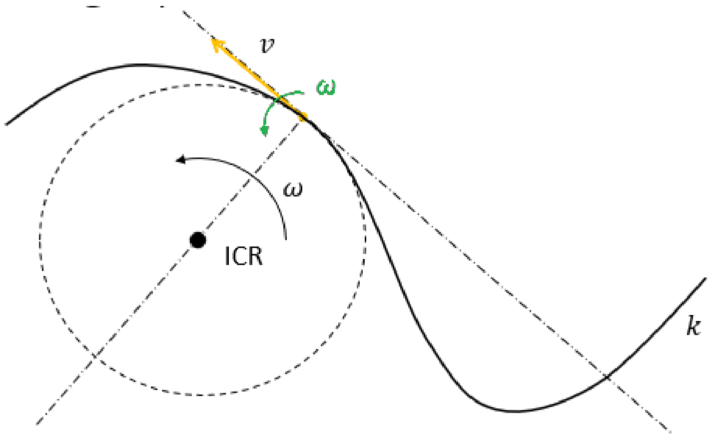
<u>(Diskrete Inverse) Kinematik</u>:

In der Kinematik geht es um die Bewegung von Objekten. Zur Beschreibung werden die Größen Ort, Geschwindigkeit und Beschleunigung verwendet.

Die Ursache der Bewegung (beschrieben durch Kräfte, Masse, Impuls oder Energie) ist nicht Teil der Kinematik → Kinetik

$\text{Geschwindigkeit} = \frac{Weg}{Zeit}$ d.h. $\frac{neuer\ Ort - alter\ Ort}{Zeit}$ (mathematisch in 2 Dimensionen: $v = \sqrt{\dot{d_x}^2 + \dot{d_y}^2}$)

Gutes Modell für die Bewegung eines mobilen Roboters in der Ebene entlang einer beliebigen „Kurve“ k:



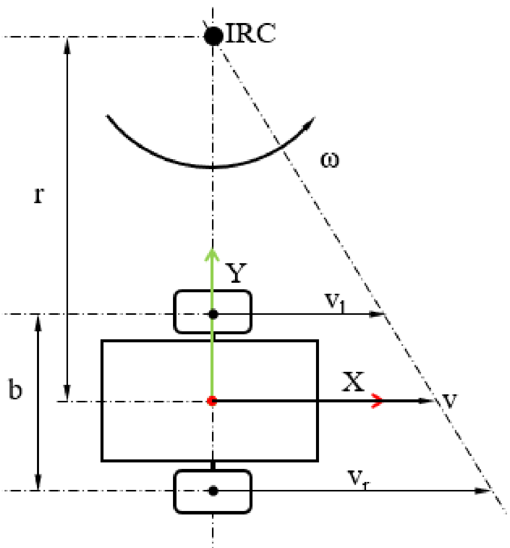
Um vom alten Ort (Punkt) auf der Kurve zu einem neuen zu kommen, benötigen wir neben der **linearen Geschwindigkeit v** ($\frac{\text{Änderung des Ortes}}{Zeit}$) auch eine **Winkelgeschwindigkeit ω** ($\frac{\text{Änderung der Richtung}}{Zeit}$).

Dieses ω kann über einen gedachten Kreis mit dem Mittelpunkt ICR (instantaneous center or rotation = „momentaner Mittelpunkt der Drehbewegung“) berechnet werden:

$\omega = \frac{v}{r}$ (lin. Geschw. durch Kreisradius) mit $r = \frac{1}{\kappa}$ (Kehrwert der Krümmung)

Bei unserem Roboter ergibt sich die (Dreh-)bewegung durch eine (unterschiedliche) Ansteuerung der 2 Räder.

Ein zusätzlicher Einflussfaktor ist dabei der Abstand der Räder b (wheel base):



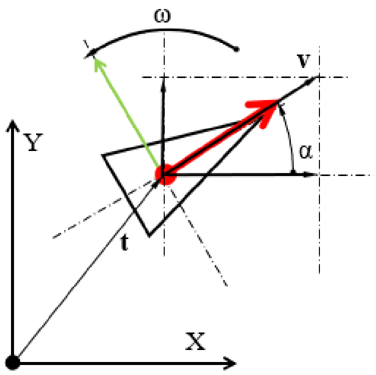
$$\begin{cases} v = \dfrac{v_r + v_l}{2} \\ \omega = \dfrac{v_r - v_l}{b} \end{cases}$$

$$r = \frac{v}{\omega} = \frac{b}{2} \frac{v_r + v_l}{v_r - v_l}$$

Die direkte Kinematik errechnet aus den Geschwindigkeiten der Räder (v_l und v_r) das v und ω,

die inverse Kinematik aus der Vorgabe von v und ω die Radbewegungen v_l und v_r

Direkte Kinematik:



$$\begin{cases} \dot{t}_x = v \cdot \cos(\alpha) = \dfrac{v_r + v_l}{2} \cdot \cos(\alpha) \\ \dot{t}_y = v \cdot \sin(\alpha) = \dfrac{v_r + v_l}{2} \cdot \sin(\alpha) \\ \dot{\alpha} = \omega = \dfrac{v_r - v_l}{b} \end{cases}$$

In der Praxis rechnet man keine (mathematisch exakten) Ableitungen ($\dot{t_x}$, $\dot{t_y}$, $\dot{\alpha}$), sondern den Differential- bzw. Differenzenquotienten im aktuellen Punkt (k+1). Man spricht von einer diskreten Kinematik.

Bezeichnung	Formel	Bedeutung	Geometrische Bedeutung
Differenzenquotient	$\frac{f(x)-f(x_0)}{x-x_0}$	mittlere Änderungsrate	Sekantensteigung
Differentialquotient	$\lim\limits_{x \to x_0} \frac{f(x)-f(x_0)}{x-x_0}$	lokale bzw. momentane Änderungsrate	Tangentensteigung

$$\dot{t_x}(k) = \frac{t_x(k) - t_x(k-1)}{\Delta t}$$

$$\dot{t_y}(k) = \frac{t_y(k) - t_y(k-1)}{\Delta t}$$

$$\alpha(k) = arctan \frac{t_y(k) - t_y(k-1)}{t_x(k) - t_x(k-1)}$$

$$\dot{\alpha}(k) = \frac{\alpha(k) - \alpha(k-1)}{\Delta t}$$

$$v(k) = \sqrt{\dot{t}_x^2 + \dot{t}_y^2}$$

$$\omega(k) = \dot{\alpha}(k)$$

Damit ergibt sich

für die (diskrete)

inverse Kinematik:

$$\begin{cases} v_l(k) = \dfrac{v(k) - b\omega(k)}{2} \\ v_r(k) = \dfrac{v(k) + b\omega(k)}{2} \end{cases}$$

1. In unserem Programm, wo eine Kurve k mathematisch vorgegeben ist, müssen wir uns also die x- und y-Komponenten des aktuellen Punkt $P(k) = \binom{t_x(k)}{t_y(k)}$ ausrechnen.

2. Mit dem gespeicherten alten Punkt $P(k-1) = \binom{t_x(k-1)}{t_y(k-1)}$ können nun über $\dot{t_x}(k)$ und $\dot{t_y}(k)$ die lineare Geschwindigkeit v(k) und die Winkelgeschwindigkeit ω(k) berechnet werden.

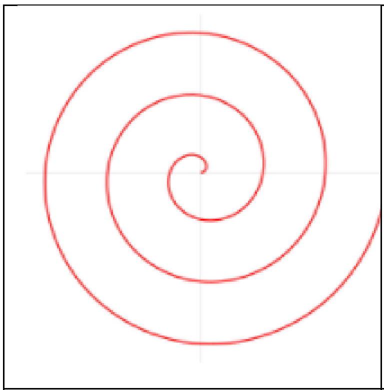
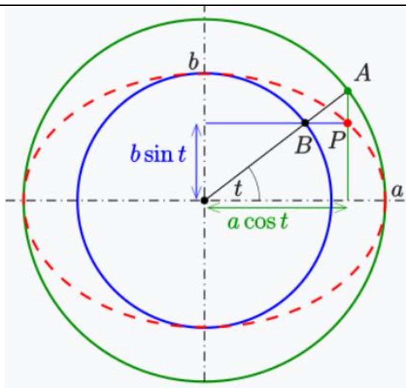
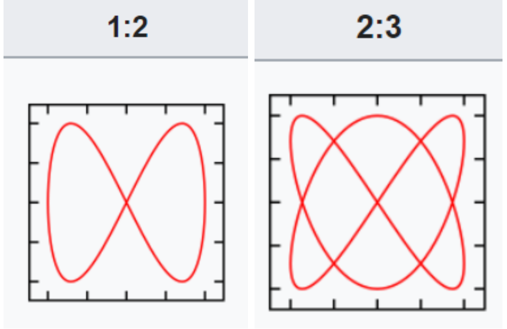
3. Über das Modell mit dem ICR können über die inverse Kinematik die Ansteuerungen für den linken und rechten Motor $v_l(k)$ und $v_r(k)$ ausgerechnet werden.

Unserem Roboter fehlen Sensoren, welche die Drehbewegung der Räder messen können (Odometrie) und wir steuern die beiden Motoren mit einem Wert zwischen 0 und 255 an (der Wert beeinflusst den PWM-Tastgrad der Motortreiber). Wir können also leider keine exakten Wege programmieren, sondern müssen dies ausprobieren. Dabei wird sich jeder Roboter individuell ein wenig anders verhalten…

Als Kurven, die wir mit dem Roboter fahren können, bieten sich (geschlossene) ebene Figuren an:

1.) Spirale
2.) Ellipse
3.) Lissajous-Figuren

x und y werden bei diesen Figuren am einfachsten aus der Polardarstellung errechnet. Dabei läuft eine Variable φ (phi) in kleinen Schritten von 0° (0 rad) nicht nur bis 360° (2π), sondern bis unendlich…

		
$x = a \cdot \varphi \cdot cos\varphi$ $y = a \cdot \varphi \cdot sin\varphi$ Wenn $a = \frac{1}{2\pi}$ dann ist der Abstand zwischen den Spuren 1	$x = a \cdot cos\varphi$ $y = b \cdot sin\varphi$	$x = a \cdot \sin(n\varphi)$ $y = b \cdot \sin(m\varphi)$ Das Verhältnis n : m bestimmt das Aussehen. Bei diesen Beispielen (und Formeln) ist die Phasenverschiebung 0

Ein paar Vorschläge zum Programmieren mit makecode:

- Da wir keine genauen Distanzen einhalten können, rechnen wir in Zentimeter (z.B. wheel base = 10) und verwenden bei der Motor-Ansteuerung die Variable „scale“
- Δt (Variable „delta-t“) kann z.B. 100ms sein; das ist für unsere langsame Bewegung ausreichend. In der Hauptschleife „dauerhaft“ reicht dazu ein Pause-Befehl von 70ms, da die restlichen 30ms hauptsächlich bei den Ansteuerungen der Motoren vergehen.
- Um den Ablauf auf dem Roboter zu verfolgen, kann man φ (Variable „phi“) als Säulendiagramm auf dem micro:bit ausgeben (Zahlenausgaben würden blockieren!). Dazu kann man eine Modulo-Division durch 2π bzw. 6,2832 (Block „Rest von…“ in der „Mathematik“-Bibliothek) machen und den Maximalwert des Säulendiagrammes auf 2π setzen
- Die Winkelfunktionen sind als Block „Quadratwurzel…“ in der „Mathematik“-Bibliothek verfügbar und werden über das Auswahlmenü (weißer kleiner Pfeil nach unten im Block) konfiguriert
- Die arctan-Funktion (atan2) springt bei 180° von + π auf – π: wir müssen ω deshalb begrenzen

Hier eine (noch nicht vollständige) Vorlage für eine Spirale mit a = 10/2π: