Appendix 10

Computation of the dynamic control law in the task space

In this appendix, we present the computation of the decoupling nonlinear control in the task space [Khalil 87a]. The dynamic model is computed by a specialized Newton-Euler algorithm, which takes into account many variables that are evaluated for the kinematic models.

The number of operations of this control law for the Stäubli RX-90 robot, assuming symmetrical links whose inertial parameters are given in Table 9.4, is 316 multiplications and 237 additions.

A10.1. Calculation of the location error $\mathbf{e_x}$

The current location of the terminal link is given by the homogeneous transformation matrix ${}^0\mathbf{T}_n$, which can be obtained using a customized symbolic algorithm (Chapter 3):

$$ {}^0\mathbf{T}_n = {}^0\mathbf{T}_1 \, {}^1\mathbf{T}_2 \ldots {}^{n-1}\mathbf{T}_n = \begin{bmatrix} {}^0\mathbf{s}_n & {}^0\mathbf{n}_n & {}^0\mathbf{a}_n & {}^0\mathbf{P}_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [A10.1] $$

Let the desired position and orientation be given by ${}^0\mathbf{P}_n^d$ and $[{}^0\mathbf{s}_n^d \; {}^0\mathbf{n}_n^d \; {}^0\mathbf{a}_n^d]$ respectively. The location error, denoted by $\mathbf{e_x}$, is given by:

$$ \mathbf{e_x} = [\; \mathbf{dX}_p^T \;\; \mathbf{dX}_r^T \;]^T \qquad [A10.2] $$

where $\mathbf{dX}_p$ and $\mathbf{dX}_r$ indicate the position error and orientation error respectively.

The position error is obtained by:

$$\mathbf{dX}_p = \mathbf{dP} = {}^0\mathbf{P}_n^d - {}^0\mathbf{P}_n \tag{A10.3}$$

The orientation error is given by [Luh 80a]:

$$\mathbf{dX}_r = {}^0\mathbf{u}\,\alpha \tag{A10.4}$$

where ${}^0\mathbf{u}$ and α are obtained by solving $\mathbf{rot}(\mathbf{u},\alpha)\,{}^0\mathbf{A}_n = {}^0\mathbf{A}_n^d$.

Let us assume that:

$${}^0\mathbf{A}_n^d\,{}^0\mathbf{A}_n^T = [\,\mathbf{s}\ \ \mathbf{n}\ \ \mathbf{a}\,] \tag{A10.5}$$

If α is small, the orientation error $\mathbf{dX}_r$ can be considered to be equal to $\mathbf{u}\sin(\alpha)$ or equal to ${}^0\boldsymbol{\delta}_n$ (§ 2.5). Using equation [2.35], we obtain:

$$\mathbf{dX}_r = \mathbf{u}\sin(\alpha) = \frac{1}{2}\begin{bmatrix} n_z - a_y \\ a_x - s_z \\ s_y - n_x \end{bmatrix} \tag{A10.6}$$

Using equation [5.59], we obtain:

$$\mathbf{dX}_r = {}^0\boldsymbol{\delta}_n = \boldsymbol{\Omega}_{CD}^{+}\begin{bmatrix} {}^0\mathbf{s}_n^d - {}^0\mathbf{s}_n \\ {}^0\mathbf{n}_n^d - {}^0\mathbf{n}_n \\ {}^0\mathbf{a}_n^d - {}^0\mathbf{a}_n \end{bmatrix} \tag{A10.7}$$

$$\mathbf{dX}_r = \frac{1}{2}[{}^0\mathbf{s}_n \times {}^0\mathbf{s}_n^d + {}^0\mathbf{n}_n \times {}^0\mathbf{n}_n^d + {}^0\mathbf{a}_n \times {}^0\mathbf{a}_n^d] \tag{A10.8}$$

A10.2. Calculation of the velocity of the terminal link $\dot{\mathbf{X}}$

The terminal link velocity is composed of the linear velocity ${}^0\mathbf{V}_n$ and of the angular velocity ${}^0\boldsymbol{\omega}_n$:

$$\dot{\mathbf{X}} = \begin{bmatrix} {}^0\mathbf{V}_n \\ {}^0\boldsymbol{\omega}_n \end{bmatrix} \tag{A10.9}$$

${}^{0}V_n$ and ${}^{0}\omega_n$ are calculated using the following recursive equations, which are developed in Chapter 9:

For j = 1, ..., n:

$$ {}^{j}V_j = {}^{j}A_{j-1}({}^{j-1}V_{j-1} + {}^{j-1}\omega_{j-1} \times {}^{j-1}P_j) + \sigma_j \dot{q}_j\, {}^{j}a_j \quad [A10.10]$$

$$ {}^{j}\omega_{j-1} = {}^{j}A_{j-1}\, {}^{j-1}\omega_{j-1} \quad [A10.11]$$

$$ {}^{j}\omega_j = {}^{j}\omega_{j-1} + \bar{\sigma}_j \dot{q}_j\, {}^{j}a_j \quad [A10.12]$$

The vectors ${}^{0}V_n$ and ${}^{0}\omega_n$ are obtained by ${}^{0}V_n = {}^{0}A_n\, {}^{n}V_n$ and ${}^{0}\omega_n = {}^{0}A_n\, {}^{n}\omega_n$. We note that ${}^{1}\omega_1, \ldots, {}^{n}\omega_n$ are also required for the inverse dynamic algorithm.

A10.3. Calculation of $\dot{J}\,\dot{q}$

The calculation of this vector by differentiating the Jacobian matrix with respect to time and the multiplication of the result by $\dot{q}$ would need a prohibitive number of operations. We propose to use an efficient recursive algorithm derived from the second order kinematic model. Many intermediate variables of this algorithm are used for the computation of the inverse dynamic model. The second order kinematic model is given by:

$$ \ddot{X}_n = \begin{bmatrix} \dot{V}_n \\ \dot{\omega}_n \end{bmatrix} = J(q)\,\ddot{q} + \dot{J}(q, \dot{q})\,\dot{q} \quad [A10.13]$$

From equation [A10.13], we deduce that $\dot{J}(q, \dot{q})\,\dot{q}$ is equal to $\ddot{X}_n$ when setting $\ddot{q} = \mathbf{0}$.

$$ \dot{J}(q, \dot{q})\,\dot{q} = \begin{bmatrix} \dot{V}_n(\ddot{q} = 0) \\ \dot{\omega}_n(\ddot{q} = 0) \end{bmatrix} = \begin{bmatrix} \phi_n \\ \Psi_n \end{bmatrix} \quad [A10.14]$$

Consequently, $\dot{J}\,\dot{q}$ can be computed using equations [9.86], [9.87] and [9.90] after setting $\ddot{q} = \mathbf{0}$. The algorithm is given as follows:

For j = 1, ..., n:

$$ {}^{j}\Psi_j = {}^{j}A_{j-1}\, {}^{j-1}\Psi_{j-1} + \bar{\sigma}_j ({}^{j}\omega_{j-1} \times \dot{q}_j\, {}^{j}a_j) \quad [A10.15]$$

$$ {}^{j}U_j^{*} = {}^{j}\hat{\Psi}_j + {}^{j}\hat{\omega}_j\, {}^{j}\hat{\omega}_j \quad [A10.16]$$

$$ {}^{j}\phi_j = {}^{j}A_{j-1}({}^{j-1}\phi_{j-1} + {}^{j-1}U_{j-1}^{*}\, {}^{j-1}P_j) + 2\sigma_j\, {}^{j}\omega_{j-1} \times (\dot{q}_j\, {}^{j}a_j) \quad [A10.17]$$

The initial values are: ${}^{0}\Psi_0 = \mathbf{0}$ and ${}^{0}\phi_0 = \mathbf{0}$.

A10.4. Calculation of $\mathbf{J}(\mathbf{q})^{-1}\,\mathbf{y}$

The vector y indicates the term $(\mathbf{w}(t) - \dot{\mathbf{J}}\,\dot{\mathbf{q}})$ as given in equation [14.33]. This problem is treated in Chapter 6. The solution for the regular case of the Stäubli RX-90 robot is developed in Example 6.1.

A10.5. Modified dynamic model

We modify the inverse dynamic model developed in Chapter 9 to take into account the availability of $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$. Equations [9.86] and [9.87], giving $\dot{\omega}_j$ and $\dot{\mathbf{V}}_j$, are replaced by:

$$^{j}\dot{\omega}_j = {}^{j}\mathbf{\Psi}_j + {}^{j}\boldsymbol{\varepsilon}_j \qquad [A10.18]$$

$$^{j}\dot{\mathbf{V}}_j = {}^{j}\boldsymbol{\phi}_j + {}^{j}\mathbf{b}_j \qquad [A10.19]$$

where $^{j}\boldsymbol{\varepsilon}_j$ and $^{j}\mathbf{b}_j$ represent $^{j}\dot{\omega}_j$ and $^{j}\dot{\mathbf{V}}_j$ respectively, when $\dot{\mathbf{q}} = \mathbf{0}$:

$$^{j}\boldsymbol{\varepsilon}_j = {}^{j}\mathbf{A}_{j-1}\,{}^{j-1}\boldsymbol{\varepsilon}_{j-1} + \bar{\sigma}_j\,\ddot{q}_j\,{}^{j}\mathbf{a}_j \qquad [A10.20]$$

$$^{j}\mathbf{b}_j = {}^{j}\mathbf{A}_{j-1}({}^{j-1}\mathbf{b}_{j-1} + {}^{j-1}\hat{\boldsymbol{\varepsilon}}_{j-1}\,{}^{j-1}\mathbf{P}_j) + \sigma_j\,\ddot{q}_j\,{}^{j}\mathbf{a}_j \qquad [A10.21]$$

with the initial values $^{0}\boldsymbol{\varepsilon}_0 = \mathbf{0}$ and $^{0}\mathbf{b}_0 = -\mathbf{g}$.

The matrix $^{j}\mathbf{U}_j$, defined in equation [9.90], is computed using $^{j}\mathbf{U}_j^{*}$ (equation [A10.16]) and $^{j}\boldsymbol{\varepsilon}_j$ (equation [A10.20]) such that:

$$^{j}\mathbf{U}_j = {}^{j}\mathbf{U}_j^{*} + {}^{j}\hat{\boldsymbol{\varepsilon}}_j \qquad [A10.22]$$

Taking into account that the angular velocities have been evaluated while computing $\dot{\mathbf{X}}$, the modified dynamic model needs 110 multiplications and 82 additions instead of 160 multiplications and 113 additions.