Basic Linear Algebra

Linear algebra deals with matrixes: two-dimensional arrays of values. Here's a matrix:

$$\begin{bmatrix} 1 & 5 & 7 \\ 9 & 3 & 11 \end{bmatrix}$$

Often matrices are used to describe in a simpler way a series of linear equations. In the example above the matrix might be thought of as the following two linear expressions:

$$\begin{gathered} 1x + 5y + 7z \\ 9x + 3y + 11z \end{gathered}$$

A $1 \times n$ matrix is called a *column vector*:

$$\begin{bmatrix} 4 \\ 9 \\ 7 \end{bmatrix}$$

Matrices can be added if they have exactly the same dimensions. In this case, you just add the respective values in each position:

$$\begin{bmatrix} 1 & 5 & 7 \\ 9 & 3 & 11 \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \\ 400 & 500 & 600 \end{bmatrix} = \begin{bmatrix} 101 & 205 & 307 \\ 409 & 503 & 611 \end{bmatrix}$$

Matrices can also be *multiplied*, but this means something different than you'd expect. When you multiply the matrices $A \times B = C$, an element at position $\langle x, y \rangle$ in C is equal to multiplying each element in *row x* of A times the corresponding element in *column y* of B, then adding up the sum. This means that the *number of columns* of A must be equal to the *number of rows* in B, and C's rows are the number of rows in A, and its columns are the number of columns in B.

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} m & p \\ n & q \\ o & r \end{bmatrix} = \begin{bmatrix} am + bn + co & ap + bq + cr \\ dm + en + fo & dp + eq + fr \end{bmatrix}$$

Matrix multiplication is associative. That is, $(AB)C = A(BC)$. It's also distributive over addition: $A(B+C) = AB + AC$. But unlike normal multiplication, matrix multiplication is *not commutative*. The following is *not true*: $AB = BA$. However, addition is commutative: $A + B = B + A$.

What can you use matrix multiplication for? Consider the following multiplication:

$$\begin{bmatrix} 1 & 5 & 7 \\ 9 & 3 & 11 \end{bmatrix} \begin{bmatrix} 4 \\ 9 \\ 7 \end{bmatrix} = \begin{bmatrix} 1\times4 + 5\times9 + 7\times7 \\ 9\times4 + 3\times9 + 11\times7 \end{bmatrix} = \begin{bmatrix} 98 \\ 140 \end{bmatrix}$$

If you look carefully, basically the two items in the final column vector are the results of $f_1(x,y,z) = 1x + 5y + 7z$ and $f_2(x,y,z) = 9x + 3y + 11z$ if you pass 4, 9, and 7 in as x, y, and z respectively. When you multiply a matrix against a column vector, think of the matrix's rows as *linear functions* and the column vector as the *values of the variables* you'll assign to the linear functions. The result is the output of the functions.

Since multiple matrices can be multiplied, consider two matrices M and N and a column vector Z. If you write NZ this produces the results of using Z's variables in the functions in the rows of N. You can then pass the results as variables for M's functions: $M(NZ)$. But since matrix multiplication is associative, this is the same thing as $(MN)Z$. Let MN equal the matrix O, so now we have OZ. What is O? We have *composed* the various functions in M *against* the various functions in N to produce the new functions in O. OZ is the equivalent of multiplying Z by N and then taking that and multiplying it against M.

Addition has similar notions: if you add two matrices M and N, this essentially "adds" their respective sets of functions.

Rotation Matrices

You can use matrices to rotate a vector by a certain amount. Let your initial vector be the column vector C. If your rotation matrix is called $R(\theta)$ (it rotates vectors by the angle θ, then the resulting vector is computed as $R(\theta)C$.

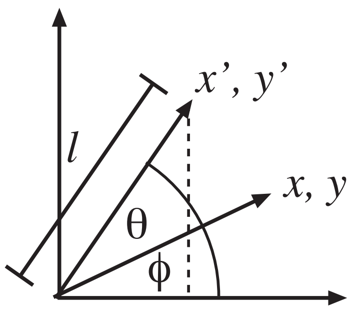
A two-dimensional $R(\theta)$ is defined as:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

To rotate the vector $\langle x, y\rangle$ by θ, resulting in the new vector $\langle x', y'\rangle$ of the same length l, you have:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Here's a picture:



The rotation matrix's equations are thus:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Proof Let's prove these equations to make sure the matrix is right. We have the following trigonometric identities:

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$
$$\sin(\alpha+\beta) = \cos\alpha\sin\beta + \sin\alpha\cos\beta$$

And also we know $l = \sqrt{x^2+y^2} = \sqrt{x'^2+y'^2}$ (they're the same length). From the picture above it should be clear that

$$x' = l\cos(\theta+\phi)$$

From our identity, we get

$$x' = l\cos\theta\cos\phi - l\sin\theta\sin\phi$$

Since $\cos\phi = \frac{x}{l}$ (see the picture), then $\phi = \cos^{-1}\frac{x}{l}$. Likewise $\phi = \sin^{-1}\frac{y}{l}$ by the same argument. So we have:

$$\begin{aligned} x' &= l\cos\theta\cos(\cos^{-1}\frac{x}{l}) - l\sin\theta\sin(\sin^{-1}\frac{y}{l}) \\ &= l(\cos\theta)\frac{x}{l} - l(\sin\theta)\frac{y}{l} \\ &= x\cos\theta - y\sin\theta \end{aligned}$$

Similarly,

$$y' = x\sin\theta + y\cos\theta$$

Reverse Rotation To rotate a vector by $-\theta$ (to "rotate it back"), this is of course

$$\begin{bmatrix} \cos-\theta & -\sin-\theta \\ \sin-\theta & \cos-\theta \end{bmatrix}$$

...but since $\cos-\alpha = \cos\alpha$ and $\sin-\alpha = -\sin\alpha$, we can simplify this to

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Notice that this is just the original matrix where we've swapped the rows for the columns. This is called *transposing* a matrix. The transpose of a matrix M is M^T. Thus $R(-\theta) = R(\theta)^T$.

Translating a Vector

Translating a vector is much easier than rotating it: just add it to another vector representing the amount you want to translate each element by. For example, if you want to add $\langle\triangle x, \triangle y\rangle$ to $\langle x, y\rangle$ resulting in $\langle x', y'\rangle = \langle x+\triangle x, y+\triangle y\rangle$, you simply do:

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} = \begin{bmatrix} x + \triangle x \\ y + \triangle y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Rotating and Translating in 3D

3D vectors look like this:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

There are three ways you can rotate a vector in 3D: along the X axis, the Y axis, and the Z axis. The matrices are:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

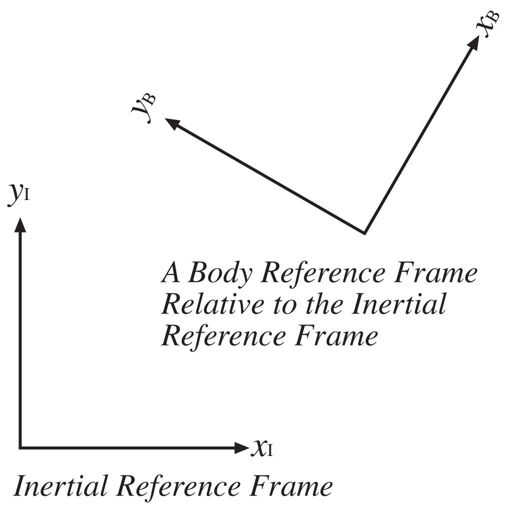
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translating a vector is still just addition:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle z \end{bmatrix} = \begin{bmatrix} x + \triangle x \\ y + \triangle y \\ z + \triangle z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Reference Frames


x_B
y_B
y_I
A Body Reference Frame Relative to the Inertial Reference Frame
x_I
Inertial Reference Frame

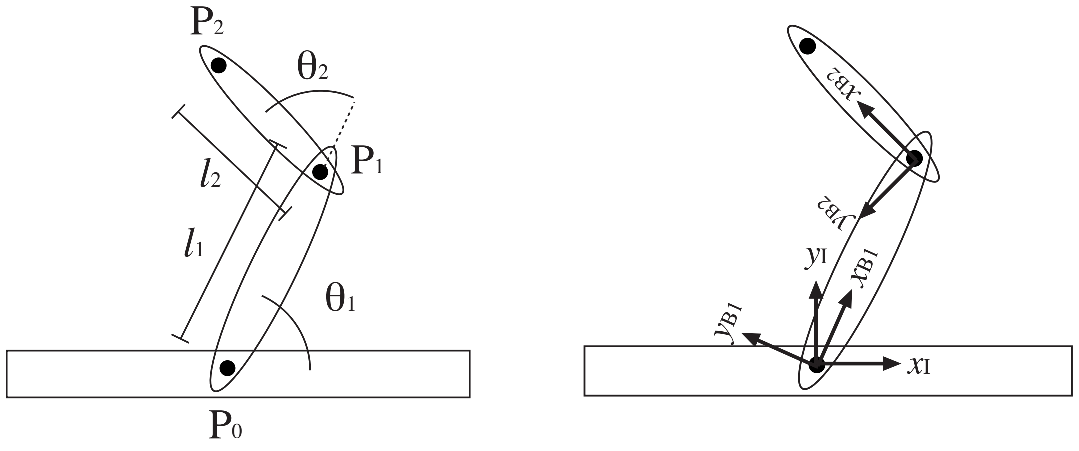
A reference frame is a set of axes which define where the $\langle 0,0\rangle$, $\langle 1,0\rangle$, and $\langle 0,1\rangle$ points are in some cartesian space. An *inertial reference frame* is a global reference frame which never changes. A *body reference frame* is a reference frame which may be rotated and translated etc. relative to the inertial reference frame, and which can change its relative rotation and translation over time. Body reference frames are a convenient way to think of parts of a robot body: as a robot arm (say) rotates and translates, the arm's body reference frame moves with it.

If a vector is described in the coordinate system of one reference frame, and you want to know how it's described in the coordinate system of another reference frame, rotate and translate the vector just the second reference frame is rotated and translated relative to the first. Such rotation and translation can be done, as you might imagine, via rotation and translation.

Forward Kinematics for a 2D Arm with Two Degrees of Freedom

Forward kinematics for a robot arm involves figuring out a function that takes as it's inputs the angles of each joint and computes the position of the end point P_i:

$$f(\theta_1,\theta_2) = \begin{bmatrix} x \\ y \end{bmatrix}$$



Here's a diagram os a two-link robot arm. On the left we define the arm lengths l_1 and l_2, and the joint angles θ_1 and θ_2. On the right we have defined three reference frames. The inertial ("global") reference frame is marked with the axes x_I and y_I. The body reference frame for arm 1 has axes X_{B1} and Y_{B1}, and is rotated from the inertial reference frame by θ_1. The body reference frame for arm 2, marked with X_{B2} and Y_{B2}, is translated l_1 units down arm1 to point P_1 in body reference frame 1 and is then rotated θ_2.

To figure out where the point P_2 is in the intertial refence frame, first we're going to figure out where it is in frame 2, which is easy:

$$P_{2,B2} = \begin{bmatrix} l_2 \\ 0 \end{bmatrix}$$

Next we need to figure out where it is in frame 1. This entails both a rotation, because frame 2 is rotated with respect to frame one by an angle of θ_2, and also a translation, since the origin of frame 2 is translated with respect to the origin of frame 1 by a distance of l_1. Therefore,

$$P_{2,B1} = \begin{bmatrix} l_1 \\ 0 \end{bmatrix} + R(\theta_2)P_{2,B2} = \begin{bmatrix} l_1 \\ 0 \end{bmatrix} + R(\theta_2)\begin{bmatrix} l_2 \\ 0 \end{bmatrix}$$

Finally, we need to express $P_{2,B1}$ in the inertial frame. This is just a rotation by an angle of θ_1; there is no translation because the origin of frame 1 and the intertial frame coincide. Therefore,

$$P_{2,I} = R(\theta_1)P_{2,B1} = R(\theta_1)\left(\begin{bmatrix} l_1 \\ 0 \end{bmatrix} + R(\theta_2)\begin{bmatrix} l_2 \\ 0 \end{bmatrix}\right)$$

If you work out all of the matrix-vector multiplications, which is a little tedious but not hard, you end up with

$$P_{2,I} = \begin{bmatrix} l_1\cos(\theta_1) + l_2\left(\cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)\right) \\ l_1\sin(\theta_1) + l_2\left(\sin(\theta_1)\cos(\theta_2) + \cos(\theta_1)\sin(\theta_2)\right) \end{bmatrix} = \begin{bmatrix} l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) \\ l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) \end{bmatrix}$$

and we're done. This is a general algorithm, in the sense that if you define enough reference frames and put them in the right spots, you can correctly derive the forward kinematics for any robot arm. You don't need any other tricks; simply placing reference frames at each of the joints and working your way from the top of the arm to the bottom will do the job.

1 Inverse Kinematics for a 2D Arm with Two Degrees of Freedom

Inverse kinematics for a robot arm entails finding a function that takes as its input $P_{2,I}$ and computes θ_1 and θ_2. In other words, here we are specifying where we want the end of the arm to be and figuring out which joint angles we need to get it there.

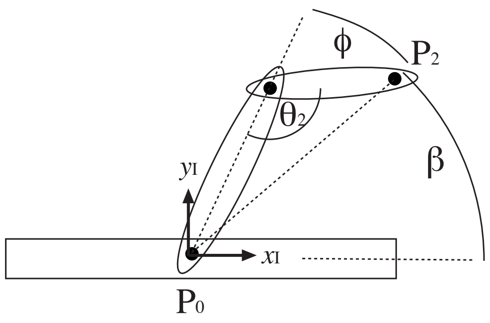
The inverse kinematics problem is much more difficult than the forward kinematics problem. There is no general way to come up with the inverse kinematics function for a robot arm in symbolic form. Instead, numerical algorithms are usually used. Our example arm is simple enough, though, that we can solve the inverse kinematics problem for it. To do that, we're going to use a trigonometric equation called the Law of Cosine. The Law of Cosine is a generalization of the Pythagorean Theorem, which states that for any right triangle,

$$c^2 = a^2 + b^2$$

where c is the hypotenuse. The Law of Cosines works for any triangle, not just right ones, and it states:

$$c^2 = a^2 + b^2 - 2ab\cos(\theta)$$

where now, c doesn't have to by the hypotenuse; it's just the side opposite from the corner who's angle is θ.



Now look at the diagram above. We have defined two new angles: β, which is the angle between the x axis of the inertial frame and the line from the origin to point P_2, and ϕ, which is the angle between the first arm and point P_2. We need to find expressions for θ_1 and θ_2 given point P_2. We can immediately use the Law of Cosine to figure out θ_2 because arm 1, arm 2, and the line from the origin to P_2 form a triangle. In this case, our c is the length of the line from the origin to P_2, which we know is $\sqrt{x^2 + b^2}$; our a and b are l_1 and l_2, respectively; and our θ is $180 - \theta_2$. Therefore, after a little simple algebra, we come up with the equation

$$\theta_2 = 180 - \cos^{-1}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right)$$

This equation looks simple, but it has a slight complication — what does it mean to take an inverse cosine? If you plot a cosine you will notice that it's symmetric across the y-axis. There are therefore two angles that lead to the same cosine value: θ and $-\theta$. Usually $\cos^{-1}$ picks the positive one, but it doesn't have to — we could just as easily pick the negative one. What does this mean? Well, picture the reflection of the arm across the line from the origin to P_2 — the arm's elbow could be either above or below the line. I other words, there are *two* values of θ_1 and θ_2 that could lead to the same P_2. In the diagram we've shown $theta_2$ as less than 180, but we could choose it to be greater than 180. Therefore, really, the equation should read:

$$\theta_2 = 180 \pm \cos^{-1}\left(\frac{x^2+y^2-l_1^2-l_2^2}{2l_1 l_2}\right)$$

The problem is now half solved. We still need to come up with an equation for θ_1. We can use the Law of Cosines again, this time to find the angle ϕ. This time, our c is l_2, our a and b are l_1 and $\sqrt{x^2+y^2}$, and our θ is just ϕ. The Law of Cosines yields:

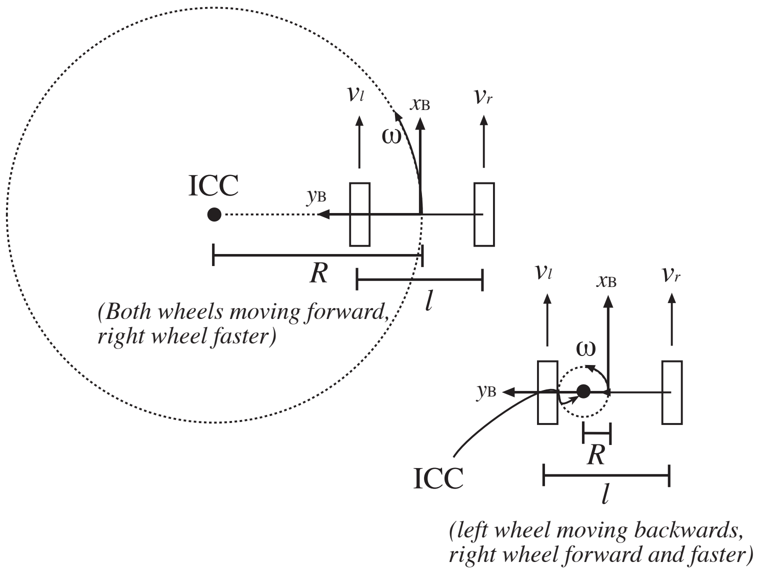
$$\phi = \cos^{-1}\left(\frac{x^2+y^2+l_1^2-l_2^2}{2l_1\sqrt{x^2+y^2}}\right)$$

We now notice by looking at the diagram that $\theta_1 = \beta + \phi$. Actually, if you look hard enough you can convince yourself that θ_1 could also be $\beta - \phi$; if we choose to pick a value of θ_2 that's greater than 180, then we also have to pick $\theta_1 = \beta + \phi$. If we choose θ_2 less than 180 then $\theta_1 = \beta - \phi$. Therefore, we have three equations allow us to start with P_2 and calculate θ_1 and θ_2, which is what we wanted, so the problem is solved. Actually, we get a choice of two values for θ_1 and θ_2, which is even better.

2 Forward kinematics for a differential–drive mobile robot

The forward kinematics equations for a robot arm allow us to specify the position of each of the actuators (in this case, the actuators are the motors that drive each of the arm joints, and their positions are the joint angles θ_1 and θ_2) and calculate the position of the end of the arm in the inertial reference frame. Notice that for a robot arm, the past positions of the joints don't make any difference — we can always calculate the current position of the end of the arm from the current joint angles. In contrast, the actuators for a differential drive mobile robot are the wheels. Ideally, we would like to come up with forward kinematics equations for the mobile robot that are analogous to those for a robot arm. In other words, we would like to calculate the position of the robot in the inertial frame just by using the position of the wheels — but that isn't possible. For a mobile robot, the past position of the actuators (called the time history) are important because there are many trajectories that could end up with the same wheel positions, and unlike a robot arm each of those trajectories moves the mobile robot to a different place.

If we can't come up with forward kinematics equations that are similar to those for a robot arm, what can we do? One possibility is to derive some equations that tell us what the current velocities of the mobile robot are in the inertial reference frame given the velocities of the wheels. If we could do that, and if we knew where the robot started, then in theory we could always calculate the robot's current position by integrating it's velocity. That's what old ship navigators did — they knew their velocity and they knew the time, and they could then estimate their position by using dead reckoning.



v_l
x_B
v_r
ω
ICC
y_B
R
l
(Both wheels moving forward, right wheel faster)
v_l
x_B
v_r
ω
y_B
ICC
R
l
(left wheel moving backwards, right wheel forward and faster)

To start we have to derive an equation that relates the robot's wheel velocities to the chassis velocity in a body — fixed frame. Since our mobile robot operates in a plane (the floor), is has three potential degrees of freedom — x (forward), y (sideways), and θ (rotation). We can easily determine an equation for the y velocity — it must be zero:

$$v_{y,B} = 0$$

In order to calculate the forward and rotational velocities, first imagine what the robot will do if you cause the wheels to spin at constant, random velocities. There are two possibilities: if the velocities happen to be exactly the same, the robot will travel in a straight line. Otherwise, it will travel in a circle. If the wheel velocities are close to each other, the circle will be large; if they are very different from each other, the circle will be small. We'll call the center of the circle the ICC (Instantaneous Center of Curvature). Also, we will call the rotational velocity of the robot (the rate of speed, in radians per second, that it takes for the robot to complete one full rotation around the circle) ω. Recall that the circumference of a circle of radius r is $2\pi r$; the length of the partial circumference of the pie–shaped part of the circle defined by an angle of ϕ radians is ϕr. Therefore, the rate of speed of a point that is traveling along the circumference of a circle with radius r and sweeping out ω radians per second is ωr units per second. If the robot is traveling along a circle whose center is the ICC, then, the inner wheel v_l has a velocity of $\omega(R - l/2)$ and the outer wheel v_r has a velocity of $\omega(R + l/2)$.

By subtracting these two expressions we can come up with an expression for ω in terms of v_r

and v_l:

$$\begin{aligned} v_r - v_l &= \omega(R + l/2) - \omega(R - l/2) \\ &= \omega l \end{aligned}$$

or

$$\omega = (v_r - v_l)/l$$

By adding them we can come up for an expression for R in terms of v_r and v_l:

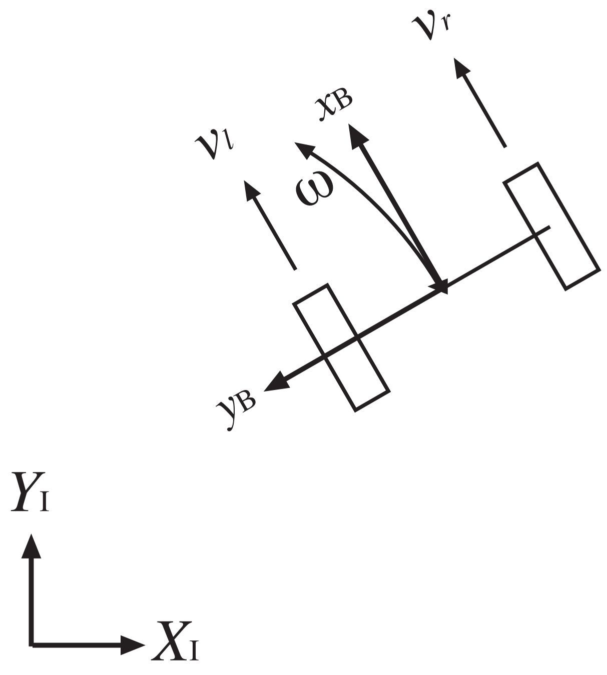
$$\begin{aligned} v_r + v_l &= \omega(R + l/2) + \omega(R - l/2) \\ &= 2\omega R \end{aligned} \tag{1}$$

and by substituting in our expression for ω and solving for R we get

$$R = \left(\frac{l}{2}\right)\left(\frac{v_r + v_l}{v_r - v_l}\right)$$

Now we know what the forward velocity of the chassis is: it's just ωR, since the chassis is traveling along a circle of radius R with a rotational velocity of ω. And since we can now calculate both R and ω from the wheel velocities, we can write an expression for the velocity of the chassis in the body fixed frame:

$$\begin{bmatrix} v_{x,B} \\ v_{y,B} \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(v_l + v_r) \\ 0 \\ \frac{1}{l}(v_r - v_l) \end{bmatrix}$$



The only thing left to do is to write the translational velocities v_x and v_y in terms of the inertial frame, which means simply multiplying them by a rotation matrix:

$$\begin{bmatrix} v_{x,I} \\ v_{y,I} \end{bmatrix} = R(\theta) \begin{bmatrix} \frac{1}{2}(v_l + v_r) \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\theta(v_r + v_l) \\ \frac{1}{2}\sin\theta(v_r + v_l) \end{bmatrix}$$

where θ is the angle between the inertial frame and the body–fixed frame.

Are we done? Think really hard for a minute about how you would code up these equations. Is there anything you don't know?

Answer: yes. You don't know θ — in fact, θ is one of the quantities we started out with the goal of computing. We have an expression for ω, and ω and θ are related:

$$\dot{\theta} = \omega$$

but we don't yet have a way to calculate θ itself. What now?

The only way to calculate θ is to integrate ω over the entire time the robot is moving. There are many algorithms for doing this, but the simplest one (called a backward Euler integration scheme) goes back to what you learned in basic calculus. Remember than geometrically, the integral of $f(t)$ is just a way of calculating the area under the curve defined by $f(t)$. Here, our curve is defined by ω, and we need to calculate the area under it. When you learned about the theory behind integration, you learned that you can divide the curve up into small slices, assume the curve was constant over the slices, and use the formula for the area of a rectangle to calculate the area of the slice. Then to approximate the area under the entire curve you just added the areas of each of the slices. That's what we are going to do here. Suppose you divide the curve formed by ω into slices of length δ. Then we can approximately calculate θ by the following equation:

$$\theta_n \approx \theta_{n-1} + \delta_{n-1}\omega_{n-1}$$

where θ_0 is the angle that the robot started out at with respect to the inertial reference frame, and θ_n is the angle it's at after a time slice of length δ_n. Note also that as long as you can measure the elapsed time between iterations, the time slices for each iteration doesn't have to be exactly the same length.

Now that you can calculate θ, then you can calculate $v_{x,I}$ and $v_{y,I}$. And you can do one more thing: you can use our approximate integration algorithm to calculate x_I and y_I. Here's the complete algorithm:

$$\begin{aligned} \theta_n &\approx \theta_{n-1} + \delta_{n-1}(v_{r,n-1} - v_{l,n-1})/l \\ x_{I,n} &\approx x_{I,n-1} + \frac{\delta_{n-1}}{2}\cos\theta_{n-1}(v_{r,n-1} + v_{l,n-1}) \\ y_{I,n} &\approx y_{I,n-1} + \frac{\delta_{n-1}}{2}\sin\theta_{n-1}(v_{r,n-1} + v_{l,n-1}) \end{aligned}$$

where $x_{I,0}$ and $y_{I,0}$ are the x and y starting positions of the robot in the inertial frame and $v_{r,n}$ and $v_{l,n}$ are the wheel velocities at the beginning of time slice δ_n. Now as long as you have a reasonably fast processor, so that δ is small, and a reasonably good clock, so that you can measure δ_n accurately, you can calculate your robot's position.