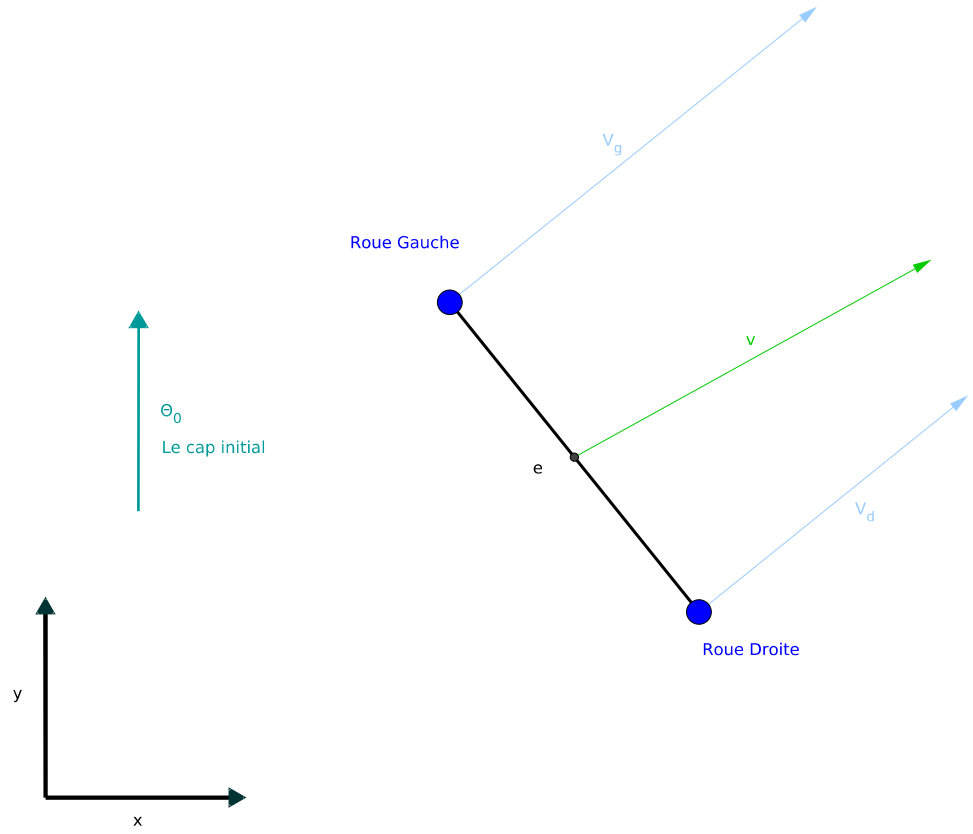
Odométrie

1. Données – glossaire

Pour commencer on définit un certain nombre de valeurs :

- d_g d_d : Déplacement des roues Gauche et Droite
- v_g v_d : Vitesse des roues Gauche et Droite
- x, y, Θ : Coordonnées et orientation au centre des 2 roues
- d : Déplacement du robot
- v : Vitesse du robot
- e : Écart entre les roues



2. Modèle simplifié

Pour commencer à étudier le mouvement du robot, nous allons utiliser un modèle simplifié où l'ont considère que la vitesse du robot est fixe, et que le robot parcoure un arc de cercle. C'est à dire qu'une de ses roue va plus vite que l'autre.

Lorsqu'on parle du robot, cela signifie en fait qu'on regarde le point au centre des 2 roues. La trajectoire du robot suit donc sur un cercle de rayon R , et si la vitesse angulaire est $\frac{d\Theta}{dt}$, alors la vitesse du robot est $v = R\frac{d\Theta}{dt}$

À partir de cela, on peut déterminer la vitesse de chaque roues :

- $v_g = (R - \frac{e}{2})\frac{d\Theta}{dt} = (R - \frac{e}{2})\frac{v}{R}$
- $v_d = (R + \frac{e}{2})\frac{d\Theta}{dt} = (R + \frac{e}{2})\frac{v}{R}$

3. Modèle inverse

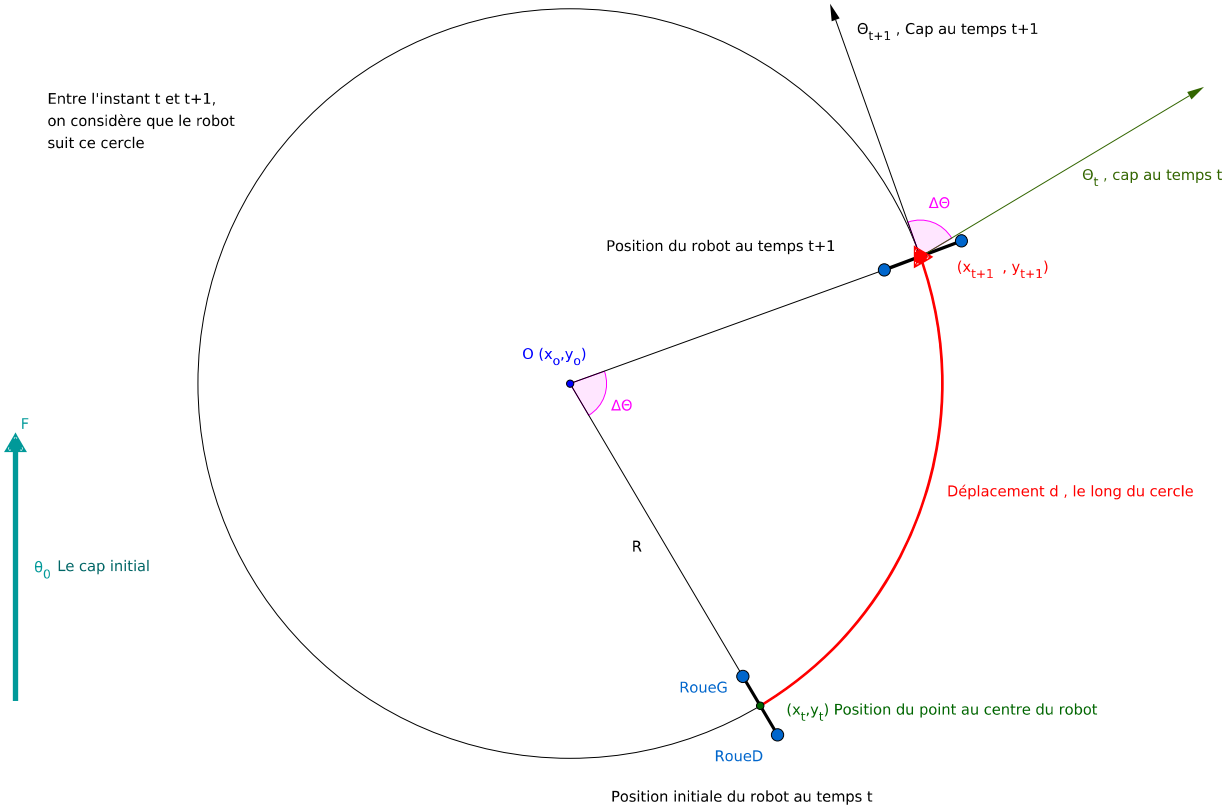
Nous allons à partir de maintenant considérer qu'entre 2 instant très proche, la trajectoire du robot est un arc de cercle. À partir de cette hypothèse, et en inversant le modèle nous pouvons retrouver le rayon de courbure R et la vitesse v du robot.

On trouve donc les relations suivantes :

- $v = \frac{v_g + v_d}{2}$, ce qui permet de déduire $d = \frac{d_g + d_d}{2}$
- $R = \frac{e}{2}\frac{v_d + v_g}{v_d - v_g}$

4. Calcul de l'angle

Nous allons étudier la variation du cap sur une itération du calcul d'odométrie. L'ancien cap sera appelé Θ_t et le nouveau Θ_{t+1}, l'angle entre les deux : $\Delta\Theta$.
Comme nous considérons que sur l'intervalle de temps [t,t+1], que le robot suit un arc de cercle, nous allons appeler le centre de ce cercle le point O de coordonnées (x_o, y_o).



Puisque l'intervalle entre chaque itération est constant, et qu'on connait la distance parcourue par chaque roue, on peut calculer la vitesse de chaque roue : v_g v_d.
À partir ce cela et du modèle inverse, on peut calculer le rayon de courbure R et la vitesse v du robot.

Avec ces informations, on peut déterminer la variation de cap du robot $\Delta\Theta = \frac{d}{R}$

NB : Il parait évident que les angles prit en exemple sur les graphiques sont volontairement trop élevés, en situation réelle, il faut faire en sorte que cette angle soit le plus petit possible. De manière à limiter l'erreur entre la trajectoire supposée et la trajectoire réelle du robot.

5. Calcul de la position en (x,y)

Le calcul de la position se fait en 2 partie, tout d'abord nous devons trouver la position du point $O\ (x_o, y_o)$, centre du cercle.

Pour cela, nous utilisons le cap précèdent Θ_t pour calculer :

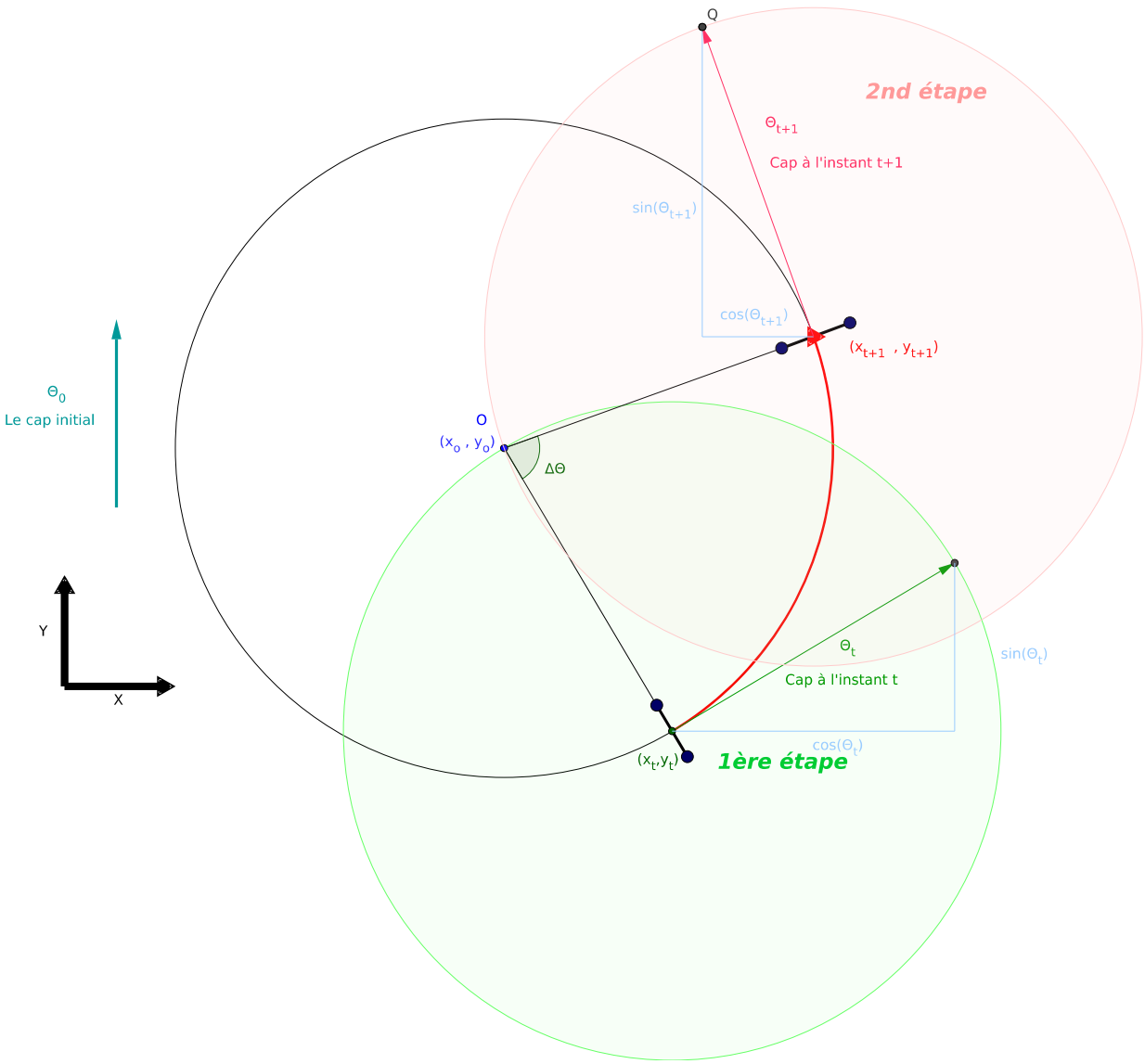
- $R * \cos(\Theta_t)$, ce qui nous donne la distance entre x_t et x_o
- $R * \sin(\Theta_t)$, ce qui nous donne la distance entre y_t et y_o

Nous pouvons donc établir que la position du point $O\ (x_o, y_o)$ est :

$$x_o = x_t - R\sin(\Theta_t)$$
$$y_o = y_t + R\cos(\Theta_t)$$

Une fois la position du point O calculé, et en utilisant la même logique, on détermine que la nouvelle position du robot (x_{t+1}, y_{t+1}) est :

$$x_{t+1} = x_O + R * \sin(\Theta_{t+1})$$
$$y_{t+1} = y_O - R * \cos(\Theta_{t+1})$$



6. Un exemple pratique

Imaginons maintenant que nous faisons notre petite odométrie avec comme matériel des codeurs quelconques et surtout une Arduino ! Utiliser cette petite bestiole va nous imposer de limiter au maximum les calculs à effectuer.

Pour commencer, pourquoi s'embêter à calculer la vitesse de chaque roue ? Après tout, si on remet à zéro les compteurs à chaque rafraichissement de l'odométrie, la valeur que l'on récupère représente une vitesse en distance élémentaire de l'encodeur par période de rafraichissement. On peut très bien calculer la position actuelle dans une unité arbitraire du moment que l'on sait comment la convertir en unité du SI.

À partir de cette situation, le calcul du rayon de courbure devient $R = \frac{e}{2} \frac{d_d + d_g}{d_d - d_g}$, et par conséquent le calcul de la variation du cap se simplifie et devient : $\Delta\Theta = \frac{d_d - d_g}{e}$

Ensuite, pour éviter de faire des calculs en virgule flottante sur de petits nombres qui sont souvent générateur d'erreur de calcul, on va transformer nos distances élémentaires d'encodeur, en une unité choisis arbitrairement. Appelons la « Unitée Odométrique » ou UO, elle sera évidement très petite, et pour ne pas trop alourdir le calcul, le plus simple consiste à multiplier nos valeurs par une puissance de 2 (exemple: 1024).

Assez souvent le robot ne bouge pas entre 2 période de rafraichissement de l'odométrie, donc la distance va être nulle, ainsi que le delta théta. Si nous gérons pas ce cas limite, nous devrions avoir au pire un plantage (division par zero) ou au « mieux » un rayon de courbure très grand (NB : pour une ligne droite, le rayon de courbure est infinie).

7. Pseudo Code de l'exemple

```
#define UO_PAR_UNITE_CODEUR 1024

// On récupère les données enregistrés par les codeurs
Dg = codeurG.valeur() * UO_PAR_UNITE_CODEUR ;
Dd = codeurD.valeur() * UO_PAR_UNITE_CODEUR ;

//On reset les codeurs
codeurG.reset() ;
codeurG.reset() ;

//On commencer par calculer la variation d'angle et la distance parcourue
d = ( Dd+Dg ) / 2 ;
diffCodeurs =  Dd – Dg;
deltaTheta = ( diffCodeurs ) / e;


if ( abs(diffCodeurs) < 1 ) // On considère la trajectoire comme rectiligne
{
        X = X + d*cos(theta);
        Y = Y + d*sin(theta);
}
else
{
        //On Calcule le rayon de courbure et le point au centre du cercle
        R = d / deltaTheta;
        X0 = X - R*sin(theta);
        Y0 = Y + R*cos(theta);

        //On met à jour la position actuelle
        theta = theta + deltaTheta;
        X = X0 + R*sin(theta);
        Y = Y0 + R*cos(theta)
}
```

8. Sitographie

- Code source de l'asservissement de Xevel (www.xevel.fr) utilisé à l'arrache pendant la coupe 2009/2010 : http://trac.assembla.com/XD_DSbot/browser/tags/2009/Arduino/CarteMoteur
- Un article similaire à celui sur wikipedia : http://www.robotique.wikibis.com/odometrie.php
- Un article du club elek : http://clubelek.insa-lyon.fr/joomla/fr/base_de_connaissances/informatique/asservissement_et_pilotage_de_robot_auto_2.php
- Un article en anglais expliquant la théorie développée ici et un modèle plus simple : http://rossum.sourceforge.net/papers/DiffSteer/
- Pour trouver d'autres articles essayez les mots clefs : « dead reckoning » avec « differential drive », « odométrie », ou même « asservissement »