Advanced Robot Control

Input-output decoupling

Wojciech Domski

Chair of Cybernetics and Robotics,
Wrocław University of Science and Technology

Presentation compiled for taking notes during lecture

Wrocław University
of Science and Technology

Outline

Wrocław University of Science and Technology

Object

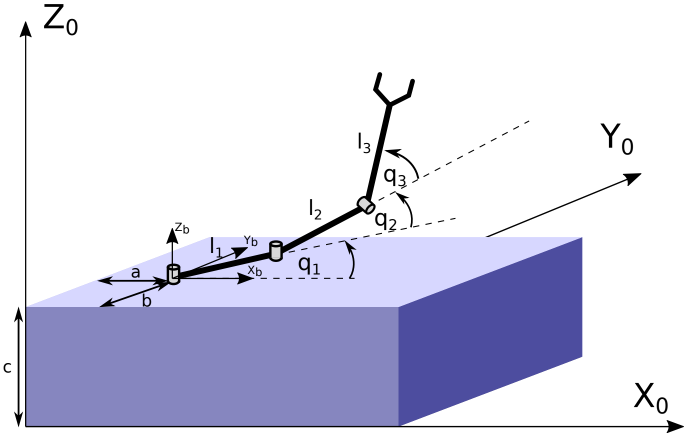

Figure: 3R rigid manipulator

Wrocław University
of Science and Technology

$$q = [q_1, q_2, q_3]^T, \quad q \in R^n, \quad n = 3. \tag{1}$$

Control engineering and robotics

$$\dot{x} = Ax + Bu \qquad\qquad y = C^T x$$

$$\text{input} \xrightarrow{\text{differential equations}} \text{state} \xrightarrow{\text{algebraic equation}} \text{output}$$

Figure: Control engineering

$$M\ddot{q} + C\dot{q} + D + T = u \qquad\qquad y = k(q)$$

$$\text{input} \xrightarrow{\text{dynamics}} \text{state} \xrightarrow{\text{kinematics}} \text{output}$$

Figure: Robotics

Kinematics (1/2)

A 3R robotic arm can follow a trajectory in a 3D space associated with the global coordination system $X_0 Y_0 Z_0$ marked in Fig. 1. Placement of the manipulator mount point is in $X_b Y_b Z_b$.
Transformations from $X_0 Y_0 Z_0$ to $X_b Y_b Z_b$ is defined as

$$A_0^b = Trans(X, a) Trans(Y, b) Trans(Z, c). \tag{2}$$

Kinematics (2/2)

End of the 1st manipulator link is in relation to global coordination system $X_0 Y_0 Z_0$ expressed as

$$A_0^1 = A_0^b Rot(Z, q_1) Trans(X, l_1). \tag{3}$$

The 2nd link location is described as

$$A_0^2 = A_0^1 Rot(Z, q_2) Trans(X, l_2) Rot(X, \frac{\pi}{2}). \tag{4}$$

The position of 3rd link, the end effector, is presented in the following relation

$$A_0^3 = A_0^2 Rot(Z, q_3) Trans(X, l_3). \tag{5}$$

Dynamics (1/7)

The manipulator dynamics can be expressed as [5]

$$M\ddot{q} + C\dot{q} + D + T = u \tag{6}$$

where

- $M \in R^{n \times n}$ is an inertia matrix,
- $C \in R^{n \times n}$ is a Coriolis and centrifugal forces matrix,
- $D \in R^{n}$ is a gravitation vector,
- $T \in R^{n}$ is a friction vector,
- $u \in R^{n}$ is an input control vector.

Wrocław University of Science and Technology

Dynamics (2/7)

Properties of the dynamics model [2]:

1. The M matrix is symmetric ($M = M^T$) and is positively defined ($M > 0$). It means that all eigenvalues of M are positive. Thus, the M matrix is invertible and not singular.
2. There is a skew symmetry between M and C.

$$\dot{M} = C + C^T \tag{7}$$

Dynamics (3/7)

To calculate inertia matrix M we have to calculate kinetic energy for each link. It can be calculated with following formula

$$E_i = \frac{1}{2} tr\{\dot{A}_0^i J_i (\dot{A}_0^i)^T\} = \frac{1}{2} \dot{q}^T Q_i \dot{q}. \tag{8}$$

The J_i is a pseudoinertia matrix of ith link and the Q_i is inertia matrix for ith link.
J_i can be calculated in following way

$$J_i = \begin{bmatrix} \int_{L_i} x^2 dm & \int_{L_i} xydm & \int_{L_i} xzdm & m_i\bar{x}_i \\ \int_{L_i} yxdm & \int_{L_i} y^2 dm & \int_{L_i} yzdm & m_i\bar{y}_i \\ \int_{L_i} zxdm & \int_{L_i} zydm & \int_{L_i} z^2 dm & m_i\bar{z}_i \\ m_i\bar{x}_i & m_i\bar{y}_i & m_i\bar{z}_i & m_i \end{bmatrix}. \tag{9}$$

Wrocław University of Science and Technology

Dynamics (4/7)

The integrals are the inertia moments calculated at the end of the link while $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is the placement of the center of mass of the link in local coordinate system.

Thus, the inertia matrix M of the system is a sum of inertia matrices of each link

$$M = \sum_{i=1}^{n} Q_i. \tag{10}$$

Wrocław University of Science and Technology

Dynamics (5/7)

The Coriolis matrix can be calculated from the inertia matrix M by using the Christoffel symbols of first kind. $C \in R^{n \times n}$ and each element of the matrix is equal to

$$C_{ij}(q, \dot{q}) = \sum_{k=1}^{n} c^{i}_{kj}(q)\dot{q}_k,$$

where

$$c^{i}_{kj}(q) = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right). \tag{11}$$

Dynamics (6/7)

We can calculate gravity vector D with following formula [4]

$$D_i = -\sum_{k=i}^{n} m_k \left\langle g, \frac{\partial A_{0i}}{\partial q_i} R_i \right\rangle \tag{12}$$

where $g = [g_x, g_y, g_z, 0]^T$ and $R_i = [\bar{x}_i, \bar{y}_i, \bar{z}_i, 1]^T$.

Wrocław University of Science and Technology

Dynamics (7/7)

Lastly, to calculate friction forces in the manipulator joints we can use Tustin friction model [1]

$$T(\dot{q}) = T_v \dot{q} + T_s sgn(\dot{q}) \tag{13}$$

where T_v is a viscous friction coefficient while T_s is a static friction coefficient.

Implementation

However, the non-linear function *sgn*() should be replaced rather with $\tanh()$ or $\arctan()$ because of computation reasons based on

$$\lim_{x \to +\infty} tanh(s_c x) = sgn(x) \tag{14}$$

where $s_c > 0$ and defines how closely $sgn(x)$ is approximated.

Motivation

Input-output decoupling is a method which enables one to control end-effector's trajectory. It other words, it translates desired trajectory given in task coordinates like (X_0, Y_0, Z_0) into joint space q.

What is more, the input-output decoupling can be used instead of inverse kinematics.

Algorithm (1/5)

Let

$$y_i = k_i(q), \quad i = 1, \ldots, n \tag{15}$$

where $k_i(q)$ is ith element of end-effector kinematics vector. Then

$$\dot{y}_i = \frac{d}{dt}k_i(q) = \frac{\partial k_i}{\partial q}\frac{dq}{dt} = J_i(q)\dot{q}. \tag{16}$$

The time derivative of above equation gives

$$\begin{aligned} \ddot{y}_i &= \frac{d^2}{dt^2}k_i(q) = \dot{J}_i(q)\dot{q} + J_i(q)\ddot{q} \\ &= \dot{q}^T\frac{\partial^2 k_i}{\partial q^2}\dot{q} + J_i\ddot{q} = P_i + J_i\ddot{q}. \end{aligned} \tag{17}$$

Algorithm (2/5)

By collecting all output variables we obtain a following matrix equation

$$\ddot{y} = P + J\ddot{q}. \tag{18}$$

Let's consider dynamics of the manipulator (6). The real inertia matrix M is always positively defined, therefore we can reformulate (6) to

$$\ddot{q} = M^{-1}\left(u - C\dot{q} - D - T\right). \tag{19}$$

After substitution of dynamic equation (19) to (18) it yields

$$\begin{aligned} \ddot{y} &= P + JM^{-1}\left(u - C\dot{q} - D - T\right) \\ &= P - JM^{-1}C\dot{q} - JM^{-1}D - JM^{-1}T + JM^{-1}u. \end{aligned} \tag{20}$$

Wrocław University of Science and Technology

Algorithm (3/5)

The equation (20) is an affine system with following equation

$$\ddot{y} = F + Gu \tag{21}$$

where

$$F = P - JM^{-1}C\dot{q} - JM^{-1}D - JM^{-1}T, \tag{22}$$

$$G = JM^{-1}. \tag{23}$$

We assume that G is square and invertible. Therefore, the J has to be square and invertible, too.

of Science and Technology

Algorithm (4/5)

Injecting the control law given below as

$$u = G^{-1}\left(-F + v\right) \tag{24}$$

to the affine system (21), where v is a new input to the system, we obtain the closed-loop system expressed in the form of double linear integrator.

$$\ddot{y} = v.$$

Wrocław University of Science and Technology

Algorithm (5/5)

To ensure that the desired trajectory is followed with end-effector by moving only its joints we propose PD controller with correction

$$v = \ddot{y}_d - K_d\dot{e} - K_p e \tag{25}$$

where y_d is a desired trajectory of the end-effector, $K_p = K_p^T > 0$, $K_d = K_d^T > 0$, and the system error is defined as $e = y - y_d$ and its time derivative equals to $\dot{e} = \dot{y} - \dot{y}_d$. To ensure that the procedure of input-output decoupling is possible, the necessary condition defined by Isidori has to be met [3]. It says that the number of inputs to the system has to be equal to the number of system's outputs.

Results (1/2)

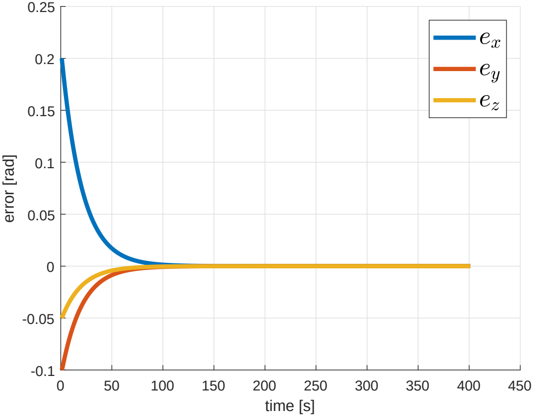

Figure: Errors between real and desired trajectory in task space

Results (2/2)

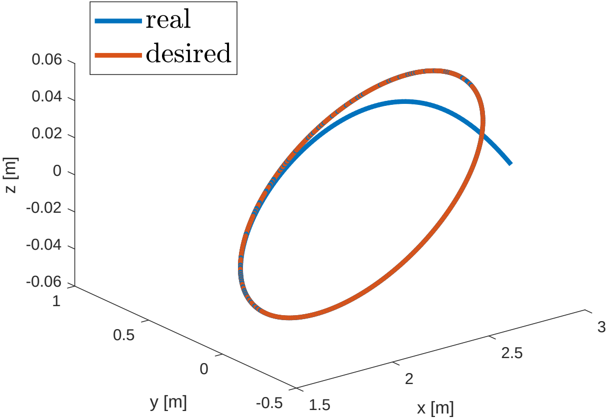

Figure: Real and desired trajectory

Matlab

Model of dynamics is defined as a set of 1^{st} order differential equations. It means that instead of simulating dynamics which is given as

$$\ddot{q} = f(q, \dot{q}, t)$$

we have to rewrite equations to a set of 1^{st} order differential equations, e.g.

$$\begin{aligned}
q_{d,0} &= q, \\
q_{d,1} &= \dot{q}_{d,0}, \\
q_{d,2} &= \dot{q}_{d,1}, \\
&\vdots \\
q_{d,n} &= \dot{q}_{d,n-1}, \\
q_{d,n} &= g(q_{d,0}, q_{d,1}, \ldots, q_{d,n-1}, t).
\end{aligned}$$

Ordinary differential equation (ODE) (1/2)

Calling ODE solver in Matlab can be done with following instructions

```
modelNameFun = str2func('model');
opts = odeset('RelTol',1e-6,'AbsTol',1e-6);
opts = odeset(opts,'OutputFcn','odeplot');
[t, youtput] = ode45(@(t,y) modelNameFun(t,y,parameters), ...
    [0:sample_time:tEnd], ic, opts);
```

Ordinary differential equation (ODE) (2/2)

The model function should comply with requirements of an ODE function model.

```
function [ output_args , additional ] = model( t , input_args ,
    parameters)
qr_d1 = input_args(1:3);
qr = input_args(4:6);
% ...
output_args = zeros(6,1);
output_args(1:3) = qr_d2;
output_args(4:6) = qr_d1;
additional.param = param1;
additional.vector = vector1;
```

Wrocław University of Science and Technology

Literature (1/2)

H. Berghuis.
Model-based Robot Control: from Theory to Practice.
CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG, 1993.

I. Dulęba.
Modeling and control of mobile manipulators.
In *Proc. of the 6th IFAC Symposium on Robot Control, SYROCO'00*, pages 687–692, 2000.

A. Isidori.
Nonlinear Control Systems.
Springer-Verlag London, 1995.

W. Jacak and K. Tchoń.
Podstawy robotyki.
Politechnika Wrocławska, 1992 (in Polish).

Literature (2/2)

K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, and R. Muszyński.
Mobile manipulators and robots: models, motion planning, control.
PLJ Publisher, 2000 (in Polish).