How to Calculate a Robot's Forward Kinematics in 5 Easy Steps

by Alex Owen-Hill. Last updated on Dec 19, 2024
Posted on Nov 25, 2015 in Robots
9 min read time

Calculating the Forward Kinematics is often the first step to using a new robot. But, how do you get started?

While there are some good tutorials available online, up until now there hasn't been a simple step-by-step guide for calculating Forward Kinematics.

Here's a simple guide to calculating the kinematics of any robotic manipulator.

[Since I first published this article in 2015, it has become one of our most popular articles ever! I've since updated and improved it, but the core simplicity remains the same..]

Calculating kinematics is a cornerstone skill for robotics engineers. But, kinematics can sometimes be a pain (e.g. understanding the difference between forward and inverse kinematics).

When I first started working in robotics research, I was often told: *"go and calculate the Forward Kinematics of this robot".* The phrase is basically robotics research shorthand for *"go and get familiar with this robot".*

Calculating the forward kinematics is the vital first step when using any new robot in research, particularly for manipulators.

Even though I had learned the theory of kinematics in university, it wasn't until I had calculated various kinematic solutions for a few real robots that the whole process started to feel intuitive. Even then, because I was not calculating kinematics every day I had to go back to my notes to remind myself how to do it every time I encountered a new robot.

It would have been really helpful to have a step-by-step guide of which stages to go through. That way, I wouldn't have to read through hundreds of pages of academically written equations in textbooks.

A sort of kinematics "cheat sheet" would have been useful.

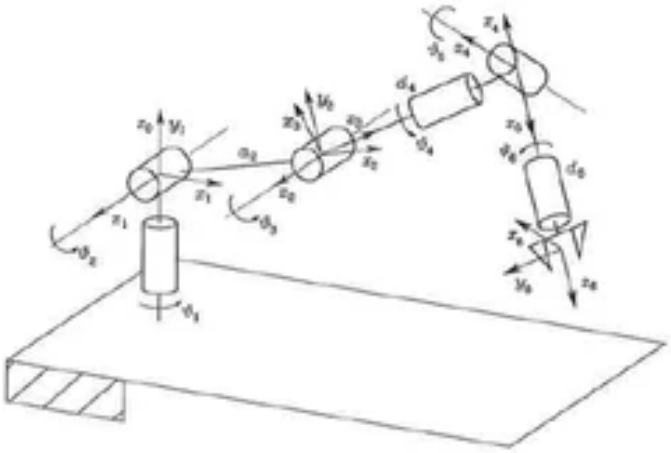
This post is exactly that cheat sheet.

I'll primarily focus on the Devanit-Hartenberg (DH) approach to Forward Kinematics, as it's the most common.

I hope you enjoy it!

Step 1: Get a pencil and paper

It can be tempting to jump straight for your computer when starting with a new robot. However, even if the robot looks like a "standard" 6R manipulator (the most common robot type) I always sit down with a pencil and paper to draw out the kinematic diagram.



Anthropomorphic arm with spherical wrist

This simple task forces you to carefully consider the actual physical configuration of the robot, avoiding false assumptions that can wreak havoc later on during coding.

There are various ways to draw a kinematic chain. Pick whichever style you prefer.

I favor simple cylinders for the revolute joints and lines for the links, as shown in the image. Do a Google Image Search for "kinematic diagram" and see some of the different styles available.

As you draw, work out which way each joint moves and draw this motion as double-ended arrows onto the diagram.

Step 2: Figure out your axes

The next key step is to draw the axes onto each joint. The DH approach assigns a different axis to each movable joint.

If you set up your axes correctly then working with the robot will be easy. Set them up incorrectly and you will suffer countless headaches. These axes will be required by simulators, inverse kinematic solvers, and your colleagues on your team (nobody wants to solve a Forward Kinematic solution if someone else has already done it).

Have a look at this video to see how to set them up:

The two important axes to work out are:

- Z-axis — The z-axis should lie on the axis of rotation for a revolute joint or axis of extension for a prismatic joint.
- X-axis — The x-axis should lie along the "common normal", which is the shortest orthogonal line between the previous z-axis and the current z-axis (seriously, watch the video).
- Y-axis — Once you've calculated the other two, this axis should fall into place by following the "right hand rule" (see below).

Personally, I draw the axes using the following coloring: z-axis (blue), x-axis (red) and y-axis (green). Incidentally, this is the coloring scheme that's used in the RViz visualizer from ROS, which I used extensively during my PhD.

Back in my undergraduate days, our lecturer encouraged us to make an axis "sculpture" out of three colored straws stuck into a sphere of blue-tack to explain the theory to us. Though this might seem a bit "playschool", it can be very helpful as you can position the sculpture next to the physical robot to make sure you've got the axes pointing in the right direction. For a virtual version of this, check out this interactive tool.

Alternatively, you can use the "right hand rule".

The right hand rule

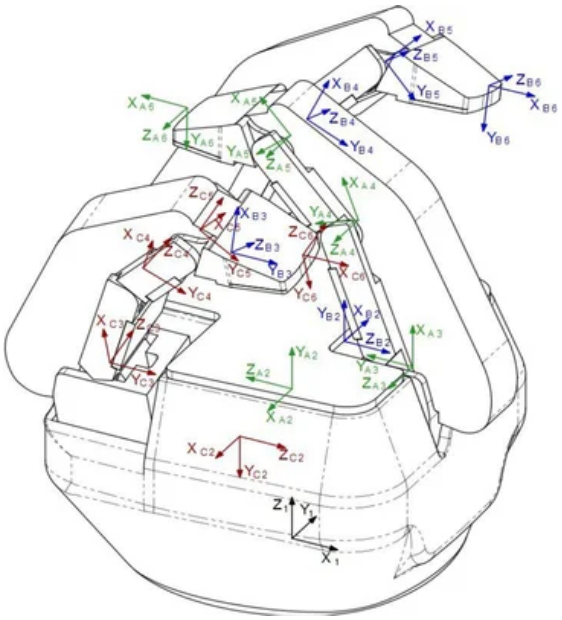
A quick and easy way to remember the direction of your y-axis is to follow the right hand rule. This is a mnemonic (memory aid) used extensively in physics to remember the orientations of 3-dimensional axis.

To use it, hold out your right hand in front of you, sticking out your thumb, index finger, and middle finger at 90 degrees to each other. Each finger then corresponds to an axis:

- Thumb = z-axis.
- Index finger = x-axis.
- Middle finger = y-axis.

By orienting your thumb and index finger to follow the z and x axis of the robot joint, your middle finger will naturally fall into the direction of the y-axis.

Step 3: Remember your end effector



The goal of calculating the Forward Kinematics is to be able to calculate the end effector pose from the position of the joints.

Most Forward Kinematic tutorials will generalize the end effector as a single distance from the final joint. This is fine for a simple "open-close" gripper. However, as modern grippers are often more complicated than this, it's worth considering how the end effector operates.

For example, the Robotiq 3-Finger Adaptive Gripper has a few different gripping modes. Each mode will correspond to a slightly different desired end effector pose. If you want to pinch the object between its fingers, this will require a different distance than if you wanted to wrap the fingers around the object.

You should always consider the end effector carefully when formulating the kinematic model.

Step 4: Calculate the DH parameters

Denavit-Hartenberg (DH) parameters are often required to enter the robot model into a simulator and start performing any sort of analysis on it.

The best way to visualize the DH parameters is to watch the video I already included above.

The DH parameters break down each joint of the robot into four parameters, each taken with reference to the previous joint. They are calculated in reference to the "common normal" described above. Note that if the previous z-axis intersects the current z-axis, which is often the case, the common normal has a length of zero.

- *d* - the distance between the previous x-axis and the current x-axis, along the previous z-axis.
- *θ* - the angle around the z-axis between the previous x-axis and the current x-axis.
- a (or r) - the length of the common normal, which is the distance between the previous z-axis and the current z-axis
- α - the angle around the common normal to between the previous z-axis and current z-axis.

Go through each joint on your drawing and write down the DH parameters for each joint. Each joint should have one value which is a variable, representing the actuated joint.

For a more detailed explanation and some examples, I recommend this handout by Peter Corke or this chapter from Introduction to Robotics.

Alternatives to the DH parameters

The DH approach is the most common approach to Forward Kinematics, but it's not perfect. One of its failings is that it doesn't handle parallel z-axes very elegantly. There are various alternatives, including Screw Theory representations, Hayati-Roberts, and other geometric modelings (see this paper for a comparison). These may (or may not) be better approaches. However, most kinematic libraries do accept the DH parameters and for that reason, it's a reasonable approach to begin with.

Step 5: Combine parameters into a whole robot

The final step is to combine all of your DH parameters into an entire robot. There are two ways to do this, a hard way and an easy way:

The hard way: Create your own solver

The "purist" method of using the DH parameters is to "roll your own" Forward Kinematic solver using your favorite programming language. I've taken this approach myself in the past, though I probably wouldn't do it these days.

Once you have your DH parameters for each joint, you can use this method to code it into a Forward Kinematics solver:

1. Find a library in your programming language which allows you to do matrix multiplication. Alternatively, code your own using the methods in this list.
2. For each joint of the robot, populate a new 4 x 4 matrix with the following values:

$$T = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & r\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & r\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

3. Multiply all of the matrices together, starting with the first joint all the way up to the end effector.
4. The final T vector will contain the position of the end effector. The R matrix will contain the orientation of the end effector.

If you just want to try this out with some values, without coding your own solver, you can use this handy online tool to create a worked example of a complete robot from its DH parameters.

In my experience, creating your own solver doesn't offer many more benefits than using an existing library. However, it is a good learning exercise.

The easy way: Use existing libraries

A far more effective way to calculate Forward Kinematics, once you've got your DH parameters, is to use an existing library.

There are loads of kinematic software libraries and many of them do far more than just calculate Forward Kinematics. Most of them include Inverse Kinematic solvers, dynamics, visualization, motion planning and collision detection, to name just a few features. These libraries will transform

your DH parameters into matrices, which are then multiplied together to calculate the relationship between joint positions and end effector pose.

Some good development libraries include Robotics Library, Orocos Kinematics and Dynamics Library, ROS MoveIt, OpenRave, RoboAnalyzer, and the Matlab Robotics Toolbox.

Even though you'll usually require Inverse Kinematics to actually control the robot, computing the Forward Kinematics is a necessary step to get familiar with any new robotic arm.

If you found this article useful, make sure to bookmark it so you can find it when you next encounter a new robot!

What is your favorite method of getting familiar with a new robot's kinematics? Do you prefer another method over the DH parameters? Have you got any questions on implementing Forward Kinematics in your robot? Tell us in the comments below or join the discussion on LinkedIn, Twitter or Facebook.

Leave a comment

farid mehr

6/16/2016, 5:00:39 PM

I interest in robotics

Alex Owen-Hill

6/17/2016, 4:18:53 AM

Hi Farid.

I'm glad to hear it. Well, you're in the right place!

Thanks for your comment.
Cheers,
Alex

luay mahmood

4/4/2017, 4:55:57 AM

Dear Mr.Mehr
I would like to control kossel delta 3D printer via PLC so kindly can you tell me about the best way or method to apply delta kinematics to PLC so it can execute g code .
best regards

Alex Owen-Hill

6/2/2017, 11:34:10 AM

Hi Luay,

Thanks for your question. However, I'm afraid I'm not really sure how to answer it. I don't have any experience programming 3D printers with PLCs. You could try asking on a 3D printing forum.

Cheers,
Alex

Nick

6/18/2016, 11:53:46 PM

Will you do a tutorial on rolling your own solver? That sounds cool.

Alex Owen-Hill

6/19/2016, 5:33:24 AM

Hi Nick.

Great suggestion! Sure, I'll have a look and see if we can come up with a good post about making a FK library/solver "from scratch."

It will have to wait until after our Live Blogging of Automatica 2016 though. Only 2 more days to go!

Cheers,
Alex

Shravista Kashyap

7/2/2017, 12:49:49 AM

Suppose, a robot is made up of only two links. Also it does have two revolute joints and an end effector. These links are placed in such a way that two revolute joint's axis are perpendicular to each other. To assign coordinate frames to its joints, it is placed in such a way that 'z1' axis is along the axis of revolution of joint 1 and 'x1' axis is perpendicular to 'z1' axis and along the joint 2(or parallel to 'z2' axis i.e, parallel to the axis of rotation to the joint 2). So now how to place axis 'x2' to the joint 2 where 'z2' is the axis of revolution to the joint 2, so that DH rules are followed.
and this could be the answer for obvious reason.
And it is as follows:
x2 should be perpendicular to both z1 and z2. So if z1 is pointing up on the page, x1 is pointing right on the page, and z2 is pointing right on the page, x2 should either be coming out of the page, or going into the page.

But now I have a question for you right now as given below.
f it so, then if theta1 is joint variable to joint axis 'z1' and theta2 is joint variable to axis 'z2'. Then according to the DH rules, to calculate the parameters for link 2, what will be the value of joint variable?
To compute DH matrix.
Can you please explain the above to solve it.

Alex Owen-Hill

4/3/2018, 3:54:08 AM

Hi Shravista,

Thanks for your interesting question.

The first part of your question is perhaps easiest to explain by pointing you toward the wikipedia page of the “right-hand rule” - I find that using my hands as the axes makes it very straightforward to visualise where the axes should be pointing. As long as you use the same rule for both axes, the answer should be easy.
https://en.wikipedia.org/wiki/Right-hand_rule
However, either way (left-hand or right-hand rule) you will find that x2 is coming out of the page.

The second half of your question feels a bit like a homework question, so I am dubious about working the answer out for you. I would suggest that you go through this post again, as it explains how you can generate the DH parameters with the information you have.

Cheers,
Alex

Nikesh Bisht

2/5/2018, 11:57:56 PM

Hi Nick,
Thank you for the great tutorial.
I think there's a slight mistake in defining the'd' parameter.
"d - the distance between the previous x-axis and the current x-axis, along the common normal".
Should it not be along the previous 'z' axis?

Machel Allen

4/30/2020, 4:15:21 PM

Could you provide me with your email address?

I will send the sample exam questions and proposed solution

Alex Owen-Hill

4/3/2018, 4:08:37 AM

Hi Nikesh,

It’s Alex, actually :)

You’re right! Thanks a lot. I hadn’t spotted that typo (nor indeed had anyone else, which is a bit worrying).

I have updated it.

Absolutely, the new x axis IS the common normal, so what it said didn’t make much sense.

Thanks!
Alex

Danie Sandova

6/19/2018, 11:05:44 AM

Thank you for the captivating article

Gbenga Odesanmi

4/8/2020, 5:11:19 AM

thanks for this writeup, please I am trying to do teleoperation for position control, can you refer me to any

implementation material. thanks

Sourabh Karmakar

6/16/2020, 11:07:33 PM

This is a very simple way of explanation and easy to learn by any new student in robotics. I referred several books and videos, but this stepwise elaboration made everything very clear at one go! I am a new student in robotics and started working in 6 DOF parallel robots. Do you have any such simple way of doing forward and reverse kinematics for parallel robots? I referred to a lot of books so far, but the vector calculations are still making me confused.

Mr. Inquisitive

9/22/2020, 12:34:35 PM

Sir, I have not yet understood.... can you help me out?

Justin Vaughn

11/28/2020, 10:24:40 PM

great article but you should check your links because most of them are broken or lead to pages that no longer exist.

Roel

12/4/2020, 6:27:09 AM

Thanks for the example, really helped me a lot to figure this out. Went for the hard way and created my own solver thanks to this article.

In the populate matrix for translation, r and d are swapped I think?

---

claude

5/9/2022, 9:49:19 AM

great post, could you do one on how to calculate the inverse kinematics once you have the DH parameters/transformation matrix

---

Related posts

Industrial Robots: 5 Most Popular Applications

Industrial robots have been in use for about 50 years. The first industrial robot was used for material handling in a General...

By Jean-Philippe Jobin - December 19, 2024

Read More

How to Combat The Great Resignation and Keep Talent with Robots

As the dust is settling on the 2020 global pandemic, many workers are finding themselves in a unique position. An abundance of...

By Alex Owen-Hill - January 6, 2023

Read More

How to Convince Employees to Accept

Robot Workers

When a company like yours first wants to adopt robots into the workplace, there can be resistance from employees.

Some people...

By Alex Owen-Hill - October 25, 2022

Read More