SOLUTIONS MANUAL

INTRODUCTION TO ROBOTICS

MECHANICS AND CONTROL

THIRD EDITION

JOHN J. CRAIG

Upper Saddle River, New Jersey 07458

Associate Editor: *Alice Dworkin*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Craig Little*
Supplement Cover Manager: *Daniel Sandin*
Manufacturing Buyer: *Ilene Kahn*

Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0201-54362-1

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Inc., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., *Upper Saddle River, New Jersey*

Contents

Solutions Manual

Matlab Exercises – Solutions 107

Chapter 1
Introduction

Exercises

1.1) Here's just an example of a reasonable response: (ref. [8] in Chap. 1)

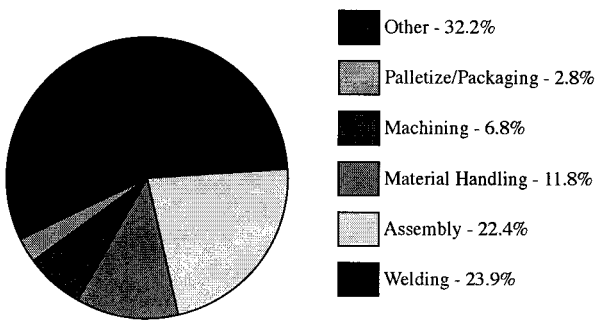
1955	Denavit & Hartenberg developed methodology for describing linkages.
1961	George Devol patents design of first robot.
1961	First unimate robot installed.
1968	Shakey Robot developed at S.R.I.
1975	Robot institute of America formed.
1975	Unimation becomes first Robot Co. to be protable.
1978	First Puma Robot shipped to GM.
1985	Total U.S. market exceeds 500 million dollars (annual revenue).

Developments might be split into a technical list and a business list.

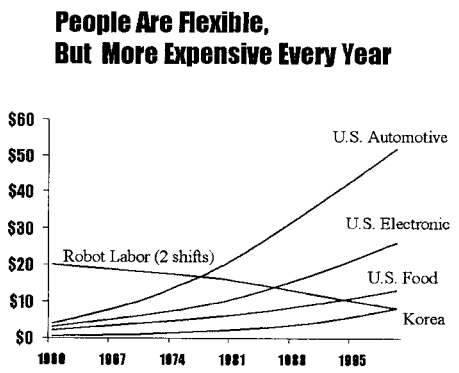
1.2) (Based on 1981 numbers)

Source:

L. Conigliaro, "robotics presentation, institutional investors conf.", May 28, 1981, Bache Newsletter 81–249.:



1.3)



1.4) *Kinematics* is the study of position and derivatives of position without regard to forces which cause the motion. *Workspace* is the locus of positions and orientations achievable by the end-effector of a manipulator. *Trajectory* is a time based function which specifies the position (and higher derivatives) of the robot mechanism for any value of time.

1.5) *Frame* is a coordinate system, usually specified in position and orientation relative to some imbedding frame. *Degrees of freedom* is the number of independent variables which must be specified in order to completely locate all members of a (rigid-body) mechanism. *Position control* implies the use of a control system, usually in a closed-loop manner, to control the position of one or more moving bodies.

1.6) *Force control* is the use of (usually closed-loop) algorithms to control the forces of contact generated when a robot touches its work environment. A *robot programming language* is a programming language intended for use in specifying manipulator actions.

1.7) *Structural stiffness* is the "K" in $F = K\Delta X$ (A.K.A "Hooke's law") which describes the rigidity of some structure. *Nonlinear control* refers to a closed loop control system in which either the system to be controlled, or the control algorithm itself is nonlinear in nature. *Off line programming* is the process of creating a program for a device without access to that device.

1.8) See references. For example, in 1985 average labor costs of \$15 to \$20 per hour are reasonable (depending how fringe benefits are calculated).

1.9) Obviously it has increased dramatically. Recently (1988–1990) the ratio doubles or even triples each year.

1.10) See Figure 1.3, but use latest data you can find.

Chapter 2

Spatial Descriptions and Transformations

Exercises

2.1) $R = \text{rot}(\hat{x}, \phi)\,\text{rot}(\hat{z}, \theta)$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\phi & -S\phi \\ 0 & S\phi & C\phi \end{bmatrix} \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta & -S\theta & 0 \\ C\phi S\theta & C\phi C\theta & -S\phi \\ S\phi S\theta & S\phi C\theta & C\phi \end{bmatrix}$$

2.2) $R = \text{rot}(\hat{x}, 45^\circ)\,\text{rot}(\hat{y}, 30^\circ)$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & .707 & -.707 \\ 0 & .707 & .707 \end{bmatrix} \begin{bmatrix} .866 & 0 & .5 \\ 0 & 1 & 0 \\ -.5 & 0 & .866 \end{bmatrix}$$

$$= \begin{bmatrix} .866 & 0 & .5 \\ .353 & .707 & -.612 \\ -.353 & .707 & .612 \end{bmatrix}$$

2.3) Since rotations are performed about axes of the frame being rotated, these are Euler-Angle style rotations:

$R = \text{rot}(\hat{z}, \theta)\,\text{rot}(\hat{x}, \phi)$

We might also use the following reasoning:

${}^{A}_{B}R(\theta, \phi) = {}^{B}_{A}R^{-1}(\theta, \phi)$

$= [\text{rot}(\hat{x}, -\phi)\,\text{rot}(\hat{z}, -\theta)]^{-1}$

$= \text{rot}^{-1}(\hat{z}, -\theta)\,\text{rot}^{-1}(\hat{x}, -\phi)$

$= \text{rot}(\hat{z}, \theta)\,\text{rot}(\hat{x}, \phi)$

Yet another way of viewing the same operation:

1st rotate by $\text{rot}(\hat{z}, \theta)$

2nd rotate by $\text{rot}(\hat{z}, \theta)\,\text{rot}(\hat{x}, \phi)\,\text{rot}^{-1}(z, \theta)$

2.3) *(continued)*

(This is a similarity transform)

Composing these two rotations:

$$= \text{rot}(\hat{z}, \theta)\,\text{rot}(\hat{x}, \phi)\,\text{rot}^{-1}(z, \theta) \cdot \text{rot}(\hat{z}, \theta)$$

$$= \text{rot}(\hat{z}, \theta)\,\text{rot}(\hat{x}, \phi)$$

$$= \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\phi & -S\phi \\ 0 & S\phi & C\phi \end{bmatrix}$$

$$= \begin{bmatrix} C\theta & -S\theta C\phi & S\theta S\phi \\ S\theta & C\theta C\phi & -C\theta S\phi \\ 0 & S\phi & C\phi \end{bmatrix}$$

2.4) This is the same as 2.3 only with numbers.

$$R = \text{rot}(\hat{z}, 30^\circ)\,\text{rot}(\hat{x}, 45)$$

$$= \begin{bmatrix} .866 & -.353 & .353 \\ .50 & .612 & -.612 \\ 0 & .707 & .707 \end{bmatrix}$$

2.5) If V_i is an eigenvector of R, then

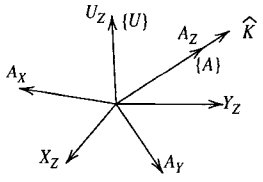
$$RV_i = 7V_i$$

If the eigenvalue associated with V_i is 1, then

$$RV_i = V_i$$

Hence the vector is not changed by the rotation R.
So V_i is the axis of rotation.

2.6) Imagine a frame $\{A\}$ whose $\hat{z}$ axis is aligned with the direction $\hat{k}$:



Then, the rotation with rotates vectors about $\hat{k}$ by θ degrees could be written:

$$R = {}^{U}_{A}R\,\text{rot}({}^{A}\hat{z}, \theta){}^{A}_{U}R \quad [1]$$

We write the description of $\{A\}$ in $\{U\}$ as:

$${}^{U}_{A}R = \begin{bmatrix} A & D & K_x \\ B & E & K_y \\ C & F & K_z \end{bmatrix}$$

If we multiply out Eq. [1] above, and then simplify using $A^2 + B^2 + C^2 = 1$, $D^2 + E^2 + F^2 = 1$, $[ABC] \cdot [DEF] = 0$, $[ABC] \otimes [DEF] = [K_x K_y K_z]$ we arrive at Eq. (2.80) in the book. Also, see [R. Paul]* page 25.

2.7) Let $R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$

(1) Compute $R_{11} + R_{22} + R_{33} = N$

(2) If $N = 3$, then $\theta = A\cos\left(\frac{N-1}{2}\right) = 0°$. Since rotation is zero, $\hat{K}$ is arbitrary.

(3) If $N = -1$, then $\theta = A\cos(-1) = 180°$. In this case Eq. (2.80) becomes:

$$\text{rot}(\hat{K}, 180°) = \begin{bmatrix} 2K_x^2 - 1 & 2K_xK_y & 2K_xK_z \\ 2K_xK_y & 2K_y^2 - 1 & 2K_yK_z \\ 2K_xK_z & 2K_yK_z & 2K_z^2 - 1 \end{bmatrix}$$

so:

$$2K_x^2 - 1 = R_{11} \Rightarrow K_x = \pm\sqrt{(R_{11}+1)/2}$$

$$2K_xK_y = R_{21} \Rightarrow K_y = R_{12}/2K_x$$

$$2K_xK_z = R_{31} \Rightarrow K_z = R_{31}/2K_x$$

However, if $K_x \cong 0$, then this is ill-defined, so use a different column for solution (not the first column as above).

(4) If $-1 < N < 3$ (so that $0 < \theta < 180°$) the use Eq. (2.82) in book.

2.8) Procedure RMTOAA is given essentially in the solution to 2.7. However, writing clean code to check the various cases is a good exercise in itself. Procedure AATORM is given by Eq. (2.80) and is easy.

2.9) The subroutines encode Eq. (2.64) and equations (2.66), (2.67), and (2.68).

2.10) The subroutines follow from eq. (2.72) and equations (2.74), (2.75), and (2.76).

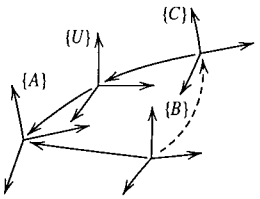
2.11) When they represent rotations about the same axis.

2.12) Velocity is a "free vector" and only will be affected by rotation, and not by translation:

$${}^AV = {}^A_BR\,{}^BV = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

$${}^AV = [-1.34 \quad 22.32 \quad 30.0]^T$$

2.13) By just following arrows, and reversing (by inversion) where needed, we have:



$$ {}^B_C T = {}^B_A T \; {}^U_A T^{-1} \; {}^C_U T^{-1} $$

Inverting a transform is done using eq. (2.40) in book. Rest is boring.

2.14) This rotation can be written as:

$$ {}^A_B T = \text{trans}({}^A\hat{P}, |{}^A P|)\, \text{rot}(\hat{K}, \theta)\text{trans}(-{}^A\hat{P}, |{}^A P|) $$

Where $\text{rot}(\hat{K}, \theta)$ is written as in eq. (2.77),

$$ \text{And} \quad \text{trans}({}^A\hat{P}, |{}^A P|) = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \text{And} \quad \text{trans}(-{}^A\hat{P}, |{}^A P|) = \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Multiplying out we get:

$$ {}^A_B T = \left[\begin{array}{ccc:c} R_{11} & R_{12} & R_{13} & Q_x \\ R_{21} & R_{22} & R_{23} & Q_y \\ R_{31} & R_{32} & R_{33} & Q_2 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] $$

where the R_{ij} are given be eq. (2.77). And:

$$ Q_x = P_x - P_x(K_x^2 V\theta + C\theta) - P_y(K_x K_y V\theta - K_z S\theta) $$
$$ - P_z(K_x K_z V\theta + K_y S\theta) $$

$$ Q_y = P_y - P_x(K_x K_y V\theta + K_z S\theta) - P_y(K_y^2 V\theta + C\theta) $$
$$ - P_z(K_y K_z V\theta + K_x S\theta) $$

$$ Q_z = P_z - P_x(K_x K_z V\theta - K_y S\theta) - P_y(K_y K_z V\theta + K_x S\theta) $$
$$ - P_z(K_z^2 V\theta + C\theta) $$

2.15) Recall that $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

so $e^{K\theta} = I + K\theta + \frac{1}{2!}K^2\theta^2 + \frac{1}{3!}K^3\theta^3 + \cdots$

$$K^2 = \begin{bmatrix} -K_y^2 - K_z^2 & K_xK_y & K_xK_z \\ K_xK_y & -K_x^2 - K_z^2 & K_yK_z \\ K_xK_z & K_yK_z & -K_x^2 - K_y^2 \end{bmatrix}$$

Writing out the (1,2) element of $e^{K\theta}$ (as an example) we have:

$$(e^{K\theta})_{1,2} = O + (-K_z)\theta + \frac{1}{2!}(K_xK_y)\theta^2 + \frac{1}{3!}K_2\theta^3 + \cdots$$

Recall that:

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots$$

$$\cos\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots.$$

$$V\theta = 1 - C\theta = \frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \cdots$$

We can write:

$$(e^{K\theta})_{1,2} = (-K_z)\theta + \frac{1}{3!}K_2\theta^3 - \frac{1}{5!}K_2\theta^5 + \cdots$$

$$+ \frac{1}{2!}(K_xK_y)\theta^2 - \frac{1}{4}(K_xK_y)\theta^4 + \cdots$$

Or:

$$(e^{K\theta})_{1,2} = -K_2S\theta + K_xK_yV\theta$$

Which is the value given in (2.80). Other elements may be checked similarly.

2.16) In method 1, the multiplication of two 3×3 matrices requires 27 multiplications and 18 additions. the computation of ${}^A_B R$ takes two matrix multiplies, or 57 multiplications, 36 additions the computation of ${}^A_B R\ {}^B P$ is 9 mult. and 6 additions. Hence, in one second this method will require:

$30 \times$ computation of ${}^A_D R = 30 \times 54$ mult.

30×36 add.

$100 \times$ computation of ${}^A p = 100 \times 9$ mult.

100×6 add.

Total = 2520 mult., 1680 Add.

In method two, computation of ${}^C_D R\ {}^D P$ requires 9 mult. and 6 add.; likewise the computation of ${}^B_C R\ {}^C P$ and ${}^A_B R\ {}^B P$, for a total of 27 mult. and 18 add. These must occur 100 times/sec., so in one second we have:

27×100 mult. = 2700 mult.

18×100 add. = 1800 add.

Therefore, method 1 is superior, but not by much.

2.17) $${}^A P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} R\cos\theta \\ R\sin\theta \\ Z \end{bmatrix}$$

2.18) $${}^A P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} R\cos\alpha\cos\beta \\ R\sin\alpha\cos\beta \\ R\sin\beta \end{bmatrix}$$

2.19) In the z-y-z Euler Angle set, the first rotation is:

$$R_1 = \text{rot}(\hat{z}, \alpha)$$

The second rotation expressed in fixed coordinates is:

$$R_2 = \text{rot}(\hat{z}, \alpha)\,\text{rot}(\hat{y}, \beta)\,\text{rot}^{-1}(\hat{z}, \alpha)$$

The third is:

$$R_3 = (R_2 R_1)\,\text{rot}(\hat{z}, \gamma)(R_2 R_1)^{-1}$$

The result is:

$$R = R_3 R_2 R_1 = \text{rot}(\hat{z}, \alpha)\,\text{rot}(\hat{y}, \beta)\,\text{rot}(\hat{z}, \gamma)$$

Which can be multiplied out to give the result of (2.72).

2.20) This is easily derived if you work backwards. i.e, substitute into Rodriquez's formula wherever $\hat{K} \otimes \hat{Q}$ or $\hat{K} \cdot \hat{Q}$ occur, collect terms, and you'll get (2.80).

2.21) Just use the given approximations in (2.80) to obtain:

$$R_K(\delta\theta) = \begin{bmatrix} 1 & -K_Z\delta\theta & K_Y\delta\theta \\ K_Z\delta\theta & 1 & -K_X\delta\theta \\ -K_Y\delta\theta & K_X\delta\theta & 1 \end{bmatrix}$$

More on this in Chapter 5.

2.22) So, given $R_1 = R_J(\alpha)$ and $R_2 = R_K(\beta)$ with $\alpha \ll 1$ and $\beta \ll 1$; show $R_1R_2 = R_2R_1$ if we form the product R_1R_2 and use $\alpha\beta \cong 0$ we have:

$$R_1R_2 = \begin{bmatrix} 1 & -J_Z\alpha - K_Z\beta & J_Y\alpha + K_Y\beta \\ J_Z\alpha + K_Z\beta & 1 & -J_X\alpha - K_X\beta \\ -J_Y\alpha - K_Y\beta & J_X\alpha + K_X\beta & 1 \end{bmatrix}$$

We see that j and k, as well as α and β appear symmetrically, so $R_1R_2 = R_2R_1$.

2.23) By definition ${}^U P_{\text{AORG}} = {}^U P_1$. Next, the vector from P_1 TO P_2 is a vector along the positive x-axis, so: ${}^U X_A = {}^U P_2 - {}^U P_1$, which normalized is: ${}^U \hat{X}_A = {}^U X_A / \|{}^U X_A\|$.

Now ${}^U V = {}^U P_3 - {}^U P_1$. A vector parallel to the positive Y_A-axis, can be formed using the Gram-Schmidt orthogonalization:

$${}^U Y_A = {}^U V - ({}^U V \cdot {}^U \hat{X}_A){}^U \hat{X}_A$$

and

$${}^U \hat{Y}_A = {}^U Y_A / |{}^U Y_A|$$

Finally, the unit vector ${}^U \hat{Z}_A$ can be found by a simple cross-product:

$${}^U \hat{Z}_A = {}^U \hat{X}_A \otimes {}^U \hat{Y}_A$$

Now:

$${}^U_A T = \left[\begin{array}{ccc:c} {}^U \hat{X}_A & {}^U \hat{Y}_A & {}^U \hat{Z}_A & {}^U P_{\text{Aorb}} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

2.24) This is a bit tricky here's most of it: ${}^{A}P = {}^{A}_{B}R\,{}^{B}P$ doesn't change length, so

$$\therefore\ {}^{A}P \cdot {}^{A}P - {}^{B}P \cdot {}^{B}P = 0$$

$$\underbrace{({}^{A}P - {}^{B}P)}_{F} \cdot \underbrace{({}^{A}P + {}^{B}P)}_{G} = 0$$

$$\therefore\ F \perp G$$

$$F = ({}^{A}_{B}R - I)\,{}^{B}P \quad G = ({}^{A}_{B}R + I)\ {}^{B}P$$

${}^{B}P = ({}^{A}_{B}R + I)^{-1}G$ (You can show ${}^{A}_{B}R + I$ non-shawar)

$$\therefore\ F = \underbrace{({}^{A}_{B}R - I)({}^{A}_{B}R + I)^{-1}}_{B}\,G$$

so, $F = BG$

(Now, one can show that if $X \cdot Y = 0$ and if $X = AY$ then a is skew-symmetric)

$\therefore\ B \in$ skew-sym matrices

$({}^{A}_{B}R - I) = B({}^{A}_{B}R + I)$

$(I - B){}^{A}_{B}R = I + B$ (now can show (I-B) non-show)

$$\therefore\quad \boxed{{}^{A}_{B}R = (I - B)^{-1}(I + B)} \quad \text{Q.E.D.}$$

2.25) Def. of e-val: $\lambda_i V_i = RV_i$ from physical insight we know $V_i = RV_i$ when V_i is the axis of rotation, $\therefore\ \boxed{\lambda_1 = 1}$ from (2.80) one can show (with some work) that $\det(R) = 1$. From linear algebra we have:

$$\Sigma_i \lambda_i = \text{trace(r)}\ \ \pi_i \lambda_i = \det(\text{r})$$

$\therefore\ \lambda_1\lambda_2\lambda_3 = 1$ or $\underline{\lambda_2\lambda_3 = 1}$ **[1]**

And compute the trace(r) from (2.80) to get

$\lambda_1 + \lambda_2 + \lambda_3 = 1 + 2\cos\theta$ or $\underline{\lambda_2 + \lambda_3 = 2\cos\theta}$ **[2]**

Now solve [1] & [2] above for λ_2 and λ_3 to get

$$\lambda_2 + \frac{1}{\lambda_2} = 2\cos\theta; \quad \lambda_2^2 - 2\cos\theta\lambda_2 + 1 = 0$$

$$\lambda_2 = \cos\theta \pm \sqrt{\cos^2\theta - 1}$$

$$\lambda_2 = \cos\theta \pm i\sin\theta$$

$$\therefore\ \lambda_1 = 1$$

$$\lambda_2 = \cos\theta + i\sin\theta = e^{i\theta}$$

$\lambda_3 = \cos\theta - i\sin\theta = e^{-i\theta}$ Q.E.D.

2.26) Somebody please send me a simple proof of this.
(For any given Euler convention it is not hard.)

2.27) $${}^{A}_{B}T = \begin{bmatrix} -1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.28) $${}^{A}_{C}T = \begin{bmatrix} 0 & -0.5 & 0.866 & 3 \\ 0 & 0.866 & 0.5 & 0 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.29) $${}^{B}_{C}T = \begin{bmatrix} 0 & 0.5 & -0.866 & 0 \\ 0 & -0.866 & -0.5 & 0 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.30) $${}^{C}_{A}T = \begin{bmatrix} 0 & 0 & -1 & 2 \\ -0.5 & 0.866 & 0 & 3 * 0.5 \\ 0.866 & -0.5 & 0 & -3 * 0.866 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.31) $${}^{A}_{B}T = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.32) $${}^{A}_{C}T = \begin{bmatrix} 0.866 & 0.5 & 0 & -3 \\ 0.5 & -0.866 & 0 & 4 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.33) $${}^{B}_{C}T = \begin{bmatrix} -0.866 & -0.5 & 0 & 3 \\ 0 & 0 & +1 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.34) $${}^{C}_{A}T = \begin{bmatrix} 0.866 & 0.5 & 0 & -3*.86 + 2 \\ 0.5 & -0.866 & 0 & -4*.86 - 1.5 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.35) Any R can be given: $R = R_X(\alpha)R_Y(\beta)R_Z(\gamma)$
and $\det(R) = \det(R_X(\alpha)) \cdot \det(R_Y(\beta)) \cdot \det(R_Z(\gamma))$

Using the formulas for rotation about a principle axis ((2.77) – (2.79)) its easy to show

$\det(R_x(\alpha)) = 1 \quad \forall\alpha$

$\det(R_y(\beta)) = 1 \quad \forall\beta$

$\det(R_z(\gamma)) = 1 \quad \forall\gamma$

$\therefore \det(R) = 1 \cdot 1 \cdot 1 = 1$ Q.E.D.

2.36) From Cayley's formula, the number of parameters needed to specify a rotation is the number of free parameters in an NXN skew symmetric matrix, which is $\frac{1}{2}(N^2 - N)$. The translational degrees of freedom are, of course n, so total is:

$\text{dof}(N) = N + \frac{1}{2}(N^2 - N) = \frac{1}{2}(N^2 + N)$ Q.E.D.

2.37) Form (2,4) element of $-{}^A_B R^T {}^A P_{\text{borb}}$

To get: -6.4

2.38) $V_1^T V_2 = \cos\theta$, R preserves angles, so,

$(RV_1)^T(RV_2) = V_1^T V_2$

$V_1^T R^T R V_2 = V_1^T V_2 \quad \therefore \quad R^T R = I \Rightarrow R^T = R^{-1}$

2.39) $\varepsilon_y^2 = \frac{1}{y}(1 + r_{11} + r_{22} + r_{33})$

$\varepsilon_y^2 >$ epsilon?

Yes	No
$\varepsilon_y = \sqrt{\varepsilon_y^2}$	$\varepsilon_y = 0$
$\varepsilon_1 = (r_{23} - r_{32})/4\varepsilon_y$	$\varepsilon_1^2 = -1/2(r_{22} + r_{33})$
$\varepsilon_2 = (r_{31} - r_{13})/4\varepsilon_y$	$\varepsilon_1^2 >$ epsilon?
$\varepsilon_3 = (r_{12} - r_{21})/4\varepsilon_y$	Yes: $\varepsilon_1 = \sqrt{\varepsilon_1^2}$; $\varepsilon_2 = r_{12}/2\varepsilon_1$; $\varepsilon_3 = r_{13}/2\varepsilon_1$
	No: $\varepsilon_1 = 0$; $\varepsilon_2^2 = \frac{1}{2}(1 - r_{33})$
	$\varepsilon_2^2 >$ epsilon?
	Yes: $\varepsilon_2 = \sqrt{\varepsilon_2^2}$; $\varepsilon_3 = r_{23}/2\varepsilon_2$
	No: $\varepsilon_2 = 0$; $\varepsilon_3 = 1$

2.40) Similar to algorithms of Section 2.8.

2.41) Similar to algorithms of Section 2.8.

Chapter 3

Manipulator Kinematics

Exercises

3.1)

α_{i-1}	a_{i-1}	d_i
0	0	0
0	L_1	0
0	L_2	0

$$ {}^0_1T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^1_2T = \begin{bmatrix} C_2 & -S_2 & 0 & L_1 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^2_3T = \begin{bmatrix} C_3 & -S_3 & 0 & L_2 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0_3T = {}^0_1T\,{}^1_2T\,{}^2_3T = \begin{bmatrix} C_{123} & -S_{123} & 0 & L_1C_1 + L_2C_{12} \\ S_{123} & C_{123} & 0 & L_1S_1 + L_2S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where:

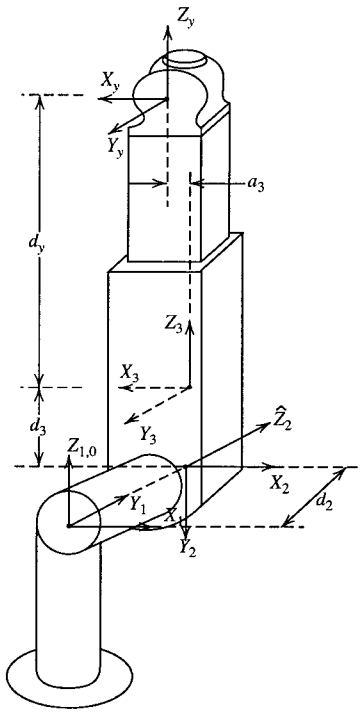
$C_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$

$S_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$, etc.

3.2)

α_{i-1}	a_{i-1}	d_i	θ_i
0	0	0	θ_1
$-90°$	0	d_2	θ_2
$90°$	0	d_3	$180°$
0	a_3	d_4	θ_4
90	0	0	θ_5
-90	0	0	θ_6

When $d_3 = 0$ the origins of frames 2 and 3 coincide. Frame 3 is fixed to link 3.



$$ {}^0_1T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^1_2T = \begin{bmatrix} C_2 & -S_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -S_2 & -C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^2_3T = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -d_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^3_4T = \begin{bmatrix} C_4 & -S_4 & 0 & a_3 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^4_5T = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

3.2 (*Continued*)

$$ {}^{5}_{6}T = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S_6 & -C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{0}_{6}T = {}^{0}_{3}T\,{}^{3}_{6}T $$

$$ {}^{0}_{3}T = \begin{bmatrix} -C_1C_2 & S_1 & C_1S_2 & -d_2S_1 + d_3C_1S_2 \\ -S_1C_2 & -C_1 & S_1S_2 & d_2C_1 + d_3S_1S_2 \\ S_2 & 0 & C_2 & d_3C_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{6}T = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -(C_4C_5S_6 + S_4C_6) & -C_4S_5 & a_3 \\ (S_4C_5C_6 + C_4S_6) & -S_4C_5S_6 + C_4C_6 & -S_4S_5 & 0 \\ S_5C_6 & -S_5S_6 & C_5 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{6}T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & P_x \\ R_{21} & R_{22} & R_{23} & P_y \\ R_{31} & R_{32} & R_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where:

$$ R_{11} = -C_1C_2C_4C_5C_6 + C_1C_2S_4S_6 + S_1S_4C_5C_6 + S_1C_4S_6 + C_1S_2S_5S_6 $$

$$ R_{12} = C_1C_2C_4C_5S_6 + C_1C_2S_4C_6 - S_1S_4C_5S_6 + S_1C_4C_6 - S_1S_2S_5S_6 $$

$$ R_{13} = C_1C_2C_4S_5 - S_1S_4S_5 + C_1S_2C_5 $$

$$ R_{21} = -S_1C_2C_4C_5C_6 + S_1C_2S_4S_6 - C_1S_4C_5C_6 - C_1C_4S_6 + S_1S_2S_5C_6 $$

$$ R_{22} = S_1C_2C_4C_5S_6 + S_1C_2S_4C_6 + C_1S_4C_5S_6 - C_1C_4C_6 - S_1S_2S_5S_6 $$

$$ R_{23} = S_1C_2C_4S_5 + C_1S_4S_5 + S_1S_2C_5 $$

$$ R_{31} = S_2C_4C_5C_6 - S_2S_4S_6 + C_2S_5C_6 $$

$$ R_{32} = -S_2C_4C_5S_6 - S_2S_4C_6 - C_2S_5S_6 $$

$$ R_{33} = -S_2C_4C_5 + C_2C_5 $$

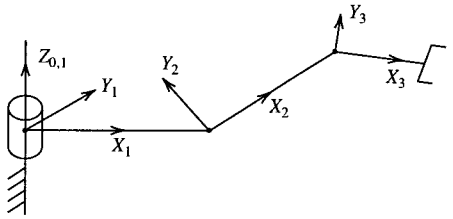
$$ P_x = -d_2S_1 + (d_3 + d_4)C_1S_2 - a_3C_1C_2 $$

$$ P_y = d_2C_1 + (d_3 + d_4)S_1S_2 - a_3S_1C_2 $$

$$ P_z = (d_3 + d_4)C_2 + a_3S_2 $$

3.3)

α_{i-1}	a_{i-1}	d_i
0	0	0
90°	L_1	0
0	L_2	0



$$ {}^{0}_{1}T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} C_2 & -S_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

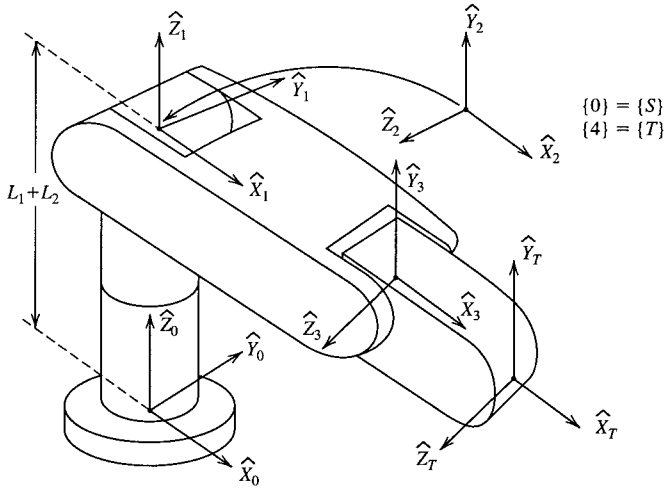
3.3) *(Continued)*

$$ {}^{2}_{3}T = \begin{bmatrix} C_3 & -S_3 & 0 & L_2 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{B}_{W}T = {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T $$

$$ {}^{B}_{W}T = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & L_1C_1 + L_2C_1C_2 \\ S_1C_{23} & -S_1S_{23} & -C_1 & L_1S_1 + L_2S_1C_2 \\ S_{23} & C_{23} & 0 & L_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

3.4)

α_{i-1}	a_{i-1}	d_i	θ_i
0	0	$L_1 + L_2$	θ_1
90°	0	0	θ_2
0	L_3	0	θ_3
0	L_4	0	0



$$ {}^{0}_{1}T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 + L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} C_2 & -S_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} C_3 & -S_3 & 0 & L_3 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

3.5) The subroutine computes Eq. (3.13). By carefully grouping common terms, the multiplication count can be reduced below the obvious brute-force calculation.

3.6) The subroutine computes the product of matrices given in Eq. (3.7). (About 30 mults)

3.7) The subroutine computes ${}^{B}_{W}T$ given in the solution to Exercise 3.3. (About 43 mults)

3.8) When $\{G\} = \{T\}$ we have:

$${}^{B}_{W}T\,{}^{W}_{T}T = {}^{B}_{S}T\,{}^{S}_{G}T$$

so, $${}^{W}_{T}T = {}^{B}_{W}T^{-1}\,{}^{B}_{S}T\,{}^{S}_{G}T$$

3.9) ${}^{0}P_{TIP} = {}^{0}_{2}T\,{}^{2}P_{TIP}; {}^{2}P_{TIP} = \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix}$

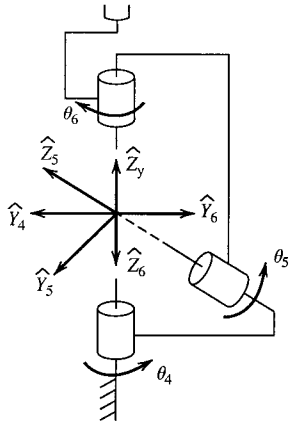
$${}^{0}P_{TIP} = \begin{bmatrix} C_1C_2 & -C_1S_2 & S_1 & L_1C_1 \\ S_1C_2 & -S_1S_2 & -C_1 & L_1S_1 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_2 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} L_1C_1 + L_2C_1C_2 \\ L_1S_1 + L_2S_1C_2 \\ L_2S_2 \end{bmatrix}$$

3.10) The computation can be structured as follows.
KK5, KK6, K8, ETC. Are precomputed constants.

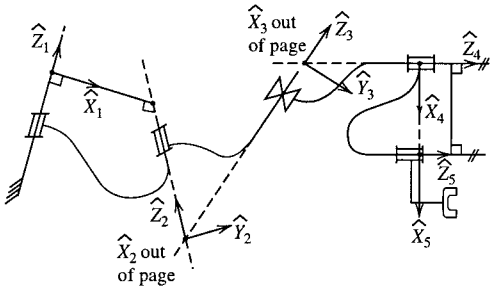
```
BEGIN
  cp2:=sk2*a[2]+lambda2;
  cp2:=(cp2*cp2+kk5)/kk6;
  cp3:=sk3*a[3]+lambda3;
  cp3:=(cp3*cp3+kk7)/kk8;
  sp2:=sqrt(1.0-cp2*cp2);
  sp3:=sqrt(1.0-cp3*cp3);
  t[3,4]:=kk1*cp2+kk2*sp2+kk3*cp3-kk4*sp3+hb:
  q:=-kk2*cp2+kk1*sp2+kk4*cp3+kk3*sp3;
  theta1:=sk1*a[1]+omega1;
  c1:=cos (theta1);
  s1:=sin (theta1);
  t[1,4]:=c1*q;
  t[2,4]:=s1*q;
  theta234:=k8-sk4*a[4];
  theta5:=k9-sk5*a[5];
  c234:=cos (theta234);
  s234:=sin (theta234);
  c5:=cos (theta5);
  s5:=sin (theta5);
  x1:=c1*c5;
  x2:=s1*s5;
  x3:=c1*s5;
  x4:=s1*c5;
  t[1,1]:=x1*c234-x2;
  t[2,1]:=x4*c234+x3;
  t[3,1]:=-c5*s234;
  t[1,2]:=-x3*c234-x4;
  t[2,2]:=-x2*c234+x1;
  t[3,2]:=s5*s234;
  t[1,3]:=c1*s234;
  t[2,3]:=s1*s234;
  t[3,3]:=c234;
END;  { atox }
```

3.11) Mechanism lies in page as drawn. All $\hat{X}$-axes are normal to page.



3.12) No! an arbitrary transformation requires six parameters.

3.13)



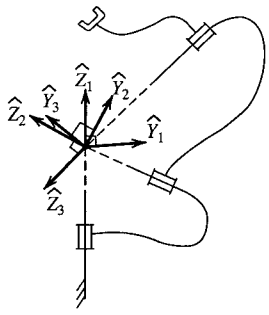
3.14) Draw a vector diagram of the situation, then not to hard to find:

$$a = \text{ABS}((Q - P) \cdot \hat{c})$$

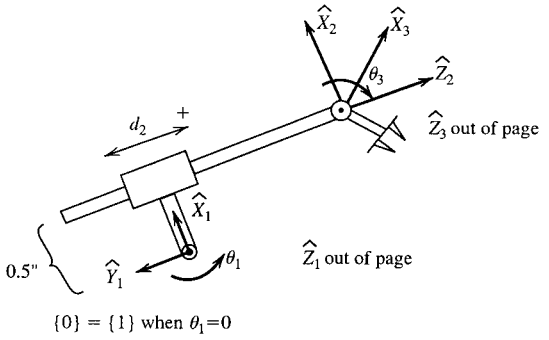
where $\hat{C} = \dfrac{\hat{M} \otimes \hat{N}}{\|\hat{M} \otimes \hat{N}\|}$

$$\alpha = \text{SGN}((Q - P) \cdot \hat{C}) \cos^{-1}(\hat{M} \cdot \hat{N})$$

3.15)



3.16) $a_0 = 0 \quad a_1 = 0.5^\circ \quad a_2 = 0$
$\alpha_0 = 0 \quad \alpha_1 = 90^\circ \quad \alpha_2 = -90^\circ$
$d_1 = 0 \qquad\qquad d_3 = 0$
$\theta_2 = 0^\circ$

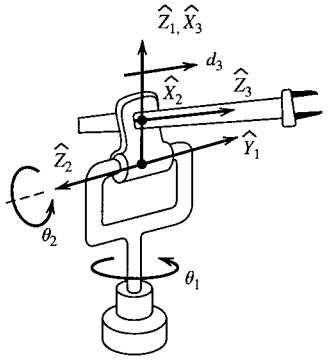


3.17) As shown:

$\theta_1 = 0$

$\theta_2 = 90^\circ$

$d_3 = 0$

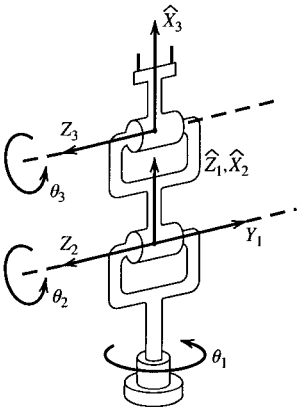


3.18) As shown:

$\theta_1 = 0^\circ$

$\theta_2 = 90^\circ$

$\theta_3 = 0^\circ$

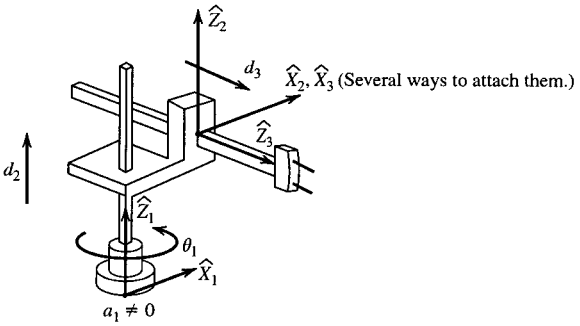


3.19) As shown:

$\theta_1 = 0$

$d_2 \cong 1''$

$d_3 = 0$

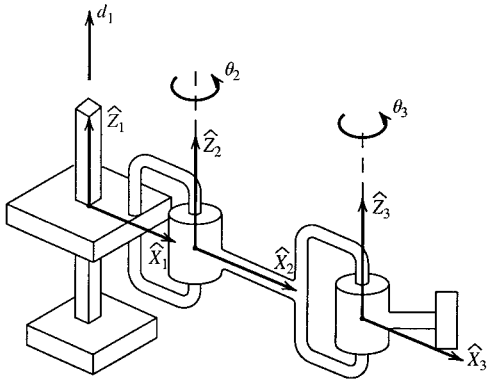


3.20) *Shown*:

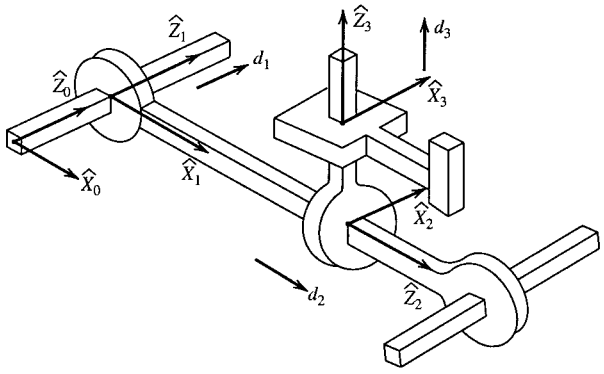
$d_1 = 0$

$\theta_2 = 0$

$\theta_3 = 0$



3.21) (Many possibilities)

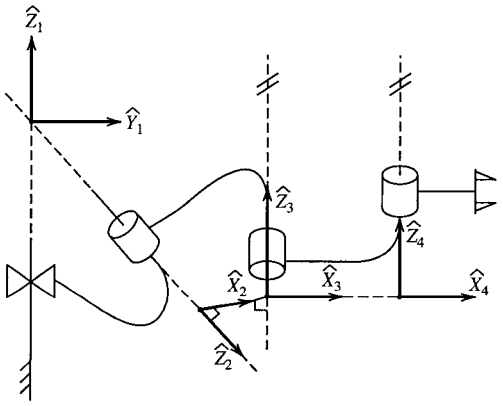


3.22) $\{y\}$ could be placed differently. Namely, with $d_3 > 0$.

$d_2 > 0$

$d_3 = 0$

$a_2 > 0 (a_i > 0$ always)

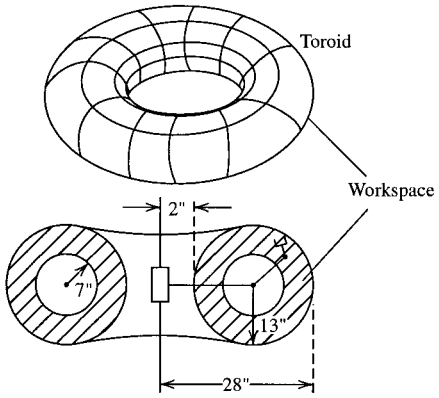


Chapter 4

Inverse Manipulator Kinematics

Exercises

4.1)



4.2) This problem can have different solutions depending how it is interpreted. I intended that a goal is specified which includes a desired orientation of the last link. In this case, the solution is fairly easy.

${}^{S}_{T}T$ is given, so compute:

$${}^{B}_{W}T = {}^{B}_{S}T\,{}^{S}_{T}T\,{}^{W}_{T}T^{-1}$$

Now ${}^{B}_{W}T = {}^{0}_{3}T$ which we write out as:

$${}^{0}_{3}T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & P_x \\ R_{21} & R_{22} & R_{23} & P_y \\ R_{31} & R_{32} & R_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From the solution of exercise 3 from chapter 3 we have:

$${}^{0}_{3}T = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & C_1(C_2L_2 + L_1) \\ S_1C_{23} & -S_1S_{23} & -C_1 & S_1(C_2L_2 + L_1) \\ S_{23} & C_{23} & 0 & S_2L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equate elements $(1, 3)$: $S_1 = R_{13}$

Equate elements $(2, 3)$: $-C_1 = R_{23}$

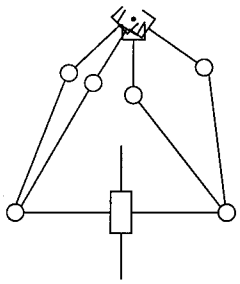
$\therefore$ $\boxed{\theta_1 = \text{atan2}(R_{13}, -R_{23})}$

If both $R_{13} = 0$ and $R_{23} = 0$ the goal is unattainable.

Equate elements $(1, 4)$: $P_x = C_1(C_2L_2 + L_1)$

Equate elements $(2, 4)$: $P_y = S_1(C_2L_2 + L_1)$

4.2) *(Continued)*



If $C_1 \neq 0$ then $C_2 = \frac{1}{L_2}\left(\frac{P_x}{C_1} - L_1\right)$

Else $C_2 = \frac{1}{L_2}\left(\frac{P_y}{S_1} - L_1\right)$

Equate Elements (3,4): $P_z = S_2 L_2$

so, $\boxed{\theta_2 = \text{atan2}\left(\frac{P_z}{L_2}, C_2\right)}$

Equate elements (3, 1): $S_{23} = R_{31}$

Equate elements (3, 2): $C_{23} = R_{32}$

so, $\boxed{\theta_3 = \textbf{atan2}(R_{31}, R_{32}) - \theta_2}$

If both R_{31} and R_{32} are zero, the goal is unattainable.

A second interpretation of the problem is that only a desired position is given (no orientation). In this there may be up to four solutions:

Assume[3] $P_{\text{tool}} = L_3 \hat{X}_3$, then

$$^0P_{\text{tool}} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} L_1C_1 + L_2C_1C_2 + L_3C_1C_{23} \\ L_1S_1 + L_2S_1C_2 + L_3S_1C_{23} \\ L_2S_2 + L_3S_{23} \end{bmatrix}$$

First,

$$S_1 = \frac{P_y}{L_1 + L_2C_2 + L_3C_{23}} \quad C_1 = \frac{P_x}{L_1 + L_2C_2 + L_3C_{23}}$$

so, $\boxed{\theta_1 = \textbf{atan2}(P_y, P_x) \textbf{ or } \textbf{atan2}(-P_y, -P_x)}$

Since the sign of the "$L_1 + L_2C_2 + L_3C_{23}$" term may be + or −.

Next, define:

$$\alpha = \begin{cases} \frac{P_x}{C_1} - L_1 & \text{if } C_1 \neq 0 \\ \frac{P_y}{S_1} - L_1 & \text{if } S_1 \neq 0 \end{cases}$$

And we have:

$$L_2C_2 + L_3C_{23} = \alpha$$

$$L_2S_2 + L_3S_{23} = P_z$$

Square and add these two equations to get:

$$L_2^2 + L_3^2 + 2L_2L_3C_3 = \alpha^2 + P_z^2$$

$$C_3 = \frac{1}{2L_2L_3}(\alpha^2 + P_z^2 - L_2^2 - L_3^2)$$

$S_3 = \pm\sqrt{1 - C_3^2}$; $\boxed{\theta_3 = \text{atan2}(S_3, C_3)}$

4.2) *(Continued)*

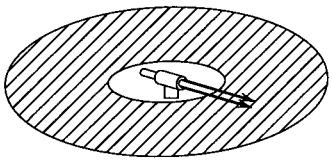
Finally,

$$L_3C_{23} = \alpha - L_2C_2$$

$$L_3S_{23} = P_z - L_2S_2$$

so, $\boxed{\theta_2 = \mathbf{atan2}(P_z - L_2S_2, \alpha - L_2C_2) - \theta_3}$

4.3) A flat disk of zero thickness with inner radius equal to minimum extension, and outer radius equal to maximum extension.



4.4) For an arm like this it is reasonable to specify a goal by giving the desired x & y coordinates of the tip and a value for θ_3, the wrist roll. After transforming back to P_X & P_T of the wrist (i.e. remove offset due to L_3) the solution is simple and corresponds to the conversion between cartesian and polar coordinates.

4.5) Turn the results of section 4.7 into a computer algorithm, plus check each solution to see if joints are in range.

4.6) To derive the "nearest" solution, we would like to minimize the rotation of each joint. Denote the starting angle of each joint as θ_{jo} (for j-th joint), and the final position of each joint as θ_{jF}. For each proposed solution, compute:

$$S = \sum_{j=1}^{N} |\theta_{jF} - \theta_{jo}|$$

And choose the "nearest" as the one which minimizes S. Sometimes a weighting factor is used (to penalize motion of "large" joints, for example) and so the score for each proposed solution is:

$$S = \sum_{j=1}^{N} W_j|\theta_{jF} - \theta_{jo}|$$

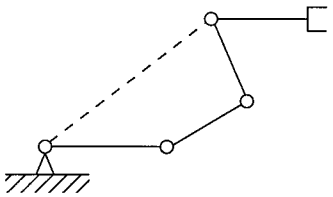
4.7) Repeatability is affected by:

1) Stepoy state error is servo system
2) Flexibility of links
3) Backlash in gears
4) Looseness in bearings
5) Noise in sensor readings
6) Thermal effects

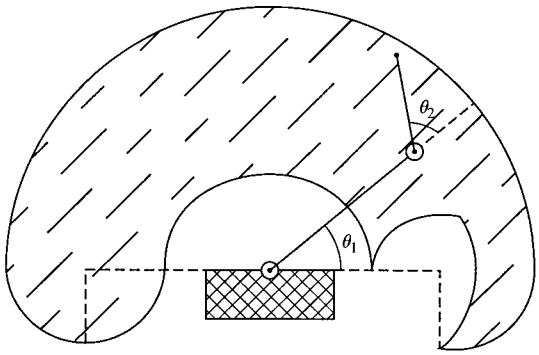
Accuracy is affected by all of the above, plus:

1) Imprecise knowledge of D.H. Parameters.

4.8) There are an infinite number of solutions. Imagine fixing the last link in position and orientation: Then, the first 3 links form a "4-bar linkage" which can take on an infinity of positions since it has a degree of freedom.



4.9) This is slightly trickier than it looks at first. Approximately:



4.10) This subspace with be given in terms of an expression for ${}^0_3T$ which is a function of three independent variables x, y, and θ. As these variables run throughout their ranges, a subspace is swept out.

As in Example 4.2 the origin of the end-effector frame (here, {3}) must have zero z-component. Also, $\hat{z}_3$ must have no z-component and it's direction is given by the coordinates of ${}^0P_{3ORG}$. So we have

$$ {}^0_3T = \begin{bmatrix} A & D & \alpha & X \\ B & E & \beta & Y \\ C & F & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where:

$$ \alpha = \frac{X}{\sqrt{x^2+y^2}} \text{ and } \beta = \frac{Y}{\sqrt{x^2+y^2}} $$

To find expressions for A,B,C,D,E,F as a function of independent variable, θ, consider rotating the vector $[001]^T$ about $\hat{Z}_3$ (given by $[\alpha\beta 0]^T$) by amount θ. This serves as our expression for $\hat{X}_3$ and can be found from eq. (2.80) Using $K = [\alpha\beta 0]^T$, and just taking the 3rd column.

This yields:

$$ \hat{X}_3 = [-\beta C\theta \; \alpha C\theta \; S\theta]^T $$

Then we compute $\hat{Y}_3 = \hat{Z}_3 \otimes \hat{X}_3$ to get

$$ {}^0_3T = \begin{bmatrix} -\beta C\theta & \beta S\theta & \alpha & X \\ \alpha C\theta & -\alpha S\theta & \beta & Y \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

4.10) *(Continued)*

where:

$$\alpha = \frac{X}{\sqrt{x^2 + y^2}} \quad \beta = \frac{Y}{\sqrt{x^2 + y^2}}$$

4.11) $\hat{V}$ must be rotated into alignment with $\hat{Z}_0$: ${}^0\hat{Z}_0 = {}^0_2R\,{}^2\hat{V}$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} C_1C_2 & -C_1S_2 & S_1 \\ S_2 & C_2 & 0 \\ -S_1C_2 & S_1S_2 & C_1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

There are two solutions:

$\theta_1 = \text{atan2}(-\sqrt{V_x^2 + V_y^2}, V_z)$

$\theta_2 = \text{atan2}(-V_y, V_x)$

And

$\theta_1 = \text{atan2}(\sqrt{V_x^2 + V_y^2}, V_Z)$

$\theta_2 = \text{atan2}(-V_y, V_x) + 180°$

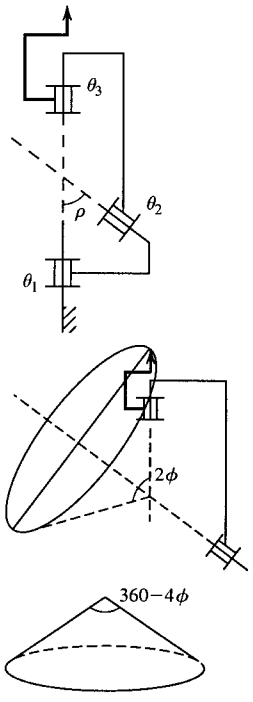
If $V_x = V_y = 0$ then *singular* and θ_2 is arbitrary.

4.12) *Find*: the set of unattainable orientations

Solution: Starting with joint 3, spinning it, frame 3 can attain any orientation about its $\hat{z}_3$ axis, pointing upward. The problem thus reduces to finding restrictions on the pointing direction of $\hat{z}_3$.

Next, spinning about joint 2, $\hat{z}_3$ can attain any orientation on the surface of a cone of apex angle 2ϕ:

Spinning this "cone" about joint 1, $\hat{z}_3$ can attain any orientation except in a cone shaped zone of apex angle $360 - 4\phi$:



4.13) 1. Faster computationally
2. Finds *all* solutions
3. Sometimes more accurate

4.14) No. Pieper's method gives the closed form solution for any 3-DOF manipulator. (See his thesis for all the cases)

4.15) See references [8] and [9].

4.16) Since $S\alpha_1 = 0$, easy to use Pieper's method:

$z = K_4$ (see pages 129–131 of text)

or

$${}^{o}P_{4z} = C\alpha_1 F_3 + C\alpha_1 d_2$$

$${}^{o}P_{4z} = F_3$$

$${}^{o}P_{4z} = A_3 S\alpha_2 S_3 + D_3 C\alpha_2 \text{ (other terms are zero)}$$

$$1.707 = \sqrt{2}\frac{\sqrt{2}}{2}S_3 + \sqrt{2}\frac{\sqrt{2}}{2}$$

$$1.707 = S_3 + 1$$

$$S_3 = 1.707 - 1 = 0.707$$

since $-180 < \theta_3 < 180$, there are 2 sols:

$$\boxed{\theta_3 = 45^\circ \text{ or } \theta_3 = 135^\circ}$$

4.17) Since $a_1 = 0$, we use Pieper's method with $r = K_3$

$3 = 3 + 2F_3 \quad \therefore F_3 = 0$

$0.707\, S_3 + 0.707 = 0 \quad \therefore S_3 = -1 \quad \therefore \theta_3 = -90$
Continuing on yields. . . .

$\theta = [0, 0, -90]$

4.18) 2

4.19) 4

4.20) 1

4.21) 2

4.22) 1

4.23) Just plug in the $\tan\theta/2$ substitution and collect terms. You'll find you arrive at a quartic in U. You'll also see a condition in which the coefficients of U^3 and U become zero — so it becomes quadratic in U^2.

4.24)

Revolute	*Linear*
$\alpha_{i-1} = \text{atan2}(-T_{23}, T_{33})$	$\alpha_{i-1} = \text{atan2}(-T_{23}, T_{33})$
$\theta_{i-1} = T_{14}$	$a_{i-1} = T_{14}$
$d_i = \sqrt{T_{24}^2 + T_{34}^2}$	$\theta_i = \text{atan2}(-T_{12}, T_{11})$

Chapter 5

Jacobians: Velocities and Static Forces

Exercises

5.1) The Jacobian in frame $\{o\}$ is:

$$^0J(\underline{\theta}) = \begin{bmatrix} -L_1S_1 - L_2S_{12} & -L_2S_{12} \\ L_1C_1 + L_2C_{12} & L_2C_{12} \end{bmatrix}$$

$$\text{DET } (^0J(\underline{\theta})) = -(L_2C_{12})(L_1S_1 + L_2S_{12}) + (L_2S_{12})(L_1C_1 + L_2C_{12})$$

$$= -L_1L_2S_1C_{12} - L_2^2S_{12}C_{12} + L_1L_2C_1S_{12} + L_2^2S_{12}C_{12}$$

$$= L_1L_2C_1S_{12} - L_1L_2S_1C_{12} = L_1L_2(C_1S_{12} - S_1C_{12})$$

$$= L_1L_2S_2$$

$\therefore$ The same result as when you start with $^3J(\underline{\theta})$, namely, the singular configurations are $\theta_2 = 0°$ or $180°$.

5.2) From exercise 3.3 we have:

$$^0_3T = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & L_1C_1 + L_2C_1C_2 \\ S_1C_{23} & -S_1S_{23} & -C_1 & L_1S_1 + L_2S_1C_2 \\ S_{23} & C_{23} & 0 & L_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and:

$$^3_4T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad ^0_4T = {}^0_3T\,{}^3_4T$$

we could then find $^0J(\underline{\theta})$ quite easily by differentiating $^0P_{\text{YORG}}$. Finally, $^4J(\underline{\theta})$ can be calculated as $^4_0R\,^0J(\underline{\theta})$. This might be tedious, so lets try "standard" velocity propagation as done in the text:

$$^1W_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \quad ^1V_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$^2W_2 = {}^2_1R\,^1W_1 + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix}$$

$$^2W_2 = \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad ^2V_2 = {}^2_1R(^1V_1 + {}^1W_1 \times {}^1P_2)$$

$$^2V_2 = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ L_1\dot{\theta}_1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ -L_1\dot{\theta}_1 \end{bmatrix}$$

5.2) *(Continued)*

$$^3W_3 = {}^3_2R\,{}^2W_2 + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix}$$

$$^3W_3 = \begin{bmatrix} S_{23}\dot{\theta}_1 \\ C_{23}\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix} \quad ^3V_3 = {}^3_2R(^2V_2 + {}^2W_2 \times {}^2P_3)$$

$$^3V_3 = \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ -L_1\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ L_2\dot{\theta}_2 \\ -L_2C_2\dot{\theta}_1 \end{bmatrix} \right)$$

$$^3V_3 = \begin{bmatrix} S_3L_2\dot{\theta}_2 \\ C_3L_2\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix} \quad ^4W_4 = {}^3W_3$$

$$^4V_4 = {}^4_3R(^3V_3 + {}^3W_3 \times {}^3P_y) = {}^3V_3 + {}^3W_3 \times {}^3P_4$$

$$= \begin{bmatrix} S_2L_2\dot{\theta}_2 \\ C_3L_2\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ L_3(\dot{\theta}_2 + \dot{\theta}_3 \\ -L_3C_{23}\dot{\theta}_1 \end{bmatrix}$$

$$= \begin{bmatrix} S_3L_2\dot{\theta}_2 \\ C_3L_2\dot{\theta}_2 - L_3(\dot{\theta}_2 + \dot{\theta}_3) \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 - L_3C_{23}\dot{\theta}_1 \end{bmatrix}$$

$$\therefore {}^4J(\underline{\theta}) = \begin{bmatrix} 0 & S_3L_2 & 0 \\ 0 & C_3L_2 + L_3 & L_3 \\ (-L_1 - L_2C_2 - L_3C_{23}) & 0 & 0 \end{bmatrix}$$

5.3) First, velocity analysis:

$$^1W_1 = {}^1_0R\,{}^0W_0 + \dot{\theta}_1\,{}^1\hat{z}_1 = \dot{\theta}_1\hat{z}$$

$$^1V_1 = {}^1_0R(^0V_0 + {}^0W_0 \times {}^0P_1) = 0$$

$$^2W_2 = {}^2_1R\,{}^1W_1 + \dot{\theta}_2\,{}^2\hat{Z}_2$$

$$^2W_2 = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$^2V_2 = {}^2_1R(^1V_1 + {}^1W_1 \times {}^1P_2)$$

$$^2V_2 = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \times \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \right)$$

$$^2V_2 = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ L_1\dot{\theta}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -L_1\dot{\theta}_1 \end{bmatrix}$$

5.3) *(Continued)*

$${}^3W_3 = {}^3_2R\,{}^2W_2 + \dot{\theta}_3\,{}^3\hat{Z}_3$$

$${}^3W_3 = \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} S_2C_3\dot{\theta}_1 + C_2S_3\dot{\theta}_1 \\ -S_2S_3\dot{\theta}_1 + C_2C_3\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix}$$

$${}^3W_3 = \begin{bmatrix} S_{23}\dot{\theta}_1 \\ C_{23}\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix}$$

$${}^3V_3 = {}^3_2R({}^2V_2 + {}^2W_2 \times {}^2P_3)$$

$${}^3V_3 = {}^3_2R\left(\begin{bmatrix} 0 \\ 0 \\ -L_1\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \times \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix}\right)$$

$${}^3V_3 = \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ L_2\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} L_2S_3\dot{\theta}_2 \\ L_2C_3\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix}$$

$${}^4W_4 = {}^4_3R\,{}^3W_3 + O; \quad {}^4_3R = I; \quad {}^4W_4 = {}^3W_3$$

$${}^4V_4 = {}^4_3R({}^3V_3 + {}^3W_3 \times {}^3P_4)$$

$${}^4V_4 = {}^4_3R\left(\begin{bmatrix} L_2S_3\dot{\theta}_2 \\ L_2C_3\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} S_{23}\dot{\theta}_1 \\ C_{23}\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix} \times \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix}\right)$$

$${}^4V_4 = \left(\begin{bmatrix} L_2S_3\dot{\theta}_2 \\ L_2C_3\dot{\theta}_2 \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ L_3(\dot{\theta}_2 + \dot{\theta}_3) \\ -L_3C_{23}\dot{\theta}_1 \end{bmatrix}\right)$$

$${}^4V_4 = \begin{bmatrix} L_2S_3\dot{\theta}_2 \\ L_2C_3\dot{\theta}_2 + L_3(\dot{\theta}_2 + \dot{\theta}_3) \\ -L_1\dot{\theta}_1 - L_2C_2\dot{\theta}_1 - L_3C_{23}\dot{\theta}_1 \end{bmatrix}$$

$${}^4V_4 = {}^4J(\underline{\theta})\underline{\dot{\theta}}$$

$$\therefore \quad {}^4J(\underline{\theta}) = \begin{bmatrix} 0 & L_2S_3 & 0 \\ 0 & L_2C_3 + L_3 & L_3 \\ -L_1 - L_2C_2 - L_3C_{23} & 0 & 0 \end{bmatrix}$$

Next, using force analysis:

$${}^4F_4 = \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} {}^4N_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$${}^3F_3 = {}^3_4R\,{}^4F_4 = \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix}$$

$${}^3N_3 = {}^3_4R\,{}^4N_4 + {}^3P_4 \times {}^3F_3 = \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} = \begin{bmatrix} 0 \\ -L_3F_Z \\ L_3F_Y \end{bmatrix}$$

$${}^2F_2 = {}^2_3R\,{}^3F_3 = \begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} = \begin{bmatrix} C_3F_X - S_3F_Y \\ S_3F_X + C_3F_Y \\ F_Z \end{bmatrix}$$

5.3) *(Continued)*

$$^2N_2 = \tfrac{2}{3}R^3N_3 + {}^2P_3 \times {}^2F_2$$

$$^2N_2 = \begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ -L_3F_Z \\ L_3F_Y \end{bmatrix} + \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} C_3F_X - S_3F_Y \\ S_3F_X + C_3F_Y \\ F_Z \end{bmatrix}$$

$$^2N_2 = \begin{bmatrix} L_3S_3F_Z \\ -L_2F_Z - L_3C_3F_Z \\ L_2(S_3F_X + C_3F_Y) + L_3F_Y \end{bmatrix}$$

$$^1F_1 = \tfrac{1}{2}R^2F_2 = \begin{bmatrix} C_2 & -S_2 & 0 \\ 0 & 0 & -1 \\ S_2 & C_2 & 0 \end{bmatrix}\begin{bmatrix} C_3F_X - S_3F_Y \\ S_3F_X + C_3F_Y \\ F_Z \end{bmatrix}$$

$$^1F_1 = \begin{bmatrix} C_2(C_3F_X - S_3F_Y) - S_2(S_3F_X + C_3F_Y) \\ -F_Z \\ S_2(C_3F_X - S_3F_Y) + C_2(S_3F_X + C_3F_Y) \end{bmatrix}$$

$$^1N_1 = \tfrac{1}{2}R^2N_2 + {}^1P_2 \times {}^1F_1$$

$$^1N_1 = \begin{bmatrix} C_2 & -S_2 & 0 \\ 0 & 0 & -1 \\ S_2 & C_2 & 0 \end{bmatrix}\begin{bmatrix} L_3S_3F_Z \\ -L_2F_Z - L_3C_3F_Z \\ L_2(S_3F_x + C_3F_Y) + L_3F_Y \end{bmatrix} + \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \times {}^1F_1$$

$$^1N_1 = \begin{bmatrix} C_2L_3S_3F_Z + S_2L_2F_Z + L_2S_2C_3F_Z \\ -L_2(S_3F_X + C_3F_Y) - L_3F_Y \\ L_3S_2S_3F_Z - L_2C_2F_Z - L_3C_2C_3F_Z \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ -L_1S_2(C_3F_X - S_3F_Y) - L_2C_2(S_3F_X + C_3F_Y) \\ -L_1F_Z \end{bmatrix}$$

To compute torques, take the z-component of the ^iN_i:

$$\tau_1 = [-L_1 - L_2C_2 + L_3(S_2S_3 - C_2C_3)]F_Z$$

$$\tau_2 = L_2S_3F_X + (L_2C_3 + L_3)F_Y$$

$$\tau_3 = L_3F_Y$$

$$\underline{\tau} = {}^4J^T(\underline{\theta})\begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix}$$

Which leads to same expression as before for $^4J(\underline{\theta})$.

Finally, by differentiation of kinematic equations:

$$^\circ P_{4ORG} = \begin{bmatrix} L_1C_1 + L_2C_1C_2 + L_3C_1C_{23} \\ L_1S_1 + L_2S_1C_2 + L_3S_1C_{23} \\ L_2S_2 + L_3S_{23} \end{bmatrix} \triangleq P$$

$$^\circ J(\underline{\theta}) = \begin{bmatrix} \frac{\partial \rho_X}{\partial \theta_1} & \frac{\partial \rho_X}{\partial \theta_2} & \frac{\partial \rho_X}{\partial \theta_3} \\ \frac{\partial \rho_Y}{\partial \theta_1} & \frac{\partial \rho_Y}{\partial \theta_2} & \frac{\partial \rho_Y}{\partial \theta_3} \\ \frac{\partial \rho_Z}{\partial \theta_1} & \frac{\partial \rho_Z}{\partial \theta_2} & \frac{\partial \rho_Z}{\partial \theta_3} \end{bmatrix}$$

5.3) *(Continued)*

$$^{\circ}J(\underline{\theta}) = \begin{bmatrix} -L_1S_1 - L_2S_1C_2 - L_3S_1C_{23} & -L_2C_1S_2 - L_3C_1S_{23} & -L_3C_1S_{23} \\ L_1C_1 + L_2C_1C_2 + L_3C_1C_{23} & L_2S_1S_2 - L_3S_1S_{23} & -L_3S_1S_{23} \\ 0 & L_2C_2 + L_3C_{23} & L_3C_{23} \end{bmatrix}$$

$$^{\circ}V_4 = {}^{\circ}J(\underline{\theta})\dot{\underline{\theta}}; {}^4V_4 = \underbrace{{}^4_0R^{\circ}J(\underline{\theta})}\, J(\underline{\theta})\dot{\underline{\theta}}$$

$${}^4_0R = \begin{bmatrix} C_1C_{23} & S_1C_{23} & S_{23} \\ -C_1S_{23} & -S_1S_{23} & C_{23} \\ S_1 & -C_1 & 0 \end{bmatrix}$$

Multiplying out ${}^4_0R^{\circ}J(\underline{\theta})$ is tedious, but sure enough, it leads again to the same expression for ${}^4J(\underline{\theta})$.

5.4) The mapping which potentially can be singular is: $Y = J(\theta)\dot{\underline{\theta}}$ for the "position domain", and $\underline{\tau} = J^T(\theta)F$ for the "force domain". Now since transposition has nothing to do with the rank of a (square) matrix, its clear that the singularities of $J(\underline{\theta})$ are the same as those of $J^T(\underline{\theta})$.

5.5) See "differential kinematic control equations for simple manipulators" by R. Paul, B. Shimand, and G. Mayer in IEEE trans. On systems, man, and cybernetics, Vol. smc-11, No. 6, June, 1981. The answer is given in equation (18) in that paper, but there are some typos, so I have included a pascal listing below which is corrected. Actually, in the Puma 560 there is some mechanical coupling between joints 4, 5, and 6 which is not taken into account in these formulations.

```
Procedure Jacobian (VAR THETA: VECT6; VAR J:MAT6);

VAR
  c2,c3,c4,c5,c6 : real;
  s2,s3,s4,s5,s6 : real;
  s23,c23 : real;
  k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11: real;
  J: mat6;
BEGIN

  fsc(theta[2],s2,c2); {Table lookup of sine-cosine}
  fsc(theta[3],s3,c3); {pairs,first argument is input,}
  fsc(theta[4],s4,c4); {second two are output.}
  fsc(theta[5],s5,c5);
  fsc(theta[6],s6,c6);
  s23:=s2*c3+c2*s3;
  c23:=c2*c3-s2*s3;

  k1:=d4*s23+a3*c23+a2*c2;          {note: 23 here}
  k2:=c4*c5*c6-s4*s6;
  k3:=c4*c5*s6+s4*s6;                 {has opposite sign}
  k4:=a3+a2*c3;           {corrected} {OF a3 in text!}
  k5:=d4+a2*s3;           {corrected}
  k6:=s4*c5*c6+c4*s6;
  k7:=-s4*c5*s6+c4*c6;
  k8:=s5*c6;
```

```
    k9:=s5*s6;
    k10:=s4*s5;
    k11:=c4*s5;

    j[1,1]:=k1*k6-d3*(c23*k2-s23*k8);
    j[1,2]:=k4*k8+k5*k2;                    {corrected}
    j[1,3]:=a3*k8+d4*k2;
    j[1,4]:=0.0;
    j[1,5]:=0.0;
    j[1,6]:=0.0;

    j[2,1]:=k1*k7-d3*(-c23*k3+s23*k9);
    j[2,2]:=-k4*k9-k5*k3;                   {corrected}
    j[2,3]:=-a3*k9-d4*k3;
    j[2,4]:=0.0;
    j[2,5]:=0.0;
    j[2,6]:=0.0;

    j[3,1]:=k1*k10-d3*(c23*k11+s23*c5);
    j[3,2]:=k5*k11-k4*c5;                   {corrected}
    j[3,3]:=-a3*c5+d4*k11;
    j[3,4]:=0.0;
    j[3,5]:=0.0;
    j[3,6]:=0.0;

    j[4,1]:=-(s23*k2+c23*k8);
    j[4,2]:=k6;
    j[4,3]:=k6;
    j[4,4]:=-k8;
    j[4,5]:=s6;
    j[4,6]:=0.0;

    j[5,1]:=s23*k3+c23*k9;
    j[5,2]:=k7;
    j[5,3]:=k7;
    j[5,4]:=k9;
    j[5,5]:=c6;
    j[5,6]:=0.0;

    j[6,1]:=-s23*k11+c23*c5;                {corrected}
    j[6,2]:=k10;
    j[6,3]:=k10;
    j[6,4]:=c5;
    j[6,5]:=0.0;
    j[6,6]:=1.0;

END;   { Jacobian }
```

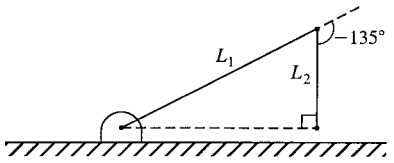
5.6) I think this is true! see B. Shimano, "the kinematic design and force control of computer controlled manipulators", Stanford A.I. Lab, memo # 313, 1978.

5.7) See Figure 11.9–it must be a purely cartesian manipulator:

$$\underline{V} = J\underline{\dot{\theta}}; \underline{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \underline{\dot{\theta}}$$

must have 3 orthogonal prismatic joints, so $\underline{\dot{\theta}} = [\dot{d}_1 \dot{d}_2 \dot{d}_3]^T$.

5.8) The Jacobian of this 2-link is:



$$^3J(\underline{\theta}) = \begin{bmatrix} L_1S_2 & 0 \\ L_1C_2 + L_2 & L_2 \end{bmatrix}$$

An isotropic point exists if

$$^3J = \begin{bmatrix} L_2 & 0 \\ 0 & L_2 \end{bmatrix} \text{ so,}$$

$$L_1S_2 = L_2$$

$$L_1C_2 + L_2 = 0$$

$$\text{or, } S_2 = \frac{L_2}{L_1}C_2 = \frac{-L_2}{L_1}$$

Now $S_2^2 + C_2^2 = 1$,

$$\text{so } \left(\frac{L_2}{L_1}\right)^2 + \left(\frac{-L_2}{L_1}\right)^2 = 1$$

$$\text{or } L_1^2 = 2L_2^2 \rightarrow L_1 = \sqrt{2}L_2$$

$$\text{Under this condition } S_2 = \frac{1}{\sqrt{2}} = \pm.707$$

and $C_2 = -.707$

$\therefore$ An isotropic point exists if $L_1 = \sqrt{2}L_2$ and in that case it exists when $\theta_2 = \pm 135°$

In this configuration, the manipulator looks momentarily like a Cartesian manipulator.

5.9) Unsolved. A small part of the answer can be found in Reference [7] of Chap. 15.

5.10) $\underline{\tau} = {}^3J^T\,{}^3\underline{F} \quad \therefore {}^3\underline{F} = {}^3J^{-T}\underline{\tau}$

$$^3J = \begin{bmatrix} L_1S_2 & 0 \\ L_1C_2 + L_2 & L_2 \end{bmatrix}$$

$$^3J^T = \begin{bmatrix} L_1S_2 & L_1C_2 + L_2 \\ 0 & L_2 \end{bmatrix}$$

so,

$$^3J^{-T} = \frac{1}{L_1L_2S_2}\begin{bmatrix} L_2 & -L_1C_2 - L_2 \\ 0 & L_1S_2 \end{bmatrix}$$

5.11) From (5.103):

$$ {}^{B}v = \begin{bmatrix} {}^{B}_{A}R & -{}^{B}_{A}R\,{}^{A}PX \\ 0 & {}^{B}_{A}R \end{bmatrix} {}^{A}v $$

$$ {}^{B}_{A}R\,{}^{A}PX = \begin{bmatrix} 0.86 & 0.5 & 0 \\ -0.5 & 0.86 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -5 & 0 \\ 5 & 0 & -10 \\ 0 & 10 & 0 \end{bmatrix} $$

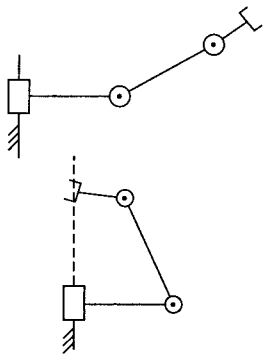
$$ = \begin{bmatrix} 2.5 & -4.3 & -5.0 \\ 4.3 & 2.5 & -8.6 \\ 0 & 10 & 0 \end{bmatrix} $$

$$ {}^{B}v = \begin{bmatrix} 0.86 & 0.5 & 0 & -2.5 & 4.3 & 5.0 \\ -0.5 & 0.86 & 0 & -4.3 & -2.5 & 8.6 \\ 0 & 0 & 1 & 0 & -10 & 0 \\ 0 & 0 & 0 & 0.86 & 0.5 & 0 \\ 0 & 0 & 0 & -0.5 & 0.86 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ -3 \\ 1.41 \\ 1.41 \\ 0 \end{bmatrix} $$

$$ {}^{B}v = [3.52 \;\; -7.80 \;\; -17.1 \;\; 1.91 \;\; 0.51 \;\; 0]^{T} $$

5.12) "Workspace boundary" any angle set: $\{\theta_1, \theta_2, 0\}$

"Workspace interior" any angle set such that: $L_1 + L_2C_2 + L_3C_{23} = 0$ (θ_1 is arbitrary)



5.13) $\underline{\tau} = {}^{O}J^{T}(\underline{\theta})\,{}^{O}\underline{F}$

$$ \underline{\tau} = \begin{bmatrix} -L_1S_1 - L_2S_{12} & L_1C_1 + L_2C_{12} \\ -L_2S_{12} & L_2C_{12} \end{bmatrix} \begin{bmatrix} 10 \\ 0 \end{bmatrix} $$

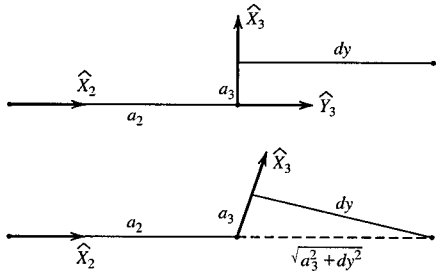
$$ \tau_1 = -10S_1L_1 - 10L_2S_{12} $$

$$ \tau_2 = -10L_2S_{12} $$

5.14) So, at singular position:

$\theta_3 = -\text{ATAN2}(d_y, a_3)$

If $a_3 \to 0$, then $\theta_3 \to -\text{ATAN2}(d_y, 0)$ which is $-90°$.



5.15) The kinematics can be done easily to obtain:

$$^0P_{\text{YORG}} = \begin{bmatrix} (d_2 + L_2 + L_3)S_1 \\ -(d_2 + L_2 + L_3)C_1 \\ 0 \end{bmatrix}$$

$$^0V = {}^0J\dot{\theta}$$

$$^0J = \begin{bmatrix} \frac{\partial^0 P_{\text{YORGX}}}{\partial\theta_1} & \frac{\partial^0 P_{\text{YORGX}}}{\partial\theta_2} & \frac{\partial^0 P_{\text{YORGX}}}{\partial\theta_3} \\ \frac{\partial^0 P_{\text{YORGY}}}{\partial\theta_1} & \frac{\partial^0 P_{\text{YORGY}}}{\partial\theta_2} & \frac{\partial^0 P_{\text{YORGY}}}{\partial\theta_3} \\ \frac{\partial^0 P_{\text{YORGZ}}}{\partial\theta_1} & \frac{\partial^0 P_{\text{YORGZ}}}{\partial\theta_2} & \frac{\partial^0 P_{\text{YORGZ}}}{\partial\theta_3} \end{bmatrix}$$

So,

$$^0J = \begin{bmatrix} (d_2 + L_2 + L_3)C_1 & S_1 & 0 \\ (d_2 + L_2 + L_3)S_1 & -C_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

5.16) You're in luck! The answer is given by Equation (5.42):

$$W = \begin{bmatrix} 0 & -S_1 & C_1S_2 \\ 0 & C_1 & S_1S_2 \\ 1 & 0 & C_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

5.17) For a prismatic joint i, the motion of the tool due to the joint is:

$$^0V_i = \dot{d}_i {}^0\hat{z}_i$$

$$^0W_i = O$$

Now,

$$^0V_{\text{TOOL}} = {}^0V_1 + {}^0V_2 + \cdots + {}^0V_6$$

$$^0W_{\text{TOOL}} = {}^0W_1 + {}^0W_2 + \cdots + {}^0W_6$$

so,

$$J(\theta) = \left[\begin{array}{c|c|c|c|c|c} A_1 & A_2 & A_3 & A_4 & A_5 & A_6 \\ \hline B_1 & B_2 & B_3 & B_4 & B_5 & B_6 \end{array}\right]$$

Where each A_i or B_i is a 3×1 vector given by:

$$A_i = \begin{cases} {}^0\hat{Z}_i \otimes ({}^0P_{\text{tool}} - {}^0P_{\text{iorg}}) & \text{if } i \text{ revolute} \\ {}^0\hat{Z}_i & \text{if } i \text{ linear} \end{cases}$$

$$B_i = \begin{cases} {}^0\hat{Z}_i & \text{if } i \text{ revolute} \\ O & \text{if } i \text{ linear} \end{cases}$$

5.18) Only the 4^{TN} column of ${}^0_3T$ matters.

Just directly differentiate it w.r.t. θ:

$$ {}^0J(\theta) = \begin{bmatrix} -L_1S_1 - L_2S_1C_2 & -L_2C_1S_2 & 0 \\ L_1C_1 + L_2C_1C_2 & -L_2S_1S_2 & 0 \\ 0 & L_2C_2 & 0 \end{bmatrix} $$

5.19) By partial differentiation:

$$ {}^0\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}_{20\text{Rf}} = \underbrace{\left[\begin{array}{c:c} -A_1S_1 - d_2C_1 & -S_1 \\ \hdashline A_1C_1 - d_2S_1 & C_1 \end{array}\right]}_{{}^0J(\theta)} \begin{bmatrix} \dot{\theta}_1 \\ \dot{d}_2 \end{bmatrix} $$

$\det({}^0J(\theta)) = -d_2$

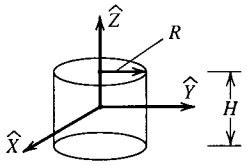
$\therefore$ Singular when $d_2 = 0$

5.20) At a singularity an N-DOF robot only has N-1 dof remaining. Hence, it can move freely in some N-1 dimensional subspace. However, it is still true that it has n joints. Therefore, we have a device which has one more joint than the dimensionality of the space it's described in — and that's what a redundant manipulator is.

Chapter 6
Manipulator Dynamics

Exercises

6.1) Use (6.17), but written in polar form is easier. For example, I_{ZZ}:



$$I_{ZZ} = \int_{-H/2}^{H/2}\int_{0}^{2\pi}\int_{0}^{R}(X^2+Y^2)\rho r\,dr\,d\theta dZ$$

$$X = R\cos\theta,\ Y = R\sin\theta,\ X^2+Y^2 = R^2(r^2)$$

$$I_{ZZ} = \int_{-H/2}^{H/2}\int_{0}^{2\pi}\int_{0}^{R}\rho r^3 dr d\theta dZ$$

$$I_{ZZ} = \frac{\pi}{2}R^4H\rho,\ \text{Volume} = \pi r^2 H$$

$$\therefore \text{Mass} = M = \rho\pi r^2 H \therefore \boxed{I_{ZZ} = \tfrac{1}{2}MR^2}$$

Similarly (only harder) is:

$$\boxed{I_{XX} = I_{YY} = \frac{1}{4}MR^2 + \frac{1}{12}MH^2}$$

From symmetry (or integration) $\boxed{I_{XY} = I_{XZ} = I_{YZ} = 0}$

$$\therefore {}^{C}I = \begin{bmatrix} \frac{1}{4}MR^2+\frac{1}{12}MH^2 & 0 & 0 \\ 0 & \frac{1}{4}MR^2+\frac{1}{12}MH^2 & 0 \\ 0 & 0 & \frac{1}{2}MR^2 \end{bmatrix}$$

6.2) ${}^{C_i}I$ given by (6.27). $\boxed{{}^{C_i}I_{ZZi} = \frac{Mi}{12}(L_i^2+W_i^2)}$

Other moments will not matter.

$${}^{1}P_{C_1} = r_1\hat{X}_1,\quad {}^{2}P_{C_2} = r_2\hat{X}_2 \quad \left(r_i = \frac{L_i}{2}\right)$$

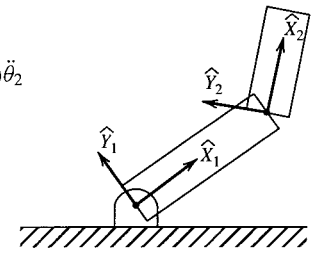
Derivation follows that of section 6.7 quite closely.
The answer is:

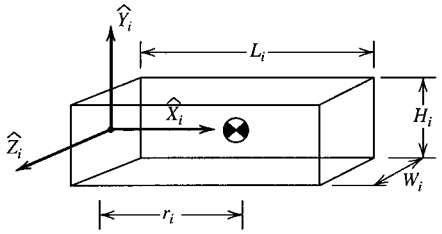
$$\tau_1 = (I_{ZZ1} + I_{ZZ2} + 2M_2r_2L_1C_2 + M_2L_1^2 + M_1r_1^2 + M_2r_2L_2)\ddot{\theta}_1$$
$$+ (M_2r_2L_2 + I_{ZZ2} + M_2L_1r_2C_2)\ddot{\theta}_2 - M_2L_1r_2S_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)$$
$$+ M_2r_2gC_{12} + M_1r_1gC_1 + M_2L_1gC_1$$

6.2) *(Continued)*

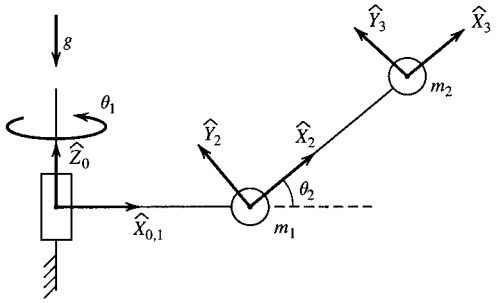
$$\tau_2 = (I_{ZZ2} + M_2L_1r_2C_2 + M_2L_2r_2)\ddot{\theta}_1 + (I_{ZZ2} + M_2L_2r_2)\ddot{\theta}_2$$
$$+ M_2L_1r_2S_2\dot{\theta}_1^2 + M_2r_2gC_{12}$$

Further compaction could be done since $r_i = \dfrac{L_i}{2}$.





6.3) This quite a bit of work, and I'm too lazy to write it out. Problem 6.5 is similar, only simpler since only 2 links, and a point-mass assumption. I will do that solution in detail. This one is similar, only more book keeping.



6.4)

$${}^{i+1}W_{i+1} = {}^{i+1}_{i}R\,{}^{i}W_i$$

$${}^{i+1}\dot{W}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{W}_i$$

$${}^{i+1}\dot{V}_{i+1} = {}^{i+1}_{i}R({}^{i}\dot{W}_i \times {}^{i}P_{i+1} + {}^{i}W_i \times ({}^{i}W_i \times {}^{i}P_{i+1}) + {}^{i}\dot{V}_i)$$
$$+ 2{}^{i+1}W_i \times \dot{d}_{i+1}{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}{}^{i+1}\hat{Z}_{i+1}$$

$${}^{i+1}\dot{V}_{C_{i+1}} = {}^{i+1}\dot{W}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}W_{i+1}$$
$$\times ({}^{i+1}W_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{V}_{i+1}$$

$${}^{i+1}F_{i+1} = m_{i+1}{}^{i+1}\dot{V}_{C_{i+1}}$$

$${}^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}{}^{i+1}\dot{W}_{i+1} + {}^{i+1}W_{i+1} \times {}^{C_{i+1}}I_{i+1}{}^{i+1}W_{i+1}$$

$${}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i$$

$${}^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1}$$

$$\tau_i = {}^{i}f_i^T\,{}^{i}\hat{Z}_i$$

6.5)

α_{i-1}	a_{i-1}	d_i	θ_i
0	0	0	θ_1
90°	L_1	0	θ_2
0	L_2	0	0

$$ {}^0_1T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1_2T = \begin{bmatrix} C_2 & -S_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2_3T = \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1P_2 = L_1\hat{X}_1 \quad {}^2P_3 = L_2\hat{X}_2 $$

$${}^1P_{C_1} = L_1\hat{X}_1 \; {}^2P_{C_2} = L_2\hat{X}_2 \; {}^{C_1}I_1 = 0 \; {}^{C_2}I_2 = 0$$

$${}^0\dot{V}_0 = g\hat{Z}_0. \quad \text{(since gravity points in } -\hat{Z}_0 \text{ Dir.)}$$

$$W_0 = \dot{W}_0 = 0 \quad \text{(base stationary)}$$

$$F_3 = x_3 = 0 \quad \text{(no forces on hand)}$$

Forward Velocity & Acceleration Iterations:

Link 1

$${}^1W_1 = {}^1_0R\,{}^0W_0 + \dot{\theta}_1\,{}^1\hat{Z}_1 = \dot{\theta}_1\,{}^1\hat{Z}_1 = [0 \quad 0 \quad \dot{\theta}_1]^T$$

$${}^1\dot{W}_1 = {}^1_0R\,{}^0\dot{W}_0 + {}^1_0R\,{}^0W_0 \otimes \dot{\theta}_1\,{}^1\hat{Z}_1 + \ddot{\theta}_1\,{}^1\hat{Z}_1$$

$${}^1\dot{W}_1 = \ddot{\theta}_1\,{}^1\hat{Z}_1 = [0 \quad 0 \quad \ddot{\theta}_1]^T$$

$${}^1\dot{V}_1 = {}^1_0R({}^0\dot{W}_0 \otimes {}^0P_1 + {}^0W_0 \otimes ({}^0W_0 \otimes {}^1P_2) + {}^0\dot{V}_0)$$

$${}^1\dot{V}_1 = \begin{bmatrix} C_1 & S_1 & 0 \\ -S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$${}^1\dot{V}_{C_1} = {}^1\dot{W}_1 \otimes {}^1P_{C_1} + {}^1W_1 \otimes ({}^1W_1 \otimes {}^1P_{C_1}) + {}^1\dot{V}_1$$

$${}^1\dot{V}_{C_1} = \ddot{\theta}_1\hat{Z}_1 \otimes L_1\hat{X}_1 + \dot{\theta}_1\hat{Z}_1 \otimes (\dot{\theta}_1\hat{Z}_1 \otimes L_1\hat{X}_1) + g\hat{Z}_1$$

$${}^1\dot{V}_{C_1} = L_1\ddot{\theta}_1\hat{Y}_1 + \dot{\theta}_1\hat{Z}_1 \otimes L_1\dot{\theta}_1\hat{Y}_1 + g\hat{Z}_1$$

$${}^1\dot{V}_{C_1} = L_1\ddot{\theta}_1\hat{Y}_1 + (-L_1\dot{\theta}_1^2)\hat{X}_1 + g\hat{Z}_1 = \begin{bmatrix} -L_1\dot{\theta}_1^2 \\ L_1\ddot{\theta}_1 \\ g \end{bmatrix}$$

Link 2

$${}^2W_2 = {}^2_1R\,{}^1W_1 + \dot{\theta}_2\,{}^2\hat{Z}_2$$

$${}^2W_2 = \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

6.5) *(Continued)*

$${}^2\dot{W}_2 = {}^2_1R\,{}^1\dot{W}_1 + {}^2_1R\,{}^1W_1 \otimes \dot{\theta}_2\hat{Z}_2 + \ddot{\theta}_2\hat{Z}_2$$

$${}^2\dot{W}_2 = \begin{bmatrix} S_2\ddot{\theta}_1 \\ C_2\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_2 \end{bmatrix}$$

$${}^2\dot{W}_2 = \begin{bmatrix} S_2\ddot{\theta}_1 \\ C_2\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} C_2\dot{\theta}_1\dot{\theta}_2 \\ -S_2\dot{\theta}_1\dot{\theta}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} S_2\ddot{\theta}_1 + C_2\dot{\theta}_1\dot{\theta}_2 \\ C_2\ddot{\theta}_1 - S_2\dot{\theta}_1\dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix}$$

$${}^2\dot{V}_2 = {}^2_1R({}^1\dot{W}_1 \times {}^1P_2 + {}^1W_1 \otimes ({}^1W_1 \otimes {}^1P_2) + {}^1\dot{V}_1)$$

$${}^2\dot{V}_2 = {}^2_1R(\ddot{\theta}_1\hat{Z}_1 \otimes L_1\hat{X}_1 + \dot{\theta}_1\hat{Z}_1 \otimes (\dot{\theta}_1\hat{Z}_1 \otimes L_1\hat{X}_1) + g\hat{Z}_1)$$

$${}^2\dot{V}_2 = {}^2_1R(L_1\ddot{\theta}_1\hat{Y}_1 - L_1\dot{\theta}_1^2\hat{X}_1 + g\hat{Z}_1)$$

$$= \begin{bmatrix} C_2 & 0 & S_2 \\ -S_2 & 0 & C_2 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -L_1\dot{\theta}_1^2 \\ L_1\ddot{\theta}_1 \\ g \end{bmatrix}$$

$${}^2\dot{V}_2 = [\,-L_1C_2\dot{\theta}_1^2 + S_2g,\ L_1S_2\dot{\theta}_1^2 + C_2g,\ -L_1\ddot{\theta}_1\,]^T$$

$${}^2\dot{V}_{C_2} = {}^2\dot{W}_2 \otimes {}^2P_{C_2} + {}^2W_2 \otimes ({}^2W_2 \otimes {}^2P_{C_2}) + {}^2\dot{V}_2$$

$${}^2\dot{V}_{C_2} = \begin{bmatrix} S_2\ddot{\theta}_1 + C_2\dot{\theta}_1\dot{\theta}_2 \\ C_2\ddot{\theta}_1 - S_2\dot{\theta}_1\dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} \otimes \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\otimes \left(\begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \otimes \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \right) + {}^2\dot{V}_2$$

$${}^2\dot{V}_{C_2} = \begin{bmatrix} 0 \\ L_2\ddot{\theta}_2 \\ -L_2C_2\ddot{\theta}_1 + L_2S_2\dot{\theta}_1\dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} S_2\dot{\theta}_1 \\ C_2\dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ L_2\dot{\theta}_2 \\ -L_2C_2\dot{\theta}_1 \end{bmatrix} + {}^2\dot{V}_2$$

$${}^2\dot{V}_{C_2} = \begin{bmatrix} -(L_1C_2 + L_2C_2^2)\dot{\theta}_1^2 - L_2\dot{\theta}_2^2 + S_2g \\ (L_1S_2 + L_2S_2C_2)\dot{\theta}_1^2 + L_2\ddot{\theta}_2 + C_2g \\ 2L_2S_2\dot{\theta}_1\dot{\theta}_2 - L_1\ddot{\theta}_1 - L_2C_2\ddot{\theta}_1 \end{bmatrix}$$

Inertial Equations:

Link 1

$${}^1F_1 = M_1{}^1\dot{V}_{C_1}$$

$${}^1F_1 = \begin{bmatrix} -M_1L_1\dot{\theta}_1^2 \\ M_1L_1\ddot{\theta}_1 \\ M_1g \end{bmatrix}$$

$${}^1N_1 = {}^{C_1}I_1{}^1\dot{W}_1 + {}^1W_1 \otimes {}^{C_1}I_1{}^1W_1$$

$${}^1N_1 = [0 \quad 0 \quad 0]^T$$

Link 2

$${}^2F_2 = M_2{}^2\dot{V}_{C_2}$$

6.5) *(Continued)*

$$^2F_2 = \begin{bmatrix} -M_2(L_1 + L_2C_2)C_2\dot{\theta}_1^2 - M_2L_2\dot{\theta}_2^2 + M_2S_2g \\ M_2(L_1 + L_2C_2)S_2\dot{\theta}_1^2 + M_2L_2\ddot{\theta}_2 + M_2C_2g \\ 2M_2L_2S_2\dot{\theta}_1\dot{\theta}_2 - M_2L_1\ddot{\theta}_1 - M_2L_2C_2\ddot{\theta}_1 \end{bmatrix}$$

$$^2N_2 = {}^{C_2}I_2{}^2\dot{W}_2 + {}^2W_2 \otimes {}^{C_2}I_2{}^2W_2 = [0 \quad 0 \quad 0]^T$$

Backward Force Iterations:

Link 2

$$^2F_2 = {}^2_3R^3F_3 + {}^2F_2 = {}^2F_2 \text{ (see above)}$$

$$^2n_2 = {}^2N_2 + {}^2_3R^3n_3 + {}^2P_{C_2} \otimes {}^2F_2 + {}^2P_3 \otimes {}^2_3R^3F_3$$

$$^2n_2 = {}^2P_{C_2} \otimes {}^2F_2 = L_2\hat{X}_2 \otimes {}^2F_2$$

$$^2n_2 = \begin{bmatrix} 0 \\ M_2L_1L_2\ddot{\theta}_1 - 2M_2L_2^2S_2\dot{\theta}_1\dot{\theta}_2 + M_2L_2^2C_2\ddot{\theta}_1 \\ M_2L_2(L_1 + L_2C_2)S_2\dot{\theta}_1^2 - M_2gL_2C_2 + M_2L_2^2\ddot{\theta}_2 \end{bmatrix}$$

Link 1

$$^1F_1 = {}^1_2R^2F_2 + {}^1F_1 \text{ (not needed, so skip)}$$

$$^1n_1 = {}^1N_1 + {}^1_2R\,{}^2n_2 + {}^1P_{C_1} \otimes {}^1F_1 + {}^1P_2 \otimes {}^1_2R\,{}^2F_2$$

$${}^1_2R\,{}^2n_2 = \begin{bmatrix} C_2 & -S_2 & 0 \\ 0 & 0 & -1 \\ S_2 & C_2 & 0 \end{bmatrix} {}^2n_2$$

$${}^1_2R\,{}^2n_2 = \begin{bmatrix} * \\ * \\ M_2L_1L_2C_2\ddot{\theta}_1 - 2M_2L_2^2S_2C_2\dot{\theta}_1\dot{\theta}_2 + M_2L_2^2C_2^2\ddot{\theta}_1 \end{bmatrix}$$

$$^1P_{C_1} \otimes {}^1F_1 = \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} -M_1L_1\dot{\theta}_1^2 \\ M_1L_1\ddot{\theta}_1 \\ M_1g \end{bmatrix} = \begin{bmatrix} 0 \\ -M_1gL_1 \\ M_1L_1^2\ddot{\theta}_1 \end{bmatrix}$$

$${}^1_2R\,{}^2F_2 = \begin{bmatrix} C_2 & -S_2 & 0 \\ 0 & 0 & -1 \\ S_2 & C_2 & 0 \end{bmatrix} \begin{bmatrix} -M_2(L_1 + L_2C_2)C_2\dot{\theta}_1^2 - M_2L_2\dot{\theta}_2^2 + M_2S_2g \\ M_2(L_1 + L_2C_2)S_2\dot{\theta}_1^2 + M_2gC_2 \\ 2M_2L_2S_2\dot{\theta}_1\dot{\theta}_2 - M_2L_1\ddot{\theta}_1 - M_2L_2C_2\ddot{\theta}_1 \end{bmatrix}$$

$${}^1_2R\,{}^2F_2 = \begin{bmatrix} * \\ M_2L_2C_2\ddot{\theta}_1 + M_2L_1\ddot{\theta}_1 - 2M_2L_2S_2\dot{\theta}_1\dot{\theta}_2 \\ * \end{bmatrix}$$

$$^1P_2 \otimes {}^1_2R\,{}^2F_2 = \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} * \\ M_2L_1\ddot{\theta}_1 - 2M_2L_2S_2\dot{\theta}_1\dot{\theta}_2 + M_2L_2C_2\ddot{\theta}_1 \\ * \end{bmatrix}$$

$$^1P_2 \otimes {}^1_2R\,{}^2F_2 = \begin{bmatrix} * \\ * \\ M_2L_1^2\ddot{\theta}_1 - 2M_2L_1L_2S_2\dot{\theta}_1\dot{\theta}_2 + M_2L_1L_2C_2\ddot{\theta}_1 \end{bmatrix}$$

$$\therefore {}^1n_1 = \begin{bmatrix} * \\ * \\ (M_1L_1^2 + M_2L_1^2 + M_2L_2^2C_2 + 2M_2L_1L_2C_2)\ddot{\theta}_1 - 2(L_1 + L_2C_2)M_2L_2S_2\dot{\theta}_1\dot{\theta}_2 \end{bmatrix}$$

6.5) *(Continued)*

Joint Torque Equations

$\tau_i = {}^i n_i \cdot {}^i \hat{Z}_i + V_i \dot{\theta}_i$ Added viscous friction term.

$$\tau_1 = (M_1 L_1^2 + M_2(L_1 + L_2 C_2)^2)\ddot{\theta}_1$$
$$- 2(L_1 + L_2 C_2) M_2 L_2 S_2 \dot{\theta}_1 \dot{\theta}_2 + V_1 \dot{\theta}_1$$

$$\tau_2 = M_2 L_2^2 \ddot{\theta}_2 + (L_1 + L_2 C_2) M_2 L_2 S_2 \dot{\theta}_1^2 + M_2 g L_2 C_2 + V_2 \dot{\theta}_2$$

or

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta)$$

where:

$$M(\theta) = \begin{bmatrix} (M_1 L_1^2 + M_2(L_1 + L_2 C_2)^2) & 0 \\ 0 & M_2 L_2^2 \end{bmatrix}$$

$$V(\theta_1 \theta) = \begin{bmatrix} -2(L_1 + L_2 C_2) M_2 L_2 S_2 \dot{\theta}_1 \dot{\theta}_2 \\ (L_1 + L_2 C_2) M_2 L_2 S_2 \dot{\theta}_1^2 \end{bmatrix}$$

$$G(\theta) = \begin{bmatrix} 0 \\ M_2 g L_2 C_2 \end{bmatrix}$$

6.6) The jacobian written in frame {0} is:

$${}^0 J(\theta) = \begin{bmatrix} -L_1 S_1 - L_2 S_{12} & -L_2 S_{12} \\ L_1 C_1 + L_2 C_{12} & L_2 C_{12} \end{bmatrix} \quad (5.67)$$

The inverse:

$${}^0 J^{-1}(\theta) = \frac{1}{L_1 L_2 S_2} \begin{bmatrix} L_2 C_{12} & L_2 S_{12} \\ -L_1 C_1 - L_2 C_{12} & -L_1 S_1 - L_2 S_{12} \end{bmatrix}$$

And the time derivative of ${}^0 J(\theta)$:

$${}^0 \dot{J}(\theta) = \begin{bmatrix} -L_1 C_1 \dot{\theta}_1 - L_2 C_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 C_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ -L_1 S_1 \dot{\theta}_1 - L_2 S_{12}(\dot{\theta}_1 + \dot{\theta}_2) & -L_2 S_{12}(\dot{\theta}_1 + \dot{\theta}_2) \end{bmatrix}$$

$$M_X(\theta) = J^{-T}(\theta) M(\theta) J^{-1}(\theta)$$

$$M_X(\theta) = \frac{1}{L_1^2 L_2^2 S_2^2} \begin{bmatrix} M_{11} L_2^2 C_{12}^2 + M_{22}(L_1 C_1 + L_2 C_{12})^2 & \alpha \\ \alpha & M_{11} L_2^2 S_{12}^2 + M_{22}(L_1 S_1 + L_2 S_{12})^2 \end{bmatrix}$$

$$\alpha = M_{11} L_2^2 S_{12} C_{12} + M_{22}(L_1 S_1 + L_2 S_{12})(L_1 C_1 + L_2 C_{12})$$

$$M_{11} = (M_1 L_1^2 + M_2(L_1 + L_2 C_2)^2) \quad M_{22} = M_2 L_2^2$$

The rest is messy, but follows from (6.99):

$$V_X(\theta_1 \dot{\theta}) = {}^0 J^{-T}(\theta)(V(\theta_1 \dot{\theta}) - M(\theta)\, {}^0 J^{-1}(\theta)\, {}^0 \dot{J}(\theta)\dot{\theta})$$

$$G_X(\theta) = {}^0 J^{-T}(\theta) G(\theta)$$

6.7) To describe the state $(\theta, \dot{\theta}, \ddot{\theta})$ of a *single* joint when each dimension of state is quantized to 16 "bins" requires $16 \times 16 \times 16$ addresses. But when joint θ_i is in any one of these "bins", joint θ_j might be in $16 \times 16 \times 16$ different states. So, for general

$\tau = f(\theta, \dot{\theta}, \ddot{\theta})$for 3 joints

We would need $(16 \times 16 \times 16)^3$ locations. Each location stores three values (τ_1, τ_2, τ_3), so memory size required is:

$\boxed{3(16 \times 16 \times 16)^3 \text{ locations}}$

6.8) For this manipulator:

$$ {}^0_1T = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1_2T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$${}^0P_1 = 0 \quad {}^1P_2 = d_2\hat{Y}_1 \quad {}^1P_{C_1} = 0^2 P_{C_2} = 0$$

$${}^0\dot{V}_0 = g\hat{Z}_0 \quad W_0 = \dot{W}_0 = 0 \quad f_3 = n_3 = 0$$

Forward Vel. & Acc. Iterations:

Link 1

$${}^1W_1 = {}^1_0R\,{}^0W_0 + \dot{\theta}_1\hat{Z}_1 = [0 \quad 0 \quad \dot{\theta}_1]^T$$

$${}^1\dot{W}_1 = {}^1_0R\,{}^0\dot{W}_0 + {}^1_0R\,{}^0W_0 \otimes \dot{\theta}_1\hat{Z}_1 + \ddot{\theta}\hat{Z}_1$$

$${}^1\dot{W}_1 = [0 \quad 0 \quad \ddot{\theta}_1]^T$$

$${}^1\dot{V}_1 = {}^1_0R({}^0\dot{W}_0 \otimes {}^0P_1 + {}^0W_0 \otimes ({}^0W_0 \otimes {}^1P_2) + {}^0\dot{V}_0)$$

$${}^1\dot{V}_1 = [0 \quad 0 \quad g]^T$$

$${}^1\dot{V}_{C_1} = {}^1\dot{W}_1 \otimes {}^1P_{C_1} + {}^1W_1 \otimes ({}^1W_1 \otimes {}^1P_{C_1}) + {}^1\dot{V}_1$$

$${}^1\dot{V}_{C_1} = [0 \quad 0 \quad g]^T$$

Link 2 (Prismatic)

$${}^2W_2 = {}^2_1R\,{}^1W_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\dot{\theta}_1 \\ 0 \end{bmatrix}$$

$${}^2\dot{W}_2 = {}^2_1R\,{}^1\dot{W}_1 = [0 \quad -\ddot{\theta}_1 \quad 0]^T$$

$${}^2\dot{V}_2 = {}^2_1R({}^1\dot{W}_1 \otimes {}^1P_2 + {}^1W_1 \otimes ({}^1W_1 \otimes {}^1P_2) + {}^1\dot{V}_1)$$

$$+ 2\,{}^2W_2 \otimes \dot{d}_2\hat{Z}_2 + \ddot{d}_2\hat{Z}_2$$

6.8) *(Continued)*

$${}^2\dot{V}_2 = {}^2_1R(-d_2\ddot{\theta}_1\hat{X} + (-d_2\dot{\theta}_1^2\hat{Y}) + g\hat{Z})$$

$$+ 2(-\dot{d}_2\dot{\theta}_1\hat{X}) + \ddot{d}_2\hat{Z}$$

$${}^2\dot{V}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} -d_2\ddot{\theta}_1 \\ -d_2\dot{\theta}_1^2 \\ g \end{bmatrix} + \begin{bmatrix} -2\dot{d}_2\dot{\theta}_1 \\ 0 \\ \ddot{d}_2 \end{bmatrix}$$

$${}^2\dot{V}_2 = \begin{bmatrix} -2\dot{d}_2\dot{\theta}_1 - d_2\ddot{\theta}_1 \\ -g \\ -d_2\dot{\theta}_1^2 \qquad +\ddot{d}_2 \end{bmatrix}$$

$${}^2\dot{V}_{C_2} = {}^2\dot{V}_2 (\text{since } {}^2P_{C_2} = 0)$$

Inertial Equations:

Link 1

$${}^1F_1 = M_1{}^1\dot{V}_{C_1} = [\,0 \quad 0 \quad M_1g\,]^T$$

$${}^1N_1 = {}^{C_1}I_1{}^1\dot{W}_1 + {}^1W_1 \otimes {}^{C_1}I_1{}^1W_1$$

$${}^1N_1 = [\,0 \quad 0 \quad I_{ZZ1} \quad \ddot{\theta}_1\,]^T$$

Link 2

$${}^2F_2 = M_2{}^2\dot{V}_{C_2} = \begin{bmatrix} -2M_2\dot{d}_2\dot{\theta}_1 - M_2d_2\ddot{\theta}_1 \\ -M_2g \\ -M_2d_2\dot{\theta}_1^2 + M_2\ddot{d}_2 \end{bmatrix}$$

$${}^2N_2 = {}^{C_2}I_2{}^2\dot{W}_2 + {}^2W_2 \otimes {}^{C_2}I_2{}^2W_2 = [\,0 \quad 0 \quad 0\,]^T$$

Backward Force Iterations:

Link 2

$${}^2f_2 = {}^2_3R^3F_3 + {}^2F_2 = {}^2F_2\begin{pmatrix} \text{see} \\ \text{last} \\ \text{page} \end{pmatrix}$$

$${}^2n_2 = {}^2N_2 + {}^2_3R^3n_3 + {}^2P_{C_2} \otimes {}^2F_2 + {}^2P_3 \otimes {}^2_3R^3F_3$$

$${}^2n_2 = [\,0 \quad 0 \quad 0\,]^T$$

Link 1

$${}^1f_1 = {}^1_2R^2F_2 + {}^1F_1 (\text{not needed, so skip})$$

$${}^1n_1 = {}^1N_1 + {}^1_2R^2n_2 + {}^1P_{C_1} \otimes {}^1F_1 + {}^1P_2 \otimes {}^1_2R^2F_2$$

$${}^1n_1 = \begin{bmatrix} 0 \\ 0 \\ I_{ZZ1}\ddot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ d_2 \\ 0 \end{bmatrix} \otimes \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} -2M_2\dot{d}_2\dot{\theta}_1 & -M_2d_2\ddot{\theta}_1 \\ -M_2g & \\ -M_2d_2\dot{\theta}_1^2 & +M_2\ddot{d}_2 \end{bmatrix}\right)$$

$${}^1n_1 = \begin{bmatrix} 0 \\ 0 \\ I_{ZZ1}\ddot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ d_2 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} -2M_2\dot{d}_2\dot{\theta}_1 - M_2d_2\ddot{\theta}_1 \\ -M_2d_2\dot{\theta}_1^2 \qquad +M_2\ddot{d}_2 \\ M_2g \end{bmatrix}$$

$${}^1n_1 = \begin{bmatrix} M_2d_2g \\ 0 \\ I_{ZZ1}\ddot{\theta}_1 + 2M_2d_2\dot{d}_2\dot{\theta}_1 + M_2d_2^2\ddot{\theta}_1 \end{bmatrix}$$

6.8) *(Continued)*

$$\tau_1 = I_{ZZ1}\ddot{\theta}_1 + 2M_2 d_2 \dot{d}_2 \dot{\theta}_1 + M_2 d_2^2 \ddot{\theta}_1$$

$$\tau_2 = -M_2 d_2 \dot{\theta}_1^2 + M_2 \ddot{d}_2$$

6.9) Very similar to 6.8. Assume the mass of link 3 is M_3 although this was not stated.

6.10) First, find ${}^2V_2$:

$${}^2V_2 = {}^2_1R({}^1V_1 + {}^1W_1 \otimes {}^1P_2) + \dot{d}_2\hat{Z}_2$$

$${}^2V_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -d_2\dot{\theta}_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{d}_2 \end{bmatrix}$$

$${}^2V_2 = \begin{bmatrix} -d_2\dot{\theta}_1 \\ 0 \\ \dot{d}_2 \end{bmatrix} = \begin{bmatrix} -d_2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{d}_2 \end{bmatrix}$$

$$\therefore {}^2J(\theta) = \begin{bmatrix} -d_2 & 0 \\ 0 & 1 \end{bmatrix}$$

$${}^2J^{-1}(\theta) = \begin{bmatrix} -\frac{1}{d_2} & 0 \\ 0 & 1 \end{bmatrix} \quad {}^2\dot{J}(\theta) = \begin{bmatrix} -\dot{d}_2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$M_X(\theta) = J^{-T}M(\theta)J^{-1}$$

$$M_X(\theta) = \begin{bmatrix} -\frac{1}{d_2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{ZZ1} + M_2 d_2^2 & 0 \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} -\frac{1}{d_2} & 0 \\ 0 & 1 \end{bmatrix}$$

$$M_X(\theta) = \begin{bmatrix} M_2 + \frac{I_{ZZ1}}{d_2^2} & 0 \\ 0 & M_2 \end{bmatrix}$$

$$V_X(\theta_1\dot{\theta}) = J^{-T}(V(\theta_1\dot{\theta}) - M(\theta)J^{-1}\dot{J}\dot{\theta})$$

$$= J^{-T}V(\theta_1\dot{\theta}) - M_X(\theta)\dot{J}\dot{\theta}$$

$$= \begin{bmatrix} -\frac{1}{d_2} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2M_2 d_2 \dot{d}_2 \dot{\theta}_1 \\ -M_2 d_2 \dot{\theta}_1^2 \end{bmatrix} - M_X(\theta) \begin{bmatrix} -\dot{\theta}_1 \dot{d}_2 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -2M_2 \dot{d}_2 \dot{\theta}_1 \\ -M_2 d_2 \dot{\theta}_1^2 \end{bmatrix} + \begin{bmatrix} \dot{\theta}_1 \dot{d}_2 \left(M_2 + \frac{I_{ZZ1}}{d_2^2} \right) \\ 0 \end{bmatrix}$$

$$V_X(\theta_1\dot{\theta}) = \begin{bmatrix} \left(\frac{I_{ZZ1}}{d_2^2} \right) \dot{\theta}_1 \dot{d}_2 - M_2 \dot{\theta}_1 \dot{d}_2 \\ -M_2 \dot{\theta}_1^2 d_2 \end{bmatrix}$$

$$G_X(\theta) = J^{-T}G(\theta) = [0 \quad 0]^T$$

6.11) By gear ratio of 100, I mean that the motor spins 100 times faster than the link.

The torque on the link is related to the acceleration of the link by: $\tau_L = I_{ZZ1}\dot{W}_L$ the gears increase torque by 100, so the motor torque, τ_M, is: $\tau_M = \dfrac{1}{100}\tau_L$; the motor acc. is $\dot{W}_M = 100\dot{W}_L$.

Combining, we have: $\tau_M = \left(\dfrac{1}{100}\right)^2 I_{ZZ1}\dot{W}_M$

Hence, inertia appears to motor as: $\left(\dfrac{1}{100}\right)^2 I_{ZZ1}$

6.12) $\theta_1(t) = Bt + Ct^2$, so

$\dot{\theta}_1 = B + 2Ct, \quad \ddot{\theta} = 2C$

so,

$$^1\dot{W}_1 = \ddot{\theta}_1\hat{Z}_1 = 2C\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ 2C \end{bmatrix}$$

$$^1\dot{V}_{C_1} = \begin{bmatrix} 0 \\ 0 \\ 2C \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \otimes \left(\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0 \\ 4C \\ 0 \end{bmatrix} + \begin{bmatrix} -2\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix}$$

$$^1\dot{V}_{C_1} = \begin{bmatrix} -2(B + 2Ct)^2 \\ 4C \\ 0 \end{bmatrix}$$

6.13)

$$^3M_X(\theta) = \begin{bmatrix} M_2 & 0 \\ 0 & \left(M_1 \dfrac{L_1^2}{(L_1 + L_2C_2)^2} + M_2\right) \end{bmatrix}$$

$$^3V_X(\theta_1\dot{\theta}) = \begin{bmatrix} -M_2L_2\dot{\theta}_1^2S_2 \\ -\left[\dfrac{M_1L_1^2L_2}{(L_1 + L_2C_2)^2} + M_2(2L_1 + L_2(1 + 2C_2))\right]\dot{\theta}_1\dot{\theta}_2S_2 \end{bmatrix}$$

$$^3G_X(\theta) = \begin{bmatrix} M_2gC_2 \\ 0 \end{bmatrix}$$

$$^3F_X(\theta_1\dot{\theta}) = \begin{bmatrix} \dfrac{V_2\dot{\theta}_2}{L_2} \\ \dfrac{V_1\dot{\theta}_1}{L_1 + L_2C_2} \end{bmatrix}$$

6.14) The bogus terms are:

$$\tau_1 = \cdots + M_1 d_2 \ddot{\theta}_1 + (M_1 d_2 \dot{\theta}_1 + M_2 \dot{d}_2) g C_1$$

$$F_2 = \cdots + M_1 \dot{d}_2 \ddot{\theta}_1 - M_1 d_1 \dot{d}_2 + M_2 d_2 g S_1$$

6.15) Well, the answer is the same as that shown in Example 6.5.

6.16) Upon inspection, this problem quite simple.**

$$\tau = \begin{bmatrix} f_1 \\ \tau_2 \end{bmatrix} = M(\theta)\ddot{\theta} + V(\theta_1 \dot{\theta}) + G(\theta)$$

where $\theta = \begin{bmatrix} d_1 \\ \theta_2 \end{bmatrix}$

then:

$$M(\theta) = \begin{bmatrix} M_1 + M_2 & 0 \\ 0 & I_{ZZ2} \end{bmatrix}, V(\theta_1 \dot{\theta}) = 0, G(\theta) = 0$$

6.17) See reference [22] from chapter 10, and the papers that it references. See also example 10.6.

6.18) Any reasonable $F(\theta, \dot{\theta})$ probably has the property that the friction force (or torque) on joint i depends only on the velocity of joint i, i.e.

$$F(\theta, \dot{\theta}) = [f_1(\theta, \dot{\theta}_1) f_2(\theta, \dot{\theta}_2) \ldots . f_N(\theta, \dot{\theta}_N)]^T$$

Also, each $f_i(\theta, \dot{\theta}_i)$ should be "passive", i.e. the function should lie in the 1st & 3rd quadrants. ** solution written by candle light in aftermath of 7.0-earthquake, 17.oct.89!

Chapter 7
Trajectory Generation

Exercises

7.1) Three cubics are required to connect a start point, two via points, and a goal point. That is, three for each joint, for a total of 18 cubics. Each cubic has four coefficients, so 72 coefficients are stored.

7.2) Use (7.6) with $\theta_o = -5, \theta_f = 80, t_f = 4.0$

$a_0 = -5$

$a_1 = 0$

$a_2 = \dfrac{3 \times 85}{16}$

$a_3 = -\dfrac{85}{32}$

7.3) Using (7.23) we see that the acceleration must be $\ddot{\theta} \geq \dfrac{4(85)}{16} = \dfrac{85}{4}$. If we choose exactly $\ddot{\theta} = \dfrac{85}{4}$ deg / sec^2, then the linear portion shrinks to zero, and $t_b = 2$ seconds.

Let's instead choose $\ddot{\theta} = 85$ deg / sec^2, then from (7.22) we have $t_B = 2 - \sqrt{3}$. So, the linear portion is $2\sqrt{3}$ seconds long, and the velocity during the linear portion is $\dot{\theta} = (85)(2 - \sqrt{3})$ deg / sec.

7.4) This is basically a programming exercise. The data describing the path (via points, duration, etc.) should be kept in a linked list, with one node per path segment. Likewise, the planned path (blend times, velocities, etc.) should be kept in a second linked list. This linked list is the output of the routine.

7.5) Use (7.15) to find coefficients.

7.6) Use (7.11) with the following for the first segment:

$\theta_o = 5 \quad \theta_f = 15 \quad \dot{\theta}_o = 0 \quad \dot{\theta}_f = 17.5 \quad t_f = 1$

This gives:

$a_0 = 5a_1 = 0 \quad a_2 = 12.5 \quad a_3 = -2.5$

And use (7.11) again for the second segment with:

$\theta_o = 15 \quad \theta_f = 40 \quad \dot{\theta}_o = 17.5 \quad \dot{\theta}_f = 0 \quad t_f = 1$

which gives:

$a_0 = 15 \quad a_1 = 17.5 \quad a_2 = 40.0 \quad a_3 = -32.5$

Then evaluate (7.3) to plot curves.

7.7) Neither segment is an "interior" segment, so (7.26) and (7.28) are used. From (7.26) calculate:

$$\ddot{\theta}_1 = 80$$

$$t_1 = 1 - \sqrt{1 - \frac{2(15-5)}{80}} = 1 - \frac{\sqrt{3}}{2}$$

$$\dot{\theta}_{12} = \frac{10}{1 - \frac{1}{2}\left(1 - \frac{\sqrt{3}}{2}\right)} = \frac{40}{2 + \sqrt{3}}$$

From (7.28):

$$\ddot{\theta}_3 = SGN(15-40)80 = -80$$

$$t_3 = 1 - \sqrt{1 + \frac{2(25)}{-80}} = 1 - \sqrt{1 - \frac{5}{8}}$$

$$\dot{\theta}_{23} = \frac{25}{1 - \frac{1}{2}\left(1 - \sqrt{1 - \frac{5}{8}}\right)} = \frac{50}{1 + \sqrt{1 - \frac{5}{8}}}$$

Finally, t_2 is an interior point so its computed from (7.24):

$$t_2 = \frac{\dot{\theta}_{23} - \dot{\theta}_{12}}{80}$$

7.8) Use (7.15) to compute coefficients, and use (7.3) to evaluate splines.

7.9) For first segment, use (7.11) with:

$$\theta_o = 5 \quad \theta_f = 15 \quad \dot{\theta}_o = 0 \quad \dot{\theta}_f = 0 \quad t_f = 2$$

This gives:

$$a_0 = 5 \quad a_1 = 0 \quad a_2 = \frac{15}{2} \quad a_3 = -\frac{5}{2}$$

For second segment, use (7.11) with:

$$\theta_o = 15 \quad \theta_f = -10 \quad \dot{\theta}_o = 0 \quad \dot{\theta}_f = 0 \quad t_f = 2$$

This gives:

$$a_0 = 15 \quad a_1 = 0 \quad a_2 = -\frac{75}{4} \quad a_3 = \frac{25}{4}$$

Then evaluate (7.3) to plot curves.

7.10) This is like exercise 7.7. use (7.26) to calculate:

$$\ddot{\theta}_1 = \text{sgn}(15 - 5)60 = 60$$

$$t_1 = 2 - \sqrt{4 - \frac{20}{60}} = 2 - \sqrt{\frac{11}{3}}$$

$$\dot{\theta}_{12} = \frac{10}{2 - \frac{1}{2}\left(2 - \sqrt{\frac{11}{3}}\right)} = \frac{20}{2 + \sqrt{\frac{11}{3}}}$$

From (7.28):

$$\ddot{\theta}_3 = \text{sgn}(15 - (-10))60 = 60$$

$$t_3 = 2 - \sqrt{4 + \frac{-50}{60}} = 2 - \sqrt{\frac{19}{6}}$$

$$\dot{\theta}_{23} = \frac{-25}{2 - \frac{1}{2}\left(2 - \frac{\sqrt{19}}{6}\right)} = \frac{-50}{2 + \sqrt{\frac{19}{6}}}$$

Finally, from (7.24):

$$t_2 = \frac{\dot{\theta}_{23} - \dot{\theta}_{12}}{-60}$$

7.11) ${}^{s}X_G = [\,10 \quad 20 \quad 30 \quad 0 \quad 0 \quad 30\,]^T$

7.12) ${}^{s}T_G = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & .707 & -.707 & -20 \\ 0 & .707 & .707 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

7.13) Programming exercise. Note: assume same trajectory for each joint.

7.14) Programming exercise. Note: assume same trajectory for each joint.

7.15) let $\dot{\theta}_v = \dfrac{1}{2(t_{f1}+t_{f2})}\left[3(\theta_v-\theta_o)\dfrac{t_{f2}}{t_{f1}} + 3(\theta_G-\theta_v)\dfrac{t_{f1}}{t_{f2}}\right]$

Then:

$a_{20} = \theta_v$

$a_{21} = \dot{\theta}_v$

$a_{22} = \dfrac{3}{t_{f2}^2}(\theta_G-\theta_v) - \dfrac{2}{t_{f2}}\dot{\theta}_v$

$a_{23} = \dfrac{-2}{t_{f2}^3}(\theta_G-\theta_v) + \dfrac{1}{t_{f2}^2}\dot{\theta}_v$

$a_{10} = \theta_o$

$a_{11} = 0$

$a_{12} = \dfrac{3}{t_{f1}^3}(\theta_v-\theta_o) + \dfrac{1}{t_{f1}}\dot{\theta}_v$

$a_{13} = \dfrac{-2}{t_{f1}^3}(\theta_v-\theta_o) + \dfrac{1}{t_f^2}\dot{\theta}_v$

7.16) $t_f > \max\left[\dfrac{3}{2\dot{\theta}_{\max}}|\theta_f-\theta_o|, \sqrt{\dfrac{6}{\ddot{\theta}_{\max}}|\theta_f-\theta_o|}\right]$

Then, cubic coefficients given by (7.11).

7.17) By differentiation:

$\dot{\theta}(t) = 180t - 180t^2$

$\ddot{\theta}(t) = 180 - 360t$

Then, evaluating at $t = 0$ and $t = 1$, we have:

$\theta(0) = 10 \quad \dot{\theta}(0) = 0 \quad \ddot{\theta}(0) = 180$

$\theta(1) = 40 \quad \dot{\theta}(1) = 0 \quad \ddot{\theta}(1) = -180$

7.18) Using same equations as for (7.17), and evaluating at $t = 2$.

$\theta(2) = -110 \quad \dot{\theta}(2) = -360 \quad \ddot{\theta}(2) = -540$

7.19) So,

$\dot{\theta}(t) = 5 + 140t - 135t^2$

$\ddot{\theta}(t) = 140 - 270t$

$\theta(0) = 10 \quad \dot{\theta}(0) = 5 \quad \ddot{\theta}(0) = 140$

$\theta(1) = 40 \quad \dot{\theta}(1) = 10 \quad \ddot{\theta}(1) = -130$

7.20) Using equations from 7.19, and evaluating at $t = 2$:

$\theta(2) = -60 \quad \dot{\theta}(2) = -255 \quad \ddot{\theta}(2) = -400$

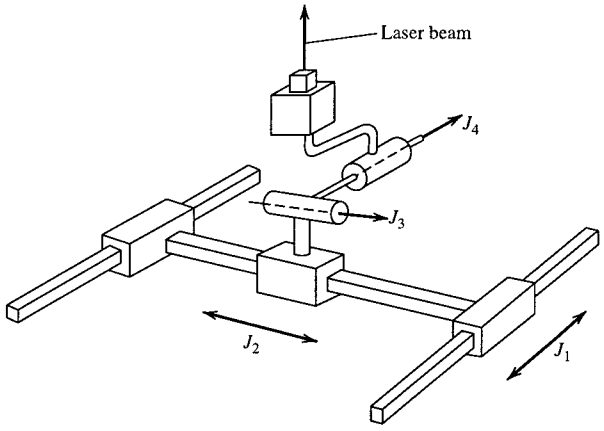
Chapter 8

Manipulator Mechanism Design

Exercises

8.1) The laser beam can be considered a line in space. In general, it takes 4 D.O.F.S to position a line in space. So, the answer is 4. Note that this is somewhat "theoretical" in that there is probably no single 4-DOF design that could perform all such cutting tasks — however, see problem 8.2 below for one example.

8.2)



8.3) Using (8.1) we have:

$$L = \sum_{i=1}^{3}(a_{i-1} + d_i)$$

$$= (0+0) + (0+0) + (0 + (U - L)) = U - L$$

$$W = \frac{4}{3}\pi U^3 - \frac{4}{3}\pi L^3 = \frac{4}{3}\pi(U^3 - L^3) \left\{ \begin{array}{c} \text{A} \\ \text{"Hollow"} \\ \text{Sphere} \end{array} \right.$$

$$\therefore \quad Q_L = \frac{U - L}{\sqrt[3]{\frac{4}{3}\pi(U^3 - L^3)}}$$

8.4) Let $K_{0.2}$ be stiffness of the 0.2 cm diameter shaft, and $k_{0.3}$ be that of the 0.3 cm shaft. Then, using (8.14) and (8.19):

$$\frac{1}{K_{\text{total}}} = \frac{1}{K_{0.2}n^2} + \frac{1}{K_{0.3}} \quad \text{where } n = 8$$

From (8.15):

$$K_{0.2} = \frac{(7.5 \times 10^{10})(\pi)(0.002)^4}{(32)(0.30)} = 0.393 \text{ ntm/rad}$$

$$K_{0.3} = \frac{(7.5 \times 10^{10})(\pi)(0.003)^4}{(32)(0.30)} = 1.988 \text{ ntm/rad}$$

$$\frac{1}{K_{\text{total}}} = \frac{1}{(.393)8^2} + \frac{1}{1.988}$$

$$= \frac{1}{25.152} + \frac{1}{1.988} = 0.5427$$

$\therefore$ $\boxed{K_{\text{total}} = 1.8426 \text{ ntm/rad}}$

8.5) Torsional load due to acceleration of the center of mass about the joint axis is:

$\tau = (0.30)(10)(2)(9.8)$ ntm $= 58.8$ ntm

A torsional spring is given by $\tau = K\Delta\theta$, so needed stiffness is

$$K = \frac{\tau}{\Delta\theta} = \frac{58.8}{0.1} = 588 \text{ ntm/rad}$$

From (8.15), solving for d gives:

$$d = \sqrt[4]{\frac{32LK}{G\pi}}$$

$$d = \sqrt[4]{\frac{(32)(0.30)(588)}{(7.5 \times 10^{10})(\pi)}} = 0.01244 \text{ m}$$

or $\boxed{d = 1.244 \text{ cm}}$

8.6) From (8.14):

$$\frac{1}{K_{\text{total}}} = \frac{1}{1000} + \frac{1}{300} = 4.333 \times 10^{-3}$$

$\therefore K_{\text{total}} = 230.77$ ntm/rad

8.7) If 3 of the attachment points on the top *or* bottom plate coincide. For example, in fig. 8.22, if $q_i = q_j = q_k$ for any $i \neq j \neq k$, or if $p_i = p_j = p_k$ for any $i \neq j \neq k$.

8.8) In Grübler's formula (8.9) we have:

$N = 18$ (12 universal joints, 6 prismatic joints)

$L = 14$ (2 parts per activator = 12, top plate, bottom plate)

$\sum_{i=1}^{18} f_i = 12 \times 3 + 6 = 42$

so, $F = 6(14 - 18 - 1) + 42 = \boxed{12}$

8.9) Stiffness of input drive as seen at gear #1:

$$\frac{1}{K_{\text{gear 1}}} = \underbrace{\frac{1}{100}}_{\text{coupling \#1}} + \underbrace{\frac{1}{400}}_{\text{shaft}} + \underbrace{\frac{1}{100}}_{\text{coupling \#2}} = 0.0225; \quad K_{\text{gear 1}} = 44.44$$

Assume that shaft from gear 2 to gear 3 is rigid....

$$\frac{1}{K_{\text{gear 3}}} = \frac{1}{2000} + \frac{1}{6^2(K_{\text{gear 1}})} = 1.125 \times 10^{-3}; \quad K_{\text{gear 3}} = 888.88$$

$$\frac{1}{K_{\text{gear 4}}} = \frac{1}{2000} + \frac{1}{6^2(K_{\text{gear 3}})} = 5.3125 \times 10^{-4};$$

$\boxed{K_{\text{gear 4}} = 1882.35 \text{ ntm/rad}}$

8.10) Error is $\dfrac{2000 - 1882.35}{1882.35} \times 100 = \boxed{6.25\%}$

8.11) See solution to exercise 4.12

8.12) For $i = 1, 2, \ldots 6$

$\boxed{d_i = //{}^B_T T q_i - p_i//}$

8.13) To maximize $W = /L_1 L_2 S_2/$ over all configs means maximizing $W' = L_1 L_2$.

Since $L_1 + L_2 = C$ (C = some constant); $L_2 = C - L_1$
so, $W' = L_1(C - L_1) = CL_1 - L_1^2$

$$\frac{dw'}{dL_1} = c - 2L_1 \quad \frac{d^2w'}{dL_1^2} = -2$$

$\therefore$ Maximum of W' exists when $C - 24 = 0$

or $L_1 = C/2 \quad \therefore \quad \boxed{L_1 = L_2 \text{ is optimal}}$

8.14) Same as exercise 8.13 since 3rd dof is orthogonal to first two.

$\boxed{L_1 = L_2 \text{ is optimal}}$

8.15) LU-decomposition allows one to write the jacobian in the form $J(\theta) = U^T \Omega V$ where u and v are orthogonal matrices. Determinant of a product equals product of determinants, so

DET J = (DETU^T)(DETΩ)(DETV) = 1. DETΩ.1

and det Ω is the product of the eigenvalues. Q.E.D.

8.16) From (8.15)

$$K = \frac{G\pi d^4}{32L} = \frac{(0.33 \times 7.5 \times 10^{10})(\pi)(0.001)^4}{(32)(0.40)}$$

$$= \boxed{0.006135 \text{ ntm/rad}}$$

Vert flimst, because diameter is 1 mm!

8.17) From (8.12) $n = r_2/r_1 = 12/2 = \boxed{6}$

8.18) From the general case of needing 6, subtract 2 because grasping "along" and "about" the cylinder's axis is a free variable. Hence, $\boxed{4}$.

8.19) In Grübler's formula:

$N = 12$ (9 real joints, 3 ball-in-socket contacts)

$L = 11$ (3 links per finger, base, ball)

$\sum_{i=1}^{12} f_i = 9 + 3 \times 3 = 18$

so, $F = 6(11 - 12 - 1) + 18 = -12 + 18 = \boxed{6}$

8.20) In Grüblers formula:

$N = 6$ (3 ball-in-socket, 3 universal)

$L = 5$ (ground, object, 3 links)

$\sum_{i=1}^{N} f_i = 3 \times 3 + 3 \times 2 = 15$

so, $F = 6(5 - 6 - 1) + 15 = -12 + 15 = \boxed{3}$

8.21) Start with

$$F = K_{\text{sum}}(\delta X_1 + \delta X_2)$$

And substitute in expressions for $\delta X_1, \delta X_2$:

$$F = K_{\text{sum}}\left(\frac{F}{K_1} + \frac{F}{K_2}\right)$$

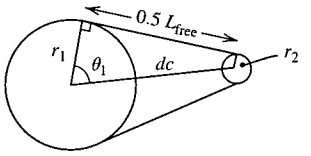
Multiply both sides by $\dfrac{1}{FK_{\text{sum}}}$:

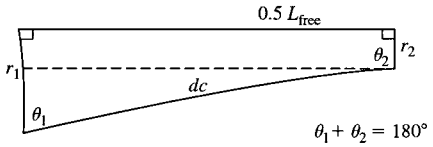
$$\boxed{\frac{1}{K_{\text{sum}}} = \frac{1}{K_1} + \frac{1}{K_2}} \quad \text{Q.E.D.}$$

8.22) From (8.22) we have

$$K = \frac{AE}{L} \quad \text{where} \quad L = L_{\text{free}} + \frac{1}{3}L_{\text{contact}}$$

So the problem is to figure out L_{free} and L_{contact} for the general 2-pulley setup.





$$\therefore \quad 0.5L_{\text{free}} = \sqrt{d_c^2 - \text{ABS}(r_1 - r_2)^2}$$

$$\therefore \quad L_{\text{free}} = 2\sqrt{d_c^2 - (r_1 - r_2)^2}$$

$$L_{\text{contact}} = 2[(\pi - \theta_1)r_1 + \theta_1 r_2]$$

where $\theta_1 = \cos^{-1}\left(\dfrac{r_1}{d_c}\right)$ and $r_1 > r_2$

Chapter 9

Linear Control of Manipulators

Exercises

9.1) In the general solution,

$$x(t) = c_1' e^{s_1 t} + c_2' e^{s_2 t}$$

The constants c_1' and c_2' must be (in general) imaginary since the S_i are and yet $x(t)$ must of course be real. Substituting for the S_i:

$$x(t) = c_1' e^{(\lambda+\mu i)t} + c_2' e^{(\lambda o-\mu i)t}$$

$$x(t) = c_1' e^{(\lambda t} e^{\mu i t} + c_2' e^{\lambda t} e^{-\mu i t}$$

Using Euler's relation (9.13) we have

$$x(t) = c_1' e^{\lambda t}(\cos \mu t + i \sin \mu t) + c_2' e^{\lambda t}(\cos \mu t - i \sin \mu t)$$

$$x(t) = (c_1' + c_2') e^{\lambda t} \cos \mu t + (c_1' - c_2') i e^{\lambda t} \sin \mu t$$

To satisfy the constraint that $x(t)$ be real, we have:

$$c_1' + c_2' = c_1$$

$$c_2' - c_1' = c_2 i$$

where c_1 and c_2 are real. This leads to

$$x(t) = c_1 e^{\lambda t} \cos \mu t + c_2 e^{\lambda t} \sin \mu t \quad \text{Q.E.D.}$$

9.2) From (9.5)

$$S_1 = \frac{-6}{2 \times 2} + \frac{\sqrt{36 - 4 \times 2 \times 4}}{2 \times 2}$$

$$= -1.5 + 0.5 = -1.0$$

$$S_2 = -1.5 - 0.5 = -2.0$$

$\therefore \quad x(t) = c_1 e^{-t} + c_2 e^{-2t}$ and $\dot{x}(t) = -c_1 e^{-t} - 2c_2 e^{-2t}$

At $t = 0 \quad x(0) = 1 = c_1 + c_2 \qquad [1]$

$$\dot{x}(0) = 0 = -c_1 - 2c_2 \quad [2]$$

Adding [1] and [2] gives

$$1 = -c_2$$

so: $c_2 = -1$ and $c_1 = 2$

$\therefore \boxed{x(t) = 2e^{-t} - e^{-2t}}$

9.3) $\boxed{x(t) = 4(1+t)e^{-t}}$

9.4) $\boxed{x(t) = 2e^{-2t}\cos t + 4e^{-2t}\sin t}$

9.5) $\boxed{x(t) = \frac{7}{3}e^{-2t} - \frac{4}{3}e^{-5t}}$

9.6) Minimum when
$c_2 = -1 \Rightarrow L_2^2M_2 - 2L_1L_2M_2 + L_1^2(M_1 + M_2)$

Maximum when
$c_2 = 1 \Rightarrow L_2^2M_2 + 2L_1L_2M_2 + L_1^2(M_1 + M_2)$

$\therefore$ min $= 0.5 - 1 + 1.5 = 1$; max $= 0.5 + 1 + 1.5 = 3$;

$\therefore$ $\boxed{66.67\%}$

9.7) Minimum effective inertia:

$I_{min} + n^2I_m = 1.0 + (400)(0.01) = 5.0$

Maximum effective inertia:

$I_{max} - n^2I_m = 3.0 + (400)(0.01) = 7.0$

Variation as % of maximum is $\frac{2}{7} \times 100 = \boxed{28.57\%}$

9.8) Closed loop system is:

$MX'' + B'X'' + K'X = 0$ (9.38)

where $B' = B + K_v$ and $K' = K + K_p$ and
$B' = 2\sqrt{MK'}$

To achieve critical damping.

Using rule (9.72), if $W_{RES} = 6$ rad/sec then we should design servo to have $W_N = 3$ rad/sec. From (9.20) $W_N = \sqrt{K/M}$ (in our case K')

so,

$3 = \sqrt{K'/M}$; $3 = \sqrt{K'}$; $K' = 9$

$K_p = K' - K = 9 - 5 = 4$ $\boxed{K_p = 4}$

$B' = 2\sqrt{MK'} = 2\sqrt{9} = 6$; $K_v = B' - B = 6 - 4 = 2$

$\therefore$ $\boxed{K_v = 2}$

9.9) Min. effect. Inertia $= 4 + (100)(0.01) = 5$

Max. effect. Inertia $= 5 + (100)(0.01) = 6$

Use lowest resonance, so $W_N \leq (1/2)(8)$, or $W_N \leq 4$ rad/sec to insure never underdamped, design using max effect. Inertia, so use $M = 6$

Since we didn't provide info about any damping or stiffness in the open-loop system, we'll assume its zero. Therefore:

$\boxed{\begin{matrix}\alpha = 6 \\ \beta = 0\end{matrix}}$ Since $\alpha = 6$ we now effectively have a $M = 1$ system:

$K_p = W_N = \sqrt{K_e/M}; \quad 4 = \sqrt{K_e/1}; \quad \boxed{K_p = 16}$

$K_v = 2\sqrt{MK_p} = 2\sqrt{1 \times 16} \equiv 8; \quad \boxed{K_v = 8}$

9.10) Using (8.24) and assuming aluminum:

$$K = \frac{(0.333)(2 \times 10^{11})(0.05^4 - 0.04^4)}{(4)(0.50)} = 123,000.0$$

Using info from figure 9.13, the equivalent mass is $(0.23)(5) = 1.15$ kg

so,

$$W_{\text{RES}} = \sqrt{K/M} = \sqrt{\frac{123,000.0}{1.15}} \cong \boxed{327.04 \text{ rad/sec}}$$

This is very high—so the designer is probably wrong in thinking that this link vibrating represents the lowest unmodeled resonance!

9.11) $W_{\text{RES}} = \sqrt{1000/1} \cong 31.62$ rad/sec $\cong \boxed{5.03 \text{ Hz}}$

9.12) Shaft appears stiffer due to gears:

$K = 500 \times 8^2 = 32000; \quad W_{\text{RES}} = \sqrt{32000/1}$

$= 178.88$ rad/sec $\cong \boxed{28.47 \text{ Hz}}$

9.13) As in problem 9.12, the effective stiffness is $K = 32000$. Now, the effective inertia is

$I = 1 + (0.1)(64) = 7.4$

$\therefore W_{\text{RES}} = \sqrt{32000/7.4} \cong 65.76$ rad/sec $\cong \boxed{10.47 \text{ Hz}}$

9.14) $\boxed{\begin{array}{ll} K_p = 612.5 & \alpha = 6 \\ K_r = 49.5 & \beta = 0 \end{array}}$

9.15) Stiffness of transmission is, as in Exercise 8.4:

$$\frac{1}{K} = \frac{1}{n^2 K_{0.2}} + \frac{1}{K_{0.3}}; \quad K = 1.843 \text{ ntm/rad}$$

$$W_{\max} = \sqrt{\frac{1.843}{1}} \cong 1.357 \text{ rad/sec} \cong \boxed{0.2 \text{ Hz}}$$

$$W_{\min} = \sqrt{\frac{1.843}{4}} \cong 0.679 \text{ rad/sec} \cong \boxed{0.11 \text{ Hz}}$$

Very low because 0.2 and 0.3 cm shaft are floppy!

Chapter 10

Nonlinear Control of Manipulators

Exercises

10.1) Let $\tau = \alpha\tau' + \beta$

$\alpha = 2\sqrt{\theta} + 1$

$\beta = 3\dot{\theta}^2 - \sin\theta$

then: $\tau' = \ddot{\theta}_D + K_W\dot{e} + K_\theta e$

where $e = \theta_D - \theta$

For closed loop stiffness, K_{CL}, use:

$K_\theta = K_{CL} = 10$

$K_W = 2\sqrt{K_{CL}} = 2\sqrt{10}$

10.2) Let $\tau = \alpha\tau' + \beta$

$\alpha = 2\beta \quad \beta = 5\theta\dot{\theta} - 13\dot{\theta}^3 + 5$

and: $\tau' = \ddot{\theta}_D + K_v\dot{e} + K_p e$

where $e = \theta_D - \theta$

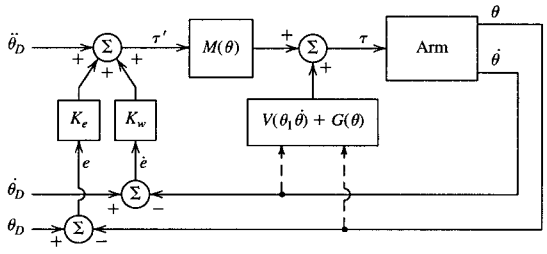
and: $K_p = 10$

$K_v = 2\sqrt{10}$

10.3) Where $M(\theta)$, $V(\theta, \dot{\theta})$, and $G(\theta)$ are as found in section 6.7.

$$K_\theta = \begin{bmatrix} K_{\theta 1} & 0 \\ 0 & K_{\theta 2} \end{bmatrix} \quad K_W = \begin{bmatrix} K_{W1} & 0 \\ 0 & K_{W2} \end{bmatrix}$$

Where $K_{wi} = 2\sqrt{K_{\theta i}}$

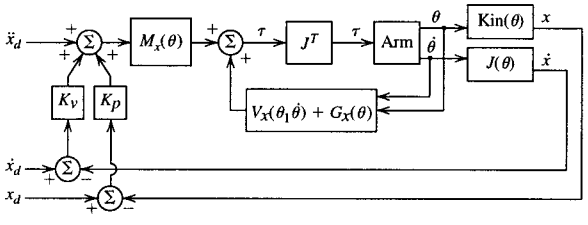


10.4) Where $M_x(\theta)$, $V_x(\theta, \dot{\theta})$, $G_x(\theta)$ are as found in example 6.6. KIN(θ) is the kinematics (forward) for this simple two-link.

Also:

$$K_P = \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \quad K_V = \begin{bmatrix} K_{V1} & 0 \\ 0 & K_{V2} \end{bmatrix}$$

with $K_{Vi} = 2\sqrt{K_{Pi}}$



10.5) $\tau = M\ddot{\theta} + V$

$$M = \begin{bmatrix} M_1 L_1^2 & 0 \\ M_2 L_2^2 & M_2 L_2^2 \end{bmatrix} \quad V = \begin{bmatrix} M_1 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \\ V_2 \dot{\theta}_2 \end{bmatrix}$$

So, as usual:

$$\tau = \alpha\tau' + \beta$$

with $\alpha = M \quad \beta = V$

$$\tau' = \ddot{\theta}_D + K_V \dot{e} + K_P e$$

$$e = \theta_D - \theta$$

and:

$$K_P = \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \quad K_V = \begin{bmatrix} K_{V1} & 0 \\ 0 & K_{V2} \end{bmatrix}$$

with $K_{Vi} = 2\sqrt{K_{Pi}}$

Since the mass matrix, M, is not symmetric this can not represent a coupled mechanical system.

10.6) In steady state, the system error equation is:

$$K_p e = \frac{1}{\hat{M} L^2} (\psi_m L g \cos\theta)$$

so:

$$e = \frac{1}{K_p \hat{M} L^2} (\psi_M L g \cos\theta)$$

$$e = \frac{\psi g \cos\theta}{K_p \hat{M} L}$$

Note that it is a function of $\hat{M}$ as well as the other variables mentioned in problem.

It is maximum when $\theta = 0°$ or $180°$, since this error comes from gravity loading, this is not surprising.

10.7) Not stated in problem, but we will assume than an activator (a gas rocket thruster) can apply a force F_i on each mass. Positive F_i tends to increase X_i. That is:

Drawing free body diagrams leads to equations of motion:

$$\begin{cases} f_1 = M_1 \ddot{X}_1 + (B_1 + B_2)\dot{X}_1 - B_2 \dot{X}_2 + K X_2 - K X_1 \\ F_2 = M_2 \ddot{X}_2 + B_2(\dot{X}_2 - \dot{X}_1) - K(X_2 - X_1) \end{cases}$$

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad Q = \begin{bmatrix} (B_1 + B_2)\dot{X}_1 - B_2 \dot{X}_2 + K(X_2 - X_1) \\ B_2(\dot{X}_2 - \dot{X}_1) - K(X_2 - X_1) \end{bmatrix}$$

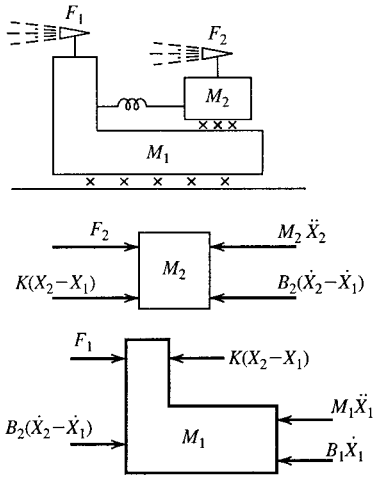
10.7) *(Continued)*

Then let $F = \alpha F' + \beta$ with $\alpha = M, \beta = Q$

$$F' = \ddot{X}_D + K_v \dot{E} + K_p E \begin{cases} F = [f_1 f_2]^T \\ X = [X_1 X_2]^T \\ E = [e_1 e_2]^T \\ e_i = X_{Di} - X_i \end{cases}$$

$$K_v = \begin{bmatrix} K_{v1} & 0 \\ 0 & K_{v2} \end{bmatrix} \quad K_p = \begin{bmatrix} K_{p1} & 0 \\ 0 & K_{p2} \end{bmatrix}$$

with $K_{v_i} = 2\sqrt{K_{p_i}}$



10.8) We would like to say something about

$$\frac{\partial \tau}{\partial \theta} = \begin{bmatrix} \frac{\partial \tau_1}{\partial \theta_1} & \frac{\partial \tau_1}{\partial \theta_2} \\ \frac{\partial \tau_2}{\partial \theta_1} & \frac{\partial \tau_2}{\partial \theta_2} \end{bmatrix} \quad [\text{A}]$$

First, note (from section 6.7) that the only θ_1 dependence is in $G(\theta)$, so we have:

$$\frac{\partial \tau}{\partial \theta} = \begin{bmatrix} \frac{\partial G_1}{\partial \theta_1} & \frac{\partial \tau_1}{\partial \theta_2} \\ \frac{\partial G_2}{\partial \theta_1} & \frac{\partial \tau_2}{\partial \theta_2} \end{bmatrix} \quad [\text{B}]$$

$$\frac{\partial G_1}{\partial \theta_1} = -M_2 L_2 g S_{12} - (M_1 + M_2) L_1 g S_1$$

$$\frac{\partial G_2}{\partial \theta_1} = -M_2 L_2 g S_{12}$$

The second column of [B] above is given by:

$$\frac{\partial (M(\theta)\ddot{\theta})}{\partial \theta_2} + \frac{\partial B(\theta)}{\partial \theta_2}\dot{\theta}_1\dot{\theta}_2 + \frac{\partial (C(\theta)[\dot{\theta}^2])}{\partial \theta_2} + \frac{\partial G(\theta)}{\partial \theta_2}$$

which is

$$\frac{\partial \tau_1}{\partial \theta_2} = -L_1 L_2 M_2 S_2 (2\ddot{\theta}_1 + \ddot{\theta}_2) - 2M_2 L_1 L_2 C_2 \dot{\theta}_1 \dot{\theta}_2$$

$$- M_2 L_1 L_2 C_2 \dot{\theta}_2^2 - M_2 L_2 g S_{12}$$

$$\frac{\partial \tau_2}{\partial \theta_2} = -L_1 L_2 M_2 S_2 \ddot{\theta}_1 + M_2 L_1 L_2 C_2 \dot{\theta}_1^2 - M_2 L_2 g S_{12}$$

10.8) *(Continued)*

To say more we need to make some assumptions: Lets say average velocity is $\dot{\theta}_A$, and lets also assume that average acceleration $\ddot{\theta}_A = \dot{\theta}_A^2$. For many robots this is about right (units are rads). At this speed $\delta\theta = \dot{\theta}_A \Delta t$, where Δt is update interval of configuration-dependent terms. We then have:

$$|\delta\tau_1| = \Delta t(\dot{\theta}_A^3 L_1 L_2 M_2(3S_2 + 2C_2) + 2M_2 L_2 g S_{12}\dot{\theta}_A + (M_1 + M_2)L_1 g S_1 \dot{\theta}_A)$$

$$|\delta\tau_2| = \Delta t(\dot{\theta}_A^3 L_1 L_2 M_2(S_2 + C_2) + 2M_2 L_2 g S_{12}\dot{\theta}_A)$$

From here things are quite heuristic, perhaps the maximum value of τ is found for "average" trajectories, and $\delta\tau$ is required to be less than 5% of this max. Then a Δt might bell found based on an expected $\dot{\theta}_A$ and other params.

10.9) Like exercise 10.8. Result will be in terms of average cartesian velocity, rather than average joint velocity. These might be converted using some average norm of the Jacobian.

10.10) Let $f = \alpha f' + \beta$

with $\alpha = 2, \quad \beta\beta = 5X\dot{X} - 12$

and $f' = \ddot{X}_D + K_v\dot{e} + K_p e, \quad e = X_D - X$

$K_p = 20, \quad K_v = 2\sqrt{20}.$

10.11) Closed loop system, given by:

$$M(\theta)\ddot{\theta} + V_M(\theta, \dot{\theta})\dot{\theta} + G(\theta) = -K_p\theta - M(\theta)K_v\dot{\theta} + G(\theta)$$

$V = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} + \frac{1}{2}\theta^T K_p\theta$ (Lyapunov candidate)

$$\dot{V} = \tfrac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} + \dot{\theta}^T M(\theta)\ddot{\theta} + \dot{\theta}^T K_p\theta$$

$$\dot{V} = \tfrac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} + \dot{\theta}^T[-V_M(\theta, \dot{\theta})\dot{\theta} - G(\theta) - K_p\theta - M(\theta)K_v\dot{\theta} + G(\theta)] + \dot{\theta}^T K_p\theta$$

$$\dot{V} = -\dot{\theta}^T M(\theta)K_v\dot{\theta}$$

This is non-positive if $M(\theta)K_v$ is positive definite. This matrix product is positive def. if $K_v = k_v I_N$, where K_v is positive scalar. Q.E.D.

10.12) Completely analogous to exercise 10.11.

10.13) Use

$$V = \frac{1}{2}\dot{\theta}^{\tau} M(\theta)\dot{\theta} + \int_{s_1}^{s_2} M(\theta) K_p \theta \frac{\delta\theta}{\delta 5} d5 > 0$$

To yield:

$$\dot{V} = -\dot{\theta}^{\tau} M(\theta) K_v \dot{\theta} < 0 \text{ if } K_v = K_v I_N,\ K_v > 0$$

10.14) Analogous to exercise 10.13.

10.15) Use

$$V = \tfrac{1}{2}\dot{\theta}^{\tau} M(\theta)\dot{\theta} + \tfrac{1}{2}\theta^{\tau} K_p \theta + P(\theta)$$

where $P(\theta)$ is the potential energy store in mechanism.

$$\dot{v} = -\dot{\theta}^{\tau} G(\theta) - \dot{\theta}^{\tau} K_v \dot{\theta} + \dot{p}(\theta)$$

Now, $G(\theta) = \dfrac{\partial}{\partial\theta} p(\theta)$(from Lagrange)

$\therefore$ can show $\dot{\theta}^{\tau} G(\theta) = \dot{p}(\theta)$

$\therefore\ \dot{V} = -\dot{\theta}^{\tau} K_v \dot{\theta}$

From which stability follows.

Do steady state analysis (set $\dot{\theta} = \ddot{\theta} = 0$) to get

$$K_p\theta + G(\theta) = 0$$

or

$$\theta = -K_p^{-1} G(\theta) \text{ steady state error.}$$

10.17

$$\alpha_p = ax^2\dot{x}$$
$$\beta = b\dot{x}^2 + c\sin x0$$
$$k_v = 2\sqrt{k_p}$$

10.18 $M\ddot{\theta} + B\dot{\theta}^2 + C\dot{\theta} = M[\ddot{\theta}_D + K_v\dot{e} + K_p e] + \sin\theta$

Thats it. It can be re-written many ways.

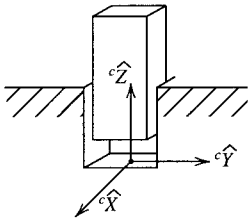
Chapter 11

Force Control of Manipulators

Exercises

11.1) Note this problem asks for "Natural" constraints only.

Natural	Constraints
$V_X = 0$ $V_Y = 0$ $W_X = 0$ $W_Y = 0$ $W_Z = 0$	$F_Z = 0$



11.2) The artificial constraints for the task above would be:

$V_Z = -a_1$	$F_X = 0$ $F_Y = 0$ $N_X = 0$ $N_Y = 0$ $N_Z = 0$

where "a_1" is the speed of insertion.

11.3) $F = MK_e^{-1}[\ddot{F}_d + K_{VF}\dot{e}_F + K_{PF}e_F] + F_d$ (11.14)

$$F = MK_e^{-1}\ddot{F}_e + F_e + F_{\text{dist}} \qquad (11.9)$$

Setting (11.9) and (11.14) equal:

$$MK_e^{-1}[\ddot{F}_d + K_{VF}\dot{e}_F + K_{PF}e_F] + F_d = MK_e^{-1}\ddot{F}_e + F_e + F_{\text{dist}}$$

$$MK_e^{-1}[\ddot{e}_F + K_{VF}\dot{e}_F + K_{PF}e_F] = F_e - F_d + F_{\text{dist}}$$

$$\ddot{e}_F + K_{VF}\dot{e}_F + K_{PF}e_F = M^{-1}K_e(-e_F) + M^{-1}K_eF_{\text{dist}}$$

$$\ddot{e}_F + K_{VF}\dot{e}_F + (K_{PF} + M^{-1}K_e)e_F = M^{-1}K_eF_{\text{dist}}$$

Hence, to damp properly (i.e. to choose K_{VF}) one needs to know K_e (which is generally unknown).

11.4) Use (5.105) with frames $\{A\}$ and $\{B\}$ reversed. First find ${}^B_A T$, so invert ${}^A_B T$:

$$ {}^B_A T = \begin{bmatrix} 0.866 & 0.5 & 0 & -8.66 \\ -0.5 & 0.866 & 0 & 5.0 \\ 0 & 0 & 1 & -5.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now ${}^B F = {}^B_A R \, {}^A F = [\,1 \quad 1.73 \quad -3\,]^T$

$$ {}^B N = {}^B P_{\text{aorg}} \otimes {}^B F + {}^B_A R \, {}^A N = [\,-6.3 \quad -30.9 \quad -15.8\,]^T $$

$$ \therefore {}^B \mathcal{J} = [\,1.0 \quad 1.73 \quad -3 \quad -6.3 \quad -30.9 \quad -15.8\,]^T $$

11.5) Like 11.4.

$$ {}^B_A T = \begin{bmatrix} 0.866 & -0.5 & 0 & -8.66 \\ 0.5 & 0.866 & 0 & -5 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^B F = {}^B_A R \, {}^A F = [\,2.19 \quad 8.19 \quad 0\,]^T $$

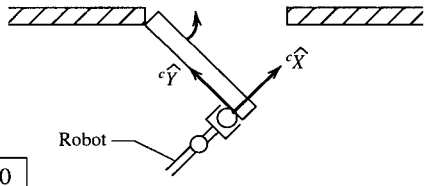
$$ {}^B N = {}^B P_{\text{aorg}} \otimes {}^B F + {}^B_A R \, {}^A N $$

$$ = [\,41 \quad -11 \quad -60.3\,]^T + [\,4.3 \quad 2.5 \quad 0\,]^T $$

$$ \therefore {}^B \mathcal{J} = [\,2.19 \quad 8.19 \quad 0 \quad 44.3 \quad -8.5 \quad -60.3\,]^T $$

11.6) When crowded a good strategy (assuming a fairly stiff book) is to put one corner of the book into the crack and then twist the book towards the vertical position while applying force to push it into the slot. Students should give some version of this strategy.

11.7) Good idea is to attach $\{C\}$ to the door, so it moves as the task progresses.



Then:

Natural

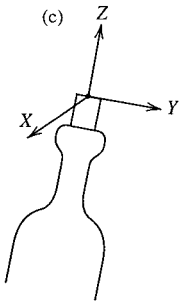
$V_Y = 0$ $V_Z = 0$ $W_X = 0$ $W_Y = 0$	$F_X = 0$ $N_Z = 0$

Artificial

$V_X = a_1$ $W_Z = \dfrac{a_1}{r}$	$F_Y = 0$ $F_Z = 0$ $N_X = 0$ $N_Y = 0$

where "r" is radius of door (that is, distance from hinge to origin of $\{C\}$). "a_1" is linear speed of manipulator hand.

11.8) Pretty easy:



Natural	Constraints
$V_X = 0$ $V_Y = 0$ $W_X = 0$ $W_Y = 0$ $W_Z = 0$	$F_Z = 0$

Here I've assumed that because it is a very tight-fitting cork, it is like a square peg in a square hole, and rotation about $\hat{Z}$ is not possible. This assumption is not crucial, however. Then:

Artificial constraints

$V_Z = a_1$	$F_X = 0$ $F_Y = 0$ $N_X = 0$ $N_Y = 0$ $N_Z = 0$

11.9) Use control law:

$$\tau = \ddot{\theta}_D + J^T K_{PX} J E + K_V \dot{E} \quad [A]$$

where $E = \theta_D - \theta$, and the plant has already been decoupled, that is, the plant is: $\tau = \ddot{\theta}$ [B]

Equate [A] and [B] to get:

$$\ddot{E} + K_V \dot{E} + J^T K_{PX} J E = 0$$

Choose the Lyapunov equation:

$$L = E^T J^T K_{PX} J E + \dot{E}^T \dot{E}$$

This is positive definite in both $E \& \dot{E}$ (note $J^T K_{PX} J$ is pos. def. If K_{PX} is pos. def. Since $J^T K_{PX} J$ is a "congruence" transform which cannot change the sign of eigenvalues — see noble's book on linear algebra, or others.) Differentiating:

$$\dot{L} = -2\dot{E}^T K_V \dot{E} \left(\begin{array}{c} \text{Here we needed to} \\ \text{use approx. that} \\ J \cong 0 \end{array} \right)$$

Which is negative definite iff K_V is positive definite. A few more things would need to be said to make the proof rigorous, but this is the general idea.

11.10) This problem is really beyond the scope of the book, as it involves multivariable control theory. The system studied is a multi-input, multi-output system which is coupled, so the first problem is "what does "critical damping" mean in this context. Let me just mention a damping scheme which has given reasonable results in lab experiments: In the same way that a scalar system with stiffness k_p is damped with the value $2\sqrt{k_p}$, we have used $J^T 2\sqrt{K_{PX}} J$ to damp the system with stiffness of $J^T K_{PX} J$. Note $\sqrt{K_{PX}}$ Just has the square roots of the diagonal elements of K_{PX} on the diagonal.

11.11)

Natural	Constraints
$V_X = 0$	$F_Y = 0$
$V_Z = 0$	
$W_X = 0$	
$W_Y = 0$	
$W_Z = 0$	

Chapter 12

Robot Programming Languages and Systems

In this chapter the exercises are of a programming nature. I have left the choice of which robot programming language to use optional (with 8 or so suggestions). Thus it is hard to give solutions, and I have elected not to do so. Instructors will probably choose to assign problems in the language of some available robot on campus (if any) and the problem may use set-ups or fixtures local to that lab. Hence, I expect a lot of deviation from the suggested problems in the text.

Chapter 13

Off-line Programming Systems

Exercises

13.1) Collision detection (in this context) refers to an algorithm capable of determining if two or more object (spatially modelled) touch or intersect each other. Collision avoidance is the (difficult) problem of computing a path along which an object (or manipulator) is to move such that no collisions occur. Collision-free path planning is a more descriptive term for collision avoidance.

13.2) A world model is a set of data stored in computer memory which describes some aspects of the manipulator or its environment. Path planning emulation is a term used to describe an algorithm in a simulation system which attempts to model (or emulate) the path-planning algorithm of the robot controller. Dynamic emulation is a synonym for dynamic simulation — but implies that the dynamic simulation is attempting to match some physical system.

13.3) Automatic robot placement is the result of an algorithm which can determine the relative placement between a robot and a workpiece so as to achieve certain objectives — kinematic reachability, singularity avoidance, collision avoidance, etc. Time optimal paths are those which get from A to B in minimum time. Error propagation analysis is an algorithm to compute estimates of certainty of knowledge contained in a world model.

13.4) Wireframe graphics depict polyhedra by rendering all of the edles of the polyhedra. In shaded surface display, all the facets of polyhedra are projected onto the screen as 2-D polygons and then "filled" — i.e. all the pixels within those polygons are illuminated. Often, polygon sorting or A 2-buffer technique is used to perform hidden line elimination. Also, hidden line elimination can be performed with edle-based rendering by the use of some fancy algorithms.

13.5) RPL — *R*obot *P*rogramming *L*anguages are more-or-less conventional computer languages with motion statements and other robot-specific commands added. TLP — *T*ask *L*evel *P*rogramming refers to languages or systems in which it is possible to program at a high level, such as "pick up bolt # B-127".
OLP — *O*ff-*L*ine *P*rogramming is defined in the first sentence of Chapter 13!

13.6) Calibration (in the context of OLP systems) refers to the task of bringing the simulated workcell into close correspondence with the actual cell. Coordinated motion is when two or more devices must move together in kinematically synchronized motion. Automatic scheduling is an algorithm that can automatically divide tasks in an automated setting up between multiple machines.

13.7) This is changing each year, so check the latest news. In the late 80's and into the early 90's graphic power is probably increasing by a factor of 2 to 5 each year!

13.8) Placing components on a pc board, spot-welding an automobile, arc-welding a part, painting an automobile, deburring a casting.

13.9) The more "intelligent" a programming system is, the larger the world model required (roughly speaking!). However, the larger the world model, the more ways it can be inaccurate.

Solutions to the Programming Exercises (Parts 2-7, 9-11)

Programming Exercise (Part 2)

```
PROGRAM PP2 (INPUT,OUTPUT);

CONST PI=3.14159;
      HALFPI=1.5707;
      DTOR=0.017453;
      RTOD=57.2957;
TYPE
      FRAME=ARRAY[1..2,1..3] OF REAL;
      VEC3=ARRAY [1..3] OF REAL;
VAR
      UARELU,UARELB,UURELC,UCRELB:VEC3;
      I.J:INTEGER;
      IARELU,IARELB,IURELC,IURELA:FRAME;
      IURELB,ICRELU,ICRELB:FRAME;

FUNCTION AT2 (A,B:REAL): REAL;
VAR ANS:REAL;
  BEGIN
    IF B=0.0 THEN
      BEGIN
        IF A=0.0 THEN ANS:=0.0
        ELSE
        IF A >0.0 THEN ANS:=HALFPI
        ELSE ANS:=-HALFPI;
      END
    ELSE
      BEGIN
        ANS:=ARCTAN (A/B);
        IF B<0.0 THEN
       IF A<0.0 THEN ANS:=ANS-PI ELSE ANS:=ANS+PI;
     END;
     AT2:=ANS;
  END;

PROCEDURE TMULT (VAR BRELA,CRELB,CRELA:FRAME);

VAR I,J,K: INTEGER;
BEGIN
  FOR I:=1 TO 2 DO
  BEGIN
    CRELA[I,3]:=BRELA[I,3];
    FOR J:=1 TO 2 DO
    BEGIN
      CRELA[I,3]:=CRELA[I,3]+BRELA[I,J]*CRELB[J,3];
      CRELA[I,J]:=0.0;
      FOR K:=1 TO 2 DO
        CRELA[I,J]:=CRELA[I,J]+BRELA[I,K]*CRELB[K,J];
    END;
  END;
END;

PROCEDURE TINVERT (VAR BRELA,ARELB:FRAME);

VAR I,J:INTEGER;

BEGIN
ARELB[1,1]:=BRELA[1,1];
ARELB[1,2]:=BRELA[2,1];
ARELB[2,1]:=BRELA[1,2];
```

```
ARELB[2,2]:=BRELA[2,2];
FOR I: =1 TO 2 DO

BEGIN
  ARELB[I,3]:=0.0;
  FOR J:=1 TO 2 DO
    ARELB[I,3]:=ARELB[I,3]-BRELA[J,I]*BRELA[J,3];
END;
END;

PROCEDURE UTOI (VAR UFORM:VEC3; VAR IFORM:FRAME):

BEGIN
  IFORM[1,3]:=UFORM[1];
  IFORM[2,3]:=UFORM[2];
  IFORM[1,1]:=COS (UFORM[3]*DTOR);
  IFORM[1,2]:=-SIN (UFORM[3]*DTOR);
  IFORM[2,1]:=-IFORM[1,2];
  IFORM[2,2]:=IFORM[1,1];
END;

PROCEDURE ITOU (VAR IFORM:FRAME; VAR UFORM: VEC3);

BEGIN
UFORM[1]:=IFORM[1,3];
UFORM[2]:=IFORM[2,3];
UFORM[3]:=AT2 (IFORM[2,1],IFORM[1,1])*RTOD;
END;

BEGIN {MAINPGM.}
  WRITE ('ENTER ELEMENTS OF ARELU,ARELB,AND URELC' );
  FOR I:=1 TO 3 DO
    READ(UARELU[I]);
  FOR I:=1 TO 3 DO
    READ (UARELB[I]);
  FOR I:=1 TO 3 DO
    READ(UURELC[I]);
  UTOI(UARELU,IARELU);
  UTOI(UARELB,IARELB);
  UTOI(UURELC,IURELC);
   TINVERT (IARELU,IURELA);
   TINVERT (IURELC,ICRELU);
   TMULT (IARELB,IURELA,IURELB);
   TMULT (IURELB,ICRELU,ICRELB);
     WRITELN ('C REL B,INTERNAL FORM,ROW-WISE ');
     FOR I:=1 TO 2 DO
       BEGIN
         FOR J:=1 TO 3 DO
           WRITE (ICRELB[I,J]:8:3);
           WRITELN;

       END;
  ITOU(ICRELB,UCRELB);
    WRITELN('CRELB,USER FORM');
    FOR I:=1 TO 3 DO
       WRITE(UCRELB[I]:8:3);
END.
```

Programming Exercise (Part 3)

```
program pp3 (INPUT,OUTPUT);

CONST

  pi=3.14159;
  halfpi=1.5707;
```

```
  dtor=0.017453292; { Conversion factors: rad/deg }
  rtod=57.29578;    {                     deg/rad }

TYPE

  frame=ARRAY[1..2,1..3] OF REAL;  { 2× 2 Rot. plus 2× 1 Pos. }
  vec3=ARRAY [1..3] OF REAL;

VAR

  i: integer;
  theta: vec3;          { Joint angles for the 3-link arm }
  wrelb: frame;         { Loc. of the wrist wrt the base }
  11,12: REAL;          { Link lengths of the arm }
  trelw: frame;         { Defines the tool location }
  srelb,brels: frame;   { Arm base location }
  trels: frame;
  trelwuser: vec3;
  srelbuser: vec3;
  trelsuser: vec3;
  ans: char;

PROCEDURE WRITEFRAME (VAR foo: frame);
VAR i,j: integer;
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    WRITE ('[');
    FOR j:=1 TO 3 DO
      WRITE (foo[i,j]: 10:3);
    WRITELN(']');
  END;
END;

PROCEDURE WRITEVECT (VAR foo; vec3);
VAR i: integer;
BEGIN
  FOR i: = 1 TO 3 do
    WRITELN('[',foo[i]:10:3,']');
END;

FUNCTION AT2 (a,b: REAL): REAL; { 4 Quadrant Arc Tangent }
VAR ans: REAL;
BEGIN
  IF b= 0.0 THEN
  BEGIN
    IF a=0.0 THEN ans:= 0.0
    ELSE
    IF a>0.0 THEN ans:=halfpi
    ELSE
    ans:=-halfpi;
  END
  ELSE
  BEGIN
    ans:=arctan (a/b);  { Pascal defined arctan }
    IF b< 0 THEN
    IF a< 0 THEN ans:=ans-pi ELSE ans:=ans+pi;
  END;
  at2:=ans;
END;  { atan2 }
PROCEDURE TMULT (VAR brela,crelb,crela: frame);
VAR i,j,k: INTEGER;
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
```

```
    crela[i,3]:=brela[i,3];
    FOR j:=1 TO 2 DO
    BEGIN
      crela[i,3]:=crela[i,3]+brela[i,j]*crelb[j,3];
      crela[i,j]:=0.0;
      FOR k:=1 TO 2 DO
        crela[i,j]:=crela[i,j]+brela[i,k]*crelb[k,j];
    END;
  END;
END; { tmult }

PROCEDURE TINVERT (VAR brela,arelb: frame);
VAR i,j: INTEGER;
BEGIN
  arelb[1,1]:=brela[1,1]; { Transpose rotation part }
  arelb[1,2]:=brela[2,1];
  arelb[2,1]:=brela[1,2];
  arelb[2,2]:=brela[2,2];
  FOR i:=1 TO 2 DO        { Calculate position part }
  BEGIN
    arelb[i,3]:=0.0;
    FOR j:=1 TO 2 DO
      arelb[i,3]:=arelb[i,3]-brela[j,i]*brela[j,3];
  END;
END; { tinvert }

PROCEDURE KIN (VAR theta: vec3; VAR wrelb: frame);
VAR beta: real;
BEGIN
  wrelb[1,3]:=l1*cos (theta[1]) + l2*cos (theta[1] + theta[2]);
  wrelb[2,3]:=l1*sin (theta[1]) + l2*sin (theta[1] + theta[2]);
  beta:=theta[1] + theta[2] + theta[3];      { beta is the total wrist rotation }
  wrelb[1,1]:=cos (beta);
  wrelb[1,2]:=-sin (beta);
  wrelb[2,1]:=-wrelb[1,2];         { note that 2x2 rotation matrices are anti- }
  wrelb[2,2]:=wrelb[1,1];          { symmetric so that half of the matrix is }
                                   { already computed }
END:

PROCEDURE UTOI(VAR uform: vec3; VAR iform: frame);
BEGIN
  iform[1,3]:=uform[1];
  iform[2,3]:=uform[2];
  iform[1,1]:=cos (uform[3]*dtor);
  iform[1,2]:=-sin (uform[3]*dtor);
  iform[2,1]:=-iform[1,2];       { Again, the 2x2 rotation matrix is anti-}
  iform[2,2]:=iform[1,1];        { symmetric, simplifying the calculations }
END;

PROCEDURE ITOU(VAR iform: frame; VAR uform: vec3);
BEGIN
  uform[1]:=iform[1,3];
  uform[2]:=iform[2,3];
  uform[3]:=at2(iform[2,1], iform[1,1])*rtod; { don't forget to use degrees }
END;

PROCEDURE WHERE (VAR theta: vec3; VAR trels: frame); { Find the tool wrt station }
VAR trelb,wrelb: frame;
BEGIN
  KIN(theta,wrelb);
  TMULT(wrelb,trelw,trelb); { Get the tool relative to the base }
  TMULT(brels,trelb,trels); { Get the tool relative to the station }
END;
```

```
PROCEDURE INITDATA;   { Set up the given data }
BEGIN
  l1:=0.5;            { initialize link lengths }
  l2:=0.5;
  trelwuser[1];=0.1;  { initialize tool location in user-coordinates }
  trelwuser[2]:=0.2;
  trelwuser[3];=30.0;
  srelbuser[1]:=-0.1; { initialize station location in user-coordinates }
  srelbuser[2]:=0.3;
  srelbuser[3]:=0.0;
  UTOI(srelbuser,srelb);       { convert base description to internal form }
  TINVERT(srelb,brels);        { get the description in the proper direction }
  UTOI(trelwuser,trelw);       { Get the tool location in internal form }
END;  { initdata }

BEGIN { Main Test Program }
  INITDATA;
  WRITE('Care to do some robot kinematics?');
  READLN(ans);
  WHILE ans = 'y' DO
  BEGIN
    WRITELN('Enter theta1,theta2,theta3:');
    READLN(theta[1],theta[2],theta[3]);  { All I/O with user is in Degrees }
    FOR i:=1 TO 3 DO
      theta[i]:=theta[i]*dtor; { But internally I'll use radians }
    WHERE(theta,trels);
    WRITELN('The frame description of the tool relative to the base is:');
    WRITEFRAME(trels);
    WRITELN;
    ITOU(trels,trelsuser);
    WRITELN('The user-coordinates are:');
    WRITEVECT(trelsuser);
    WRITELN:
    WRITELN;
    WRITE('Would you like to do some more?');
    READLN(ans);
  END;
END.
```

Programming Exercise (Part 4)

```
program pp4 (INPUT,OUTPUT);

CONST
  pi=3.14159;
  halfpi=1.5707;
  dtor=0.0174532925;
  rtod=57.29578;

TYPE
  vec3=ARRAY[1..3] OF REAL;
  frame=ARRAY[1..2,1..3] OF REAL;

VAR
  i: INTEGER;
  11,12: REAL;  { Link Lengths }
  current, goal: vec3;
  trelwuser: vec3;
  srelbuser: vec3;
  srelb,trelw,trels: frame;
  near, far: vec3;
  neard, fard: vec3;  { solutions in degrees for user }
  sol: BOOLEAN;
  ans: char;
```

```
FUNCTION at2(a,b: REAL): REAL;  { Atan2 }
VAR ans: REAL;
BEGIN
  IF b=0.0 THEN
  BEGIN
    IF a=0.0 THEN ans:=0.0
    ELSE
    IF a > 0.0 THEN ans:=halfpi
    ELSE
    ans:=-halfpi;
  END
  ELSE
  BEGIN
    ans:=arctan(a/b);  { Pascal defined arctan }
    IF b<0 THEN
    IF a<0 THEN ans:=ans-pi ELSE ans:=ans+pi;
  END;
  at2;=ans;
END;  { atan2 }

PROCEDURE WRITEFRAME(VAR foo: frame);
VAR i,j: integer;
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    WRITE('[');
    FOR j:=1 TO 3 DO
      WRITE(foo[i,j]:10:3);
    WRITELN(']');
  END;
END;

PROCEDURE WRITEVECT(VAR foo: vec3);
VAR i: integer;
BEGIN
  FOR i:=1 TO 3 do
    WRITELN('[',foo[i]:10:3,']');
END;

PROCEDURE ITOU(VAR iform: frame; VAR uform: vec3);
BEGIN
  uform[1]:=iform[1,3];
  uform[2]:=iform[2,3];
  uform[3]:=at2(iform[2,1],iform[1,1])*rtod;
END;

PROCEDURE UTOI(VAR uform: vec3; VAR iform: frame);
BEGIN
  iform[1,3]:=uform[1];
  iform[2,3]:=uform[2];
  iform[1,1]:=cos (uform[3]*dtor);
  iform[1,2]:=-sin (uform[3]*dtor);
  iform[2,1]:=-iform[1,2];
  iform[2,2]:=iform[1,1];
END;

PROCEDURE TMULT(VAR brela,crelb,crela: frame);
VAR i,j,k: INTEGER;
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    crela[i,3]:=brela[i,3];
    FOR j:=1 TO 2 DO
    BEGIN
      crela[i,3]:=crela[i,3] + brela[i,j]* crelb[j,3];
```

```
      crela[i,j]:=0.0;
      FOR k:=1 TO 2 DO
        crela[i,j]:=crela[i,j]+brela[i,k]*crelb[k,j];
    END;
  END;
END;  { tmult }

PROCEDURE TINVERT(VAR brela,arelb: frame);
VAR i,j: INTEGER;
BEGIN
  arelb[1,1]:=brela[1,1];
  arelb[1,2]:=brela[2,1];
  arelb[2,1]:=brela[1,2];
  arelb[2,2]:=brela[2,2];
  FOR i: = 1 TO 2 DO
  BEGIN
    arelb[i,3]:=0.0;
    FOR j:=1 TO 2 DO
     arelb[i,3]:=arelb[i,3]-brela[j,i]* brela[j,3];
  END;
END;  { tinvert }

PROCEDURE KIN(VAR theta: vec3; VAR wrelb: frame);
VAR beta: real;
BEGIN
  wrelb[1,3]:=l1* cos (theta[1])+l2* cos (theta[1] + theta[2]);
  wrelb[2,3]:=l1* sin (theta[1])+l2* sin (theta[1] + theta[2]);
  beta:=theta[1] + theta[2] + theta[3];
  wrelb[1,1]:=cos (beta);
  wrelb[1,2]:=-sin (beta);
  wrelb[2,1]:=-wrelb[1,2];
  wrelb[2,2]:=wrelb[1,1];
END;

PROCEDURE WHERE(VAR theta: vec3; VAR trels: frame);
VAR trelb,wrelb,brels: frame;
BEGIN
  KIN(theta,wrelb);
  TMULT(wrelb,trelw,trelb);  { Get the tool rel base }
  tinvert(srelb,brels);
  TMULT(brels,trelb,trels);  { Get the tool rel station }
END;  { where }

{ Define a distance measure for two joint vectors }
function dist(VAR a,b: vec3): real;
VAR d,t: REAL;
    i: INTEGER;
BEGIN
  d:=0.0;
  FOR i:=1 TO 3 DO
  BEGIN
   t:=abs(a[i]-b[i]);
   d:=d+t;
  END;
  dist:=d;
END; { dist }

PROCEDURE range(VAR a: REAL);  { puts angle in [-180 180] }
BEGIN
  WHILE a>pi DO a:=a-2* pi;
  WHILE a< - pi DO a:=a+ 2* pi;
END;  { range }
```

```
PROCEDURE invkin(VAR wrelb:frame;
                 VAR current,near,far:vec3; VAR sol:boolean);
LABEL 5;
VAR c2,s2,k1,k2,temp: REAL;
    i : INTEGER;
    goal, swap: vec3;
    solnear, solfar:BOOLEAN;
BEGIN
  itou(wrelb, goal);  { get x,y,and theta of goal point }
  goal[3]:=goal[3]* dtor;  { but put theta back into radians }
  c2:=(goal[1]* goal[1]+goal[2]* goal[2]-l1* l1- l2* l2)/(2.0* l1* l2);
  IF abs(c2)>1.0 THEN sol:=FALSE ELSE sol:=TRUE;
  IF NOT sol THEN GOTO 5;  { No sol. - just exit with flag set }
  s2:=sqrt(1.0 - c2* c2);
  near[2]:=at2(s2,c2);  { Assume Near - swap later if Far }
  far[2]:=-near[2];
  k1:=l1 + l2* c2;
  k2:=l2* s2;
  temp:=at2(k2,k1);  { temp, so we don't calc. it twice }
  near[1]:=at2(goal[2],goal[1])-temp;
  far[1]:=at2(goal[2],goal[1])+temp;
  near[3]:=goal[3]-near[1]-near[2];
  far[3]:=goal[3] - far[1] - far[2];
  FOR i:=1 TO 3 DO  { Return all angles on range [-180,180] }
  BEGIN
    range(near[i]) ;
    range(far[i]);
  END;
  IF dist(current, near)>dist(current, far)THEN  { swap }
  BEGIN
    swap:=near;
    near:=far;
    far:=swap;
 END;
 solnear:=TRUE;
 FOR i:1 TO 3 DO
   BEGIN
     IF (ABS(near[i]) >2.96) THEN
       solnear:=FALSE;
   END;
solfar:=TRUE;
FOR i:=1 TO 3 DO
  BEGIN
    IF (ABS(far[i]) >2.96) THEN
      solfar:=FALSE ;
  END;
IF ((NOT solnear) AND (NOT solfar)) THEN
  sol:=FALSE ;
IF ( solnear AND (NOT solfar )) THEN
  far:= near;
IF (solfar AND (NOT solnear)) THEN
  near:=far ;
5:
END;  { invkin }

PROCEDURE Solve(VAR trels:frame;
                VAR current,near,far:vec3; VAR sol:boolean);
VAR trelb,wrelt,wrelb: frame;
BEGIN
  tmult(srelb,trels,trelb);
  tinvert(trelw,wrelt);
  tmult(trelb,wrelt,wrelb);
  invkin(wrelb,current,near,far,sol);
END;  { Solve }
```

```
PROCEDURE INITDATA;  { Set up the given data }
BEGIN
  11:=0.5;           { initialize link lengths }
  12:=0.5;
  trelwuser[1]:=0.1;
  trelwuser[2]:=0.2;
  trelwuser[3]:=30.0;
  srelbuser[1]:=-0.1;
  srelbuser[2]:=0.3;
  srelbuser[3]:=0.0;
  UTOI(srelbuser,srelb);
  UTOI(trelwuser,trelw);
END;  { initdata }

BEGIN            { Main test program }
   initdata:     { initialize TOOL and STATION definitions }
   FOR i:=1 TO 3 DO current[i]:=0.0;
   WRITE('Care to do some robot inverse kinematics? ');
   READLN(ans);
   WHILE ans='y' DO
   BEGIN
    WRITELN('Enter GOAL: x,y,phi: ');
    READLN(goal[1],goal[2],goal[3]); { Degrees }
    utoi(goal, trels);
    Solve(trels,current,near,far,sol);
    IF NOT sol THEN WRITELN('No solution!');
    current:=near;
    WRITELN('The Near solution is:');
    FOR i:=1 to 3 do neard[i]:=near[i]*rtod;
    WRITEVECT(neard);
    WRITELN;
    WRITELN('The Far solution is:');
    FOR i:=1 to 3 do fard[i]:=far[i]*rtod;
    WRITEVECT(fard);
    WRITELN;            { Check if inverse of Where }
    where(near, trels);
    itou(trels, goal);   { into user representation }
    WRITELN('Calling Where(near) to check. ...');
    WRITELN('Goal is: ');
    writevect(goal);
    WRITELN;
    where(far,trels);
    itou(trels, goal);   { into user representation }
    WRITELN('Calling Where(far) to check. ...');
    WRITELN('Goal is: ');
    writevect(goal);
    WRITELN;
    WRITE('Would you like to do some more? ');
    READLN(ans);
  END;
END.
```

Programming Exercise (Part 5)

```
program pp5 (INPUT,OUTPUT);

CONST
  pi=3.14159;
  halfpi=1.5707;
  dtor=0.0174532925;
  rtod=57.29578;
```

```
TYPE
  frame = ARRAY[1..2,1..3] OF REAL;
  vec3 = ARRAY[1..3] OF REAL;
  mat33 = ARRAY[1..3,1..3] OF REAL;

VAR
  i: integer;
  theta: vec3;
  brelw,wrelb: frame;
  l1,l2: REAL;
  wrelt,trelw: frame;
  srelb,brels: frame;
  srelt,trels: frame;
  trelwuser: vec3;
  srelbuser: vec3;
  trelsuser: vec3;
  current,near,far: vec3;
  thetadot: vec3;
  JACnear,JACfar: mat33;
  vrelt,vrelw: vec3;
  sol: boolean;
  ans: char;

PROCEDURE WRITEFRAME (VAR foo: frame);
VAR i,j: integer;
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    WRITE ('[');
    FOR j:=1 TO 3 DO
      WRITE (foo[i,j]: 10 : 3);
    WRITELN (']');
  END;
END;

PROCEDURE WRITEVECT (VAR foo: vec3);
VAR i: integer;
BEGIN
  FOR i:=1 TO 3 do
    WRITELN ('[',foo[i]: 10 : 3,']');
END;

FUNCTION AT2 (a,b: REAL): REAL;  { 4 Quadrant Arc Tangent }
VAR ans: REAL;
BEGIN
  IF b=0.0 THEN
  BEGIN
    IF a=0.0 THEN ans:=0.0
    ELSE
    IF a>0.0 THEN ans:=halfpi
    ELSE
    ans:=- halfpi;
  END
  ELSE
  BEGIN
    ans:=arctan (a/b);  { Pascal defined arctan }
    IF b<0 THEN
    IF a<0 THEN ans:=ans-pi ELSE ans:=ans+pi;
  END;
  at2:=ans;
END;  { atan2  }

PROCEDURE TMULT (VAR brela,crelb,crela: frame);
VAR i,j,k: INTEGER;
```

```
BEGIN
  FOR i:=1 TO 2 DO
  BEGIN
    crela[i,3]:=brela[i,3];
    FOR j:=1 TO 2 DO
    BEGIN
      crela[i,3]:=crela[i,3]+brela[i,j]*crelb[j,3];
      crela[i,j]:= 0.0;
      FOR k:=1 TO 2 DO
        crela[i,j]:=crela[i,j]+brela[i,k]*crelb[k,j];
    END;
  END;
END; { tmult }

PROCEDURE TINVERT (VAR brela,arelb: frame);
VAR i,j: INTEGER;
BEGIN
  arelb[1,1]:=brela[1,1];
  arelb[1,2]:=brela[2,1];
  arelb[2,1]:=brela[1,2];
  arelb[2,2]:=brela[2,2];
  FOR i:=1 TO 2 DO
  BEGIN
    arelb[i,3]:=0.0;
    FOR j:=1 TO 2 DO
      arelb[i,3]:=arelb[i,3]-brela[j,i]*brela[j,3];
  END;
END;  { tinvert }

PROCEDURE KIN (VAR theta: vec3; VAR wrelb: frame);
VAR beta: real;
BEGIN
  wrelb[1,3]:=l1*cos (theta[1])+l2*cos (theta[1]+theta[2]);
  wrelb[2,3]:=l1*sin (theta[1])+l2*sin (theta[1]+theta[2]);
  beta:=theta [1]+theta [2] + theta [3];
  wrelb[1,1]:=cos (beta);
  wrelb[1,2]:=-sin(beta);
  wrelb[2,1]:=-wrelb[1,2];
  wrelb[2,2]:=wrelb[1,1];
END;

PROCEDURE UTOI(VAR uform: vec3; VAR iform: frame);
BEGIN
  iform[1,3]:=uform[1];
  iform[2,3]:=uform[2];
  iform[1,1]:=cos (uform[3]*dtor);
  iform[1,2]:=-sin (uform[3]*dtor);
  iform[2,1]:=-iform[1,2];
  iform[2,2]:=iform[1,1];
END;

PROCEDURE ITOU(VAR iform: frame; VAR uform: vec3);
BEGIN
  uform[1]:=iform[1,3];
  uform[2]:=iform[2,3];
  uform[3]:=at2(iform[2,1],iform[1,1])*rtod;
END;

function dist(VAR a,b: vec3): real;
VAR d,t: REAL;
    i : INTEGER;
BEGIN
  d:=0.0;
  FOR i:=1 TO 3 DO
```

```
  BEGIN
   t:=abs(a[i]-b[i]);
   d:=d+t;
  END;
  dist:=d;
END;  { dist }

PROCEDURE range(VAR a: REAL);
BEGIN
  WHILE a>pi DO a:=a-2*pi;
  WHILE a< -pi DO a:=a+2*pi;
END;  { range }

PROCEDURE invkin(VAR wrelb:frame;
                 VAR current,near,far:vec3; VAR sol:boolean);

LABEL 5;
VAR c2,s2,k1,k2,temp: REAL;
    i: INTEGER;
    goal,swap: vec3;
    solnear,solfar : BOOLEAN;
BEGIN
  itou(wrelb.goal);
  goal[3]:=goal[3]*dtor;
  c2:=(goal[1]*goal[1]+goal[2]*goal[2]-l1*l1-l2*l2)/(2.0*l1*l2);
  IF abs(c2)>1.0 THEN sol:=FALSE ELSE sol:=TRUE;
  IF NOT sol THEN GOTO 5;
  s2:=sqrt(1.0-c2*c2);
  near[2]:=at2 (s2,c2);
  far[2]:=-near[2];
  k1:=l1+l2*c2;
  k2:=l2*s2;
  temp:=at2 (k2,k1);
  near[1]:=at2(goal[2],goal[1])-temp;
  far[1]:=at2(goal[2],goal[1])+temp;
  near[3]:=goal[3]-near[1]-near[2];
  far[3]:=goal[3]-far[1]-far[2];
  FOR i:=1 TO 3 DO
  BEGIN
    range(near[i]);
    range(far[i]);
  END;
  IF dist(current,near)>dist(current,far) THEN
  BEGIN
    swap:=near;
    near:=far;
    far:=swap;
  END;

solnear:=TRUE;
FOR i:=1 TO 3 DO
  BEGIN
    IF (ABS(near[i])>2.96) THEN
       solnear:=FALSE;
  END;

solfar := TRUE;
FOR i:=1 TO 3 DO
   BEGIN
    IF ( ABS(far[i])>2.96 ) THEN
       solfar :=FALSE;
   END ;
IF ((NOT solnear ) AND (NOT solfar ) ) THEN
  sol:=FALSE ;
IF ( solnear AND ( NOT solfar)) THEN
   far:= near;
```

```
IF ( solfar AND ( NOT solnear )) THEN
   near:=far ;

5:
END;  { invkin }

PROCEDURE SOLVE(VAR trels:frame;
                VAR current,near,far:vec3; VAR sol:boolean);
VAR trelb,wrelt,wrelb: frame;
BEGIN
  tmult(srelb,trels,trelb);
  tinvert(trelw,wrelt);
  tmult(trelb,wrelt,wrelb);
  invkin (wrelb,current,near,far,sol);
END;  { Solve }

PROCEDURE WHERE (VAR theta: vec3; VAR trels: frame);
VAR trelb,wrelb: frame;

BEGIN
  KIN (theta,wrelb);
  TMULT (wrelb,trelw,trelb);
  TMULT (brels,trelb,trels);
END;

PROCEDURE VELTRANS (VAR brela: frame; VAR vrela,vrelb: vec3);
VAR
  w,px,py: real;
BEGIN
w:=vrela[3];  { planar case,w is invariant }
px:=brela[1,3];  { px and py are the org. of B wrt A }
py:=brela[2,3];
vrelb[1]:=brela[1,1]*(vrela[1]-py*w)+brela[2,1]*(vrela[2]+px*w);
vrelb[2]:=brela[1,2]*(vrela[1]-py*w)+brela[2,2]*(vrela[2]+px*w);
vrelb[3]:=w;
{ equation implemented is:
        B     A T   A
        V  =   R* (V + w*[-py])
         B     B     A  [px]                              }
END;

PROCEDURE JACOBIAN (VAR theta: vec3; VAR JAC: mat33);
VAR
  s3,c3,s23,c23: real;
BEGIN
s3:=sin (theta[3]); c3:=cos (theta[3]);
s23:=sin (theta[2] + theta[3]);
c23:=cos (theta[2] + theta[3]);
JAC[1,1]:=l2*s3 + l1*s23; JAC[1,2]:=l2*s3; JAC[1,3]:=0.0;
JAC[2,1]:=l2*c3 + l1*c23; JAC[2,2]:=l2*c3; JAC[2,3]:=0.0;
JAC[3,1]:=1.0;            JAC[3,2]:=1.0;   JAC[3,3]:=1.0;
END;

PROCEDURE VMULT (VAR M: mat33; VAR x,y: vec3);  { y = M*x  }
VAR
  i,j: integer;
BEGIN
FOR i:=1 TO 3 DO
  BEGIN
  y[i]:=0.0;
  FOR j:=1 TO 3 DO
      y[i]: = y[i] + M[i,j]*x[j];
  END;
END;

PROCEDURE INITDATA;  { Set up the given data }
BEGIN
  l1:=0.5;
```

```
  l2:=0.5;
  trelwuser[1]:=0.1;
  trelwuser[2]:=0.2;
  trelwuser[3]:=30.0;
  srelbuser[1]:=0.0;
  srelbuser[2]:=0.0;
  srelbuser[3]:=0.0;
  UTOI (srelbuser,srelb);
  TINVERT (srelb,brels);
  UTOI (trelwuser,trelw);
  TINVERT (trelw,wrelt);
END;  { initdata }

BEGIN  { Main Test Program }
  INITDATA;
  WRITE ('Care to do some robot kinematics? ');
  READLN (ans);
  WHILE ans='y' DO
  BEGIN
    WRITELN ('Enter cur pos (theta1,theta2,theta3):');
    READLN (current[1],current[2],current[3]);
    FOR i: = 1 to 3 do
      current[i]:=current[i]*dtor;
    WRITELN ('Enter tool rel station (x,y,theta):');
    READLN (trelsuser[1],trelsuser[2],trelsuser[3]);
    UTOI (trelsuser,trels);
    SOLVE (trels,current,near,far,sol);
    IF sol THEN
      BEGIN
      WRITELN ('The 2 solutions are (in radians):');
      WRITEVECT (near);
      WRITELN ('and:');
      WRITEVECT (far);
      WRITELN ('Enter joint vel (thetadot1,2,3):');
      READLN (thetadot[1],thetadot[2],thetadot[3]);
      FOR i:=1 TO 3 DO
        thetadot[i]:=thetadot[i]*dtor;

      WRITELN;
      JACOBIAN (near,JACnear);  { jacobian for the near sol }
      VMULT (JACnear,thetadot,vrelw);
      VELTRANS (trelw,vrelw,vrelt);
      WRITELN ('The tool vel for the near soln is:');
      vrelt[3]:=vrelt[3]*rtod;
      WRITEVECT (vrelt);

      WRITELN;
      JACOBIAN (far,JACfar);  { jacobian for the far sol }
      VMULT (JACfar,thetadot,vrelw);
      VELTRANS (trelw,vrelw,vrelt);
      WRITELN ('The tool vel for the far soln is:');
      vrelt[3]:=vrelt[3]*rtod;  { convert to degrees/sec }
      WRITEVECT (vrelt);
      END
    ELSE WRITELN ('Sorry, no sol for those coords.');
    WRITE ('Would you like to do some more?');
    READLN (ans);
  END;
```

Programming Exercise (Part 6)

```
PROGRAM pp6(INPUT,OUTPUT);
CONST
  dtor=0.0174532925;
  rtod=57.29578;

  l1 = 0.5;
  l2 = 0.5;
  m1 = 4.6;   { Mass of the links }
  m2 = 2.3;
  m3 = 1.0;
  Izz = 0.1;  { Moment of inertia for link 3 about Z }
   g = 9.8;   { Gravity }

  intstep = 0.005;  { Integration Step Size (seconds) }

TYPE
  vec3 = ARRAY [1..3] OF REAL;
  mat33 = ARRAY[1..3,1..3] OF REAL;

VAR
  b: vec3;     { viscous friction coefficients }
  theta,thetadot,thetadd : vec3;  {  pos.,vel.,acc. }
  m: mat33;    { manipulator mass matrix }
  minv: mat33; { inverse of mass matrix }
  v: vec3;     { vector of dynamic terms }
  tau: vec3;   { vector of joint torques }
  s1,c1,s2,c2,s12,c12: REAL;
  i,j: INTEGER;
  len: REAL;   { total length of time to simulate }
  prnt: REAL;  { how often to print (sec.) }
  time: REAL;  { time into simulation }
  ans: char;

Procedure inv33(VAR a,ainv: mat33);  { a is non-sing,sym }
VAR det: real;
BEGIN
  det:=a[1,1]*a[2,2]*a[3,3]+a[1,2]*a[2,3]*a[3,1];
  det:=det+a[1,3]*a[2,1]*a[3,2];
  det:=det-a[1,3]*a[2,2]*a[3,1]-a[1,2]*a[2,1]*a[3,3];
  det:=det-a[1,1]*a[2,3]*a[3,2];
  ainv[1,1]:=(a[2,2]*a[3,3]-a[2,3]*a[3,2])/det;
  ainv[1,2]:=(a[1,3]*a[3,2]-a[1,2]*a[3,3])/det;
  ainv[1,3]:=(a[1,2]*a[2,3]-a[2,2]*a[1,3])/det;
  ainv[2,2]:=(a[1,1]*a[3,3]-a[3,1]*a[1,3])/det;
  ainv[2,3]:=(a[2,1]*a[1,3]-a[1,1]*a[2,3])/det;
  ainv[3,3]:=(a[1,1]*a[2,2]-a[1,2]*a[2,1])/det;
  ainv[2,1]:=ainv[1,2];
  ainv[3,1]:=ainv[1,3];
  ainv[3,2]:=ainv[2,3];
END;  { inv33 }

PROCEDURE trig;
BEGIN
  s1:=sin (theta[1]);
  c1:=cos (theta[1]);
  s2:=sin (theta[2]);
  c2:=cos (theta[2]);
  s12:=sin (theta[1]+theta[2]);
  c12:=cos (theta[1]+theta[2]);
END;
```

```
PROCEDURE calcm(VAR m: mat33);  { Calculate the M Matrix }
VAR m23: REAL;

BEGIN
  m23:=m2+m3;
  m[1,1]:=l2*l2*m23+2*l1*l2*m23*c2+l1*l1*(m1+m23)+Izz;
  m[1,2]:=l2*l2*m23+l1*l2*m23*c2+Izz;
  m[1,3]:=Izz;
  m[2,1]:=m[1,2];   { M is symmetric }
  m[2,2]:=l2*l2*m23+Izz;
  m[2,3]:=Izz;
  m[3,1]:=m[1,3];   { M is symmetric }
  m[3,2]:=m[2,3];   { M is symmetric }
  m[3,3]:=Izz;
END;

PROCEDURE calcsdt(VAR v: vec3);  { Calculate V }
VAR m23: REAL;  { b[i] are viscous friction coefficients. }
BEGIN
  m23:=m2+m3;
  v[1]:=-m23*l1*l2*s2*thetadot[2]*thetadot[2];
  v[1]:=v[1]-2*m23*l1*l2*thetadot[1]*thetadot[2];
  v[1]:=v[1]+m23*l2*g*c12+(m1+m23)*l1*g*cl;
  v[1]:=v[1]+b[1]*thetadot[1];
  v[2]:=m23*l1*l2*s2*thetadot[1]*thetadot[1];
  v[2]:=v[2]+m23*l2*g*c12+b[2]*thetadot[1];
  v[3]:=b[3]*thetadot[3];
END;

PROCEDURE VMULT (VAR M: mat33; VAR x,y: vec3);
VAR
  i,j: integer;
BEGIN
FOR i:=1 TO 3 DO
  BEGIN
  y[i]:=0.0;
  FOR j:=1 TO 3 DO
      y[i]:=y[i] + M[i,j]*x[j];
  END;
END;

PROCEDURE update (VAR tau: vec3;
                  VAR period: REAL;
                  VAR theta,thetadot: vec3);
VAR i,j,k: INTEGER;
    temp: vec3;
BEGIN
  j:=round(period/instep);
  FOR i:=1 TO j DO
  BEGIN
    trig;          { do trig functions on current angles }
    calcm(m);      { calculate mass matrix }
    inv33(m,minv);   { invert mass matrix  }
    calcsdt(v);      { calculate velocity,grav.,friction }
    FOR k:=1 TO 3 DO temp[k]:=tau [k]-v[k];
    vmult(minv,temp,thetadd);  { compute joint acceleration }
    FOR k:=1 TO 3 DO           { do integration step }
    BEGIN
      theta[k]:=theta [k]+thetadot[k]*intstep
                +0.5*intstep*intstep*thetadd[k];
      thetadot[k]:=thetadot[k]+thetadd[k]*intstep;
    END;
  END;
END;  { update }
```

```
BEGIN  { Main. }
  ans:='y';
  WHILE ans='y' DO
  BEGIN
   WRITELN('Want to simulate joint friction?');
   READLN(ans);
   IF ans='y' THEN
     FOR i:=1 TO 3 DO b[i]:=5.0  { friction }
   ELSE
     FOR i:=1 TO 3 DO b[i]:=0.0;  { no friction }
  WRITELN('How long to simulate?');
  READLN(len);
  WRITELN('How often to print? (sec.)');
  READLN(prnt);
  WRITELN('Enter initial joint positions (degrees):');
  READLN(theta[1],theta[2],theta[3]);
  FOR i:=1 TO 3 DO theta[i]:=theta [i]*dtor;
  FOR i:=1 TO 3 DO tau[i]:=0.0;
  FOR i:=1 TO 3 DO thetadot[i]:=0.0;
  time:=0.0;
  j:=round(len/prnt);
  WRITELN(' Time Joint 1 Joint 2 Joint 3');
  FOR i:=1 TO j DO
  BEGIN
    update(tau,prnt,theta,thetadot);
    time:=time+prnt;
  writeln(time:9:2.theta [1]*rtod:12:3,
           theta [2]*rtod:12:3,theta [3]*rtod:12:3);
  END;
  WRITELN('Want to Simulate the Robot?');
  READLN(ans);
 END;
END.
```

Programming Exercise (Part 7)

```
program pp7 (INPUT,OUTPUT);

CONST
 pi=3.14159;
 halfpi=1.5707;
 dtor=0.0174532925;
 rtod=57.29578;

TYPE
 frame = ARRAY[1..2,1..3] OF REAL;
 vec3 = ARRAY[1..3] OF REAL;
 vec4 = ARRAY[1..4] OF REAL;
 pntlist = ARRAY[1..5] OF vec3;
 seglist = ARRAY[1..5,1..3] of vec4;
VAR
  i,j: integer;
  theta: vec3;
  brelw,wrelb: frame;
  l1,l2: REAL;
  wrelt,trelw: frame;
  srelb,brels: frame;
  srelt,trels: frame;
  trelwuser: vec3;
  srelbuser: vec3;
  trelsuser: vec3;
  place,current,far: vec3;
```

```
  npnt: integer;
  viapnt,viavel: pntlist;
  path: seglist;
  trajectory: array[1..300] of vec3;
  deltat.s: real;
  nticks: integer;
  sol: boolean;
  ans: char;

PROCEDURE WRITEFRAME (VAR foo: frame);
VAR i,j: integer;

BEGIN
  FOR  i: = 1 TO 2 DO
  BEGIN
    WRITE ('[');
    FOR j: = 1 TO 3 DO
      WRITE (foo[i,j]: 10 : 3);
    WRITELN (']');
  END;
END;

PROCEDURE WRITEVECT (VAR foo: vec3);
VAR i: integer;

BEGIN
  FOR  i:=1 TO 3 do
    WRITELN ('[',foo[i]: 10 : 3,']');
END;

FUNCTION AT2 (a,b: REAL): REAL;
VAR ans: REAL;
BEGIN
  IF  b=0.0 THEN
  BEGIN
    IF  a=0.0 THEN ans:=0.0
    ELSE
    IF a>0.0 THEN  ans:=halfpi
    ELSE
    ans:=-halfpi
  END
  ELSE
  BEGIN
    ans:=arctan (a/b);
    IF  b<0 THEN
    IF  a<0 THEN  ans:=ans-pi ELSE  ans:=ans+fpi;
  END;
  at2:=ans;
END; { atan2 }

PROCEDURE TMULT (VAR brela,crelb,crela: frame);
VAR i,j,k: INTEGER;
BEGIN
  FOR  i:= 1 TO 2 DO
  BEGIN
    crela[i,3]:=brela[i,3];
    FOR  j:=1 TO 2 DO
    BEGIN
      crela[i,3]:=crela[i,3]+brela[i,j] *crelb[j,3];
      crela[i,j]: = 0.0;
      FOR  k:=1 TO 2 DO
        crela[i,j]:=crela[i,j]+brela[i,k] *crelb[k,j];
    END;
  END;
END;  { tmult  }
```

```
PROCEDURE TINVERT (VAR brela,arelb: frame);
VAR i,j: INTEGER;
BEGIN
  arelb[1,1]:=brela[1,1];
  arelb[1,2]:=brela[2,1];
  arelb[2,1]:=brela[1,2];
  arelb[2,2]:=brela[2,2];
  FOR  i:=1 TO 2 DO
  BEGIN
    arelb[i,3]:=0.0;
    FOR  j:=1 TO 2 DO
      arelb[i,3]:=arelb[i,3]-brela[j,i] *brela[j,3];
  END;
END;  { tinvert }

PROCEDURE KIN(VAR theta: vec3; VAR wrelb: frame);
VAR beta: real;

BEGIN
  wrelb[1,3]:=l1 *cos (theta [1])+l2 *cos (theta [1]+theta [2]);
  wrelb[2,3]:=l1 *sin (theta [1])+l2 *sin (theta [1]+theta [2]);
  beta:=theta[1]+theta [2]+theta [3];
  wrelb[1,1]:=cos (beta);
  wrelb[1,2]:=-sin (beta);
  wrelb[2,1]:=-wrelb[1,2];
  wrelb[2,2]:=wrelb[1,1];
END;

PROCEDURE UTOI(VAR uform: vec3; VAR iform: frame);
BEGIN
  iform[1,3]:=uform[1];
  iform[2,3]:=uform[2];
  iform[1,1]:=cos (uform[3] *dtor);
  iform[1,2]:=-sin (uform[3] *dtor);
  iform[2,1]:=-iform[1,2];
  iform[2,2]:=iform[1,1];
END;

PROCEDURE ITOU(VAR iform: frame; VAR uform: vec3);
BEGIN
  uform[1]:=iform[1,3];
  uform[2]:=iform[2,3];
  uform[3]:=at2(iform[2,1],iform[1,1]) *rtod;
END;

PROCEDURE range(VAR a: REAL);
BEGIN
  WHILE a>pi DO  a:=a-2 *pi;
  WHILE a< -pi DO  a:=a+2 *pi;
END;  { range  }

PROCEDURE invk in(VAR wrelb:frame;
                  VAR current,near,far:vec3;
                  VAR sol:boolean);
LABEL 5;
VAR c2,s2,k1,k2,temp: REAL;
    i: INTEGER;
    goal,swap: vec3;
BEGIN
  itou(wrelb,goal);
  goal[3]:=goal[3] *dtor;
  c2:=(goal[1] *goal[1]+goal[2] *goal[2]-l1 *l1-l2 *l2)/(2.0 *l1 *l2);
  IF abs (c2)>1.0 THEN  sol:=FALSE ELSE sol:=TRUE;
  IF NOT sol THEN GOTO 5;
  s2:=sqrt(1.0-c2 *c2);
```

```
  near[2]:=at2(s2,c2);
  far[2]:=-near[2];
  k1:=l1+l2 *c2;
  k2:=l2 *s2;
  temp:=at2(k2,k1);
  near[1]:=at2(goal[2],goal[1])-temp;
  far[1]:=at2(goal[2],goal[1])+temp;
  near[3]:=goal[3]-near[1]-near[2];
  far[3]:=goal[3]-far[1]-far[2];
  FOR  i:=1 TO 3 DO
  BEGIN
    range(near[i]);
    range(far[i]);
  END;
  IF dist(current,near)>dist(current,far)THEN
  BEGIN
    swap:=near;
    near:=far;
    far:=swap;
  END;
5:
END; { invkin }

PROCEDURE SOLVE(VAR trels:frame;
                VAR current,near,far:vec3;
                VAR sol:boolean);
VAR trelb,wrelt,wrelb: frame;
BEGIN
  tmult(srelb,trels,trelb);
  tinvert(trelw,wrelt);
  tmult(trelb,wrelt,wrelb);
  invkin(wrelb,current,near,far,sol);
END;  { Solve }

PROCEDURE WHERE(VAR theta: vec3; VAR trels: frame):
VAR trelb,wrelb: frame;

BEGIN
  KIN(theta,wrelb);
  TMULT(wrelb,trelw,trelb);
  TMULT(brels,trelb,trels);
END;

PROCEDURE INITDATA;  { Set up the given data  }
BEGIN
  s:=0.3333;         { set time scaling constant }
  deltat:=0.2;
  l1:=0.5;           { initialize link lengths  }
  l2:=0.5;
  trelwuser[1]:=0.1;
  trelwuser[2]:=0.2;
  trelwuser[3]:=30.0;
  srelbuser[1]:=0.0;
  srelbuser[2]:=0.0;
  srelbuser[3]:=0.0;
  UTOI(srelbuser,srelb);
  TINVERT(srelb,brels);
  UTOI(trelwuser,trelw);
  TINVERT(trelw,wrelt);
END; { initdata  }
{ This is done differently than in the text. In this program
  a time scale factor is used instead of giving the total time
  for a cubic segment. Its really simpler to do it just as in
  the text, but this scaling business has some benefits too.}
```

```
PROCEDURE CUBCOEFF (VAR th0,thf,thdot0,thdotf: real;
                    VAR cc: vec4);
BEGIN
cc[1]:=th0;
cc[2]:=thdot0/s;
cc[3]:=3 * (thf - th0) - 2 *thdot0/s - thdotf/s;
cc[4]:=-2 * (thf - th0) + thdotf/s + thdot0/s;
END;

PROCEDURE JOINTVEL (VAR vjapnt: pntlist; VAR npnt: integer;
                    VAR viavel: pntlist);
{ Compute the joint velocities at the via points }
VAR      i,j: integer;
BEGIN
for j:=1 to 3 do
  begin
  viavel[1][j]:=0.0; viavel[npnt][j]:=0.0;
  end;
if npnt > 2 then
  begin
  for i:=2 to npnt - 1 do
     for j:=1 to 3 do
       viavel [i][j]:=0.5 *((viapnt[i][j]-viapnt[i-1][j]) +
                            (viapnt[i+1][j]-viapnt[i][j])) *s;
{ The velocity chosen for each via point is the
   average of the velocities
   for each path segment if all the via points were
   connected with straight
   lines. See chap. 7. }
   end;
 END;

PROCEDURE RUNPATH (VAR path: seglist; VAR npnt: integer);
VAR     k,i,j:integer;
        time,tprime: real;
BEGIN
time:=0.0;
i:=1;
nticks:=round((npnt - 1) *3/deltat);
writeln ('path in cartesian space (time,x,y,phi):');
for k:=1 to nticks do
  begin
  time:=time + deltat;
  tprime:=time *s;
  for j:=1 to 3 do
    theta [j]:=path[i,j][1] + path [i,j][2] *tprime +
    path [i,j][3] * sqr (tprime)+path[i,j][4] * sqr (tprime) *tprime;
  WHERE (theta,trels);
  ITOU (trels,trajectory [k]);
  write (time: 5 : 2);
  for  j:=1 to 3 do write (trajectory[k][j]:10:3);
  writeln;
  if time>3.0 then
    begin
    time:=0.0; { reset time each segment }
    i:=i+1;
    end;
  end;
END;

{ here's a hack to plot the path on a TTY }

PROCEDURE plotpath;
var i,k,j: integer;
    xscale,yscale: real;
```

```
    ch: array[0..3] of char;
    grid: array[1..60,1..80] of char;
begin
xscale:=30.0;  yscale:=22.5;
ch[0]:='<';  ch [1]:='V';  ch[2]:='>';  ch[3]:='↑';
for i:=1 to 60 do
   for j:=1 to 80 do grid[i,j]:=' ';
for j:=1 to 80 do grid [31,j]:='-';
for i:=1 to 60 do grid [i,40]:='1';
for k:=1 to nticks do
  begin
  i:=31-round (trajectory[k][2] * yscale);
  j:=round(trajectory[k][1] * xscale)+40;
  grid[i,j]:=ch[(round(trajectory[k][3]+225) div 90) mod 4];
  end;
for k:=1 to npnt do
  begin
  where (viapnt[k],trels);
  itou (trels,place);
  i:=31-round (place[2] * yscale);
  j:=round (place[1] * xscale)+40;
  grid[i,j]:=' *'
  end;
for i:=1 to 57 do
  begin
  for j:=1 to 78 do write(grid[i,j]);
  writeln;
  end;
end;
BEGIN  { Main Test Program }
  INITDATA;
  WRITE ('Care to plan some swell robot paths? ');
  READLN (ans);
  WHILE  ans='y' DO
  BEGIN
    npnt:=1;
    for i:=1 to 3 do current[i]:=0.0;
{ enter the path via points and convert to joint angles }
     WRITELN ('Enter the initial position (x,y,phi): ');
     READLN (place[1],place[2],place[3]);
     UTOI (place,trels);
     SOLVE (trels,current,viapnt [npnt],far,sol);
     while sol do
       begin
       npnt:=npnt+1;
       writeln ('Enter the next via point (x,y,phi): ');
       writeln ('(enter a point with no sol to terminate) ');
       readln (place [1],place[2],place[3]);
       UTOI (place,trels);
       SOLVE (trels,viapnt[npnt-1],viapnt[npnt],far,sol);
       end;
     npnt:=npnt-1;
     JOINTVEL(viapnt,npnt,viavel);
     for i:=1 to  npnt-1 do
      for j:=1 to 3 do
       CUBCOEFF (viapnt[i][j],viapnt[i+1][j],viavel[i][j],
                                viavel[i+1][j],path[i,j]);
    RUNPATH(path,npnt);
    PLOTPATH;
    WRITE ('Would you like to do some more? ');
    READLN (ans);
  END;
END.
```

Programming Exercise (Part 9 and 10)

```
PROGRAM pp8(INPUT,OUTPUT);

CONST
  dtor=0.0174532925;
  rtod=57.29578;

  l1 = 0.5;
  l2 = 0.5;
  m1 = 4.6;      { Mass of the links }
  m2 = 2.3;
  m3 = 1.0;
  Izz = 0.1;     { Moment of inertia for link 3 about Z }
   g = 9.8;      { Gravity }
  intstep = 0.005; { Integration Step Size (seconds) }

TYPE
  vec3 = ARRAY[1..3] OF REAL;
  mat33 = ARRAY[1..3,1..3] OF REAL;
  pathtype = ARRAY[1..3,1..3,1..4] OF REAL;

VAR
  b: vec3;       { viscous friction coefficients }
  theta, thetadot, thetadd : vec3;
  m: mat33;      { manipulator mass matrix }
  minv: mat33;   { inverse of mass matrix }
  v: vec3;       { vector of state dependent dynamic terms }
  tau: vec3;     { vector of joint torques }
  s1, c1, s2, c2, s12, c12: REAL;
  i, j: INTEGER;
  len: REAL;     { total length of time to simulate }
  nprnt: INTEGER;   { how many ticks between printing }
  serper: REAL; { servo period }
  time: REAL;   { time into simulation }
  ans: char;
  pos, vel, acc: vec3;  { desired path }
  kp, kv; vec3;   { gains }
 stepans, dynans: char;
 tauprime, temp: vec3;
 mm: mat33;      { model of mass matrix used in control law }
 vm: vec3;
 path: pathtype;        { storage of cubic coefficients }

{ Normally, this routine would do the path planning }
{ based on user input. Here, for simplicity, I have }
{ the constants from the last homework precomputed. }
{ Not elegant, but it was easiest... }

PROCEDURE initpath;
BEGIN
path[1,1, 1]:= 1.047;
path[1,1, 2]:= 0.000;
path[1,1, 3]:= -5.673;
path[1,1, 4]:= 3.167;

path[1,2, 1]:= -1.920;
path[1,2, 2]:=  0.000;
path[1,2, 3]:= 10.328;
path[1,2, 4]:= -6.994;

path[1,3, 1]:=  0.349;
path[1,3, 2]:=  0.000;
path[1,3, 3]:= -0.204;
path[1,3, 4]:=  0.161;

path[2,1, 1]:= -1.459;
```

```
path[2,1, 2]:= -1.845;
path[2,1, 3]:= -1.114;
path[2,1, 4]:=  1.776;

path[2,2, 1]:=  1.415;
path[2,2, 2]:= -0.324;
path[2,2, 3]:= -9.633;
path[2,2, 4]:=  5.974;

path[2,3, 1]:=  0.306;
path[2,3, 2]:=  0.075;
path[2,3, 3]:=  0.407;
path[2,3, 4]:= -0.289;

path[3,1, 1]:= -2.643;
path[3,1, 2]:=  1.253;
path[3,1, 3]:=  8.563;
path[3,1, 4]:= -6.127;

path[3,2, 1]:= -2.568;
path[3,2, 2]:= -1.667;
path[3,2, 3]:=  5.279;
path[3,2, 4]:= -2.964;

path[3,3, 1]:=  0.499;
path[3,3, 2]:=  0.022;
path[3,3, 3]:= -0.491;
path[3,3, 4]:=  0.320;
END;  { initpath }

PROCEDURE trig;
BEGIN
  s1:=sin(theta[1]);
  c1:=cos(theta[1]);
  s2:=sin(theta[2]);
  c2:=cos(theta[2]);
  s12:=sin(theta[1]+ theta[2]);
  c12:=cos(theta[1]+ theta[2]);
END;

PROCEDURE calcm(VAR m: mat33);
VAR m23: REAL;
BEGIN
  m23:=m2+m3;
  m[1,1]:=12*12*m23+2*11*12*m23*c2+11*11*(m1+m23)+Izz;
  m[1,2]:=12*12*m23+11*12*m23*c2+Izz;
  m[1,3]:=Izz;
  m[2,1]:=m[1,2];               { M is symmetric }
  m[2,2]:=12*12*m23+Izz;
  m[2,3]:=Izz;
  m[3,1]:=m[1,3];               { M is symmetric }
  m[3,2]:=m[2,3];               { M is symmetric }
  m[3,3]:=Izz;
END;

PROCEDURE calcsdt(VAR v: vec3);
VAR m23: REAL;
BEGIN
  m23:=m2+m3;
  v[1]:=-m23*11*12*thetadot[2]*thetadot[2];
  v[1]:=v[1]-2*m23*11*12*s2*thetadot[1]*thetadot[2];
  v[1]:=v[1]+m23*12*g*c12+(m1+m23)*11*g*c1;
  v[1]:=v[1]+b[1]*thetadot[1];
  v[2]:=m23*11*12*s2*thetadot[1]*thetadot[1];
  v[2]:=v[2]+m23*12*g*c12+b[2]*thetadot[2];
  v[3]:=b[3]*thetadot[3];
END;
```

```
PROCEDURE VMULT(VAR M: mat33; VAR x, y: vec3);
VAR
  i,j: integer;
BEGIN
FOR i:=1 TO 3 DO
  BEGIN
  y[i]:=0.0;
  FOR j:=1 TO 3 DO
      y[i]:=y[i] + M[i,j]*x[j];
  END;
END;

PROCEDURE vadd(VAR a,b,c: vec3); { finds c = a+b }
VAR i: INTEGER;
BEGIN
  FOR i: = 1 TO 3 DO c[i]:=a[i]+b[i];
END; { vadd }

PROCEDURE update(VAR tau: vec3; VAR period: REAL;
                 VAR theta, thetadot: vec3);
VAR i,j,k: INTEGER;
    temp: vec3;
BEGIN
  j:=round(period/instep);
  FOR i:=1 TO j DO
  BEGIN
    trig;
    calcm(m);
    inv33(m,minv);
    calcsdt(v);
    FOR k:=1 TO 3 DO temp[k]:=tau[k]-v[k];
    vmult(minv,temp,thetadd);
    FOR k:=1 TO 3 DO
    BEGIN
      theta[k]:=theta[k]+thetadot[k]*instep
                +0.5*intstep*intstep*thetadd[k];
      thetadot[k]:=thetadot[k]+thetadd[k]*intstep;
    END;
  END;
END;  { update }

Procedure step: { Generate Step input at time = 3.0 }
BEGIN
  IF time >= 3.00 THEN pos[2]:=-50.0*dtor;
END;

PROCEDURE spline;{ Generate Cubic Spline Path }
VAR i,j: INTEGER;
    tim: REAL;          { scaled time (always [0,1]) }
BEGIN
 IF time < 9.0 THEN
 BEGIN
  IF time>6.00 THEN i:=3 ELSE
    IF time>3.00 THEN i:=2 ELSE i:=1;
  tim:=(time-(i-1)*3.0)/3.0;
  FOR j:=1 TO 3 DO
  BEGIN
   pos[j]:=path[i,j,1]+(path[i,j,2]+(path[i,j,3]
           +path[i,j,4]*tim)*tim)*tim;
   vel[j]:=path[i,j,2]+(2.0*path[i,j,3]
           +3.0*path[i,j,4]*tim)*tim;
   vel[j]:=0.33333*vel[j];
   acc[j]:=2.0*path[i,j,3]+6.0*path[i,j,4]*tim;
```

```
   acc[j]:=0.3333*0.33333*acc[j];
  END;
 END;
END;  { spline }

PROCEDURE servo; { tauprime based on desired path and errors }
BEGIN
  FOR i: = 1 TO 3 DO
  BEGIN
    tauprime[i]:=acc[i]+kv[i]*(vel[i]-thetadot[i])
                 +kp[i]*(pos[i]-theta[i]);
  END;
END;  { servo }

BEGIN           { Main. }
 initpath;      { plan the splined path }
 ans:='y';
 WHILE ans='y' DO
 BEGIN
  WRITELN('How long to simulate?');
  READLN(len);
  WRITELN('Print position every N servo periods... N=?');
  READLN(nprnt);
  WRITELN('Spline path or Step? (y = Step)');
  readln(stepans);
  writeln('Use dynamic compensation?');
  readln(dynans);
  WRITELN('Enter velocity gains: kv[i] (3 vals):');
  READLN(kv[1], kv[2], kv[3]);
  IF dynans = 'y' THEN
  BEGIN
    kp[1]: = 175.0; kp[2]:=110.0; kp[3]:=20.0;
  END
  ELSE
  BEGIN
    kp[1]:=100.0; kp[2]:=100.0; kp[3]:=100.0;
  END;
  theta[1]:=60.0; theta[2]:=-110.0; theta[3]:=20.0;
  FOR i:=1 TO 3 DO theta[i]:=theta[i]*dtor;
  FOR i:=1 TO 3 DO
  BEGIN
    pos[i]:=theta[i];   { initialize desired path }
    vel[i]:=0.0;
    acc[i]:=0.0;
  END;
  FOR i:=1 TO 3 DO thetadot[i]:=0.0;
  FOR i:=1 TO 3 DO b[i]:=5.0;           { friction }
  time:=0.0;
  serper:=0.01;
  j:=round(len/serper);
  WRITELN('     Time   Joint 1   Joint 2   Joint 3');
  writeln(time:9:2.theta[1]*rtod:12:3,
          theta[2]*rtod:12:3,theta[3]*rtod:12:3);
  FOR i:=1 TO j DO
  BEGIN
    IF stepans='y' THEN step ELSE spline;
    servo;      { Uses gains and errors to compute tauprime }
    IF dynans='y' THEN
    BEGIN
      calcm(mm);
      calcsdt(vm);
      vmult(mm,tauprime,temp);
      vadd(temp,vm,tau);
    END ELSE tau:=tauprime;     { No dynamics used }
```

```
    update(tau,serper,theta,thetadot);
    time:=time+serper;
  IF (i MOD nprnt)=0 THEN
  BEGIN
   writeln(time:9:2,theta[1]*rtod:12:3,
           theta[2]*rtod:12:3,theta[3]*rtod:12:3);
  END;
  END;
  WRITELN('Want to Simulate the Robot?');
  READLN(ans);
END;
```

Programming Exercise (Part 11)

```
PROGRAM sm9 (INPUT,OUTPUT);

CONST
  dtor=0.0174532925;
  rtod=57.29578;


  l1 = 0.5;
  l2 = 0.5;
  m1 = 4.6;
  m2 = 2.3;
  m3 = 1.0;
  Izz = 0.1;
   g = 9.8;
  intstep = 0.005;

TYPE
  vec3 = ARRAY[1..3] OF REAL;
  mat33 = ARRAY[1..3.1..3] OF REAL;
  frame = ARRAY[1..2.1..3] OF REAL;
  pathtype = ARRAY[1..3,1..3,1..4] OF REAL;
 VAR
   b: vec3;
   theta,thetadot,thetadd: vec3;
   m: mat33;
   minv: mat33;
   v: vec3;
   tau: vec3;
   s1,c1,s2,c2,s12,c12: REAL;
   i,j: INTEGER;
   len: REAL;
   nprnt: INTEGER;
   serper: REAL;   servo period
   time: REAL;     time into simulation
   ans: char;
   pos,vel,acc: vec3;   desired path
   kp,kv: vec3;
   stepans, dynans: char;
   tauprime,temp: vec3;
   fes: vec3;      stored force acting on hand
   fe: vec3;       actual force acting on hand
   eft: vec3;      torque due to contact forces
   wrelb: frame:   wrist rel base
   kpx: vec3;      desired Cartesian stiffness
   gravtor: vec3;   torques due to gravity
```

```
PROCEDURE trig;
BEGIN
  s1:=sin(theta[1]);
  c1:=cos(theta[1]);
  s2:=sin(theta[2]);
  c2:=cos(theta[2]);
  s12:=sin(theta[1]+theta[2]);
  c12:=cos(theta[1]+theta[2]);
END;

PROCEDURE calcm(.VAR m: mat33);
VAR m23: REAL;
BEGIN
  m23:=m2+m3;
  m[1,1]:=l2*l2*m23+2*l1*l2*m23*c2+l1*l1*(m1+m23)+Izz;
  m[1,2]:=l2*l2*m23+l1*l2*m23*c2+Izz;
  m[1,3]:=Izz;
  m[2,1]:=m[1,2];
  m[2,2]:=l2*l2*m23+Izz;
  m[2,3]:=Izz;
  m[3,1]:=m[1,3];
  m[3,2]:=m[2,3];
  m[3,3]:=Izz;
END;

PROCEDURE calcsdt(VAR v: vec3);
VAR m23: REAL;
BEGIN
  m23:=m2+m3;
  v[1]:=-m23*l1*l2*s2*thetadot[2]*thetadot[2];
  v[1]:=v[1]-2*m23*l1*l2*s2*thetadot[1]*thetadot[2];
  v[1]:=v[1]+m23*l2*g*c12+(m1+m23)*l1*g*c1;
  v[1]:=v[1]+b[1]*thetadot[1];
  v[2]:=m23*l1*l2*s2*thetadot[1]*thetadot[1];
  v[2]:=v[2]+m23*l2*g*c12+b[2]*thetadot[2];
  v[3]:=b[3]*thetadot[3];
END;

PROCEDURE grav(VAR v: vec3);
VAR m23: REAL;
BEGIN
  m23:=m2+m3;
  v[1]:=m23*l2*g*c12+(m1+m23)*l1*g*c1;
  v[2]:=m23*l2*g*c12;
  v[3]:=0.0;
END;

PROCEDURE VMULT(VAR M: mat33; VAR x,y: vec3);
VAR
  i,j: integer;
BEGIN
FOR  i:=1 TO 3 DO
  BEGIN
  y[i]:=0.0;
  FOR j:=1 TO 3 DO
      y[i]:=y[i] + M[i,j]*x[j];
  END;
END;

PROCEDURE vadd(VAR a,b,c: vec3);
VAR i: INTEGER;
BEGIN
  FOR  i:=1 TO 3 DO  c[i]:=a[i]+b[i];
END;    vadd
```

```
PROCEDURE JACOBIAN (VAR theta: vec3; VAR JAC: mat33);
VAR
  s3,c3,s23,c23; real;
BEGIN
s3:=sin(theta[3]); c3:=cos(theta[3]);
s23:=sin(theta[2] + theta[3]); c23:=cos(theta[2]+theta[3]);
JAC[1,1]:=l2*s3+l1*s23; JAC[1,2]:=l2*s3; JAC[1,3]:=0.0;
JAC[2,1]:=l2*c3+l1*c23; JAC[2,2]:=l2*c3; JAC[2,3]:=0.0;
JAC[3,1]:=1.0;          JAC[3,2]:=1.0;   JAC[3,3]:=1.0;
END;

PROCEDURE transpose (VAR x: mat33; VAR y: mat33);
VAR i,j: INTEGER;
BEGIN
  FOR  i:=1 TO 3 DO
    FOR  j:=1 TO 3 DO
      y[i,j]:=x[j,i];
END;   transpose

PROCEDURE calceft (VAR torque: vec3);   torque = J+T Fe
VAR jac,jact: mat33;
    i: INTEGER;
BEGIN
  jacobian(theta,jac);
  transpose(jac,jact);
  vmult(jact,fe,torque);
  FOR  i:=1 TO 3 DO  torque[i]:=-torque[i];
END;   calceft

PROCEDURE update (VAR tau: vec3; VAR period: REAL;
                  VAR theta,thetadot: vec3);
VAR i,j,k: INTEGER;
    temp: vec3;
BEGIN
  j:=round(period/intstep);
  FOR  i:=1 TO j DO
  BEGIN
    trig;
    calcm(m);
    inv33(m,minv);
    calcsdt(v);
    calceft(eft);   calc. torques due to hand forces
    FOR  k:=1 TO 3 DO  temp[k]:=tau[k]-v[k]-eft[k];
    vmult(minv,temp,thetadd);
    FOR  k:=1 TO 3 DO
    BEGIN
      theta[k]:=theta[k]+thetadot[k]*intstep
                +0.5*intstep*intstep*thetadd[k];
      thetadot[k]:=thetadot[k]+thetadd[k]*intstep;
    END;
  END;
END;   update

PROCEDURE KIN (VAR theta: vec3; VAR wrelb: frame);
VAR beta: real;
BEGIN
  wrelb[1,3]:=l1*cos(theta[1]) + l2*cos(theta[1] + theta[2]);
  wrelb[2,3]:=l1*sin(theta[1]) + l2*sin(theta[1] + theta[2]);
  beta:=theta[1] + theta[2] + theta[3];
  wrelb[1,1]:=cos (beta);
  wrelb[1,2]:=-sin (beta);
  wrelb[2,1]:=-wrelb[1,2];
  wrelb[2,2]:=wrelb[1,1];
END;
```

```
PROCEDURE forcestep;   make forces appear at t=3 secs
VAR i: INTEGER;
BEGIN
  IF  time¿=3.0 THEN
   BEGIN
     FOR  i:=1 TO 3 DO  fe[i]:=fes[i];
   END;
END;   forcestep
 This control law is different than the one suggested in the
  book in that the gravity model is used to feedforward some
  torque to cancel the gravity effect. It will work without
  this, but then the arm sags due to gravity if made compliant
  in that direction.

PROCEDURE servo;
VAR e,edot: vec3;
    tpe: vec3;   torque to fix position errors
    jac,jact: mat33;   jacobian AND transpose
    kpxmat: mat33;
    temp,temp2: vec3;
BEGIN
  kpxmat[1,1]:=kpx[1]; kpxmat[1,2]:=0.0; kpxmat[1,3]:=0.0;
  kpxmat[2,1]:=0.0; kpxmat[2,2]:=kpx[2]; kpxmat[2,3]:=0.0;
  kpxmat[3,1]:=0.0; kpxmat[3,2]:=0.0; kpxmat[3,3]:=kpx[3];
  jacobian(theta,jac);   for present position
  transpose(jac,jact);
  FOR  i:=1 TO 3 DO  e[i]:=pos[i] - theta[i];
  vmult(jac,e,temp);
  vmult(kpxmat,temp,temp2);
  vmult(jact,temp2,tpe);
  grav(gravtor);   calculate gravity torques
  FOR  i:=1 TO 3 DO
  BEGIN
    edot[i]:=vel[i]-thetadot[i];
    tauprime[i]:=tpe[i]+kv[i]*edot[i]+gravtor[i];
  END;
END;   servo
BEGIN      Main.
 ans:='y';
 WHILE  ans='y' DO
 BEGIN
  fe[1]:=0.0; fe[2];=0.0; fe[3]:=0.0;
  fes[3]:=0.0;   no torques on hand
  WRITE('Enter Cartesian Force to
         apply (X and Y values in Newtons');
  READLN(fes[1],fes[2]);
  WRITELN('Forces appear at  t=3 s, How long to simulate?');
  READLN(len);
  WRITELN('Print position every N servo periods ...  N=?');
  READLN(nprnt);
  WRITELN('Enter Stiffnesses: kpx[i] (3 vals):');
  READLN(kpx[1],kpx[2],kpx[3]);
  kv[1]:=10.0; kv[2]:=10.0; kv[3]:=10.0;
  theta[1]:=60.0; theta[2]:=-90.0; theta[3]:=30.0;
  FOR  i:=1 TO 3 DO  theta[i]:=theta[i]*dtor;
  FOR  i:=1 TO 3 DO
  BEGIN
    pos[i]:=theta[i];   initialize desired path
    vel[i]:=0.0;
    acc[i]:=0.0;
  END;
  FOR  i:=1 TO 3 DO  thetadot[i]:=0.0;
  FOR  i:=1 TO 3 DO  b[i]:=5.0;
  time:=0.0;
  serper:=0.01;
```

```
  j:=round (len/serper);
  WRITELN('      Time       X pos   Y pos');
  kin(theta,wrelb);
  writeln(time:9:2,wrelb[1,3]:12:3,wrelb[2,3]:12:3);
  FOR  i:=1 TO j DO
  BEGIN
    forcestep;   Make forces appear at t=3 secs
    servo;
    tau:=tauprime;   No dynamics used
    update(tau,serper,theta,thetadot);
    time:=time+serper;
  IF (i MOD nprnt)=0 THEN
  BEGIN
   kin(theta,wrelb;
   writeln(time:9:2,wrelb[1,3]:12:3,wrelb[2,3]:12:3);
 END;
  END;
  WRITELN('Want to Simulate the Robot?');
  READLN(ans);
 END;
END.
Enter Cartesian Force to apply (X and Y values in Newtons
5 0
Forces appear at  t=3 s, How long to simulate?
8
Print position every N servo periods...  N=?
50
Enter Stiffnesses: kpx[i] (3 vals):
10 200 200

     Time         X pos         Y pos
     0.00       0.683        0.183
     0.50       0.685        0.181
     1.00       0.684        0.183
     1.50       0.684        0.183
     2.00       0.684        0.183
     2.50       0.684        0.183
     3.00       0.684        0.183
     3.50       0.698        0.180
     4.00       0.711        0.179
     4.50       0.723        0.180
     5.00       0.735        0.180
     5.50       0.746        0.181
     6.00       0.756        0.182
     6.50       0.765        0.182
     7.00       0.774        0.183
     7.50       0.783        0.184
     8.00       0.791        0.185
Want to Simulate the Robot?
y
Enter Cartesian Force to apply (X and Y values in Newtons
0 5
Forces appear at  t=3 s. How long to simulate?
8
Print position every N servo periods...  N=?
50
Enter Stiffnesses: kpx[i] (3 vals):
10 200 200

     Time         X pos       Y pos
     0.00      0.683       0.183
     0.50      0.682       0.183
     1.00      0.682       0.183
     1.50      0.682       0.183
     2.00      0.683       0.183
```

2.50	0.683	0.183
3.00	0.683	0.183
3.50	0.679	0.204
4.00	0.678	0.207
4.50	0.678	0.208
5.00	0.678	0.208
5.50	0.678	0.208
6.00	0.679	0.208
6.50	0.679	0.208
7.00	0.679	0.208
7.50	0.679	0.208
8.00	0.679	0.208

Matlab Exercises – Solutions

Robert L. Williams II
Mechanical Engineering
Ohio University
Athens, OH

Craig Robotics Text Matlab Exercises – Solutions

Robert L. Williams II

Outline

2a) Orthonormal rotation matrices

2b) Homogeneous transformation matrices

3) Forward pose kinematics – Planar 3R

4) Inverse pose kinematics – Planar 3R

5) Jacobian, determinant, resolved-rate, inverse statics – Planar 3R

6a) Inverse dynamics simulation – Planar 2R

6b) Inverse dynamics snapshot – Planar 3R

6c) Forward dynamics trajectory – Planar 3R

7) Trajectory generation – 3^{rd}, 3^{rd} w/ via, 5^{th}

8) Resolved-rate with kinematic redundancy – Planar 4R, particular only

9) Single joint control Simulink simulation

All Matlab solutions were developed on a PC running Windows 98 and Matlab R12, and the beta version 7 of the Corke Robotics Toolbox . However, these exercises and solutions should be general for other platforms, operating systems, and software version numbers (these versions or newer).

Matlab Exercise 2A

a)

i) Given $\alpha = 10°$, $\beta = 20°$, and $\gamma = 30°$; $\quad {}^{A}_{B}R = \begin{bmatrix} 0.9254 & 0.0180 & 0.3785 \\ 0.1632 & 0.8826 & -0.4410 \\ -0.3420 & 0.4698 & 0.8138 \end{bmatrix}$

Column-wise orthonormal demonstration (row-wise is similar):

$${}^{A}\hat{X}_B \times {}^{A}\hat{Y}_B = \begin{Bmatrix} 0.9254 \\ 0.1632 \\ -0.3420 \end{Bmatrix} \times \begin{Bmatrix} 0.0180 \\ 0.8826 \\ 0.4698 \end{Bmatrix} = \begin{Bmatrix} 0.3785 \\ -0.4410 \\ 0.8138 \end{Bmatrix} = {}^{A}\hat{Z}_B \quad \text{(3 scalar constraints)}$$

$$\left\| {}^{A}\hat{X}_B \right\| = \left\| \begin{Bmatrix} 0.9254 \\ 0.1632 \\ -0.3420 \end{Bmatrix} \right\| = 1 \qquad \left\| {}^{A}\hat{Y}_B \right\| = \left\| \begin{Bmatrix} 0.0180 \\ 0.8826 \\ 0.4698 \end{Bmatrix} \right\| = 1 \qquad \left\| {}^{A}\hat{Z}_B \right\| = \left\| \begin{Bmatrix} 0.3785 \\ -0.4410 \\ 0.8138 \end{Bmatrix} \right\| = 1$$

(3 scalar constraints)

Demonstrate the *beautiful* property ${}^{B}_{A}R = {}^{A}_{B}R^{-1} = {}^{A}_{B}R^{T}$:

$${}^{B}_{A}R = inv\left({}^{A}_{B}R\right) = \begin{bmatrix} 0.9254 & 0.1632 & -0.3420 \\ 0.0180 & 0.8826 & 0.4698 \\ 0.3785 & -0.4410 & 0.8138 \end{bmatrix} = {}^{A}_{B}R^{T}$$

ii) Given $\alpha = 30°$, $\beta = 90°$, and $\gamma = -55°$: $\quad {}^{A}_{B}R = \begin{bmatrix} 0 & -0.9962 & 0.0872 \\ 0 & 0.0872 & 0.9962 \\ -1 & 0 & 0 \end{bmatrix}$

b)

i) Given ${}^{A}_{B}R = \begin{bmatrix} 0.9254 & 0.0180 & 0.3785 \\ 0.1632 & 0.8826 & -0.4410 \\ -0.3420 & 0.4698 & 0.8138 \end{bmatrix}$;

Solution	α	β	γ
1	10°	20°	30°
2	−170°	160°	−150°

The second solution is not really needed – just included as an exercise. Plug the second solution row into the a. code and it yields the same rotation matrix (circular check).

ii) Given ${}^{A}_{B}R = \begin{bmatrix} 0 & -0.9962 & 0.0872 \\ 0 & 0.0872 & 0.9962 \\ -1 & 0 & 0 \end{bmatrix}$;

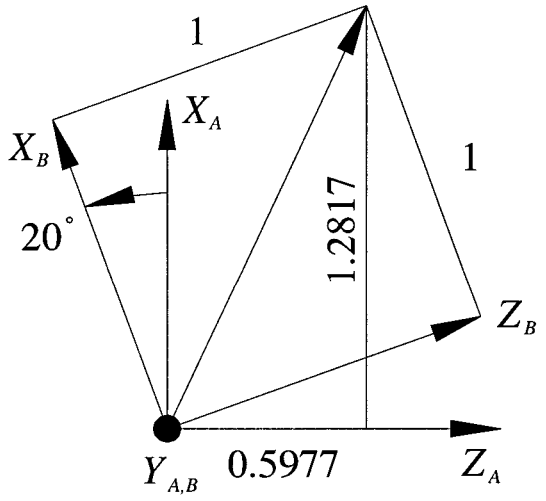
Solution	α	β	γ
1	30°	90°	-55°
2	-150°	90°	125°

Again, plug the second solution row into the a. code and it yields the same rotation matrix (circular check). This case is singular (artificial singularity) since $\beta = 90^\circ$ and the inverse solution must divide by $\cos\beta$. Therefore, one should implement the alternate artificial singularity solution, equation (2.67). However, in my solution, this was not necessary – I guess the $\cos\beta$ terms were enough different from 0 to allow the Matlab *atan* function to work properly.

c) Given $\alpha = 0^\circ$, $\beta = 20^\circ$, and $\gamma = 0^\circ$ also ${}^{B}P = \{1 \quad 0 \quad 1\}^T$;

$${}^{A}P = {}^{A}_{B}R\,{}^{B}P = \begin{bmatrix} 0.9397 & 0 & 0.3420 \\ 0 & 1 & 0 \\ -0.3420 & 0 & 0.9397 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 1.2817 \\ 0 \\ 0.5977 \end{Bmatrix}$$

Sketch:



d) Check all results with the Corke Matlab Robotics Toolbox.

```
%  Exercise 2A using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clear; clc;

DR = pi/180;

angs   = input('enter alpha, beta, gamma (deg): ');
alp    = angs(1)*DR;
bet    = angs(2)*DR;
gam    = angs(3)*DR;

Tz     = rotz(alp);      Rz = Tz(1:3,1:3);
Ty     = roty(bet);      Ry = Ty(1:3,1:3);
Tx     = rotx(gam);      Rx = Tx(1:3,1:3);

%  Part a
Rmult  = Rz*Ry*Rx;              % Z-Y-X a-b-g Euler angles a'la Craig
Trpy   = rpy2tr(alp,bet,gam)    % Same result - rpy corresponds to Z-Y-X a-b-g Euler
Rrpy   = Trpy(1:3,1:3)          % Extract 3x3 rotation matrix from 4x4

%  Part b
ang    = tr2rpy(Trpy);          % Inverse solution - rpy corresponds to Z-Y-X a-b-g Euler
ang/DR                          % Display inverse solution results

%  Part c
beta   = input('enter beta (deg): ');
Pb     = input('enter [Pb] (column-wise): ');
Tyc    = roty(beta*DR);  Ryc = Tyc(1:3,1:3);
Pa     = Ryc*Pb;
```

Using this Matlab program, the answers check for part a. The function *rpy2tr*() corresponds to the *Z-Y-X* α-β-γ Euler convention; i.e. roll-pitch-yaw is the same convention. The result returned is a 4x4 homogeneous transformation matrix with zeros in the translation vector column, so we extract the 3x3.

Also using this program, the inverse results check for part b. Only one solution is given, the second possibility is ignored. Also, case ii is a singular case, so the result in this case is:

Solution	α	β	γ
1	0°	90°	−85°

Upon plugging this singular solution into the forward function *rpy2tr*(), the correct rotation matrix results.

The result for part c also agrees with the answer given above.

Matlab Exercise 2B

a)

i) Given $\alpha = 10°$, $\beta = 20°$, and $\gamma = 30°$ and ${}^{A}P_{B} = \{1 \quad 2 \quad 3\}^{T}$:

$$ {}^{A}_{B}T = \begin{bmatrix} 0.9254 & 0.0180 & 0.3785 & 1 \\ 0.1632 & 0.8826 & -0.4410 & 2 \\ -0.3420 & 0.4698 & 0.8138 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

ii) Given $\beta = 20°$ ($\alpha = \gamma = 0°$), and ${}^{A}P_{B} = \{3 \quad 0 \quad 1\}^{T}$:

$$ {}^{A}_{B}T = \begin{bmatrix} 0.9397 & 0 & 0.3420 & 3 \\ 0 & 1 & 0 & 0 \\ -0.3420 & 0 & 0.9397 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

b) Given $\beta = 20°$ ($\alpha = \gamma = 0°$), and ${}^{A}P_{B} = \{3 \quad 0 \quad 1\}^{T}$, and ${}^{B}P = \{1 \quad 0 \quad 1\}^{T}$:

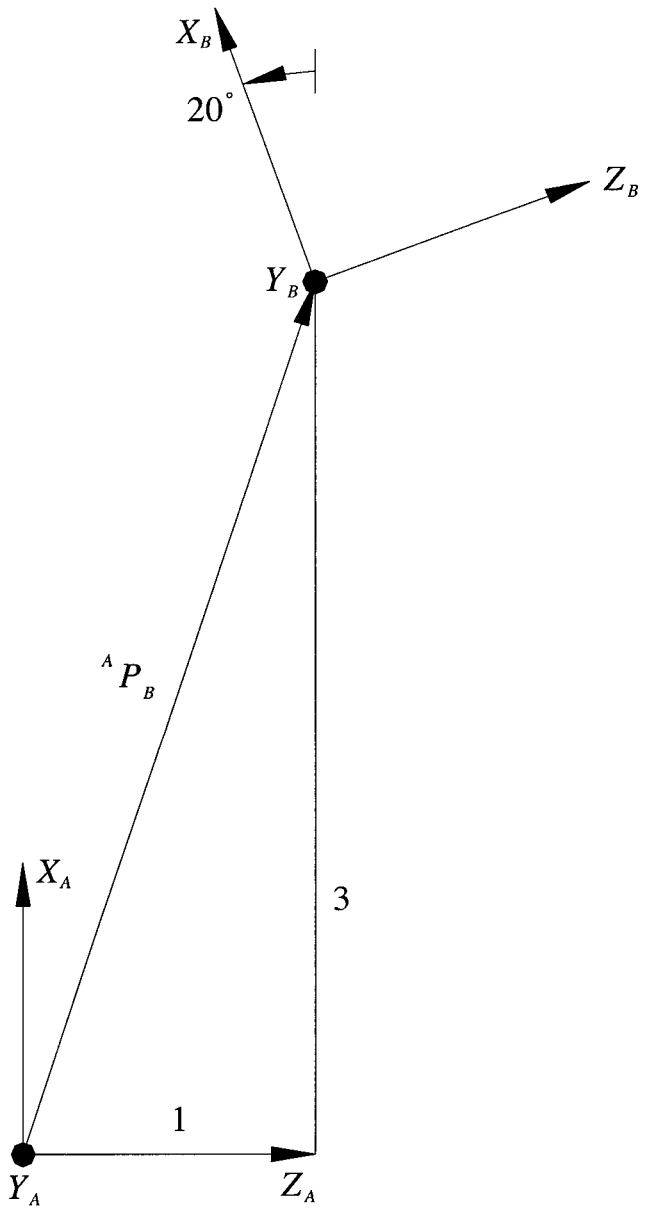
$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}P = \begin{bmatrix} 0.9397 & 0 & 0.3420 & 3 \\ 0 & 1 & 0 & 0 \\ -0.3420 & 0 & 0.9397 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 4.2817 \\ 0 \\ 1.5977 \\ 1 \end{Bmatrix} $$

3 Interpretations for Homogeneous Transformation Matrices:

1) Description of a frame

${}^{A}_{B}T$ describes the pose (position and orientation) of moving Cartesian coordinate frame $\{B\}$, w.r.t. to a reference frame, $\{A\}$. $\left\{{}^{A}P_{B}\right\}$ is the position vector giving the location of the origin of $\{B\}$ w.r.t. the origin of $\{A\}$, expressed in the basis (coordinates) of $\{A\}$. $\left[{}^{A}_{B}R\right]$ is the rotation matrix giving the orientation of $\{B\}$ w.r.t. $\{A\}$; columns are the XYZ unit vectors of $\{B\}$ projected onto the XYZ $\{A\}$ unit directions.

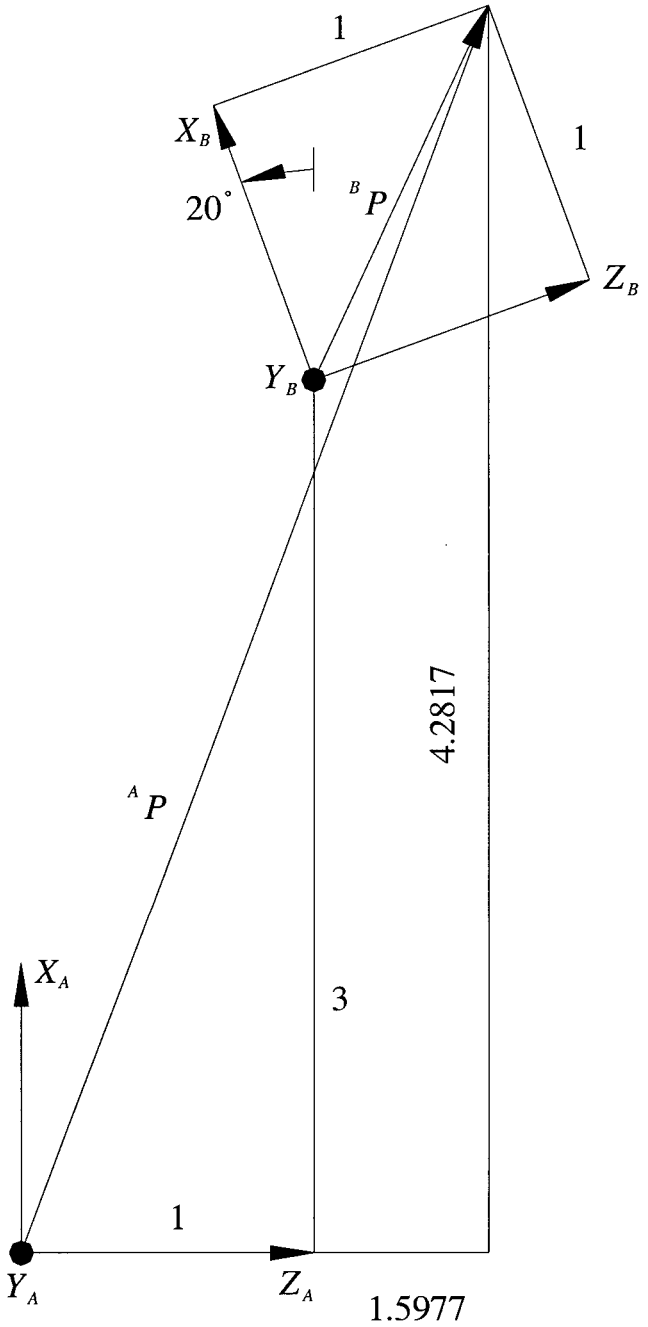
Figure for first interpretation:



2) Transform Mapping:

Matrix ${}^{A}_{B}T$ maps ${}^{B}P \rightarrow {}^{A}P$. Describes a vector known in one Cartesian coordinate frame $\{B\}$ in another frame $\{A\}$. There is both position and orientation (basis) difference in general. The ${}^{A}P = {}^{A}_{B}T\,{}^{B}P$ math has already been done above immediately following b.

Figure for second interpretation:

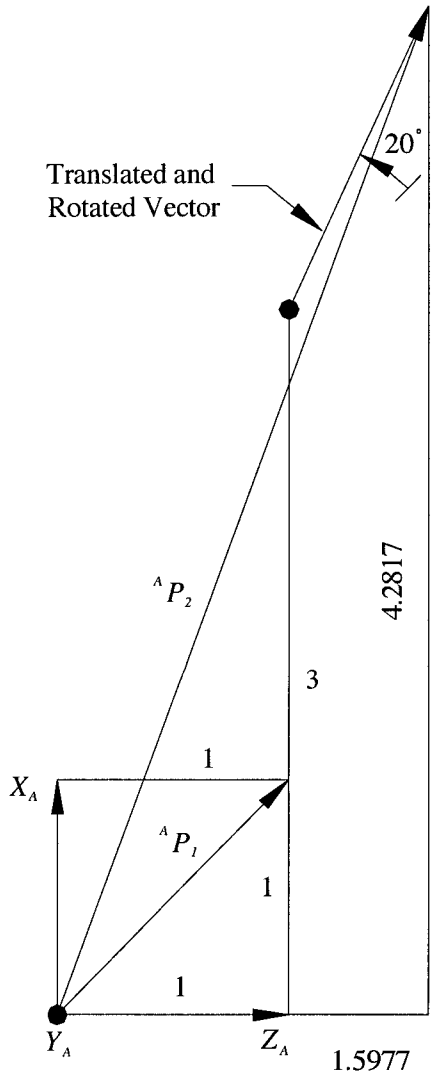


3) Transform Operator:

T operates on ${}^{A}P_1$ to yield ${}^{A}P_2$. Same Cartesian coordinate frame $\{A\}$, there is no $\{B\}$ for this interpretation. The original vector ${}^{A}P_1$ is translated and rotated to new vector ${}^{A}P_2$ via T. Order of translation and rotation doesn't matter if we assume rotations always occur about the tail of vectors. Let us use the same given numbers, i.e. $\beta = 20^\circ$ ($\alpha = \gamma = 0^\circ$), and $P = \{3 \quad 0 \quad 1\}^T$, and ${}^{A}P_1 = \{1 \quad 0 \quad 1\}^T$:

$$ {}^{A}P_2 = T\,{}^{A}P_1 = \begin{bmatrix} 0.9397 & 0 & 0.3420 & 3 \\ 0 & 1 & 0 & 0 \\ -0.3420 & 0 & 0.9397 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 4.2817 \\ 0 \\ 1.5977 \\ 1 \end{Bmatrix} $$

Figure for third interpretation:



c)

i) Both methods (symbolic formula, Matlab function *inv*) yield:

$$ {}^{A}_{B}T^{-1} = \begin{bmatrix} 0.9254 & 0.1632 & -0.3420 & -0.2257 \\ 0.0180 & 0.8826 & 0.4698 & -3.1927 \\ 0.3785 & -0.4410 & 0.8138 & -1.9380 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

ii) Both methods (symbolic formula, Matlab function *inv*) yield:

$$ {}^{A}_{B}T^{-1} = \begin{bmatrix} 0.9397 & 0 & -0.3420 & -2.4771 \\ 0 & 1 & 0 & 0 \\ 0.3420 & 0 & 0.9397 & -1.9658 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For both cases, ${}^{A}_{B}T\,{}^{A}_{B}T^{-1} = {}^{A}_{B}T^{-1}\,{}^{A}_{B}T = I_4$ can be shown.

d)

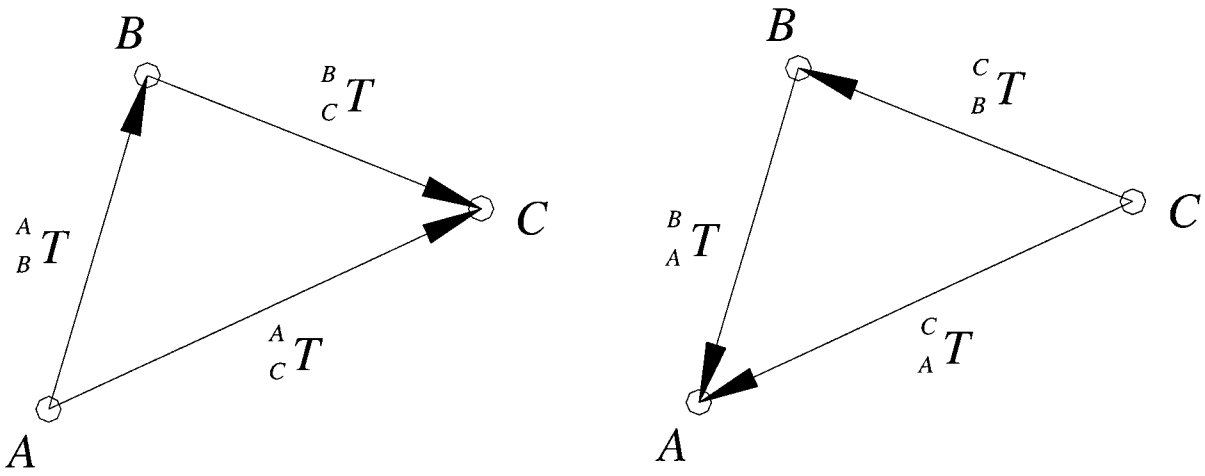
i) Calculate ${}^{A}_{C}T$ and show the relationship via a transform graph. Ditto for ${}^{C}_{A}T$.

$$ {}^{A}_{C}T = {}^{A}_{B}T\,{}^{B}_{C}T = \begin{bmatrix} 0.9254 & 0.0180 & 0.3785 & 1 \\ 0.1632 & 0.8826 & -0.4410 & 2 \\ -0.3420 & 0.4698 & 0.8138 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.9397 & 0 & 0.3420 & 3 \\ 0 & 1 & 0 & 0 \\ -0.3420 & 0 & 0.9397 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} 0.7401 & 0.0180 & 0.6722 & 4.1548 \\ 0.3042 & 0.8826 & -0.3586 & 2.0486 \\ -0.5997 & 0.4698 & 0.6477 & 2.7877 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{C}_{A}T = {}^{C}_{B}T\,{}^{B}_{A}T = {}^{B}_{C}T^{-1}\,{}^{A}_{B}T^{-1} = \begin{bmatrix} 0.7401 & 0.3042 & -0.5997 & -2.0263 \\ 0.0180 & 0.8826 & 0.4698 & -3.1927 \\ 0.6722 & -0.3586 & 0.6477 & -3.8641 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Transform graphs:



ii) Given ${}^{A}_{C}T$ and ${}^{B}_{C}T$ from d)*i*; assume you don't know ${}^{A}_{B}T$ - calculate it and compare with the answer you know.

$$ {}^{A}_{B}T = {}^{A}_{C}T\,{}^{C}_{B}T = {}^{A}_{C}T\,{}^{B}_{C}T^{-1} = \begin{bmatrix} 0.9254 & 0.0180 & 0.3785 & 1 \\ 0.1632 & 0.8826 & -0.4410 & 2 \\ -0.3420 & 0.4698 & 0.8138 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

iii) Given ${}^{A}_{C}T$ and ${}^{A}_{B}T$ from d)*i*; assume you don't know ${}^{B}_{C}T$ - calculate it and compare with the answer you know.

$$ {}^{B}_{C}T = {}^{B}_{A}T\,{}^{A}_{C}T = {}^{A}_{B}T^{-1}\,{}^{A}_{C}T = \begin{bmatrix} 0.9397 & 0 & 0.3420 & 3 \\ 0 & 1 & 0 & 0 \\ -0.3420 & 0 & 0.9397 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

e) Check all results with the Corke Matlab Robotics Toolbox.

```
%  Exercise 2B using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clear; clc;

DR = pi/180;

%  Part a
angs    = input('enter alpha, beta, gamma (deg):  ');
alp     = angs(1)*DR;
bet     = angs(2)*DR;
gam     = angs(3)*DR;

Pba     = input('enter Pba (column-wise):  ');

Trpy    = rpy2tr(alp,bet,gam);              %  Rotation matrix 4x4, Z-Y-X a-b-g Euler convention
Ttrn    = transl(Pba);                %  Translation matrix 4x4 with Pba
Tba     = Ttrn*Trpy;                  %  4x4 Homogeneous transformation matrix Tba

%  Part b
Pb      = input('enter Pb (column-wise):  ');
Pa      = Tba*[Pb;1];

%  Part c
Tba_inv= inv(Tba);                    %  Inverse homogeneous transformation matrix - Matlab function
eye(4) - Tba*Tba_inv;                 %  Check - should be zero
eye(4) - Tba_inv*Tba;                 %  Check - should be zero

%  Part d
angs2   = input('enter alpha2, beta2, gamma2 (deg):  ');
alp2    = angs2(1)*DR;
bet2    = angs2(2)*DR;
gam2    = angs2(3)*DR;

Pcb     = input('enter Pcb (column-wise):  ');

Trpy2   = rpy2tr(alp2,bet2,gam2);     %  Rotation matrix 4x4, Z-Y-X a-b-g Euler convention
Ttrn2   = transl(Pcb);                %  Translation matrix 4x4 with Pcb
Tcb     = Ttrn2*Trpy2;                %  4x4 Homogeneous transformation matrix Tcb

Tca     = Tba * Tcb                   %  Part d.i
Tac     = inv(Tcb) * inv(Tba)
Tac2    = inv(Tca)                    %  Check - should be same as Tac

Tba2    = Tca * inv(Tcb)              %  Part d.ii

Tcb2    = inv(Tba) * Tca              %  Part d.iii
```

Using this Matlab program, the answers check for all parts. We use functions *rpy2tr*() and *transl*() to build the overall homogeneous transformation matrices. There is no special toolbox function to invert homogeneous transformation matrices so we use the Matlab function *inv*().

Matlab Exercise 3

a) DH parameters:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

b) Neighboring homogeneous transformation matrices; Also ${}^{B}_{0}T$ and ${}^{8}_{H}T$:

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & L_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{H}T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

c) Forward pose kinematics transformation:

$$ \left[{}^{0}_{3}T\right] = \left[{}^{0}_{1}T(\theta_1)\right]\left[{}^{1}_{2}T(\theta_2)\right]\left[{}^{2}_{3}T(\theta_3)\right] = \begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1 + L_2c_{12} \\ s_{123} & c_{123} & 0 & L_1s_1 + L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

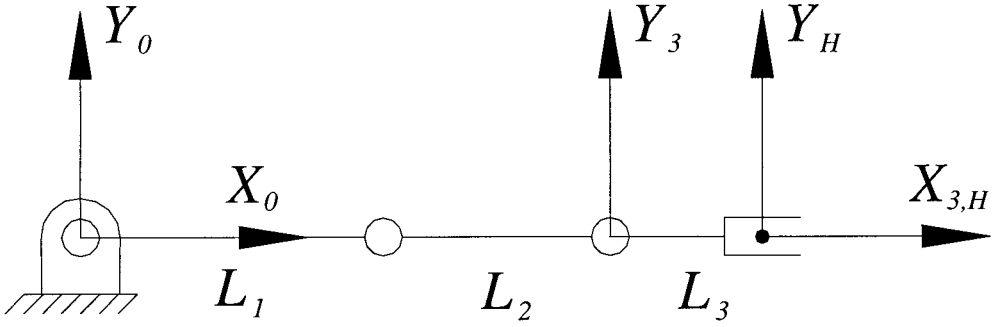
$$ \left[{}^{0}_{H}T\right] = \left[{}^{0}_{3}T(\theta_1,\theta_2,\theta_3)\right]\left[{}^{3}_{H}T(L_3)\right] = \begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1 + L_2c_{12} + L_3c_{123} \\ s_{123} & c_{123} & 0 & L_1s_1 + L_2s_{12} + L_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where: $s_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$, $c_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$, etc.

i) $\Theta = \{\theta_1 \quad \theta_2 \quad \theta_3\}^T = \{0 \quad 0 \quad 0\}^T$

$$ {}^0_3T = \begin{bmatrix} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^0_HT = \begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Sketch to verify:

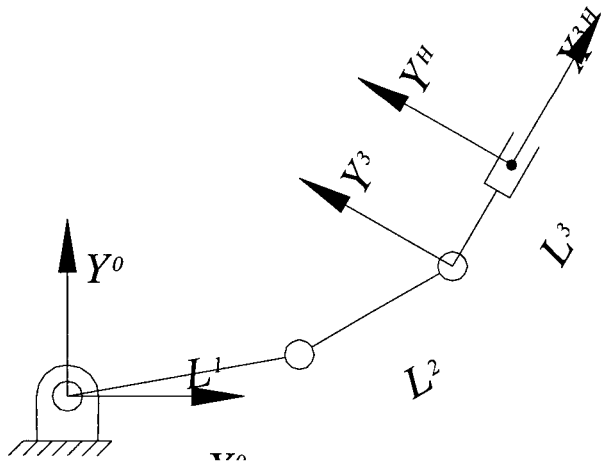


From this drawing, the above values can be derived by inspection (position and orientation).

ii) $\Theta = \{10^\circ \quad 20^\circ \quad 30^\circ\}^T$

$$ {}^0_3T = \begin{bmatrix} 0.5 & -0.866 & 0 & 6.5373 \\ 0.866 & 0.6 & 0 & 2.1946 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^0_HT = \begin{bmatrix} 0.5 & -0.866 & 0 & 7.5373 \\ 0.866 & 0.6 & 0 & 3.9266 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Sketch to verify:

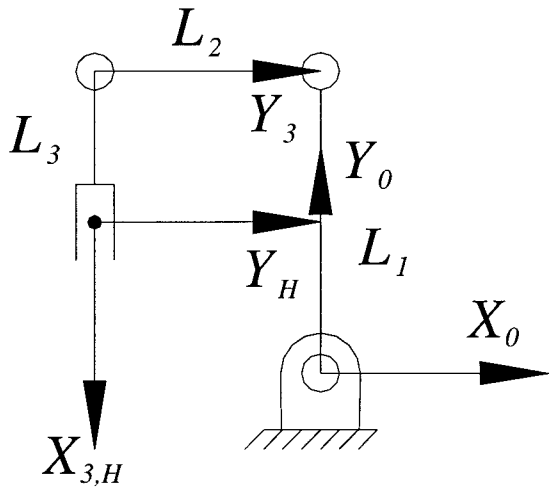


From this drawing, the above values can be derived by inspection (position and orientation).

iii) $\Theta = \{90^\circ \quad 90^\circ \quad 90^\circ\}^T$

$$ {}^0_3T = \begin{bmatrix} 0 & 1 & 0 & -3 \\ -1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^0_HT = \begin{bmatrix} 0 & 1 & 0 & -3 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Sketch to verify:



From this drawing, the above values can be derived by inspection (position and orientation).

d) Check all results with the Corke Matlab Robotics Toolbox.

```
%  Exercise 3 using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

%        Constants
DR = pi/180;
L1 = 4;  L2 = 3;  L3 = 2;

%  DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0],'mod');      % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0],'mod');
L{3} = link([alp(3),a(3),th(3),d(3),0],'mod');

ThreeR = robot(L, 'Plan3R');                        % Create 3R robot object

q1 = [0 0 0]*DR;
q2 = [10 20 30]*DR;
q3 = [90 90 90]*DR;

%  Forward Pose Kinematics
T30_1 = fkine(ThreeR,q1);                           % T30
T30_2 = fkine(ThreeR,q2);
T30_3 = fkine(ThreeR,q3);

TH3 = [1 0 0 L3;0 1 0 0;0 0 1 0;0 0 0 1];

TH0_1 = T30_1 * TH3;                                % TH0
TH0_2 = T30_2 * TH3;
TH0_3 = T30_3 * TH3;
```

Using this Matlab program, the answers check for all parts. We use function *link*() to assign the Craig-convention ('mod' for modified, or Craig-convention) DH parameters for each link. Then we build the Planar 3R robot object using function *robot*(). Finally, the basic forward pose transformation ${}^{0}_{3}T$ is found with function *fkine*(). There is no special toolbox function to find ${}^{0}_{H}T$ so we do it in the same manner as before.

Matlab Exercise 4

a) The book presents three methods to solve this problem; I favor the tangent-half angle algebraic approach, but all three will yield identical results. There are two solution sets for $\{\theta_1 \quad \theta_2 \quad \theta_3\}$, elbow down and elbow up configurations in the plane of motion.

b)

i) $${}^0_H T = \begin{bmatrix} 1 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ First calculate: $${}^0_3 T = {}^0_H T \, {}^3_H T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution	θ_1	θ_2	θ_3
Elbow Down	0	0	0
Elbow Up	0	0	0

ii) $${}^0_H T = \begin{bmatrix} 0.5 & -0.866 & 0 & 7.5373 \\ 0.866 & 0.6 & 0 & 3.9266 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

First calculate: $${}^0_3 T = {}^0_H T \, {}^3_H T^{-1} = \begin{bmatrix} 0.5 & -0.866 & 0 & 6.5373 \\ 0.866 & 0.6 & 0 & 2.1946 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution	θ_1	θ_2	θ_3
Elbow Down	10°	20°	30°
Elbow Up	27.1°	−20°	52.9°

iii) $${}^0_H T = \begin{bmatrix} 0 & 1 & 0 & -3 \\ -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ First calculate: $${}^0_3 T = {}^0_H T \, {}^3_H T^{-1} = \begin{bmatrix} 0 & 1 & 0 & -3 \\ -1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution	θ_1	θ_2	θ_3
Elbow Down	90°	90°	90°
Elbow Up	163.7°	−90°	−163.7°

iv) In this case, all joint angle solutions are imaginary; this means the solution does not exist. The commanded position vector is outside of the reachable workspace of the robot.

For cases *i-iii*, the Elbow Down solution is the same as the Forward Pose Kinematics input in Exercise 3A cases *i-iii*, which validates the inverse pose solution. The Elbow Up solution checks out when plugged into the forward pose program, i.e. the commanded ${}^{0}_{3}T$ and ${}^{0}_{H}T$ result. Note the Elbow Down and Elbow Up cases are identical for case *i*, which means this case lies on the solution boundary between multiple solutions; this is the elbow-straight singular case, i.e. the reachable workspace boundary.

d) Check all results with the Corke Matlab Robotics Toolbox.

```
%  Exercise 4 using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

%        Constants
DR = pi/180;
L1 = 4;  L2 = 3;  L3 = 2;

%  DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0],'mod');        % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0],'mod');
L{3} = link([alp(3),a(3),th(3),d(3),0],'mod');

ThreeR = robot(L, 'Plan3R');                          %  Create 3R robot object

TH3 = [1 0 0 L3;0 1 0 0;0 0 1 0;0 0 0 1];

TH0_1 = [1 0 0 9;0 1 0 0;0 0 1 0;0 0 0 1];
T30_1 = TH0_1 * inv(TH3);
TH0_2 = [0.5 -0.866 0 7.5373;0.866 0.5 0 3.9266;0 0 1 0;0 0 0 1];
T30_2 = TH0_2 * inv(TH3);
TH0_3 = [0 1 0 -3;-1 0 0 2;0 0 1 0;0 0 0 1];
T30_3 = TH0_3 * inv(TH3);
TH0_4 = [0.866  0.5 0 -3.1245;-0.5 0.866 0 9.1674;0 0 1 0;0 0 0 1];
T30_4 = TH0_4 * inv(TH3);

%  Inverse Pose Kinematics
M = [1 1 0 0 0 1];                                    % Mask for planar motion

%Guess1 = [0 0 0]*DR;
Guess1 = [90 -90 90]*DR;
Q1 = ikine(ThreeR, T30_1, Guess1, M);
Q1D = Q1/DR

%Guess2 = [0 0 0]*DR;                                 %  For Elbow Down
Guess2 = [20 -10 50]*DR;                              %  For Elbow Up
Q2 = ikine(ThreeR, T30_2, Guess2, M);
Q2D = Q2/DR

%Guess3 = [70 60 80]*DR;                              %  For Elbow Down
Guess3 = [150 -80 -150]*DR;                           %  For Elbow Up
Q3 = ikine(ThreeR, T30_3, Guess3, M);
Q3D = Q3/DR

Guess4 = [0 0 0]*DR;
Q4 = ikine(ThreeR, T30_4, Guess4, M);
Q4D = Q4/DR
```

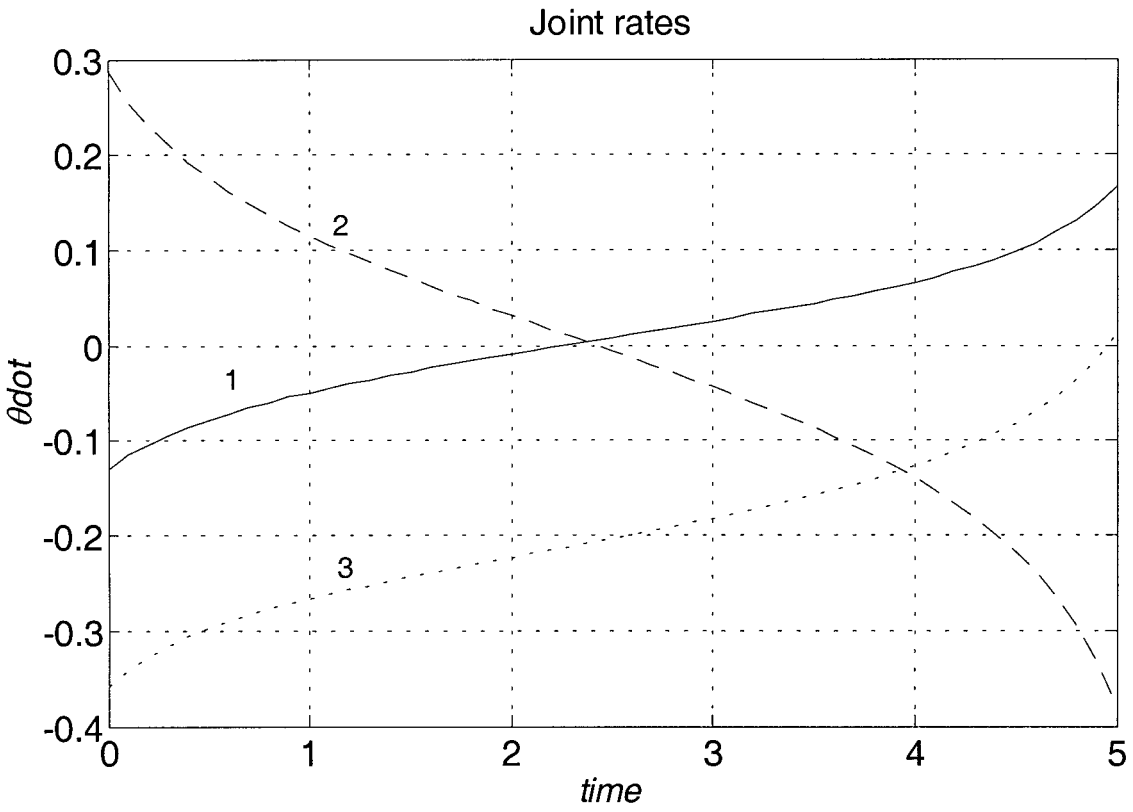
Using this Matlab program, the answers check for all parts. We use the mask M to indicate planar motion only (x and y translation, rotation about z). An initial guess is required for iterative function *ikine*(). For case *i*, the solution converges to the same result for both initial guesses shown, as expected. For cases *ii* and *iii*, different initial guesses were used to find the Elbow Down and Elbow Up solutions, as shown.

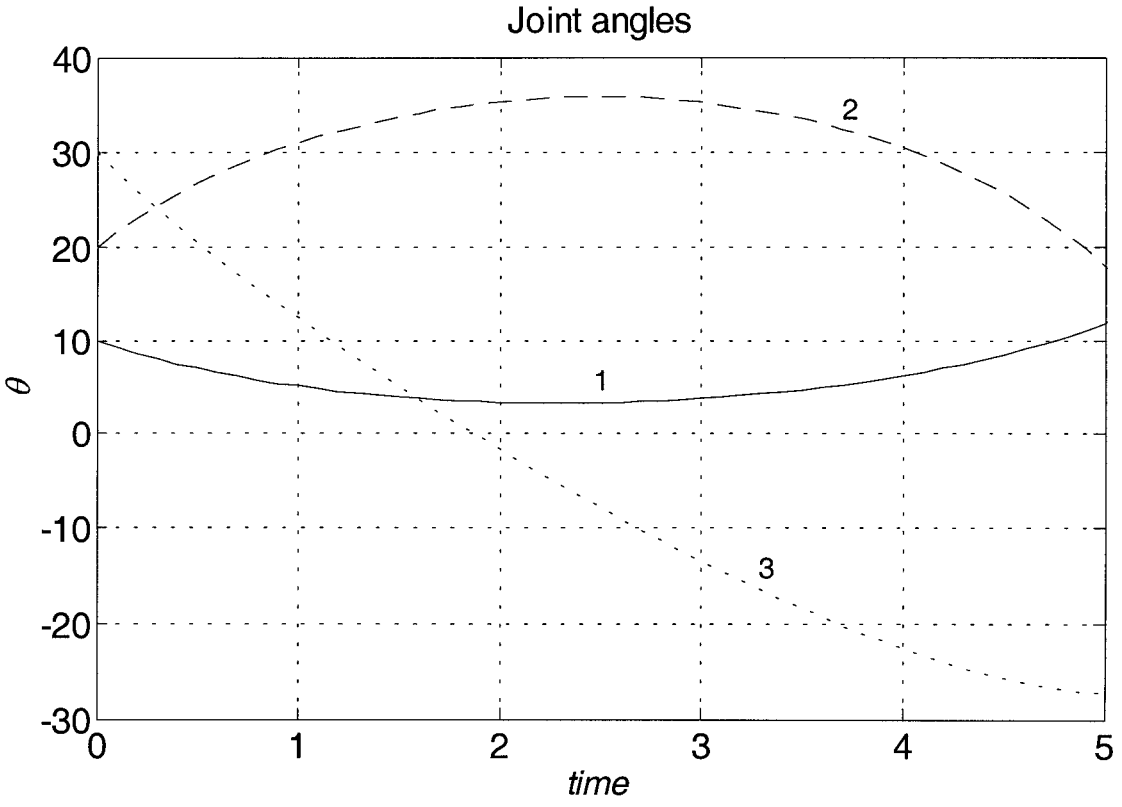
For case *iv*, there was no convergence to a solution. This is expected since the commanded position is out of reach for the robot. The *Matlab* error message is displayed below:

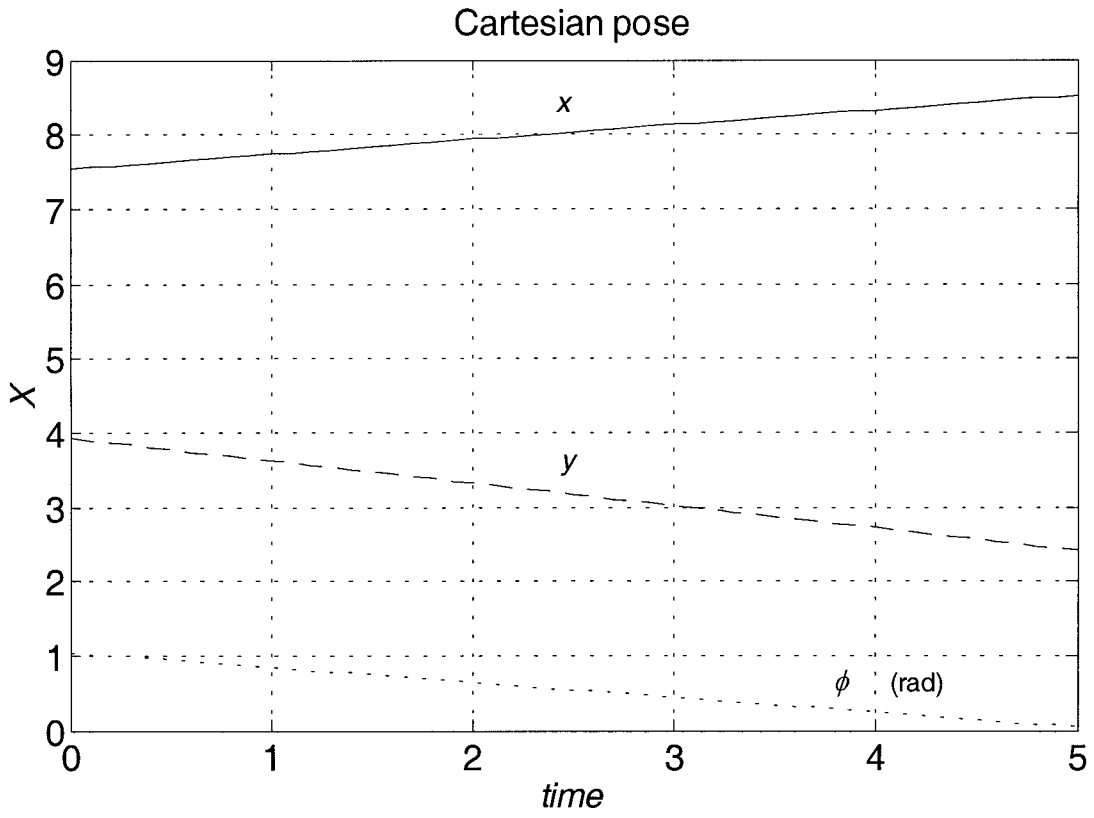
```
??? Error using ==> ikine
Solution wouldn't converge

Error in ==> C:\mydocs\matfiles\Corke\Bob4A.m
On line 55  ==> Q4 = ikine(ThreeR, T30_4, Guess4, M);
```

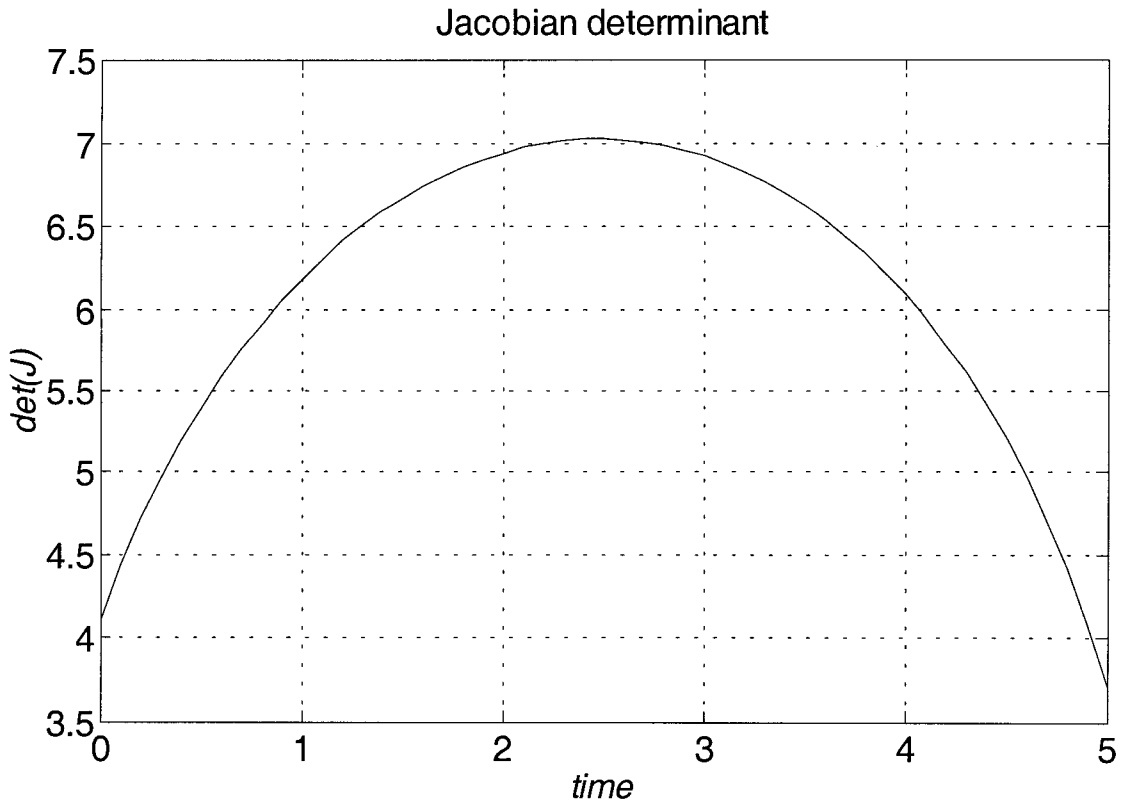
Matlab Exercise 5

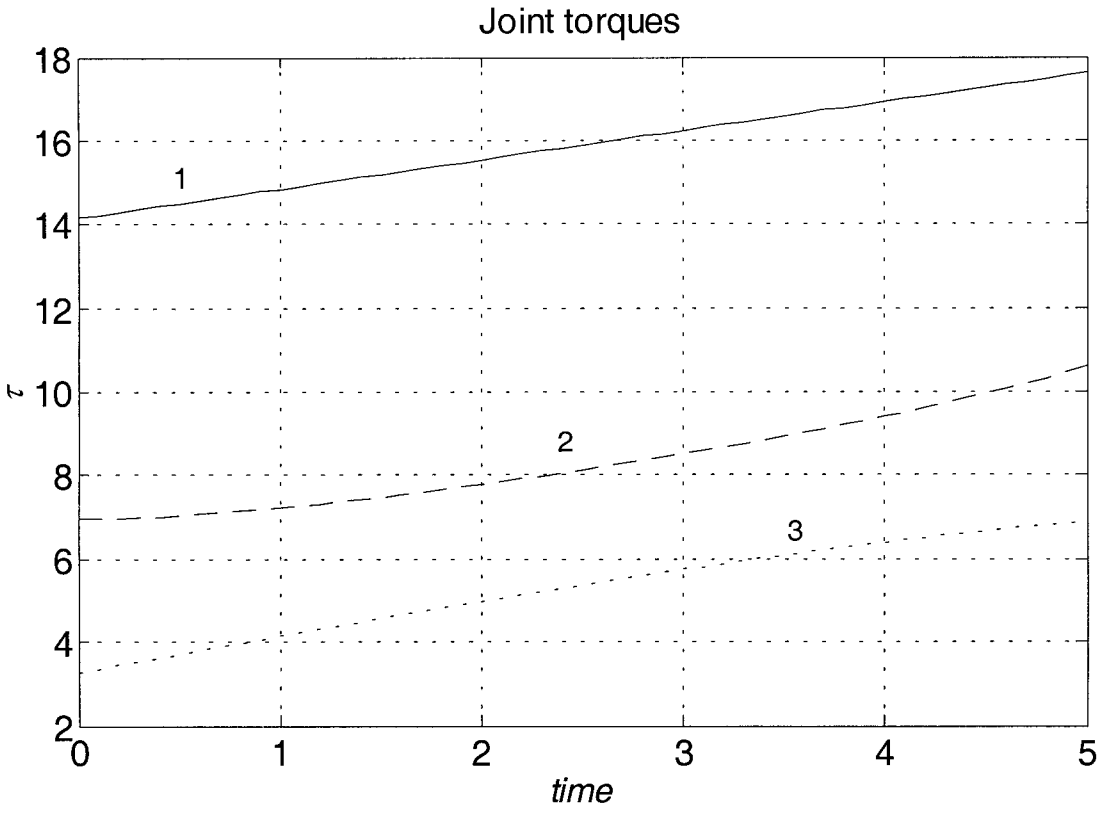
a) solution, parts 1. through 5.











Towards the end of the simulated time, this robot is approaching the elbow-straight singularity. If the simulation time goes beyond 5 *sec*, the Jacobian matrix determinant will approach zero, and the joint rates will approach infinity. For this exercise, this occurs after t=5.4 *sec*, to the nearest one-tenth second. The elbow-straight singularity corresponds to a loss in one translational freedom, in the direction of links 1 and 2 aligned in a straight line.

b) Check your Jacobian matrix results for the initial and final joint angle sets.

```
%  Exercise 5 using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

% Constants
DR = pi/180;
L1 = 4;  L2 = 3;  L3 = 2;

% DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0],'mod');      % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0],'mod');
L{3} = link([alp(3),a(3),th(3),d(3),0],'mod');

ThreeR = robot(L, 'Plan3R');                        % Create 3R robot object

q0 = [10 20 30]*DR;                                 % Initial angles
qf = [12.84 15.84 -27.12]*DR;                       % Final angles (from my Matlab solution)

% Jacobian {3} wrt {0}, expressed in {0}
J_00 = jacob0(ThreeR,q0);                           % Initial
J00  = J_00(1:2,:); J00 = [J00;J_00(6,:)];          % Extract 3x3 matrix
J_f0 = jacob0(ThreeR,qf);                           % Final
Jf0  = J_f0(1:2,:); Jf0 = [Jf0;J_f0(6,:)];          % Extract 3x3 matrix
```

Using this Matlab program, the following Jacobian matrices result:

$$\text{Initial: } {}^0J_3 = \begin{bmatrix} -2.19 & -1.50 & 0 \\ 6.54 & 2.60 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{Final: } {}^0J_3 = \begin{bmatrix} -2.33 & -1.44 & 0 \\ 6.53 & 2.63 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

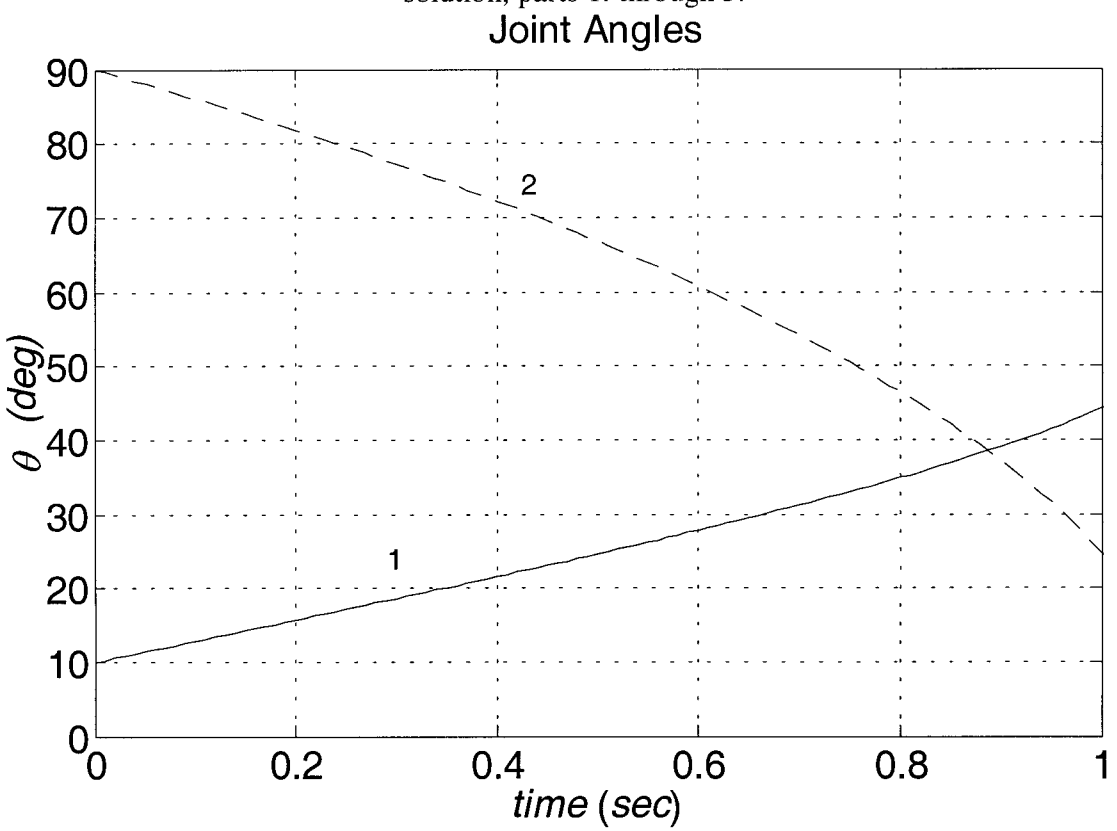
Note the Corke toolbox results in the Jacobian matrix relating the motion of {3} with respect to {0}, expressed in {0} coordinates. For the Jacobian matrix expressed in {3} coordinates, use function *jacobn()* instead of *jacob0()*. The above matrices do not correspond to the symbolic Jacobian matrix given in the problem statement since that was for motion of {*H*} with respect to {0}, expressed in {0} coordinates; the rate equations for the Corke results are (motion of {3}, not {*H*}):

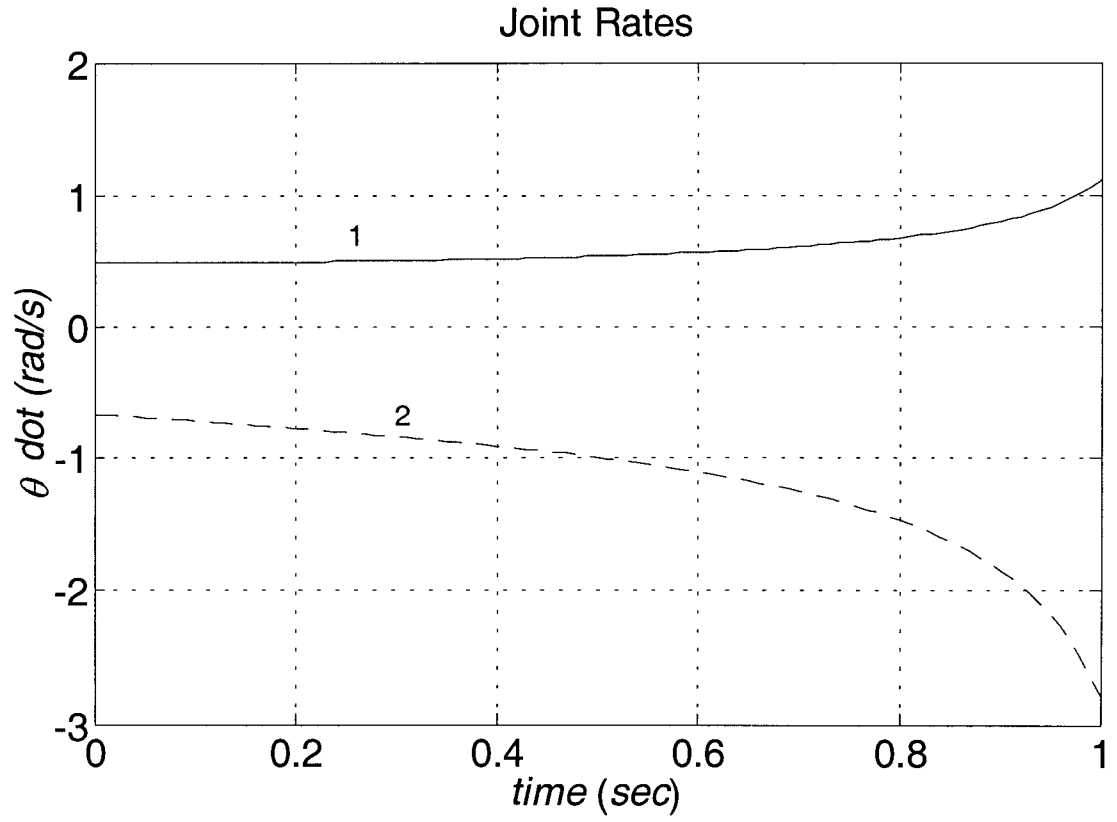
$$ {}^0\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \omega_z \end{Bmatrix}_3 = \begin{bmatrix} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} & 0 \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} $$

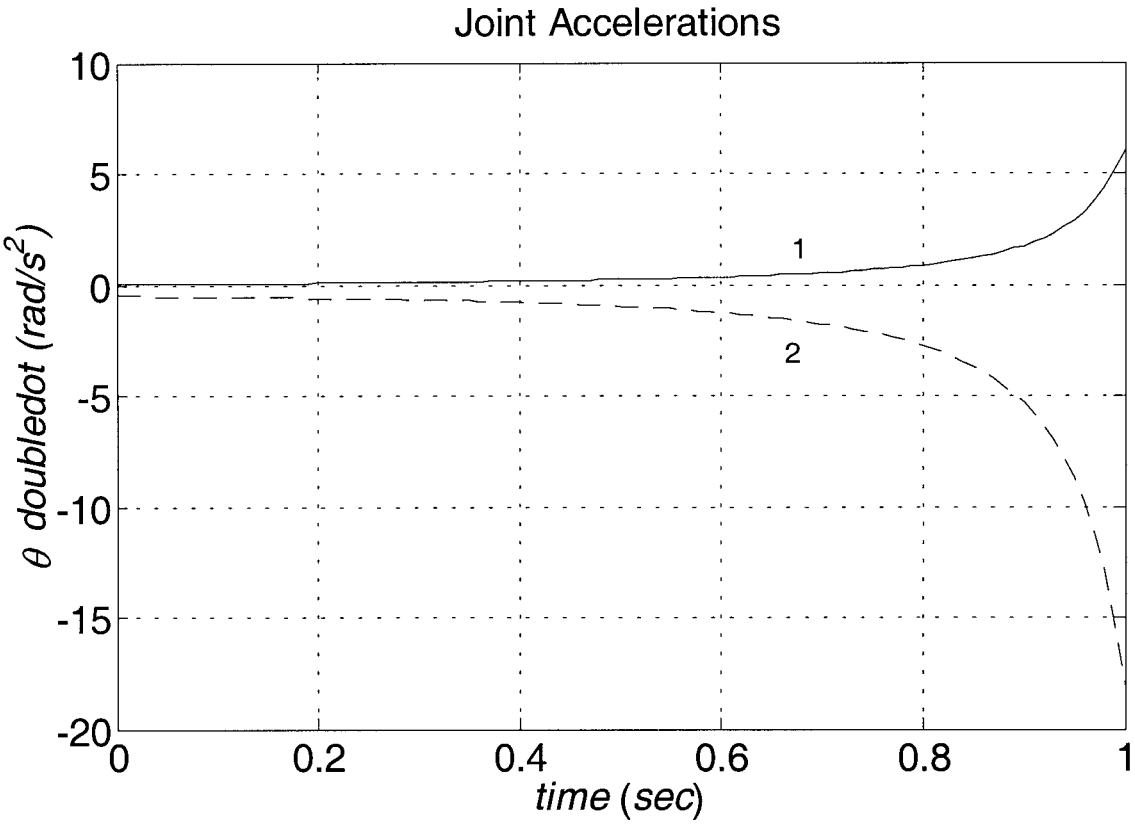
So, when using the Corke toolbox Jacobian matrices, one must first numerically transform Cartesian velocities and force/moment vectors from {*H*} to {3} using the rigid-body velocity and static force transformations given in the book (Section 5.11).

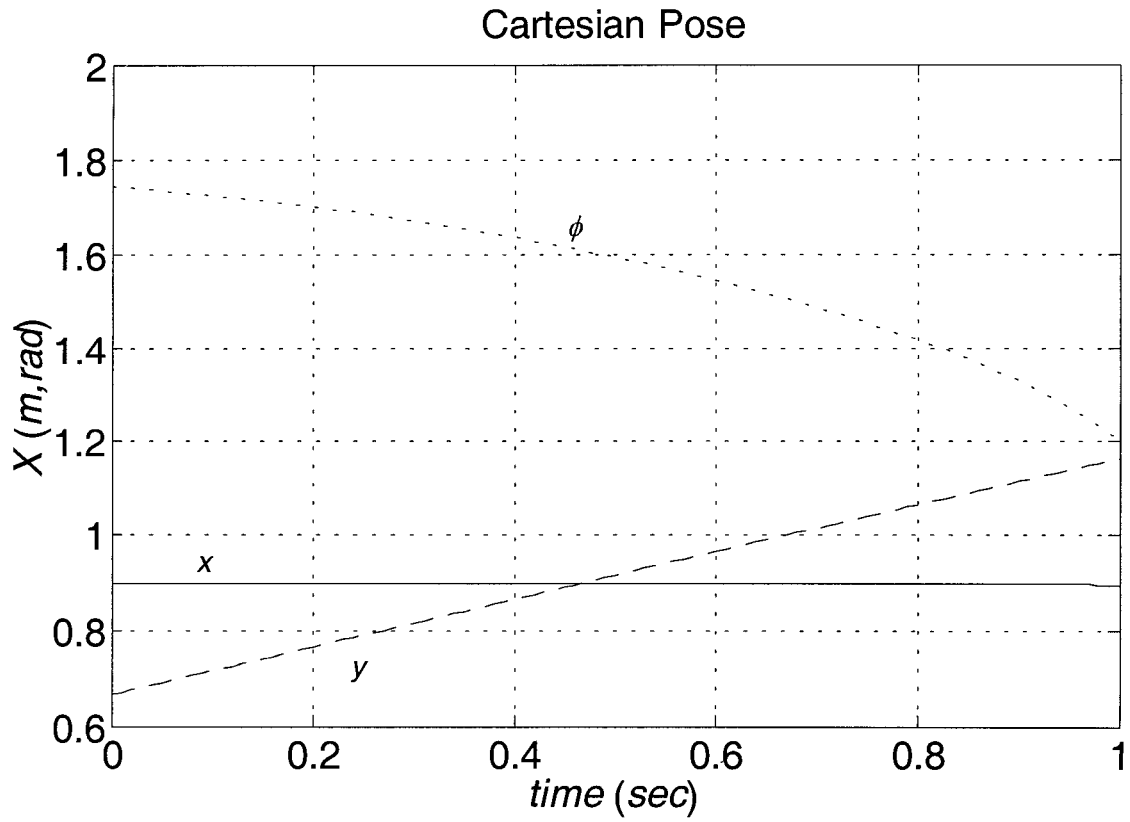
Matlab Exercise 6A

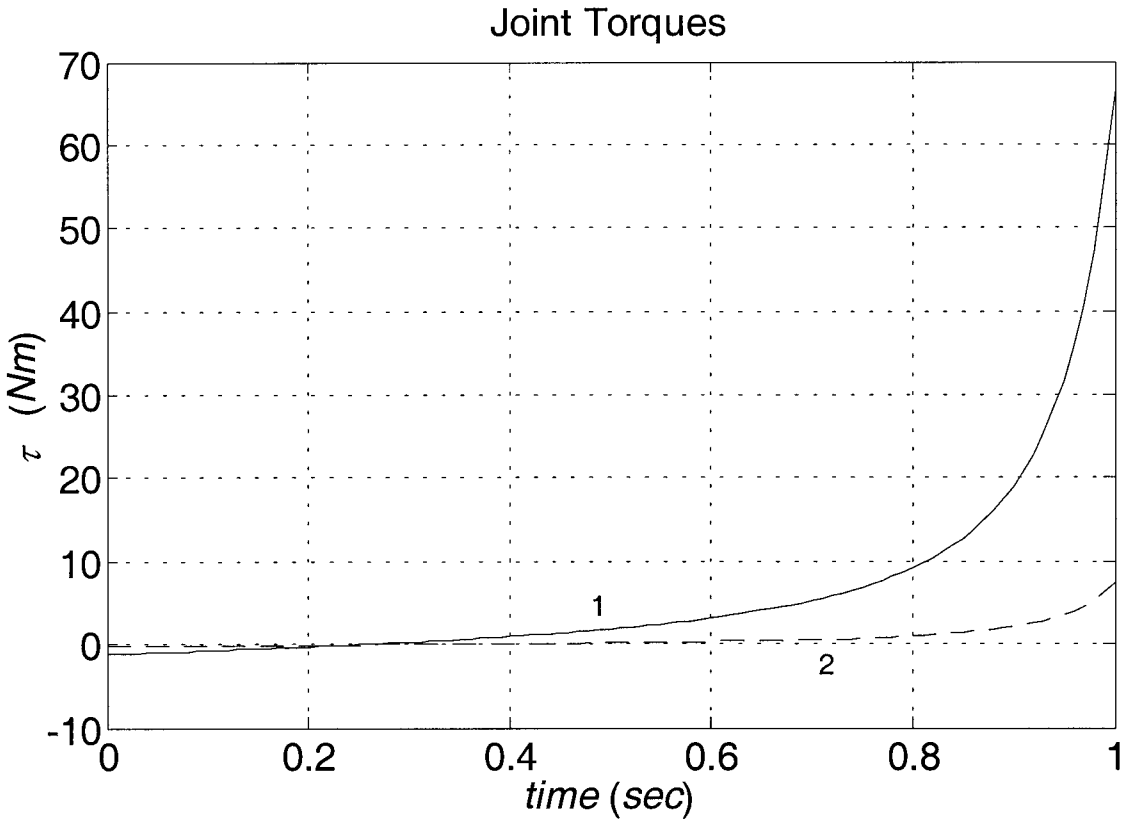
solution, parts 1. through 5.







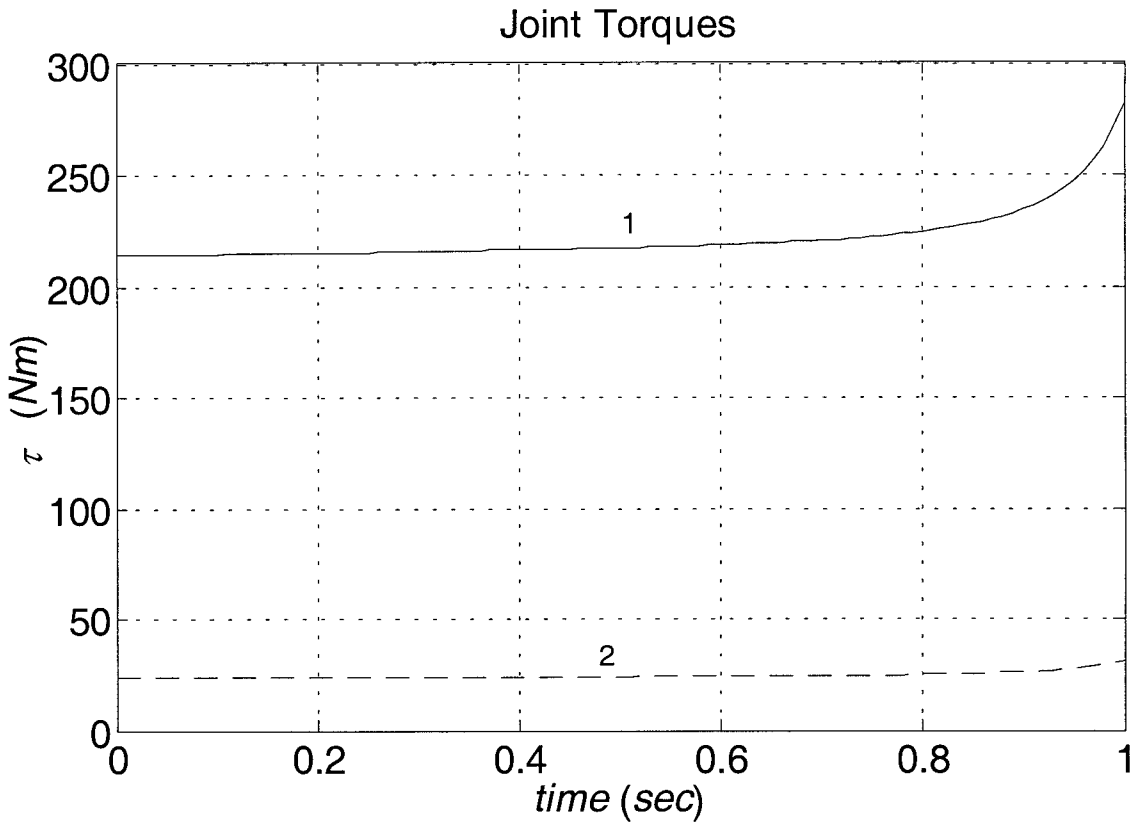




Without Gravity Effect

Towards the end of the simulated time, this robot is approaching the elbow-straight singularity. If the simulation time goes beyond 1 *sec*, the joint rates, accelerations, and torques will approach infinity. For this exercise, this occurs after t=1.09 *sec*, to the nearest one-hundredth second.

The above joint torque results are for robot dynamics without gravity (i.e. the plane of motion is normal to gravity). The next plot shows the joint torque results for robot dynamics considering gravity (i.e. g is in the negative Y direction). All of the other plots are identical.



With Gravity Effect

Matlab Exercise 6B

a) For the given robot, parameters, and motion condition, the recursive Newton-Euler inverse dynamics solution yields the following required driving joint torques:

$$\mathrm{T} = \begin{Bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{Bmatrix} = \begin{Bmatrix} 853.6 \\ 637.2 \\ 296.8 \end{Bmatrix} (Nm)$$

As a matter of interest, the required joint torques to resist gravity only (i.e. static condition, no joint velocities or accelerations) are given below. These gravity joint torques may be easily verified using FBDs and methods of statics. The above answer gives the total joint torques, i.e. gravity plus dynamics torques.

$$\mathrm{T}_g = \begin{Bmatrix} \tau_{g1} \\ \tau_{g2} \\ \tau_{g3} \end{Bmatrix} = \begin{Bmatrix} 1847.6 \\ 495.1 \\ 49.1 \end{Bmatrix} (Nm)$$

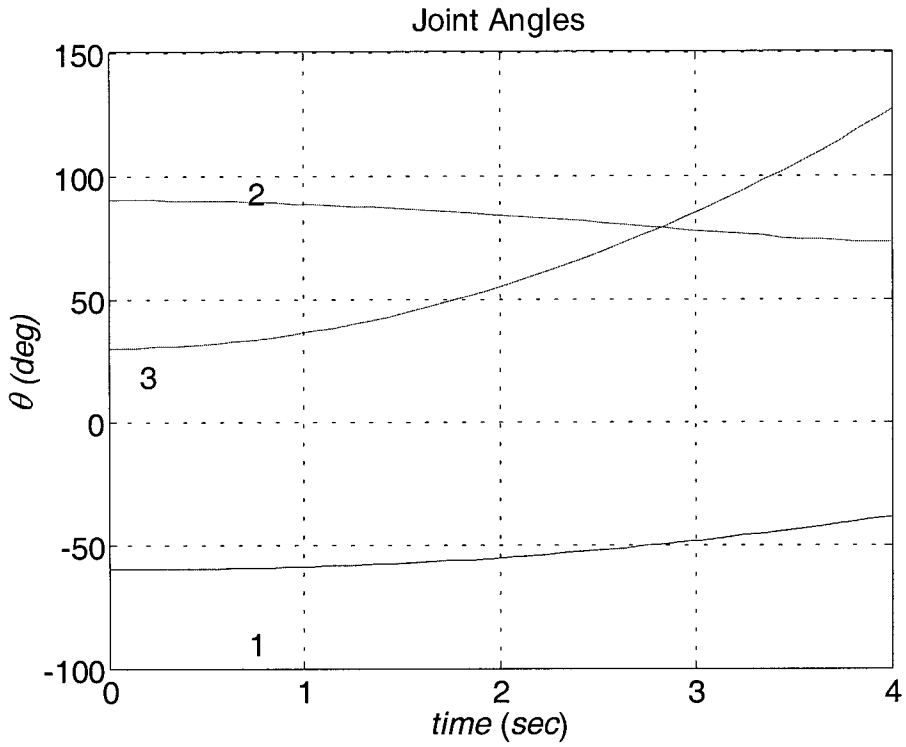
b) Check your a) results with the Corke Matlab Robotics Toolbox.

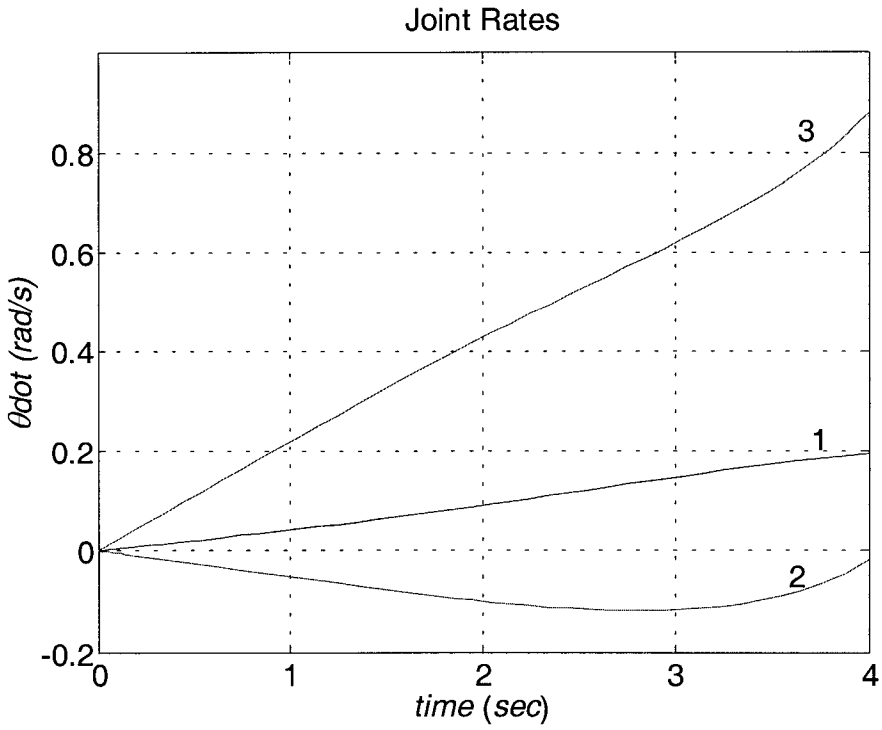
```
%  Exercise 6B using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

%        Constants
DR = pi/180;
L1 = 4;  L2 = 3;  L3 = 2;

%  DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0,20,L1/2,0,0,0,0,0.5,0,0,0,0,1,0,0,0]);   % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0,15,L2/2,0,0,0,0,0.2,0,0,0,0,1,0,0,0]);   % 'help DYN' to see what each term is
L{3} = link([alp(3),a(3),th(3),d(3),0,10,L3/2,0,0,0,0,0.1,0,0,0,0,1,0,0,0]);

L{1}.mdh = 1;    L{2}.mdh = 1;    L{3}.mdh = 1;    %  Specify modified (Craig) DH convention

ThreeR = robot(L, 'Plan3R');              %  Create 3R robot object

grav = [0;9.81;0];                        % To simulate g down, must accelerate entire robot up by same g mag

Q = [10 20 30]*DR;                        % Assigned motion conditions
Qd = [1 2 3];
Qdd = [0.5 1 1.5];

tau = rne(ThreeR, [Q Qd Qdd], grav)       % Newton-Euler recursion to calculate total dynamics joint torques

tau_G = gravload(ThreeR,Q,grav)           % Calculate gravity joint torques only
```

Using this Matlab program, the same joint torques result as given above. Note the gravity vector must be specified as g in the $+Y$ direction as explained in the comment for *grav* above and also in Chapter 6.

Matlab Exercise 6C

Forward dynamics solution given initial conditions and constant joint torques:





Exercise 6C Corke Matlab Robotics Toolbox Program

```
%  Exercise 6C using the Corke Matlab Robotics Toolbox
%      Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

%        Constants
DR = pi/180;
L1 = 4;  L2 = 3;  L3 = 2;

%  DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0,20,L1/2,0,0,0,0,0.5,0,0,0,0,1,0,0,0]);  % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0,15,L2/2,0,0,0,0,0.2,0,0,0,0,1,0,0,0]);  % 'help DYN' to see what each term is
L{3} = link([alp(3),a(3),th(3),d(3),0,10,L3/2,0,0,0,0,0.1,0,0,0,0,1,0,0,0]);

L{1}.mdh = 1;
L{2}.mdh = 1;
L{3}.mdh = 1;

ThreeR = robot(L, 'Plan3R');       %  Create 3R robot object

t0 = 0;  tf = 4;
N=40;    dt = (t1-t0)/N;
t = [t0:dt:tf];                    %  Specify time array
Q0 = [-60 90 30]*DR;               %  Initial joint angles and rates
Qd0 = [0 0 0];

[tsim,Q,Qd] = fdyn(ThreeR, t0, tf, 'torqfun', Q0, Qd0);          %  Solve forward dynamics numerically


% torqfun.m Function for Bob6c.m, torque profiles for forward dynamics of planar 3R

function tau = torqfun(t,Q,Qd)
tau = [20; 5; 1];                  %  Given constant joint torques
```

This program was used to generate the required forward dynamics simulation results for the given planar 3R robot. Function 'torqfun' is used to specify the given constant driving joint torques. The initial joint angles and rates are clearly visible in the resulting plots.

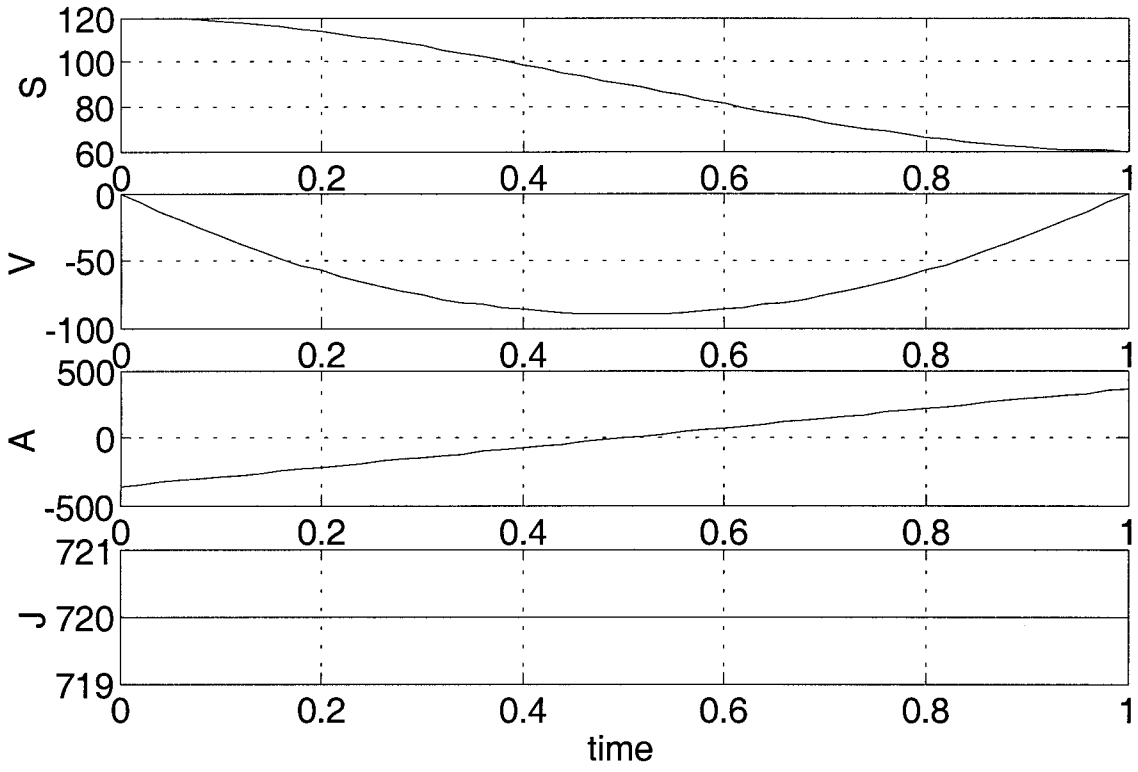
Matlab Exercise 7

a) *Third-order polynomial.*

Result:

$$\theta(t) = 120t^3 - 180t^2 + 120$$
$$\dot{\theta}(t) = 360t^2 - 360t$$
$$\ddot{\theta}(t) = 720t - 360$$
$$\dddot{\theta}(t) = 720$$

(Note: *deg* units throughout)



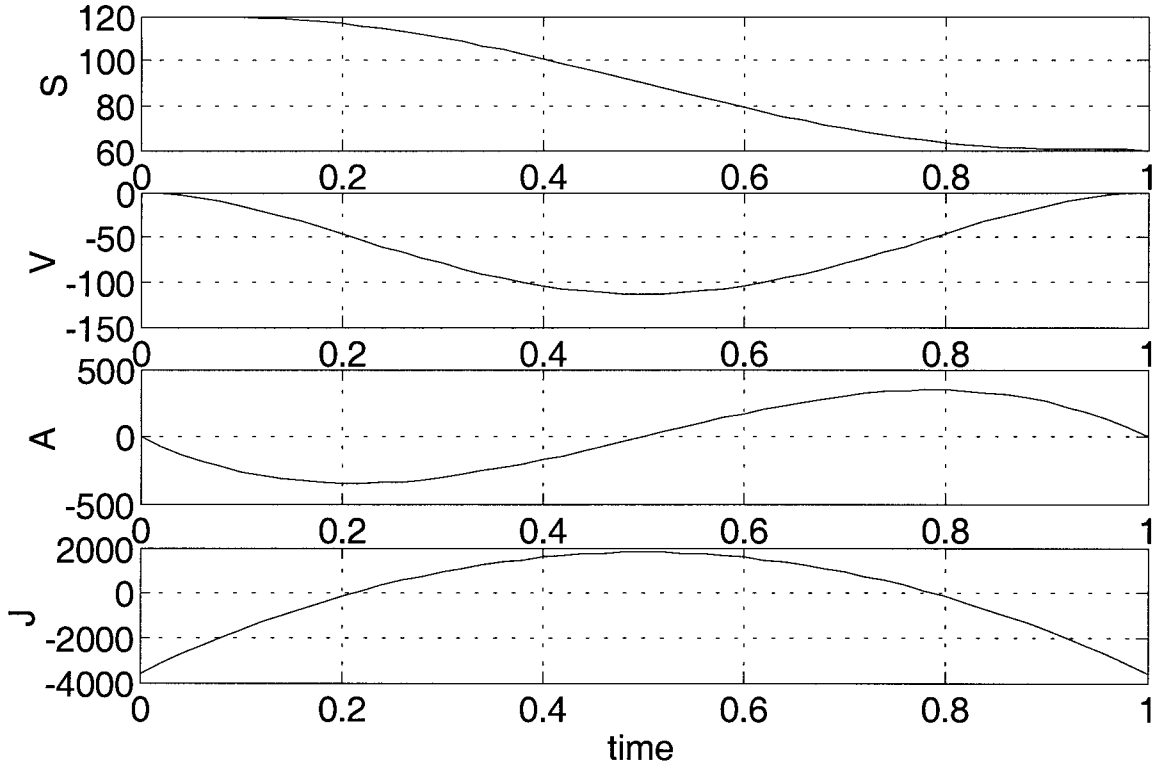
Note: though the Jerk J looks fine (constant of 720, the slope of the A plot), there is an infinite spike (negative) at the start and also at the finish. This is due to the finite change in A with zero change in time at those time locations. To fix this potential problem, use a 5th order polynomial (next page).

b) *Fifth-order polynomial.*

Result:

$$\theta(t) = -360t^5 + 900t^4 - 600t^3 + 120$$
$$\dot{\theta}(t) = -1800t^4 + 3600t^3 - 1800t^2$$
$$\ddot{\theta}(t) = -7200t^3 + 10800t^2 - 3600t$$
$$\dddot{\theta}(t) = -21600t^2 + 21600t - 3600$$

(Note: *deg* units throughout)



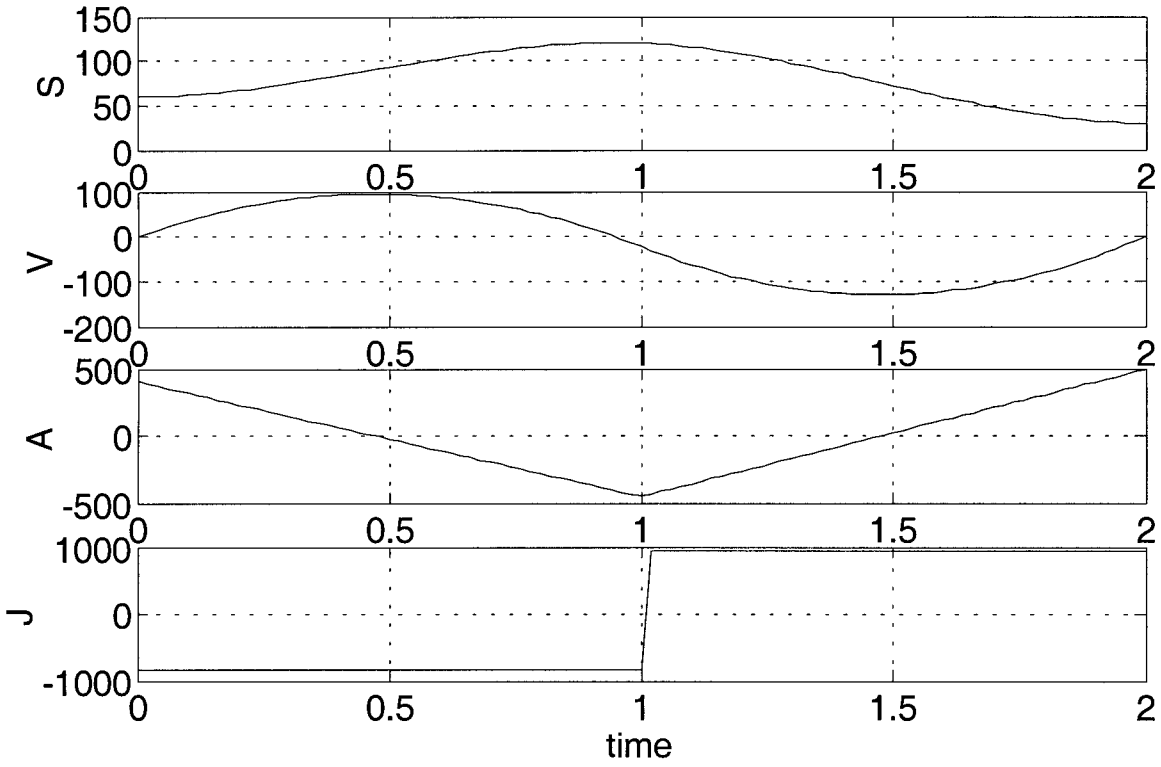
Now there is a finite jump in J at the start and finish (the slope of A is non-zero at the start and finish), but there are no longer infinite spikes in J at the start and finish. The cost is a higher-order polynomial. Note the slope of V starts and ends at zero (zero A at the start and finish) as required; this was not the case with the third-order polynomial since it was not required.

c) *Two third-order polynomials with via point.*

Result:

$$\theta_1(t) = -142.5t^3 + 202.5t^2 + 60 \qquad \theta_2(t) = 157.5t^3 - 225t^2 - 22.5t + 120$$
$$\dot{\theta}_1(t) = -427.5t^2 + 405t \qquad \dot{\theta}_2(t) = 472.5t^2 - 450t - 22.5$$
$$\ddot{\theta}_1(t) = -855t + 405 \qquad \ddot{\theta}_2(t) = 945t - 450$$
$$\dddot{\theta}_1(t) = -855 \qquad \dddot{\theta}_2(t) = 945$$

(Note: *deg* units throughout)



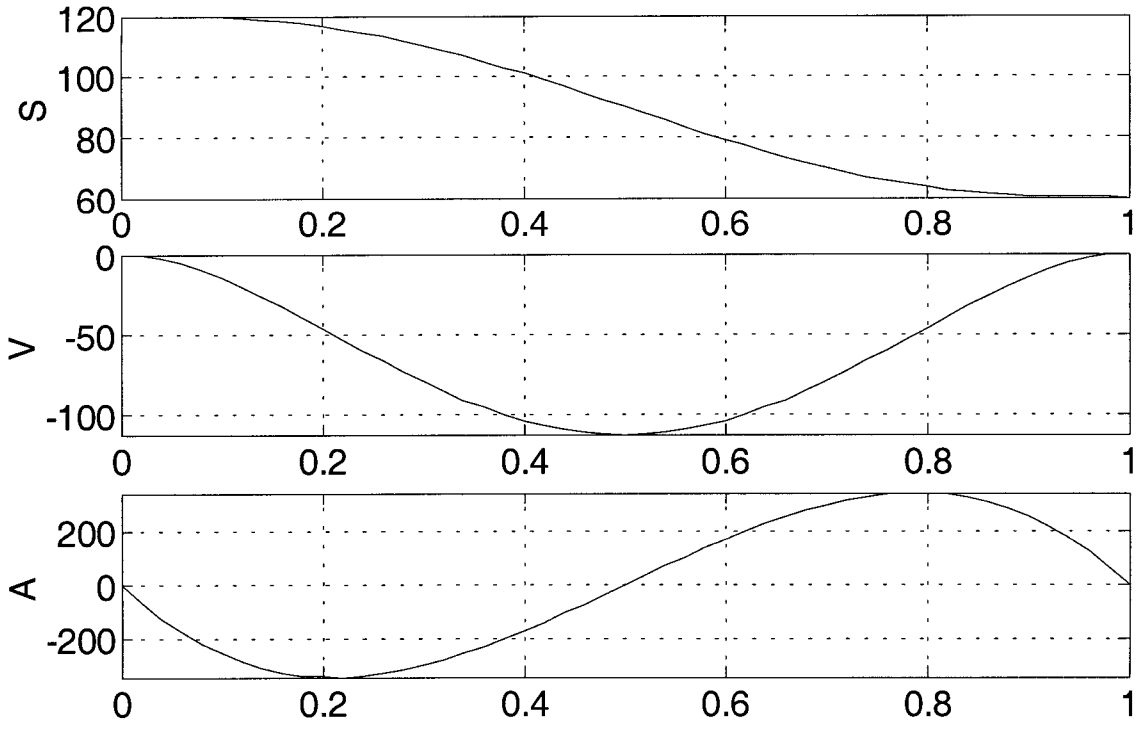
Though it is difficult to see on the S plot, the slope of S is not flat at the via time; i.e., the velocity is not zero at this point since it is not required. S peaks at 120.57° (slightly higher than the required $\theta_v = 120°$) just before the via time, so that S is sloping downwards at the via time.

The Jerk J has the same potential problem as with the 3rd order polynomial: there is an infinite spike in J at the start and also at the finish. Since we matched A at the via point (in addition to V), the Jerk has a finite jump at the via point, but no infinite spike.

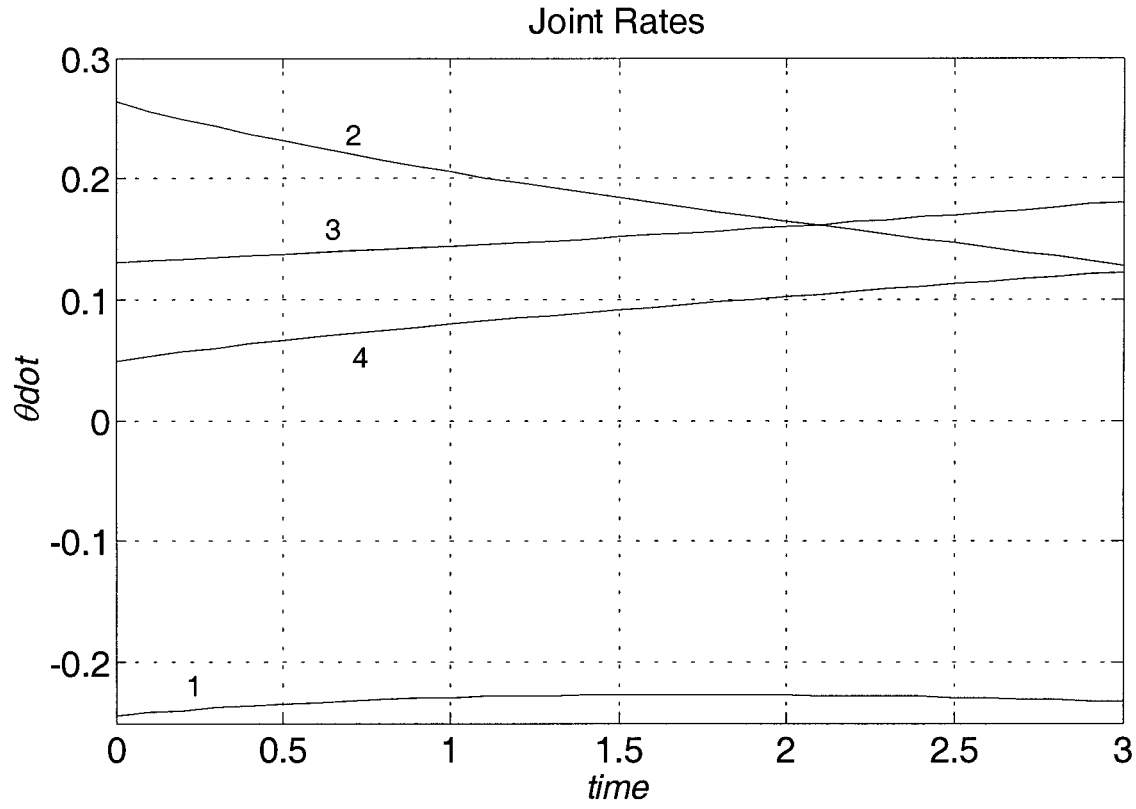
d) Check the a. and b. results with the Corke Matlab Robotics Toolbox.

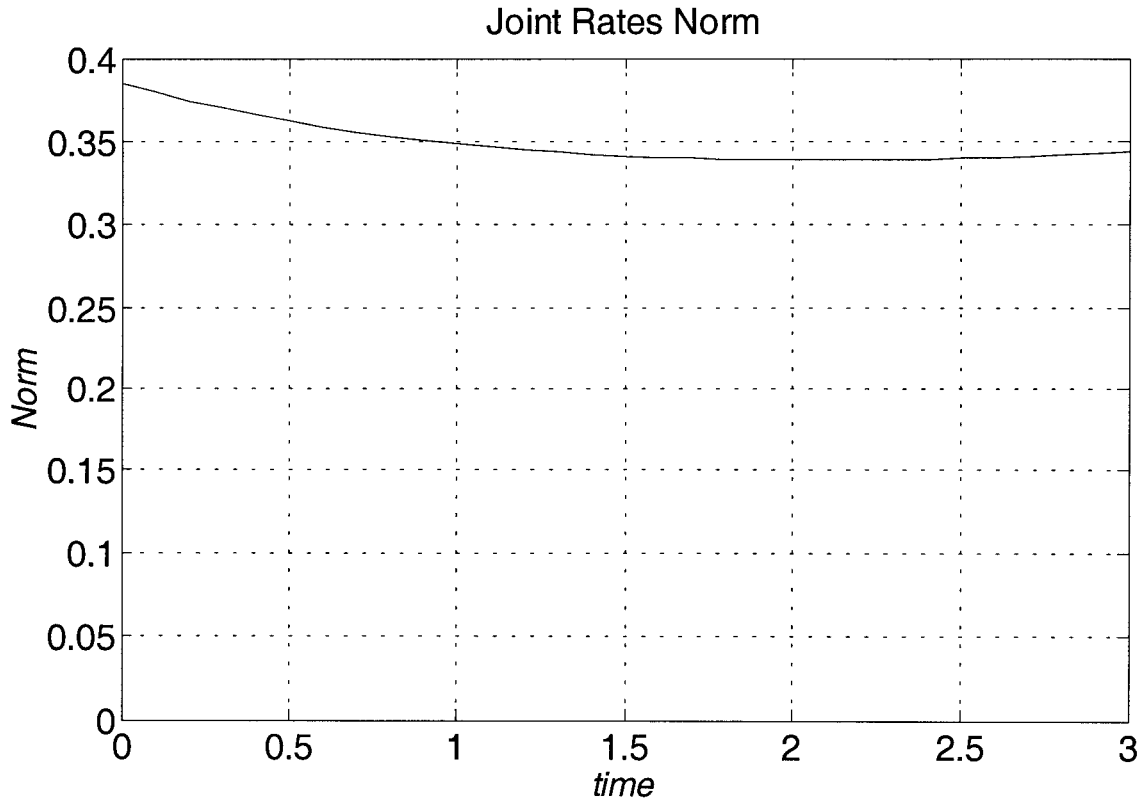
```
%  Exercise 7 using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;
tf = 1;  n = 50;  t = [0:tf/n:tf];      % User-defined time array
q0 = 120;  qf = 60;                     % Initial and final joint angles, single joint; degree units throughout

[q qD qDD] = jtraj(q0, qf, t) ;         % Calculate 7th-order joint trajectory

figure;                                 % Plot results
subplot(311); plot(t,q,'k');
set(gca,'FontSize',16);
grid; axis([0 tf min(q) max(q)]); ylabel('S');
subplot(312); plot(t,qD,'k');
set(gca,'FontSize',16);
grid; axis([0 tf min(qD) max(qD)]); ylabel('V');
subplot(313); plot(t,qDD,'k');
set(gca,'FontSize',16);
grid; axis([0 tf min(qDD) max(qDD)]); ylabel('A');
```

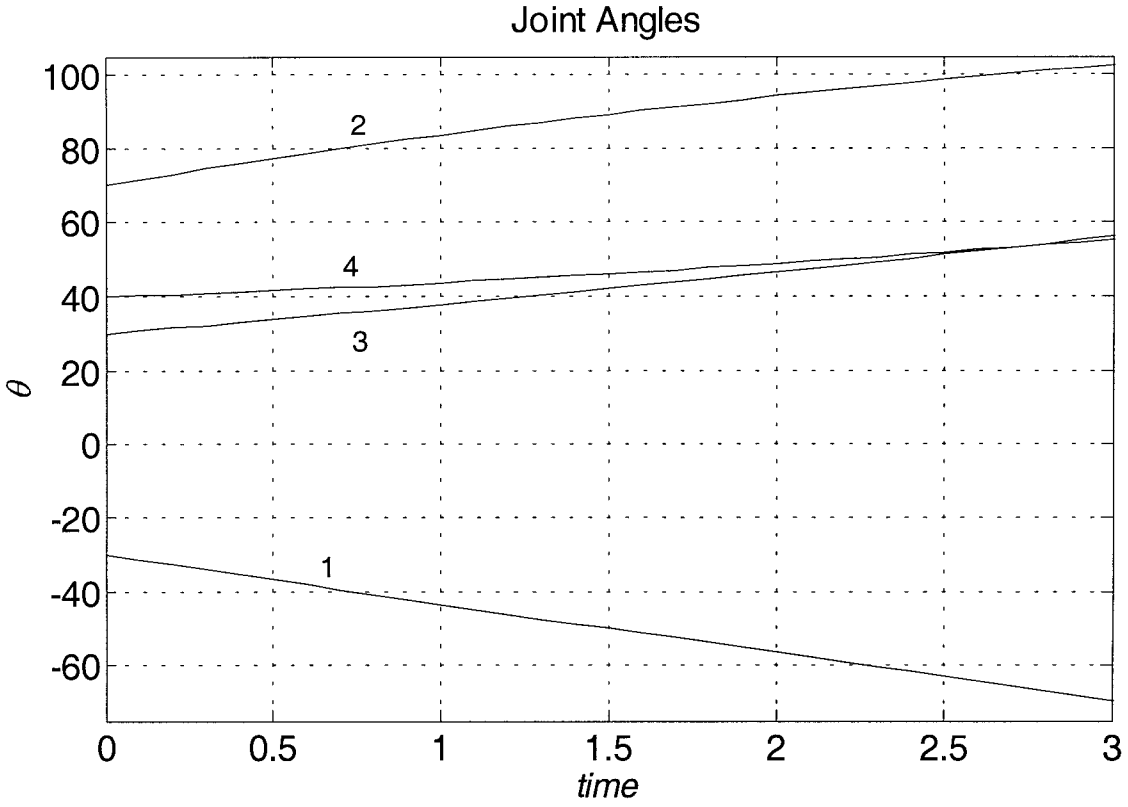
Using this Matlab program, the following joint trajectory plots result. Note that the toolbox only gives us access to the angle (S), angular rate (V), and angular acceleration arrays(A) (i.e., no angular jerk). The toolbox uses a 7^{th}-order polynomial for joint trajectories, so the results should compare closely, but not exactly, with the b) results; the results are different from the a) results due to the 3^{rd}-order polynomial's inability to provide zero angular acceleration at the start and end. The 7^{th}-order polynomial should allow for zero jerk at the start and end; however, this does not appear to be the case from the plot since the slope of A appears to be non-zero at the start and end.

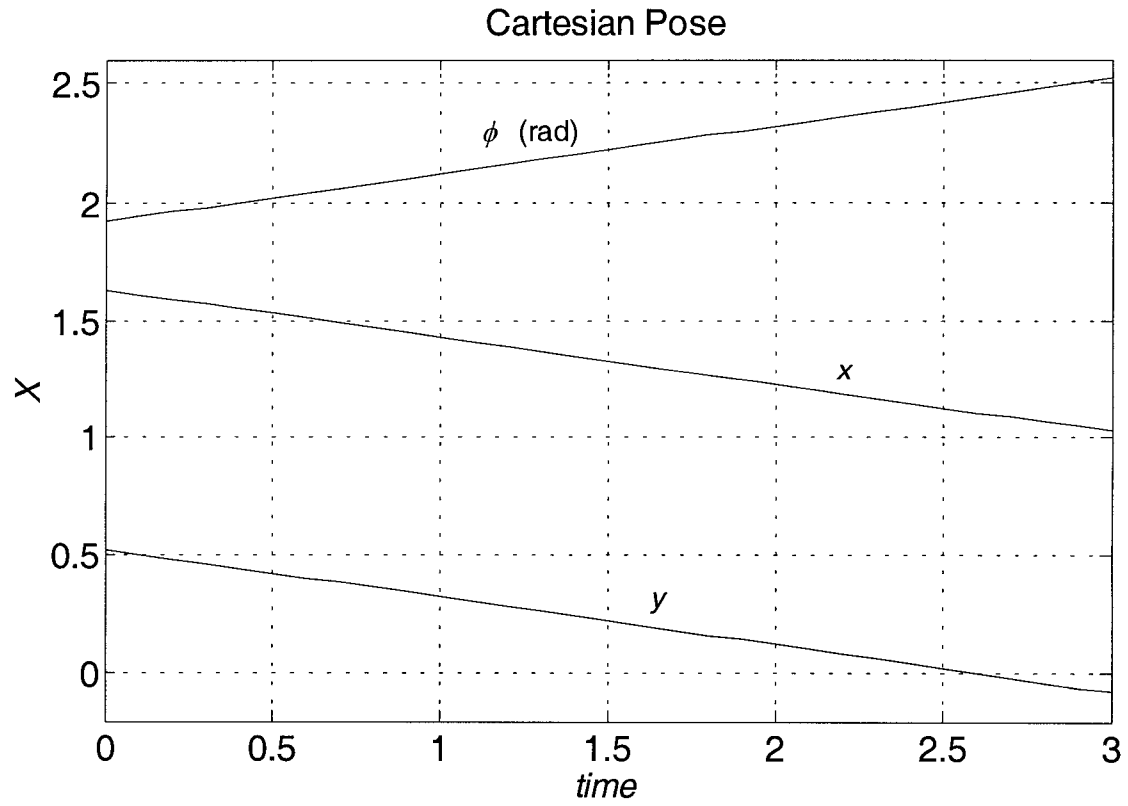


Matlab Exercise 8









b) Check your Jacobian matrix results for the initial and final joint angle sets.

```
%  Exercise 8 using the Corke Matlab Robotics Toolbox
%     Dr. Bob Williams, Ohio University, 2/2002

clc; clear;

% Constants
DR = pi/180;
L1 = 1;  L2 = 1;  L3 = 0.2;  L4 = 0.2;

%  DH parameters
alp(1) = 0;  a(1) =  0;  d(1) = 0;  th(1) = 0;
alp(2) = 0;  a(2) = L1;  d(2) = 0;  th(2) = 0;
alp(3) = 0;  a(3) = L2;  d(3) = 0;  th(3) = 0;
alp(4) = 0;  a(4) = L3;  d(4) = 0;  th(4) = 0;

L{1} = link([alp(1),a(1),th(1),d(1),0],'mod');      % R joints and Craig DH convention
L{2} = link([alp(2),a(2),th(2),d(2),0],'mod');
L{3} = link([alp(3),a(3),th(3),d(3),0],'mod');
L{4} = link([alp(4),a(4),th(4),d(4),0],'mod');

FourR = robot(L, 'Plan4R');                         % Create 4R robot object

q0 = [-30 70 30 40]*DR;                             % Initial angles
qf = [-69.65  102.62  56.15  55.26]*DR;             % Final angles (from my Matlab solution)

% Jacobian {4} wrt {0}, expressed in {0}
J_00 = jacob0(FourR,q0);                            % Initial
J00  = J_00(1:2,:); J00 = [J00;J_00(6,:)]           % Extract 3x4 matrix
J_f0 = jacob0(FourR,qf);                            % Final
Jf0  = J_f0(1:2,:); Jf0 = [Jf0;J_f0(6,:)]           % Extract 3x4 matrix
```

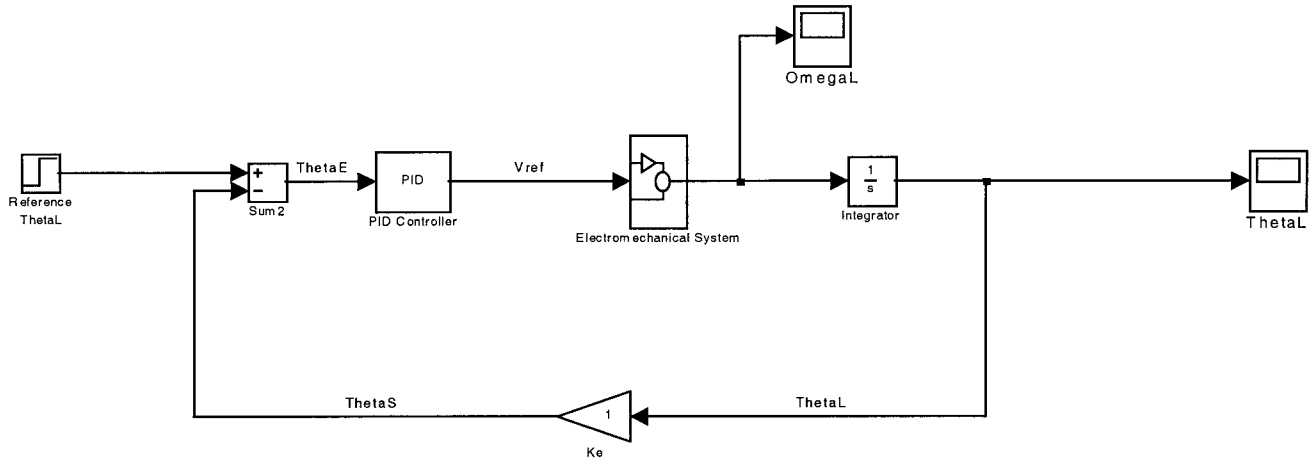
Using this Matlab program, the following Jacobian matrices result:

$$\text{Initial: } {}^0J_4 = \begin{bmatrix} -0.33 & -0.83 & -0.19 & 0 \\ 1.70 & 0.83 & 0.07 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \text{Final: } {}^0J_4 = \begin{bmatrix} 0.19 & -0.74 & -0.20 & 0 \\ 1.19 & 0.84 & 0.00 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

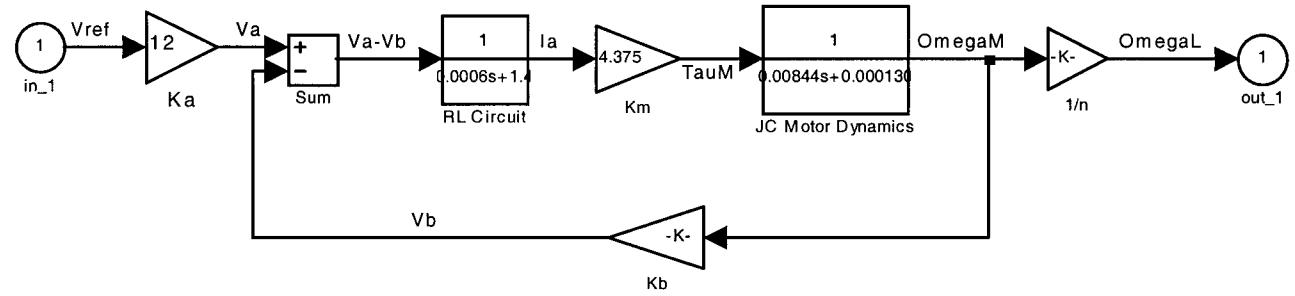
Note the Corke toolbox results in the Jacobian matrix relating the motion of {4} with respect to {0}, expressed in {0} coordinates. For the Jacobian matrix expressed in {4} coordinates, use function *jacobn()* instead of *jacob0()*. This issue is similar to that from Matlab Exercise 5. When using the Corke toolbox Jacobian matrices, one must first numerically transform Cartesian velocities from {*H*} to {4} using the rigid-body velocity transformation given in the textbook (Section 5.11).

Matlab Exercise 9

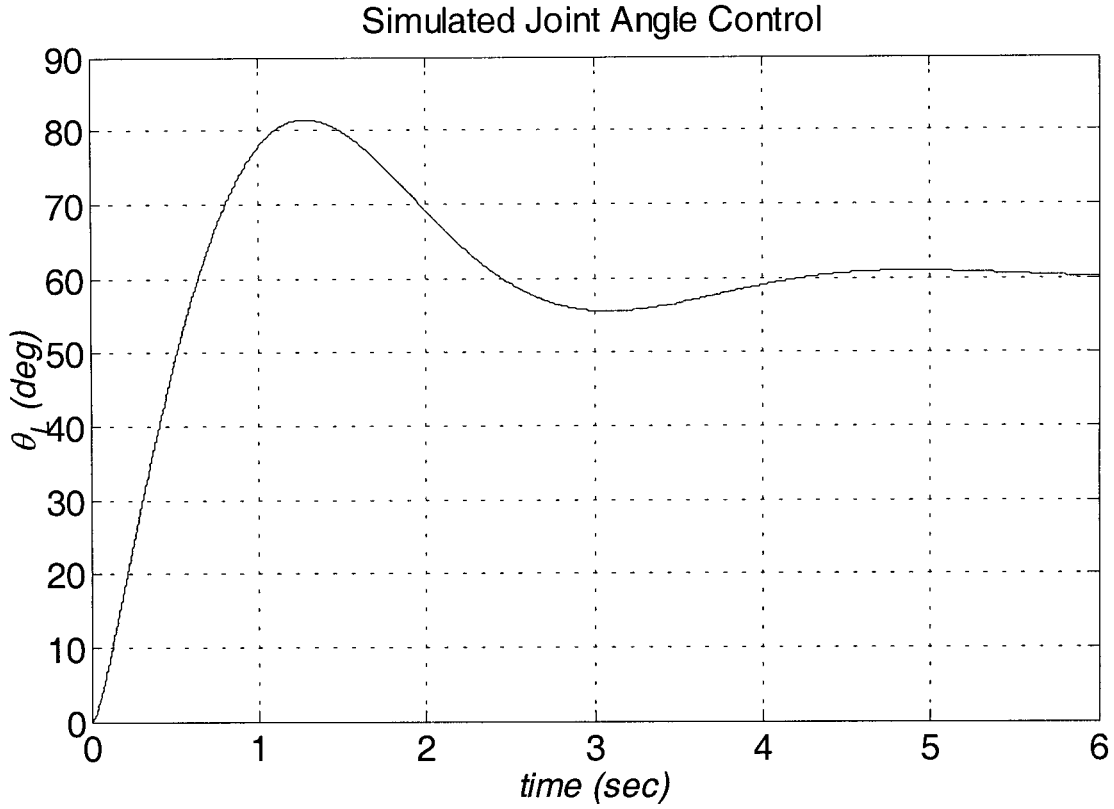
Here is the Simulink implementation of the closed-loop feedback diagram:



where the Electromechanical system is the open-loop system:

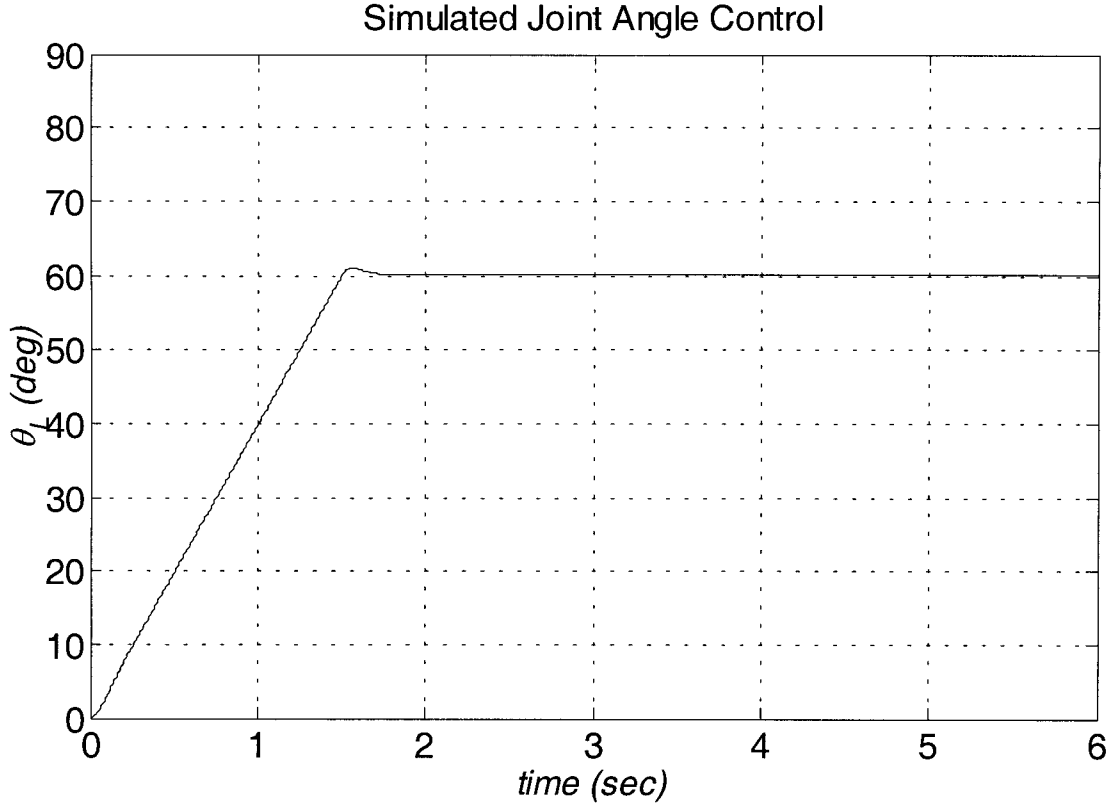


Now, for trial-and-error PID controller design with the step input, there are infinite solutions for "good" performance. One possible solution is $K_P = 2$, $K_I = 4$, $K_D = 1$; the resulting load angle output from the closed-loop feedback control simulation is shown below:



To plot the control effort (armature voltage V_a with back emf V_b on the same graph), it is an easy matter to connect the lines V_a and V_b on the Electromechanical system diagram to a multiplexer (Mux) and then send the output of the Mux to a scope; this result is not shown for any cases.

1) As mentioned in the problem statement, it is very frustrating to do PID controller design for the step input in this case; the above result is unacceptable for robot control; the percent overshoot is too big and the settling time is too high. Therefore, let us use the ramped step input. The controller diagrams are the same as those given above, but replace the step input with a ramped step input (HINT: turn on a ramp at time 0 with a slope of +40; then add another ramp starting at time 1.5 *sec* with a slope of –40). For this case, "good" values for PID gains are $K_P = 16$, $K_I = 8$, $K_D = 1$; the resulting load angle output from the closed-loop feedback control simulation is shown below:

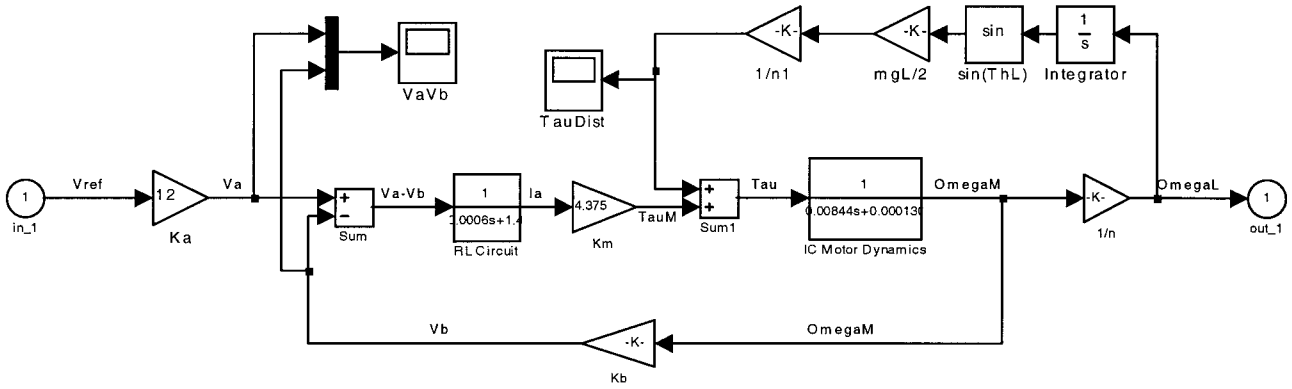


As seen in the above figure, the control of the output shaft angle is much improved when using a reasonable input, the ramped step. There is a small transient effect at the start and a small percent overshoot at the 1.5 sec time where the ramp ends, but otherwise this simulated control output plot overlays perfectly with the input command. The settling time is also much improved.

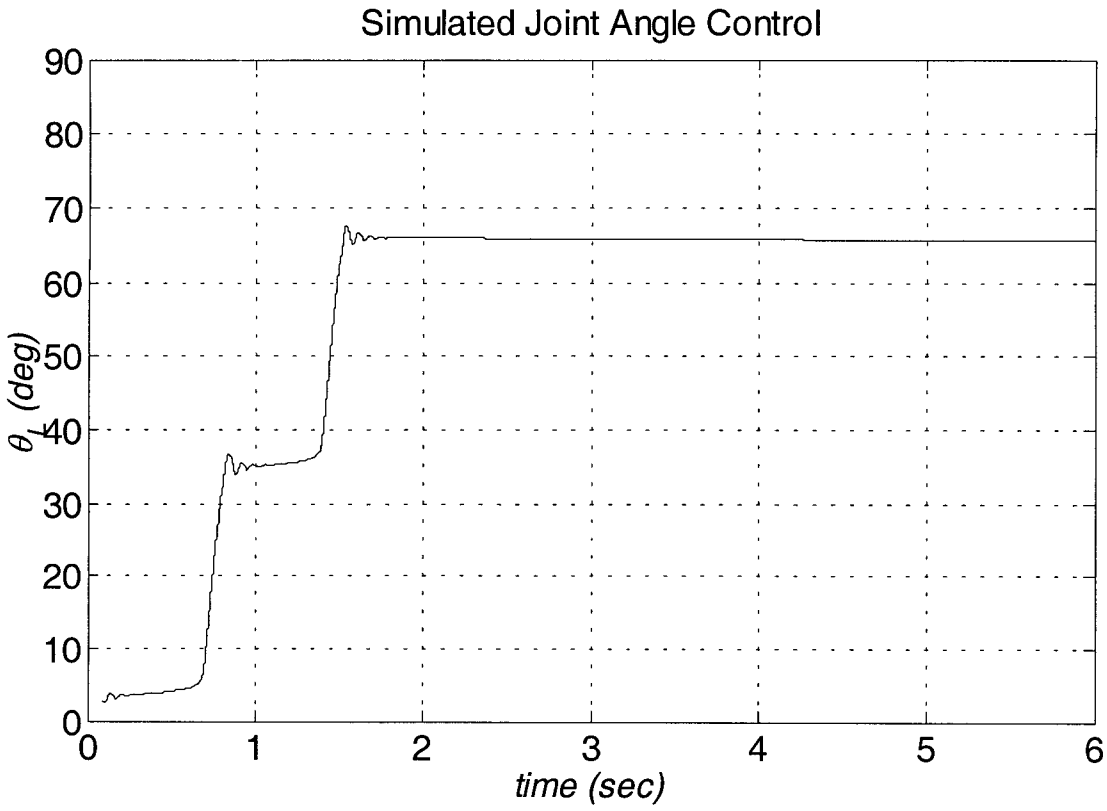
2) Usually the electrical system time constant $\frac{L}{R}$ is small relative to the mechanical system time constant $\frac{J}{C}$. This means that when voltage $V_a(t)$ is applied to the armature circuit, the armature current $i_a(t)$ rises much faster than the motor shaft angular velocity $\omega_M(t)$ does when $i_a(t)$ is applied to generate motor torque $\tau_M(t)$. Therefore, the modeling of the inductor is not significant in this system and L can be ignored in favor of R; this is equivalent to ignoring the circuit dynamics and including only Ohm's law. This can be verified via Simulink simulation.

3) This is left to the student – simply increase J_L and C_L by the same factor and re-run the simulation several times, keeping the same PID gains from 1). Note the degradation in performance as the parameters increase.

4) From a statics free-body diagram, the disturbance torque acting at the load shaft is $mgL\sin\theta_L/2$. The gear ratio reduces this effect further by $1/n$ as felt at the motor shaft. The modified Electromechanical system diagram to include the gravity disturbance at the motor shaft is shown below:



Without changing the previous best PID gains from 1), the resulting load angle output is shown below from the closed-loop feedback control simulation considering the gravity disturbance:



Obviously, the gravity disturbance has a large effect; the simulated transient response for the ramped step input is much degraded and there is significant steady-state error (the final angle is 65.5° instead of the commanded 60°). Thus, the PID gains should be re-designed considering this effect.