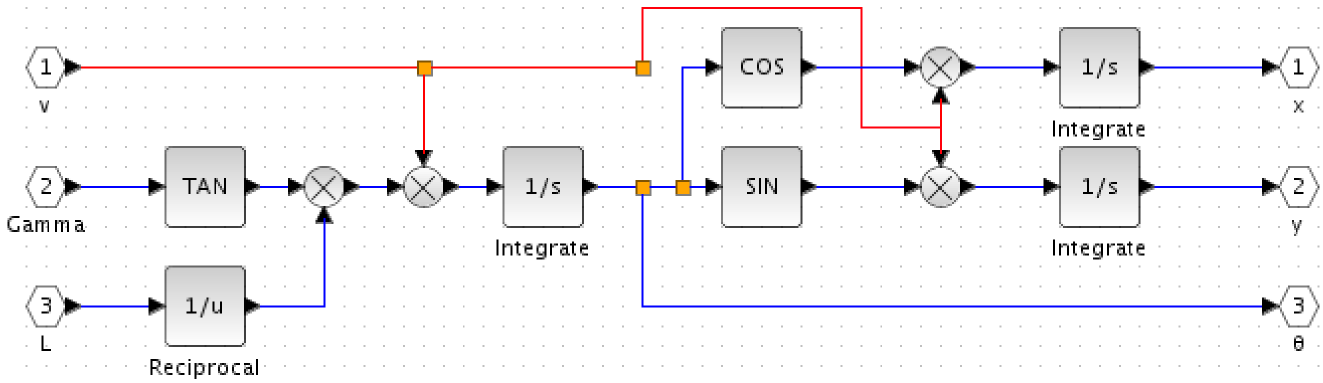
Nicolas Fredrickson
Assignment 1

Bicycle Model



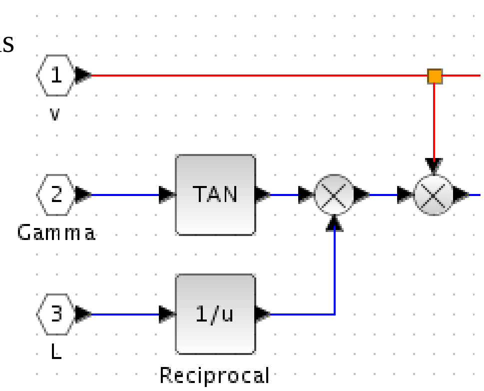
All of my models accept **v, γ** (rendered above as Gamma due to the poor font), and **L**. **v** (velocity) and **γ** (orientation) are the standard control variables for the bicycle model.

L is the length between wheels, and was included as a "parameter" here because I was under the impression that **L** was needed for calculations both inside the model and outside. While this turned out not to be the case, it did make editing the value for **L** much easier.

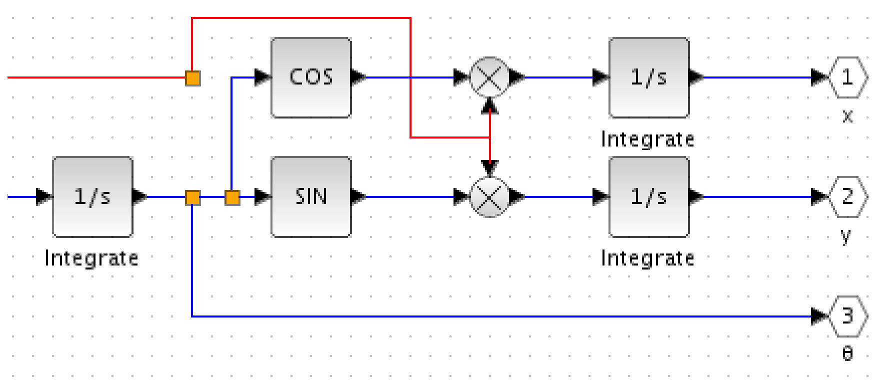
The equations for the model are:

$$\dot{\theta} = \frac{v}{L}\tan(\gamma) \qquad \dot{x} = v\cos(\theta) \qquad \dot{y} = v\sin(\theta)$$

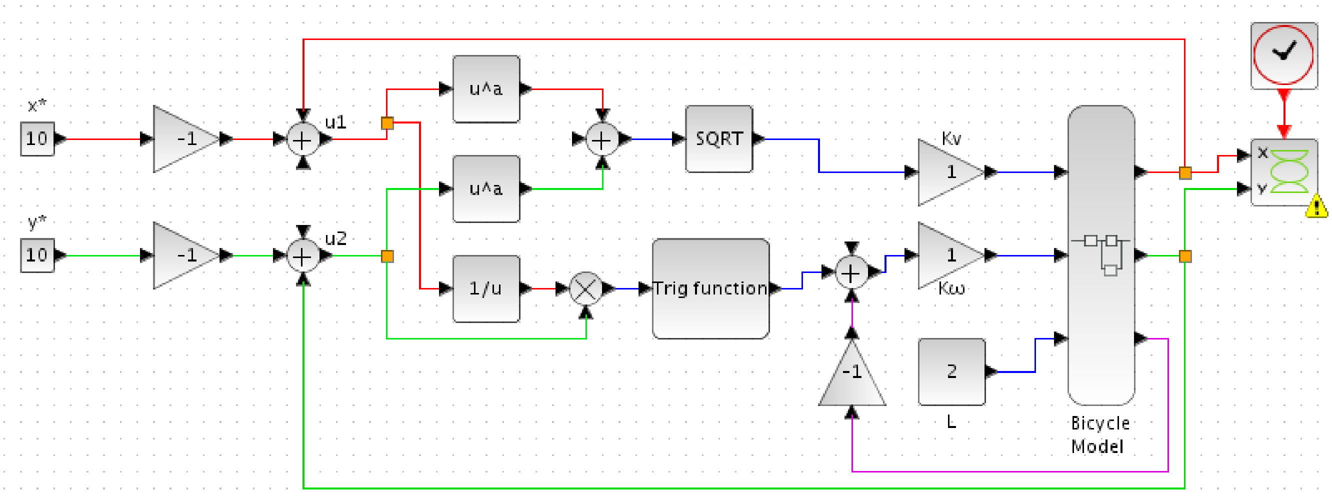
We can see the first equation on the left-hand side of the equation, placed here for ease of access. First, **L** is reciprocated and **γ** is put through a tangent function. The two are multiplied together, along with **v**, to produce **θ – dot**.



Below, you'll see our other two equations. **θ – dot** is integrated to produce **θ** (blue line). This value, run through cos or sin and multiplied by **v** (red line) produces **x – dot** and **y – dot**, respectively. These values, integrated, produce our **x** and **y** values.



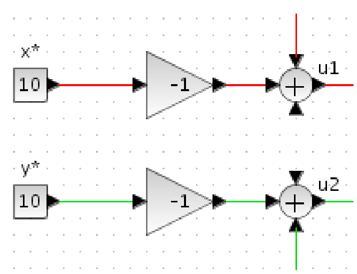
Get-to-the-Point (G2P), Bicycle Model



This system relies on two equations:

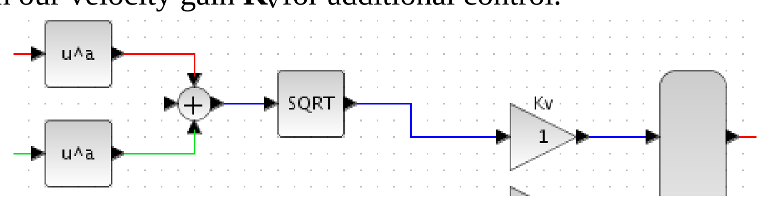
$$v = K_v\sqrt{(x-x^*)^2+(y-y^*)^2} \qquad \gamma = K_\omega\left(\arctan\left(\frac{y-y^*}{x-x^*}\right)-\theta\right)$$

First, notice how the green and red lines coming from the model loop around and have our target x and y, $\mathbf{x}^*$ and $\mathbf{y}^*$, subtracted from them. Since we reuse this calculation, we group these together into $\mathbf{u_1}$ and $\mathbf{u_2}$. So let's look at our first equation in those terms.



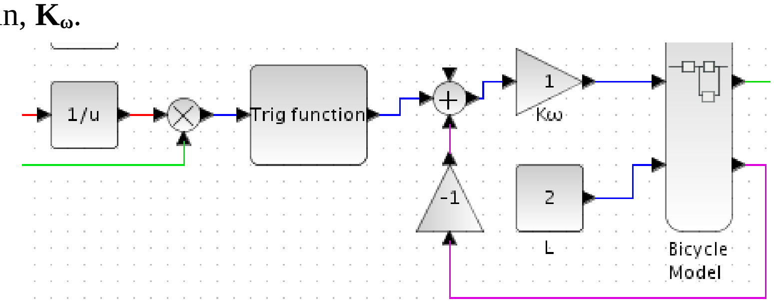
$$v = K_v\sqrt{(u_1)^2+(u_2)^2}$$

This is implemented in the excerpt below – both $\mathbf{u_1}$ (red line) and $\mathbf{u_2}$ (green line) are squared, added together, and then put through a square root function. This produces a usable value for **v –** in this case, our distance from the our target point. However, we then put this value through our velocity gain $\mathbf{K_v}$ for additional control.

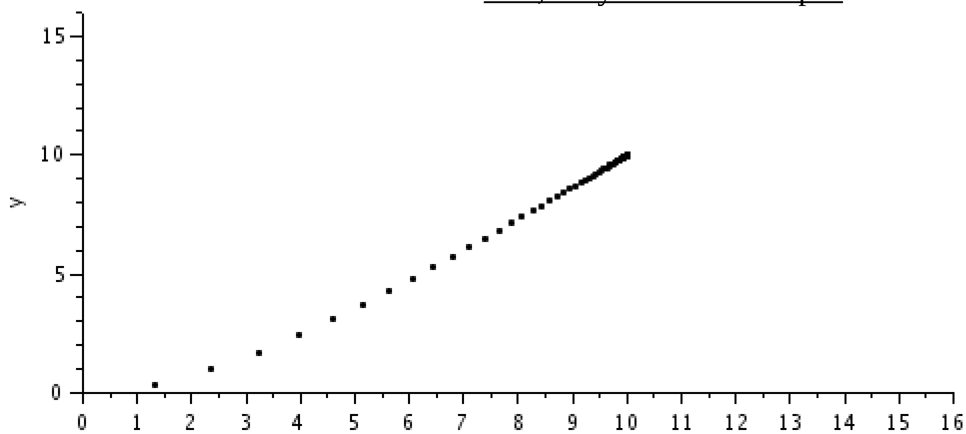


The second equation (lower left) is implemented like so (lower right). $\mathbf{u_1}$ (red line) is reciprocated and the multiplied with $\mathbf{u_2}$ (green line). This is then put through an arctangent function, and has our current $\boldsymbol{\theta}$ (purple-magenta line) subtracted. This produces a working $\boldsymbol{\gamma}$ **–** our angular difference from the point – but like before, we subject it to a gain, $\mathbf{K_\omega}$.

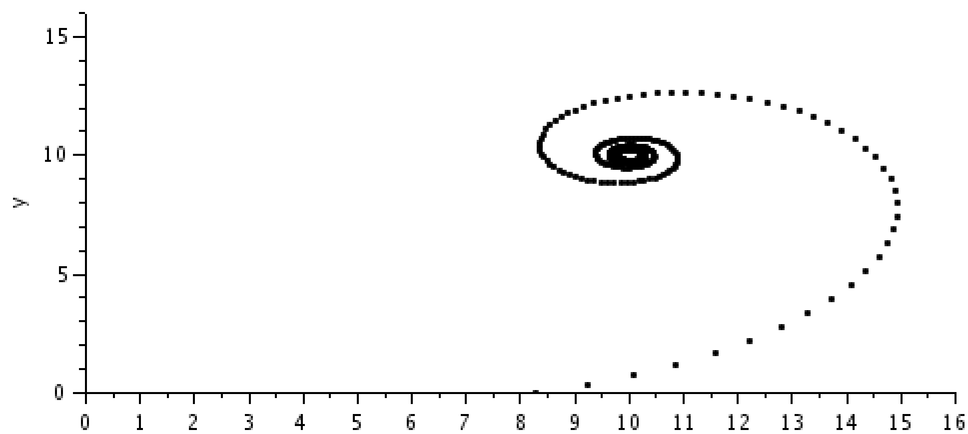
$$\gamma = K_\omega\left(\arctan\left(\frac{u_2}{u_1}\right)-\theta\right)$$



G2P, Bicycle Model Graphs

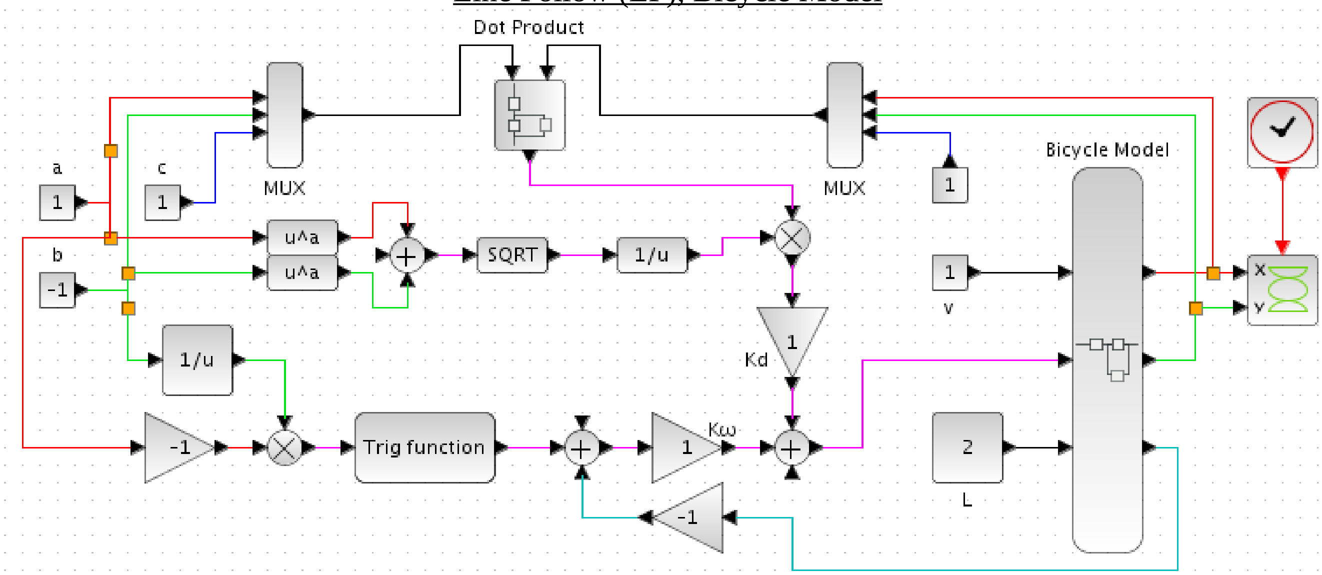


The G2P model shown on the other page shows what were essentially the "standard" values. Our target was (10, 10) and our **L** was 2. Running this "standard" configuration produced this output, which shows the characteristic slow-down as the target is approached.



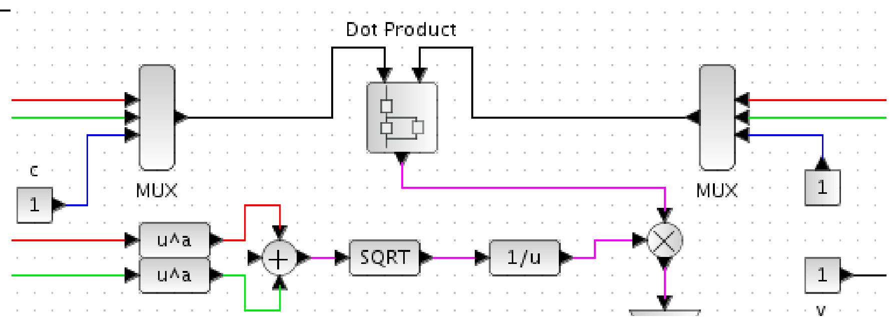
In this variant, our $\mathbf{K}_{\omega}$ was set to 0.25. This reduced the model's ability to correct it's angle, leading it to loop around its target. It still exhibits the slow-down, but can't reach its target exactly. This creates a sort of black blotch around our target point.

Line Follow (LF), Bicycle Model

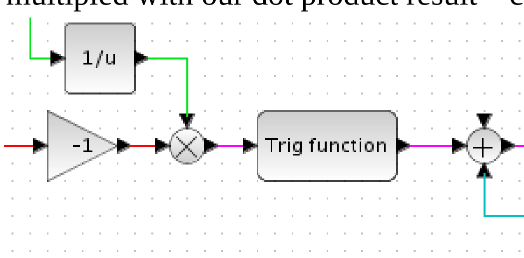


This system actually only relies on one giant equation, but for the sake of sanity I will break it into three components:

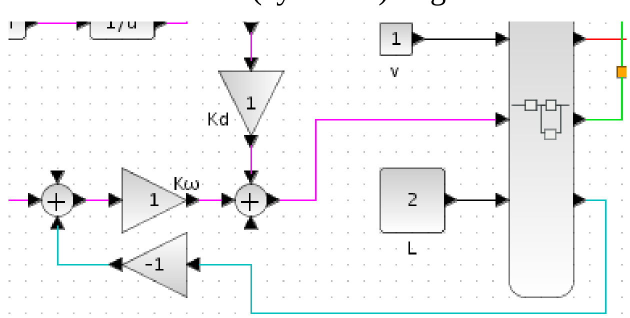
$$\gamma = K_\omega \omega + K_d d \qquad \omega = \arctan\left(\frac{-a}{b}\right) - \theta \qquad d = \frac{(a,b,c)\cdot(x,y,1)}{\sqrt{a^2+b^2}}$$



Let's start with the third equation – our distance from the line, **d**. This is defined in the section shown here. The left multiplexer (**mux**) combines **a** (red), **b** (green), and **c** (blue). The right mux combines **x** (red), **y** (green), and **1** (blue). These two vectors are then processed through a custom dot product block (the instructions for the other dot product block were vague and confusing). Meanwhile **a** and **b** are squared, added, and then square-rooted. This semi-distance formula is then reciprocated and multipled with our dot product result – creating **d**.

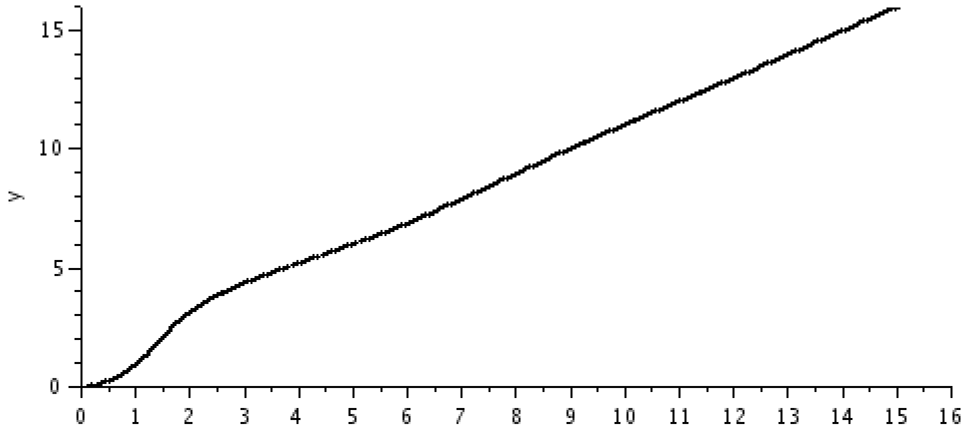


Our **ω** is defined by our current angle, subtracted from the angle of the line. The angle of the line is found by multiplying a reciprocated **b** (green line, also called **rise**) with a negated **a** (red line, also called **run**). This effectively gives us our slope; this is then put through arctangent to get the angle of the line. We then subtract our current **θ** (cyan line) to get our **ω**.

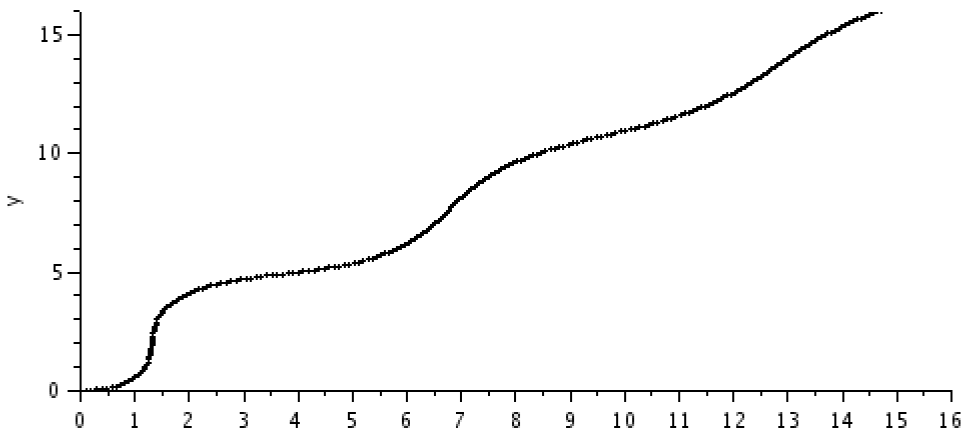


We can now combine these two values, and put them through the model – as seen on the right. $\mathbf{K_d}$ gains **d** and $\mathbf{K_\omega}$ gains **ω**. Also seen here is our constant velocity **v**, our length **L,** and the negation of our current angle **θ**.

LF, Bicycle Model Graphs

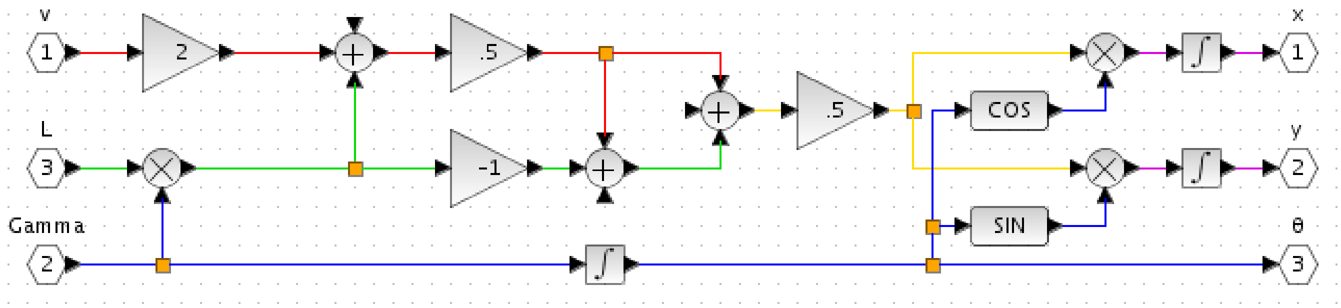


This is the result of using the “standard” values depicted above. It’s important to note that changing the velocity does little to change the graph – higher numbers only decrease the number of individual points that are sampled.



This was the result of decreasing our value for $\mathbf{K}_{\omega}$. The bicycle oscillates back and forth, attempting to match the angle of the line. Notice how it becomes more accurate over time – given enough time, it will match the slope and stop oscillating.

Differential Model



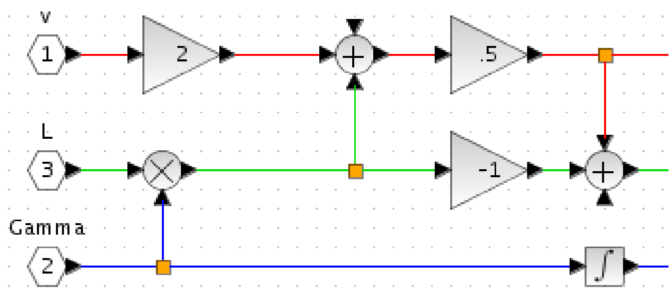
We've already gone over our different inputs. What makes this model unique is the calculations we must do on **v, γ,** and **L**. Specifically, we are converting these inputs into $\mathbf{v_r}$ and $\mathbf{v_L}$. There are two equations we must start with:

$$\frac{1}{2}(v_r+v_L)\cos\theta=v\cos\theta \qquad \gamma=\frac{v_L-v_r}{L}$$

Together, these two form a system of equations. It can eventually be shown that:

$$v_L=\frac{\gamma L+2v}{2} \qquad v_r=v_L-\gamma L$$

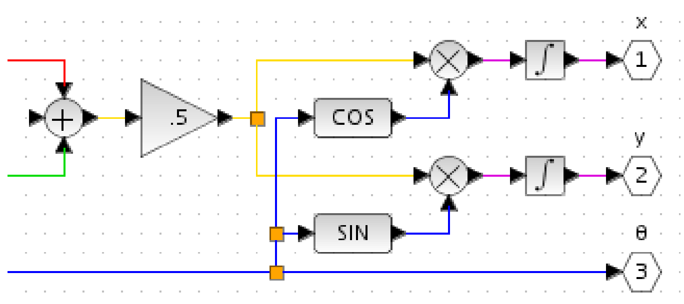
These two equations are defined here:



Our **L** is immediately multiplied by **γ** to form **γL**. This is then added to **2v**, and halved – producing $\mathbf{v_L}$. **γL** is then subtracted from $\mathbf{v_L}$ to create $\mathbf{v_r}$. Also, we integrate **γ**, to get **θ**. With all these values converted, we can now use $\mathbf{v_r}$ and $\mathbf{v_L}$ in our **x** and **y** formulas:

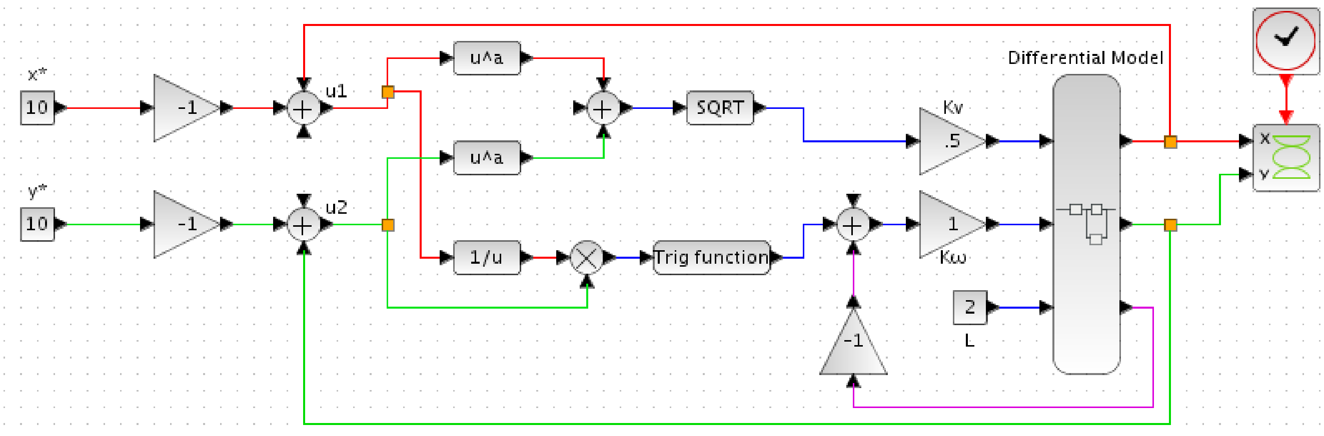
$$\dot{x}=\frac{1}{2}(v_r+v_L)\cos\theta \qquad \dot{y}=\frac{1}{2}(v_r+v_L)\sin\theta$$

Fortunately, the addition and halving is a common factor to both, which makes implementing these fairly simple:



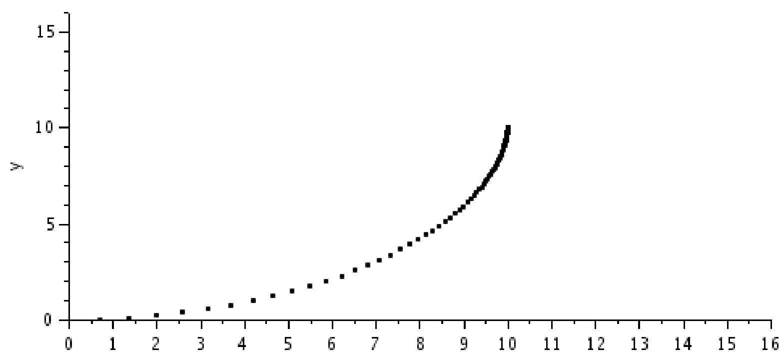
Add the two components, half them, and multiply by either sine or cosine. Easy and simple.

Get-to-the-Point (G2P), Differential Model

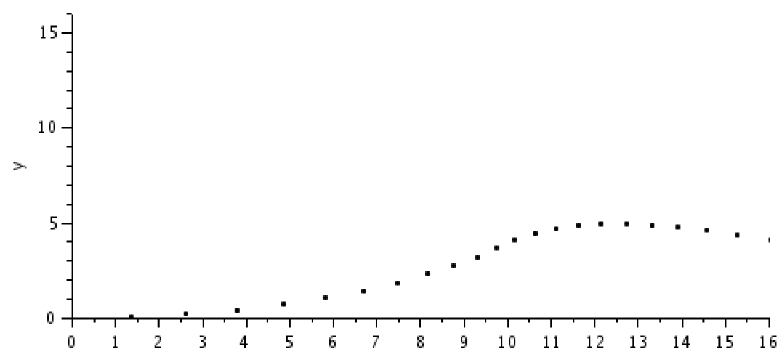


The beauty of this type of model is that its encapsulated – same inputs, same outputs, but different things are happening under the hood. Point is, I don't have to bother re-explaining any of it because we've already gone over this. The only thing that is different here is the formating and our "standard" $\mathbf{K_v}$ (we'll get to that).

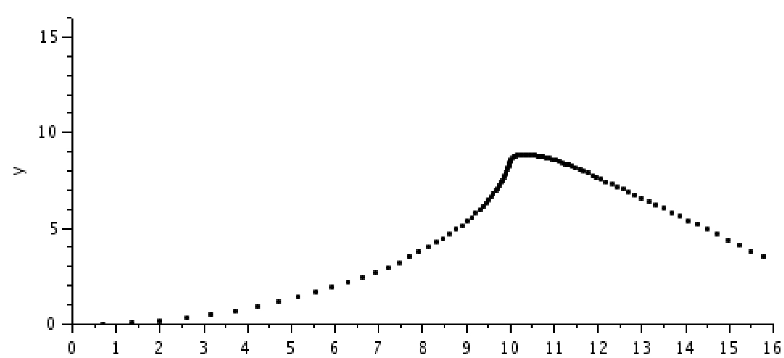
G2P, Differential Model Graphs



This is the our "standard" graph. Notice the bowed shape – this is because of the differential drive. Since the wheels can move at different speeds, the model can make angular adjustments much faster. It still exhibits the slow-down effect seen in the G2P graphs for the bicycle model.

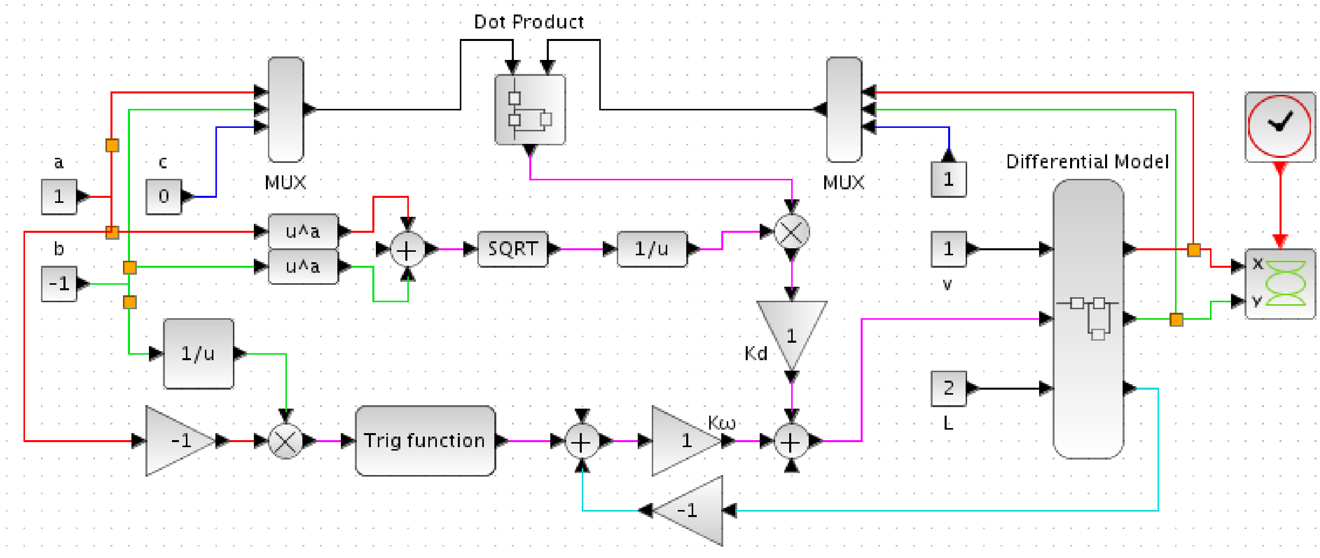


Setting $\mathbf{K_v}$ to a value roughly greater than 0.5 yields a graph like this. I'm honestly at a loss as to why this happens – maybe it's because we are working with two wheels, and the velocity must be split between them? Curious, to say the least. In the graph to the left, $\mathbf{K_v} = 1$.



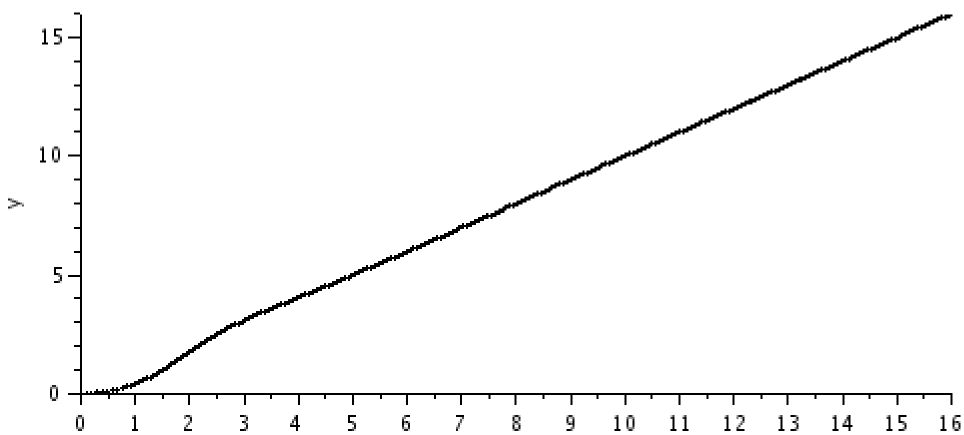
A similar phenomenon happens for values of $\mathbf{K_\omega}$ that are roughly less than one. The value used in this graph is $\mathbf{K_\omega} = 0.9$.

Line Follow (LF), Differential Model

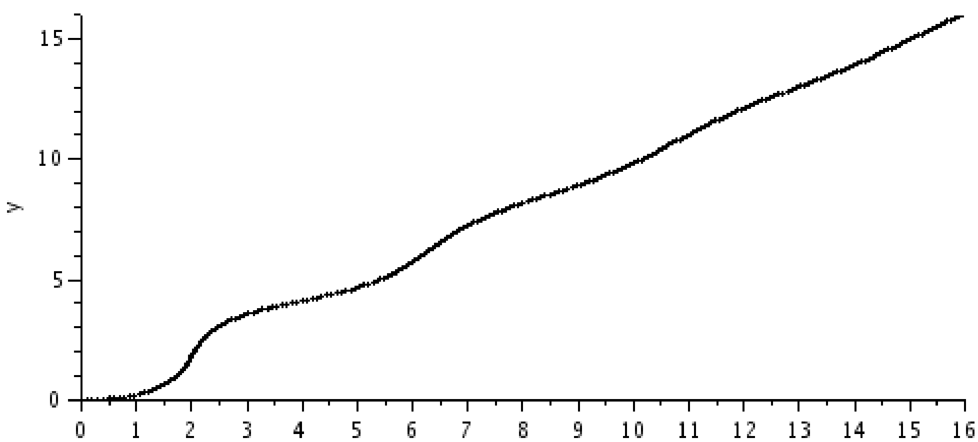


Yeah, yeah, same-old same-old. Let's look at the graphs.

LF, Differential Model Graph



Standard and plain line graph. Real simple stuff here. Of course, do notice that the differential model adjusts to the line much more quickly than the Bicycle model.



This graph was the result of setting $\mathbf{K}_\omega = 0.25$. The oscillation is still there, but much less pronounced. In fact, by the time the line reaches the edge of the graph, the line has stopped oscillating entirely. Another advantage of the differential model.

Derivation of d

So, we have a line equation, and from that we can get a slope:

$$ax+by+c=0 \qquad slope=\frac{-a}{b}$$

Now, what we're looking for is the length of the perpendicular line from our point to the line. This perpendicular line will have a slope:

$$slope_P=\frac{b}{a}$$

Let's assume (m, n) to be the point of intersection between our line and the perpendicular line that passes through our point, (x_0, y_0). The line through those two points is the perpendicular, so

$$\frac{y_0-n}{x_0-m}=\frac{b}{a}$$

With some algebraic rearrangement this becomes

$$a(y_0-n)-b(x_0-m)=0$$

Now, we can square the thing:

$$(a(y_0-n)-b(x_0-m))^2=0$$

And now begins the math magic! Let's expand this:

$$(a(y_0-n)-b(x_0-m))^2=a^2(x_0-m)^2+2ab(y_0-n)(x_0-m)+b^2(y_0-n)^2$$

$$a^2(x_0-m)^2+2ab(y_0-n)(x_0-m)+b^2(y_0-n)^2=(a^2+b^2)((x_0-m^2)+(y_0-n^2))$$

$$(a^2+b^2)((x_0-m^2)+(y_0-n^2))=0$$

Okay, now what if we expanded in a different way?

$$(a(y_0-n)-b(x_0-m))^2=(ax_0+by_0-am-bn)^2$$

Now, since (m, n) are on our line, it is true that

$$am+bn+c=0$$

$$c=-am-bn$$

Therefore,

$$(ax_0+by_0-am-bn)^2=(ax_0+by_0-am-bn)^2$$

$$(ax_0+by_0+c)^2=0$$

Now, we can combine our two equations:

$$(a^2+b^2)((x_0-m^2)+(y_0-n^2))=(ax_0+by_0+c)^2$$

Now, if we square root, and shift some things around, we get:

$$\sqrt{(x_0-m^2)+(y_0-n^2)}=\frac{ax_0+by_0+c}{\sqrt{a^2+b^2}}$$

And notice that the left hand side is the perpendicular distance from our point to the line. Thus,

$$d=\frac{ax_0+by_0+c}{\sqrt{a^2+b^2}}$$