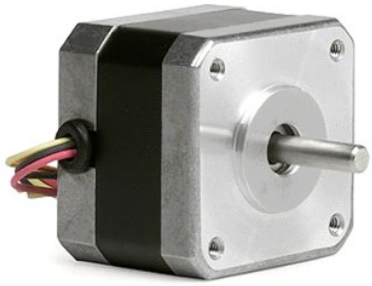
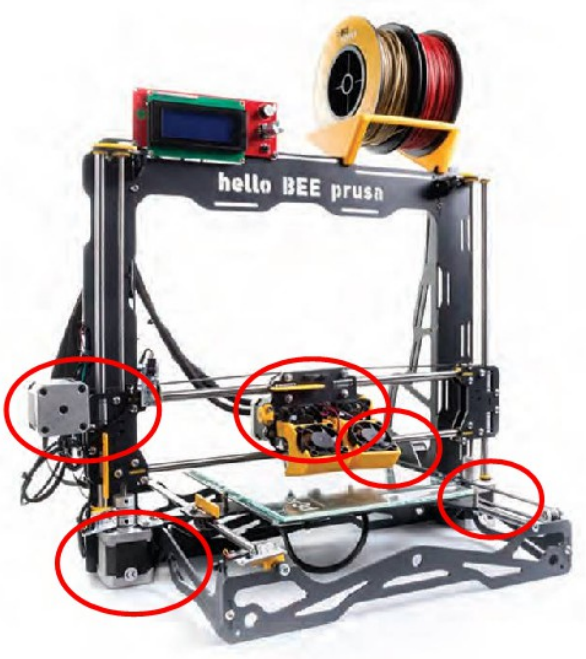
Modélisation d'un moteur pas à pas

Présentation des moteurs pas à pas



Familles de moteurs pas à pas

Les moteurs pas à pas sont utilisés pour les positionnements angulaires précis (imprimantes, scanners, disques durs ...). Par exemple, l'imprimante 3D HelloBeePrusa en compte 5 ! Contrairement aux moteurs à courant continu, il ne nécessitent pas de boucle d'asservissement et sont plus simples à commander. Dans tous les types de moteur, on positionne le rotor en modifiant la direction d'un champ magnétique créé par les bobinages du stator. Ils nécessitent donc non seulement un circuit de puissance mais également un circuit de commande qui contient une partie logique. Cette dernière détermine pour chaque pas quelles sont les bobines alimentées et le sens de rotation. La fréquence de l'horloge du circuit logique détermine la vitesse de rotation comme nous le verrons dans la modélisation.



Les moteurs pas à pas sont classés en fonction de la manière dont on alimente les bobines et du nombre de bobines ainsi que de la nature du rotor qui est soit un barreau de fer doux (moteur à réluctance variable), soit un rotor avec aimant permanent.

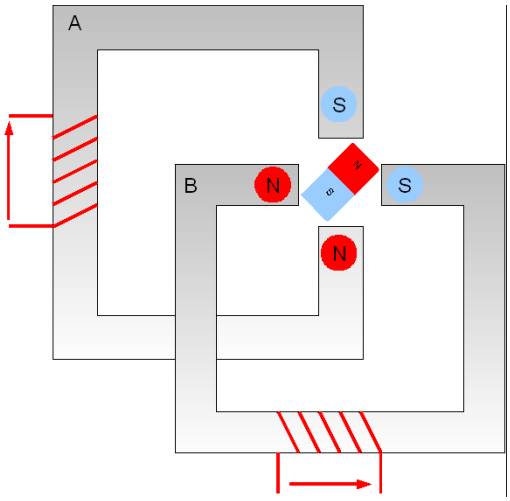
On parle ainsi de moteur à réluctance variable ou à aimant permanent. Les moteurs hybrides allient les deux principes (plusieurs dents aimantées sur le rotor).

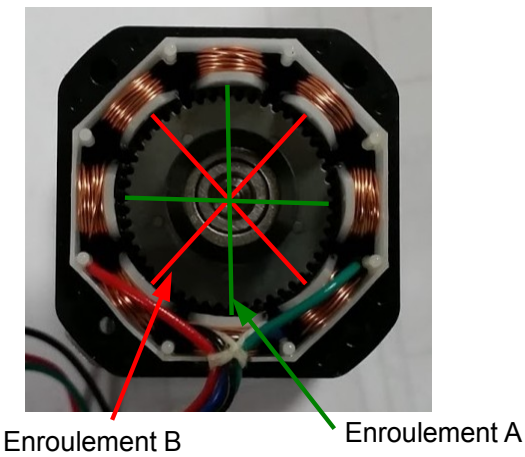
Nous n'étudierons dans cet article que le principe de base des moteurs pas à pas sans distinguer la technologie de réalisation de la structure du moteur. Le moteur Nema 17 (moteur pas à pas 1,8° ou 200 pas par révolution) fait ainsi parti des moteurs hybrides les plus répandus.

Moteurs pas à pas à aimants permanents

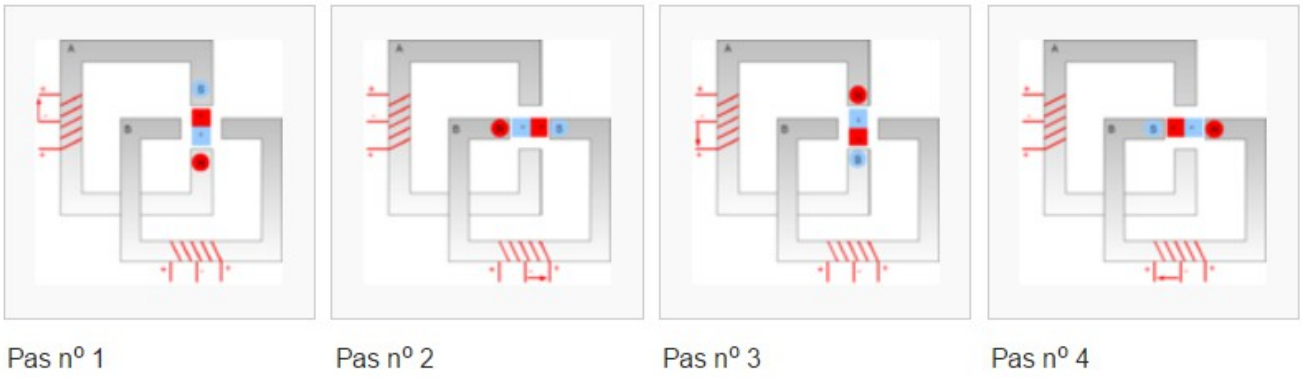
Dans la famille des moteurs pas à pas à aimants permanents, on distingue les moteurs pas à pas bipolaires qui contiennent deux enroulements (bobines). Chaque enroulement est commandé en courant positivement ou négativement. Le rotor aimanté possède plusieurs pôles Nord-Sud. Ces moteurs possèdent 4 fils (2 par bobine).

La photo de la structure interne du moteur pas à pas Nema 17 montre deux enroulements répartis en croix ainsi qu'un rotor constitué de nombreux pôles.



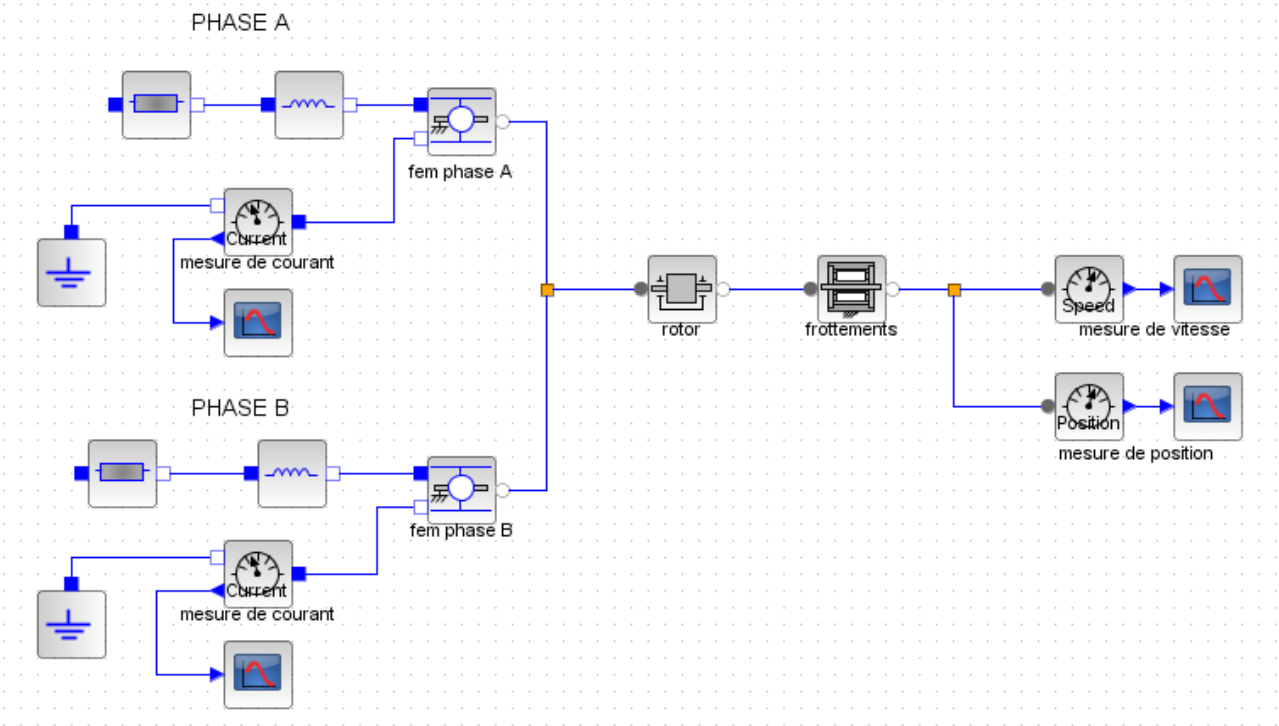


Les moteurs pas à pas unipolaires possèdent 6 fils correspondant à 4 demi-bobines. Chaque demi-bobine est alimentée toujours de la même manière ce qui permet de ne jamais avoir besoin d'inverser le sens du courant dans une bobine. Ils possèdent davantage d'enroulements et sont donc plus compliqués à réaliser.



Modélisation d'un moteur pas à pas

La modélisation multiphysique permet de modéliser relativement simplement un moteur pas à pas. Chaque phase est modélisée par une association d'une inductance, d'une résistance et d'une fem particulière.
L'induction engendre deux relations entre, d'une part, vitesse du rotor (notée ω) et tension induite e et d'autre part entre le courant i dans la phase et le couple magnétique exercé C.
Pour la phase A, les relations sont les suivantes $e_A = k\omega\sin(n\phi)$ et $C_A = ki_A\sin(n\phi)$ avec ϕ l'angle de rotation du rotor, k une constante de fem et n le nombre de paires de pôles aimantés (pour un moteur tel que le NEMA 17 possédant 2 paires de bobines A et B et ayant 200 pas par tour, n est égal à 50).
Pour la phase B, les relations sont déphasées de 90° : $e_B = k\omega\sin\left(n\phi+\frac{\pi}{2}\right)$ et $C_B = ki_B\sin\left(n\phi+\frac{\pi}{2}\right)$.
Les actions exercées par ces deux phases sont appliquées sur le rotor modélisé par une inertie (axe en rotation) à laquelle est ajoutée des frottements visqueux (et secs).



Pour les simulations réalisées dans la suite, on utilise les valeurs suivantes obtenues principalement dans la documentation du moteur NEMA 17.

k	0,3 Nm/A	L	0,00028 H
n	50	J	$1e^{-7}$ kg m^2
R	1,65 Ohm	f	0,001 Nm s/rad

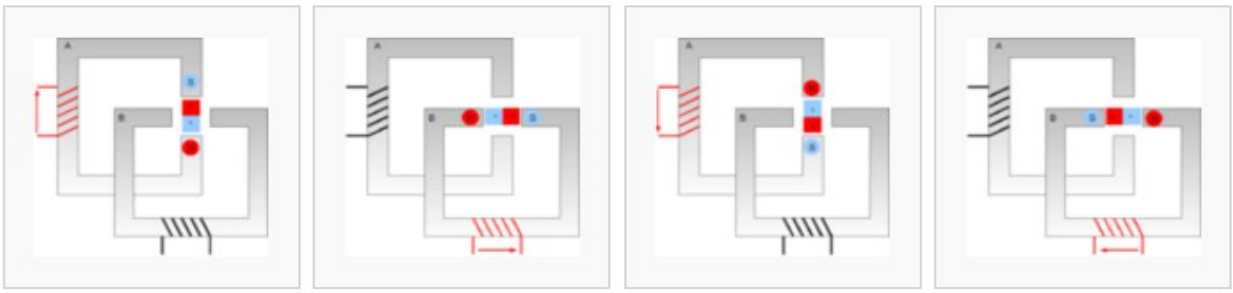
Fonctionnement du moteur pas à pas bipolaire

Modes pas entier - demi-pas

Plusieurs fonctionnements sont possibles pour les moteurs pas à pas en fonction de l'alimentation des bobines.

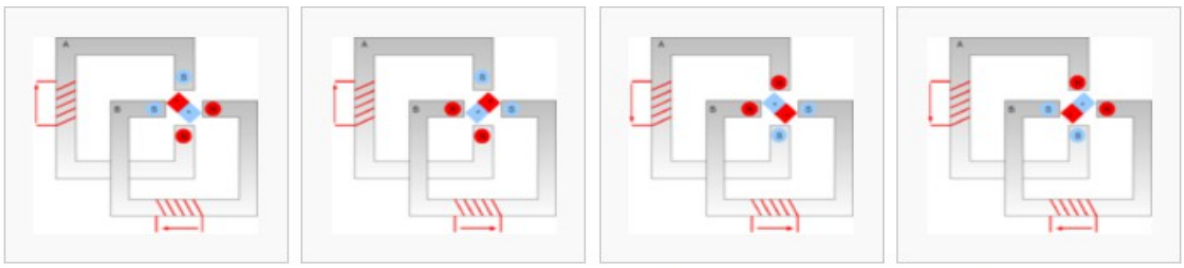
Le premier mode est le mode à pas complet (« full step » en anglais) qui consiste à alimenter successivement les bobines pour faire avancer le moteur d'une dent à chaque fois. L'illustration simplifiée suivante suppose que deux enroulements sont disponibles et que le rotor n'est constitué que d'un seul aimant. Le principe est exactement le même avec des enroulements répartis différemment et plus de pôles sur le rotor, le couple sera plus grand et la précision plus grande.

La flèche rouge indique que le courant circule dans une bobine dans un sens donné. On représente en noir la bobine non alimentée. L'aimant s'aligne selon le champ B créé par la bobine.



Pas n° 1 | Pas n° 2 | Pas n° 3 | Pas n° 4

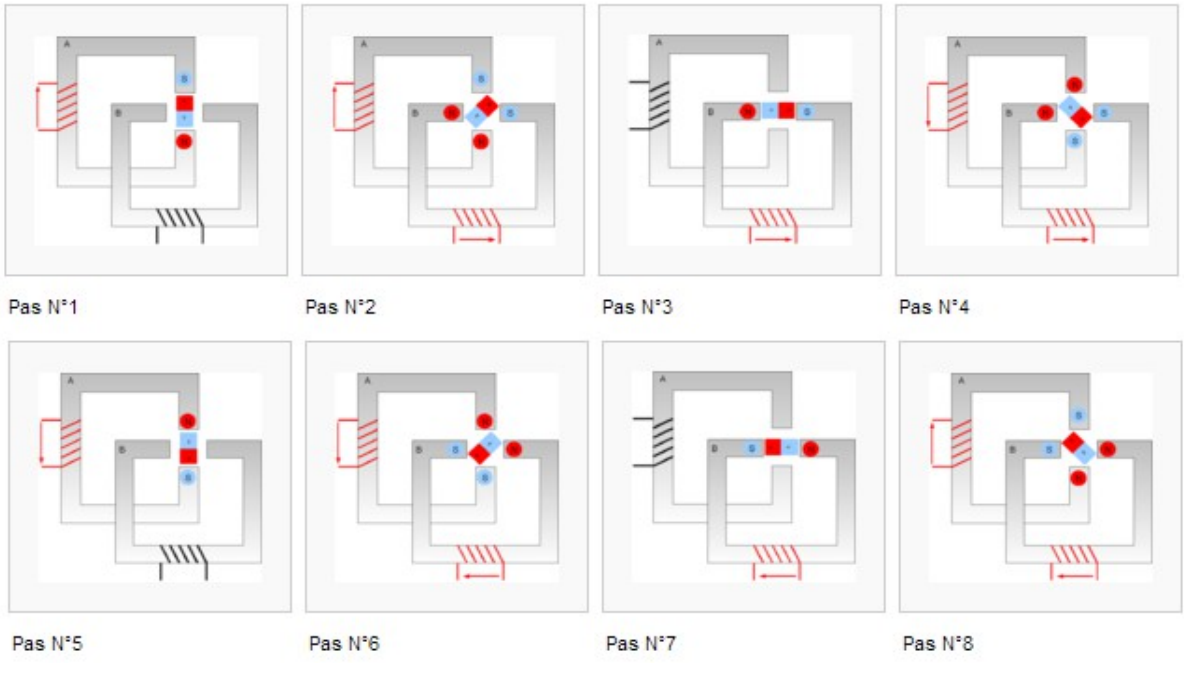
Le mode pas entier suivant est appelé fonctionnement à couple maximal. Dans ce cas, le champ magnétique est toujours plus grand ($\sqrt{2}$ fois plus important) et donc le couple est plus important.



Pas n° 1 | Pas n° 2 | Pas n° 3 | Pas n° 4

En modifiant l'alimentation des bobines, il est possible d'obtenir des déplacements beaucoup plus petits.

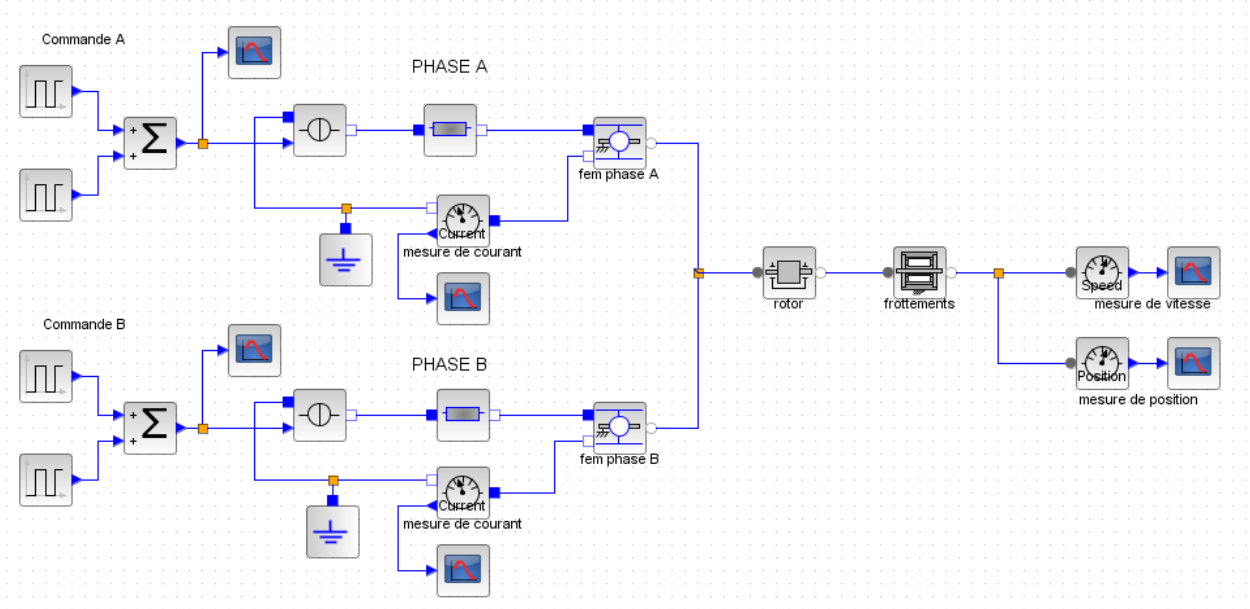
Le mode demi-pas correspond au schéma suivant et permet d'obtenir 2 fois plus de pas.



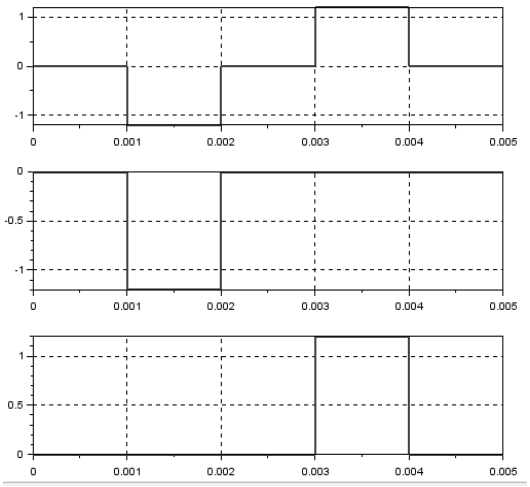
Pas N°1 | Pas N°2 | Pas N°3 | Pas N°4

Pas N°5 | Pas N°6 | Pas N°7 | Pas N°8

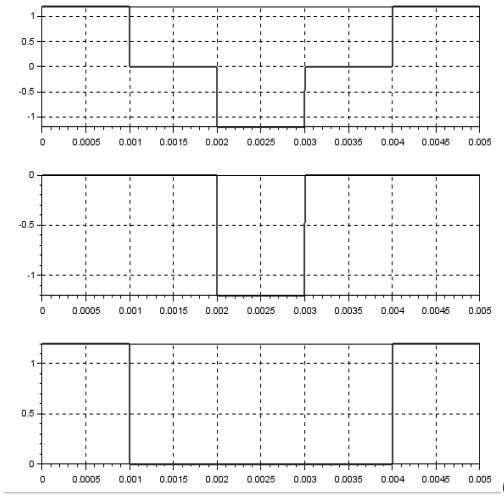
Pour pouvoir simuler le comportement de ce moteur pas à pas d'un point de vue idéal, on enlève l'inductance de chaque phase et on fait l'hypothèse que les bobinages sont modélisés par des sources de courant.



La génération de la commande en mode pas complet standard revient à superposer deux signaux périodiques.

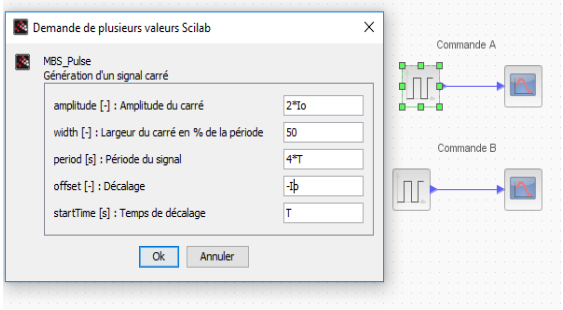


Commande B (de haut en bas) : signal de commande pour le mode pas entier, décomposition par deux signaux périodiques sommés

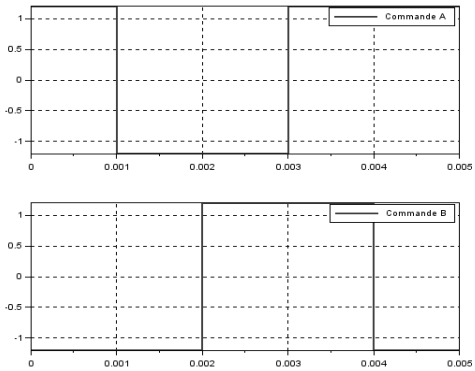


Commande A : même principe que pour la commande de la phase B

De la même manière, il est possible de générer une commande en pas entier à couple maximal.



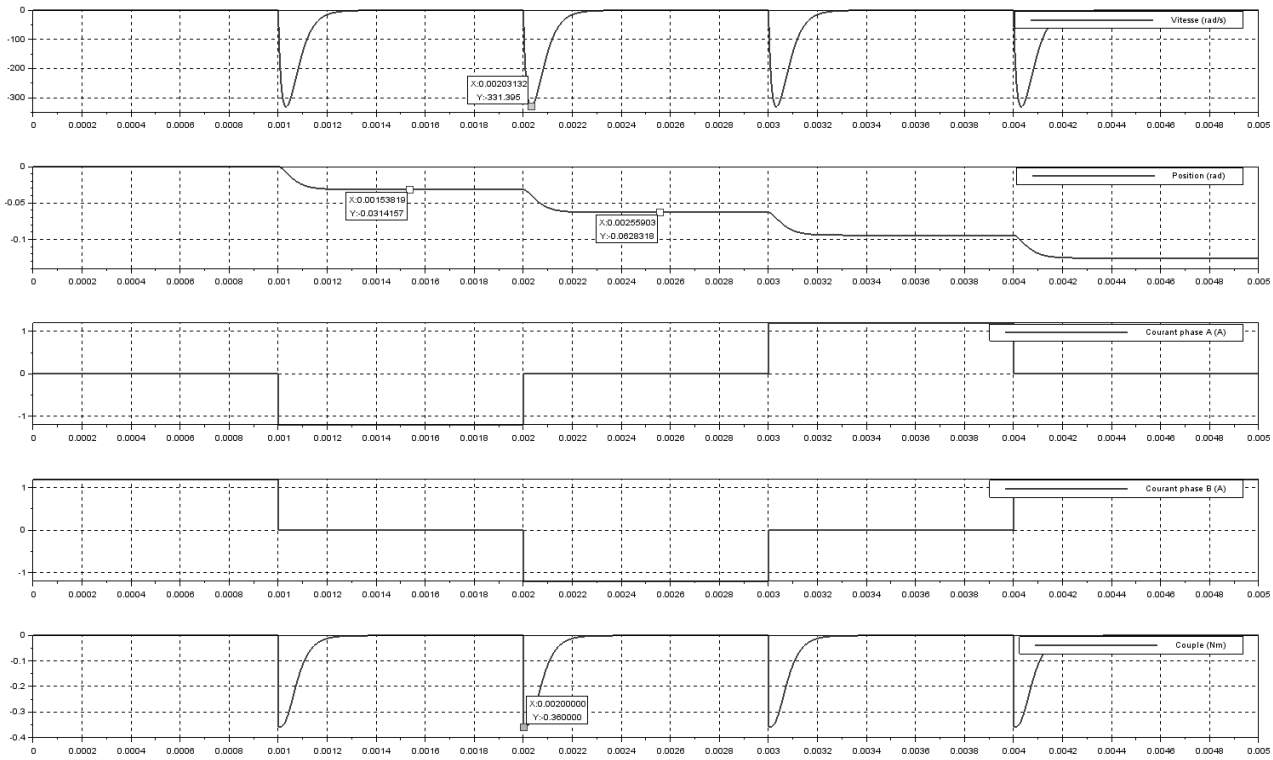
Paramétrage des blocs pour la commande en pas complet à couple maximal



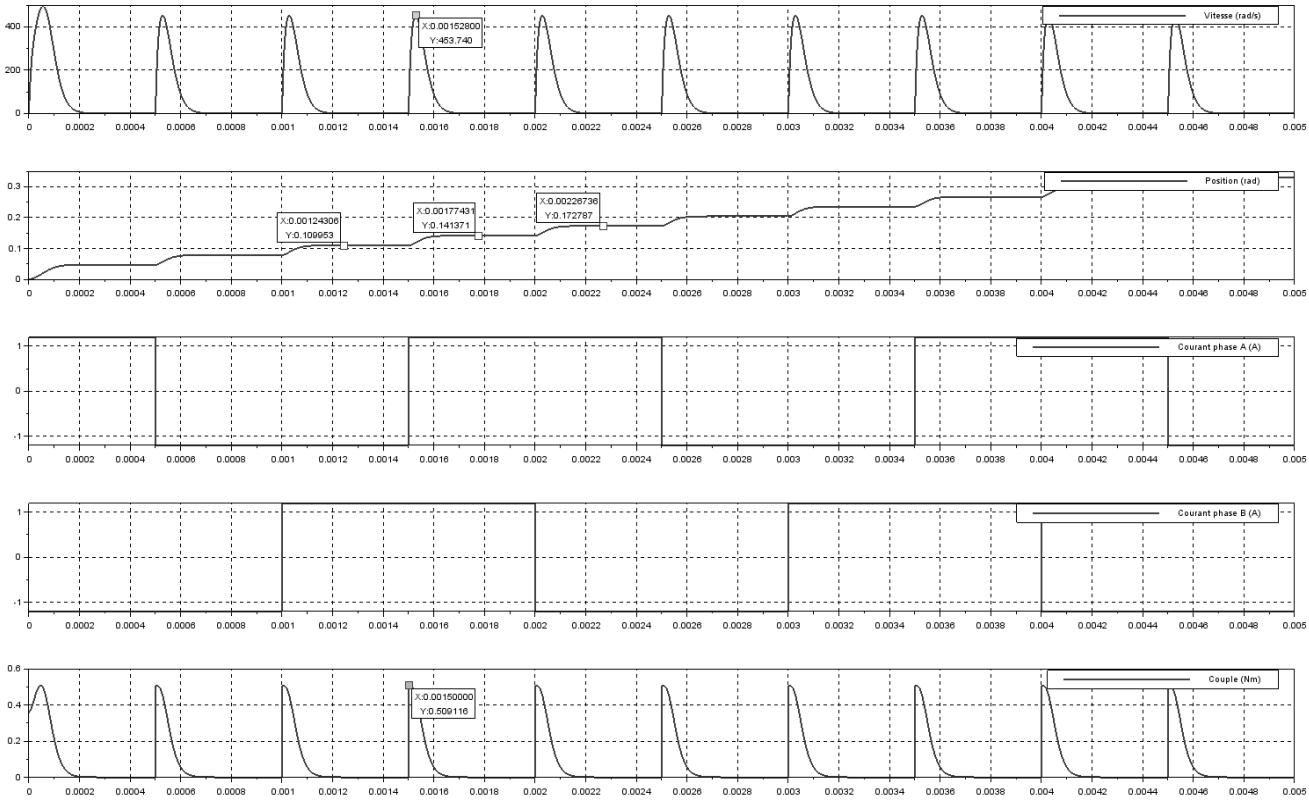
Commandes phase A et B en pas entier à couple maximal

Simulation du moteur pas à pas pour les différents modes

On obtient alors par simulation sur 5 pas les résultats suivants.



Simulation en mode pas entier standard



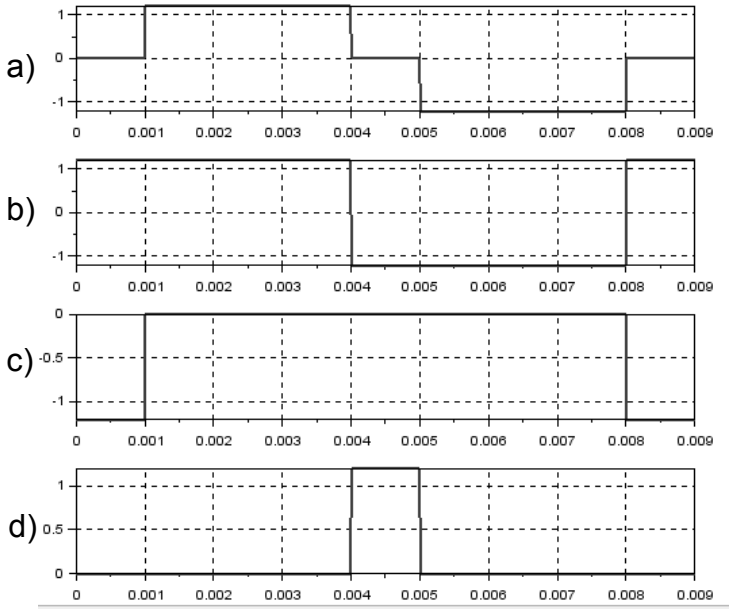
Simulation en mode pas entier à couple maximal

Dans les deux cas, on constate qu'on avance de 1,8° ce qui correspond bien à un fonctionnement en pas entier

pour un moteur possédant 200 pas par tour.

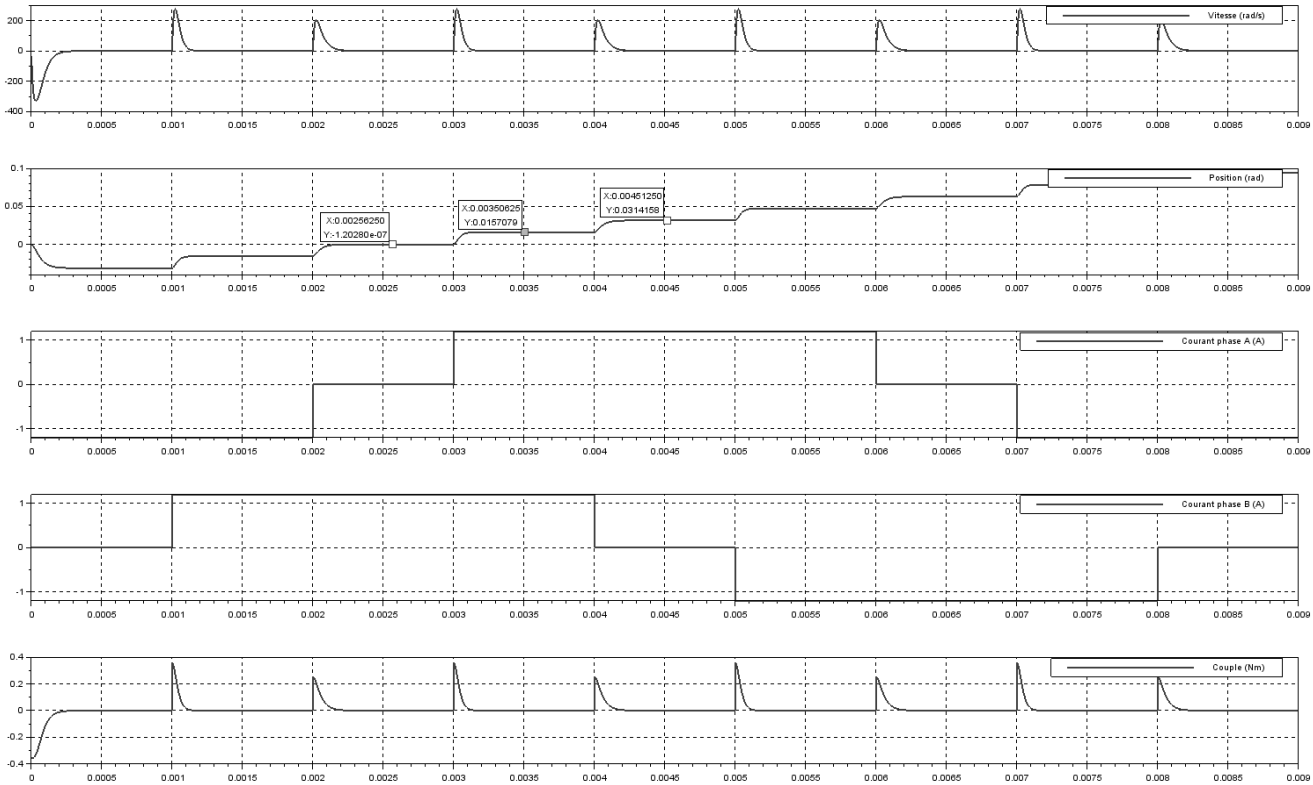
On constate également qu'en changeant le mode de commande, on obtient un couple $\sqrt{2}$ fois plus grand qu'en mode standard. La vitesse est également plus importante. Il faut diviser par deux le temps d'un pas pour avancer de la bonne quantité (1,8°) dans le mode à couple maximal.

Pour un pilotage en demi-pas, on associe plusieurs signaux créneaux de manière particulière pour générer le signal de commande ; ceci permet d'obtenir des déplacements plus petits.



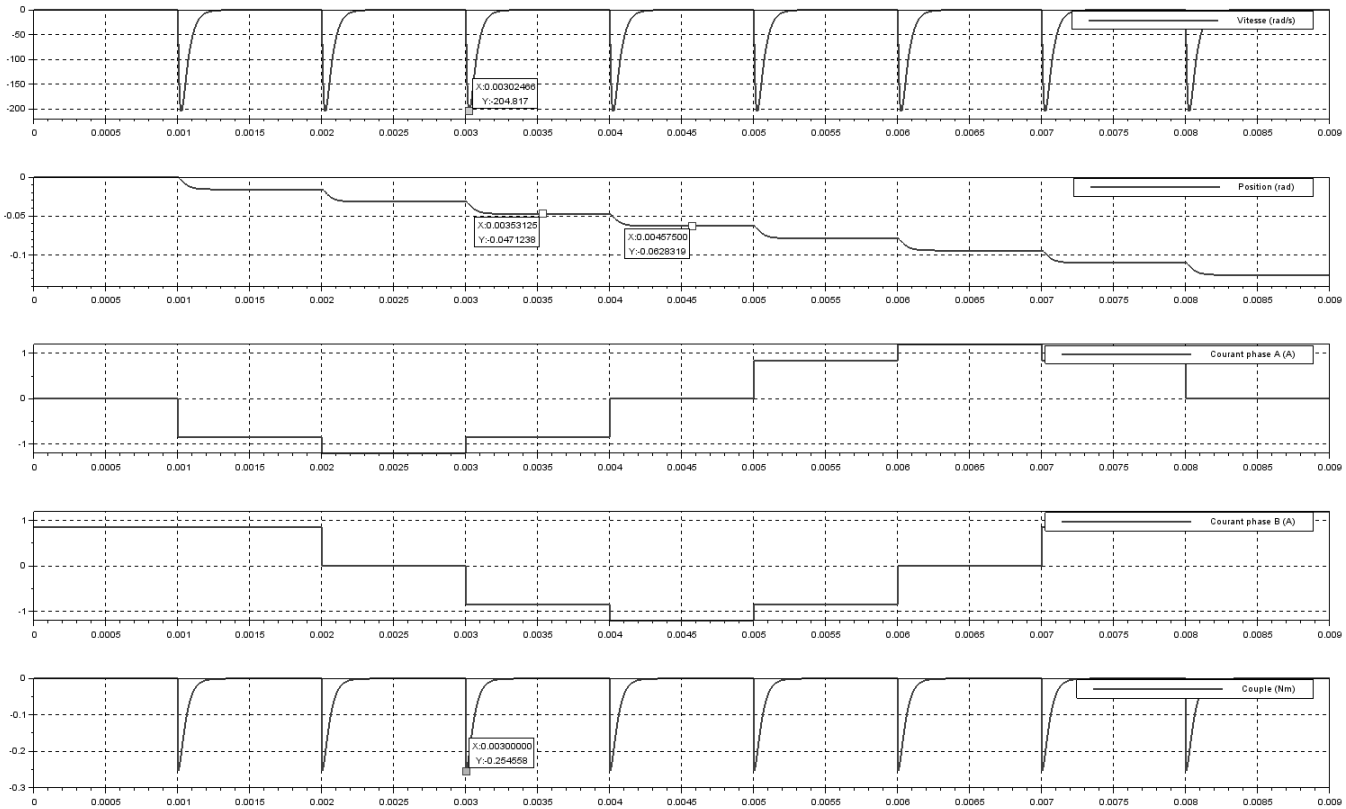
Signal de Commande pour une phase (a) (l'autre commande est déphasée d'un quart de période). Les signaux b), c) et d) sont sommés pour former a)

On obtient alors le comportement suivant :



Fonctionnement en demi-pas avec courant constant par morceaux

On constate donc bien qu'on avance d'un demi-pas à chaque fois. On remarque que l'ondulation de couple ou de vitesse n'est pas régulière. Il faut donc adapter la commande en courant pour fonctionner à ondulation de couple constant. Le fonctionnement demi-pas est moins rapide.



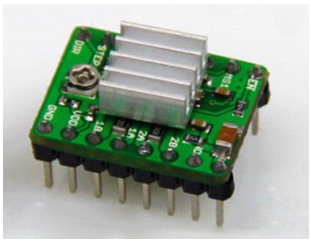
daptation du courant pour fonctionner à ondulation de couple constante

Expérimentalement, on retrouve ce type de comportement pour l'ondulation de courant.

Driver de moteur pas à pas

Compte-tenu des simulations réalisées précédemment, on remarque qu'il est nécessaire de non seulement pouvoir générer une commande particulière selon le mode de fonctionnement retenu mais qu'il est également nécessaire de contrôler le courant.

Fonctionnalités de driver type A4988



Driver A4988

Un driver « microstepping » tel que l'A4988 permet non seulement de fournir la puissance nécessaire à la commande des bobines (hacheur intégré) mais également de simplifier le pilotage et choisir facilement le mode de pas à pas. Il offre ainsi non seulement la possibilité de travailler en pas complet et demi-pas mais également deux autres modes : 1/4 de pas et 1/8^{e} de pas en autorisant des positions intermédiaires dans un pas. Cela est rendu possible en modulant intelligemment la quantité de courant dans les bobines du moteur pas-à-pas. Par exemple, piloter un moteur en mode « 1/4 de pas » permet d'obtenir 800 microsteps (micro-pas) sur un moteur prévu pour 200 pas par révolution et cela en imposant 4 niveau de courants différents pour chacun des micro-pas. Le driver est donc équipé d'un asservissement de courant pour gérer correctement celui-ci dans les bobines. Un potentiomètre permet de définir la limite de courant (ou consigne de courant maximale) qui doit être choisie en fonction de la charge entraînée par le moteur et de la vitesse souhaitée.

La configuration du driver se fait à l'aide des broches MS1, MS2 et MS3.

Le câblage du driver est fourni ci-après. Outre l'alimentation de la carte en 5V, le raccordement des 4 fils du moteur pas à pas et l'alimentation de puissance, on trouve plusieurs broches à raccorder au microcontrôleur :

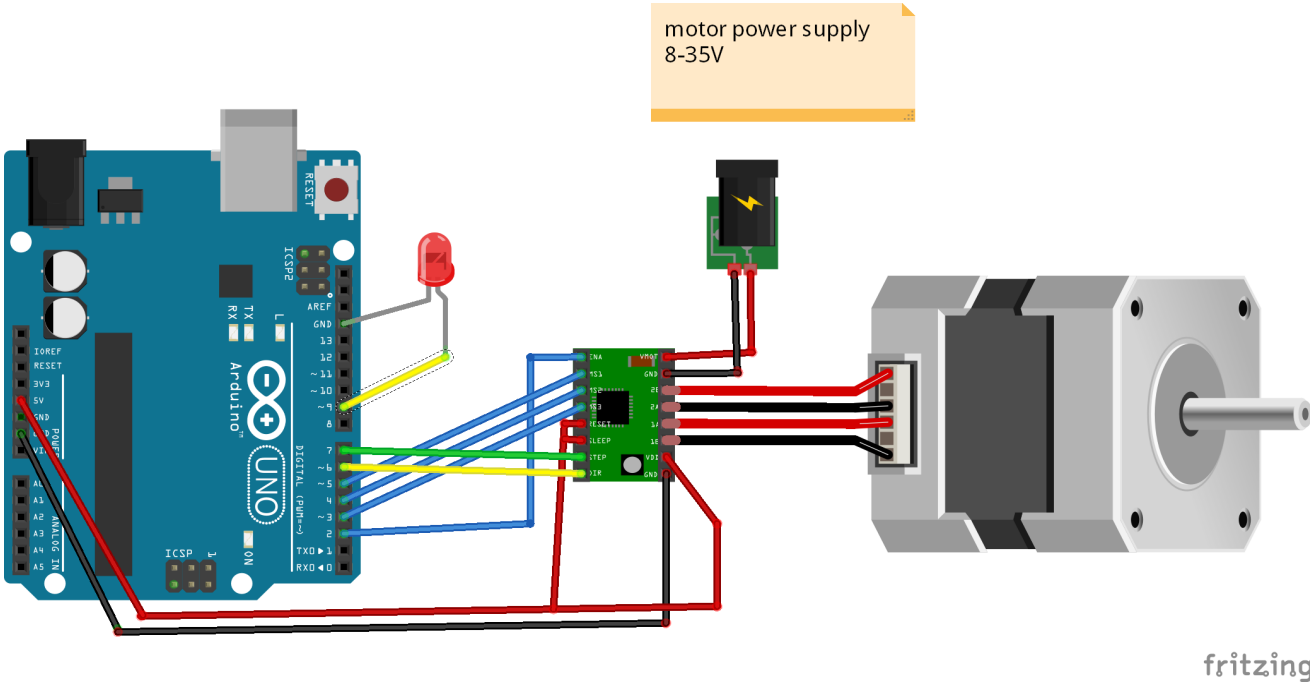
- ENABLE : entrée de validation, active à l'état logique haut permettant de commander l'alimentation de la carte (sinon le moteur reste maintenu dans un pas donné par les bobines et consomme du courant)

- DIR : permet de spécifier le sens de rotation du moteur

- STEP : permet de spécifier le nombre de pas ou micro-pas à effectuer (le déplacement effectif du moteur dépendra du mode choisi pas entier, demi-pas etc)

- MS1 à MS3 permettent de choisir le mode de fonctionnement du moteur. Ces entrées disposent de résistance de tirage à l'état bas (pull-down) internes, ce qui signifie que si l'on ne place pas le potentiel de ces broches au niveau logique haut = High = VDD (avec VDD la tension choisie pour la logique de commande) alors elles seront automatiquement ramenée au niveau logique bas = Low. Le tableau suivant permet de comprendre le mode de pas choisi en fonction des broches MS1 et MS2.

MS1	MS2	MS3	Résolution Microstepping
Low	Low	Low	Pas complet (full step)
High	Low	Low	1/2 pas
Low	High	Low	1/4 de pas
High	High	Low	1/8ème de pas
High	High	High	1/16ème de pas

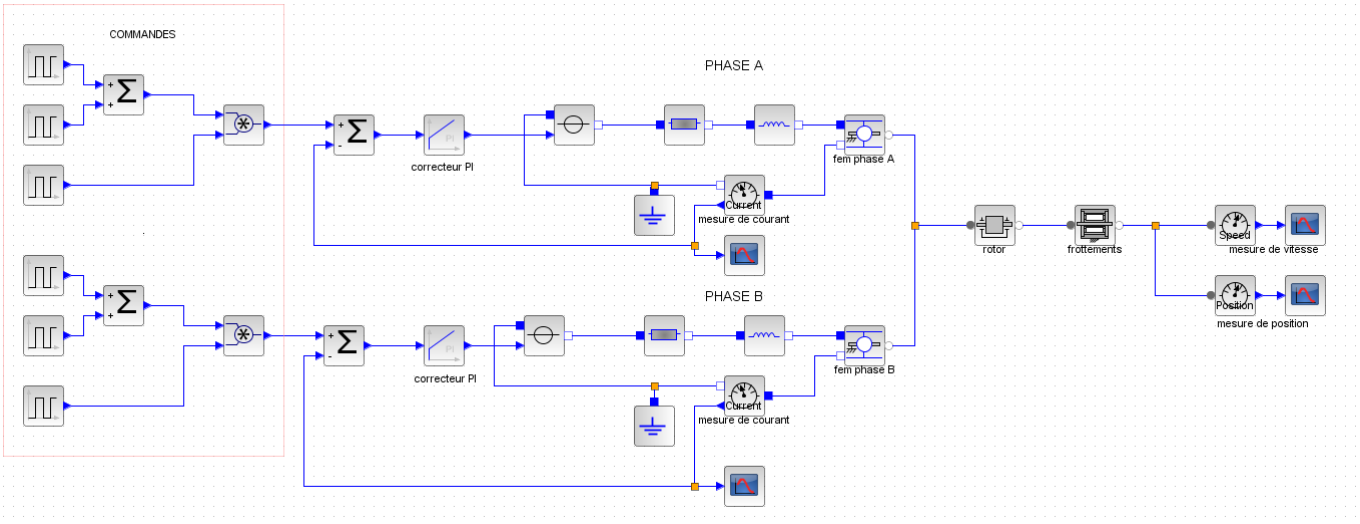


Câblage du driver de moteur pas à pas

Contrôle de courant

Dans les simulations réalisées précédemment, la commande a été réalisée avec des sources de courant. Si on utilise directement la commande en tension, il est nécessaire de mettre en place une boucle de courant avec correcteur PI.

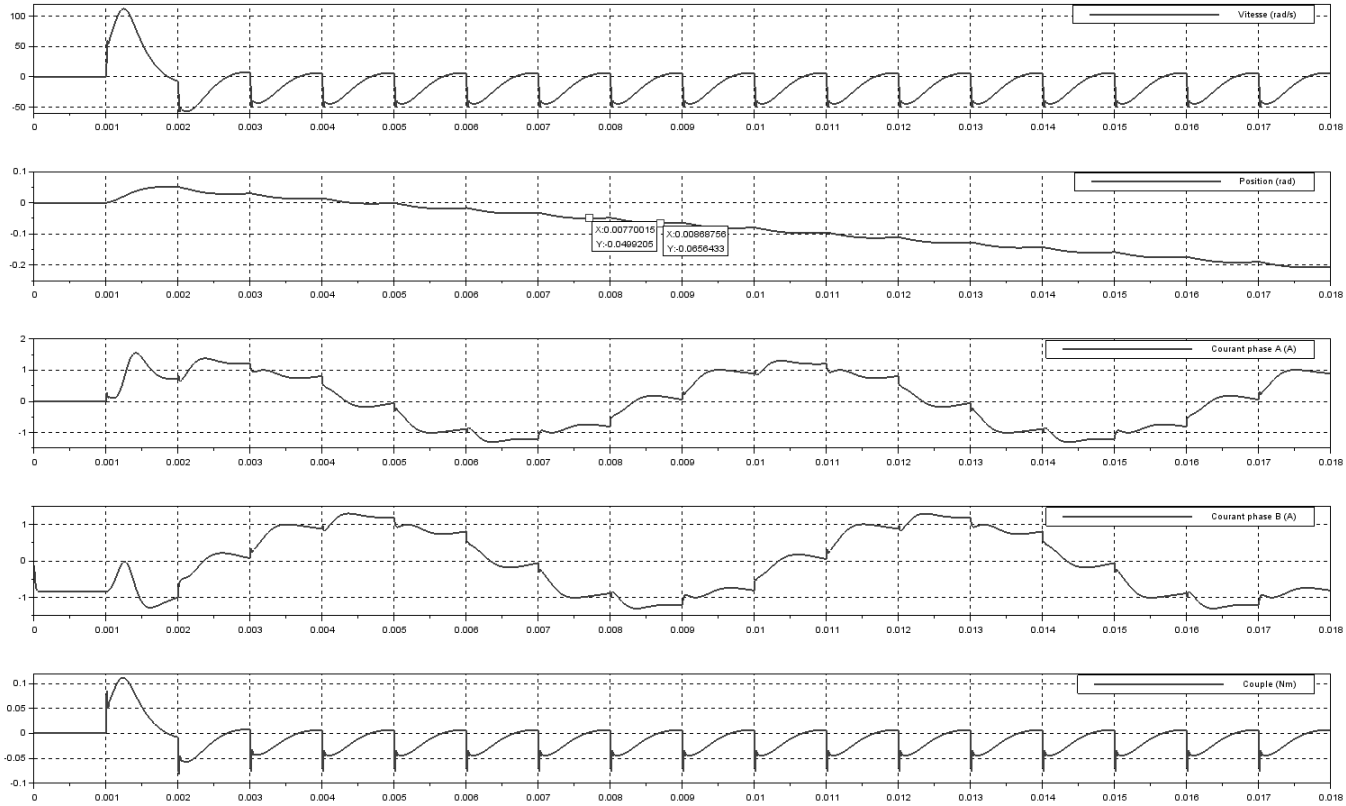
Ce correcteur PI est simplement réglé de manière à compenser la constante de temps de temps électrique L/R.



Remarque : il faut ajouter une saturation en tension (12 V) après le correcteur PI, mais globalement la tension ne dépasse pas beaucoup 12 V pendant les simulations pour le réglage fait du correcteur.

La simulation en demi-pas passe difficilement pour des questions d'erreur de convergence des schémas d'intégration numérique utilisés.

On donne ci-dessous la réponse en mode demi-pas pour 2 périodes. On observe que le fonctionnement est correct car le moteur avance bien à chaque fois d'un demi-pas (0,9° soit $\frac{\pi}{4}$ rad). L'ondulation de couple et de vitesse est moins propre qu'avec une source de courant.

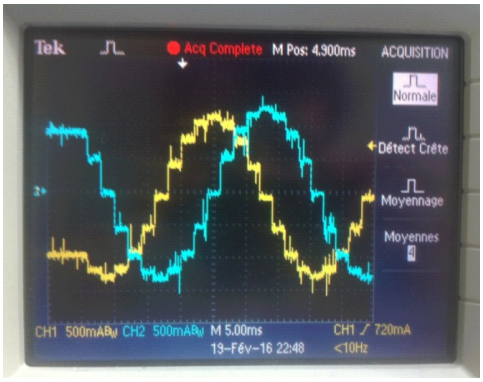


R

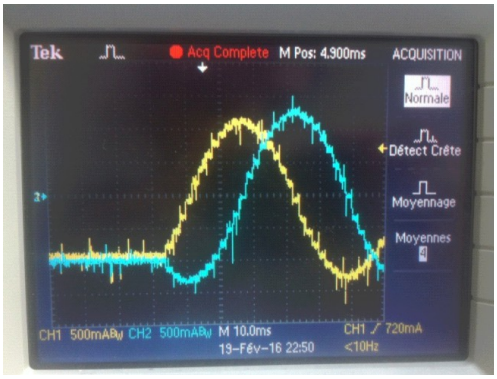
emarque : en réalité, le contrôle de tension est fait par un hacheur. La tension mesurée n'est donc pas continue mais sous forme de créneau.

Etant donné que le courant est contrôlé grâce à l'asservissement de courant, il est possible de changer la consigne de courant et d'obtenir ainsi plusieurs modes de pas beaucoup plus précis.

Les relevés à l'oscilloscope suivant montrent le courant pour ces différents modes de micro-pas. On constate qu'on tend pratiquement vers un courant sinusoïdal (ce qui se rapproche du pilotage des moteurs sans balais).



Mesures de courant dans les deux phases en mode quart de pas



Mesures de courant dans les deux phases en mode huitième de pas

Utilisation standard du moteur pas à pas

Pilotage par carte Arduino

La carte Arduino couplée à deux cartes de puissance type hacheur (une par phase) permet de piloter le moteur pas à pas. Une librairie Stepper est disponible ; elle permet de gérer la logique de commande du moteur pas à pas (on renseigne uniquement les 4 pins utilisés pour connecter les 2 hacheurs et on indique le nombre de pas et le sens souhaité). Cependant, compte-tenu des explications et simulations précédentes, on comprend que le courant ne sera pas bien régulier étant donné qu'il n'y a pas d'asservissement de courant et que la carte Arduino basique ne permet pas de faire cet asservissement facilement.

L'avantage du driver est de simplifier la commande en n'ayant à définir seulement la durée des micro-pas et le nombre de pas. On sélectionne bien évidemment également le mode de pas choisi à l'aide des pins MS1, MS2 et MS3. En réalité, on ne donne pas directement la durée des micro-pas et le nombre de pas mais on envoie un signal de type créneau dont la période est utilisée pour les micro-pas, ceci pendant une durée correspondant au nombre de pas souhaité.

Le programme ci-contre permet de réaliser un tour du moteur pas à pas avec un temps de pas égal à 1.6 ms.

```
// Commande d'un moteur pas-à-pas à l'aide d'un pilote A4988 avec Arduino.

#define STEP_PIN 7    // Signal de PAS (avancement)
#define DIR_PIN 6     // Sens de rotation
#define MS1_PIN 3     // Mode MS1
#define MS2_PIN 4     // Mode MS2
#define MS3_PIN 5     // Mode MS3
#define ENABLE_PIN 2    // ENABLE

void setup(){
  Serial.begin(115200);
  Serial.println("Test A4988");
  pinMode( STEP_PIN, OUTPUT );
  pinMode( DIR_PIN, OUTPUT );
  pinMode( MS1_PIN, OUTPUT );
  pinMode( MS2_PIN, OUTPUT );
  pinMode( MS3_PIN, OUTPUT );
  pinMode( ENABLE_PIN, OUTPUT );
  digitalWrite( DIR_PIN, LOW);
  digitalWrite( MS1_PIN, LOW);
  digitalWrite( MS2_PIN, LOW);
  digitalWrite( MS3_PIN, LOW);
  digitalWrite( ENABLE_PIN, LOW);
}

int stopper=0;
void loop(){
  // Avance de 200 pas
  if (stopper==0){
    for (int i=0; i<200; i++){
      Serial.println( i );
      digitalWrite( STEP_PIN, HIGH );
      delayMicroseconds( 800 ); // Retard de 800 µs
      digitalWrite( STEP_PIN, LOW );
      delayMicroseconds( 800 );
    }
    stopper=1;
    digitalWrite( ENABLE_PIN, HIGH );
  }
}
```

Pilotage en Python

La librairie py2duino développée par l'équipe Démosciences permet de communiquer en python avec une carte Arduino. Il suffit de téléverser un programme dans l'Arduino qui fera l'interface entre python et les entrées-sorties de la carte Arduino.

Cette librairie a été modifiée pour intégrer le pilotage des moteurs pas à pas.

Remarque : il est important de noter que l'on ne peut pas piloter directement le moteur via des commandes Python en traduisant celles définies précédemment sous Arduino. En effet, le temps nécessaire lors de la communication « Ping-Pong » pour que la carte Arduino transmette et reçoive des informations issues de l'ordinateur est trop long vis à vis du retard imposé entre les instructions. La bibliothèque py2duino développée permet de palier ce problème.

On définit au préalable les sorties de la carte Arduino reliées au driver de moteur pas à pas, puis on envoie une commande en spécifiant le nombre de pas souhaité et le temps du micro-pas. On peut également changer le mode de pas en utilisant des sorties digitales classiques.

A l'aide de ce type de programmation, il est possible de définir et piloter plusieurs moteurs pas à pas (jusqu'à 4) en envoyant les commandes « move » les unes à la suite.

```
COM_ARDUINO = 8
ENABLE_PIN = 2
MS1_PIN = 3
MS2_PIN = 4
STEP_PIN = 7
DIR_PIN = 6

from py2duino import *
import time

#déclaration
MaCarte=Arduino(COM_ARDUINO)
time.sleep(2) #on attend pour que la communication s'établisse bien

MonStepper=Stepper(MaCarte,1,STEP_PIN,DIR_PIN)
MS1=DigitalOutput(MaCarte,MS1_PIN)
MS2=DigitalOutput(MaCarte,MS2_PIN)
ENABLE=DigitalOutput(MaCarte,ENABLE_PIN)

#déplacement d'un tour dans un sens
MonStepper.move(200,0.8)
```

Conclusion

Nous avons montré à travers cet article que le principe de fonctionnement par driver d'un moteur pas à pas peut s'expliquer à l'aide d'une modélisation multiphysique. Si la compréhension de ce modèle est relativement simple à l'aide d'une telle représentation, l'obtention de résultats de calculs est beaucoup plus hasardeuse que pour un modèle dit causal (avec fonctions de transfert). En effet les problèmes numériques dépendant des paramètres physiques et la causalité à respecter sont autant de sources de non convergence possible.

Un bloc Stepper ainsi qu'un bloc Driver pourraient être intégrés au module SIMM disponible sous Scilab.

L'utilisation de python pour piloter des moteurs pas à pas permet notamment de gérer facilement la cinématique d'imprimante cartésiennes mais aussi de type Delta par exemple. Il est également envisageable de traiter les problèmes de prise d'origine de ces systèmes à événements discrets.