Apports des outils numériques sur l'enseignement de l'automatique : ateliers à partir de notebook Jupyter

Culture Sciences de l'Ingénieur

école normale supérieure paris—saclay

Javier OJEDA

Édité le 08/07/2020

Cette ressource est issue d'une publication du numéro 101 de La Revue 3EI de juillet 2020. *Javier Ojeda est Maître de Conférences au Département EEA (Électronique, Électricité et Automatique) de l'ENS Paris-Saclay.*

L'évolution des outils numériques aussi bien logiciels que du matériel permet d'envisager l'enseignement de l'automatique de manière plus illustrative et participative. Illustrative, car des outils numériques, le plus souvent libres, sont disponibles pour les étudiants afin de mettre en image des notions parfois complexes de l'automatique et participative, car au travers d'ateliers réalisés par les étudiants, ceux-ci peuvent appréhender des notions principales ou annexes par leurs simulations et expérimentations. Par ailleurs, participative, par le fait que les outils numériques permettent des échanges enseignant-apprenant en ligne via un site web, des plateformes moodle, slack, etc.

Dans cette ressource, nous allons nous intéresser aux outils numériques logiciels par le biais du langage Python et du notebook Jupyter. Pour cela nous allons décrire trois ateliers enseignant-apprenant réalisables en ligne et qui abordent trois notions de l'automatique : l'identification, le dimensionnement automatique d'un correcteur de type PID et le calcul des pôles en boucle fermée.

1 – Introduction

Avec l'évolution des différents outils numériques, aussi bien des plateformes de partage de connaissance (moodle, slack, etc.), que des outils logiciels en eux-mêmes, ont amenés un profond changement de la manière d'enseigner [1] notamment en permettant que les apprenants possèdent des ressources numériques supplémentaires pour continuer à étudier en autonomie.

Dans le cadre de la préparation à l'agrégation de Sciences Industrielles de l'Ingénieur et Ingénierie Electrique [2], au département EEA de l'Ecole Normale Supérieure Paris-Saclay, il est demandé aux étudiants de préparer des ateliers pédagogiques qui illustrent une partie du cours d'automatique et ayant pour objectifs d'être utilisés durant un cours, td ou tp à un niveau post-bac. C'est dans cet esprit que trois ateliers sont proposés dans cet article. Ces trois ateliers doivent mettre en lumière trois parties du cours d'automatique :

- L'identification d'un système linéaire par une méthode ad hoc ;
- L'utilisation de la marge de gain pour le dimensionnement automatique d'un correcteur P, PI ou PID ;
- Le calcul symbolique des pôles d'un système en boucle ouverte et fermée.

Ces trois ateliers utilisent un notebook Jupyter et le langage Python. Les codes des trois ateliers sont téléchargeables librement depuis le site Culture Sciences de l'Ingénieur en pdf et python [3].

2 – Présentation du cadre des ateliers

2.1 - Les outils numériques logiciels pour l'automatique

Il existe différentes familles de logiciels numériques pour l'automatique. Le but de cet article n'est pas d'affirmer qu'il faut utiliser l'un plutôt que l'autre, mais plutôt de mettre en avant les avantages (subjectif) et inconvénients (également subjectif) de ces solutions.

- Matlab, Simulink & ses toolboxes : Logiciel de loin le plus complet. Permets la simulation sous forme graphique avec Simulink ou bien codée. Des toolboxes comme *Control System Toolbox* permet la simulation du point de vue temporel et fréquentiel des systèmes SISO et MIMO, le dimensionnement automatique de correcteurs, visualiser les pôles, etc. Néanmoins, cette solution est payante et peut être un lourd poids au budget des établissements. La difficulté de prise en main pour les étudiants est minime ;
- Scilab et Xcos [4]: Logiciel libre de simulation multiphysique et de programmation. Il offre des possibilités analogues à Matlab.

Python [5], [6]: Langage libre de programmation, interprété, multiparadigme et multiplateforme. C'est l'un des principaux langages préconisés par le ministère de l'Éducation nationale pour l'enseignement de l'informatique. Ces domaines d'application sont très vastes, de la NASA ou EDF pour ses solveurs éléments finis, Youtube, les jeux vidéo et la programmation scientifique notamment pour le machine learning (TensorFlow). Dans le domaine du calcul scientifique, des toolboxs comme Numpy offrent une riche variété de fonctions et affiche des temps de calcul se rapprochant de langages compilés comme le C. Pour l'automatique, Numpy et la toolbox Control System [7] permettent de gérer toutes les fonctions utiles au développement des ateliers.

1.2 - Outils de programmation et de visualisation

Pour l'écriture des ateliers, nous allons nous baser sur le langage Python et sur le notebook JupyterLab 1.0. JupyterLab [8], [9] est un notebook basé sur une application web exécutable depuis son ordinateur ou à distance. Jupyter gère des codes Python, mais également Julia, R et Scala. Ces différentes lignes de codes s'exécutent à la volée. Les graphiques sont inclus dans le notebook et des applications permettent la modification de valeurs en temps réel. Nous pouvons y écrire des parties en langage Markdown et Latex afin de créer des espaces de texte, paragraphes, explications, etc. Il permet ainsi de partager du code contenant du texte, du code, des équations et des figures. Par ces différents aspects, il se rapproche des *Live Scripts* de Matlab. Un exemple de fenêtre Jupyter a été représenté en Figure 1.

Figure 1 : Exemple d'un notebook Jupyter

L'une des fonctions très intéressantes d'un notebook Jupyter est la possibilité d'exporter le résultat final en de nombreux formats. Les formats les plus intéressants pour l'enseignement sont les exports en pdf via latex afin d'avoir un document synthétique disponible sous des plateformes de cours et l'export en html pour les sites web.

3 – Atelier n°1 : Identification d'un système d'ordre supérieur à 2

3.1 - Objectif de l'atelier et pourquoi le choix d'un ordre supérieur à 2 ?

Il est bien souvent tentant de prendre comme exemple des systèmes d'ordre 1 ou 2. Ils sont simples à appréhender pour les étudiants sur plusieurs points :

- Il est aisé de résoudre l'équation différentielle à la « main » ou de calculer la réponse à un échelon ;
- Le tracé du diagramme de Bode/Black/Nichols est aisé ;
- L'identification de ces systèmes est simple graphiquement ou via des abaques.

Cependant, lorsque le chapitre sur la stabilité des systèmes [10], [11] est abordé ces systèmes peuvent poser problème. En effet, comment aborder la notion de l'instabilité sur deux systèmes qui ne peuvent être instables ou qui ne possèdent pas de marge de gain et/ou de phase (bien que la définition de la marge de phase puisse être utilisée pour un système du premier ordre).

Ainsi, il peut être intéressant de varier les systèmes étudiés et de considérer des systèmes d'ordre supérieur à 2 dans le cadre de l'étude de la stabilité. De plus, tous les systèmes linéaires d'ordre supérieur à 2 peuvent être vus comme la sérialisation de systèmes d'ordre 1 et d'ordre 2. Pour la suite de l'article, nous allons nous intéresser à un système d'ordre 3 défini par :

$$H(p) = \frac{NUM(p)}{DENUM(p)} = \frac{K_0}{1+\tau p} \cdot \frac{1}{1 + 2m\frac{p}{\omega_0} + \left(\frac{p}{\omega_0}\right)^2}$$

Le pôle du premier ordre est le pôle dominant, il se situe à une pulsation de 1470 rd/s et le système du second ordre à une pulsation de 6670 rd/s avec un amortissement de 0,4. Les réponses indicielle et impulsionnelle (normée) de ce système sont représentées sur la Figure 2.

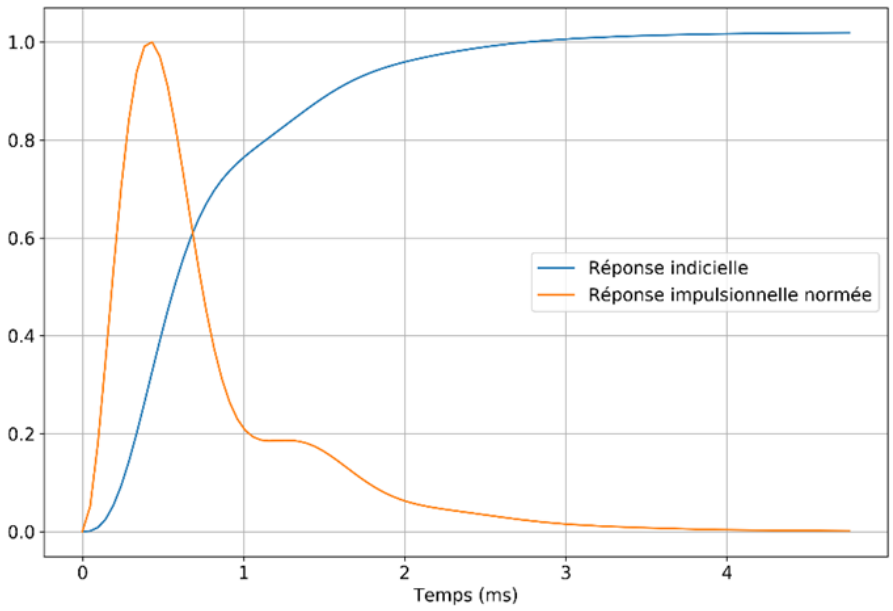

Figure 2 : Réponses indicielle et impulsionnelle du système étudié

Sur la réponse indicielle, nous observons bien un comportement majoritairement du premier ordre, car le pôle dominant est le pôle du premier ordre. Dans une première approximation, nous pourrions approximer ce système par un système du premier ordre. Cependant, nous remarquons une dérivée nulle à l'origine ce qui signifie que l'ordre du système est d'au moins 2. De plus, nous remarquons une cassure autour de 1 ms de la réponse indicielle. Ce système ne peut donc qu'être au moins un système d'ordre 3. Des remarques analogues pourraient être faites sur la réponse impulsionnelle. Nous reconnaissons en enveloppe un premier ordre, mais la réponse s'apparente à un second ordre. Nous avons donc au moins un ordre 3.

Ce système choisi est suffisamment « visuel » pour ne pas perturber les apprenants, ainsi, avec des raisonnements simples, il est tout à fait possible d'appréhender ce système (même s'il est hors programme de la plupart des sections).

3.2 - Identification du système par la méthode de Strejc

L'objectif de cet atelier est de faire programmer/simuler aux apprenants un petit algorithme d'identification automatique du système détaillé dans le paragraphe 3.1 par une méthode de Strejc [12]. La méthode d'identification de Strejc s'applique aux systèmes apériodiques et vise à approximer leur réponse indicielle par une fonction de transfert du type :

$$H_{Strejc}(p) = \frac{K_0}{(1 + T_d p)^n}$$

Dans le cas présent, cette méthode s'applique aux systèmes sans retard, mais elle peut être étendue aisément pour modéliser ces derniers.

Le principe est de venir mesurer graphiquement, Figure 3, les deux temps caractéristiques T_1 et T_2 qui vont permettre à partir d'un abaque (démontrable par quelques calculs) de calculer les paramètres T_d et n, le paramètre K_0 étant mesuré par la mesure du gain statique.

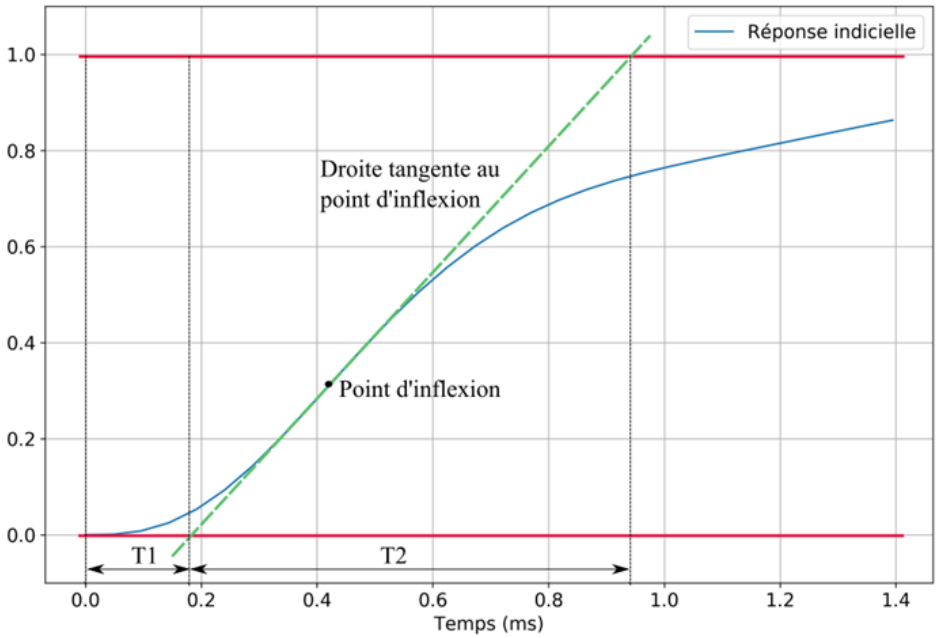

Figure 3 : Mesure graphique des paramètres de la méthode de Strejc

3.3 - Identification automatique par la méthode de Strejc

L'algorithme permettant l'identification par la méthode de Strejc se décompose en plusieurs parties :

- Déterminer la coordonnée du point d'inflexion par la détermination du maximum de la réponse impulsionnelle ;
- Déterminer l'équation de la droite passant par le point d'inflexion et qui tangente la réponse indicielle autour du point d'inflexion. La méthode employée dans cet atelier utilise une méthode des moindres carrés pour déterminer l'équation de la droite ;
- Déterminer les coordonnées des points d'intersection avec les droites des ordonnées nulles et des ordonnées égales à 1 ;
- À partir des coordonnées des intersections, détermination des temps T_1 et T_2 ;
- Lecture de l'abaque de Strejc pour déterminer T_d et n.

Sur la Figure 4 est représenté le résultat de l'algorithme d'identification par la méthode de Strejc. L'identification dans le cas présent donne un ordre n = 3 et une constante de temps T_d = 0,231ms. Cela donne un encadrement des pulsations du système :

$$\frac{1}{\tau} = 1470\ rd.s^{-1} < \frac{1}{T_d} = 4320\ rd.s^{-1} < \omega_0 = 6670\ rd.s^{-1}$$

La constante de temps identifiée est plus petite que la constante de temps dominante du système pour approcher le comportement au temps faible du second ordre. Néanmoins, à cause de cela, la représentation pour les temps approchant du régime permanent est dégradée.

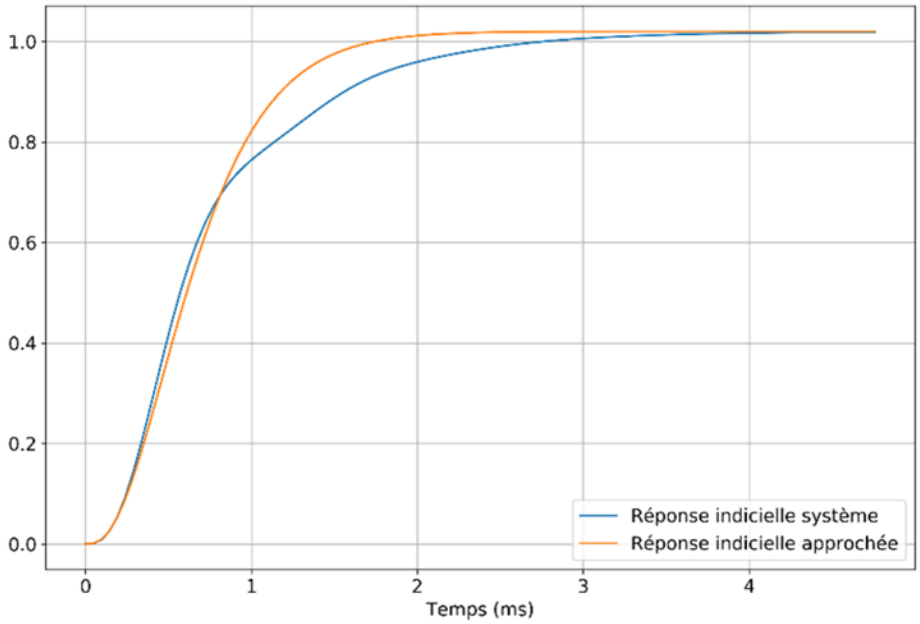

Figure 4 : Résultat de l'identification automatique par la méthode de Strejc

À partir de cette identification, il est alors possible de calculer un correcteur P, PI ou PID en utilisant la méthode de Ziegler-Nichols en boucle ouverte.

4 – Atelier n°2 : Le dimensionnement automatique de correcteur P, PI et PID à partir de la méthode de Ziegler-Nichols en boucle fermée

4.1 - Objectif de l'atelier

L'étude de la stabilité des systèmes passe pour les systèmes usuels par l'analyse des marges de stabilité : marge de phase, de gain et de module. En ce qui concerne le dimensionnement des correcteurs, c'est le plus souvent la marge de phase qui est employée. Dans cet atelier, nous proposons d'étudier la marge de gain pour dimensionner un correcteur et ainsi approfondir cette notion.

4.2 - Méthode de Ziegler-Nichols en boucle fermée

Dans cet atelier, nous proposons de dimensionner un correcteur P, PI ou PID en utilisant la méthode de Ziegler-Nichols en boucle fermée [13]-[15]. Cette méthode se base sur l'identification de deux paramètres caractéristiques en boucle fermée par un retour unitaire et un gain proportionnel dans la chaine directe lors du passage en fonctionnement instable. K_C, représentera le gain qui amène en limite de stabilité et ω_C, la pulsation associée à ce gain limite (point critique).

La marge de gain quant à elle est définie comme étant l'inverse du gain obtenu pour une phase de -180° et pour la pulsation ω_C. Ainsi, la marge de gain peut être définie par :

$$M_\varphi = \frac{1}{K_c}\Big|_{\omega=\omega_c}$$

Cette dernière se mesure très facilement par une méthode graphique sur une représentation de Bode ou Black-Nichols.

La mesure des marges de stabilité s'effectue grâce à la fonction :

```
INFO_SYSTEME = ctl.stability_margins(SYS_TF)

K_C = INFO_SYSTEME[0] # Marge de gain

W_C = INFO_SYSTEME[3] # Pulsation associée à la marge de gain
```

À partir de ces deux grandeurs, nous pouvons déterminer les valeurs du réglage d'un correcteur P, PI ou PID par l'abaque de Ziegler-Nichols.

Méthode de Ziegler-Nichols			
Correcteur	**K_p**	**T_i**	**T_d**
P	$0.5K_C$		
PI	$0.45K_C$	$T_u/1.2$	
PID	$0.6K_C$	$T_u/2$	$T_u/8$
Léger dépassement	$0.33K_C$	$T_u/2$	$T_u/3$

Tableau 1 : Paramètres des correcteurs pour la méthode de Ziegler-Nichols

Avec ces notations, le correcteur PID aura comme forme :

$$C(p) = K_p\left(1 + \frac{1}{T_i p} + T_d p\right)$$

La valeur de T_u est obtenue par :

$$T_u = \frac{2\pi}{\omega_c}$$

Il est intéressant de remarquer que les valeurs indiquées ne sont qu'indicatives. Il faudrait, ce que nous pouvons d'ailleurs retrouver dans certains ouvrages, pour chaque type de système calculer les paramètres qui donneraient une réponse avec/sans dépassement, avec une marge de phase adéquate, etc. Néanmoins, il est également notable que la partie proportionnelle du correcteur n'est qu'une fraction du gain critique, ce qui assure « partiellement » contre des réglages initiaux qui rendraient le système bouclé instable.

Le résultat de ce dimensionnement automatique en utilisant la ligne PID du Tableau 1 est représenté sur la Figure 5.

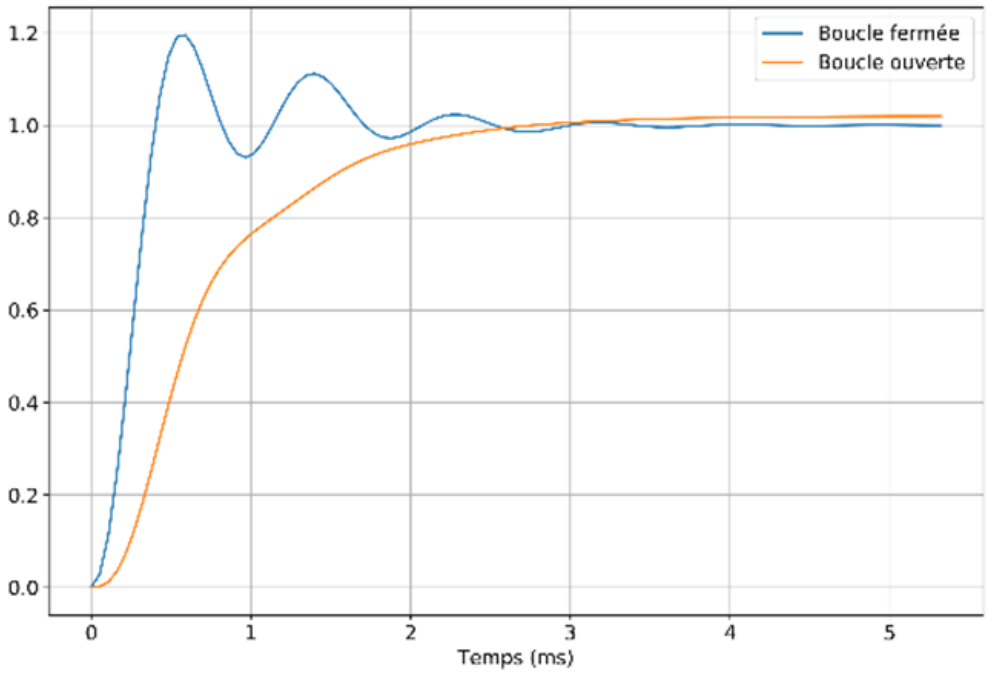

Figure 5 : Comparaison de la boucle fermée avec un correcteur PID et la boucle ouverte

Ce premier réglage obtenu par le dimensionnement automatique est intéressant. Premièrement, il amène à un système stable, mais également sans erreur statique et avec un temps de montée amélioré par rapport à la boucle ouverte. Nous pouvons cependant noter que le terme dérivé est un peu trop fort, ce qui est visible par la partie exponentielle décroissante pour les temps faibles de la réponse. Si le terme dérivé est diminué, il faudra diminuer le terme proportionnel en conséquence.

Ces différents réglages disponibles ne sont qu'un point de départ pour ensuite pouvoir affiner le résultat selon les besoins du cahier des charges.

5 – Atelier n°3 : Représentation des pôles pour l'étude de la stabilité

5.1 - Objectif de l'atelier

L'objectif de cet atelier est de présenter une méthode pour le calcul des pôles en boucle ouverte ou boucle fermée à partir de fonctions numériques. Cela permet de donner un éclairage supplémentaire et complémentaire des méthodes classiques comme le critère de Routh, etc., notamment dans le cas de l'étude de la stabilité.

5.2 - Utilisation du calcul symbolique

L'utilisation du package Sympy de Python permet d'effectuer du calcul symbolique. L'utilisation du calcul symbolique [16] permet de résoudre l'équation caractéristique du système et ainsi d'avoir la valeur de pôle pour l'étude de la stabilité.

```
import sympy as sm # Package pour le calcul symbolique

# Système à étudier
K_0 = 1.02              # Gain statique
TAU = 0.68e-3           # Constante de temps du 1er ordre
M = 0.4                 # Amortissement du 2nd ordre
W0 = 1/150e-6           # Pulsation du 2nd ordre

############################################################
# Fonction de transfert en BO
#

NUM = [K_0] # Numérateur de la fonction de transfert

# Dénominateur de la fonction de transfert par ordre
décroissant en p ou s
DENUM = [TAU/W0**2, 2*M/W0*TAU + 1/W0**2, TAU + 2*M/W0, 1]

SYS_TF = ctl.tf(NUM, DENUM) # Fonction de transfert en p

p = sm.symbols('p') # Variable de Laplace

# Résolution de l'équation caractéristique
POLES_TF_SM = sm.solve(DENUM[0]*p**3 + DENUM[1]*p**2 +
DENUM[2]*p + DENUM[3], p)

print('Les pôles sont ' + str(POLES_TF_SM[0]) + ' ; ' +
str(POLES_TF_SM[1]) + ' ; ' + str(POLES_TF_SM[2]))
```

5.3 - Le lieu des Yôles d'Evans

Il est intéressant également d'avoir une représentation graphique des pôles pour la stabilité et le comportement dynamique du système [17].

```
POLES_TF, ZEROS_TF = ctl.pzmap(SYS_TF) # Calcul des pôles et des zéros
puis représentation dans un plan
```

Sur la Figure 6 sont représentés les pôles du système en boucle ouverte calculés par la fonction pzmap, mais pourraient être représentés également à partir du calcul symbolique des pôles.

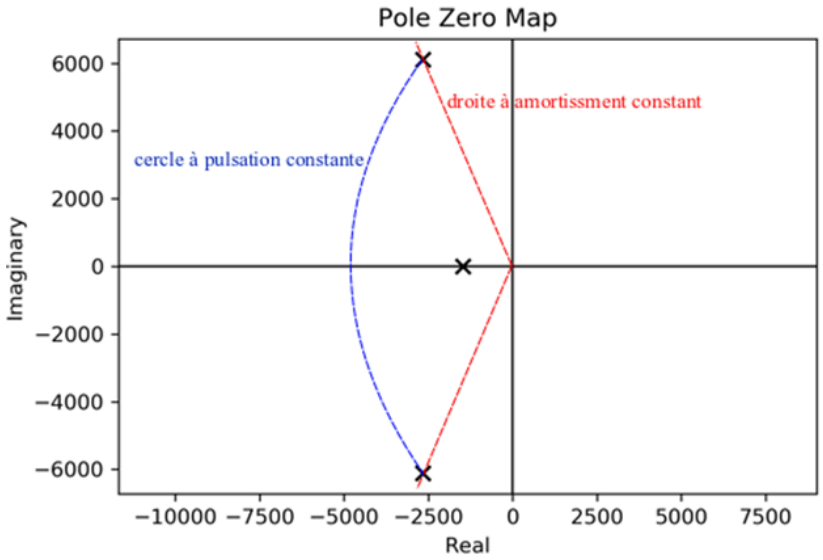

Figure 6 : Lieu des pôles en boucle ouverte

Nous voyons apparaître le pôle dominant du premier ordre et les deux pôles complexes conjugués du second ordre. Il est possible et souhaitable de faire figurer sur ce graphe les échelles pour les pôles du second ordre à coefficient d'amortissement constant (droites passant par l'origine) et à pulsation constante (cercles de rayon constant ayant pour centre l'origine). Nous pouvons alors caractériser le système en termes de stabilité et de prédire le comportement du système de manière qualitative. À partir de cette représentation des pôles en boucle ouverte, nous pouvons déduire le comportement du système pour un bouclage par un retour unitaire et un correcteur proportionnel dans la chaîne directe. Le gain de ce correcteur est variable entre un minimum et un maximum. Le lieu de tous les pôles en boucle fermée pour les différentes valeurs de correcteur proportionnel est alors représenté dans le lieu d'Evans (fonction root_locus), Figure 7.

```
VAL_K = np.linspace(1, 10, 100) # Valeurs souhaitées pour le correcteur
proportionnel

POLES_BF, K_POLES = ctl.root_locus(SYS_TF, VAL_K)
```

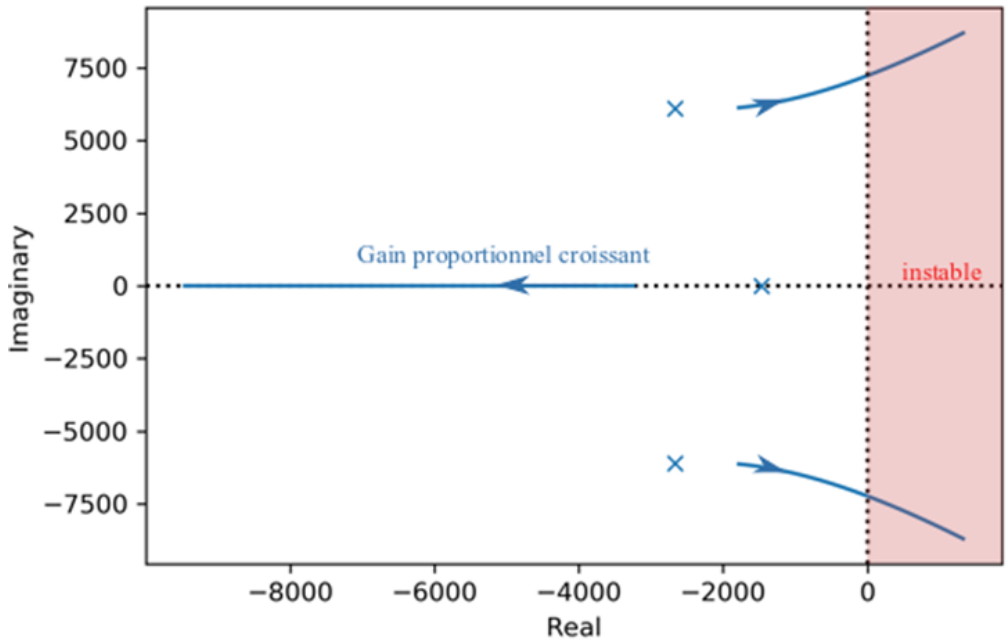

Figure 7 : Lieu des pôles d'Evans en boucle fermée par un correcteur proportionnel

À partir de ce lieu des pôles, il est alors possible de déterminer si le système sera stable ou non par un bouclage et de prédire la valeur du gain du correcteur proportionnel qui mènera à l'instabilité (gain critique). Il est possible d'étendre le calcul du lieu des pôles en boucle fermée pour un correcteur proportionnel intégral. À partir de la résolution de l'équation caractéristique pour une paire {gain proportionnel, constante d'intégration}, le lieu des pôles est représenté en Figure 8.

```
NB_K = 10 # Nombre d'éléments dans le vecteur des paramètres du
correcteur proportionnel
MAX_K = 10 # Valeur maximale de K
MIN_K = 1 # Valeur minimale de K
VAL_K = np.linspace(MIN_K, MAX_K, NB_K) # Valeurs souhaitées pour le
correcteur proportionnel

NB_TI = 10 # Nombre d'éléments dans le vecteur des paramètres du
correcteur intégral
MAX_TI = 10e-4 # Valeur maximale de Ti
MIN_TI = 1e-4 # Valeur minimale de Ti
VAL_TI = np.linspace(MIN_TI, MAX_TI, NB_TI) # Valeurs souhaitées pour le
correcteur intégral
```

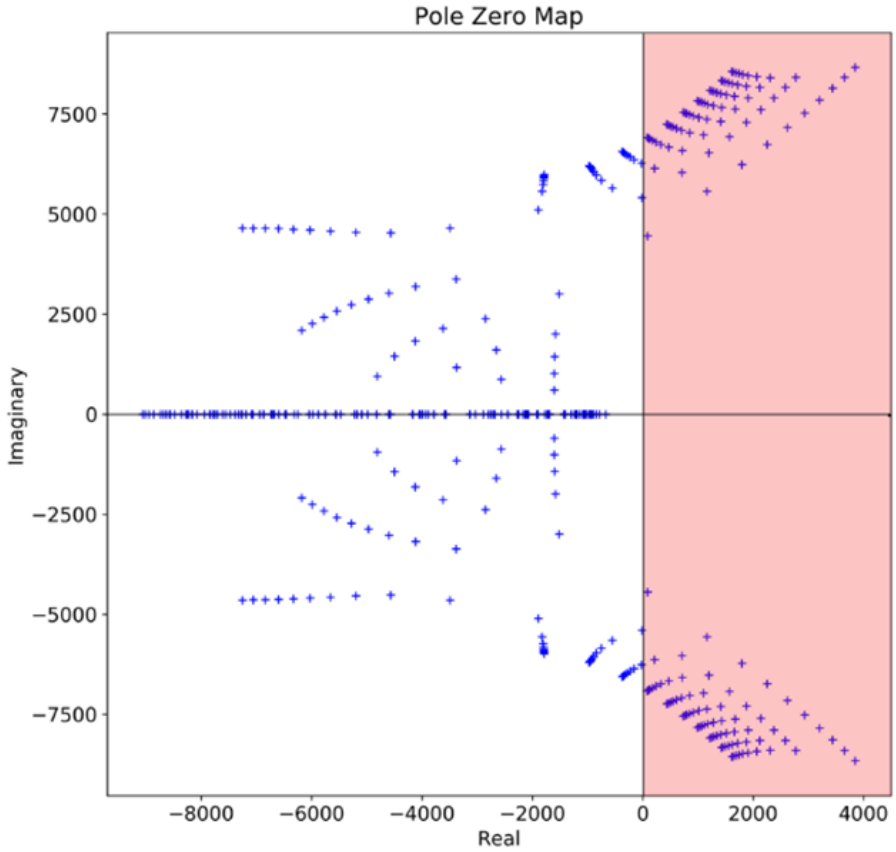

Figure 8 : Lieu des pôles en boucle fermée pour un correcteur PI

À partir de la Figure 8, il est possible de suivre les trajectoires des pôles en fonction des couples {gain proportionnel, constante d'intégration} et ainsi trouver les valeurs limites des paramètres. L'objectif est d'introduire par cette représentation la notion de placement de pôles. À partir d'un cahier des charges, celui-ci peut être traduit en termes de placement de pôles et par conséquent, un lieu géographique dans le lieu des pôles. Dans le cas d'un correcteur proportionnel, il est intéressant de chercher graphiquement la valeur de paramètre qui amènera les pôles en boucle ouverte vers les pôles en boucle fermée du cahier des charges.

6 – Conclusion

Au travers de trois ateliers, nous avons pu mettre en lumière des concepts de l'automatique très complémentaires de ce qui est enseigné principalement dans la discipline les que l'identification d'un système par la méthode de Strejc, l'utilisation de la marge de gain pour dimensionner un correcteur PID ou le calcul symbolique et la représentation des pôles dans un plan complexe.

Ces ateliers ont été également le prétexte pour découvrir l'utilisation du notebook Jupyter qui offre de nombreuses possibilités pédagogiques pour de l'enseignement synchrone ou asynchrone via un site web, des applications de partage, etc.

Références :

[1] "Enseigner avec le numérique." https://eduscol.education.fr/pid26435/enseigner-avec-le-numerique.html

[2] "Agrégation Sciences Industrielles de l'Ingénieur et Ingénierie Électrique." https://eduscol.education.fr/sti/formations/agregation-sciences-industrielles-de-lingenieur-et-ingenierie-electrique

[3] “Apports des outils numériques sur l’enseignement de l’automatique : ateliers à partir de notebook Jupiter”, J. Ojeda, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/apports-des-outils-numeriques-sur-lenseignement-de-lautomatique

[4] “Scilab/Xcos pour l’enseignement des sciences de l’ingénieur (french).” https://www.scilab.org/tutorials/scilabxcos-pour-l’enseignement-des-sciences-de-l’ingénieur-french

[5] “Une ressource : Python.” https://eduscol.education.fr/maths/actualites/actualites/article/une-ressource-python.html

[6] “Python Anaconda.” https://www.anaconda.com/products/individual

[7] “Python Control Systems Library.” https://python-control.readthedocs.io/en/0.8.3/

[8] “Jupyter Notebook.” https://jupyter.org/

[9] L. S.-E. Jean-Marc Kraëber, “Tutoriel : Utiliser un Jupyter Notebook.” http://lycee-saint-exupery.fr/tutoriel-utiliser-un-jupyter-notebook/

[10] P. Prouvost, Automatique - Contrôle et régulation. 2010.

[11] Y. Granjon, Automatique ; systèmes linéaires, non linéaires ; à temps continu ; à temps discret ; représentation d’état ; événements discrets ; cours et exercices corrigés (3e édition). 2015.

[12] S. Skoczowski and A. Osadowski, “A Simple Identification Method for the Order of the Strejc Model and its Application to Autotuning,” IFAC Proc. Vol., vol. 27, no. 3, pp. 319–325, Jun. 1994, doi: 10.1016/S1474-6670(17)46129-5.

[13] J. A. Romero, R. Sanchis, and P. Balaguer, “PI and PID auto-tuning procedure based on simplified single parameter optimization,” J. Process Control, vol. 21, no. 6, pp. 840–851, 2011, doi: 10.1016/j.jprocont.2011.04.003.

[14] G. P. Liu and S. Daley, “Optimal-tuning PID control for industrial systems,” Control Eng. Pract., vol. 9, no. 11, pp. 1185–1194, Nov. 2001, doi: 10.1016/S0967-0661(01)00064-8.

[15] K. J. Åström and T. Hägglund, “Automatic tuning of simple regulators with specifications on phase and amplitude margins,” Automatica, vol. 20, no. 5, pp. 645–651, Sep. 1984, doi: 10.1016/0005-1098(84)90014-1

[16] P. Sibille, “L’automatique des modèles linéaires revisitée à la lumière du calcul symbolique,” J3eA, vol. 7, p. 1001, Feb. 2008, doi: 10.1051/j3ea:2008001.

[17] G. W. Evans, “Bringing root locus to the classroom,” IEEE Control Syst., vol. 24, no. 6, pp. 74–81, Dec. 2004, doi: 10.1109/MCS.2004.1368483.