1. TP 0 : Prise en main (2H)

Reprise d'exos du premier semestre à la machine.
Utilisation de l'aide, manipulations de base

2. TP Algèbre linéaire 1 (1H30)

Gauss à la semi-mano par multiplication matricielles
Cf. poly d'exos

3. TP Algèbre linéaire 2 (1H30)

L'objectif de ce TD est de voir quelques applications de la *diagonalisation* de matrices, en particulier pour l'étude des suites récurrentes. Comme d'habitude, c'est à vous de déterminer, au cas par cas, quels calculs il vaut mieux faire à la machine ou à la main. Les exercices sont de difficulté croissante.

Exercice 1. *La première application concerne la modélisation de la dynamique des populations, telle qu'étudiée en écologie. Le cas d'école que nous allons considérer n'est pas nouveau, puisqu'il a été proposé en 1202 par Leonardo Pisano comme problème récréatif dans un de ses ouvrages, le Liber Abaci.*

Possédant initialement un couple de lapins, combien de couples obtient-on en douze mois si chaque couple engendre tous les mois un nouveau couple à compter du second mois de son existence ? ■

Nous allons considérer un modèle très simple, dans lequel on notera f_n le nombre nombre de paires de lapins au n-ième mois dans une population idéale telle que :
- *le premier mois, il y a juste une paire de lapereaux,*
- *les lapereaux ne sont pubères qu'à partir du deuxième mois,*
- *chaque mois, toute paire susceptible de procréer engendre effectivement une nouvelle paire de lapereaux,*
- *les lapins ne meurent jamais.*

(1) *Donner la relation de récurrence satisfaite par la suite f_n.*

(2) *Donner le surnom de Leonardo Pisano.*

(3) *Nous souhaitons maintenant calculer f_n pour n relativement grand. Pour cela, nous allons mettre le problème sous forme matricielle en notant pour $n >= 2$, $F_n := \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$.*

  (a) *Donner F_2, et déterminer une matrice M telle que $F_{n+1} = MF_n$ pour $n \geq 2$.*

  (b) *À l'aide d'une boucle* `for`*, calculer F_{1026} et F_{65636}.*

  (c) *Calculer $F_{1026} = M^{1024}F_2$ et $F_{65638} = M^{65636}F_2$. À votre avis, comment s'y prend la machine ?*

  (d) *Pouvez-vous l'égaler en utilisant* uniquement *des produits de matrices ?*

  (e) *Évaluer le nombre d'opérations arithmétiques (+, * sur les coefficients des matrices) nécessaires pour calculer M^{2^n}.*

  (f) *Utiliser la commande* `time` *pour comparer le temps de calcul de $M^{2^{20}}$ et de $float(M)^{2^{20}}$. Interpréter.*

  (g) *Calculer numériquement $f_{2^{20}}$, $f_{2^{23}}$, $f_{2^{24}}$. Interpréter.*

(4) *Nous cherchons maintenant donner une formule exacte pour f_n. Pour cela, on va* diagonaliser *la matrice M.*

  (a) *Vérifier que $M^2 - M - I_2 = 0$. Comparer avec le résultat de* `linalg : :minimalPolynomial(M,z)`*.*

(b) *Soit v un vecteur non nul et λ un réel tels que $Mv = \lambda v$. Montrer que $\lambda^2 - \lambda - 1 = 0$. En déduire les deux valeurs $\lambda_1 < \lambda_2$ de λ possibles.*

(c) *Chercher deux vecteurs non nuls v_1 et v_2 tels que $Mv_1 = \lambda_1 v_1$ et $Mv_2 = \lambda_2 v_2$. On pourra utiliser la commande `linalg : :nullspace`, en remarquant que l'équation $Mv = \lambda v$ est équivalente à $(M - \lambda I_2)v = 0$.*

(d) *λ_1 et λ_2 sont les* valeurs propres *de M, et v_1 et v_2 des* vecteurs propres *correspondants. comparer avec le résultat des commandes : `linalg : :eigenvalues(M)` et `linalg : :eigenvectors(M)`.*

(e) *Soit B la base canonique de $\mathbb{R}^2$, et B' la base (v_1, v_2). Donner la matrice de passage de B à B' puis de B' à B.*

(f) *Calculer au moyen des formules usuelles de changement de base la matrice M' dans B' de l'application qui a pour matrice M dans la base B. Interpréter.*

(g) *Calculer M'^n. En déduire M^n, puis une formule close pour f_n.*

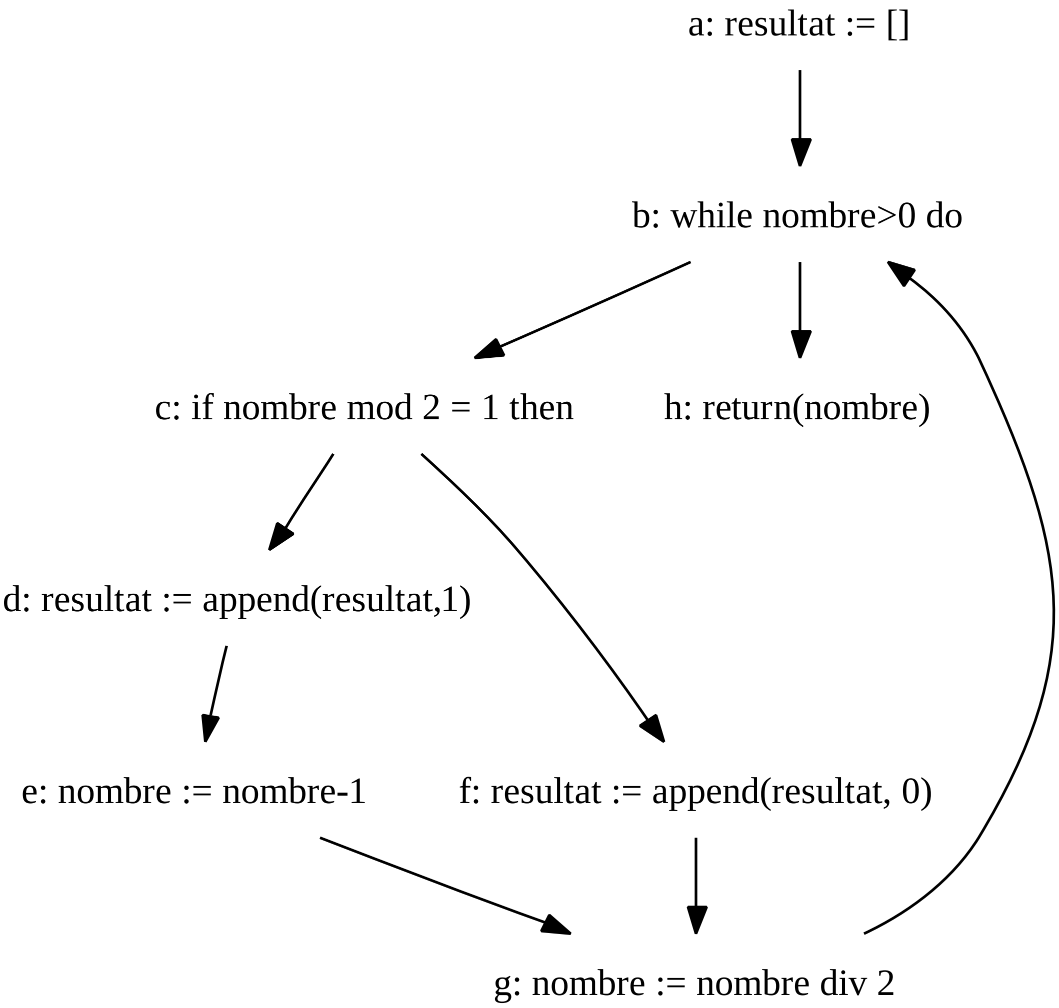
(h) *Utiliser cette formule pour calculer de manière exacte puis numériquement quelques valeurs de f_n pour n grand (voir `?radsimp`).*

Exercice 2. *La deuxième application concerne le test automatique de programmes informatiques. À quelques simplifications près, le modèle que l'on va étudier est utilisé dans le monde industriel (voir `http://www.lri.fr/~gouraud/recherche.fr.html`), et les questions que l'on va rencontrer sont significatives des problèmes rencontrés dans le monde du test.*

Dans ce modèle, un programme est décrit par un graphe de contrôle. Par exemple, le programme suivant :

```
decompositionBinaire :=
proc(nombre)
begin
    resultat := [];
    while nombre>0 do
        if nombre mod 2 = 1 then
             resultat := append(resultat,1);
             nombre := nombre-1
        else
             resultat := append(resultat,0)
        end_if;
        nombre := nombre div 2;
    end_while;
    resultat;
end_proc:
```

sera décrit par le graphe suivant :



Une trace d'exécution est alors une suite de sommets consécutifs allant du sommet a au sommet h dans ce diagramme. Par exemple, en appliquant ce programme au nombre 5, on obtient la trace $(a, b, c, d, e, g, b, c, f, g, b, c, d, e, g, b, h)$. *Vous pouvez le vérifier à la main, ou en utilisant la commande* `debug(decompositionBinaire(5))`. *La longueur d'une trace est le nombre de sommets parcourus, soit 17 dans l'exemple précédent.*

Le principe de cette méthode de test est d'étudier les traces d'exécutions possibles, et de déterminer si elles sont correctes. Ici, notre objectif est simplement de compter le nombre h_k *de traces d'exécutions de longueur* n*, afin de déterminer jusqu'à quelle longueur on peut envisager de faire du test exhaustif. En particulier, on ne se préoccupera pas ici des valeurs des variables du programme. Formellement, le problème est de compter les chemins dans un graphe orienté.*

(1) *Lister toutes les traces d'exécutions de longueur inférieure à 17.*

(2) *On considère le vecteur*

(1) $$V_n := \begin{pmatrix} a_n \\ b_n \\ c_n \\ d_n \\ e_n \\ f_n \\ g_n \\ h_n \end{pmatrix}$$

où c_n compte le nombre de traces d'exécutions du sommet a au sommet c de longueur n, et similairement pour $a_n, \dots, h_n$

(a) *Calculer les vecteurs V_0, V_1, V_2, V_3, V_4.*

(b) *Donner une formule de récurrence permettant d'exprimer g_n en fonction de $a_{n-1}, \dots, h_{n-1}$. Exprimer de même les autres coordonnées de V_n.*

(c) *Déterminer une matrice M telle que $V_n = MV_{n-1}$.*

(d) *Calculer le nombre de traces d'exécutions de longueur 1024. Évaluer très grossièrement le temps nécessaire pour tester toutes ces traces, sur un ordinateur à 2GHz (on admettra que le traitement de chaque sommet prend environ 100 cycles d'horloge).*

(3) *Pour donner une formule approximative pour h_n, on va procéder comme dans l'exercice précédent en diagonalisant la matrice M.*

(a) *Calculer les valeurs propres de M à l'aide de la commande `linalg : :eigenvalues(M)`. Interprétation ? Calculer le polynôme minimal de M avec `linalg : :minimalPolynomial`*

(b) *Pour simplifier, nous nous contenterons de faire un calcul numérique approché, en considérant la matrice de flottants `float(M)`.*

(c) *Calculer numériquement les vecteurs propres de M.*

(d) *En déduire une base B' de R^n dans laquelle M est diagonale.*

(e) *Calculer les matrices de changement de base.*

(f) *Calculer la matrice M' correspondant à M dans cette nouvelle base.*

(g) *Calculer M'^n, puis M^n, puis V_n puis h_n.*

(h) *Donner une approximation de h_n sous la forme a^n.*

(i) *Jusqu'à quelle valeur de n peut-on raisonnablement faire du test exhaustif ?*

- Analyse de données Plot de nuage de points

Objectif : constater dans la pratique que les vecteurs propres de la matrice d'inertie d'un objet mécanique donnent ses axes principaux de rotation. À faire en 2D, puis en 3D, avec une centaine de points.

Application à l'analyse de donnée : représenter un nuage de points par un patatoïde. => infos sur les corrélations.

En principe, on se ramène à des variables réduites pour avoir une bonne métrique malgré les unités différentes.

Exemple (pour lequel les unités sont homogènes ; donc on peut prendre la métrique euclidienne). Voir TP d'Albane.

4. TP : Géométrie

Le calcul matriciel est un outil fondamental en géométrie. En particulier, il donne des notations compactes et puissantes pour le graphisme 2D et 3D. Dans ce TP, nous allons explorer comment utiliser les matrices pour implanter des transformations géométriques naturelles comme les rotations, les symétries ou les projections.

Tout au long de ce TP, nous visualiserons l'effet des transformations sur le polygone suivant, dont chaque sommet est décrit par un vecteur de $\mathbb{R}^2$[1] :

```
points := [matrix(point)
           $ point in [[   0, 1755], [1980,  180], [2070, 1260], [3150,  720],
                       [2340, 3240], [4095, 1125], [3060, 3735], [4860, 1260],
                       [4050,  900], [4590,  900], [7650, 2070], [6300, 1890],
                       [5130, 1395], [3645, 4275], [3735, 4320], [5175, 1485],
                       [5445, 1620], [4950, 3105], [5355, 3285], [6795, 2250],
                       [7695, 2295], [4590, 4860], [4365, 4725], [4860, 3915],
                       [4590, 3780], [3915, 4455], [2790, 3780], [3060, 2880],
                       [2070, 3645], [1710, 2655], [2970, 990],  [1440, 2295],
                       [1845, 450], [1035, 2160]]
```

Pour visualiser le polygône, utiliser :

```
plot(plot::Polygon2d(points))
```

Si vous avez des talents de graphistes, merci de nous proposer d'autres polygones esthétiques pour les TP de l'année prochaine !

Exercice 3. *Soit M une matrice M de dimension 2×2, et ϕ l'application linéaire de $\mathbb{R}^2$ dans $\mathbb{R}^2$ correspondante. Notre objectif est d'interpréter géométriquement la transformation ϕ lorsque cela est possible.*

Considérons d'abord la matrice :

```
M := matrix ( [ [ 0,1]
                [ 1,0] ]):
```

Étant donné un vecteur v de $\mathbb{R}^2$, l'image $\phi(v)$ de v par ϕ peut être obtenue en calculant `M*v`. Calculer l'image des vecteurs de la base canonique de $\mathbb{R}^2$.

Pour visualiser l'image du polygone par ϕ, il suffit donc de faire :

```
plot(plot::Polygon2d([ M*v $ v in points]))
```

(1) *Quelle est la transformation géométrique effectuée par ϕ ?*

(2) *Même question pour les matrices suivantes :*

```
M := matrix ( [ [-1, 0]
                [ 0,-1]])

M := matrix ( [ [ 4, 0]
                [ 0, 1]])

M := matrix ( [ [ 1, 0]
                [ 0, 1]])
```

[1] Rappel : `matrix(1,2)` permet de construire le vecteur de $\mathbb{R}^2$ de coordonnées $(1, 2)$; pour comprendre la notation \$, essayer `f(x) $ x in [a,b,c,d]`.

```
M := matrix ( [ [ 1, 0]
                [ 0,-1]])
```

(3) *L'animation suivante permet de visualiser l'effet de la matrice*

```
M := matrix ( [ [ 1, 0]
                [ 0, 1/2^n]])
```

lorsque n tend vers l'infini.

```
plot(plot::Polygon2d([ matrix([[1,0],[0,1/2^n]])*v $ v in points], n=1..10))
```

Quelle est la transformation géométrique pour la matrice limite

```
M := matrix ( [ [ 1, 0]
                [ 0, 0]])
```

(4) *Chercher la matrice de l'application linéaire ϕ qui effectue une homothétie (zoom) de facteur 2, et plus généralement de facteur lambda. Vérifier le résultat en visualisant le polygone transformé.*

(5) *En vous inspirant de l'exemple précédent, construire une animation où le polygone grossit progressivement.*

```
plot(plot::Polygon2d([matrix([[lambda,0],
                              [0, lambda]])*v
                     $ v in points],
                    lambda=1..100))
```

(6) *Chercher la matrice de l'application linéaire ϕ qui effectue une rotation d'un quart de tour dans le sens trigonométrique.*

(7) *Chercher la matrice de l'application linéaire ϕ qui effectue une rotation d'angle θ dans le sens trigonométrique (indication : chercher l'image par ϕ des vecteurs de la base canonique).*

(8) *Construisez une animation où le polygone tourne autour du point de coordonnées $(0,0)$.*

```
r := 2*PI/360:
plot(plot::Polygon2d([matrix([[cos(theta*r),-sin(theta*r)],
                              [sin(theta*r), cos(theta*r)]])*v
                     $ v in points],
                    theta=0..359))
```

(9) *Même chose, en faisant grossir progressivement le polygone au fur et à mesure qu'il tourne.*

```
r := 2*PI/360:
plot(plot::Polygon2d([matrix([[theta*cos(theta*r),-theta*sin(theta*r)],
                              [theta*sin(theta*r),theta*cos(theta*r)]])*v
                     $ v in points],
                    theta=1..5000))
```

(10) *Chercher la matrice de l'application linéaire ϕ qui envoie le vecteur $b_1 := \begin{matrix}1\\1\end{matrix}$ sur $2b_1$, et $b_2 := \begin{matrix}1\\-1\end{matrix}$ sur $\frac{1}{2}b_2$ (indication : donner la matrice de ϕ dans la base (b_1, b_2), puis appliquer les formules de changement de base pour obtenir sa matrice M dans la base canonique).*

(11)

Exercice 4 (Transformations affines)**.** *Toutes les transformations que nous avons rencontrées pour l'instant sont* linéaires *; en particulier le vecteur* 0 *reste inchangé.*

(1) *Quelle opération matricielle pourrait-on utiliser pour décaler le polygone de* 1000 *points vers la gauche ?*

(2) *Plus généralement pour effectuer une translation le long d'un vecteur de coordonnées* (x, y) *?*

(3) *Construire une animation où le polygone part vers la droite de plus en plus vite.*

(4) *Ce que nous venons de faire, c'est de nous placer dans une classe plus grande de transformations : les* transformations affines. *Celles si sont de la forme* $v \mapsto Mv + u$*, où* M *est une matrice* 2×2*, et* u *un vecteur de* $\mathbb{R}^2$*.*

(5) *Chercher la transformation affine correspondant à la rotation d'angle 45 degrés autour du centre du polygone.*

(6) *Construire une animation où le polygone tourne, grossit et se déplace simultanément.*

Exercice 5 (3D)**.** *Nous voulons maintenant animer le polygone en 3D. Afin de comprendre la mécanique interne, nous n'utiliserons pas les possibilités natives de* `MuPAD`*. Pour simplifier, nous nous contenterons ici de perspective cavalière.*

(1) *Pour plonger le polygone dans* $\mathbb{R}^3$*, nous pouvons utiliser l'application linéaire de* $\mathbb{R}^2$ *dans* $\mathbb{R}^3$ *qui envoie les deux vecteurs de la base canonique de* $\mathbb{R}^2$ *sur les deux premiers vecteurs de* $\mathbb{R}^3$*. Donner la matrice de cette application linéaire.*

(2) *Réciproquement, pour visualiser un objet en 3D sur l'écran (qui n'a que deux dimensions), il faut faire une projection de* $\mathbb{R}^3$ *dans* $\mathbb{R}^2$*. Nous supposerons que l'observateur regarde depuis le haut, de sorte que la projection se fera sur le plan des* x *et* y*. Donner la matrice de cette application linéaire.*

(3) *Calculer le produit des deux matrices correspondant au plongement de* $\mathbb{R}^2$ *dans* $\mathbb{R}^3$ *suivi de la projection de* $\mathbb{R}^3$ *dans* $\mathbb{R}^2$*. Interprétation ?*

(4) *Afficher le polygone après plongement puis projection.*

(5) *Donner la matrice de la rotation de* $\mathbb{R}^3$ *d'angle* θ *autour de l'axe des* x*. Indication : faire un dessin, et chercher l'image des vecteurs de la base canonique. Afficher le polygone après plongement, rotation de 45 degrés puis projection. Animer le résultat pour* θ *variant de 0 à 359 degrés.*

Exercice 6 (Perspective cavalière)**.** *Probablement trop ambitieux.*

5. TP : Polynômes 1 (1H30)

L'objectif de ce TP est d'explorer plusieurs façons d'approximer une fonction f de $\mathbb{R}$ dans $\mathbb{R}$ par une fonction plus simple.

Exercice 7 (Approximation par une fonction affine)**.** *Soit* $f := x \mapsto \sin(x)$*. Tracer son graphe avec*

```
f := sin:
plot(plot::Function2d(f))
```

Notre objectif est de construire des fonctions simples qui approximent f *aux alentours d'un point* x_0*.*

(1) *Chercher une constante* a_0 *telle que la fonction constante* $g := a_0$ *approxime au mieux* f *aux environs de* 0*. Tracer son graphe par dessus celui de* f *avec*

```
plot(plot::Function2d(g(x)), plot::Function2d(g(x)))
```

Zoomer le graphique.

(2) *Chercher une fonction affine* $g := a_0 + a_1x$ *qui approxime au mieux* f *aux environs de* $x_0 := 0$*. La comparer graphiquement avec* f*.*

(3) *Chercher une fonction affine* $g := a_0 + a_1(x - x_0)$ *qui approxime au mieux* f *aux environs de* $x_0 := \pi/4$ *(*π *se note* `PI`*).*

(4)

```
xmin := -2*PI:
xmax := 2*PI:
plot(plot::Function2d(sin(x), x=xmin..xmax),
     plot::Function2d(f(x0) + f'(x0)*(x-x0), x=xmin..xmax, x0=xmin..xmax),
     ViewingBoxYRange=-2..2)
```

Exercice 8 (Interpolation de Lagrange)**.**

- Approximation d'une fonction par un polynôme : interpolation / DL / Padé ? Objectif : leur faire passer la notion de DL

6. TP : Polynômes 2 (1H30)

Exercice 9. *Un vieux pirate est sur son lit de mort. Dans sa jeunesse il a enfoui un Fabuleux Trésor dans la lagune interminable de l'Ile de la Tortue, quelque part à l'est du Grand Cocotier. Il a réuni ses dix lieutenants préférés pour leur transmettre l'information secrète indispensable : la distance entre le Grand Cocotier et le Trésor. Connaissant bien ses lieutenants, et dans un étonnant dernier sursaut de justice, il ne voudrait pas qu'une conjuration de quelques uns d'entre eux assassines les autres pour empocher seuls le trésor. En tenant cependant compte de la mortalité habituelle du milieu, il souhaite donner une information secrète à chacun de ses lieutenants pour que sept survivants puissent toujours retrouver ensemble le trésor, mais pas moins. Comment peut-il s'y prendre ?*

(1) ■ *The art of doing mathematics consists in finding that special case which contains all the germs of generality* ■ *– David Hilbert*

Autrement dit, la première chose à faire pour attaquer ce genre de problème est de le généraliser pour des paramètres quelconques, et de commencer par essayer de comprendre ce qu'il se passe pour de petites valeurs des paramètres. Ici, les paramètres naturels sont le nombre n de lieutenants, et le nombre k de survivants. Que peut on dire pour $n \leq 1$? pour $k \leq 1$?

(2) *Étudions maintenant le cas $n = k = 2$. Soit b la distance secrète. Le pirate choisit au hasard un coefficient a qu'il garde secret, et considère la fonction $f(x) = ax + b$. Puis il choisit au hasard deux points d'évaluation $x_1 := -3$ et $x_2 := 7$ qu'il annonce publiquement. Enfin, il dit en secret au premier pirate que $f(x_1) = 5$ et au deuxième pirate que $f(x_2) = 7$. Les deux pirates se mettent ensemble ; retrouver la valeur de b. Est-ce que le premier pirate aurait pu la retrouver tout seul ?*

(3) *Généralisation pour $n = 10$ et $k = 7$: choisir une valeur secrète b. Tirer au hasard les coefficients d'un polynôme $f(x) = a_6x^6 + \cdots + a_1x + b$ de degré 6 et dont le terme constant est b. Tirer au hasard dix points d'évaluation $x_1, \ldots, x_10$ distincts et non nuls, et calculer $f(x_1), \ldots, f(x_10)$. À l'aide de la commande* `interpolate`*, et en utilisant uniquement $x_1, \ldots, x_7$ et $f(x_1), \ldots, f(x_7)$ retrouver la valeur de b. Pour se mettre en condition réelle, donner à chaque étudiant du TP un couple $(x_i, f(x_i))$, et laissez les retrouver ensemble la valeur secrète.*

Codage Reed Solomon / crypto secret partagé

7. TP : Analyse 1 (1H30)

- Notion de limite : exemples de fonctions avec et sans limites, exemples où la machine se plante, ...

8. TP : Analyse 2 (1H30)

Calcul numérique d'intégrale (TP préparé par Caroline ?)