# Petit tuto pour utiliser Weyl-Newton.py (01/17)

## Introduction

• Dans de nombreuses applications, nous serons amenés à localiser les racines de polynômes complexes. Parfois, il nous faudra en connaître de bonnes valeurs approchées, et parfois, il nous suffira par exemple de savoir qu'elles sont toutes à l'intérieur du cercle unité, ou bien qu'elles sont toutes de partie réelle négative, ou bien encore qu'aucune est réelle.

On peut voir la question sous l'angle suivant : on se fixe un polynôme

$$a_0 + a_1 Z + \cdots + a_n Z^n \quad .$$

Disons qu'une partie du plan est

- ▶ coupable si elle contient des racines,
- ▶ innocente si elle n'en contient pas,
- ▶ suspecte si son statut coupable/innocente n'est pas encore tranché.

Si l'on nous donne une partie concrète du plan, $K$, (compacte, pour être exact) qui est *a priori* suspecte, nous voulons pouvoir lui délivrer un certificat d'innocence, ou bien la marquer définitivement comme coupable. Est-ce possible ?

• Dans la première partie, nous décrivons un algorithme (de Weyl) qui délivre en temps fini un certificat d'innocence **lorsque $K$ est innocente**. Si $K$ est coupable, l'algorithme ne permet pas de le certifier en temps fini, mais il permet quand même de traquer les racines, et de les enfermer dans des parties qui deviennent de plus en plus petites. On peut alors considérer que l'on a des bonnes valeurs approchées des quelques points qui sont très probablement des racines.

• Dans la seconde partie, nous explorerons le problème du certificat de culpabilité : comment faire pour se débarasser du "très probablement" dans la dernière phrase. (En chantier, les arguments sont déjà là).

• Enfin, nous explorerons la méthode de Newton pour trouver des valeurs approchées des racines (ça, c'est pas encore fait).

# 1 L'algorithme de Weyl

## 1.1 L'inégalité fondamentale

Soit

$$P(Z) = a_0 + a_1 Z + \cdots + a_n Z^n$$

un polynôme de degré $n$. On peut écrire

$$P(z_0 + Z) = b_0 + b_1 Z + \cdots + b_n Z^n$$

et avec ces notations, obtenir (grâce à l'inégalité triangulaire) l'inégalité

$$|P(z_0 + z)| \geq |b_0| - |b_1|.|z| - \cdots - |b_n|.|z|^n$$

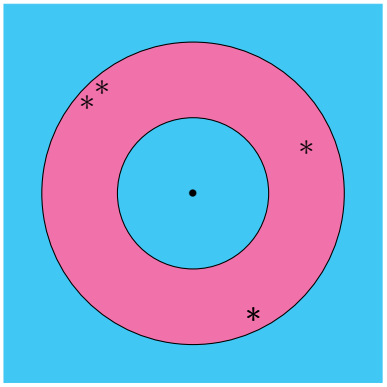
• En particulier, si pour un certain $r > 0$, on a $|b_0| - |b_1|h - \cdots - |b_n|h^n > 0$, alors on sait que le disque de centre $z_0$ et de rayon $h$ ne contient pas de racine. C'est là notre **test d'innocence** pour le disque $D(z_0, h)$.

• On en déduit aussi les inégalités de Cauchy :
posant

$$M^- = \max_{i>0} \frac{|a_i|}{|a_0|} \quad \text{et} \quad M^+ = \max_{i<n} \frac{|a_i|}{|a_n|} \quad ,$$

si $z$ est une racine de $P$, alors on a les inégalités

$$\frac{1}{1 + M^-} \leq |z \leq |1 + M^+| \quad .$$



Autrement dit, les racines (les astérisques sur le dessin) de $P(X)$ sont coincées dans une couronne (en rose sur le dessin) ...

Justifions vite fait l'inégalités de Cauchy :

$$\begin{array}{rcll} |P(z) - a_0| & = & |a_n z^n + \cdot + a_1 z| & \\ & \leq & |a_n|.|z^n| + \cdot + |a_1|.|z| & \text{(Inégalité triangulaire)} \\ & \leq & (\max_{i>0} |a_i|).|z|(|z^{n-1}| + \cdot + |z| + 1) & (|a_i| \leq m_1) \\ & = & (\max_{i>0} |a_i|).|z|\frac{1-|z|^n}{1-|z|} & \text{(Somme géométrique)} \\ & \leq & (\max_{i>0} |a_i|).\frac{|z|}{1-|z|} & (|z| < 1) \end{array}$$

Si $z$ est une racine, on a $P(z) = 0$, et donc

$$1 - |z| \leq M^-|z| \quad \Leftrightarrow z \geq \frac{1}{1 + M^-}$$

Justifions maintenant la seconde inégalité. Il y a une jolie astuce qui permet de s'en sortir rapidement :

Posons

$$P(\frac{1}{z}) = a_0 + \frac{a_1}{z} + \cdots + \frac{a_n}{z^n} = \frac{a_0.z^n + \cdots + a_1 z + a_n}{z^n} = \frac{Q(z)}{z^n}$$

Les racines (nécessairement non nulles) de $P(z)$ et de $Q(z)$ se correspondent par $\omega \mapsto \frac{1}{\omega}$. La première inégalité de Cauchy fournit une majoration des racines de $Q$, donc une minoration des racines de $P$, qui est celle reportée dans l'énoncé.

## 1.2 La battue

• L'idée est d'utiliser les inégalités de Cauchy pour réduire immédiatement la zone où il peut y avoir des racines, puis d'utiliser des tests d'innocence pour réduire de plus en plus l'espace où elles peuvent se cacher. Pour être efficace, cette traque doit être organisée : c'est ce que nous apprend à faire l'algorithme de Weyl.

• On commence avec le carré centré en 0 et de côté $2(1 + M_+)$. Ainsi il contient (tout juste) le disque coupable $D(0, 1 + M_+)$. En particulier, il est très suspect !

On le découpe alors en 4 plus petits carrés, *a priori* tous suspects, et de faire on fait des tests d'innocence pour ces carrés. Si un carré est innocenté, on l'oublie. S'il reste suspect, on le redécoupe en 4 plus petits carrés etc.

• Notre test d'innocence pour un carré $C$ est le suivant :

0. $C$ est un carré de centre $z_0$, de côté $c$.
1. On calcule les coefficients $b_i$ dans l'écriture

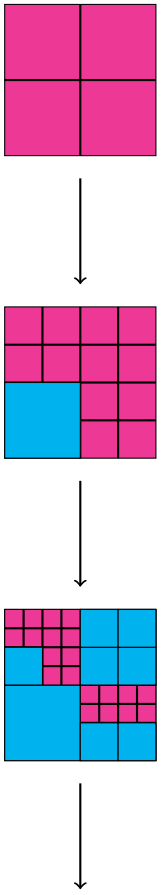
$$P(z_0 + h) = b_0 + b_1 h + b_2 h^2 + \cdots b_n h^n$$

2. On calcule

$$|b_0| - (|b_1| r + \cdots + |b_n| r^n)$$

avec $r = \frac{\sqrt{2}}{2} c$.

3. Si le résultat est positif, le carré est innocenté, sinon, il reste suspect.



$C_0$ est suspect, donc on le divise en 4. On applique le test à chaque nouveau carré. Si un carré est innocenté, on le colorie en bleu. Sinon, on le divise en 4

On applique le test à chaque nouveau carré. Si un carré est innocenté, on le colorie en bleu. Sinon, on le divise en 4

etc.

Remarque : le test d'innocence pour les carrés se déduit du test d'innocence pour les disques : Le carré de centre $P$ et de côté $2c$ est contenu dans le disque de centre $P$ et de rayon $\sqrt{2}c$.

## 1.3 Le module Weyl.py

L'algorithme de Weyl tel qu'il est décrit ici est implémenté en Python dans le fichier `Weyl.py`. Téléchargez-le, et dans une console, tapez

```
>>> from  Weyl2 import *
```

Ainsi, toutes les commandes définies dans `Weyl.py` sont maintenant utilisables. En particulier, le module cmath est importé au passage, et permet d'utiliser des nombres complexes, et le module Polynomials l'est aussi, sous le nom de Poly.

Les commades suivantes définissent le sens des symboles $X$ (le monôme de degré 1) et $I$ (l'unité imaginaire de $\mathbf{C}$).

```
>>> X=Poly([0,1])
>>> I=complex(0,1)
```

On peut alors définir des polynômes d'au moins 3 façons : par ses coefficients, comme combinaisons de puissances de $X$, ou en donnant ses racines (il existe un unique polynôme unitaire ayant ces racines là, et c'est celui que renvoie la commande `Poly.fromroots`).

```
>>> coefs=[-1,1,-1,0,1,-2,1]
>>> Pisot8 = Poly(coefs)
>>>
>>> Cyclotomic20=X**20-1
>>>
>>> roots=[1,1+2*I,-1-I,3-I,-1+3*I,-I]
>>> FromRoots=Poly.fromroots(roots)
```

Ces trois polynômes sont

$$\text{Pisot8} = Z^6 - 2Z^5 + Z^4 - Z^2 + Z - 1 \quad , \quad \text{Cylotomic20} = Z^{20} - 1 \quad ,$$

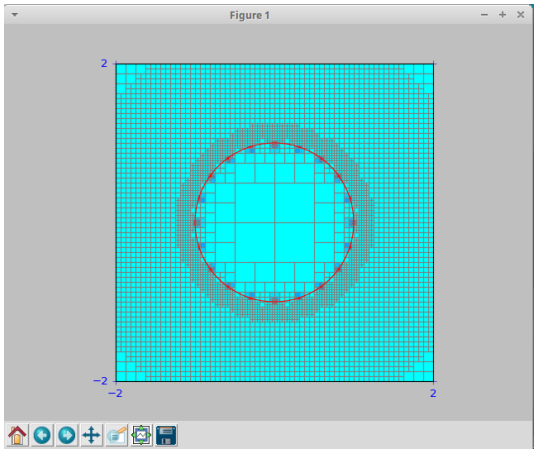
$$\text{FromRoots} = (Z-1)(Z-1-2i)(Z+1+i)(Z-3+i)(Z-i+3)(Z+i)$$

Nous connaissons bien les racines du polynôme Cyclotomic20 : ce sont les racines 20-èmes de l'unité $e^{i\frac{k\pi}{5}}$ pour $i = 0, \ldots, 9$.

Appliquons notre algorithme :

```
>>> PlotWeyl(Cyclotomic20,10)
```

Le nombre 10 en deuxième argument est le nombre de subdivisions au bout duquel l'algorithme doit s'arrêter.
Au bout de quelques secondes de calcul, une fenêtre s'ouvre et vous voyez apparaître l'image ci-à-droite




Le cercle rouge est le cercle unité. Ici, on a clairement $M^+ = 1$, et l'algorithme commence donc avec le carré centré en 0 et de côté $2(1 + M^+)$, dont les coins sont bien en $(\pm 2, \pm 2)$.

En promenant votre curseur sur l'image, ses coordonnées apparaîtront dans le coin en bas à droite, ce qui peut être utile pour connaitre une valeur approchée d'une racine.

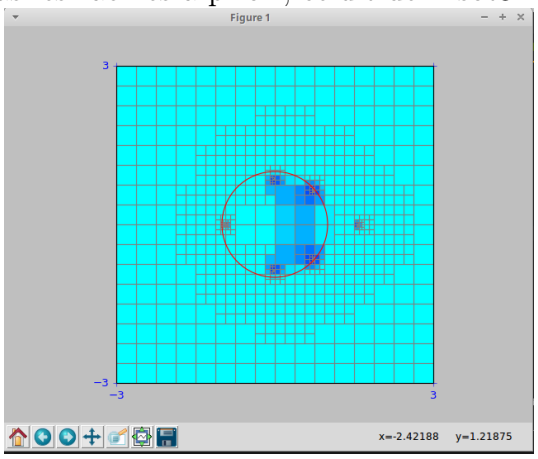
Pour plus de précision, on peut zoomer, en cliquant sur la petite loupe de la barre d'outils du bas.

Voyons cela sur un exemple dans lequel on ne connaît pas les racines a priori, celuit de Pisot8.

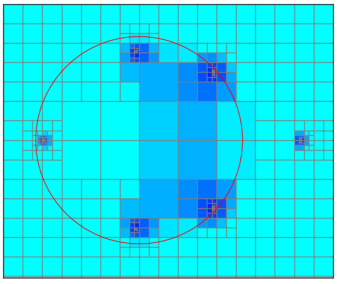
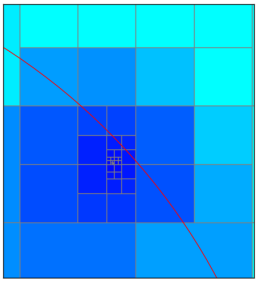
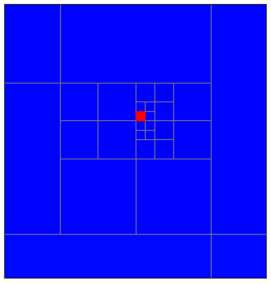
Appliquons notre algorithme :

```
>>> PlotWeyl(Pisot8,15)
```

Cette fois, le degré étant plus petit, on peut se permettre de demander beaucoup d'itérations (15). Au bout de 0.47 seconde de calcul, une fenêtre s'ouvre et vous voyez apparaître l'image ci-à-droite




On voit bien une racine réelle de module strictement plus grand que 1, une autre négative de module $< 1$, une paire de racines conjuguées de partie réelle légèrement négative et de module $< 1$, et une paire de racines conjuguées de module proche de 1. Pour en savoir plus, zoomons sur celle qui est en haut à droite. On obtient successivement :

Il est bien clair au second zoom que cette racine est à l'intérieur du disque unité, donc de module $< 1$. Notre polynôme a donc 6 racines dont une seule est de module strictement plus grand que 1. En zoomant autour de cette racine-là et en utilisant les coordonnées du curseur, on voit qu'elle est comprise entre 1.561 et 1.562.

## 2 L'accusation

On constate que les racines sont entourées de carrés dont les modules sont de plus en plus petit, ce qui n'est pas très étonnant : la fonction $z \mapsto P(z)$ est continue, et doit donc tendre vers 0 lorsque l'on se rapproche d'une racine.
Réciproquement, il semble bien, aussi, qu'il n'y ait jamais de carré bleu foncé entouré par des carrés plus clairs. Autrement dit, il semblerait que le théorème suivant soit vrai :

**Théorème.** *Soit $P$ un polynôme non-constant. Alors si $z \mapsto |P(z)|$ admet un minimum local en $z_0$, on a $P(z_0) = 0$.*

Ce théorème est vrai, et pas trop dur à démontrer : soit $z_0$ un complexe quelconque. On peut écrire

$$P(z_0 + Z) = b_0 + b_1 Z + \cdots + b_n Z^n$$

Si $a_0$ est nul, il n'y a rien à démontrer. Si le degré de $P$ est inférieur ou égal à 1, il n'y a rien à démontrer.
Supposons donc que $b_0$ soit non nul. Supposons aussi, pour simplifier que $b_1$ est non nul (le cas où $b_1, \ldots, b_i$ sont nuls se traite de la même manière). On peut alors écrire

$$P(z_0 - \frac{b_0 h}{b_1}) = b_0(1+h) + b'_2 h^2 + \cdots + b'_n h^n \quad .$$

Posant $m = \max_{i>1} b'_i b_0$, on en déduit l'inégalité

$$|P(z_0 - \frac{b_0 h}{b_1}) \leq |b_0|(|(1-h)| + (n-2)m|h|^2)$$

Lorsque $h$ est pris réel positif tendant vers 0, le terme $(n-2)m|h|^2$ devient négligeable devant $h$. Il sera en particulier inférieur à $h$ dès que $h$ sera inférieur à $\frac{1}{(n-2)m}$, ce qui donnera $|P(z_0)| > |(P(z_0 + \frac{b_0 h}{b_1})$ et montrera donc que $z \mapsto P(z)$ ne peut atteindre un minimum local en $z_0$. □

**Corollaire.** *Si après un certain nombre d'itérations, il existe un chemin de carrées bleus*

$$C_0 - C_1 - \cdots - C_s = C_0$$

*entourant des carrés rouges $c_1, \cdots, c_k$, et si le minimum du module de $P(z)$ sur ces carrés $C_i$ est supérieur ou égal au module d'au moins un point contenu à l'intérieur d'un $c_j$, alors l'un des $c_i$ contient une racine.*

En effet, la fonction $z \mapsto |P(z)|$ étant continue, elle admet un minimum $M_c$ sur chaque carré $c$ bleu ou rouge. Ce minimum peut *a priori* être sur un bord de $c$. Supposons que le plus petit de ces minimas est atteint en $z_0$. D'après nos hypothèses, ce $z_0$ peut être choisi dans l'intérieur de la partie entourée par notre chemin de carrés bleus. C'est donc un minimum local de $P$, et donc une racine. □

Ainsi, nous avons un moyen de décerner un certificat de culpabilité en temps fini.

**Remarque.** Il n'est pas difficile, en choisissant un réel $R$ assez grand, de montrer qu'en dehors du disque de rayon $R$, le module de $P(z)$ est supérieur à celui de $b_0$. Le théorème implique alors qu'il y a au moins une racine dans ce disque ... et donc que le polynôme a au moins une racine. C'est le théorème de d'Alembert Gauss, aussi connu sous le nom de théorème fondamental de l'algèbre.

## 2.1 En résumé

En jouant seulement avec les inégalités triangulaires, on a montré comment obtenir, pour tout $\varepsilon > 0$, en temps fini, un petit nombre de carrés de côté $< \varepsilon$ qui contiennent tous au moins une racine.