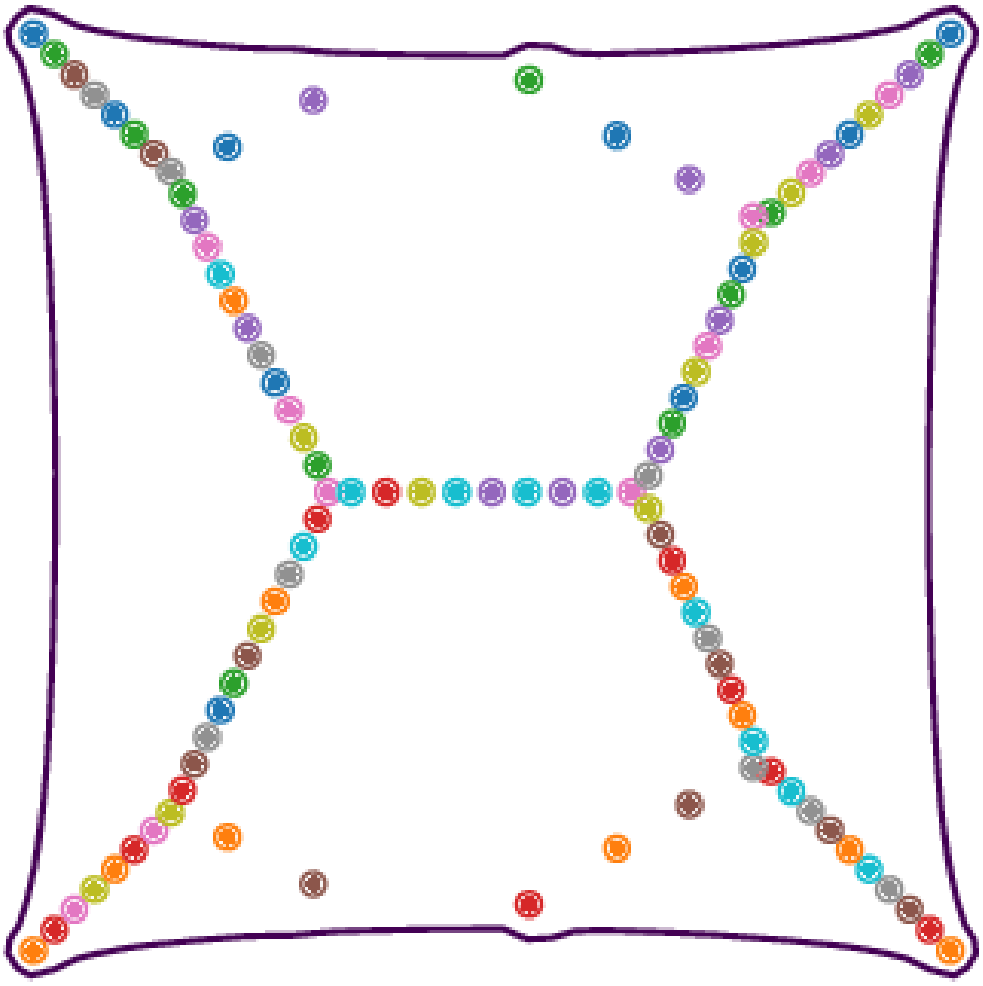
POLYTECH
PARIS-UPMC

Rapport
Projet pluridisciplinaire



PseudoSpectres

Encadré par : Stef Graillat
Coordinateur : Thibaut Hilaire

Membres du groupe :
Inès Benzenati
Alexia Zounias-Sirabella
Fatine Bentires Alj

Table des matières

Introduction

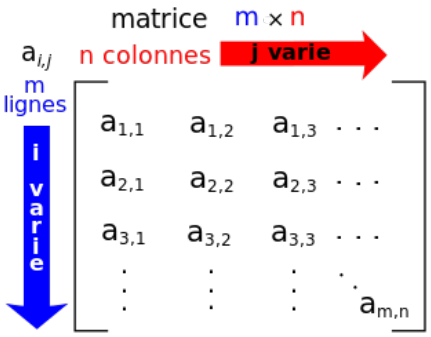
Les matrices sont des outils très puissants avec de nombreuses applications en mécanique, calcul des résistances des matériaux, changement de base...etc
On appelle valeur propre d'une matrice un scalaire λ vérifiant $\mathbf{Av}=\lambda\mathbf{v}$ avec $\mathbf{v}\neq\mathbf{0}$ et spectre d'une matrice l'ensemble de ses valeurs propres. Les valeurs propres d'une matrice sont de loin les outils les plus utilisés en mathématiques appliquées et divers domaines scientifiques font appels à ces dernières. On les retrouve facilement dans la résolution d'équations différentielles ou en spectroscopie. Leurs applications ne s'arrêtent pas là puisqu'elles ont une utilité dans des domaines où nous n'en voyons pas aux premiers abords comme en musique ou en économie.
Un problème se pose cependant. Il a été montré que les matrices ont souvent des erreurs au niveau de leurs valeurs propres qui font que certains résultats en sont faussés. Ceci a entraîné l'introduction d'une nouvelle notion, celle de pseudospectre. Le pseudospectre correspond au spectre des matrices qui ont subit une perturbation. L'objectif de notre projet est d'étudier ces pseudospectres afin de se rapprocher des vraies valeurs propres.
Deux techniques existent déjà à ce propos, la méthode grid et celle de prédiction-correction. Nous allons étudier ces deux méthodes, les implémenter en Matlab et en Python et enfin les comparer pour voir laquelle est la plus efficace.
En particulier, nous appliquerons ces programmes à une matrice fixée afin d'avoir une meilleure compréhension du problème. Nous commencerons ainsi par expliquer dans un premier temps les notions théoriques rattachées au pseudospectre puis nous décrirons dans un second temps les méthodes utilisées.

Première partie

Étude théorique des pseudospectres

1 Introduction aux notions de matrice

Dans un premier temps il est important d'introduire quelques définitions. Commençons par définir proprement ce qu'est une matrice. Une matrice peut être vue comme un tableau à $m \times n$ entrées où m représente les lignes et n les colonnes.



Cette matrice va être constituée de coefficients que l'on va noter **a** et indexer d'indices m et n représentant le numéro de la ligne et de la colonne où se trouve cet indice. Par exemple $a_{1,2}$ va correspondre au coefficient se trouvant à la 1ère ligne et à la 2 eme colonne. Mais ce qui nous intéresse réellement dans ce projet ce sont les valeurs propres de la matrice.

1.1 Valeurs propres

1.1.1 Principe

On peut donner deux grandes définitions des valeurs propres. La première donne une relation permettant l'obtention de chaque valeur propre de manière isolée et la seconde une relation où la une matrice est constituée de toutes les valeurs propres :

— Soit A une matrice carrée de taille $N \times N$ avec des coefficients réels ou complexes, on note $A \in \mathbb{C}^{\mathbb{N}\times\mathbb{N}}$, soit v un vecteur colonne non nul de taille N, on note v $\in \mathbb{C}^{\mathbb{N}}$, on appelle valeur propre un scalaire $\lambda \in \mathbb{C}$ réel ou complexe vérifiant

$$Av = \lambda v.$$

— Soit V la matrice des vecteurs propres V_j et A une matrice quelconque. Si on note Λ la matrice diagonale des valeurs propres,alors :

$$AV = V\Lambda.$$

1.1.2 Trois principales applications

Les valeurs propres d'une matrice ont bien évidemment énormément d'applications. Cependant, nous pouvons en noter trois principales :

1. L'utilisation des valeurs propres pour la décomposition de problèmes complexes en problèmes plus simples.
2. L'étude des valeurs propres en tant que fréquences de résonance.
3. L'étude de la forme et des caractéristiques d'un objet en fonction de la localisation des valeurs propres dans l'espace.

1.1.3 Exemple de calcul de valeur propre

Déterminer les valeurs propres de la matrice $A = \begin{pmatrix} 5 & -3 \\ 6 & -4 \end{pmatrix}$

Les valeurs propres de A sont les **scalaires** l vérifiant :

$$\det(A - \lambda I_2) = 0 \quad \Leftrightarrow \begin{vmatrix} 5-\lambda & -3 \\ 6 & -4-\lambda \end{vmatrix}$$
$$= -(5-\lambda)(4+\lambda) + 18$$
$$= \lambda^2 - \lambda - 2$$
$$= (\lambda + 1)(\lambda - 2) = 0$$

d'où les valeurs propres : $l_1 = -1$ et $l_2 = +2$

1.2 Valeurs singulières

Soit une matrice $M \in \mathbb{R}^{m \times n}$, on appelle valeur singulière toute racine carrée des valeurs propres de la matrice $M^* \times M$, où M^* est la transconjuguée de la matrice M, elles sont répertoriées en ordre décroissant, l'ensemble forme le spectre qui est noté σ . Ainsi pour tout vecteur unitaire $u \neq 0$ et $v \neq 0$, on a

$$Mv = \sigma u$$

et

$$M^* u = \sigma v.$$

1.3 Spectre d'une matrice

1.3.1 Qu'est-ce qu'un spectre ?

Comme nous l'avons dit en introduction, le spectre d'une matrice A correspond à l'ensemble des valeurs propres de cette matrice. Il est noté $\sigma(A)$. Une autre propriété tout aussi intéressante indique que le spectre d'une matrice A correspond à l'ensemble des $z \in \mathbb{C}$ telle que la matrice $(zI - A)^{-1}$ n'existe pas, c'est-à-dire, telle que la matrice $(zI - A)$ ne soit pas inversible.

1.3.2 Limites

Contrairement à ce que l'on pourrait penser, la recherche des valeurs propres d'une matrice n'est pas forcément évidente. Il a été montré qu'une matrice normale (une matrice qui commute avec son adjoint) a des approximations proches des valeurs réelles. Cependant, pour les matrices non normales, le problème se complexifie. Il en résulte des différences non négligeables entre valeur réelle et valeur approchée. Le spectre d'une matrice ne suffirait donc plus pour son étude... Pour palier au problème, la notion de pseudospectre a été introduite. Nous y reviendrons un peu plus tard.

1.4 Les vecteurs propres

1.4.1 Principe

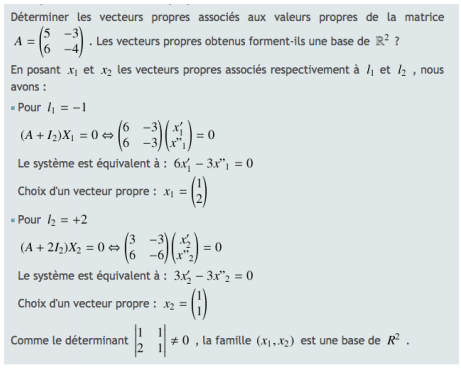
On note $v \neq 0$ un vecteur propre associé à une valeur propre λ tel que

$$Av = \lambda v.$$

Les vecteurs propres d'une application linéaire f correspondent aux axes privilégiés selon lesquels l'application se comporte comme une dilatation, multipliant les vecteurs par une même constante. Ce rapport de dilatation est appelée valeur propre.
En considérant une matrice comme une matrice de transformation, ses vecteurs propres sont des vecteurs dont la direction n'est pas affectée par cette transformation,et satisfait l'équation suivante[1] :

$$f(v) = \lambda v.$$

1.4.2 Exemple de calcul de vecteur propre


Déterminer les vecteurs propres associés aux valeurs propres de la matrice $A = \begin{pmatrix} 5 & -3 \\ 6 & -4 \end{pmatrix}$. Les vecteurs propres obtenus forment-ils une base de $\mathbb{R}^2$?

En posant x_1 et x_2 les vecteurs propres associés respectivement à l_1 et l_2, nous avons :

- Pour $l_1 = -1$

$(A + I_2)X_1 = 0 \Leftrightarrow \begin{pmatrix} 6 & -3 \\ 6 & -3 \end{pmatrix}\begin{pmatrix} x'_1 \\ x''_1 \end{pmatrix} = 0$

Le système est équivalent à : $6x'_1 - 3x''_1 = 0$

Choix d'un vecteur propre : $x_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

- Pour $l_2 = +2$

$(A + 2I_2)X_2 = 0 \Leftrightarrow \begin{pmatrix} 3 & -3 \\ 6 & -6 \end{pmatrix}\begin{pmatrix} x'_2 \\ x''_2 \end{pmatrix} = 0$

Le système est équivalent à : $3x'_2 - 3x''_2 = 0$

Choix d'un vecteur propre : $x_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Comme le déterminant $\begin{vmatrix} 1 & 1 \\ 2 & 1 \end{vmatrix} \neq 0$, la famille (x_1, x_2) est une base de R^2.

2 Pseudospectre d'une matrice

2.1 Choix de la norme

Le pseudospectre d'une matrice noté $\sigma_\epsilon(A)$ correspond à l'ensemble des valeurs propres de toutes les matrices au voisinage d'une matrice A pour une norme donnée. [2] On comprend ainsi que le spectre d'une matrice sera contenu dans son ϵ-pseudospectre pour tout $\epsilon > 0$. Contrairement au spectre, le pseudospectre dépend d'une norme. Commençons donc par définir ce qu'est qu'une norme.

Soit $A, B \in M_{n\times n}(\mathbb{R})$ deux matrices réelles, l'application $\|.\|$ est une norme matricielle si elle satisfait :

1. $\|A\| \geq 0, \forall\ A \in A_{m\times n}$ et $\|A\| = 0 \leftrightarrow A = 0$
2. $\|\beta A\| = \mid \beta \mid \|\beta A\|\ \forall A$ et $B \in M_{n\times n}$
3. $\|A + B\| \leq \|A\| + \|B\|\ \forall A$ et $B \in M_{n\times n}$
4. $\|AB\| \leq \|A\|\ \|B\|\ \forall A$ et $B \in M_{n\times n}$

Dans la suite du projet nous noterons $\|.\|$ la norme subordonnée issue de $\|.\|_2$ définie par

$$\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

2.2 Définition de la notion de pseudospectre

Un pseudospectre peut être définit de quatre manières principales :

1. Soit A $\in \mathbb{C}^{\mathbb{N}\times\mathbb{N}}$ et $\epsilon > 0$. Le ϵ-pseudospectre $\sigma(A)$ est l'ensemble des z $\in \mathbb{C}$ tels que : $\|(z-A)^{-1}\| < \epsilon^{-1}$
2. $\sigma_\epsilon(A)$ est l'ensemble des $z \in \mathbb{C}$ tels que : $z \in \sigma(A+E)$ pour un certain $E \in \mathbb{C}^{\mathbb{N}\times\mathbb{N}}$ avec $\|E\| < \epsilon$
3. $\sigma_\epsilon(A)$ est l'ensemble des $z \in \mathbb{C}$ tel que $\|(z-A)v\| > \epsilon$ pour $v \in \mathbb{C}^{\mathbb{N}}$ avec $\|v\| = 1$.
4. Pour $\|.\| = \|.\|_2$, $\sigma_\epsilon(A)$ est l'ensemble des $z \in \mathbb{C}$ tel que $\sigma_{min}(z-A) < \epsilon$

avec z une ϵ-pseudo valeur propre et v un ϵ-pseudo vecteur propre de A. En d'autres mots, un ϵ-pseudospectre est donc l'ensemble des ϵ-pseudo valeurs propres. On retiendra que les trois premières définitions sont équivalentes.

Deuxième partie

Algorithmique

3 Décomposition en valeurs singulières

La décomposition en valeur singulière est une méthode de factorisation d'une matrice M de taille $m \times n$ sous la forme d'un produit $M = U\Sigma V^*$ où les matrices U $m \times m$ et V^* $n \times n$sont unitaire et où Σ est une matrice diagonale avec coefficients réels et positifs. [3]

Exemple :

$$U = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2,236 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad V^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0,447 & 0 & 0 & 0 & 0,894 \\ 0 & 0 & 0 & 1 & 0 \\ -0,894 & 0 & 0 & 0 & 0,447 \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}$$

Les coefficients diagonaux de la matrice Σ correspondent aux valeurs singulières de la matrice M. Cet algorithme basique est loin d'être optimal mais permet de représenter $\sigma_{min}(z - M)$ dans le plan complexe.

4 GRID

4.1 Méthode de Gershgorin appliquée au pseudospectre

Notre but va être dans un premier temps de se placer dans le bon "environnement". Pour cela il va falloir que l'on délimite le plan complexe. Ceci va être possible grâce à la méthode Gershgorin. Le principe est plutôt simple à comprendre. Il nous faut construire une sphère qui va borner les valeurs propres $V_p(A)$ d'une matrice A telle que

$$\lambda \in V_p(A) \rightarrow \mid \lambda - a_{i,i} \mid \; < \sum_{j=1}^{n} \mid a_{i,j} \mid .$$

N'ayant pas accès au spectre mais seulement au pseudospectre, nous avons cherché à redéfinir cette formule et nous avons obtenu

$$\lambda \in \Lambda(A) \rightarrow \mid \lambda - a_{i,i} \mid \leq \sum_{j=1}^{n} \mid a_{i,j} \mid + \sqrt{n}.\epsilon.$$

Preuve : voir Annexe

les coordonnées du centre des sphères est la valeurs des coefficients diagonaux de la matrice A et les rayons sont les somme des normes des autres coefficients.

4.2 Pseudo-Algorithme

Entrée : matrice(A)
Sortie : pseudospectre(z)
Pour k allant de 1 à m faire :
    Pour j allant de 1 à m faire :
        sigmin(j,k) ← min(svd((x(k)+y(j)*i)*eye(N)-A))
    end
end
contour(x,y,log10(sigmin))

4.3 Particularité de l'algorithme GRID

En absence d'information, construire un plan complexe peut s'avérer un peu compliqué. En effet, comment savoir où s'arrêter ?
La méthode GRID repose sur l'utilisation d'un intervalle numérique noté w(A) qui va imposer une limite au pseudospectre de manière à ce que ce dernier ne sorte jamais de l'intervalle.

4.4 Implémentation en Matlab

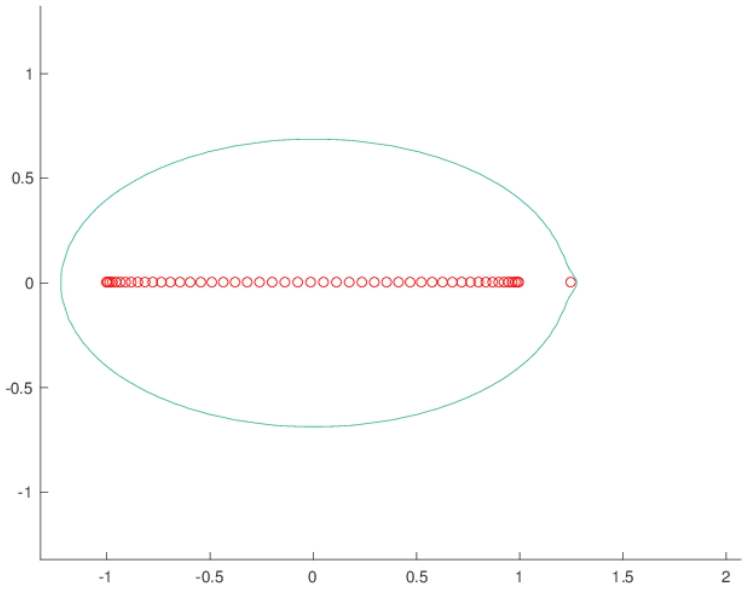
La construction de l'algorithme se fait en plusieurs étapes.

Dans un premier temps, il est nécessaire d'implémenter la fonction de Gershgorin illustrée précédemment en y ajoutant une spécificité. En effet, la méthode de Gershgorin trace un cercle autour des pseudo-valeurs propres. Notre objectif est alors de récupérer les valeurs maximales de l'axe des réels x et des imaginaires y. Pour cela nous avons utilisé le rayon et le centre des cercles.

Une fois les valeurs maximales de x et de y obtenues, il va falloir dans un second temps fractionner ces nombres en intervalles. Ce fractionnement est important puisqu'il va nous permettre de construire nos axes.

La troisième étape correspond bien évidemment à l'utilisation de ces intervalles dans la fonction GRID. Si on se reporte au pseudo-algorithme, $x(k)$ et $y(j)$ seront deux vecteurs contenant les intervalles obtenus précédemment. Ils permettent ainsi la construction de l'axe x des réels et y des imaginaires. Le code est disponible en annexe.

Appliquons maintenant cet algorithme a notre matrice, on peut visualiser les valeurs propres au centre du pseudospectre ce qui confirme les hypothèses précédentes.



4.5 Implémentation en Python

Une fois le programme en Matlab achevé, nous avons décidé de l'implémenter dans un autre langage : Python.

Pour cela, il nous a fallu dans un premier temps nous renseigner sur les multiples bibliothèques existantes (sympy, numpy, pylab, tkinter, matplotlib...etc). Après s'être familiarisé avec les bibliothèques nous sommes passé au code.

Notre code Python est basé sur le même principe que celui en Matlab. Cependant il existe certaines variantes que nous allons détailler. Il est constitué de 4 fonctions principales.

La première est **matrice**. Elle nous donne une matrice aléatoire à chaque tour.

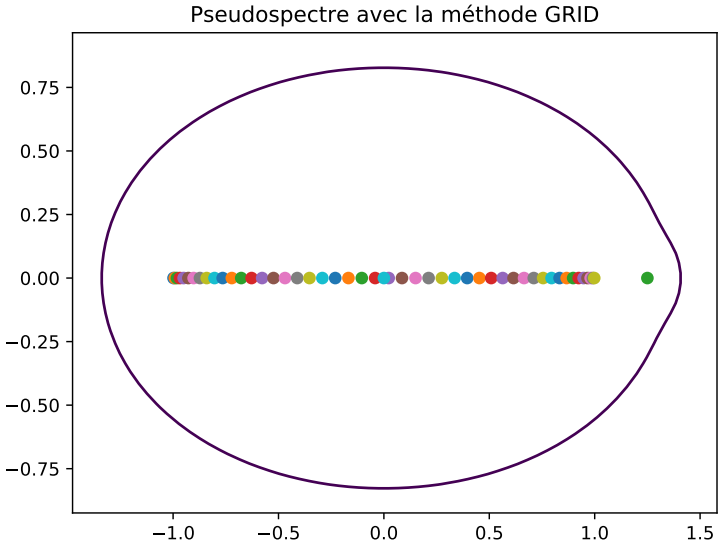
La seconde est **Gershgorin**. C'est une des fonctions les plus importantes. Le principe de fonctionnement est le même qu'en Matlab. On récupère les bornes afin d'utiliser la fonction GRID. Pour cela on identifie les coordonnées des valeurs propres les plus éloignées possibles et on construit deux intervalles qui vont correspondre aux axes des abscisses et des ordonnées.

La troisième fonction qui porte le nom de **minou** n'est pas indispensable puisqu'elle existe en Python mais nous voulions l'implémenter afin d'avoir une idée plus claire du fonctionnement. Elle donne la plus grande valeur.

Enfin, la fonction **gridou** correspond à la fonction grid. Elle prend en entrée les bornes X, Y données par la fonction Gershgorin, la matrice M, les valeurs propres ainsi qu'epsilon et le pas qui correspond au nombre de points. Elle fait appel à la fonction svd qui correspond à la décomposition en valeurs singulières puis dessine le pseudospectre grâce à la fonction contour et les valeurs propres grâce à la fonction plot.

Le code en annexe possède quelques éléments supplémentaires que nous n'avons pas détaillé. Ceci est du au fait qu'ils visent à l'interaction avec l'interface que nous allons introduire par la suite.

Tout comme en Matlab, voici un graphique représentant le pseudospectre et ses valeurs propres implémentés en Python.



5 Prédiction-correction

5.1 Notion de prédiction-correction

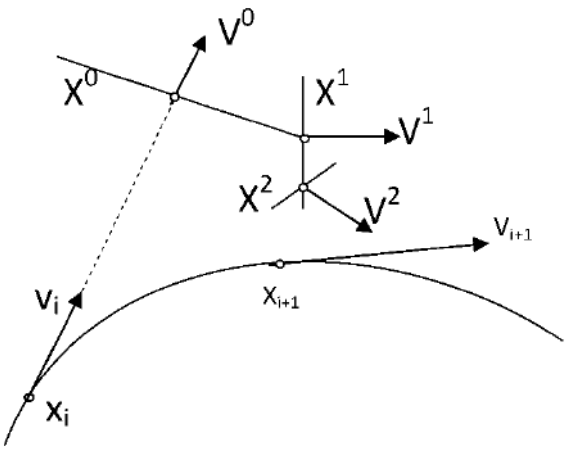
L'algorithme GRID est très intéressant puisqu'il est simple à appliquer et très efficace. Cependant, il est également très coûteux et ne peut pas s'appliquer aux matrices de très grande taille. L'algorithme de prédiction-correction pallie au problème.

Notons le pseudospectre d'une matrice. L'algorithme de prédiction-correction se décompose en trois étapes [4]
- Sélection d'un point z_1 sur le pseudospectre
- Prédiction
- Correction

Dans un premier temps il faut sélectionner une première pseudo-valeur propre z_1 qui va nous permettre de déterminer les pseudo-valeurs propres restantes par une méthode de récurrence. Nous choisissons alors un z_1 qui va correspondre à la valeur numérique d'une valeur propre à laquelle on ajoute ϵ.

Dans un second temps, nous déterminons une direction $r_k \in \mathbb{C}$ tel que $\mid r_k \mid = 1$ et $z_k^{'} = z_k + \tau_k r_k$. Il va falloir prédire la trajectoire au fur et à mesure de manière à obtenir une bonne approximation. Il s'agit de l'étape dite de prédiction.

Enfin, il va falloir corriger la trajectoire obtenue. On utilise alors $z_k = z_k^{'} + \theta_k d_k$ où θ doit être un angle adapté. Le schéma peut être visualisé de la sorte [5]



5.2 Particularité de l'algorithme

L'algorithme repose sur la méthode de Newton. Le principe est le suivant.

L'objectif est de résoudre l'équation $f(x^*) = 0$ avec $x^* \in \mathbb{R}^{\mathbb{N}}$. En général il n'est pas possible de donner la solution exacte c'est pourquoi il faut la calculer de manière approchée. La méthode de Newton a pour but de créer une bonne approximation d'un zéro d'une fonction en considérant son développement de Taylor au premier ordre.

En dimension N=1, la méthode de Newton consiste à construire la suite par la relation de récurrence suivante : pour $x^{(0)}$ donné, on a

$x^{(n+1)}$=$x^{(n)}$ - $\frac{f(x^{(n)})}{f'(x^{(n)})}$ $\forall$ n $\geq$ 0.

Dans notre cas la faction est $f(z) = \sigma_{min}(zI - A) - \epsilon$ ce qui nous donne :

$z_{new} = z_{old} - \frac{\sigma_{min} - \epsilon}{u^*_{min} v_{min}}$

Graphiquement, cela consisterait au tracé de la tangente en chaque point de manière à obtenir un cercle de points qui va correspondre au pseudospectre.

5.3 Pseudo-Algorithme

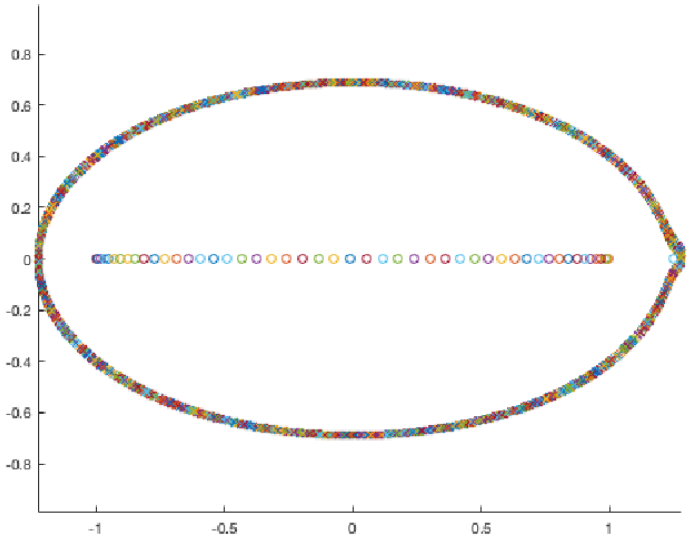
Entrée : matrice(A),ϵ,tolérance tol, nombre d'itérations maximal N
Pour chaque valeur propre de A faire :
    Initialiser z_0 : $z_0 \leftarrow$ vp + ϵ
    Tant que ($\sigma_{min}(z_0$*I-A)-$\epsilon)/\epsilon >$ tol faire :
        Corriger z_0
    Fin Tant que
    Tant que le nombre d'itérations i <N et $\mid z_i - z_0 \mid > 0.001 * z_0$ faire :
        Déterminer la direction r et la longueur du pas τ
        $z_i \leftarrow z_{i-1} + \tau$*r
        Corriger z_i
    Fin Tant que
Fin Pour
Sortie : pseudospectre(z)

5.4 Implémentation en Matlab

Pour implémenter l'algorithme de prédiction-correction en matlab nous avons utilisé la fonction Gershgorin modélisée précédemment. En effet, elle nous donne les bornes permettant la construction de la grille. La fonction que nous allons considérer dans cette partie est **predcor(A,epsi,tol,machin)**. Son fonctionnement est le suivant :

Tout d'abord, nous déterminons $z[0]$ le premier point du pseudospectre. Il s'agit de choisir une valeur propre de départ et de lui ajouter une erreur ϵ. Nous déterminons ensuite une erreur $\frac{\sigma_{min}(zI-A)-\epsilon}{\epsilon}$. Nous effectuons le même procédé pour réinitialiser $z[0]$ tant que les valeurs de l'erreur sont supérieures à tol. Tol correspond à une tolérance que nous nous imposons. Au delà de tol nous ne considérerons plus les valeurs propres. Puis, nous définissons une deuxième erreur relative entre deux points consécutif $\frac{|z[k]-z[0]|}{z[0]}$ qui arrête la boucle si la distance entre le $z[0]$ de départ et un nouveau $z[k]$ est inférieur à 0.01. Ensuit nous calculons les rk qui correspondent à la direction de la trajectoire et enfin nous déterminons les vraies valeurs de z à l'aide de τ_k qui est la longueur du pas, grâce a la formule qui suit : $z[k] = z[k-1] + (\tau_k rk)$.

La représentation suivante illustre les résultats obtenus avec la méthode de prédiction-correction implémentée en Matlab.



5.5 Implémentation en Python

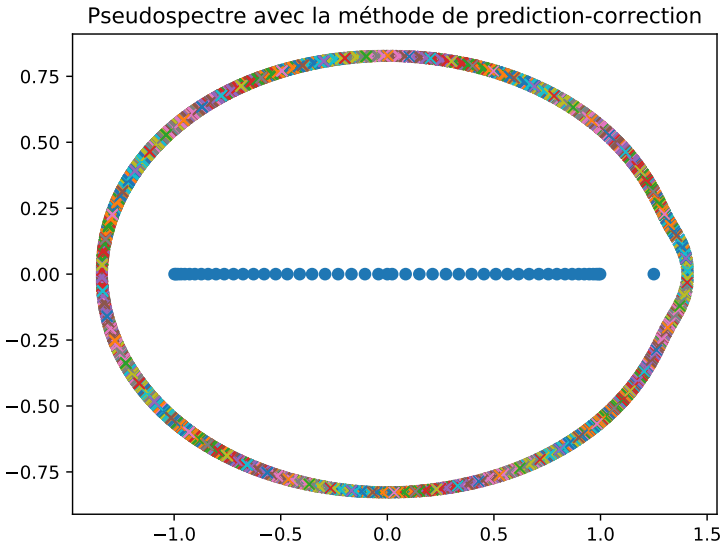
Implémenter la méthode de prédiction-correction en Python s'est avérée être un peu plus compliqué que ce qu'on pensait. Pour commencer, nous avons fait appel à la fonction gershgorin que nous avons définit dans l'algorithme de GRID. Cette fonction nous permet de définir les bornes les plus adaptées et repose sur la démonstration faite en annexe.

Une fois le grillage établit, nous avons pu créer la fonction **predcore**. Son principe est le même qu'en Matlab.

- Nous initialisons le premier point z à la valeur prise par une valeur propre + un ϵ
- Avec la méthode svd nous récupérons les vecteurs $umin, vmin$ et $sigmin$ pour créer la boucle while
- on définit l'erreur ainsi : $\frac{\sigma_{min}(zI-A)-\epsilon}{\epsilon}$
- Tant que (while) l'erreur est supérieure à une tolérance on définit un $z[0]$ qui va correspondre au premier point de la suite
- on introduit une erreur relative entre deux points consécutifs qui est : $\frac{|z[k]-z[0]|}{z[0]}$
- Si cette erreur est supérieure à 0.01 **flag=True** et tant qu'on n'a pas dépassé un nombre d'itérations maximum **stop** on cherche les autres points $z[k]$ par la même méthode
- Si par contre l'erreur relative est inférieure à 0.01 ou si le nombre de passage est supérieur à *stop*, on s'arrête
- Enfin, on affiche à l'aide d'un plot les valeurs propres et le pseudospectre.

Vous remarquerez que le code en annexe possède quelques variantes. Tout comme pour GRID, il s'agit de modifications faites pour fonctionner avec l'interface que nous allons détailler dans la partie qui suit.

L'image suivante est une visualisation du problème en Python.



La représentation d'un même procédé en deux langages différents permet une compréhension totale du phénomène. Le graphique est donc cohérent puisqu'il concorde avec les résultats obtenus en Matlab.

6 Interface graphique

6.1 Principe de fonctionnement

Étant donné que plusieurs programmes sont à lancer, nous avons décidé de créer une interface permettant de donner le choix à l'utilisateur.

L'interface repose sur l'appel des différents programmes grâce à la commande **os.system()**. Elle donne ainsi le choix à l'utilisateur de lancer GRID ou Prédiction-Correction puis lui permet par exemple de rentrer la taille de la matrice, de définir epsilon, de zoomer ou encore de relancer le programme partiellement ou en entier.

6.2 Modifications apportées aux programmes

Pour pouvoir demander par la suite l'entrée de variables, il nous a fallu effectuer quelques modifications. Ainsi par exemple au lieu de définir la taille de la matrice nous avons permis son entrée à l'aide de la commande **sys.argv[1]**, l'élément 0 correspondant au nom du programme par défaut.

Pour le programme prédiction-correction, l'ajout de la fonction zoom accompagnée du nouvel affichage du pseudospectre a nécessité l'ajustement de l'affichage des plots. Pour cela, nous avons défini une variable **on** que le programme prend en argument. Si l'utilisateur clique sur le bouton relancer dans le programme de l'interface, **on** prend la valeur 0, sinon elle prend la valeur 1.

Si la valeur de **on** est définie sur 1, le programme s'effectue normalement. Sinon, nous recalculons les bornes en prenant celles du zoom effectué et relançons le programme en affichant le plot sans le zoom d'un côté et le plot avec de l'autre.

6.3 Algorithmique

L'interface se décompose en deux parties : la partie GRID et la partie Prédiction-correction. Le principe étant le même nous allons parler de manière générale.

- L'interlocuteur choisit quelle partie l'intéresse
- Une fois la partie choisie, il peut lire la documentation, lancer le programme ou quitter l'interface
- Si il décide de lancer la documentation il peut tout de même lancer le programme par la suite
- Si il décide de lancer le programme on lui demande quel langage il désire, matlab ou python
- Dans le cas où son choix se porte sur le premier langage, on ouvre Matlab
- Sinon on lui demande d'entrer des variables comme la taille de la matrice, epsilon ...
- Une fois le programme lancé il peut décider de zoomer et/ou de relancer le programme uniquement sur la partie zoomée
- Sinon, il peut également décider de relancer tout le programme avec de nouvelles valeurs

L'interface possède quelques spécificités puisqu'elle est constituée d'images et de commentaires, nous vous laissons les découvrir.

Conclusion

L'étude des spectres est très intéressante car ils ont de grandes applications dans la vie de tous les jours. Cependant, dans de nombreux cas, des erreurs de mesure nommées epsilon, font qu'un spectre se transforme en pseudospectre. Il existe de multiples implémentations de ce pseudospectre et la plus simple reste celle de GRID, mais qui dit simple à implémenter rime souvent avec limites. Pour palier à ces problèmes, une autre méthode prime, celle de Prédiction-Correction. Elle repose sur une technique très utilisée en analyse, la méthode de Newton. Elle permet l'étude de matrices de grandes dimensions et rend de bons résultats. Il est cependant possible d'aller encore plus loin, il s'agit de l'étude des pseudospectres par composante.

Vous trouverez nos codes sur notre page Github à l'adresse suivante :

`https://github.com/3302011/Projet_Pseudospectre`

Bibliographie

1. **Images**

   Logo Polytech UPMC.
   Available : http ://www.polytech.upmc.fr

   Valeur propre et vecteur propre.
   Available : http ://uel.unisciel.fr/physique/outils_nancy/outils_nancy
   _ch11/co/apprendre_ch11_20.html

   Décomposition en valeurs singulières.
   Available : http ://pagesperso.lina.univ-nantes.fr/ prie-
   y/archives/VEILLE-2009-2012/2012/index/livreBlanc.html

   Prédiction-correction.
   Available :https ://goo.gl/images/AJMyfg.

2. **Site Web**

   [1] Vecteurs propres.
   Available : http ://uel.unisciel.fr/physique/outils_nancy/outils_nancy
   _ch11/co/apprendre_ch11_20.html

   [2] Norme Euclidienne.
   Available : http ://www.math.univ-toulouse.fr/ jroyer/TD/2013-14-Projet-L3/Projet-Blohorn-Boquen-Dubois.pdf

   [3] Décomposition en valeurs singulières.
   Available : https ://www.cs.cmu.edu/ venkatg/teaching/CStheory-infoage/book-chapter-4.pdf

3. **Livres**

   [4] Lloyd Nicholas Trefethen, Mark EmbreeSpectra and pseudospectra :The Behavior of Non-normal Matrices and Operators. Mathématiques. Princeton University Press, 2005.

   [5] A. N. Malyshev and M. Sadkane. Componentwise pseudospectrum of a matrix. Linear. Algebra Appl., 378 :283–288, 2004.

Annexes

Algorithme GRID

Preuve de la méthode de Gershgorin appliquée au pseudospectre

On pose $E_\epsilon = \epsilon\ I_n$, $E_\epsilon \in M_n(\mathbb{C})$ et $||\ E_\epsilon\ ||_2 = ||\ \epsilon I_n\ ||_2 = \epsilon\ ||\ I_n\ || = \sqrt{n}\ \epsilon$

Soit Λ_ϵ(A) est le pseudospectre de la matrice A.

$\Lambda_\epsilon(\mathrm{A}) = \ \lambda \in \mathbb{C} : \lambda \in V_p(X)$ avec X $\in M_n(\mathbb{C}) \Leftrightarrow ||\ X - A\ ||_2 \leq \epsilon\ \Rightarrow$

$\forall$ A $\in M_n(\mathbb{C})$, $\exists\ \epsilon_0$ sup 0 tel que A = X + $\epsilon \Rightarrow$ A = X + E_{ϵ_0} où $\epsilon_0 = \frac{\epsilon}{\sqrt{n}}$

Soit $\lambda \in V_p(X)$ et V le vecteur associé V=(V1,...,V_n)$^t \neq 0$

$$\begin{aligned}\mathrm{AV} &= \mathrm{V}(\mathrm{X}+E_{\epsilon_0}) \\ &= \mathrm{XV} + E_{\epsilon_0}\mathrm{V} \\ &= \lambda\mathrm{V} + E_{\epsilon_0}\mathrm{V} \Rightarrow \textstyle\sum a_{i,j}V_j = \lambda V_j + \sum \epsilon_0 V_j\end{aligned}$$

Si on prend un indice i pour lequel le module V_i est maximal comme pour la méthode de Gershgorin on a :

$$(\lambda\ \text{-}\ a_{i,i})\ V_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}V_j\ \text{-}\ \sum \epsilon_0 V_j$$

$$|\ \lambda\ \text{-}\ a_{i,i}\ | = |\ \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{Vj}{Vi} - \sum_{\substack{j=1 \\ j\neq i}}^{n} \epsilon_0 \frac{Vj}{Vi}\ |$$

$$\begin{aligned}|\ \lambda\ \text{-}\ a_{i,i}\ | &\leq |\ \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j}\frac{Vj}{Vi}\ | + |\ \sum_{\substack{j=1 \\ j\neq i}}^{n} \epsilon_0 \frac{Vj}{Vi}\ | \\ &\leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |\ a_{i,j}\ | + \epsilon_0 \sum_{\substack{j=1 \\ j\neq i}}^{n} |\ 1\ | \\ &\leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |\ a_{i,j}\ | + \epsilon_0\ \mathrm{n} = \sum_{\substack{j=1 \\ j\neq i}}^{n} |\ a_{i,j}\ | + \sqrt{n}\ \epsilon\end{aligned}$$

On retrouve donc $\lambda \in \Lambda_\epsilon$(A) $\Rightarrow |\ \lambda\ \text{-}\ a_{i,i}\ | \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |\ a_{i,j}\ | + \sqrt{n}\ \epsilon$

Code Matlab

```
function pseudospectre()

%%% Variables %%%

n = 10; %Taille de la matrice qu'il sera possible de changer
A = ones(n,n);
eps=1;

%%% Fonctions %%%
% 1. Premiere fonction Random
% Fonction qui va donner un nombre aleatoire pour les nombres reels
% et les nombres imaginaires

j = sqrt(-1);
A = rand(n) + j*rand(n);
%2. Fonction qui va tracer le disque de Gershgorin

[X,Y] = gershdisc(A,eps)

%3. Fonction SVD qui va nous servir pour GRID
[U,S,V]=svd(A);

%4. Fonction GRID
sigmin = grid1(A,X,Y,eps);

figure(1)
gershdisc(A,eps)
figure(2)
grid1(A,X,Y,eps)

end

%%% Appel de fonctions %%%

function [X,Y] = gershdisc(A,eps)

%1ere etape: si la matrice n'est pas carree -> erreur

if size(A,1) ~= size(A,2)
    error('La matrice que vous avez choisi n est pas carree');
    return;
end

%2nd etape: creation du grillage avec elements imaginaires et reels
n=size(A,1);
X = [0,0]; % X = [xmin,xmax]
Y = [0,0]; % Y = [ymin,ymax]

for i=1:size(A,1)
    h=real(A(i,i));
    k=imag(A(i,i)); %Nombres imaginaires et reels
    r=0;
    for j=1:size(A,1)
        if i ~= j
           %Calcul du rayon qui est la somme des normes pour i!=j
            r=r+(norm(A(i,j)));
        end

    end
    r = r + sqrt(size(A,1))*eps;

%3eme etape: Trace des elements

%A. Le cercle

    N=256;
    t=(0:N)*2*pi/N;
    if ((h+r) > X(2))
```

```
        X(2) = h+r;
    end
    if ((k+r) > Y(2))
        Y(2) = k+r;
    end
    if ((h-r)<X(1))
        X(1) = h-r;
    end
    if ((k-r)<Y(1))
        Y(1) = k-r;
    end
%B. Le vecteur

    %plot( r*cos(t)+h, r*sin(t)+k ,'-');

%C. Le centre du cercle

    %hold on;
    %plot( h, k,'+');
end

%D. Pour avoir des axes egaux

%axis equal;

%E. Valeurs propres

ev=eig(A);
%for i=1:size(ev)
    %rev=plot(real(ev(i)),imag(ev(i)),'ro');
    %title('Cercle de Gershgorin')
    %xlabel('Axe reel')
    %ylabel('Axe Imaginaire')
%end
%hold off;
%legend(rev,'Valeurs propres');

end

function sigmin=grid1(A,X,Y,eps)

% X et Y contiennent la limite du rectangle

N=100;
i=sqrt(-1);
pasX = (X(2)-X(1))/(N-1);
pasY = (Y(2)-Y(1))/(N-1);
x = X(1):pasX:X(2);
y = Y(1):pasY:Y(2);
sigmin = zeros(N,N);

for k=1:N
    for j=1:N
        sigmin(j,k)=min(svd((x(k)+y(j)*i)*eye(size(A,1))-A));
    end
end

ev=eig(A);
hold on
for i=1:size(ev)
    rev=plot(real(ev(i)),imag(ev(i)),'ro');
end
contour(x,y,sigmin,[eps eps])
hold off

end
```

Code Python

```
########### code python Gershgorin ###############

from sympy import *
from numpy import linalg #permet le calcul des valeurs propres
import numpy as ny
from random import randint
from pylab import *
from tkinter import * #pour tracer des cercles entre autres
import matplotlib.pyplot as plt
from math import cos,sin,sqrt
import easygui as eg
import sys


###### on cree une matrice random ########

def matrice(n):
    M=[[0 for j in range(0,n)]for i in range(0,n)]

    for i in range(0,n):
        for j in range(0,n):
            M[i][j]=complex(randint(0,10),randint(0,10))
            #genere des nombres aleatoires de 1 a 10
    return(M)


if len(sys.argv) > 2:
    sys.argv[1], sys.argv[2]
else:
     if len(sys.argv) > 1:
         sys.argv[1]
     else:
        print("pb")

M=matrice(int(sys.argv[1]))

##### on calcule ses valeurs propres #############
valeurs_p,vecteurs=linalg.eig(M)
print(valeurs_p)


##########On applique le theoreme de gershgorin au pseudospectre
afin de renvoyer les valeurs maximales que prennent x et y #########

#1.utilisation gershgorin (il faut qu'on l'applique au pseudospectre)

eps =int(sys.argv[2])

def gershgorin(M,eps,valeurs_p):
    ligne,colonne=shape(M)
    N=256
    t=[]
    h = 0
    k = 0
    t.append(0)
    r=0
    A = []
    B = []
    X = [0,0]
    Y = [0,0]

    for z in range(0,N+1): #pour constituer le cercle
        t.append(z*2*pi/N)

    if ligne!=colonne:
        print('Il faut une matrice carree')
    else:
        for i in range(0,len(M)): #on recupere les centres
```

```
            h = M[i][i].real
            k = M[i][i].imag
            r = 0
            for j in range(1,len(M)):
                if i!=j:
                    r=r+norm(M[i][j])
            r = r + sqrt(len(M)) * eps
            for a in range(len(t)):
                A.append(r*cos(t[a]) + h)
                B.append(r*sin(t[a]) + k)
            #plt.plot(A,B,'-')
            #plt.plot(h,k,'+')
            if ((h+r) > X[1]):
                X[1] = h+r
            if ((k+r) > Y[1]):
                Y[1] = k+r
            if ((h-r)<X[0]):
                X[0] = h-r
            if ((k-r)<Y[0]):
                Y[0] = k-r
    return(X,Y)

#constitution des bornes pour grid
X,Y = gershgorin(M,eps,valeurs_p)

def minou(T):
    minette = T[0]
    for i in range(len(T)):
        if (abs(minette) > abs(T[i])):
            minette = T[i]
    return(minette)

###### Fonction GRID#######
def gridou(M,eps,X,Y,valeurs_p):
    n = 100
    i = complex(0,1)
    x = ny.linspace(X[0],X[1],n)
    y = ny.linspace(Y[0],Y[1],n)
    sigminou= zeros((n,n))
    print(X,Y)
    for k in range(n):
        for j in range(n):
            A,B,C = linalg.svd((x[k]+y[j]*i)*eye(len(M))-M)
            sigminou[j][k] = minou(B)
    contour(x,y,sigminou,[eps])

    for i in range(len(valeurs_p)):
        plt.plot(valeurs_p[i].real,valeurs_p[i].imag,'o')
    title("Pseudospectre avec la methode GRID")
    axis('equal')
    plt.show()

gridou(M,eps,X,Y,valeurs_p)
```

Algorithme Prédiction-correction

Matlab

```
function friend()

n = 3;
L = valeurs(n,1);
A = choix_matrice(n,L);

epsi = 0.1;
%stop = 100;
tol = 0.1;
machin = 0.1;
%[X,Y] = gershdisc(A,0.01)
predcor(A,epsi,tol,machin);
end

function M = choix_matrice(n,vp)
    D = eye(n).*vp; % D est la mat diagonale contenant les vp
    P= zeros(n,n);
    for i = 1:n
        for j = 1:n
            P(i,j)=rand(1);
        end
    end
    % On cree une matrice aleatoire
    while(det(P) == 0)
        for i = 0:n
            for j = 0:n
                P(i,j)=rand(1); % On recalcule P dans ce cas
            end
        end
    end
    M = P*D;
    M = mtimes(M,inv(P));
    %M a les bonnes valeurs propres

end

function Liste = valeurs(n,lambda0)
    Liste=zeros(n,1);
    for i = 1:n % on cree des valeurs propres separees par 2
        Liste(i) = lambda0;
        lambda0=lambda0+2;
    end
end


function [X,Y] = gershdisc(A,eps)

%1ere etape: si la matrice n'est pas carree -> erreur

if size(A,1) ~= size(A,2)
    error('La matrice que vous avez choisi n est pas carree');
    return;
end

%2nd etape: creation du grillage avec elements imaginaires et reels
n=size(A,1);
X = [0,0]; % X = [xmin,xmax]
Y = [0,0]; % Y = [ymin,ymax]

for i=1:size(A,1)
    h=real(A(i,i));
    k=imag(A(i,i)); %Nombres imaginaires et reels
    r=0;
    for j=1:size(A,1)
        if i ~= j
            r=r+(norm(A(i,j)));
            %Calcul du rayon qui est la somme des normes pour i!=j
```

```
        end

    end
    r = r + sqrt(size(A,1))*eps;

%3eme etape: Trace des elements

%A. Le cercle

    N=256;
    t=(0:N)*2*pi/N;
    if ((h+r) > X(2))
        X(2) = h+r;
    end
    if ((k+r) > Y(2))
        Y(2) = k+r;
    end
    if ((h-r)<X(1))
        X(1) = h-r;
    end
    if ((k-r)<Y(1))
        Y(1) = k-r;
    end
%B. Le vecteur

    %plot( r*cos(t)+h, r*sin(t)+k ,'-');

%C. Le centre du cercle

    %hold on;
    %plot( h, k,'+');
end

%D. Pour avoir des axes egaux

%axis equal;

%E. Valeurs propres

ev=eig(A);
%for i=1:size(ev)
    %rev=plot(real(ev(i)),imag(ev(i)),'ro');
    %title('Cercle de Gershgorin')
    %xlabel('Axe reel')
    %ylabel('Axe Imaginaire')
%end
%hold off;
%legend(rev,'Valeurs propres');

end




function z =predcor(A,epsi,tol,machin)

    n = size(A,1);
    I = eye(n); %matrice identite
    N = 1000; % Nombre d'iterations maxi
    z=zeros(N,1);
    vecteur = eig(A);
    j = sqrt(-1);
    hold on

    for k=1:length(eig(A))

        vp=vecteur(k);
        %%z = zeros(N,1);
        r = zeros(N,1);
        tau = zeros(N,1);
```

```
    %1ere etape, on determine z
        z(1)=vp+epsi;
        while((abs(svds((z(1)*I-A),1,'smallest')-epsi)/epsi)>tol)

            [umin,sigmin,vmin]=svds((z(1)*I-A),1,'smallest');
            z(1)=z(1) -(sigmin-epsi)/(umin'*vmin);

        end

        flag = 1;
        i = 2;

    %2nd etape, on determine tous les zk et g(k)
        while(flag == 1 && i <N)

            [umin,sigmin,vmin]=svds((z(i-1)*I-A),1,'smallest');

%3 eme etape, les rk correspondent a la direction de la trajectoire
            r(i)=(sqrt(-1)*vmin'*umin)/(abs(vmin'*umin));

    %4eme etape: on determine tau pour trouver le z final
            tau(i)=min(machin,abs(z(i-1)-vp)/2);
            z(i)=z(i-1)+(tau(i)*r(i));

            while((abs(svds((z(i)*I-A),1,'smallest')-epsi)/epsi)>tol)

                [umin,sigmin,vmin]=svds((z(i)*I-A),1,'smallest');
                z(i)= z(i) -(sigmin-epsi)/(umin'*vmin);
             end

             plot(real(z(i)),imag(z(i)),'x')
             if(abs(z(i)-z(1))<0.001*z(1))
                 flag = 0;
             end

             i = i+1;
        end

    end
    axis('equal')
 hold off
end
```

Python

```
from numpy import linalg #permet le calcul des valeurs propres d'une matrice
import numpy as np
from random import randint
from pylab import *
from tkinter import * #pour tracer des cercles entre autres
import matplotlib.pyplot as plt
import matplotlib.axes
from math import *
import sys
from easygui import *



#Pour avoir une matrice particuliere qui a une belle forme
def s(n):
    S=[]
    x=[]
    S.append(0)
    x.append(0)
    M=[[0 for j in range (0,n)]for i in range(0,n)]
    k=0

    for j in range(0,n):
        x.append((2*pi*j)/n)

    for i in range(0,j):
       S.append(2*sin(x[i]))

    for i in range(0,n-1):
        M[i][i]=S[i]
        M[i][i+1]=1
        M[i+1][i]=-1
    return M

n=int(sys.argv[1])
M=s(n)


###Variables globales###

epsi=float(sys.argv[2])  # argument 2
on=int(sys.argv[3]) #pour voir si on continue ou non 0=oui 1=non


x_axis=0 #coordonnee x du zoom
y_axis=0 #coordonnee y du zoom
stop=100
tol=0.1
machin=0.1
TOTO1=0 #variable globale
TOTO2=0


#pour retrouver les coordonnees apres avoir zoome

def onclick(event):

    print('button=%d, x=%d, y=%d, xdata=%f, ydata=%f' %
          (event.button, event.x, event.y, event.xdata, event.ydata))
    global TOTO1
    global TOTO2
    TOTO1=event.xdata
    TOTO2=event.ydata


### Valeurs propres ###

valeurs_p,vecteurs=linalg.eig(M)
```

```
### gershgorin ###

def gershgorin(M, epsi, valeurs_p):
    ligne, colonne=shape(M)
    N=256
    t=[]
    h = 0
    k = 0
    t.append(0)
    r=0
    A = []
    B = []
    X = [0,0]
    Y = [0,0]

    for z in range(0,N+1): #pour constituer le cercle
        t.append(z*2*pi/N)

    if ligne!=colonne:
        print('Il faut une matrice carree')
    else:
        for i in range(0,len(M)): #on recupere les centres
            h = M[i][i].real #va nous donner les points pour les centres des cercles
            k = M[i][i].imag
            r = 0
            for j in range(0,len(M)):
                if i!=j:
                    r=r+norm(M[i][j])
            r = r + sqrt(len(M)) * epsi
            for a in range(len(t)):
                A.append(r*cos(t[a]) + h)
                B.append(r*sin(t[a]) + k)

            if ((h+r) > X[1]):
                X[1] = h+r
            if ((k+r) > Y[1]):
                Y[1] = k+r
            if ((h-r)<X[0]):
                X[0] = h-r
            if ((k-r)<Y[0]):
                Y[0] = k-r

    return(X,Y)


#constitution des bornes pour grid

X,Y = gershgorin(M, epsi, valeurs_p)


##### Programme de prediction-correction en Python ###
def ifon(xmin,xmax,ymin,ymax): #appel de predcore recurcivement

    predcore(M, epsi, stop, tol, machin, [xmin,xmax], [ymin,ymax])


def predcore(M, epsi, stop, tol, machin,X,Y):
    n=len(M)
    I=eye(n)
    valeurs_p, vecteurs=linalg.eig(M) #bien prendre valeur
    plt.plot(valeurs_p.real, valeurs_p.imag, 'o')
    coor1=0
    coor2=0


    for k in range(0,len(valeurs_p)):
        z = []
        vp=valeurs_p[k]
        r = []
        tau = []
        #1ere etape: on determine z
```

```
        z.append(vp+complex(0,1)*epsi)
        [umin,sigmin,vmin] = svd((z[0]*I)-M)
        while abs((sigmin[n-1]-epsi)/epsi)>tol:
            [umin,sigmin,vmin]=svd((z[0]*I)-M)
            z[0]=z[0]-(sigmin[n-1]-epsi)/(np.vdot(umin[:,n-
            1],np.conjugate(vmin[n-1,:])))


        #2nd etape: on determine les autre zk
        a = 1
        flag = True
        while(flag == True and a<stop ):
            [umin,sigmin,vmin]=svd((z[a-1]*I)-M)

            #3eme etape: on determine les rk
            r.append(complex(0,1)*(np.vdot(np.conjugate(vmin[n-1,:]),
            umin[:,n-1])) /abs(np.vdot(np.conjugate(vmin[n-1,:]),umin[:,n-1])))

            #4eme etape: on determine tau
            tau.append(min(machin,abs(z[a-1]-vp)/2))
            z.append(z[a-1]+(tau[a-1]*r[a-1]))
            [umin,sigmin,vmin]=svd((z[a]*I)-M)

            [umin,sigmin,vmin]=svd((z[a]*I)-M)
            z[a]=z[a]-(sigmin[n-1]-epsi)/(np.vdot(umin[:,n-
            1],np.conjugate(vmin[n-1,:])))


            fig=plt.figure(1)
            plt.plot(z[a].real,z[a].imag,'x')


            cid = fig.canvas.mpl_connect('button_press_event', onclick)
            title("Pseudospectre avec la methode de prediction-correction")


            if (abs(z[a]-z[0]) < 0.001*abs(z[0])):
                flag = False
            a += 1



    plt.show()

    xmin=TOTO1
    print(TOTO1)
    xmax=TOTO1+0.1
    ymin=TOTO2-0.2
    ymax=TOTO2

    X=[xmin,xmax,ymin,ymax,on]
    return(X) #on retourne les coordonnees du zoom ainsi
    que on pour voir si on demande le zoom

####### Appel de fonctions #######


X=predcore(M,epsi,stop,tol,machin,X,Y)

if(X[4]==0):
    print(X[0],X[1])
    ifon(X[0],X[1],X[2],X[3]) #si jamais on demande un zoom
    on appelle la fonction ifon qui appelle predcore de facon recursive

elif(X[4]==1):
    i=1#on ne fait rien
```

Interface Graphique

```
from tkinter import *
import subprocess as sp
from easygui import *
import os
import sys
from math import * #pour la barre de progression
import time


######## PRESENTATION #######
msgbox("Bonjour et bienvenue dans la modelisation des pseudospectres
cree par Alexia ZOUNIAS-SIRABELLA, Ines BENZENATI et Fatine BENTIRES ALJ.
Deux methodes de representation sont disponibles: GRID et Prediction-Correction.")

######## GRID OU PREDICTION CORRECTION #######
choices=["Voir GRID","Voir la partie Prediction-correction"]
reply=buttonbox(image="pho.gif",msg="Que voulez vous faire?",
title="Pseudospectre d'une matrice",choices=choices)


################## GRID ####################

if reply==choices[0]:

    msgbox("Bienvenue dans la partie GRID!")
    choices=["Lire la documentation","Lancer le programme !", "Exit"]
    reply=buttonbox(image="homer.gif",msg="ET maintenant?",title="Pseudospectre
    d'une matrice",choices=choices)


########## choix 1 = lire la doc #############


    if reply=="Lire la documentation":
        os.system("Open Ps.pdf")

        choix=["Oui","Exit"]
        reponse=choicebox("On va quand meme pas s'arreter la... on lance le
        programme?",choices=choix)

        if reponse==choix[0]:
            cho=["Python","Matlab"]
            reply=buttonbox(image="now.gif",msg="Plutot Python ou plutot
            Matlab?",title="Va falloir se decider",choices=cho)

            if(reply==cho[1]):
                os.system("open -a fin.m")

            else:
                fields=["Taille de la matrice","Epsilon","Nombre de points"]
                msgbox("Un bon exemple serait de choisir une matrice de taille 10 avec
                un Epsilon egal a 5 et 100 points. Vous pouvez zoomer avec la loupe.")
                n=multenterbox(msg="Afin de personnaliser votre
                pseudospectre",title="Variables pour lancer le
                programme",fields=fields,values=())
                os.system("Python3 gersh.py"+" "+n[0]+" "+n[1]+" "+n[2])

                msgbox("Pour choisir de nouveau toutes les valeurs tapez sur Relancer,
                pour ne choisir que le nombre de points tapez sur Retracer. Si vous
                voulez quitter le programme tapez sur Exit.")
                fields=["Relancer","Retracer de facon plus precise","Exit"]
                res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous
                lacher !",title="Toujours plus!",choices=fields)

                while res==fields[0] or res==fields[1]:

                    fields=["Relancer","Retracer de facon plus precise","Exit"]
                    res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous
                    lacher !",title="Toujours plus!",choices=fields)
```

```
                    if res==fields[0]:
                        f=["Taille de la matrice","Epsilon","Nombre de points"]
                        n=multenterbox(msg="Afin de personnaliser votre
                        pseudospectre",title="Variables pour lancer le
                        programme",fields=f,values=())
                        os.system("Python3 gersh.py"+" "+n[0]+" "+n[1]+" "+n[2])

                    if res==fields[1]:
                        s=enterbox(msg="Donnez le nombre de points",title="",default="")
                        os.system("Python3 gersh.py" + " " + n[0] + " "+ n[1] + " " + s)

                if res=="Exit":
                    c=["Exit"]
                    buttonbox(image="done.gif",msg="",choices=c)

########## choix 2 : lancer le programme ############

    if reply==choices[1]:

        choic=["Python","Matlab"]
        reply=buttonbox(image="now.gif",msg="Plutot Python ou plutot Matlab?",title="Va
        falloir se decider",choices=choic)


######### si Matlab ############

        if(reply==choic[1]):
            os.system("open -a MATLAB_R2016b")

######## si Python ###############

        else:
            msgbox("Un bon exemple serait de choisir une matrice de taille 10 avec un
            Epsilon egal a 5 et 100 points. Vous pouvez zoomer avec la loupe.")
            fields=["Taille de la matrice","Epsilon","Nombre de points"]
            n=multenterbox(msg="Afin de personnaliser votre
            pseudospectre",title="Variables pour lancer le
            programme",fields=fields,values=())
            os.system("Python3 gersh.py"+" "+n[0]+" "+n[1]+" "+n[2])

            msgbox("Pour choisir de nouveau toutes les valeurs tapez sur Relancer. Si
            vous voulez quitter le programme tapez sur Exit.")
            fields=["Relancer","Exit"]
            res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous lacher
            !",title="Toujours plus!",choices=fields)

            while res==fields[0]:
                fields=["Relancer","Exit"]
                res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous
                lacher !",title="Toujours plus!",choices=fields)

                if res==fields[0]:
                    fie=["Taille de la matrice","Epsilon","Nombre de points"]
                    n=multenterbox(msg="Afin de personnaliser votre
                    pseudospectre",title="Variables pour lancer le
                    programme",fields=fie,values=())
                    os.system("Python3 gersh.py"+" "+n[0]+" "+n[1]+" "+n[2])

            if res=="Exit":
                c=["Exit"]
                buttonbox(image="done.gif",msg="",choices=c)


    else:
        c=["Exit"]
        buttonbox(image="done.gif",msg="",choices=c)


######################## PREDICTION CORRECTION ######################################
```

```
if reply=="Voir la partie Prediction-correction":
    msgbox("Bienvenue dans la partie prediction-correction!")
    choices=["Lire la documentation","Lancer le programme !", "Exit"]
    reply=buttonbox(image="homer.gif",msg="ET maintenant?",title="Pseudospectre d'une
    matrice",choices=choices)


########## choix 1 = lire la doc ##############

    if reply=="Lire la documentation":
        os.system("Open Ps.pdf")

        choix=["Oui","Exit"]
        reponse=choicebox("On va quand meme pas s'arreter la... on lance le
        programme?",choices=choix)

        if reponse==choix[0]:
            cho=["Python","Matlab"]
            reply=buttonbox(image="now.gif",msg="Plutot Python ou plutot
            Matlab?",title="Va falloir se decider",choices=cho)

            if(reply==cho[1]):
                os.system("open -a friend.m")

            else:
                fields=["Taille de la matrice","Epsilon","Difference entre les valeurs
                propres"]
                msgbox("Un bon exemple serait de choisir une matrice de taille 3 avec
                un Epsilon egal a 0.1. Vous pouvez zoomer avec la loupe.")
                n=multenterbox(msg="Afin de personnaliser votre
                pseudospectre",title="Variables pour lancer le
                programme",fields=fields,values=())
                os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(1)+" "+n[2])

                f=["Relancer apres zoom","Exit"]
                res=buttonbox(msg="Relancer apres zoom?",title="Zoom",choices=f)

                if res==f[0]:
                    os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(0)+" "+n[2])

                fields=["Relancer","Exit"]
                res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous
                lacher !",title="Toujours plus!",choices=fields)

                while res==fields[0]:

                    f=["Taille de la matrice","Epsilon","Difference entre les valeurs
                    propres"]
                    n=multenterbox(msg="Afin de personnaliser votre
                    pseudospectre",title="Variables pour lancer le
                    programme",fields=f,values=())
                    os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(1)+" "+n[2])


                if res=="Exit":
                    c=["Exit"]
                    buttonbox(image="done.gif",msg="",choices=c)

######### choix 2 : lancer le programme ############

    if reply==choices[1]:

        choic=["Python","Matlab"]
        reply=buttonbox(image="now.gif",msg="Plutot Python ou plutot Matlab?",title="Va
        falloir se decider",choices=choic)


######## si Matlab ###########

        if(reply==choic[1]):
```

```
            os.system("open -a MATLAB_R2016b")

####### si Python ##############

        else:
            msgbox("Un bon exemple serait de choisir une matrice de taille 3 avec un
            Epsilon egal a 0.1. Vous pouvez zoomer avec la loupe.")
            fields=["Taille de la matrice","Epsilon","Difference entre les valeurs
            propres"]
            n=multenterbox(msg="Afin de personnaliser votre
            pseudospectre",title="Variables pour lancer le
            programme",fields=fields,values=())

            os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(1)+" "+n[2])
            #matrice, epsilon, on

            f=["Relancer apres zoom","Exit"]
            res=buttonbox(msg="Relancer apres zoom",title="Zoom",choices=f)

            if res==f[0]:
                os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(0)+" "+n[2])

                msgbox("Pour choisir de nouveau toutes les valeurs tapez sur Relancer.
                Si vous voulez quitter le programme tapez sur Exit.")
                fields=["Relancer","Exit"]
                res=buttonbox(image="stay.gif",msg="Decidemment, on veut pas vous
                lacher !",title="Toujours plus!",choices=fields)
                while res==fields[0]:

                    fie=["Taille de la matrice","Epsilon"]
                    n=multenterbox(msg="Afin de personnaliser votre
                    pseudospectre",title="Variables pour lancer le
                    programme",fields=fie,values=())
                    os.system("Python3 prog.py"+" "+n[0]+" "+n[1]+" "+str(1)+" "+n[2])

                    if res=="Exit":
                        c=["Exit"]
                        buttonbox(image="done.gif",msg="",choices=c)

            else:
                c=["Exit"]
                buttonbox(image="done.gif",msg="",choices=c)

############ choix 3 : quitter le programme #################

    elif reply==choices[2]:
        c=["Exit"]
        buttonbox(image="done.gif",msg="",choices=c)
```