*Le robot n’a qu’un seul but dans la vie :*

*Rejoindre le ghost*

Un ordre (de déplacement) est caractérisé par un point d’arrivé (x1,y1), un cap d’arrivé (theta1) et une flèche (cf plus tard pour la flèche).

A partir de cet ordre, on trouve un polynôme X et un polynôme Y tel que

X(0)=x0

Y(0)=y0 (où (x0,y0) est le point de départ avec le cap de départ theta0)

Le cap initial vaut theta0 (ie Arctan(Y’(0)/X’(0))=theta0)

X(1)=x1

Y(1)=y1

Le cap final vaut theta1 (ie Arctan(Y’(1)/X’(1))=theta1)

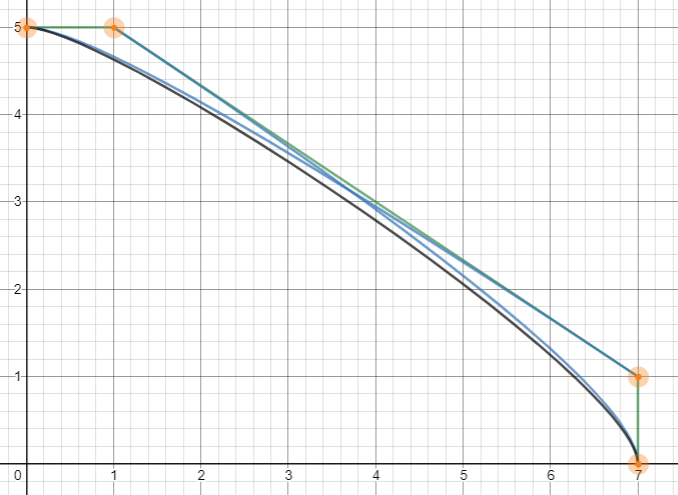
On remarque que le cap du ghost vaut systématiquement Arctan(Y’/X’)

On connait donc la position et le cap du ghost tout au long de sa trajectoire.

(Rq : La variable de ces polynome est entre 0 et 1, ce ne sont pas des fonctions du temps. J’appelle t\* cette variable qui varie entre 0 et 1 )

La formule de X et Y se trouve grâce aux courbes de Bezier

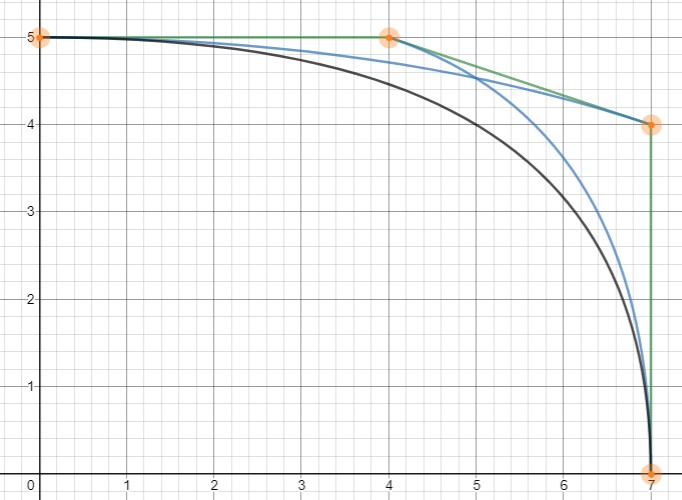
<https://www.desmos.com/calculator/cahqdxeshd>



Je pars de (0,5) avec theta=0

Pour arriver a (7,0) avec theta=-PI/2

Une autre solution :



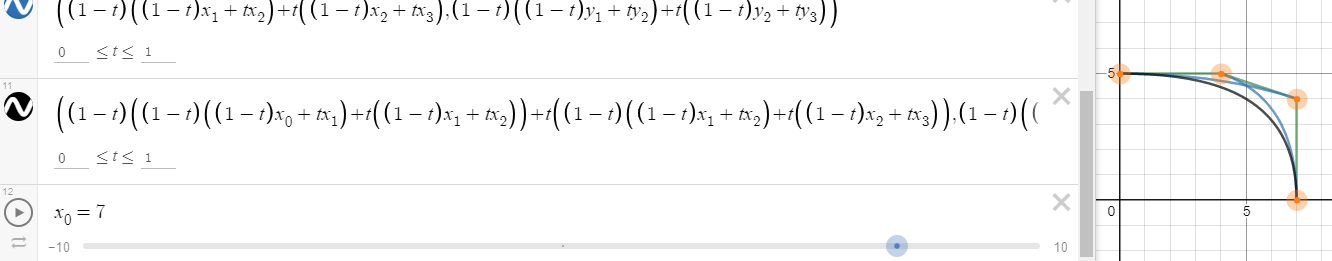
J’ai appelé la flèche, l’éloignement des points intermédiaires.

Dans ce dernier exemple elle vaut 4 alors qu’elle valait 1 avant.

(En pratique, dans le programme, la flèche est définie par rapport à la distance entre le point de départ et le point d’arrivée. Ici ces deux points sont éloignés de sqrt(7^2 + 5^2)=8,6

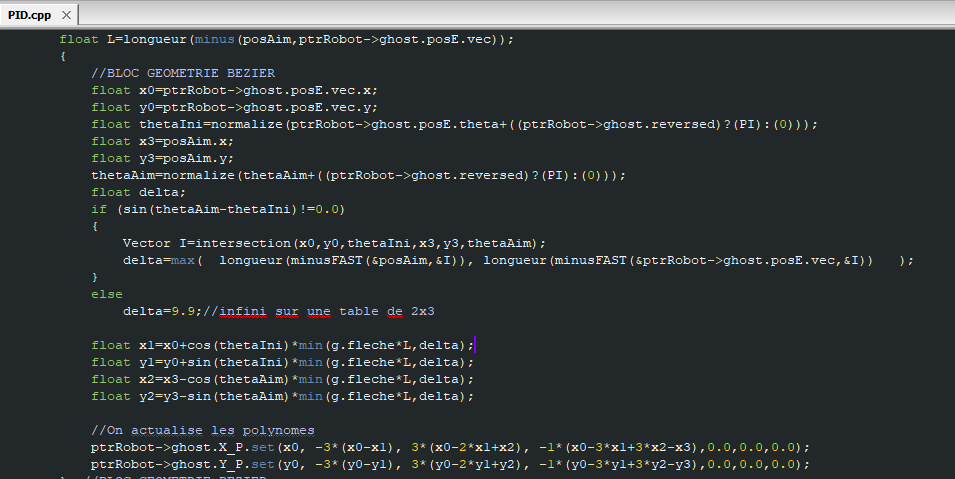
Donc dans le programme on aurait défini la flèche comme état 4/8,6= 0.46)

Sur ce meme site, on trouve la formule qui sera à implémenter :

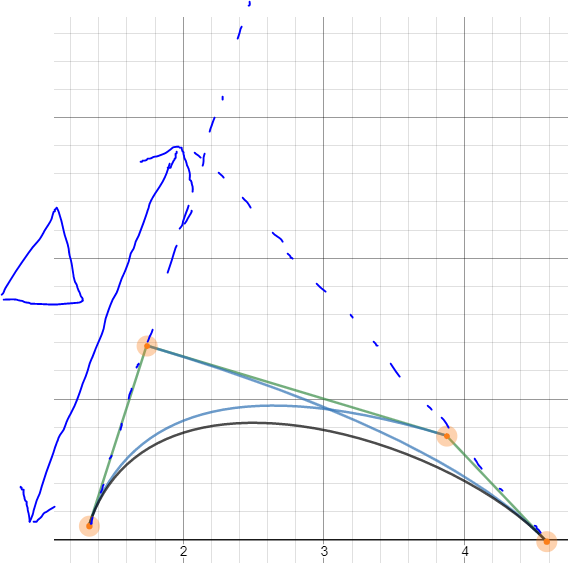
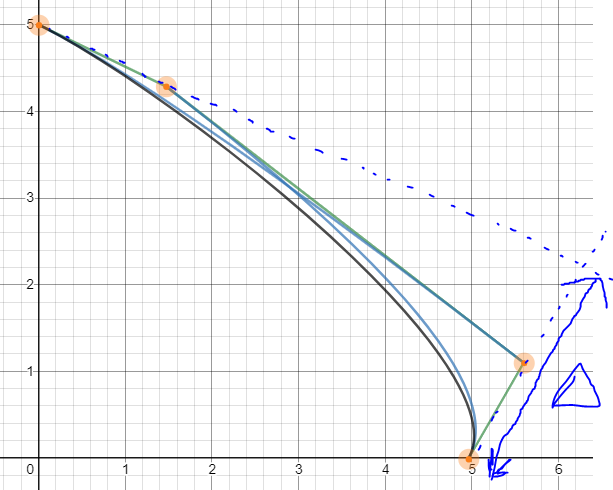


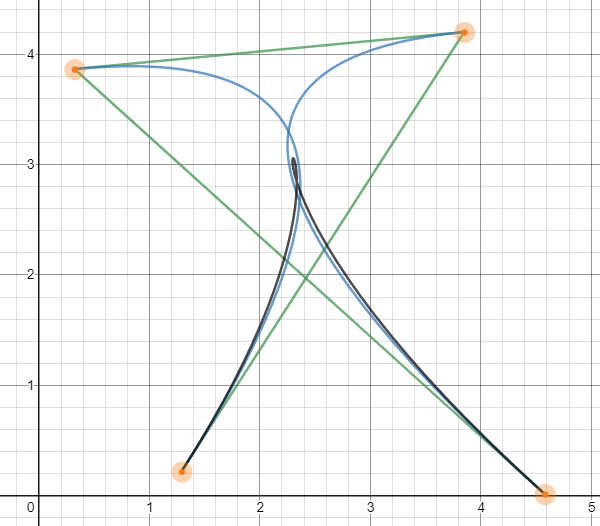
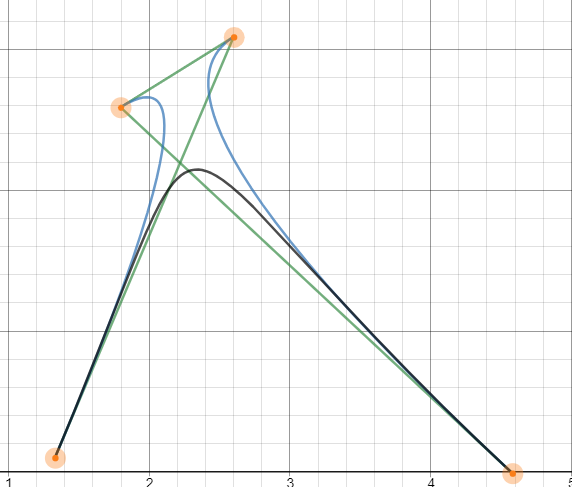
(Ouai elle est longue)

On la retrouve (sous sa forme développée dans PID.cpp)



Pour info, le « delta » est cette longueur ci :



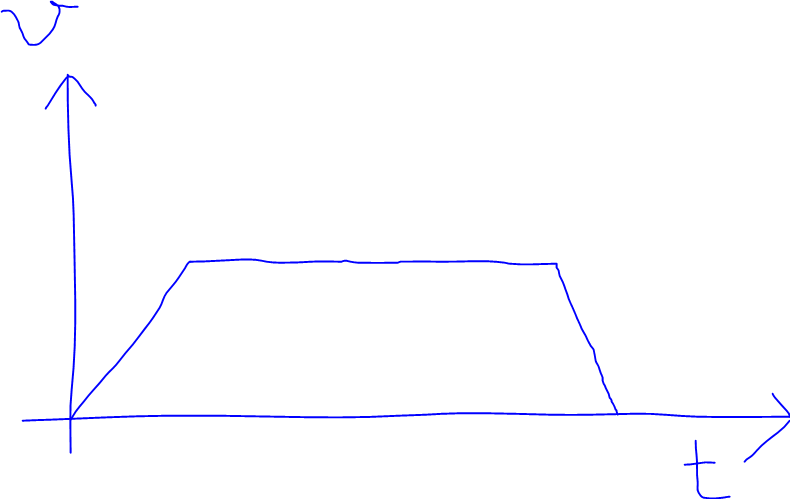
Si la flèche dépasse delta, tu vas te retrouver avec des rayons de courbure bien trop importants :

C’est un critère arbitraire, mais bien pratique (surtout que le robot est susceptible de recalculer une trajectoire n’importe quand si il a un problème (genre un choc))

Dernier point important : la gestion de l’accélération/décélération

On veut que le robot accélère et décélère doucement. Donc il faut que le ghost accélère et décélère doucement :

On définit donc la vitesse V en fonction du temps comme ça :



On connait X(t\*) et Y(t\*). Mais on ne connait pas X(t) et Y(t).

On connait la vitesse vraie V²(t)=X’(t)²+Y’(t)²

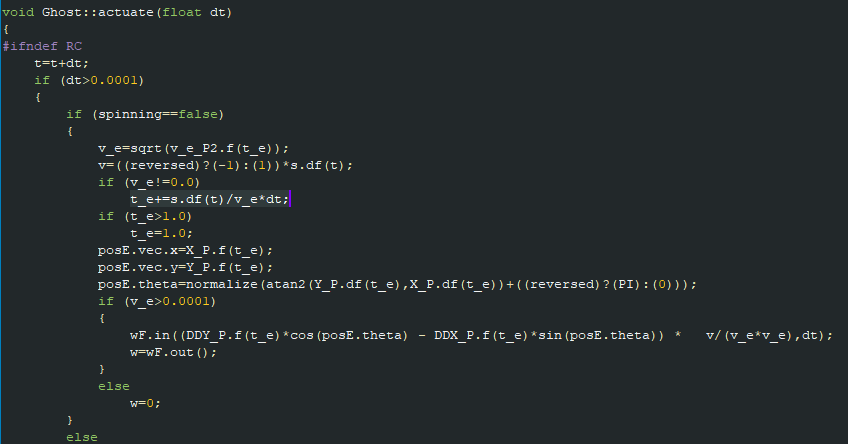
Et on sait calculer la vitesse fictive V\*²(t\*)=X’(t\*)²+Y’(t\*)²

On a V=dPosition/dt et V\*=dPosition/dt\*

Alors on trouve Vdt=V\*dt\* ie dt\*/dt=V/V\*

Donc si entre deux tours de boucle, il s’est écoulé 1ms, il faut faire avancer t\* de (V(t)/V(t\*)) x 1ms

Dans le code (Ghost.cpp) ça donne ça :



~\_e signifie que c’est une grandeur fictive (e pour étoile)

« s » est l’abscisse curviligne s.df est sa dérivée. C’est donc la vitesse réelle.

Le calcul de theta est simple (c’est la ligne avec le atan2(Y\_P.df….)), mais celui de omega (w) est plus chaud…

On en parlera plus tard ^^

Pour le coup, c’est une partie du code qui marche bien (vu que j’ai eu le temps de coder ça pépère tout au long de l’année. C’est pas un code rushé en quelque jours comme celui de la méga…).

**Le GROS souci** c’est quand on passe d’un ordre à un autre. Il faut faire très attention à ne pas téléporter le ghost

Et le Robot dans tout ça ?

Le robot doit se superposer au ghost.

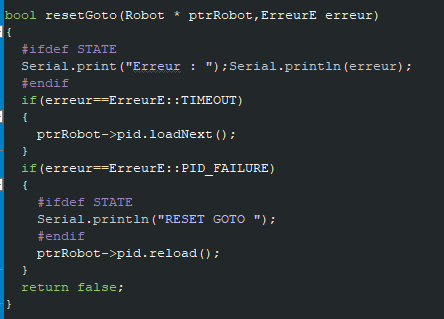
Pour cela, le robot essaye d’avoir toujours le meme cap et la meme abscisse curviligne que le ghost.

Lorsqu’on détecte que le robot s’est trop éloigné, une erreur « PID FAILURE » est levée.

**Un ordre est toujours accompagné d’une contre mesure**

Une contre mesure est une fonction qui est appelée lors qu’une erreur est levée.

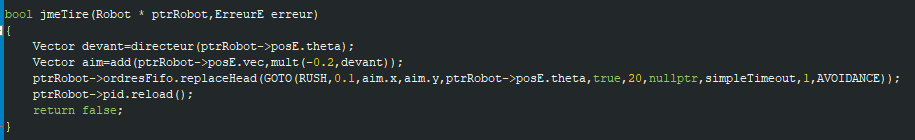
Voici par exemple la contre mesure « reset Goto »



Si l’erreur en question est un timeout, on passe à la suite (loadNext).

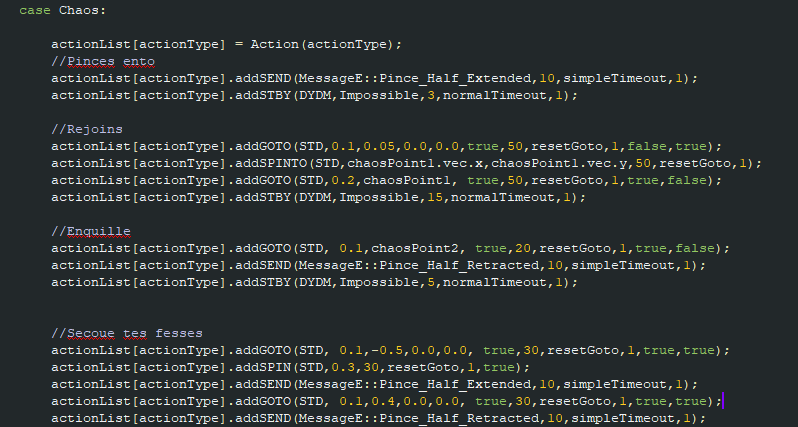
Et si l’erreur est un pid failure, on relance l’action en court (reload). Ca va recalculer les polynome X et Y.

Voici la contre mesure « jmeTire » (à chanter avec l’accent de Maitre Gims)



Il fait reculer le robot de 20cm puis recalcule l’action en cours. C’est une contremesure qui devait être appelée en cas de détection d’adversaire qui dure trop longtemps… En pratique il a fallu coder la Mega en panique donc bon, les évitements n’ont pas été notre priorité. (oui ça sent le sel par ici #JeSuisBenoitVarillon)

Ca c’est les actions qui s’effectuent lors de l’action « Chaos »



La contremesure est le 8eme argument des GOTO, le 3eme des SEND, le 4eme des STBY (standby) et le 4eme des SPIN.

Un standby qui déclanche un timeout, c’est normal, c’est juste qu’on est arrivé au bout du standby. Donc la contremesure à utiliser c’est « normalTimeout ». Qui s’en balek de l’erreur timeout.

Pour les goto et les spin on utilise « resetGoto ». (il aurait du s’appeler resetDeplacement ouai…)

Pour les SEND, on utilise simpleTimeout, car ça n’est pas normal d’avoir timeout sur un SEND (envoi d’un message à la Mega). Le simpleTimeout passe à la suite, mais garde en mémoire qu’il y a eu un fail. SI l’action suivante fail aussi, le robot abandonne le bloc d’action complet (genre le bloc Chaos)

Je pense vraiment que cette histoire de contremesure était une bonne idée. L’implémentation n’est pas parfaite dans mon code, mais l’idée est à garder ;)

Le PID

Notre force, c’est qu’on connait le futur.

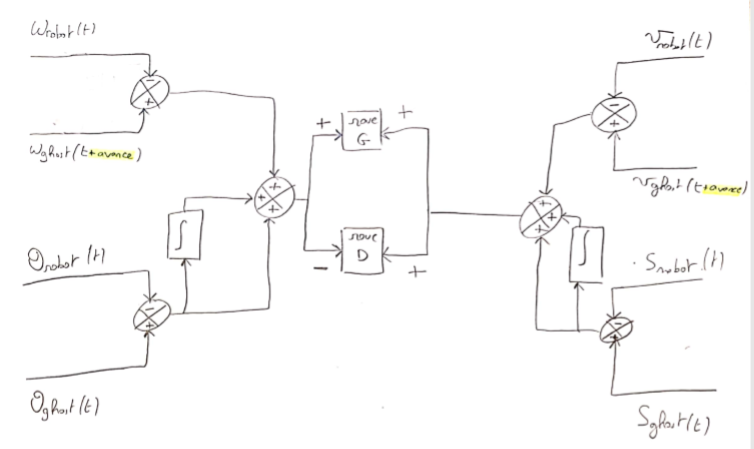
Vu qu’on a un polynome pour la position et le theta du ghost, on connait sa vitesse vraie et son omega vrai. Pas besoin de faire de dérivée approximative (et avec de la latence).

Il y a deux PID. Un qui veut que le cap du robot soit celui du ghost, et un qui veut que l’abscisse curviligne du robot soit celui du ghost.

J’essaye d’avoir la meme position que le ghost actuellement, mais j’essaye d’avoir la vitesse que le ghost aura prochainement. (idem pour le PID de rotation)

En faisant ça, on prend très légèrement de l’avance sur le ghost, et c’est ça qui fait qu’on overshoot jamais. (on freine avant meme que le ghost commence à ralentir)

En général cet avance était de 300ms environ



Rq : L’autotune des PID était une perte de temps. Je ne vous conseille pas d’essayer ^^