Joli_Resultat_Polynomes

March 27, 2019

1 Un joli résultat sur les polynômes

Marc Lorenzi
26 mars 2019

```
In [1]: import matplotlib.pyplot as plt
        import random
        import numpy.polynomial.polynomial as poly
        import cmath, math

In [3]: plt.rcParams['figure.figsize'] = (10, 10)
```

1.1 1. Introduction

1.1.1 1.1 Qu'allons-nous faire ?

Nous allons dans ce petit notebook prouver le théorème suivant :

Théorème (Gauss-Lucas) : Soit $P \in \mathbb{C}[X]$ un polynôme non constant. Les racines de P' sont dans l'enveloppe convexe de l'ensemble des racines de P.

Nous illustrerons ce théorème avec quelques expériences pythonesques sur des polynômes "aléatoires".

Convention : Dans toute la suite nous identifierons les nombres complexes et les **points** du plan, ainsi que les nombres complexes et les **vecteurs** du plan.

1.1.2 1.2 Droites, demi-plans

Soit $a \in \mathbb{C}$, vu comme un point du plan. Soit $b \in \mathbb{C}^*$, vu comme un vecteur non nul. Les points de la droite $\mathcal{D}$ passant par a et dirigée par b sont les $z \in \mathbb{C}$ de la forme $z = a + bt$ où $t \in \mathbb{R}$. Dit autrement, $z \in \mathcal{D}$ si et seulement si

$$\text{Im}\,\frac{z-a}{b} = 0$$

Orientons la droite $\mathcal{D}$ par le vecteur b. $\mathcal{D}$ partage le plan en deux demi-plans, l'un, H_g, à gauche de $\mathcal{D}$, l'autre, H_d, à droite. À droite ? À gauche ? Oui, grâce à l'orientation que nous avons imposée à $\mathcal{D}$. En nous plaçant sur la droite $\mathcal{D}$ et en regardant dans le sens de b, H_g est à notre gauche et H_d à notre droite. Ces deux demi-plans sont caractérisés par des inégalités :

$$(H_g) \quad \text{Im}\,\frac{z-a}{b} > 0$$

et

$$(H_d) \quad \text{Im}\, \frac{z-a}{b} < 0$$

On définit également les demi-plans **fermés** $\overline{H}_g = H_g \cup \mathcal{D}$ et $\overline{H}_d = H_d \cup \mathcal{D}$. Il suffit de remplacer les inégalités strictes ci-dessus par des inégalités larges.

Exemple : Prenons $a = 0$ et $b = 1$. La droite $\mathcal{D}$ est alors l'axe Ox, orienté dans le sens usuel. H_g est donné par Im $z > 0$, c'est le demi-plan supérieur de $\mathbb{C}$. Et H_d est le demi-plan inférieur.

Exercice : Prenez $a = 0$ et $b = i$. Posez $z = x + iy$, où $x, y \in \mathbb{R}$. À quelle condition sur x et y a-t-on $z \in H_g$?

Exercice : 1. Montrer que pour tous $z, z' \in H_g$, le segment $[z, z']$ est inclus dans H_g. Ce qui montre que H_g est **convexe**. 2. Montrer que pour tout $z \in H_g$ et tout $z' \in H_d$, le segment $[z, z']$ contient un point de $\mathcal{D}$.

Avec un peu de dextérité mathématique on pourrait en fait montrer le résultat numéro 2 ci-dessus en remplaçant "segment $[z, z']$" par "courbe d'origine z et d'extrémité z'". Pour les adeptes de topologie, cela prouve que $\mathbb{C} \setminus \mathcal{D}$ possède deux composantes connexes, qui sont H_g et H_d.

Remarque : On peut aussi faire un dessin et se dire que c'est évident :-).

1.1.3 1.3 Une décomposition en éléments simples

Soi P un polynôme de degré $n \geq 1$. Soient $a_1, a_2, \ldots, a_n$ les racines (éventuellement égales) de P. On a

$$P = c \prod_{k=1}^{n} (X - a_k)$$

où $c \in \mathbb{C}^*$ est le coefficient dominant de P. Dérivons P. On obtient une somme de n termes, chaque terme étant obtenu en dérivant l'un des facteurs de P.

$$P' = c \sum_{j=1}^{n} \prod_{k \neq j} (X - a_k) = \sum_{j=1}^{n} \frac{P}{X - a_j}$$

Ainsi,

$$\frac{P'}{P} = \sum_{j=1}^{n} \frac{1}{X - a_j}$$

Avant de voir le rapport avec notre théorème, faisons quelques expériences en Python.

1.2 2. Racines d'un polynôme et de sa dérivée

1.2.1 2.1 `numpy` et les polynômes

Le module `numpy.polynomial.polynomial` contient un certain nombre de fonctions permettant de manipuler des polynômes. Un polynôme $P = \sum_{k=0}^{n} a_k X^k$ y est représenté par la liste $[a_0, \ldots, a_n]$ (ou, de façon équivalente, un tableau `numpy`).

1.2.2 2.2 Polynômes "aléatoires"

Fabriquons une fonction `random_poly`. Elle prend en paramètre un entier n. Elle calcule n nombres complexes "aléatoires" $z_1, \ldots, z_n$ de parties réelle et imaginaire entre -1 et 1 et renvoie le polynôme

$$P = \prod_{k=1}^{n} (X - z_k)$$

La fonction `poly.polyfromroots` fait le travail à notre place. Elle prend la liste des racines en paramètre et renvoie la liste des coefficients du polynôme.

```
In [4]: def random_poly(n):
            s = [random.uniform(-1, 1) + 1j * random.uniform(-1, 1) for k in range(n)]
            # print(s)
            P = poly.polyfromroots(s)
            return P
```

Un petit exemple ?

```
In [5]: P = random_poly(10)
        print(P)
```

```
[ 0.0103892 -0.031124j    0.03342205+0.03488314j  0.01403988+0.25828334j
 -0.56173901-0.51958942j  0.93999965-0.14020748j  0.07327064+0.79325245j
 -1.24231099+0.42738512j -0.00327285-2.09008694j  2.33838793+1.93202161j
 -2.51670984-0.54272486j  1.        +0.j        ]
```

Remarquez le coefficient dominant égal à 1. C'est normal !

1.2.3 2.3 Représentation graphique d'un polynôme

Comment représenter graphiquement une fonction polynôme $P : \mathbb{C} \to \mathbb{C}$? Comme nous nous intéressons ici aux racines des polynômes, nous allons dessiner $|P|$. La racines de P donnent lieu au minimum, nul, de son module.

Une première possibilité serait une représentation sous forme de surface, mais les résultats obtenus pour les polynômes sont assez décevants. Orientons-nous vers un système de couleurs. Chaque réel positif sera associé à une couleur : il nous suffira donc avec cette méthode de remplir une matrice dont les coefficients sont les valeurs du module de P. Plus précisément, le coefficient ligne i, colonne j de la matrice est $|P(z)|$ où z est le nombre complexe qui "correspond" au couple (i, j). Nos polynômes ayant leurs racines dans le disque de centre O et de rayon 1, nous tracerons P pour des nombres complexes de parties réelle et imaginaire entre -1.2 et 1.2. Il n'est alors pas bien difficile d'établir une correspondance entre le couple (i, j) et z.

La fonction `plot_poly` prend en paramètres

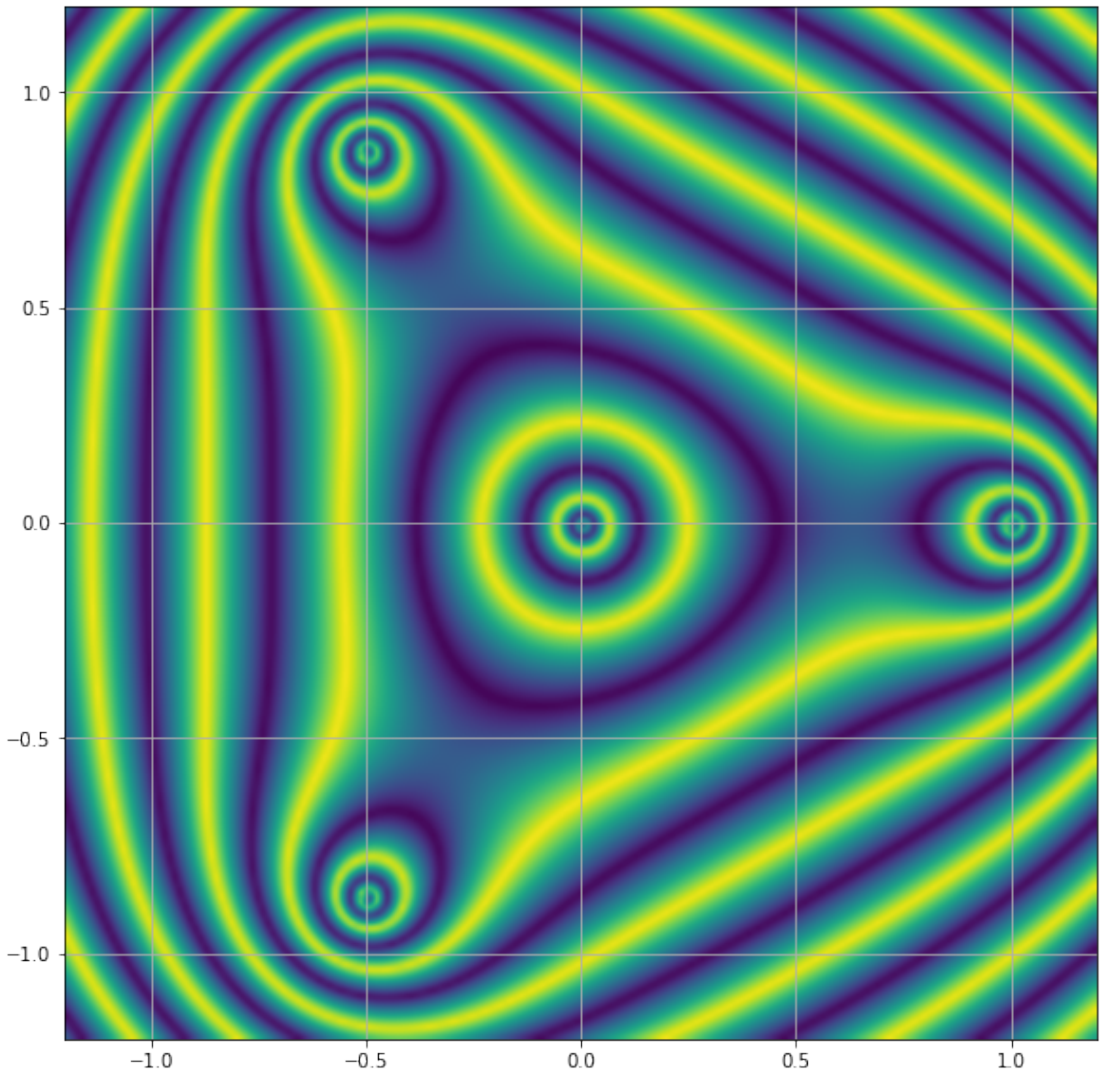
- Un polynôme P donné par la liste de ses coefficients.
- Un entier N qui est le nombre de lignes (et de colonnes) de la matrice à calculer.
- Un réel positif `m`.

La fonction renvoie les valeurs de $z \mapsto |P(z)|^{1/n} \bmod m$, où n est le degré de P, dans une matrice. Pourquoi ce paramètre m ? Plus m est petit, plus les couleurs varieront vite en fonction de z. Une petite valeur de m permet donc d'augmenter le contraste au voisinage des points où le polynôme varie peu. Pour ce que nous voulons observer, $m = 0.1$ est un bon compromis.

```
In [6]: def plot_poly(P, N=200, m=0.1):
            e = 1 / (len(P) - 1)
            z = [[0 for i in range(N)] for j in range(N)]
            for i in range(N):
                for j in range(N):
                    x = -1.2 + j * 2.4 / N
                    y = 1.2 - i * 2.4 / N
                    w = abs(poly.polyval(x + 1j * y, P)) ** e
                    if w % (2 * m) <= m: z[i][j] = (w % m) / m
                    else: z[i][j] = 1 - (w % m) / m
            plt.imshow(z, cmap='viridis', interpolation='bicubic', extent=[-1.2, 1.2, -1.2, 1.2]
```

Voici un exemple sur un polynôme dont on connaît bien les racines : $P = X(X^3 - 1)$.

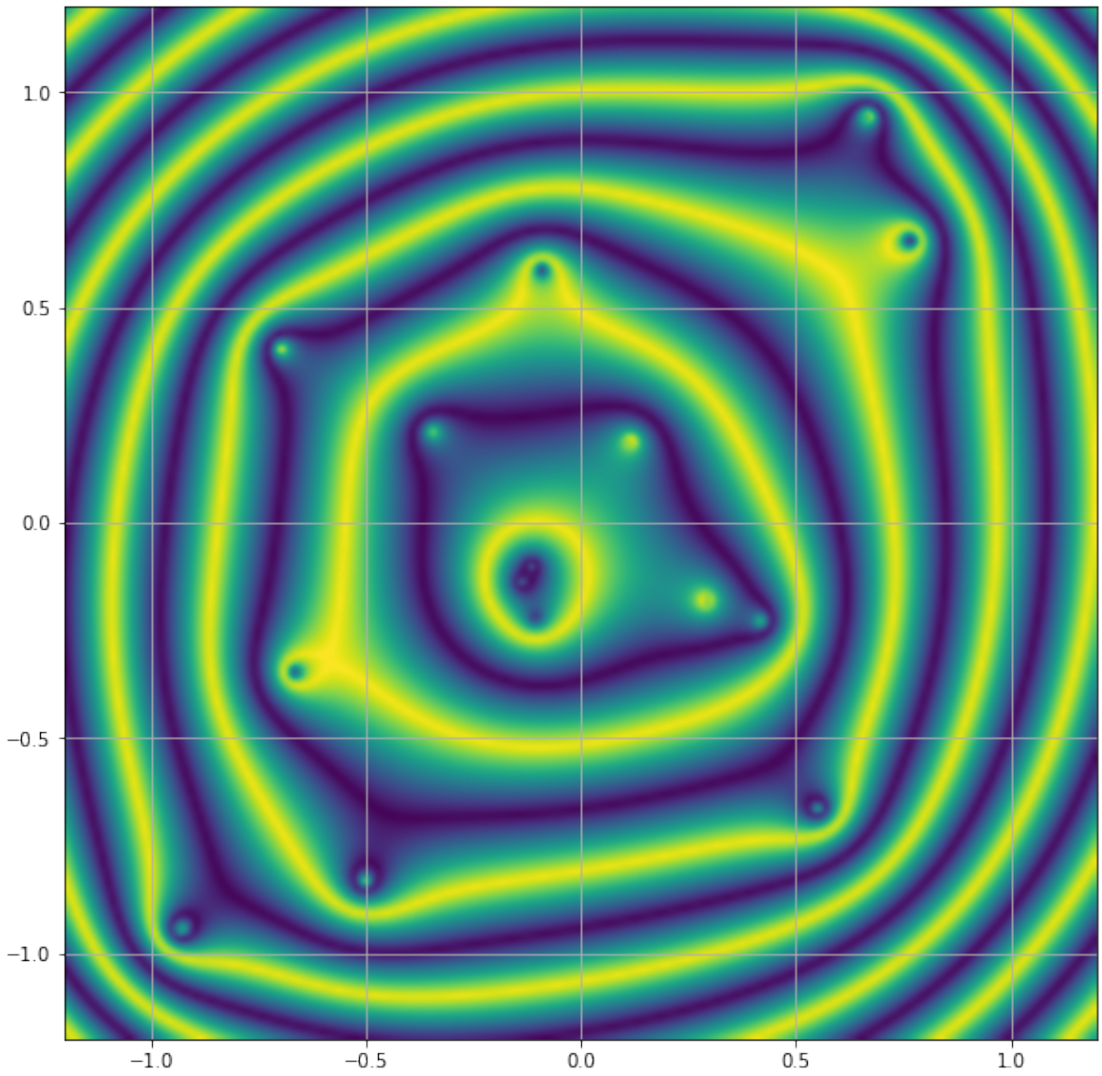
```
In [7]: j = cmath.exp(2 * 1j * math.pi / 3)
        P = poly.polyfromroots([0, 1, j, j ** 2])
        plot_poly(P)
        plt.grid()
```



Et un test sur un polynôme aléatoire.

```
In [8]: P = random_poly(15)

In [9]: plot_poly(P, m=0.1)
        plt.grid()
```



Exercice : Trouvez sur le dessin les racines de P.
Exercice : Changez ci-dessus la valeur de m pour voir ce qui se passe.

1.2.4 2.4 Racines de P

Calculons idiotement les racines de P. La fonction `poly.polyroots` est là pour ça.

Exercice : Pourquoi idiotement ? *Parce que nous avons calculé P à partir de ses racines !* Décommentez la ligne `print(s)` dans `random_poly` pour constater que `polyroots` fait bien son travail. Puis re-commentez-la.

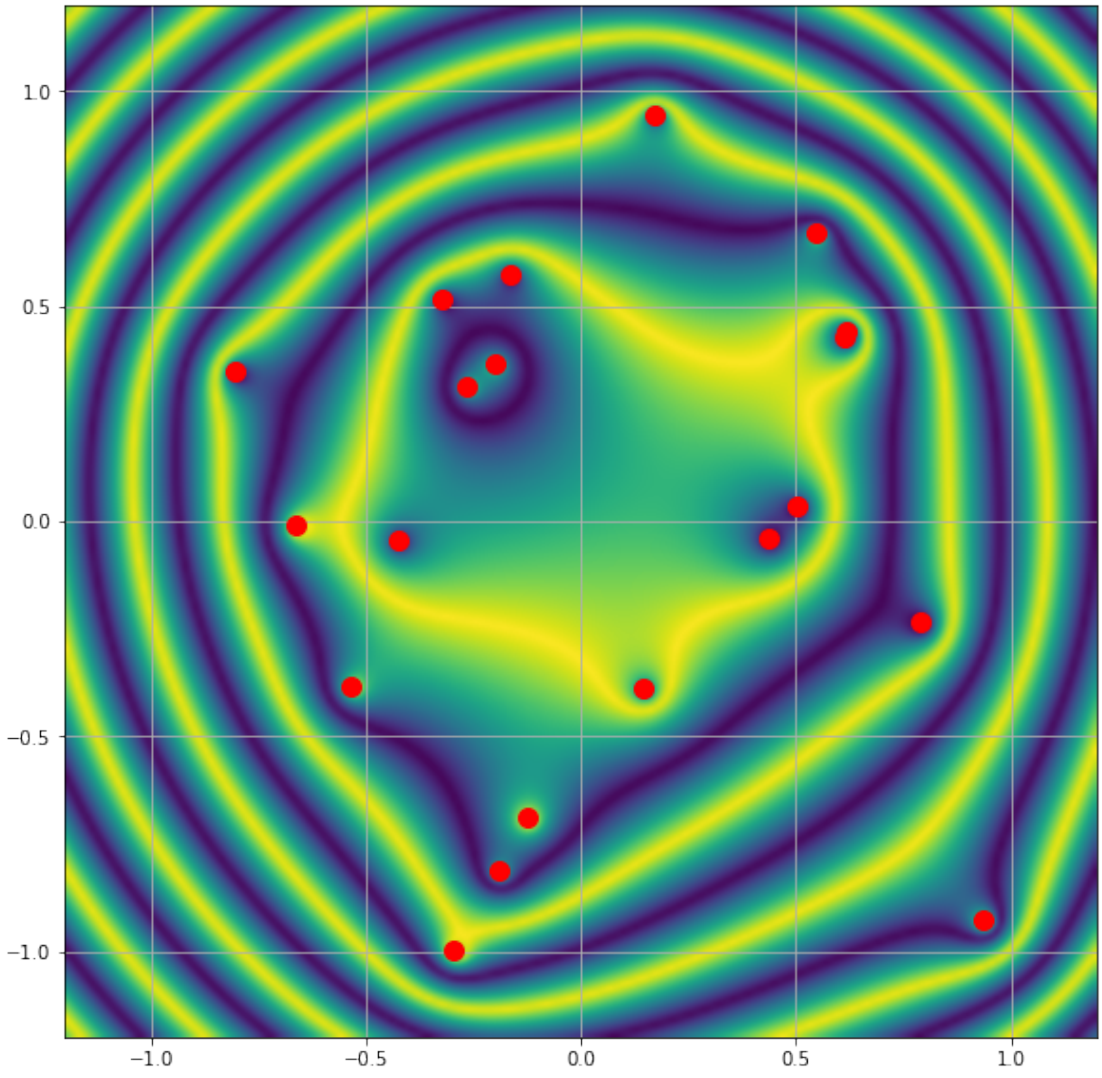
```
In [10]: rs = poly.polyroots(P)
         print(rs)
```

```
[-0.93465393-0.9380537j  -0.70171234+0.41031003j -0.67017259-0.34137805j
 -0.50664919-0.82549425j -0.34962018+0.21771449j -0.1408858 -0.13309518j
```

```
-0.11808451-0.09265996j -0.11020912-0.21573641j -0.09547809+0.59382895j
 0.11242233+0.19580826j  0.28523151-0.17460226j  0.41404599-0.22295889j
 0.5461459 -0.65825474j  0.66348648+0.95072398j  0.76090686+0.6618223j ]
```

Et dessinons les racines par dessus le tracé du polynôme.

```
In [11]: P = random_poly(20)
         plot_poly(P, m=0.1)
         rs = poly.polyroots(P)
         xs = [complex(z).real for z in rs]
         ys = [complex(z).imag for z in rs]
         plt.plot(xs, ys, 'or', markersize=10)
         plt.grid()
```



Exercice : Le tracé des cercles sur les racines est presque superflu. Commentez la ligne correspondante, vous verrez les racines tout aussi bien !

1.2.5 2.5 P' et ses racines

Dérivons maintenant P grâce à la fonction `numpy.polyder`.

```
In [12]: P1 = poly.polyder(P)
         print(P1)
```

```
[ 7.66853976e-04-5.25108334e-04j -3.75489655e-04+3.52618104e-03j
 -9.08162670e-03+1.54794407e-03j  2.71212204e-02+9.21738163e-03j
 -1.67594409e-02+1.18631747e-01j -2.42067043e-01-8.88812334e-02j
 -2.71252619e-01-3.54890313e-01j  1.23292105e-01-1.02418861e+00j
  1.86029047e+00-7.18976667e-01j  3.81328761e+00+1.33632262e+00j
  5.29286423e-01+4.94136235e+00j -5.35579552e+00+3.48767851e+00j
 -4.59666546e+00-3.09260517e+00j -2.97289192e+00-3.63118644e+00j
  9.64379635e+00-1.56706553e+01j  9.65744262e+00+1.06195650e+01j
 -2.62663868e+00-1.52775667e+00j  1.44883126e+01+1.12582121e+00j
 -1.42902836e+01-2.16441299e+00j  2.00000000e+01+0.00000000e+00j]
```

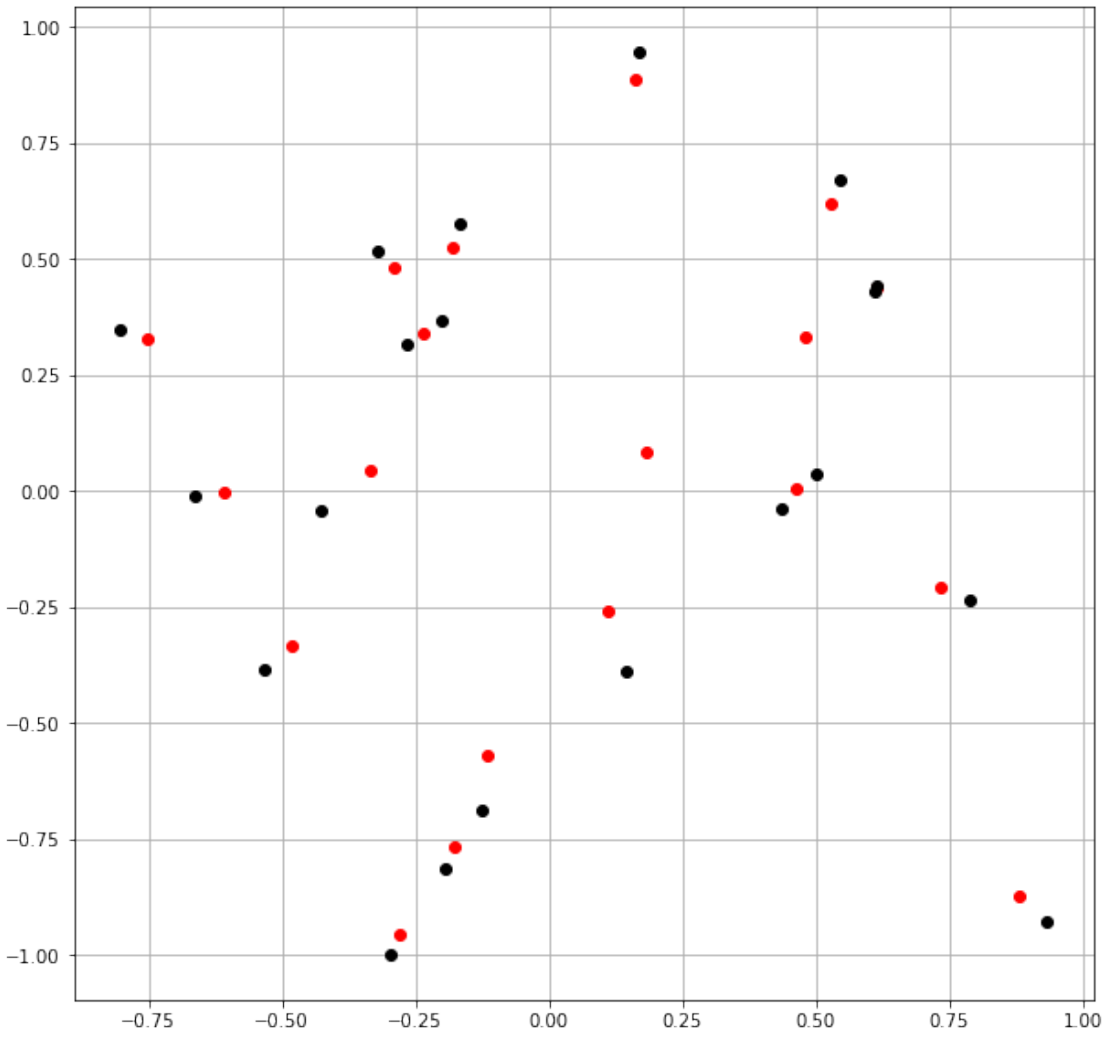
Remarquez le coefficient dominant égal au degré de P :-).
Calculons illico les racines de P'.

```
In [13]: rs1 = poly.polyroots(P1)
         print(rs1)
```

```
[-0.75130308+0.32685191j -0.60639046-0.00457066j -0.48175175-0.33315898j
 -0.33526722+0.04239523j -0.29062012+0.48290662j -0.27821381-0.95531174j
 -0.23418493+0.339036j   -0.17881162+0.52416114j -0.17613731-0.76611273j
 -0.11390086-0.56944903j  0.11136612-0.26026969j  0.16314971+0.88508501j
  0.18360592+0.08446391j  0.46345102+0.00587454j  0.48022124+0.33241406j
  0.52889115+0.61842998j  0.61341748+0.43589938j  0.73497858-0.20827672j
  0.88201415-0.87214758j]
```

Enfin, affichons le tout ! En rouge, les racines de P'. En noir, les racines de P.

```
In [14]: xs1 = [complex(z).real for z in rs1]
         ys1 = [complex(z).imag for z in rs1]
         plt.plot(xs1, ys1, 'or')
         plt.plot(xs, ys, 'ok')
         plt.grid()
```



1.2.6 2.6 Combinons le tout

Regroupons le tout dans une fonction `test_theoreme`. Les racines de P ne sont pas affichées. Les racines de P' sont marquées par des cercles rouges et blancs.

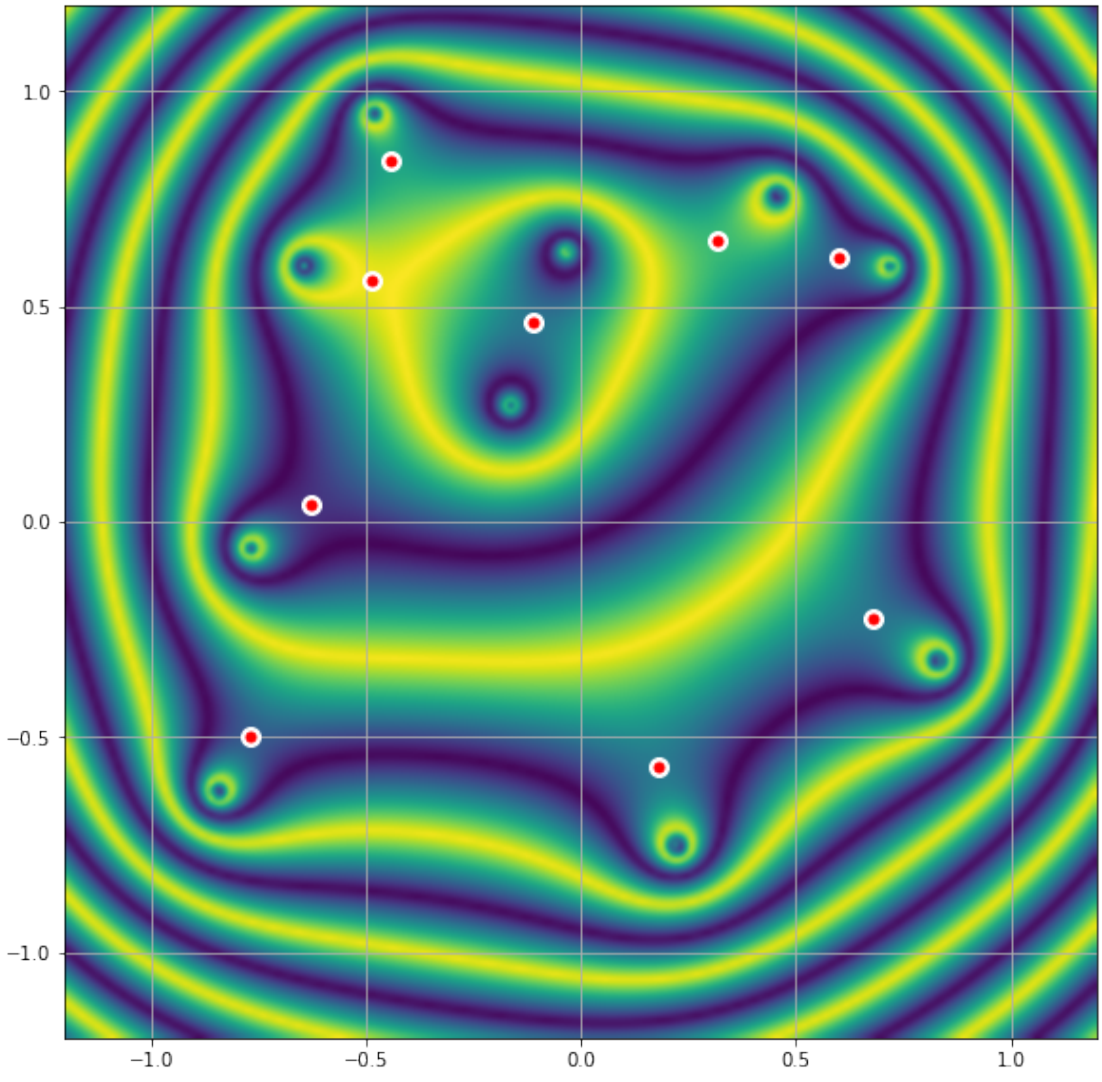
```
In [15]: def test_theoreme(P, **opt):
             plot_poly(P, **opt)
             #rs = poly.polyroots(P)
             #xs = [complex(z).real for z in rs]
             #ys = [complex(z).imag for z in rs]
             #plt.plot(xs, ys, 'or', markersize=10)
             P1 = poly.polyder(P)
             rs1 = poly.polyroots(P1)
             xs1 = [complex(z).real for z in rs1]
             ys1 = [complex(z).imag for z in rs1]
```

```
    plt.plot(xs1, ys1, 'ow', markersize=10)
    plt.plot(xs1, ys1, 'or', markersize=5)
    plt.grid()
```

Vous pouvez maintenant tester. N'abusez pas sur le degré du polynôme : pour cause d'erreurs d'arrondi et de dépassement de capacité des flottants, `numpy` renvoie des résultats erronés si le degré du polynôme est trop grand (essayez 200, juste pour rire).

```
In [24]: P = random_poly(10)
         #P = poly.polyfromroots([0, 1, -0.5 + math.sqrt(3)/2 * 1j, -0.5 - math.sqrt(3)/2 * 1j,

In [25]: test_theoreme(P, m=0.1)
```



Il est intéressant de se poser la question : Et alors ? Que voit-on ?
En fait on voit bien des choses mais nous allons nous fixer sur l'une d'entre-elles ...

Si vous tracez une droite, celle qui vous fait plaisir, et que tous les points noirs sont d'un côté de la droite, alors tous les points rouges sont aussi du même côté. En d'autres termes, tout demi-plan contenant les racines de P contient aussi les racines de P'.

Si nous notons $\mathcal{C}(P)$ l'intersection de **tous** les demi-plans contenant les racines de P, alors les racines de P' appartiennent à $\mathcal{C}(P)$.

On peut montrer que $\mathcal{C}(P)$ est une partie convexe du plan (ça c'est facile), et que c'est la plus petite partie convexe du plan contenant les racines de P (c'est un peu moins facile). C'est ce que l'on appelle **l'enveloppe convexe** de l'ensemble des racines. Encore mieux, $\mathcal{C}(P)$ est un polygone convexe dont les sommets sont des racines de P.

NB : Si vous voulez plus de détails sur le sujet des enveloppes convexes, consultez le notebook qui lui est consacré. Un algorithme permettant le calcul efficace de l'enveloppe convexe s'appelle **l'algorithme de Graham**.

Exercice : Réévaluez la cellule ci-dessus avec un polynôme unitaire de degré 2 ayant deux racines distinctes. Où est l'unique racine de P' ? Prouvez-le, c'est facile.

Exercice : Réévaluez la cellule ci-dessus avec un polynôme unitaire de degré 3 ayant trois racines distinctes et tel que P' n'ait qu'une racine, double. Où est l'unique racine de P' ? Prouvez-le aussi. Indication : $P' = 3(X-a)^2$, cela ne devrait pas être difficile de trouver P et ses racines.

1.3 3. Fin de la démonstration

Soit P un polynôme de degré $n \geq 1$ dont les racines (éventuellement égales) sont $a_1, a_2, \ldots, a_n$.

Soient $a \in \mathbb{C}$ et $b \in \mathbb{C}^*$. Soit $\overline{H}$ le demi-plan fermé défini par

$$(\overline{H}) \quad \text{Im } \frac{z-a}{b} \geq 0$$

Soit H' son complémentaire. C'est le demi-plan ouvert défini par

$$(H') \quad \text{Im } \frac{z-a}{b} < 0$$

Supposons que $\overline{H}$ contient toutes les racines du polynôme P. On a donc pour tout k entre 1 et n

$$\text{Im } \frac{a_k - a}{b} \geq 0$$

Soit $z \in H'$. On a donc Im $\frac{z-a}{b} < 0$. Pour tout k entre 1 et n, on a

$$\text{Im } \frac{z-a_k}{b} = \text{Im } (\frac{z-a}{b} - \frac{a_k - a}{b}) = \text{Im } \frac{z-a}{b} - \text{Im } \frac{a_k - a}{b} < 0$$

Le passage à l'inverse change le signe d'un nombre complexe. On a donc

$$\text{Im } \frac{b}{z-a_k} > 0$$

Rappelons-nous maintenant notre décomposition en éléments simples :

$$\frac{P'}{P} = \sum_{k=1}^{n} \frac{1}{X - a_k}$$

On en déduit

$$b\frac{P'(z)}{P(z)} = \sum_{k=1}^{n} \frac{b}{z - a_k}$$

D'où, en prenant la partie imaginaire :

$$\text{Im } \left(b\frac{P'(z)}{P(z)} \right) = \sum_{k=1}^{n} \text{Im } \frac{b}{z - a_k} > 0$$

La partie imaginaire de $b\frac{P'(z)}{P(z)}$ est non nulle, donc $P'(z) \neq 0$. Contraposons :

Si $P'(z) = 0$, alors $z \notin H'$, donc $z \in \overline{H}$. Et ceci est vrai pour **tout** demi-plan (fermé) contenant les racines de P.

Notre théorème est démontré.

In [ ]: