# Chapter 3 HW Hints

**Problem 3.1.** This is the same planar robot that's used for the Programming Exercises, so you can make use of this work for your programs also. The link frame assignments are shown in Figure 3.7, with the corresponding parameters in Figure 3.8. Simply use the general link transformation matrix of text equation (3.6) to get all the transforms, then multiply them all (in the correct order, of course) to get the last link frame relative to the base frame, that is, ${}^0_n\mathbf{T}$.

**Problem 3.4.** The frame placement for this manipulator is relatively simple; just follow the rules. After you place the frames, obtain the table of link parameters (like text Figure 3.11), then derive the link transforms ${}^{i-1}_{i}\mathbf{T}$ (text equation (3.6)).

Then, as I indicated in my 9/15/09 email, find an expression for the vector ${}^S\mathbf{P}_T$, which locates the origin of the tool frame $\{T\}$ relative to the station frame $\{S\}$ (in the same manner as in Problem 3.9). You will need the transform relating the $\{S\}$ frame and frame $\{0\}$, but you can get this by inspection of the Figure 3.30.

**Problem 3.8.** This should be another transform equation problem, which you've already done. Please use *my* method of frame diagrams (as in the Chapter 2 HW).

**Problem 3.9.** By inspection of Figure 3.32 you can find position vector ${}^2\mathbf{P}_{tip}$, and since position vectors are "fixed" vectors you can change relativity with $\mathbf{T}$ matrices. Please find the solution using the $\mathbf{T}$ matrix approach, not simple geometry (although you may wish to check your result with geometry).

**Problem 3.14.** This can be a tricky problem. I'd suggest first finding the angle $\alpha$ between the two axes (remember that we don't like using $\sin^{-1}$ or $\cos^{-1}$), then find the length $a$ along the common perpendicular (normal).

*HINT:* Two vector products with which you're familiar are the key to this problem. Specifically,

- *Cross Product:* The cross product is an operation on two vectors in a three-dimensional Euclidean space that results in another vector which is perpendicular to the plane containing the two input vectors. The cross product is defined by the formula $\mathbf{a} \times \mathbf{b} = ab \sin\theta\, \hat{\mathbf{n}}$, where $\theta$ is the measure of the smaller angle between $\mathbf{a}$ and $\mathbf{b}$, $a$ and $b$ are the magnitudes of vectors $\mathbf{a}$ and $\mathbf{b}$, and $\hat{\mathbf{n}}$ is a unit vector perpendicular to the plane containing $\mathbf{a}$ and $\mathbf{b}$ in the direction given by the right-hand rule.
- *Dot Product:* The dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$ yields the magnitude of $\mathbf{a}$ in the direction of $\mathbf{b}$, with a minus sign if the direction is opposite. This is called the *scalar projection* of $\mathbf{a}$ onto $\mathbf{b}$, or *scalar component* of $\mathbf{a}$ in the direction of $\mathbf{b}$. The dot product $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$.

I suggest drawing a figure of the two axes, and applying your reasoning about the meaning of the parameters $\alpha$ and $a$ and the vector products above.

**Programming Exercises.** You should be able to use the results of Exercise 3.1 in the Programming Exercise. The syntax of MATLAB function `kin()` should be:

```
% WrelB = kin(theta).  This function accepts a 3x1 column vector of joint
% angles "theta" in radians and computes the corresponding pose (in "internal"
% form) of the WRIST frame relative to the BASE frame, that is, WrelB.
%

function WrelB = kin(theta)

...(MATLAB code goes here)...

WrelB = [ ... ];   % Transform of Wrist relative to Base (WrelB)
```

In the same manner, the syntax of MATLAB function `where()` should be:

```
% TrelS = where(theta,TrelW,SrelB).  This function computes the pose (in
% "internal" form) of the {TOOL} frame relative to the {STATION} frame.  The
% function also accepts the transforms (in "internal" form) of the {TOOL}
% frame relative to the {WRIST} frame (TrelW, the "tool offset"), and the
% transform of the {STATION} frame relative to the {BASE} frame (SrelB, the
% "station frame offset").  Note that function where() will call function kin().

function TrelS = where(theta,TrelW,SrelB)

...(MATLAB code goes here)...

TrelS = [ ... ];   % Transform of Tool relative to Station (TrelS)
```

**Program Results.** Using the second set of joint angles given in the text, I obtained the following results:

```
>> chpt3
Enter joint angle vector in degrees, like this [th1 th2 th3]: [-23.6 -30.3, 48.0]

TrelS_u =

    0.9728
   -0.7155
   24.1000
```

**Some observations:**

- In the description of the Programming Exercise step 2, there is an error. The text states "The values of ${}^{W}_{T}\mathbf{T}$ and ${}^{S}_{B}\mathbf{T}$ should be stored..." This is *incorrect*; it should have been "${}^{W}_{T}\mathbf{T}$ and ${}^{B}_{S}\mathbf{T}$..."
- In function `kin()` you can compute the individual ${}^{i}_{i-1}\mathbf{T}$ matrices symbolically, then multiply them numerically within `kin()`—less work by hand and the slightly reduced computational efficiency is unimportant.
- If someone else confirms this result, let me know (although I think it's correct).
- When entering the joint angles in response to my prompt in `chpt3.m`, you *must* include the brackets [ ] when entering them in response to the prompt. I tried to make that clear in the prompt the user sees. That's because I used the MATLAB `input()` function.
- Actually, you can draw this thing pretty easily and verify it (it's a good learning experience). In fact, if you **DO** draw it, give yourself **ONE** point extra credit! Woo-hoo...