbf{v}_i=\mathbf{v}_{i-1}+\boldsymbol{\omega}_{i-1}{}^{i-1}r_i \tag{1.11}$$

设根节点处的速度为$v_0=0$。非递归方法杆件i的速度表达式为：

$$\mathbf{v}_i=\sum_{j=1}^{i}\boldsymbol{\omega}_{j-1}{}^{j-1}r_j \tag{1.12}$$

设m代表一个标量与向量相乘的计算代价，a代表两个向量进行相加的计算代价。那么调用一次式(1.11)的计算代价为$m+a$，调用一次式(1.12)的计算代价为$im+(i-1)a$，则利用式(1.11)计算机器人前n个杆件的计算代价为$n(m+a)$，而利用式(1.12)进行相应计算的代价为$\frac{1}{2}(n(n+1)m+n(n-1)a)$。因此，利用递归计算式(1.11)获得了复杂度为$O(n)$的高效率计算，而利用非递归计算式(1.12)则获得复杂度为$O(n^2)$的低效率计算。

非递归式计算效率低的原因在于进行了过多的冗余计算：

$$
\begin{aligned}
\mathbf{v}_1 &= \boldsymbol{\omega}_0{}^0 r_1 \\
\mathbf{v}_2 &= \boldsymbol{\omega}_0{}^0 r_1 + \boldsymbol{\omega}_1{}^1 r_2 \\
&\vdots \\
\mathbf{v}_n &= \boldsymbol{\omega}_0{}^0 r_1 + \boldsymbol{\omega}_1{}^1 r_2 + \cdots + \boldsymbol{\omega}_{n-1}{}^{n-1} r_n
\end{aligned}
$$

可见$\boldsymbol{\omega}_0{}^0 r_1$这一项被重复计算了$n$次，$\boldsymbol{\omega}_1{}^1 r_2$被计算了$n-1$次，等等。递归算法避免了此类冗余计算，因而获得了高效率。

当公式越复杂时，递归与非递归获得的效率差别更大。例如利用递归来计算加速度，则复杂度由$O(n^3)$降为$O(n)$，而力矩的计算则更是从复杂度$O(n^4)$降为$O(n)$。因此迭代牛顿欧拉算法实现了复杂度为$O(n)$的逆动力学计算。

说明：本文 2、3 节内容大多参考了文献[3]。

[1] C. Rose*, et al.*, "Efficient generation of motion transitions using spacetime constraints," presented at the Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[2] 霍伟, *机器人动力学与控制*: 高等教育出版社, 2005.

[3] R. Featherstone, *Rigid Body Dynamics Algorithms*: Springer-Verlag New York, Inc., 2007.