# Calcul des valeurs propres et des vecteurs propres d'une perturbation de matrice

Arnaud GELINET
Stef GRAILLAT

Projet réalisé sous la direction
de Claude-Pierre JEANNEROD

**Résumé**

Le but de ce compte-rendu est de présenter trois algorithmes permettant de calculer localement et à un ordre donné les developpements en série des valeurs propres et des vecteurs propres d'une perturbation de matrice. Nous faisons préalablement une etude théorique premettant de valider les algorithmes. Les algorithmes que nous présentons sont très liés à la forme de la matrice de tête (simple, semi-simple, sous forme de Jordan).

# Table des matières

# 1 Introduction

Le calcul de valeurs propres et de vecteurs propres d'une perturbation de matrice apparaît en mécanique quantique. En effet, en mécanique quantique, il est courant d'avoir à résoudre l'équation de Schrödinger. Cette équation consiste en le calcul des valeurs propres et des vecteurs propres de l'opérateur Hamiltonien $\mathcal{H}_0$. En effet, la structure cristalline des semiconducteurs est périodique et la fonction d'onde ainsi que l'énergie dépendent d'un paramètre réel $k$. L'hamiltonien $\mathcal{H}_k$ peut être écrit comme la somme de $\mathcal{H}_0$ et de la partie perturbée qui est polynomiale en $k$. La théorie des perturbations $kp$ nous amène à une représentation matricielle de l'opérateur $\mathcal{H}_k$. En projetant l'opérateur $\mathcal{H}_k$ sur un sous-espace de dimension finie, on arrive à la matrice $H(k) = H_0 + H_1 k + H_2 k^2$ qui est symétrique pour tout $k \in \mathbb{R}$ : $H^*(k) = H(k)$. Il nous faut alors trouver les valeurs propres et les vecteurs propres de $H_k$ au voisinage de $k = 0$ jusqu'à un certain ordre. (Dans toute la suite, ce petit paramètre sera noté $\varepsilon$).

# 2 Perturbations de matrices

Considérons une matrice $A$, dépendant d'un paramètre réel $\varepsilon$ voisin de 0, ayant un developpement en série entière au voisinage de $\varepsilon = 0$ :

$$A(\varepsilon) = A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \cdots$$

ou les $A_i, i \in \mathbb{N}$ appartiennent à $\mathcal{M}_n(\mathbb{C})$.
Le problème que nous nous posons est de savoir comment évoluent les valeurs propres et les vecteurs propres de $A$ en fonction de $\varepsilon$.
Afin de mieux voir les différentes possibilités, examinons tout d'abord le cas des matrices $2 \times 2$. Voici une liste des cas possibles :

1. $A_0$ a des valeurs propres distinctes (donc est diagonalisable),
   Par exemple
   $$A = \begin{bmatrix} 1 & \varepsilon \\ \varepsilon & 1 \end{bmatrix}$$
   ou les valeurs propres sont
   $$\frac{3 \pm \sqrt{1 + 4\varepsilon^2}}{2}$$
   analytiques (développables en séries entières).
2. $A_0$ a une valeur propre double et est diagonalisable,
   Exemples :
   $$\begin{bmatrix} 1 & 0 \\ 0 & 1+\varepsilon \end{bmatrix}, \begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 1+\varepsilon \end{bmatrix}$$
3. $A_0$ a une valeur propre double et n'est pas diagonalisable.
   Exemples :
   $$\begin{bmatrix} 1+\varepsilon & 1 \\ 0 & 1+\varepsilon \end{bmatrix}, \begin{bmatrix} 1 & 1+\varepsilon \\ 0 & 1 \end{bmatrix}$$

Une première approche possible serait de calculer le polynôme caractéristique $\chi(\lambda, \varepsilon)$ de $A(\varepsilon)$ défini par $\chi(\lambda, \varepsilon) = \det(\lambda I - A(\varepsilon))$.
Si la dépendance en $\varepsilon$ est analytique (ce qui est la cas lorsque les coefficients de $A(\varepsilon)$ sont des polynômes), la théorie des fonctions analytiques permet de montrer que les $n$ racines peuvent être développées au voisinsage de $\varepsilon = 0$ en séries de Puiseux convergentes. Néanmoins, bien qu'il soit assez aisé d'étudier les matrices $2 \times 2$ à l'aide de l'équation caractéristique $\det(\lambda I - A(\varepsilon)) = 0$, trouver les valeurs propres par ce genre de méthode n'est pas la chose la plus fructueuse. Il est en effet difficile de calculer les coefficients du polynôme caratéristique quand la matrice est de taille quelconque car il est assez difficile d'évaluer un déterminant. De plus, passer à l'équation caractéristique enlève une partie des informations contenues dans la matrice. Par exemple, si $A(\varepsilon)$ est symétrique ou hermitienne, les valeurs propres seront réelles et la matrice sera diagonalisable (on applique ici le théorème spectral qui dit que toute matrice auto-adjointe est diagonalisable dans une base orthonormée). Mais ces informations sont perdues lors du passage à l'équation caractéristique de $A(\varepsilon)$ ; par exemple, il n'y a pas de moyen de dire à l'aide du polynôme caractéristique si les racines sont réelles, à moins de trouver toutes ses racines. C'est pour ces raisons qu'en mécanique quantique la méthode standard est de chercher les valeurs propres et les vecteurs propres simultanément sous la forme de séries entières. Il s'agit de substituer les séries

$$\begin{aligned} \lambda(\varepsilon) &= \lambda_0 + \varepsilon\lambda_1 + \cdots \quad , \lambda_i \in \mathbb{C} \\ \nu(\varepsilon) &= \nu_0 + \varepsilon\nu_1 + \cdots \quad , \nu_i \in \mathbb{C}^n \end{aligned}$$

dans l'équation $A(\varepsilon)\nu = \lambda\nu$ et d'identifier les coefficients. Cette méthode n'est pas suffisante dans notre cas, car en effet, dans notre étude, rien ne nous garantit que le matrice est diagonalisable.
La procédure que nous allons développer ici est une version entièrement matricielle de la méthode décrite ci-dessus.
En effet, si $A(\varepsilon)$ est diagonalisable et si nous regroupons ses vecteurs propres $\nu(\varepsilon)$ en tant que colonnes d'une matrice $T(\varepsilon)$ alors

$$B(\varepsilon) = T(\varepsilon)^{-1}A(\varepsilon)T(\varepsilon)$$

est une matrice diagonale dont les éléments sont les valeurs propres $\lambda(\varepsilon)$. Nous cherchons à la fois $T(\varepsilon)$ et $B(\varepsilon)$.
L'avantage de cette méthode est qu'elle est applicable même dans le cas ou $A(\varepsilon)$ n'est pas diagonalisable sachant que dans ce cas, $B(\varepsilon)$ ne sera plus diagonale mais seulement sous forme normale ou plus simplement une matrice semblable à $A(\varepsilon)$.
En fait, l'intérêt de la forme normale est la suivante : si $A(\varepsilon)$ est sous forme normale, il existe (voir [3]) des heuristiques permettant de calculer une matrice de passage $T(\varepsilon)$ diagonale contenant des puissances fractionnaires de $\varepsilon$ telle que la matrice de

tête de $T^{-1}(\varepsilon)A(\varepsilon)T(\varepsilon)$ ne soit plus nilpotente. Cependant, notre but a été de rester rationnel (*i.e.* d'éviter les termes en $\varepsilon^\alpha$ avec $\alpha \in \mathbb{Q} - \mathbb{Z}$).

# 3 Etude théorique

Notre but est de simplifier la perturbation de la matrice en lui appliquant une similitude. Tout l'art réside dans le choix des matrices de similitude. Dans la littérature, on trouve cinq formats pour les matrices de passage.
Une première étape est de transformer $A(\varepsilon) = A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \cdots$ afin que $A_0$ soit sous forme de Jordan. Pour cela on choisit une matrice constante inversible $T_0$ tel que $T_0^{-1}A_0T_0$ soit sous forme de Jordan. Si $T_0$ est inversible, on a alors :

$$T_0^{-1}(A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \cdots)T_0 = (T_0^{-1}A_0T_0) + \varepsilon(T_0^{-1}A_1T_0) + \cdots$$

A partir de maintenant, nous considérons toujours que $A_0$ est sous forme de Jordan. Dans tous les formats, la matrice $A(\varepsilon) = A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \cdots$ est transformée en une autre matrice par des similitudes successives : $B(\varepsilon) = T(\varepsilon)^{-1}A(\varepsilon)T(\varepsilon)$ avec $T(0) = I$. Les schémas sont différenciés par le choix de $T(\varepsilon)$.
Voici la liste des choix de $T(\varepsilon)$ pour les 5 principaux formats pour l'ordre 3 :

(i) $T(\varepsilon) = (I + \varepsilon S_1)(I + \varepsilon^2 S_2)(I + \varepsilon^3 S_3)$
(ii) $T(\varepsilon) = I + \varepsilon T_1 + \varepsilon^2 T_2 + \varepsilon^3 T_3$
(iii) $T(\varepsilon) = e^{\varepsilon U_1} e^{\varepsilon^2 U_2} e^{\varepsilon^3 U_3}$
(iv) $T(\varepsilon) = e^{\varepsilon V_1 + \varepsilon^2 V_2 + \varepsilon^3 V_3}$
(v) On choisit $T(\varepsilon)$ comme solution du système :

$$\frac{dT}{d\varepsilon} = (W_1 + \varepsilon W_2 + \varepsilon^2 W_3)T$$

Dans notre étude, nous nous intéresserons plus particulièrement au **format (iv)**.
*Remarque.* Nous verrons en conclusion que la méthode utilisée est quasi-indépendante du format considéré.
Mais avant de commencer cela, il convient de mettre en place quelques notations et d'énoncer quelques théorèmes tous issus de [2] qui seront utiles par la suite. Rappelons tout d'abord la définition du crochet de Lie.

**Définition 3.1.** On appelle **opérateur de Lie** vis à vis de $X \in \mathcal{M}_n(\mathbb{C})$ l'application notée $\mathcal{L}_X : \mathcal{M}_n(\mathbb{C}) \to \mathcal{M}_n(\mathbb{C})$ et définie par $\mathcal{L}_X(Y) = [Y, X] = YX - XY$
Dans la suite du compte-rendu, nous utiliserons fréquemment la notation £ qui correspond à $\mathcal{L}_{A_0}$.
Enonçons un théorème relatif à l'opérateur de Lie et à l'exponentielle de matrice :

**Théorème 3.1.** *Pour $X, Y \in \mathcal{M}_n(\mathbb{C})$ on a :*

$$e^{-X}Ye^X = e^{\mathcal{L}_X}Y$$

*Preuve :* Nous prouverons le théorème sous la forme :

$$e^{-tX}Ye^{tX} = e^{\mathcal{L}_X Y} \quad \forall t \in \mathbb{R}$$

Le théorème précédent étant un cas particulier pour $t = 1$. Puisque $\mathcal{L}_X$ est un opérateur linéaire, $e^{t\mathcal{L}_X}$ est définie et on a :

$$e^{t\mathcal{L}_X}Y = \sum_{j=0}^{\infty} \frac{t^j}{j!}\mathcal{L}_X^j Y$$

Puisque que chaque membre de l'équation est analytique, il suffit de prouver qu'ils ont le même développement en série entière, c'est à dire :

$$\left.\frac{d^j}{dt^j}e^{-tX}Ye^{tX}\right|_{t=0} = \mathcal{L}_X^j Y \quad \forall j$$

Pour montrer cela, nous calculons d'abord

$$\frac{d}{dt}e^{-tX}Ye^{tX} = e^{-tX}(YX - XY)e^{tX} = e^{-tX}\mathcal{L}_X e^{tX}$$

En recommençant $j$ fois ce calcul, on obtient

$$\frac{d}{dt}e^{-tX}Ye^{tX} = e^{-tX}\mathcal{L}_X^j e^{tX}$$

et le résultat est obtenu en posant $t = 0$. ■

Intéressons nous plus particulièrement à notre format donné par $T(\varepsilon) = e^{\varepsilon V_1 + \varepsilon^2 V_2 + \varepsilon^3 V_3}$ (on donne ici un exemple à l'ordre 3 ce qui permet de donner une idée de l'algorithme).
Développons l'expression de $B(\varepsilon)$ à l'aide du Théorème (3.1). Nous obtenons :

$$B(\varepsilon) = e^{\mathcal{L}(\varepsilon)}A(\varepsilon)$$

où

$$\mathcal{L}(\varepsilon) = \varepsilon\mathcal{L}_1 + \varepsilon^2\mathcal{L}_2 + \varepsilon^3\mathcal{L}_3$$

avec

$$\mathcal{L}_j = \mathcal{L}_{V_j}$$

Puisque la série $\mathcal{L}(\varepsilon)$ commence avec un terme en $\varepsilon$, $\mathcal{L}(\varepsilon)^j$ commence avec un terme d'ordre $\varepsilon^j$, et il s'en suit que :

$$e^{\mathcal{L}(\varepsilon)} = I + \mathcal{L}(\varepsilon) + \frac{1}{2}\mathcal{L}(\varepsilon)^2 + \frac{1}{6}\mathcal{L}(\varepsilon)^3$$

En développant, on obtient

$$\begin{aligned}
A &= A_0 + \varepsilon A_1 + \varepsilon^2 A_2 + \varepsilon^3 A_3 \\
\mathcal{L}(\varepsilon)A &= \varepsilon\mathcal{L}_1 A_0 + \varepsilon^2(\mathcal{L}_1 A_1 + \mathcal{L}_2 A_0) + \varepsilon^3(\mathcal{L}_1 A_2 + \mathcal{L}_2 A_1 + \mathcal{L}_3 A_0) \\
\mathcal{L}(\varepsilon)^2 A &= \varepsilon^2(\mathcal{L}_1^2 A_0) + \varepsilon^3(\mathcal{L}_1^2 A_1 + \mathcal{L}_1\mathcal{L}_2 A_0 + \mathcal{L}_2\mathcal{L}_1 A_0) \\
\mathcal{L}(\varepsilon)^3 A &= \varepsilon^3(\mathcal{L}_1^3 A_0)
\end{aligned}$$

En identifiant, on obtient alors :

$$
\begin{aligned}
B_0 &= A_0 \\
B_1 &= A_1 + \mathcal{L}_1 A_0 \\
B_2 &= A_2 + \mathcal{L}_1 A_1 + \mathcal{L}_2 A_0 + \frac{1}{2}\mathcal{L}_1^2 A_0 \\
B_3 &= A_3 + \mathcal{L}_1 A_2 + \mathcal{L}_2 A_1 + \mathcal{L}_3 A_0 + \frac{1}{2}(\mathcal{L}_1^2 A_1 + \mathcal{L}_1\mathcal{L}_2 A_0 + \mathcal{L}_2\mathcal{L}_1 A_0) + \frac{1}{6}\mathcal{L}_1^3 A_0
\end{aligned}
$$

En remarquant que $\mathcal{L}_j A_0 = -\pounds V_j$, on déduit des équations précédentes qu'elles peuvent s'écrire sous la forme :

$$\pounds V_j = K_j - B_j \tag{1}$$

où $K_j$ consiste en les termes de droite des équation précédentes excepté le terme $\mathcal{L}_j A_0$. On remarque que les $K_j$ sont connus lorsque l'on en a besoin. L'équation (1) est connue sous le nom d'**équation homologique**.

Il est pratique de définir

$$A_i^{(0)} = A_i$$

et par récurrence

$$A_i^{(j+1)} = \sum_{1 \leq l \leq i+1} \mathcal{L}_l A_{k-l}^{(l)}.$$

On peut réarranger ces quantités dans un triangle appelé **Triangle de Lie** :

$$
\begin{array}{cccccc}
A_0^{(0)} & & & & & \\
A_1^{(0)} & A_0^{(1)} & & & & \\
A_2^{(0)} & A_1^{(1)} & A_0^{(2)} & & & \\
A_3^{(0)} & A_2^{(1)} & A_1^{(2)} & A_0^{(3)} & & \\
\vdots & \vdots & \vdots & & \ddots & \\
A_k^{(0)} & A_{k-1}^{(0)} & A_{k-2}^{(0)} & \dots & \dots & A_0^{(k)}
\end{array}
$$

où les coefficients sont calculés par la méthode suivante :

$$
\begin{array}{|c|c}
\cline{1-1}
A_0^{(j)} & \\
A_1^{(j)} & \\
\vdots & \\
A_i^{(j)} & \\
\hline
A_{i+1}^{(j)} & \multicolumn{1}{c|}{A_i^{(j+1)} \longleftarrow \displaystyle\sum_{1 \leq l \leq i+1} \mathcal{L}_l A_{i+1-l}^{(j)}} \\
\hline
\end{array}
$$

Une fois le triangle construit, $B_k$ peut être obtenue comme la somme pondérée sur chaque ligne du triangle :

$$B_k = \sum_{l=0}^{k} \frac{1}{l!} A_{k-l}^{(l)}$$

et on a aussi $B_k = \mathcal{L}_k A_0^{(0)} + K_k$. On remarque que pour calculer $V_k$ il faut inverser £. Dans la section suivante, nous allons étudier l'opérateur £ et montrer qu'il est facile de trouver un $V_j$ satisfaisant à $\pounds V_j = K_j - B_j$.

# 4 Etude de l'équation homologique

Dans la section précédente, les calculs nous montrent que l'on arrive à une équation homologique du type $\pounds U_j = K_j - B_j$ où $K_j$ est connu à la $j^{\grave{e}me}$ étape et où les matrices constantes $U_j$ et $B_j$ sont à déterminer.
Voici une procédure permettant de résoudre l'équation homologique :

1. Choisir un sous-espace $\mathcal{C}$ de $\mathcal{M}_n(\mathbb{C})$ supplémentaire de Im$\pounds$ tel que

$$\mathcal{M}_n(\mathbb{C}) = \mathrm{Im}\pounds \oplus \mathcal{C}$$

2. Calculer la projection $B_j$ et $K_j - B_j$ de $K_j$ dans $\mathcal{C}$ et Im$\pounds$
3. Inverser £ sur son image, c'est à dire calculer un inverse de $K_j - B_j$ par £ pour obtenir $U_j$.
4. Connaisant $U_j$ (qui n'est pas unique) on peut calculer $K_{j+1}$ et poursuivre la résolution des équations.

## 4.1 Choix du sous-espace $\mathcal{C}$

Il convient tout d'abord de rappeler quelques définitions.

**Définition 4.1.** Soit $L : \mathbb{C}^n \to \mathbb{C}^n$ une application linéaire. On dit que $L$ est semi-simple si L est diagonalisable sur $\mathbb{C}$ (*i.e.* vu comme une application linéaire $L : \mathbb{C}^n \to \mathbb{C}^n$).
Nous allons montrer un théorème très important pour la suite :

**Théorème 4.1.** *Soit $V$ un $\mathbb{C}$-espace vectoriel de dimension finie.*
*Si $L : V \to V$ est semi-simple alors*

$$V = ImL \oplus KerL$$

*Preuve :* On choisit une base dans laquelle $L$ est diagonale ou diagonale par blocs. Il résulte qu'alors Ker$L$ est engendré par les vecteurs propres associés à la valeur propre 0 et Im$L$ est engendré par le reste des élément de la base. ■

Nous allons maintenant montrer que l'opérateur £ est semi-simple. Pour cela, nous avons besoin du lemme suivant :

**Lemme 4.2.** *Si $S$ est semi-simple, alors $\mathcal{L}_S$ est semi-simple*

*Preuve :* Si $S$ est déjà sous forme diagonale (de diagonale $(\lambda_1, \ldots, \lambda_n)$) alors on a

$$\mathcal{L}_S E_{rs} = (\lambda_s - \lambda_s) E_{rs}$$

ou $E_{rs} = (\delta_{i,r}\delta_{s,j})_{i,j}$ et il vient que $\mathcal{L}_S$ a $n^2$ vecteurs propres linéairements indépendants et donc est semi-simple. Maintenant, si $S$ n'est pas diagonale, elle est diagonalisable sur $\mathbb{C}$ et donc il existe $T \in \mathcal{M}_n(\mathbb{C})$ tel que $T^{-1}ST = \Lambda$ soit diagonale. Alors les vecteurs propres sont $TE_{rs}T^{-1}$ :

$$\mathcal{L}_S(TE_{rs}T^{-1}) = [TE_{rs}T^{-1}, T\Lambda T^{-1}] = T[E_{rs}, \Lambda]T^{-1} = (\lambda_s - \lambda_r)TE_{rs}T^{-1}$$

■

**Corollaire 4.3.** *Si $A_0$ est semi-simple avec pour valeurs propres $\lambda_r$ alors £ est semi-simple avec pour valeurs propres $\lambda_s - \lambda_r$. Si $A_0$ est diagonale, alors*

$$£ E_{rs} = (\lambda_s - \lambda_r) E_{rs} \tag{2}$$

Puisque £ est semi-simple, par le théorème (4.1), on déduit que

$$\mathcal{M}_n(\mathbb{C}) = \mathrm{Im}£ \oplus \mathrm{Ker}£$$

On peut alors expliciter Ker£.

**Théorème 4.4.** *Ker£ est engendré par les $E_{rs}$ tels que $\lambda_r = \lambda_s$*

*Preuve :* Il s'agit d'une conséquence directe de la formule (2). ■

De plus, l'inversion partielle de £ est aussi très simple grâce au théorème suivant.

**Théorème 4.5.** *Soit $X = (x_{rs})$ une matrice appartenant à Im£ . On peut alors écrire*

$$X = \sum x_{rs} E_{rs}$$

*la somme étant effectuée sur les indices $(r, s)$ tel que $\lambda_r \neq \lambda_s$. Soit $Y$ définie par*

$$Y = \sum \frac{x_{rs}}{\lambda_s - \lambda_r} E_{rs}$$

*. On a alors £ $Y = X$.*

*Preuve :* Il suffit de le vérifier. ■

# 5 Implémentations

Pour l'implémentation des algorithmes, nous avons utilisé le logiciel de calcul formel MAPLE V release 5.

## 5.1 Implémentation de l'algorithme simple

Nous avons commencé par coder l'algorithme pour une matrice $A_0$ diagonalisable sur $\mathbb{C}$ et à valeurs propres toutes distinctes (dans ce cas $A_0$ est dite simple).
En effet, dans ce cas, le calcul de la projection $B_j$ et $K_j - B_j$ de $K_j$ dans $\mathcal{C}$ et Im£ est très simple, de même que le calcul de l'inverse de £ sur son image. Il suffit d'utiliser le théorème (4.4) en remarquant que dans notre cas, les valeurs propres étant toutes distinctes, on a Ker£ $=$ Vect$(E_{rr})$. Par conséquent, projeter une matrice $K$ sur Ker£ revient à prendre sa diagonale.

---

**Algorithme simple:**

ENTREE: $A$ une matrice, $\varepsilon$ l'indéterminé, $k$ l'ordre que l'on veut
SORTIE: $B$ une matrice diagonale, $P$ et $P^{-1}$ les matrices de passage de $B$ à $A$

**Etape 1** On declare les différentes variables:
$TriLie$: il s'agit du triangle de Lie
$K$ : vecteur des matrices $K_i$
$B$ : vecteur des matrices $B_i$
$n$ : dimension de la matrice

**Etape 2** On initialise les variables:
$B[0] = A_0$
$K[1] = A_1$
$TriLie[i, 1] = A_i$ pour $i = 1..n$

**Etape 3** **Pour** $j = 2..n$ **Faire**

**1** Calculer $K_j$
**2** Calculer la j$^{\text{ème}}$ ligne de $TriLie$ (sauf le coefficient $TriLie[j, 1]$ que l'on ne connait pas (on en connait pas le terme $\mathcal{L}_j A_0^{(0)}$
**3** Calculer $B_j$ en prenant la diagonale de $K_j$
**4** Connaissant $B_j$ et $K_j$, calculer $\mathcal{L}_j A_0^{(0)} = B_j - K_j$
**5** Actualiser la j$^{\text{ième}}$ ligne de $TriLie$
**6** Calculer $U_j$ inverse de $B_j - K_j$ par £ (cf Théorème (4.5)).
**Fin Pour**

---

## 5.2 Test de l'algorithme simple

```
> restart:
> read algo1:
> A:=matrix(2,2,[1,x,x,2]);
```

$$A := \begin{bmatrix} 1 & x \\ x & 2 \end{bmatrix}$$

```
> B:=algosimple(A,x,4)[1];
```

$$B := \begin{bmatrix} 1 - x^2 + x^4 & 0 \\ 0 & 2 + x^2 - x^4 \end{bmatrix}$$

```
> series((3+sqrt(1+4*x^2))/2,x=0);
```

$$2 + x^2 - x^4 + \mathrm{O}(x^6)$$

```
> series((3-sqrt(1+4*x^2))/2,x=0);
```

$$1 - x^2 + x^4 + \mathrm{O}(x^6)$$

## 5.3 Implémetation de l'algorithme semi-simple

Dans cet algorithme, nous gérons le cas ou $A_0$ est semi-simple mais à valeurs propres multiples. Dans ce cas, on applique une fois l'algorithme précédent afin d'obtenir une matrice diagonale par blocs. On traite ensuite tous les blocs dont la taille est supérieure (strictement) à 1.
Soit $B_j(\varepsilon)$ un bloc diagonale de $B$. On appelle alors l'algorithme précédent sur

$$A^{(2)} = \frac{B_j(\varepsilon) - B_j(0)}{x}$$

On effectue cela récursivement en s'arrêtant lorsque l'on arrive sur une matrice de Jordan. Il ne reste alors plus qu'à reconstruire la solution lors de la remonté de la récurrence.

## 5.4 Test de l'algorithme semi-simple

```
> restart:
> read algo2final:
> A:=matrix(2,2,[1+x^2,x,x,1+x]);
```

$$A := \begin{bmatrix} x^2 + 1 & x \\ x & 1 + x \end{bmatrix}$$

```
> B:=appelrec(A,x,2)[1];
```

$$B := \begin{bmatrix} \frac{1}{2}\,x\,\sqrt{5}+\frac{1}{2}\,x+\frac{1}{2}\,x^2-\frac{1}{10}\,x^2\,\sqrt{5}+1 & 0 \\ 0 & \frac{1}{2}\,x-\frac{1}{2}\,x\,\sqrt{5}+\frac{1}{2}\,x^2+\frac{1}{10}\,x^2\,\sqrt{5}+1 \end{bmatrix}$$

```
> eiglc(A,x,3);
```

$$time\ =,\ .080$$

$$\{1+(-\frac{1}{5}\,\mathrm{RootOf}(-1-\_Z+\_Z^2)+\frac{3}{5})\,x^2+x\,\mathrm{RootOf}(-1-\_Z+\_Z^2)\}$$

```
> solve(-1-Z+Z^2,Z);
```

$$\frac{1}{2}+\frac{1}{2}\sqrt{5},\ \frac{1}{2}-\frac{1}{2}\sqrt{5}$$

```
> expand(1+(-1/5*(1/2+1/2*sqrt(5))+3/5)*x^2+x*(1/2+1/2*sqrt(5)));
```

$$\frac{1}{2}\,x\,\sqrt{5}+\frac{1}{2}\,x+\frac{1}{2}\,x^2-\frac{1}{10}\,x^2\,\sqrt{5}+1$$

```
> expand(1+(-1/5*(1/2-1/2*sqrt(5))+3/5)*x^2+x*(1/2-1/2*sqrt(5)));
```

$$\frac{1}{2}\,x-\frac{1}{2}\,x\,\sqrt{5}+\frac{1}{2}\,x^2+\frac{1}{10}\,x^2\,\sqrt{5}+1$$

# 6 Résolution du problème lorsqu $A_0$ est sous forme de Jordan

Nous allons ici étudier le dernier cas envisagé pour $A_0$ : $A_0$ n'est pas diagonalisable, elle est seulement sous forme de Jordan.
Pour traiter ce cas, nous allons envisager une approche un peu différente du problème.
Jusqu'à présent, nous avons considéré le cas où la décomposition de $\mathcal{M}_n(\mathbb{C})$ était de la forme

$$\mathcal{M}_n(\mathbb{C}) = \mathrm{Im}\pounds \oplus \mathcal{C}$$

Avec

$$\mathcal{C} = \mathrm{Ker}\pounds$$

Nous allons maintenant envisager le cas où :

$$\mathcal{C} = \mathrm{Ker}\pounds^*$$

.
($\pounds^*$ est l'adjoint de $\pounds$, au sens du produit scalaire défini ci-dessous)
et nous allons résoudre l'équation homologique (4) dans ce cas.

## 6.1 Etude théorique préliminaire

Nous présentons ici les résultats utiles pour l'algorithme.

Rappelons tout d'abord une définition :

**Définition 6.1. Produit scalaire hermitien de matrices :**
Soient $A, B \in \mathcal{M}_n(\mathbb{C})$. Le produit scalaire de $A$ et $B$ est défini par

$$< A, B >= \sum_{i,j} a_{ij}\overline{b_{ij}}$$

*Remarque.* C'est ce produit scalaire qui sera utilisé dans toute la suite.
Ensuite, nous disposons du résultat suivant ([2],chap 2, p.30), très utile pour la suite :

**Théorème 6.1.** *Soit $V$ un $\mathbb{C}$-espace de dimension finie muni d'un produit scalaire. Soit $L : V \to V$ un opérateur linéaire.*
*Alors : $V = \mathrm{Im}L \oplus \mathrm{Ker}L^*$*
*où $L^*$ désigne l'opérateur adjoint de $L$.*

On se référera au livre de Murdock, chap 2, p.30, pour la démonstration.

*Remarque.* Ce théorème justifie la nouvelle décomposition que nous prenons. Il est aussi à noter que ce théorème est assez général. En effet, contrairement à la décomposition précédente, où L était supposé semi-simple, on ne fait aucune hypothèse restrictive sur l'opérateur L.
Nous avons aussi besoin du théorème suivant ([2],chap 3, p.97) :
**Théorème 6.2.** *Soit $A \in \mathcal{M}_n(\mathbb{C})$ On a :*

$$(\pounds_A)^* = \pounds_{A^*}$$

On utilisera en particulier l'égalité : $\pounds^* = \pounds_{A_0^*}$.

On se réfèrera de même que précédemment au livre de Murdock, chap 3, p.97, pour la démonstration.

Enfin , nous disposons du résultat suivant([2],chap 3.4.10, p.105) :
**Théorème 6.3.** *Soit $\pounds_S P = Q$ avec $P$ et $Q$ connus. Soit $R = \sum_{r=0}^{k-1}(-1)^r(\pounds_S^{-1})\pounds_N^r P$*
*Alors $\pounds_{S+N} R = Q$*

## 6.2 Implémentation de l'algorithme lorsque $A_0$ est sous forme de Jordan

Nous présentons maintenant la méthode utilisée pour résoudre l'équation homologique dans le cas où $\mathcal{C} = \mathrm{Ker}\pounds^*$.
Dans notre cas, $A_0$ est sous forme de Jordan, donc on peut la décomposer sous la forme d'une matrice diagonale S et d'une matrice nilpotente N.

---

**Algorithme de résolution de l'équation homologique :** [2] p.105

ENTREE : $A_0$ la matrice de tête
SORTIE : $B$ une matrice diagonale
**Etape 1** Décomposer $A_0$ sous la forme $S + N$

**Etape 2** **Pour** $j$ de 2 à $n$ **Faire**

**1** Calculer $K_j$ selon la même méthode que dans l'algorithme semi simple (5.3).
**2** Décomposer $K_j$ en $K_j = (K_j - B_j) + B_j$, avec $B_j \in \mathrm{Ker}\pounds^*$ : pour cela, on calcule $B_j$ selon la méthode expliquée ci-après (*Remarque 1*).
**3** Trouver $U_j$ tel que $\pounds U_j = K_j - B_j$
Pour cela, on décompose $B_j$ en $B_j^{'} + B_j^{''}$ de telle sorte que $B_j^{'} \in \mathrm{Ker}\pounds_S$. Ceci se fait simplement en prenant pour $B_j^{'}$ les blocs diagonaux de $B_j$. La partie résiduelle $B_j^{''}$ est inutilisée.

- Trouver $\widehat{U}_j$ en résolvant $\pounds_S\widehat{U}_j = K_j - B_j^{'}$ de la même manière que dans l'algorithme semi-simple. (5.3)
- Trouver $U_j^{'}$ en résolvant $\pounds U_j^{'} = K_j - B_j^{'}$. On utilise pour cela la formule le théorème 6.3.
- Résoudre $\pounds U_j^{''} = B_j^{'} - B_j$. Pour cela, on remarque que $B_j^{'} - B_j \in \mathrm{Ker}\pounds_S$ (*voir Remarque 2 ci-dessous*). Ainsi, on se ramène à résoudre $\pounds_N U_j^{''} = B_j^{'} - B_j$, ce qui est très simple (problème équivalent à une résolution de système linéaire)
- Au final, on a calculé $U_j^{'}$ et $U_j^{''}$ tels que :

$$\begin{cases} \pounds U_j^{'} &= K_j - B_j^{'} \\ \pounds U_j^{''} &= B_j^{'} - B_j \end{cases}$$

On en déduit donc par linéarité :

$$U_j = U_j^{'} + U_j^{''}$$

.

**Fin Pour**

---

*Remarque. 1* Connaissant à la $j^{ème}$ étape la matrice $K_j$, nous avons besoin de calculer, la matrice $B_j$ associée. Ceci se fait de la manière suivante :

$K_j$ est dense, mais on la partitionne conformément à la structure de Jordan de $A_0$ :

$$K_j = \left(\begin{array}{c|cc|cc} * & * & * & * & * \\ \hline * & * & * & * & * \\ * & * & * & * & * \\ \hline * & * & * & * & * \\ * & * & * & * & * \end{array}\right)$$

.

On traite chacun des blocs séparément. Pour un bloc, on annule les termes surdiagonaux, et on fait la moyenne des termes de chaque sous diagonale inférieure.
On obtient donc une matrice $B_j$ de la forme :

$$B_j = \left(\begin{array}{c|cc|cc} a & d & 0 & i & 0 \\ \hline 0 & e & 0 & j & 0 \\ b & f & e & k & j \\ \hline 0 & g & 0 & l & 0 \\ c & h & g & m & l \end{array}\right)$$

*Remarque. 2 Montrons que résoudre* $\pounds U_j^{"} = B_j^{'} - B_j$ *est équivalent à résoudre* $\pounds_N U_j^{"} = B_j^{'} - B_j$ *:*
- On a $B_j^{'} - B_j \in \mathrm{Ker}\pounds_S$. En effet, $B_j^{'} \in \mathrm{Ker}\pounds_S$ par définition de $B_j^{'}$. De plus, $B_j \in \mathrm{Ker}\pounds^*$ et $\mathrm{Ker}\pounds^* = \mathrm{Ker}\pounds_S^* \cap \mathrm{Ker}\pounds_N^*$.
Par ailleurs, $\mathrm{Ker}\pounds_S^* = \mathrm{Ker}\pounds_{S^*}$ et $\pounds_S^* = \pounds_{S^*}$ car S est diagonale. Donc, $S^* = S$. D'où $B_j \in \mathrm{Ker}\pounds_S$.
- On en déduit que $\pounds U_j^{"} \in \mathrm{Im}\pounds \cap \mathrm{Ker}\pounds_S$, donc que $\pounds U_j^{"} = \pounds_N U_j^{"}$. Finalement, on doit calculer $U_j^{"}$ tel que $\pounds_N U_j^{"} = B_j^{'} - B_j$

# 7 Conclusion et perpectives

Nous avons impémenté des algorithmes entièrement matriciels. Ils ont l'avantage d'éviter le calcul de racines de polynômes (dont ils n'existent pas de formes par radicaux par des polynôme de degré supérieur ou égal à 5).
Neanmoins, un des inconvénient de notre méthode est qu'il faut mettre certaines matrices sous forme de Jordan e qui est assez coûteux.
Un des avantages de cette méthode est qu'elle est presque indépendante du format choisi. En effet, excepté le triangle de Lie qui est modifiée, on arrive toujours à une équation homologique. La méthode de résolution proposée est encore valable.

Il aurait été intéressant de pouvoir comparer l'efficacité des differents formats mais cela n'a pu être fait faute de temps.

## Remerciements

Nous tenons à remercier Claude-Pierre Jeannerod pour ses nombreux conseils et explications tant durant la phase de compréhension des algorithmes que pendant leur codage.

## Références

[1] Alexei A.Mailybaev. « Transformation of families of matrices to normal forms and its application to stability theory ». *SIAM J.Matrix Anal. Appl.*, 21(2):396–412, novembre 1999.

[2] James A.Murdock. *Livre à paraitre ...* 2000.

[3] G. Chen. « *Solutions formelles de systèmes d'équations différentielles linéaires ordinaires homogènes* ». PhD thesis, Université Joseph Fourier, 1990.

[4] C. Cohen-Tannoudji. *Mécanique quantique*, volume I et II. Hermann, 1973.

[5] C. P. Jeannerod et N. Maillard. « Using computer algebra to diagonalize some Kane matrices ». *J. Phys. A: Math. Gen.*, 33:2857–2870, mars 2000.