Travaux dirigés avec SAGE (partie III)

Math 3 — Année 2010-2011

Sommaire

1 Vecteurs et matrices **2**
- 1.1 Construction, opérations élémentaires . . . 2
  - 1.1.1 Vecteurs . . . 2
  - 1.1.2 Matrices . . . 2
  - 1.1.3 Liste de variables . . . 3
- 1.2 Corps de base . . . 3
- 1.3 Echelonnement « à la main » . . . 4

2 Espaces et sous-espaces vectoriels **5**
- 2.1 Construction . . . 5
  - 2.1.1 Sous-espaces . . . 5
  - 2.1.2 Base utilisateur, base échelonnée . . . 5
- 2.2 Exercices . . . 6
  - 2.2.1 Rang d'une famille de vecteurs . . . 6
  - 2.2.2 Changement de base . . . 6
  - 2.2.3 Jacobien . . . 6
  - 2.2.4 Inverse et puissances d'une matrice symbolique . . . 6
  - 2.2.5 Erreur de SAGE . . . 6

3 Applications linéaires **7**
- 3.1 Matrice d'une application linéaire . . . 7
- 3.2 Noyau, image . . . 7

4 Systèmes d'équations linéaires **8**
- 4.1 Résolution matricielle . . . 8
- 4.2 Résolution avec `solve` . . . 8
- 4.3 Exercices . . . 8
  - 4.3.1 Projection orthogonale . . . 8
  - 4.3.2 Matrices commutantes . . . 9
  - 4.3.3 Carrés magiques . . . 9

5 Valeurs propres, vecteurs propres, diagonalisation **9**

6 Exercices **10**
- 6.1 Reconnaître une matrice diagonalisable . . . 10
- 6.2 Récurrence linéaire . . . 10
- 6.3 Système différentiel linéaire du premier ordre . . . 11
- 6.4 Réduction d'une conique . . . 11
- 6.5 Problème des milliardaires . . . 11

1 Vecteurs et matrices

1.1 Construction, opérations élémentaires

1.1.1 Vecteurs

Un vecteur de $\mathbb{R}^n$ est défini en donnant la liste de ses composantes. Entrez par exemple :

```
u = vector([1,3,2]) ; v = vector([2,5,4])
u ; v ; 2*u-v
```

Calculez le produit scalaire $u \cdot v$ et le produit vectoriel $u \wedge v$:

```
u.dot_product(v) ; u.cross_product(v)
```

ainsi que la norme de u (il y en a plusieurs, la seconde est la norme euclidienne) :

```
u.norm(1) ; u.norm(2) ; u.norm(infinity)
```

On accède aux composantes d'un vecteur en précisant son indice entre crochets. ATTENTION : les indices sont numérotés à partir de 0 :

```
u[0] ; u[1] ; u[2]
```

La commande `list(u)` donne la liste des composantes de u.

1.1.2 Matrices

Une matrice à n lignes et p colonnes est définie en donnant ses dimensions et la liste de ses coefficients :

```
M = matrix(2,3,[1,3,2,2,5,4]) ; M
```

On peut aussi définir M en donnant une liste de listes :

```
M = matrix([[1,3,2],[2,5,4]]) ; M
```

ou encore la liste de ses vecteurs-lignes :

```
M = matrix([u,v]) ; M
```

L'accès à un coefficient de la matrice s'obtient en précisant ses indices de ligne et de colonne entre crochets (numérotés à partir de 0) :

```
M[0,0]
```

De nombreuses méthodes s'appliquent aux matrices. Entrez par exemple :

```
M.transpose() ; M.rank() ; M.list()
```

et calculez :

```
M = M.stack(matrix([1,1,1])) ; M
```

```
A = M^3 - 7*M^2 - M - 1
A ; A == 0
```

La matrice identité s'obtient avec `identity_matrix`, la matrice nulle avec `zero_matrix`, une matrice diagonale avec `diagonal_matrix`.

1.1.3 Liste de variables

Il est fréquent d'avoir à utiliser plusieurs variables qu'on voudrait nommer par exemple `x1,x2,...,xn`. L'astuce suivante permet simultanément de déclarer ces variables et d'en construire la liste. Tapez :

```
vars = [var('x%s' % i) for i in range(1,10)]
vars
```

Explication : l'expression `'x%s' % i` est une chaîne de caractères *formatée*. Essayez par exemple :

```
print 'Le produit de %s par %s vaut %s' % (2,3,6)
```

A l'exécution, la ou les valeurs du n-uplet de droite sont insérées dans la chaîne de formatage de gauche, aux endroits marqués par `%s`.

Ainsi l'expression `var('x%s' % 1)` équivaut à `var('x1')`. De plus l'expression `var('x1')` a pour valeur `x1`, ce qui permet d'utiliser cette construction dans une expression. Par exemple, vous pouvez définir un polynôme arbitraire $P(X) = \sum_{i=0}^{8} a_i X^i$ de degré 8 avec la commande :

```
var('X')
P(X) = sum(var('a%s' % i)*X^i for i in range(9)) ; P(X)
```

Entrez la matrice :

```
M = matrix(3,3,vars) ; M
```

calculez son déterminant :

```
det(M)
```

et sa matrice inverse $N = M^{-1}$:

```
N = M^-1 ; N = N.factor()  # pour simplifier N
show(N)
```

et le produit matriciel $P = NM$:

```
P = N * M ; P.factor()
```

1.2 Corps de base

SAGE utilise les notations suivantes :
- `ZZ` : anneau $\mathbb{Z}$ des entiers relatifs (ce n'est pas un corps),
- `QQ` : corps $\mathbb{Q}$ des nombres rationnels,
- `RR` : corps $\mathbb{R}$ des nombres réels (avec 53 bits de précision par défaut),
- `CC` : corps $\mathbb{C}$ des nombres complexes (avec 53 bits de précision par défaut),
- `SR` : « corps » des expressions symboliques.

Entrez par exemple :

```
RR^3
```

Dans la définition d'un vecteur ou d'une matrice, en premier paramètre on peut imposer un corps de base. Entrez :

```
A = matrix(2,2,[1,2,3,4])
B = matrix(QQ,2,2,[1,2,3,4])
C = matrix(RR,2,2,[1,2,3,4])
A ; B ; C
```

Ces matrices sont égales mais n'ont pas le même type, ce qu'on peut vérifier avec `type` ou avec la méthode `parent` qui donne l'espace ambiant d'un objet :

```
parent(A) ; parent(B) ; parent(C)
```

Entrez les matrices complexes :

```
D = matrix(2,2,[1,I,-1,1+I])
E = matrix(CC,2,2,[1,I,-1,1+I])
D ; E
```

Ont-elles le même parent ?

Remarques :

1. Si le corps de base n'est pas précisé, SAGE choisit le plus « petit » possible.
2. Pour manipuler des matrices comportant des variables symboliques, il faut prendre comme corps de base `SR`.
3. Les approximations numériques de nombres réels sont considérées comme rationnelles. Faites l'expérience :

```
y = pi.n(digits=20) ; print y
y in QQ
```

1.3 Echelonnement « à la main »

Considérons les trois vecteurs $u = (1, 3, 2), v = (2, 5, 4), w = (3, 2, 6)$ de $\mathbb{R}^3$. Le vecteur w est-il combinaison linéaire de u et v, et si oui, quels sont les coefficients de cette combinaison linéaire ?

Entrez :

```
u = vector([1,3,2]) ; v = vector([2,5,4]) ; w = vector([3,2,6])
M = matrix(SR,[u,v,w]) ; M
```

Ajoutez à M une quatrième colonne grâce à `augment`, avec des symboles `U,V,W` (en majuscules) pour représenter les vecteurs u, v, w :

```
var('U,V,W')
M = M.augment(matrix(3,1,[U,V,W])) ; M
```

Nous allons *échelonner* M, c'est-à-dire la modifier de telle sorte que chaque ligne commence par plus de zéros que la ligne précédente, en utilisant la méthode du pivot de Gauss. Les seules actions autorisées sont :
– échanger deux lignes (méthode `swap_rows`),
– multiplier une ligne par un scalaire non nul (méthode `rescale_row`),
– ajouter à une ligne un multiple d'une autre ligne (méthode `add_multiple_of_row`).

Commencez par faire apparaître un 0 en ligne 1, colonne 0 en ajoutant à la ligne 1 la ligne 0 multipliée par -2 :

```
M.add_multiple_of_row(1,0,-2) ; M
```

puis un 0 en ligne 2, colonne 0 en ajoutant à la ligne 2 la ligne 0 multipliée par -3 :

```
M.add_multiple_of_row(2,0,-3) ; M
```

et enfin un 0 en ligne 2, colonne 1 en ajoutant à la ligne 2 la ligne 1 multipliée par -7 :

```
M.add_multiple_of_row(2,1,-7) ; M
```

Sur la dernière ligne, les trois premiers coefficients forment un vecteur nul. Le coefficient en bas à droite donne donc une combinaison linéaire nulle des vecteurs u, v, w, d'où l'on tire la réponse à notre question : $w = -11u + 7v$.

Remarque.— La question posée revenait à vérifier si $w \in \mathrm{Vect}(u, v)$ et, si oui, exprimer w en fonction de u et v. Vous allez voir que SAGE sait résoudre ce genre de problème de façon automatique.

2 Espaces et sous-espaces vectoriels

2.1 Construction

La commande `VectorSpace(K,n)` permet de définir l'espace vectoriel K^n. Définissez par exemple :

```
E = VectorSpace(QQ,3)     # ou plus simplement : E = QQ^3
E
```

La commande `MatrixSpace(K,n,p)` permet de définir l'espace des matrices à n lignes et p colonnes à coefficients dans K :

```
M22Q = MatrixSpace(QQ,2,2) ; M22Q
```

Si l'on veut faire du calcul matriciel exact et non approché, il est conseillé de travailler avec comme corps de base `QQ` ou `SR`.

2.1.1 Sous-espaces

La méthode `subspace` (ou `span`) définit le sous-espace vectoriel engendré par une liste de vecteurs. En gardant les mêmes vecteurs u, v, w que ci-dessus, définissez le sous-espace vectoriel $F = \mathrm{Vect}(u, v, w)$ de E en entrant :

```
F = E.subspace([u,v,w]) ; F
```

La dimension de F peut être obtenue par `F.dimension()`.

Pour vérifier que $w \in \mathrm{Vect}(u, v)$, entrez :

```
G = E.subspace([u,v])
w in G
```

La méthode `subspace_with_basis` définit un sous-espace vectoriel en spécifiant une base. Entrez :

```
H = E.subspace_with_basis([u,v]) ; H
```

Vérifiez que $G = H$.

On peut calculer l'intersection de deux sous-espaces :

```
x = vector([1,1,1])
H1 = E.subspace([w,x])
H.intersection(H1)
```

2.1.2 Base utilisateur, base échelonnée

Dans SAGE, tout sous-espace vectoriel de K^n possède une *base échelonnée* « canonique ». On peut par exemple obtenir la base échelonnée de H par la commande :

```
H.echelonized_basis()
```

qui rend une liste de vecteurs.

Contrairement à G, le sous-espace H possède en outre une *base utilisateur* : celle qui a servi à le définir. La méthode `basis` renvoie la base utilisateur ou, à défaut, la base échelonnée :

```
G.basis() ; H.basis()
```

Faites afficher la base canonique de `M22Q`.

La méthode `echelon_coordinates` donne les coordonnées d'un vecteur dans la base échelonnée, la méthode `coordinates` donne ses coordonnées dans la base utilisateur (ou, à défaut, dans la base échelonnée). Pour avoir les coordonnées de w dans la base utilisateur de H, sous forme de liste, entrez :

```
H.coordinates(w)
```

ou sous forme de vecteur :

```
H.coordinate_vector(w)
```

On retrouve ainsi le résultat du §1.3.

2.2 Exercices

2.2.1 Rang d'une famille de vecteurs

Soient les quatre vecteurs de $\mathbb{R}^4$:

$$\begin{aligned} u_1 &= (4,-2,-2,-6) \\ u_2 &= (3,1,2,-2) \\ u_3 &= (1,2,1,1) \\ u_4 &= (-1,-1,-3,0). \end{aligned}$$

1. Calculer le rang de la famille (u_1, u_2, u_3, u_4).
2. Exprimer u_4 comme combinaison linéaire de u_1, u_2, u_3.

2.2.2 Changement de base

Dans $\mathbb{R}^3$ on considère les vecteurs :

$$\begin{gathered} a_1 = (2,-1,1), \quad a_2 = (1,0,3), \quad a_3 = (-1,-1,1) \\ b_1 = (1,1,0), \quad b_2 = (0,1,1), \quad b_3 = (1,0,1) \end{gathered}$$

1. Montrer que les familles $\mathcal{A} = (a_1, a_2, a_3)$ et $\mathcal{B} = (b_1, b_2, b_3)$ sont des bases de $\mathbb{R}^3$.
2. Déterminer la matrice de passage de la base $\mathcal{A}$ à la base $\mathcal{B}$.
3. Automatiser le calcul précédent en programmant une fonction `matpassage` prenant en paramètres deux bases de $\mathbb{R}^3$ et qui renvoie la matrice de passage correspondante.

2.2.3 Jacobien

Programmer une fonction `jacobian` prenant en paramètre une liste de deux expressions en x, y et qui renvoie le jacobien de la transformation de $\mathbb{R}^2$ correspondante. Exemple d'exécution :

```
jacobian([x*y,y/x])
```

donne pour résultat : `2*y/x`.

2.2.4 Inverse et puissances d'une matrice symbolique

On considère la matrice formelle :

$$A = \begin{pmatrix} a-b-c & 2a & 2a \\ 2b & b-a-c & 2b \\ 2c & 2c & c-a-b \end{pmatrix}$$

1. A quelle condition sur a, b et c cette matrice est-elle inversible ? Calculer alors A^{-1}.
2. Montrer que $A^2 = (a+b+c)^2\,\mathbf{I}_3$. Que peut-on dire de A^n pour $n \in \mathbb{N}$?

2.2.5 Erreur de SAGE

Calculer le rang de la famille de vecteurs (v_1, v_2, v_3) où $v_1 = (\sin(1), \sin(2), \sin(3))$, $v_2 = (\sin(4), \sin(5), \sin(6))$, $v_3 = (\sin(7), \sin(8), \sin(9))$, puis le rang de la famille de vecteurs formels (w_1, w_2, w_3) où $w_1 = (\sin(x), \sin(2x), \sin(3x))$, $w_2 = (\sin(4x), \sin(5x), \sin(6x))$, $w_3 = (\sin(7x), \sin(8x), \sin(9x))$.
Conclusion ?

3 Applications linéaires

3.1 Matrice d'une application linéaire

Supposons données une base $\mathcal{B} = (e_1, e_2, e_3)$ de $\mathbb{R}^3$ (par exemple la base canonique) et l'application linéaire $f : \mathbb{R}^3 \to \mathbb{R}^3$ définie par :

$$\left\{ \begin{array}{rcrcrcr} f(e_1) & = & e_1 & + & 4\,e_2 & + & 7\,e_3 \\ f(e_2) & = & 2\,e_1 & + & 5\,e_2 & + & 8\,e_3 \\ f(e_3) & = & 3\,e_1 & + & 6\,e_2 & + & 9\,e_3 \end{array} \right.$$

La matrice de f dans la base $\mathcal{B}$ est par définition :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Programmez la fonction suivante (vous vous en resservirez plus tard), qui prend en paramètres :
– une liste d'expressions qui sont des combinaisons linéaires de variables symboliques,
– la liste de ces variables symboliques,
et qui renvoie la matrice des coefficients de ces variables à l'intérieur des expressions :

```
def genmatrix(Lexpr,Lvar) :
    n = len(Lexpr)
    p = len(Lvar)
    M = zero_matrix(SR,n,p)
    for i in range(n) :
        for j in range(p) :
            M[i,j] = Lexpr[i].coeff(Lvar[j])
    return M
```

Utilisez cette fonction pour générer la matrice A :

```
var('e1,e2,e3')
y1 = e1+4*e2+7*e3 ; y2 = 2*e1+5*e2+8*e3 ; y3 = 3*e1+6*e2+9*e3
A = genmatrix([y1,y2,y3],[e1,e2,e3]).transpose() ; A
```

3.2 Noyau, image

A partir de la matrice A, on obtient le noyau (resp. l'image) de f par la méthode `right_kernel` (resp. `column_space`). Entrez :

```
E = A.right_kernel() ; E
```

```
F = A.column_space() ; F
```

Notez que, dans l'affichage des bases de ces espaces, les coordonnées des vecteurs sont exprimées par rapport à la base $\mathcal{B}$.

4 Systèmes d'équations linéaires

Pour illustrer cette section, considérons les trois systèmes linéaires suivants :

$$(S_1)\quad \begin{cases} x_1 + 2\,x_2 + 2\,x_3 = 3 \\ 3\,x_1 + x_2 + x_3 = 2 \\ x_1 + 12\,x_2 + x_3 = 6 \end{cases}$$

$$(S_2)\quad \begin{cases} x_1 + 2\,x_2 + 2\,x_3 = 3 \\ 3\,x_1 + x_2 + x_3 = 2 \\ x_1 + 12\,x_2 + 12\,x_3 = 6 \end{cases}$$

$$(S_3)\quad \begin{cases} x_1 + 2\,x_2 + 2\,x_3 = 3 \\ 3\,x_1 + x_2 + x_3 = 2 \\ x_1 + 12\,x_2 + 12\,x_3 = 17 \end{cases}$$

4.1 Résolution matricielle

Etant donnés une matrice A et un vecteur b, la commande `A.solve_right(b)` (ou, de manière équivalente : `A\b`) rend *un* vecteur x solution du système linéaire $Ax = b$, s'il en existe.

Pour chacun des trois systèmes, entrez la matrice A, le vecteur b et résolvez le système. Pour (S_3), cela donne :

```
A = matrix(3,3,[1,2,2,3,1,1,1,12,12])
b = vector([3,2,17])
x = A\b ; x
```

On n'obtient ainsi qu'une solution particulière x du système (S_3) alors qu'il y en a une infinité (vérifiez que le rang de A est 2). Mais la solution générale est obtenue en ajoutant à x la solution générale du système homogène associé (i.e. avec second membre nul). Vous calculerez pour cela le noyau de A :

```
E = A.right_kernel() ; E
```

qui est une droite vectorielle. Extrayez-en le vecteur de base :

```
u = E.basis()[0] ; u
```

d'où la solution générale du système (S_3) :

```
var('C')
solution = C*u+x.change_ring(SR) ; solution
```

(notez que pour pouvoir additionner ces deux vecteurs, il faut changer le corps de base de x).

4.2 Résolution avec `solve`

Pour résoudre un système d'équations linéaires, on peut aussi utiliser la commande `solve` en lui passant deux paramètres :
– la liste des équations,
– la liste des inconnues.
Essayez cette méthode sur chacun des trois systèmes $(S_1), (S_2), (S_3)$.

4.3 Exercices

4.3.1 Projection orthogonale

Dans l'espace euclidien $\mathbb{R}^3$ rapporté à sa base orthonormée canonique, on considère le plan H passant par l'origine et de vecteur normal $n = (1, -2, 2)$. Déterminer la matrice M de la projection orthogonale de $\mathbb{R}^3$ sur H.

Méthode proposée :

– trouver une base (h_1, h_2) de H,
– prendre M arbitraire (voir §1.1.3) et exprimer que $Mn = 0, Mh_1 = h_1, Mh_2 = h_2$ pour obtenir un système de 9 équations linéaires en les coefficients de M,
– résoudre ce système par `solve` et conclure.

4.3.2 Matrices commutantes

Soit la matrice $A = \begin{pmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{pmatrix}$. Trouver une base de l'espace vectoriel des matrices carrées réelles M d'ordre 3 qui commutent avec A, c'est-à-dire telles que $MA - AM = 0$.

Indication : prendre une matrice M arbitraire, mettre en équations le problème, utiliser `genmatrix` (voir §3.1) et résoudre matriciellement.

4.3.3 Carrés magiques

On dit qu'une matrice carrée réelle est *magique* lorsque les sommes des coefficients de chaque ligne, de chaque colonne et de chacune des deux diagonales sont les mêmes. On appelle *carré magique d'ordre n* une matrice magique d'ordre n constituée des entiers successifs de 1 à n^2 (exemple : matrice A du §4.3.2).

1. Déterminer une base $B = (B_0, B_1, B_2)$ de l'espace vectoriel E_3 des matrices magiques d'ordre 3.
2. Programmer la fonction $f : \mathbb{R}^3 \to E_3$ qui à (x_0, x_1, x_2) associe la matrice $\sum_{i=0}^{2} x_i B_i$.
3. Programmer une fonction qui teste si une matrice M est formée des entiers de 1 à 9, c'est-à-dire si l'ensemble de ses coefficients est égal à l'ensemble $\{1, 2, \ldots, 9\}$ (*indication* : une liste `L` peut être convertie en ensemble par la commande `Set(L)`).
4. Définir la liste L des arrangements de 3 éléments pris dans $\{1, 2, \ldots, 9\}$ par la commande :

```
L = Arrangements(range(1,10),3).list()
```

5. En déduire la liste de tous les carrés magiques d'ordre 3 (on pourra se servir de `filter` et `map`) : il y en a 8.

5 Valeurs propres, vecteurs propres, diagonalisation

Soit $f : \mathbb{R}^3 \to \mathbb{R}^3$ l'application linéaire dont la matrice dans la base canonique est :

$$A = \begin{pmatrix} 4 & 1 & -4 \\ 8 & 3 & -10 \\ 5 & 1 & -5 \end{pmatrix}$$

Entrez la matrice A :

```
A = matrix(3,3,[4,1,-4,8,3,-10,5,1,-5]) ; A
```

La liste des valeurs propres de f est obtenue par la commande :

```
A.eigenvalues()
```

et la liste des sous-espaces propres associés par :

```
A.eigenspaces()
```

Vérifiez que les valeurs propres sont les racines du polynôme caractéristique de A :

```
var('x')
H = A.charpoly(x) ; print H
H.roots()     # donne les racines de H avec leur multiplicité
```

On constate que A est diagonalisable sur $\mathbb{R}$ (pourquoi ?). Construisez une base de $\mathbb{R}^3$ constituée de vecteurs propres . Pour cela, entrez la commande :

```
V = A.eigenvectors_right() ; V
```

qui donne dans `V` une liste de triplets (*valeur propre*, *liste de vecteurs*, *multiplicité*), la liste de vecteurs étant une base du sous-espace propre associé. D'où la base cherchée :

```
B = [V[i][1][0] for i in range(3)] ; B
```

Calculez la matrice de passage P de la base canonique à la base B :

```
P = matrix(B).transpose() ; P
```

et vérifiez que, dans la base B, la matrice Δ de f est diagonale :

```
Delta = P^-1 * A * P ; Delta
```

Les matrices Δ et P peuvent s'obtenir directement en utilisant la *réduite de Jordan* de A. Entrez :

```
Delta, P = A.jordan_form(transformation=True)
Delta ; P
```

Soit à calculer la matrice A^n (n entier naturel). On a $A = P\Delta P^{-1}$, d'où $A^n = P\Delta^n P^{-1}$. Calculez d'abord Δ^n en appliquant la fonction $x \mapsto x^n$ à la liste des coefficients de Δ (qui est une matrice diagonale) :

```
var('n')
L = map(lambda x : x^n, Delta.list())
Deltan = matrix(3,3,L).simplify() ; Deltan
```

et déduisez-en A^n :

```
An = P * Deltan * P^-1 ; An
```

Vérifiez la formule par exemple pour $n = 50$:

```
An(n=50) - A^50
```

6 Exercices

6.1 Reconnaître une matrice diagonalisable

Dire si les matrices suivantes sont diagonalisables sur $\mathbb{R}$:

$$A = \begin{pmatrix} 13 & 16 & 16 \\ -5 & -7 & -6 \\ -6 & -8 & -7 \end{pmatrix}, \; B = \begin{pmatrix} -1 & 1 & -1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \; C = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \; D = \begin{pmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{pmatrix}$$

6.2 Récurrence linéaire

Soient (u_n) et (v_n) deux suites de nombres réels définies par la donnée de leurs premiers termes u_0, v_0 et les formules de récurrence :

$$\begin{cases} u_{n+1} &= \frac{9}{10}\, u_n &+ \frac{2}{5}\, v_n \\ v_{n+1} &= \frac{1}{10}\, u_n &+ \frac{3}{5}\, v_n \end{cases}$$

1. Calculer u_n et v_n en fonction de u_0, v_0 et n.
   Indication : pour n arbitraire, déterminer la puissance $n^{\text{ième}}$ de la matrice :
   $$A = \begin{pmatrix} \frac{9}{10} & \frac{2}{5} \\ \frac{1}{10} & \frac{3}{5} \end{pmatrix}$$
2. En déduire $\lim_{n\to+\infty} u_n$, $\lim_{n\to+\infty} v_n$ et $\lim_{n\to+\infty} \frac{u_n}{v_n}$.

6.3 Système différentiel linéaire du premier ordre

Soient y_1 et y_2 deux fonctions d'une variable réelle x. On considère le système différentiel :

$$(S) \quad \left\{ \begin{array}{rcrcrcl} y_1' & = & 5\,y_1 & - & 3\,y_2 & + & \sin(x) \\ y_2' & = & 6\,y_1 & - & 6\,y_2 & + & \cos(x) \end{array} \right.$$

1. En posant :

$$Y = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}\ ,\ Y' = \begin{pmatrix} y_1'(x) \\ y_2'(x) \end{pmatrix}\ ,\ A = \begin{pmatrix} 5 & -3 \\ 6 & -6 \end{pmatrix}\ ,\ B = \begin{pmatrix} \sin(x) \\ \cos(x) \end{pmatrix}$$

   le système (S) s'écrit matriciellement : $Y' = A\,Y + B$. Trouver une matrice carrée d'ordre 2 inversible P telle que la matrice $\Delta = P^{-1}AP$ soit diagonale.

2. Remarquer que le système (S) équivaut au système différentiel (S') : $P^{-1}Y' = \Delta P^{-1}Y + P^{-1}B$. Soient z_1, z_2 deux fonctions de x telles que :

$$\begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} = P^{-1}Y\ ,\ \begin{pmatrix} z_1'(x) \\ z_2'(x) \end{pmatrix} = P^{-1}Y'$$

   Ecrire le système (S') : on obtient deux équations différentielles, l'une en z_1, l'autre en z_2. Résoudre ces équations séparément par `desolve`.

3. En déduire les solutions du système (S).

6.4 Réduction d'une conique

On considère la conique (C) du plan $\mathbb{R}^2$ définie par l'équation $f(x,y) = 0$ avec :

$$f(x,y) = 6\,x^2 - 6\,xy + 9\,y^2 + 14\,x - 100\,y + 185$$

1. Tracer (C) grâce à `implicit_plot`. Quelle est la nature de (C) ?
2. Trouver le centre de symétrie de (C) en résolvant le système d'équations :

$$\frac{\partial f}{\partial x}(x,y) = 0\ ,\ \frac{\partial f}{\partial y}(x,y) = 0$$

3. Par changement de repère, trouver l'équation réduite de (C).

6.5 Problème des milliardaires

Trois milliardaires X, Y et Z décident d'égaliser leurs fortunes de la manière suivante : X versera la moitié de sa fortune à Y, puis Y versera la moitié de sa fortune à Z, puis Z versera la moitié de sa fortune à X. Ensuite, ils renouvelleront ce cycle de trois opérations jusqu'à ce que leurs fortunes se stabilisent. Lequel des trois milliardaires a proposé cette méthode de partage saugrenue ?

Indication : en notant x_n, y_n, z_n les fortunes respectives de X, Y, Z après n cycles de transactions, calculer x_n, y_n, z_n en fonction de x_0, y_0, z_0 et n, puis passer à la limite quand $n \to +\infty$.