Sciences et Techniques

FROM CHAOS TO CONTROL

Août
05
2011

Implémenter un PID sans faire de calculs !

Robotique — Commenter

Sur beaucoup de site sur le web, lorsque l'on trouve un article parlant de la régulation PID, on se heurte souvent à des fonctions de transfert et à des équations dans le domaine de Laplace. Alors implémenter un PID est-il impossible sans avoir une base solide en mathématiques ? Heureusement non, loin de là. Outre l'approche mathématiques, le PID peut très bien s'expliquer de façon intuitive "avec les mains". Le but de cet article est justement d'essayer d'expliquer comment marche une régulation PID sans entrer dans les calculs ni sans utiliser les fonctions de transfert du système et autres équations quelque peu cabalistiques pour les non-initiés.

Asservir la vitesse d'une voiture "à la main"

Imaginez que vous conduisez votre voiture sur l'autoroute et que vous souhaitez stabiliser votre vitesse à 130 km/h pil-poil ! Dans la pratique, vous y arriverez sans trop de problèmes. Le but de cet exemple est d'essayer de comprendre comment votre esprit arrive à réguler votre vitesse intuitivement.

Règle 1 :

Tout d'abord, la chose la plus intuitive que vous faites lorsque vous voulez rouler à une certaine vitesse sur l'autoroute, c'est de vous dire : *" plus je roule lentement par rapport à la vitesse voulu et plus j'appuie sur la pédale d'accélération "*. La pression sur l'accélérateur est donc proportionnelle à l'erreur que vous commettez, c'est-à-dire, proportionnelle à la différence entre la vitesse voulue et la vitesse réelle.

Avec cette méthode, pas de problème. Vous êtes à 50 km/h, vous appuyez sur le champignon ! Votre vitesse augmente et se rapproche de la vitesse limite. Vous exercez une pression de moins en moins forte. Une fois que la vitesse atteint 130 km/h, vous avez réussi ! L'erreur est nulle, vous lâchez donc l'accélérateur. Manque de bol, comme l'accélérateur est complètement lâche, la voiture commence à ralentir. Vous recommencez donc à appuyer un peu sur l'accélérateur, puis de plus en plus au fur et à mesure que la voiture ralenti. Au final, vous arrivez à stabiliser votre vitesse à une vitesse inférieure à celle que vous avez choisi, disons 120km/h.

Règle 2 :

C'est alors que vous vous dites : *" Zut ! Je n'arrive pas à atteindre la vitesse voulue, il faut que je rajoute une règle supplémentaire à mon raisonnement ! "*. Du coup, vous décidez que si votre vitesse reste longtemps sous l'objectif, vous accélérez de plus en plus fort. Vous décidez donc qu'en plus d'accélérer proportionnellement à l'erreur commise, vous allez aussi mémoriser cette erreur au cours du temps. Plus l'erreur globale est importante et plus vous accélérez.

Ainsi, lorsque vous stabilisez votre vitesse à 120km/h, l'erreur globale augmente et vous vous mettez à appuyer de plus en plus fort sur l'accélérateur jusqu'à atteindre 130km/h... et le dépasser ! En effet, arrivé à 130km/h, l'erreur globale est positive, donc vous continuez à appuyer sur l'accélérateur. Arrivé au-delà de 130km/h, l'erreur est négative et fait donc diminuer d'erreur globale. Vous levez donc le pied de l'accélérateur de plus en plus fortement jusqu'à retourner à 130 km/h. Arrivé à 130km/h, rebelote, l'erreur est passé en négatif et vous continuez à décélérer... ainsi de suite jusqu'à finalement arriver à vous stabiliser à 130km/h après de multiples oscillations.

Règle 3 :

Arrivé à 130km/h, vous vous dites : *" ça y est, j'y suis ! Mais je n'ai pas été très efficace... Ne faudrait-il pas rajouter une troisième règle afin d'être plus performant ? "*. C'est alors que vous décidez d'anticiper votre vitesse. Plus votre vitesse se rapproche de la vitesse optimale, moins vous accélérez et moins elle se rapproche de la vitesse optimale, plus vous accélérez !

Ainsi, si vous vous rapprochez rapidement des 130 km/h, vous vous empressez de lever le pied afin de ne pas dépasser les 130 trop brutalement. Ainsi, vous réduisez les oscillations et vous vous stabilisez rapidement à la vitesse souhaitez !

Les mots clefs

Filtre numérique Centrale inertielle Domaines discrets Algorithmique Asservissement Electronique Logique Montage Estimation Java Logique floue Planification de trajectoire Prédiction Filtre de Kalman Mathématiques Kicad Automatique État de l'art Fusion de données Robotique Transformée de Fourier Traitement du signal PCB Algèbre Transformée en Z Arduino Programmation Projet Jeux Intelligence artificielle Comparatif Capteur

Qui est en ligne

4 visiteur(s) en ligne actuellement
4 visiteur(s), 0 robots, 0 membre(s)
Map of Visitors

Voilà, vous avez intuitivement fait une régulation de type PID ! 🙂

Implémentation d'un PID sur un robot

A partir de cet exemple, on voit qu'un asservissement PID n'a rien de très compliqué ! Tout ce que l'on a à faire, c'est de mémoriser l'erreur, la somme des erreurs et la différence de l'erreur courante avec l'erreur précédente.

Le régulateur proportionnel (P : première règle)

La commande de ce régulateur est proportionnelle à l'erreur.

```
commande = Kp * erreur
```

Kp est le coefficient de proportionnalité de l'erreur à régler de façon manuelle.

Le régulateur proportionnel intégral (PI : première et seconde règle)

La commande de ce régulateur est proportionnelle à l'erreur, mais aussi proportionnelle à l'intégrale de l'erreur. On rajoute donc à la commande généré par le régulateur proportionnel, la somme des erreurs commises au cours du temps.

```
commande =  Kp * erreur + Ki * somme_erreurs
```

Ki est le coefficient de proportionnalité de la somme des erreurs. Il faut aussi le régler de façon manuelle.

Le régulateur proportionnel dérivé (PD : première et troisième règle)

La commande de ce régulateur est proportionnelle à l'erreur, mais aussi proportionnelle à la dérivée de l'erreur. La dérivée de l'erreur correspond à la variation de l'erreur d'un échantillon à l'autre et se calcule simplement en faisant la différence entre l'erreur courante et l'erreur précédente (c'est une approximation linéaire et locale de la dérivée).

```
commande = Kp * erreur + Kd * (erreur - erreur_précédente)
```

Kd est le coefficient de proportionnalité de la variation de l'erreur. Il faut régler ce coefficient manuellement.

Le régulateur proportionnel intégrale dérivé (PID : première, seconde et troisième rèlge)

Ici, la commande est à la fois proportionnelle à l'erreur, proportionnelle à la somme des erreurs et proportionnelle à la variation de l'erreur.

```
commande = Kp * erreur + Ki * somme_erreurs + Kd * (erreur - erreur_précédente)
```

Vous devez donc faire une mesure sur votre système pour pouvoir calculer l'erreur et ainsi appliquer le PID. Cette mesure est à faire régulièrement à une certaine fréquence d'échantillonnage.

```
Tous les x millisecondes, faire :
    erreur = consigne - mesure;
    somme_erreurs += erreur;
    variation_erreur = erreur - erreur_précédente;
    commande = Kp * erreur + Ki * somme_erreurs + Kd * variation_erreur;
    erreur_précédente = erreur
```

Comment régler les coefficients d'un PID ?

Le réglage des coefficients Kp, Ki et Kd d'un PID peut se faire "à la main" par essais/erreurs. Tout d'abord, sachez qu'il ne sert à rien de vouloir régler les trois coefficients en même temps ! Il y a trop combinaisons possibles et trouver un triplet performant relèverait de l'exploit. Il vaut mieux y aller par étape.

- Tout d'abord, il faut mettre en place un simple régulateur proportionnel (les coefficients Ki et Kd sont donc nuls). Par essais/erreurs, il faut régler le coefficient Kp afin d'améliorer le temps de réponse du système. C'est-à-dire qu'il faut trouver un Kp qui permette au système de se rapprocher très vite de la consigne tout

en faisant attention de garder la stabilité du système : il ne faut pas que le système réponde très vite tout en oscillant beaucoup !

- Une fois ce coefficient réglé, on peut passer au coefficient Ki. Celui-là va permettre d'annuler l'erreur finale du système afin que celui-ci respecte exactement la consigne. Il faut donc régler Ki pour avoir une réponse exacte en peu de temps tout en essayant de minimiser les oscillations apportées par l'intégrateur !
- Enfin, on peut passer au dernier coefficient Kd qui permet de rendre le système plus stable. Son réglage permet donc de diminuer les oscillations.

En général, pour régler ces coefficients, on donne au système une consigne fixe (exemple : pour un moteur : tourne à 3 tours par seconde) et on observe la réponse du système (exemple : l'évolution du nombre de tours par seconde du moteur au cours du temps). le graph résultant possède donc cette forme :

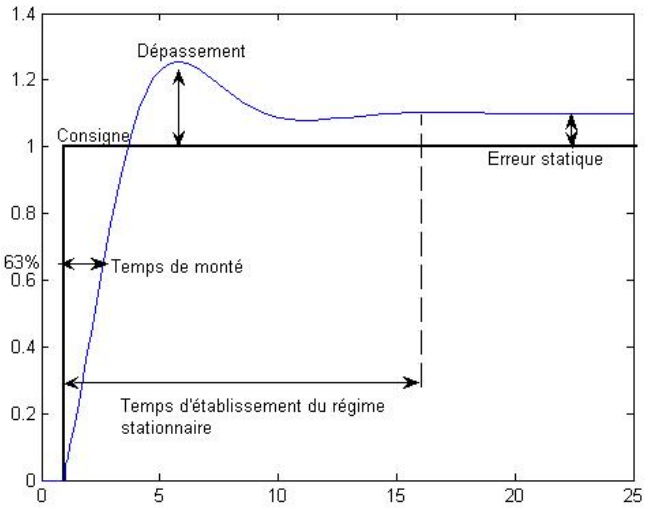

Image tirée de Wikipédia

Le PID parfait n'existe pas, tout est une question de compromis. Certaines applications autoriseront un dépassement afin d'améliorer le temps de stabilisation, alors que d'autres ne l'autoriseront pas (exemple, contrôler un stylo pour écrire sur une feuille. S'il y a dépassement dans le PID, le stylo traversera la feuille). Tout dépend donc du cahier des charges. Chacun des coefficients à un rôle à jouer sur la réponse à une consigne :

- L'erreur statique, c'est l'erreur finale une fois que le système est stabilité. Cette erreur doit être nulle. Pour diminuer l'erreur statique, il faut augmenter Kp et Ki.
- Le dépassement, c'est le rapport entre le premier pic et la consigne. Ce dépassement diminue si Kp ou Ki diminuent ou si Kd augmente.
- Le temps de montée correspond au temps qu'il faut pour arriver ou dépasser à la consigne. Le temps de montée diminue si Kp ou Ki augmentent ou si Kd diminue.
- Le temps de stabilisation, c'est le temps qu'il faut pour que le signal commette une erreur inférieure à 5% de la consigne. Ce temps de stabilisation diminue quand Kp et Ki augmentent.

Pour vous donner une petite idée de la valeur des coefficients lors de vos premiers essais, vous pouvez regarder du côté de la méthode Ziegler–Nichols. Cette méthode permet de déterminer Kp, Ki et Kd en fonction de votre cahier des charges.

Attention, les coefficients Ki et Kd dépendent de la fréquence d'échantillonnage du système ! En effet, l'intégrateur fera la somme des erreurs au cours du temps ! Si on échantillonne deux fois plus vite, on sommera deux fois plus d'échantillons. Du coup, le coefficient Ki devra être divisé par 2. A l'inverse, pour le dérivateur, si on double la fréquence d'échantillonnage, il faudra doubler le coefficient Kd afin de garder les mêmes performances du PID. Plus la fréquence d'échantillonnage est élevé et plus le PID sera performant. (En effet, plus on échantillonne souvent et plus l'intégration et la dérivée seront précises).

Voilà, vous savez maintenant comment marche un PID et comment trouver les différents coefficients ! Il ne vous reste plus qu'à tester ce régulateur sur vos différents robots 😉
Sachez aussi qu'il est possible de déterminer automatiquement les coefficients d'un PID grâce à des algorithmes génétiques par exemple. Cela vous évitera ainsi de passer trop de temps à les régler manuellement !

Pour voir une implémentation réelle d'un PID numérique, regardez cette article.

Publié par Ferdinand Piette à 17:25 Taggué avec : Asservissement, Automatique, Robotique

57 commentaires à “Implémenter un PID sans faire de calculs !”

1. **Leon** dit :
10 décembre 2011 à 11:28

Bonjour Monsieur.

Je voudrais vous remercier pour l'aide et les explications apportées sur la compréhension du PID numérique.
Je ne suis plus tout jeune, bientôt 65 ans et je m'intéresse à l'électronique depuis l'âge de 10 ans, j'ai donc commencé avec les tubes.
Je ne suis qu'un autodidacte et donc limité par rapport à vrai électronicien et encore bien plus par rapport à votre bagage.
Cependant, je fais de la régulation de vitesse mais en analogique, les valeurs des composants notamment pour le "PI" ne sont pas calculés mais ajusté par l'observation des résultats à l'oscilloscope.
Je voulais passer à la régulation par microcontrôleur (je suis sur AVR d'Atmel, Atmega 8535, Atmega 32 et pour un projet en cours d'éolienne, un Atmega 1284) et j'étais bloqué par l'application de "I" et éventuellement de "D".
Grâce à vous, je vais pouvoir m'aventurer en régulation par uC.

Je reviendrai sur votre site pour apprendre d'avantage.
Encore tous mes remerciements.

Cordialement, Léon.

Répondre

2. **Simon (Esprit)** dit :
12 juin 2012 à 07:45

Je sors d'un baccalauréat en "informatique et systèmes, finalité Automatique" où un de nos cours principal est la régulation. Je trouve que le cours devrait commencer par ton article. C'est clair et ça aide vraiment à bien visualiser ce qu'il se passe. Merci pour ce partage !

Répondre

3. **florent (swolf)** dit :
26 juin 2012 à 22:27

Salut! très bon article de vulgarisation, c'est simple et intéressant, merci beaucoup!
Par contre l'image tirée de Wikipédia ne s'affiche pas chez moi...

Répondre

4. **Gérard** dit :
11 décembre 2012 à 14:23

Bonjour,
Chapeau pour cette vulgarisation du régulateur PID!
la comparaison avec l'accélérateur la rend beaucoup plus clair et compréhensible!
cela fait quelques temps que je cherche à quoi cela correspond exactement, et me voila servi!
bonne continuation!

Répondre

troll dit :
30 mars 2014 à 15:25

...me voilà (as)servi! 😉

Répondre

5. **Jean** dit :
4 février 2013 à 11:50

Très bon article pour introduire le sujet.

Plus qu'a tester des Kx ...

Pour information, tu as quelques fautes :
Dans ton exemple d'implémentations : commande = Kp * erreur + Ki * somme_erreurs + Ki * variation_erreur; c'est Kd.
Et dans ta remarque sur la fréquence d’échantillonnage, "Attention, les coefficients Ki et Kp", c'est aussi Kd au lieu de Kp.

Concernant la fréquence d’échantillonnage, si la valeur régulée varie peu , en augmentant la fréquence, la dérivée est quasiment lisse. Du coup soit PI au lieu de PID, soit réduire l'importance de la dérivée, soit réduire la fréquence d’échantillonnage.
Y a-t-il une autre solution ?

Répondre

Ferdinand Piette dit :
5 février 2013 à 14:47

Merci pour ces corrections !

Si on augmente la fréquence, la valeur de la dérivé ne change pas. Ce qui change (et qui tend vers 0 pour une fréquence tendant vers l'infini), c'est la différence entre l'échantillon courant et le précédent.

La dérivé est calculé en divisant cette différence par le pas d'échantillonnage (qui est implicitement inclut dans Kd).
Donc si on augmente la fréquence d'échantillonnage, il faut augmenter proportionnellement le coefficient Kd ! On ne réduit donc pas l'importance de la dérivé. Celle-ci aura toujours autant d'importance.

Répondre

Jean dit :
6 février 2013 à 15:55

Dans un domaine continu oui, mais pas dans un domaine discret.

Je m'explique : j'ai un capteur de température, il a une précision de 0,1 ° mettons.
si j'augmente ma fréquence, au bout d'un moment, mes valeurs lues seront "en escalier" alors que globalement j'ai une augmentation ou une diminution. De ce fait je vais avoir un terme dérivatif nul la plupart du temps et une fois de temps en temps Kd * 0,1.

Vous voyez ce que je veux dire ?

Répondre

Ferdinand Piette dit :
19 février 2013 à 17:56

Oui, je vois ce que vous voulez dire.

Du fait de la quantification du système, on ne peut pas mesurer une faible variation. Du coup, lorsque la variation sera suffisante dans le monde continue, on verra un "saut" dans le monde discret.

Je n'avais pas songé à ce problème et ne vois pas de solutions. Il faudrait y réfléchir plus en détail.

Répondre

Ulrich NSENGUET dit :
20 novembre 2013 à 22:28

Dans ce cas, il faut penser à l'utilisation d'un observateur (ex. : celui de Luenberger) pour reconstruire l'état du système (cf. http://fr.wikipedia.org/wiki/Observateur_d%27%C3%A9tat) pour espérer estimer la dérivée de cette erreur...

6. **Gilles** dit :
5 avril 2013 à 15:26

Merci Ferdinand pour cet exposé simple et clair !
En espérant que je serai à la hauteur pour en tirer parti...

Répondre

7. **Naim** dit :
16 avril 2013 à 21:10

Bonjour et merci pour ces instructions.En fait j'ai un petit soucis pour limiter un dépassement je souhaiterais que lorsque j approche de ma consigne je commence à fermer ma vanne vapeur car en été j ai une faible inertie sur ma boucle fermée

Répondre

Ferdinand Piette dit :
16 avril 2013 à 22:03

Bonjour,

La façon de régler le PID est le même, quelque soit le système.
S'il y a trop de dépassement, il faut réduire Ki.
Voir, augmenter Kd afin d'anticiper les choses.

Bonne soirée

Répondre

8. **Dehlinher** dit :
8 mai 2013 à 23:55

Bonjour,
Dans votre article on trouve ceci:

_____
Tous les x millisecondes, faire :
erreur = consigne - mesure;
somme_erreurs += erreur;
variation_erreur = erreur - erreur_précédente;
commande = Kp * erreur + Ki * somme_erreurs + Kd * variation_erreur;
erreur_précédente = erreuruy
_____

Ne manque'il pas qqc au niveau de la ligne "sommes_erreurs += erreur" ?
Je veux dire après le "+" ?
Ne serait-ce pas "sommes_erreurs + erreur précédente = erreur" ou qqc comme ça ?

Par avance merci de votre réponse et en tout les cas, merci pour cet article simple et limpide.

Répondre

Ferdinand Piette dit :
9 mai 2013 à 00:09

Non non, il n'y a pas d'erreur 😉
http://www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-c/les-raccourcis

Répondre

9. **M4nux** dit :
3 juillet 2013 à 12:41

Bonjour Ferdinand
Merci pour cet article qui m'a permis de comprendre un peu mieux la régulation PID (l'article de wiki est trop ardu pour moi). Ceci dit, pourrais-tu éclairer ma lanterne. Intuitivement, la commande que tu définis comme
commande = Kp * erreur + Ki * somme_erreurs + Kd * (erreur - erreur_précédente)
est bien en réalité la "correction" à apporter au système pour qu'il corresponde au mieux à ta consigne non ? Si oui, alors cette commande au fil du temps, avec les erreurs qui s'additionnent dans le terme "Ki * somme_erreurs" , ne peut que croitre ? Or y'a un truc qui m'échappe, c'est qu'en principe, plus le temps avance, plus cette correction doit diminuer non ? Merci de tes éclaircissements car je suis un peu perdu.
Emmanuel

Répondre

Ferdinand Piette dit :
9 juillet 2013 à 15:57

Bonjour,

La "commande" correspond bien à la commande à appliquer au système et non au terme de correction.
"Ki * somme_erreurs" n'augmente pas forcement. Il se peut que l'erreur soit négative 😉

Donc si ton système est au repos et que tu lui appliques une consigne possitive, l'erreur sera > 0, la somme des erreurs augmentera avec le temps.
Lorsque le système dépassera la consigne, alors l'erreur est < 0 et la somme des erreurs commencera à diminuer. Ferdinand

Répondre

David dit :
4 décembre 2014 à 14:19

Bonjour,
En fait vous avez tous les deux raison.
Ce que Ferdinand explique est juste, et les doutes de M4nux sont fondés.
Ce qu'il faut comprendre c'est que la commande est la dérivée de l'évolution du système. Sur un moteur ou un système de chauffage, la commande rajoute (ou pas) de l'énergie et le système l'intègre.
C'est une intégration mécanique (ou thermique) et non numérique.
Ainsi si un moteur c'est stabilisé avec une erreur nulle et bien la commande va devenir nulle... le moteur va un petit peu ralentir et a ce moment la, la commande va renvoyer un peu de courant.

J'ai eu les mêmes doutes que M4nux car j'utilise les PID dans des simulations numériques (comme par exemple des simulations d'ABS), et dans ce cas la nous devons "dans le code" considérer que cette commande est un delta de commande et garder la commande dans une

autre variable.

Sinon un grand merci pour cet article; je l'ai conseillé a des gens qui ont du mal avec les PID 🙂

Cordialement.

Répondre

David dit :
5 décembre 2014 à 08:42

Oubliez ma réponse.
c'est bien le terme "Ki * somme_erreurs" qui permet d'obtenir une commande qui se stabilise autour d'une valeur moyenne et donc ne converge pas à zero quand le système atteint la consigne.

Cela revient a peu prés à la même chose que si cette commande n'était q'une variation de commande et si seul Kp était utilisé(c'est d'ailleurs comme ça que je faisais la majorité de mes PID quand je voulais utiliser un minimum de multiplications).

en gros, ça donne ça:

commande = 0
Tous les x millisecondes, faire :
erreur = consigne - mesure;
variation_commande = CoefQuiVaBien * erreur ;
commande = commande + variation_commande

ça a l'avantage de remplacer deux multiplications par une multiplication + une addition (sur certains systèmes ça compte) , mais ça prive d'une souplesse de réglage qu'on avais avec Kp et Ki séparés, car ça revient à un Kp = Ki = CoefQuiVaBien

Vous aviez fait le tour de la question.

Cordialement

Répondre

10. **Mathieu** dit :
3 octobre 2013 à 09:13

Bonjour !

Bravo pour le partage (toujours du succès 2 ans plus tard 😉
et félicitations pour la clarté ! merci

Pour approfondir ensuite le sujet, il y a un article de vulgarisation qui met en avant les différences de régulation entre 3 systèmes (moteur, positionnement de précision, thermique) en anglais ici PID-Without-a-PhD

Répondre

11. **Selso** dit :
6 novembre 2013 à 18:31

Bonjour,
Merci pour cette explication, qui m'a rafraîchit la mémoire. D'accord avec l'élève qui dit qu'on devrait commencer la matière avec cet article.

Répondre

12. **Valentin** dit :
11 novembre 2013 à 00:53

Coucou ! Je te connais...Ailleurs...
Si tu en a d'autres...des explications aussi lumineuses. Et ban, on est toujours preneurs.
C'est si agréable de comprendre...Gros: Merci !

Répondre

13. **David** dit :
28 novembre 2013 à 22:48

Merci pour ce tuto... c'est exactement ce que je cherchais pour réguler le ventilateur de mon système de chauffage qui pulse de l'air dans ma salle de bain

Répondre

14. **Fillon** dit :
17 janvier 2014 à 11:30

Bonjour Ferdinand,

Merci pour le partage de ton travail et ta générosité.
Je commençais à écrire un TP pour mes étudiants sur une régulation PID de moteur avec Arduino: ton projet tombe à pic.
Lorsque j'ai fini d'écrire mon TPet que celui-ci a tourné, je t'envoie mon .doc: échange de bon procédé. 😉
Je te souhaite une bonne journée, à bientôt,
Lionel Fillon.

Répondre

Ferdinand Piette dit :
17 janvier 2014 à 13:00

Super ! ça m'intéresse beaucoup 🙂
Merci et bonne journée aussi,
Ferdinand

Répondre

15. **Manu** dit :
6 mars 2014 à 09:32

Bonjour
Je suis du même avis que tous ici pour la clareté et l'interet de cet article. Merci!

Je suis arrivé ici car je cherchais des informations pour modifier de paramètres de régulation sur un PID. Je travail sur un process qui chauffe un produit dans une cuve grâce à une double envellope où circule soit de l'eau chauffer soit refroidie.
Il est très difficile d'obtenir une température produit suffisament stable.
Le premier problème, c'est que le régulateur ne commence à inverser les vannes de chaud/froid que lorsque la T° de la cuve arrive à la consigne. L'inertir présente dans l'eau de la double envellope continue dont de modifier la température produit le temps que le cycle s'inverse.
Je pense qu'intervenir sur le D permettrait de régler cela.
Etes-vous en accord avec mon idée? Avez-vous d'autres solutions?
Merci
Manu

Répondre

16. **Junior jackson mukendi** dit :
30 mars 2014 à 12:49

slt ,je suis vraiment emerveiller de ton article ,parceque cela ma permis d'enlever certaine embiguites que j'avais sur le regulateur PID

Répondre

17. **Nvince** dit :
5 avril 2014 à 23:43

Une précision,
de plus en plus Kd est appliqué uniquement à la variation de la mesure et non pas à l'écart mesure-consigne, et ce, afin d'éviter un gros accoup sur la sortie (accélérateur) en cas de changement de consigne...

Répondre

18. **Michel** dit :
16 juillet 2014 à 21:02

Bonsoir,

Testé avec de bon résultat sur une régulation de niveau d'une cuve avec un automate industriel,
avec l'avantage de pouvoir voir comment se comporte les paramètres P I D contrairement
au régulateur intégré (C'est beaucoup plus facile a régler ainsi)

Cela permet de bien comprendre le comportement du régulateur dans les différentes phases
de régulation

Répondre

19. **Alex** dit :
25 juillet 2014 à 20:25

Bonjour

je travaille sur un projet fin d'étude qui est intitule d'enregistrer d'événement sur un reseau triphase, mon probleme que je vais utiliser carte arduino et je choisis la methode PLL pour detecter la perturbation dans le reseau ,j'ai le programme mais je ne peux pas l'inserer sur la carte parce que je ne comprends pas la programmation.pouvez vous me donner une solution ou des conseils peux m'aider à resoudre ce probleme
contatez moi sur mon email
merci d'avance

Répondre

20. **Jérémie** dit :
30 juillet 2014 à 15:51

Article génial!!! La comparaison du début est très représentative de la régulation.

Répondre

21. **Christophe** dit :
28 août 2014 à 09:16

Merci beaucoup,
article explicite j'adore ! je suis en plein dedans ca m'a vraiment éclairé 🙂

Répondre

22. **Uriel** dit :
16 novembre 2014 à 21:47

Merci bien pour le calcul !!

J'ai testé le code sur arduino. Je ne trouve pas une régulation "idéale".
Donc, j'ai placé avec une résistance 10ohm chauffant une thermistance.
La puissance est réglée par pwm sur une mosfet régulant la résistance.

Je calibre Kp : OK
Je calibre Ki : là, on monte très haut au dessus de la consigne et on revient toujours se stabiliser en dessous.
Je rajoute Kd au précédent système : il rajoute de la souplesse mais garde les défauts des résultats précédents.

J'ai donc gardé Ki à 0. Je reprends un Kp qui marche assez bien et j'ai simplement élevé la température de consigne "virtuelle" afin que la température réelle se stabilise à la consigne souhaitée.
On a donc de la sorte le "remplacement" de la fonction de Ki.
Ensuite je rajoute un Kd.

Je reste sur une courbe qui monte, qui dépasse la température de consigne de 3°C environs, et vient se stabiliser.

Si d'autres font des essais, je serais curieux de voir vos résultats !

Merci !
Et bonne continuation 😉
?

Répondre

23. **Sebastien** dit :
10 février 2015 à 09:55

Super ! Merci Beaucoup !!!

Répondre

24. **Omar S.** dit :
17 mars 2015 à 22:25

Franchement TIP TOP ! ça fait deux ans que je fait de l'auto dans le cadre de mes études et je ne comprends que maintenant. Bravo ! C'est grâçe a des gens comme vous que le monde avance !

Répondre

25. **JF Delaitre** dit :
5 septembre 2015 à 09:12

Superbe esprit de synthèse et de pédagogie, je vous remercie.

Je vais travailler sur les régulations de mon installation de biogaz de façon un peu plus raisonnée.
Le raisonnement est un peu plus délicat quand on a plusieurs régulations PID à gérer et pour l'instant, je n'ai pas la main sur le paramètre Kd

Répondre

26. **Adlain** dit :
28 décembre 2015 à 11:34

Super. Je prépare un TP sur la régulation PID et tu as déjà tout fait.. ou presque! J'aimerai utiliser ce que tu as fait... avec ton consentement.
Merci beaucoup pour ce partage.

Répondre

> **Ferdinand Piette** dit :
> 22 février 2016 à 18:17
>
> Bonjour,
>
> Avec un peu de retard : oui, pas de problème.
> Bonne soirée
>
> Répondre

27. **Denis** dit :
12 février 2016 à 14:17

Super article, il est parfois difficile de visualiser certaines choses, heureusement que des gens comme toi existent.

Répondre

28. **Julien971** dit :
22 février 2016 à 17:06

Bonjour,
article très très intéressant !!!
Merci beaucoup !!!

Je suis face à un process très lent et je souhaiterais aller plus loin et programmer en C un prédicteur de Smith...
Avez-vous une idée de comment le programmer 🙂🙂

Merci pour vos conseils !!!
Julien

Répondre

29. **am** dit :
2 mai 2016 à 18:55

bonjour
super article bien expliquer (merci)
j ai une question : je veux implémenter une autre régulation autre que le pid dont j ai besoin de calculer la dérive et l'intégral du signal,
pour calculer la dérive il faut qu'on devise par le pas et de mémé pour l'intégral ?

Répondre

30. **arthur** dit :
17 mai 2016 à 19:27

Bonjour a tous. Je suis vraiment émerveillé après la lecture de l'article ainsi que vos differentes discussions. Moi je travail en ce moment sur un project de regulation de tem'expliquempérature avec un capteur lm35 pid. J'ai un souci majeur sur la communication avec la sorti. Je m'explique. En effect je doit modifié la consigne a tout instant, le circuit de puissant est formé du triac et des lampes chauffante. Je ne parvient pas a faire stabilité la température ambiante d'un box par example a celle de la consigne. Mon program prend en compte les parametres P, I et D. SVP si quelqu'un a une idee ou a deja fait cela, son aide sera la bien venu. Je programme en C dans le ccs. Merci

Répondre

31. **jeremy** dit :
15 juin 2016 à 10:20

Oui merci beaucoup + parlant 🙂

Répondre

32. **Jpc88** dit :
26 juin 2016 à 21:16

Bonjour
Je dois faite une régulation de température sur un circuit d'huile avec une vanne 3 voies et un sonde pt100.J'utilise une fonction pid sur un automate programmable et la vanne est réglable en position sur une plage de 0 a 100% non pas par

une sortie analogique mais en pilotant deux sorties tout ou rien ouverture et fermeture.Le problème c'est que je n'ais pas d'information de retour de la position de la vanne.Si la valeur de sortie du pid est par exemple de 70% ,comment fixer l'ouverture de la vanne a cette valeur?

Répondre

33. **Yastrib** dit :
3 mai 2017 à 14:17

Tous ce qui disent super, c bon, merci tout est fait ce sont ceux qui n'ont pas compris l'asservissement

Répondre

> **BERTIN GILLES** dit :
> 7 janvier 2019 à 12:04
>
> Bonjour,
> Je travaille depuis longtemps a réguler la vitesse de ma production hydroélectrique(18kw).Pour cela, j'utilise un automate M221 de chez Schneider. L'entrée de capture vitesse se fait par un codeur qui donne 250000 points par tour.Le contrôle de la vitesse se fait par l'action de 89 charges statiques, enclenchées (ou l'inverse) par une boucle PID logiciel.Toutefois la précision obtenue n'est que de +/- 5% .
> Je cherche a améliorer celle-ci. Dans les articles ci-dessus ,il manque un exemple chiffré qui permettrait d'intégrer les valeurs de chacun, pour obtenir un résultat . par exemple ci-après :
> somme_erreurs += erreur ? : la somme des erreurs est égale a combien de chacune des erreurs
> relevées toutes les bases de temps ?
> ou alors c'est le total infini, qui tend vers zéro?
>
> exemple : consigne 250000
> erreur, sur une base temps d' 1 microseconde = valeur lue - 250000
> somme de toutes les erreurs relevées : proche de zéro (normalement)
>
> variation erreur : erreur(a T-1) - erreur de l'instant
>
> La commande de frein va 0 et 89 .Donc plus l'erreur est grande, en positif , plus la commande doit se rapprocher de 89 et inversement.
>
> celle-ci serait donc : ?
> merci de votre aide
> g Bertin
>
> Répondre

34. **Elom** dit :
19 juillet 2017 à 10:12

Bonjour, merci beaucoup pour cet article super clair et la méthode facile à implémenter dans n'importe quel programme. Ca démystifie le PID pour moi ! 🙂

Un seul élément me chiffonne : l'intégrale calculée par "somme_erreurs += erreur " n'est pas limité dans le temps, alors que la dérivée est limitée au "delta t" d'échantillonnage : "erreur - erreur_precedente"

En regardant la page Wiki anglaise, on voit que l'intégrale est entre 0 et t : PID controller (anglais)

Donc je pense qu'il faut limiter la somme des erreurs à un delta t. Par exemple en additionnant les N dernières erreurs. Mais ce serait peut-être encore faux, car dans leur formule, on voit que le "t" utilisé est le même pour la dérivée que pour l'intégrale, ce qui laisserait penser à utiliser uniquement la dernière erreur, mais je pense qu'il doit exister des théories pour discrétiser de façon efficace (sur-échantillonner pour cumuler les erreurs plus souvent ? est-ce que ça aurait un intérêt ?).

Quel est l'effet néfaste de ne pas limiter l'intégrale dans le temps ? imaginons que nous ayons un moteur thermique diesel de camion (j'ai trouvé le pire exemple je crois) extrêmement lent à monter en régime, alors la somme cumulée des erreurs sera énorme avant de pouvoir devenir négative (disons 15 secondes d'erreurs cumulées de montée à 2400 tr/mn), donc l'oscillation sera d'autant plus grande que l'on a mis du temps à atteindre la consigne, ce qui me paraît contre-intuitif : le dépassement sera très long et de grande amplitude (et nous risquons le sur-régime moteur 😉).
Alors qu'en limitant dans le temps, la correction apportée au PID sera grande les 3 premières secondes (grosses erreurs cumulées), et faible lorsque l'on s'approche de la consigne (les dernières erreurs à cumuler sont faibles), ce qui produira un dépassement bien moindre.
Merci en tous cas pour cet article.

Répondre

35. **Elom** dit :
20 juillet 2017 à 09:06

Après réflexion concernant la composante "intégrale", comme elle correspond je pense à l'aire dessinée par la courbe de l'erreur pendant le temps "delta t", je pense que l'on peut considérer que l'erreur a progressé de manière linéaire et calculer :

```
erreur_integrale = delta_t * (erreur_precedente + erreur) / 2
```

Répondre

36. **BNALLEL** dit :
6 janvier 2018 à 16:39

Un grand merci, ca m'a trop aidé votre article, bien claire et trés efficace pour les applications réel, en tous cas pour moi ca marche trés bien pour le réglage de température de pression et débit sur une station de pompage, en partie grace à vous, je vous remercie infiniment

Répondre

37. **Thibault** dit :
14 mars 2018 à 14:05

Bonjour Ferdinand!

Merci beaucoup pour ces explications très claires pour un néophyte comme moi.
J'ai pu m'en servir pour programmer un contrôle de gaz afin de correspondre à la vitesse terminale dans un simulateur spatial, et cela a fonctionné à merveille après de nombreux ajustements!

Bonne continuation,
Thibault

Répondre

38. **G B** dit :
18 mars 2018 à 16:41

Bonjour,
Bonnes explications mais, a bac moins 5 , un exemple, tant concret que complet; un exemple chiffré me serait utile pour, enfin, réaliser un PID sur un automate de régulation de vitesse d'une production hydraulique .
Le pid logiciel inclus dans l'automate (M221 Schneider) n'est pas assez rapide.

exemple : valeur consigne 125000 (lecture vitesse codeur); l'action a obtenir joue sur 90 combinaisons de charges .

l'erreur entre la consigne et la lecture est facile a obtenir a l'instant T ; il est aussi possible d'en faire un cumul sur un temps déterminé par un temporisateur logiciel .

Ici vient la difficulté du comment employer ces valeurs pour sortir une valeur entre 1 et 90 qui actionne les relais statiques?

Dans les formules qui apparaissent dans ce tutoriel , la valeur temps se lit ou ? et comment l'intégrer dans ces formules?

Merci

Répondre

39. **Nicolas** dit :
3 septembre 2018 à 23:37

Bonjour,
merci pour cet article.
Je voudrais ajouter une remarque. L'implémentation

Tous les x millisecondes, faire :
erreur = consigne - mesure;
somme_erreurs += erreur;
variation_erreur = erreur - erreur_précédente;
commande = Kp * erreur + Ki * somme_erreurs + Kd * variation_erreur;
erreur_précédente = erreur

pose le problème suivant :
Une fois que le terme intégral s'est chargé d'annuler l'erreur et que l'on est en régime établi, on a
erreur = 0
variation_erreur = 0
Donc commande = Ki*somme_erreurs.
Si la consigne de vitesse du moteur est non-nulle, la commande sera non nulle également.

Si l'on est en phase de réglage des gains du correcteur, on aura un mauvaise surprise...
Pour exemple si en régime établi, la commande vaut 5 V, et que l'on passe Ki de 1 à 1.5, la commande va s'envoler à 7.5V. Alors que le moteur tournait à la bonne vitesse, il va se mettre à accélérer !

Pour résoudre cela il faut implémenter :
Commande_intégrale = 0
Tous les x millisecondes, faire :
erreur = consigne - mesure;
Commande_intégrale = Commande_intégrale + Ki*erreur;
variation_erreur = erreur - erreur_précédente;
commande = Kp * erreur + Commande_intégrale + Kd * variation_erreur;
erreur_précédente = erreur

Si l'on reprend l'exemple précédent, le moteur tourne à la vitesse de consigne (erreur = 0, variation_erreur = 0) et commande = 5V = Commande_intégrale.
On est en train d'ajuster Ki et on le fait passer de 1 à 1.5
Alors Commande_intégrale = Commande_intégrale + Ki*erreur = Commande_intégrale = 5V
Le moteur continue de tourner à la valeur de consigne et on procèdera alors tranquillement à un changement de consigne pour observer l'effet du nouveau réglage...

Répondre

40. **Max** dit :
4 février 2019 à 13:14

Merci d'avoir réussi à simplifier l'explication d'un PID. J'ai lu et regardé beaucoup de guides et celui-ci est de loin le meilleur.

Le plus intimidant c'est évidemment les longues formules mathématiques qui viennent avec la majorité des explications Ici ce n'était pas le cas, merci encore.

Répondre

41. **Amal GHAZOUANI** dit :
21 mars 2019 à 15:58

Je vous remercie énormément pour cette explication, elle est trop loin devant par rapport aux explications qui m'ont rencontré.
Je n'ai qu'un seul blocage au niveau du Soft, comment choisir la période d'échantillonnage ou encore la période du PID ??
J'attends une réponse le plus vite possible.
Merci pour l'attention que vous y porterez 🙂

Répondre

42. **Guillaume** dit :
17 juillet 2019 à 18:36

Bonjour M. Piette,
J'aurai aimé utiliser ce texte dans le cadre d'un de mes cours à l'Université.
Serait-ce possible d'en discuter par courriel ?

Répondre

43. **Louis** dit :
30 mai 2021 à 11:56

Bonjour monsieur,
dans l'implémentation du PID, ne faut-il pas diviser par l'intervalle de temps dt pour le dérivateur? De même ne faut-il pas multiplier par dt pour l'intégrateur?
Merci d'avance

Répondre

Ferdinand Piette dit :
6 septembre 2021 à 08:28

Bonjour, dans ce qui est écrit ici, la partie temporelle est intégrée aux coefficients ki et kd.
Si la fréquence d'échantillonnage est fixe, il n'est pas forcement utile d'exprimer le dt à part, il peut être intégré aux coefs (il y a un calcul en moins à faire par le processeur à chaque tick).
Par contre si les coefficients sont déjà trouvé et que la fréquence échantillonnage doit être changée, il faut alors recalculer les coefficients en fonction.

Répondre

Laisser un commentaire

Your Comment

bilink **href=""**b-quotecode:-):|:x:twisted::smile::shock::sad::roll::razz::oops::o
:lol::idea::grin::evil::cry::cool::arrow::???::?::!:?
Apercu

Vous pouvez utiliser ces tags et attributs HTML&nsbp;: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <s> <strike> <strong>`

Name (required)

E-mail (required)

URI

Enregistrer mon nom, mon e-mail et mon site web dans le commentaire.

Soumettre le commentaire

Ce site utilise Akismet pour réduire les indésirables. En savoir plus sur comment les données de vos commentaires sont utilisées.