Bachelor en Sciences et Ingénierie, filière Mathématiques

Mathématiques expérimentales

Semestre d'été 2019-2020

Objets magiques sur les entiers naturels

Auteurs :
Arjanita Dingu
Clara Duchossois

Superviseurs :
Prof. Dr. Gabor Wiese
Guendalina Palmirotta

Table des matières

1 Introduction

Au cours de ce rapport, nous nous intéressons à l'étude d'objets dits "magiques". Les objets magiques sont des représentations géométriques constituées d'entiers strictement positifs dont les sommes, sous certaines conditions, sont égales.

Dans un premier temps, nous étudions les carrés magiques, puis, nous généralisons notre étude à des objets géométriques plus divers. Pour chacun des objets considérés, nous précisons les conditions nécessaires pour le rendre magique, puis établissons le système d'équations linéaires correspondant, que l'on résout ensuite avec un logiciel. Cela permet finalement de programmer un algorithme dans le but de trouver une liste d'exemples et de donner une visualisation concrète de chaque objet magique, que nous dressons alors.

2 Carrés magiques

De manière générale, les *"carrés magiques"* sont des carrés divisés en un nombre entier $n \geq 3$ de cases contenant chacune un unique entier naturel non nul, lesquels sont tous distincts deux à deux au sein du carré.

Définition 2.1.

(a) Un carré est dit *magique* lorsque la somme des nombres situés sur chaque ligne, chaque colonne et chaque diagonale principale est identique. Cette somme correspond à ce qu'on appelle la *somme* ou *constante magique* du carré, que l'on notera par la suite par M.

(b) Un carré magique est dit d'ordre n s'il est découpé en $n \times n$ cases.

(c) Un carré magique d'ordre $n \in \mathbb{N}_{\geq 3}$ est dit *normal* s'il est constitué des n^2 premiers entiers. La constante magique y étant associée est alors

$$M = \frac{n(n^2+1)}{2}.$$

2.1 Carrés magiques 3×3

2.1.1 Paramétrisation

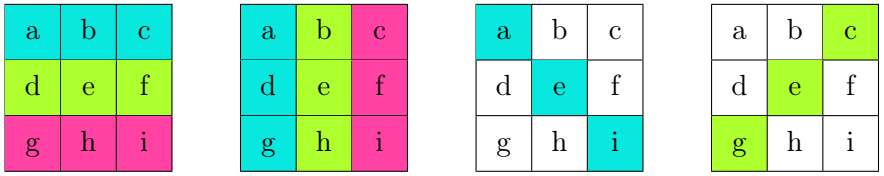
Soient $a, b, c, d, e, f, g, h, i \in \mathbb{N}$ des nombres entiers distincts.
Nous cherchons à construire un carré magique de taille 3×3 (on dit aussi d'ordre 3) que nous définissons de façon générale de la manière suivante :

a	b	c
d	e	f
g	h	i

avec,

$$a+b+c = d+e+f = g+h+i = a+d+g = b+e+h = c+f+i = a+e+i = g+e+c.$$

Ces relations d'égalité entre les différentes lignes, colonnes et diagonales du carré magique d'ordre 3 peuvent se visualiser de la manière suivante :



Ici, les couleurs permettent de représenter les sommes entre les différents entiers du carré (au sein d'une figure, les entiers surlignés de la même couleur sont sommés entre eux), ainsi que l'égalité entre ces mêmes sommes (les différentes lignes colorées sont toutes de somme égale, correspondant par définition à la somme magique M du carré).

Proposition 2.2. *Soit M la somme magique du carré de taille 3×3 à construire. On a alors*

$$M = 3e, \quad \forall e \in \mathbb{N}.$$

Démonstration. On a par définition :

$$\begin{cases} b+h &= M-e, \\ d+f &= M-e, \\ g+c &= M-e, \\ a+i &= M-e. \end{cases}$$

C'est-à-dire :

$$\begin{aligned} & a+b+c+d+f+g+h+i &&= 4(M-e) \\ \iff & \underbrace{a+b+c}_{M} + \underbrace{d+e+f}_{M} + \underbrace{g+h+i}_{M} &&= 4(M-e)+e \\ \iff & 3M &&= 4M-3e \\ \iff & M &&= 3e. \end{aligned}$$

□

Ainsi, afin de construire un carré magique de taille 3×3, il faut résoudre le système d'équations linéaires suivant :

$$\begin{cases} a+b+c &= 3e, \\ d+e+f &= 3e, \\ g+h+i &= 3e, \\ a+d+g &= 3e, \\ b+e+h &= 3e, \\ c+f+i &= 3e, \\ a+e+i &= 3e, \\ g+e+c &= 3e. \end{cases}$$

Ce système d'équations linéaires peut également être exprimé sous forme matricielle :

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}}_{=:P} \underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix}}_{=:x} = 3 \begin{pmatrix} e \\ e \\ e \\ e \\ e \\ e \\ e \\ e \end{pmatrix}.$$

À l'aide du logiciel SageMath, nous trouvons qu'une base du noyau de la matrice P est :

$$W := \langle w_0 \, , \, w_1 \rangle = \langle \begin{pmatrix} 1 \\ 0 \\ -1 \\ -2 \\ 0 \\ 2 \\ 1 \\ 0 \\ -1 \end{pmatrix} , \begin{pmatrix} 0 \\ 1 \\ -1 \\ -1 \\ 0 \\ 1 \\ 1 \\ -1 \\ 0 \end{pmatrix} \rangle.$$

De plus, nous obtenons :

$$x = \begin{pmatrix} 2 \\ 2 \\ -1 \\ -2 \\ 1 \\ 4 \\ 3 \\ 0 \\ 0 \end{pmatrix} \times e.$$

Ainsi, les solutions du système sont de la forme $x + \alpha \cdot w_0 + \beta \cdot w_1$, $\forall \alpha, \beta \in \mathbb{R}$. On remarque alors que l'élément à la cinquième position du vecteur x, lequel correspond à la variable e, est égal à 1. De plus, on observe que les éléments à la cinquième position des vecteurs w_0 et w_1 sont eux égaux à 0. Ainsi, on

peut écrire e de la manière suivante :

$$e = x - 2 \cdot w_0 - 2 \cdot w_1 = \begin{pmatrix} 0 \\ 0 \\ 3 \\ 4 \\ 1 \\ -2 \\ -1 \\ 2 \\ 2 \end{pmatrix}.$$

Par ailleurs, le vecteur w_0 étant de la forme $(1, 0, *, *, *, *, *, *, *)^T$, nous l'associons à la valeur a, et définissons $a = w_0$.
Suivant le même raisonnement, le vecteur w_1 étant de la forme $(0, 1, *, *, *, *, *, *, *)^T$, nous définissons $b = w_1$.
Finalement, on a :

$$e = x - 2 \cdot w_0 - 2 \cdot w_1 = x - 2 \cdot a - 2 \cdot b = \begin{pmatrix} 0 \\ 0 \\ 3 \\ 4 \\ 1 \\ -2 \\ -1 \\ 2 \\ 2 \end{pmatrix}.$$

On remarque ici que les variables pivots du vecteur solution sont a et b.

Ainsi, tout carré magique de taille 3×3 peut être défini uniquement à partir des variables a et b et d'une constante magique $M = 3e$ donnée. Cela nous permet de conclure que tout carré magique d'ordre 3 peut s'écrire comme suit, pour $a, b, e \in \mathbb{N}$:

a	b	$3\mathbf{e} - (\mathbf{a} + \mathbf{b})$
$4\mathbf{e} - (2\mathbf{a} + \mathbf{b})$	e	$-2\mathbf{e} + 2\mathbf{a} + \mathbf{b}$
$-\mathbf{e} + \mathbf{a} + \mathbf{b}$	$2\mathbf{e} - \mathbf{b}$	$2\mathbf{e} - \mathbf{a}$

2.1.2 Implémentation dans Python

Nous sommes à présent en mesure de créer un programme nous permettant de créer des carrés magiques à partir de nombres entiers aléatoires. L'objectif est alors que le programme soit à même de déterminer si un carré magique d'ordre 3 peut être donné à partir de 3 paramètres a, b et e générés aléatoirement, et compris dans un intervalle défini, et que lorsque les conditions requises sont remplies, il nous retourne le carré magique correspondant.

Le programme Python correspondant se trouve ci-dessous :

```
1  import random
2  import math
3  import pickle #ce module nous permettra d'enregistrer la liste
       de tous les carrés magiques construits
4
5  L=[] #on crée une liste vide dans laquelle nous enregistrerons
       plus tard les carrés magiques complets que nous trouverons
6
7  #on définit e, a et b comme étant des entiers positifs choisis
       aléatoirement entre 1 et 1000
8  e = int(random.randint(1,1000))
9  print("e=",e)
10
11 a = int(random.randint(1,1000))
12 print("a=",a)
13
14 b = int(random.randint(1,1000))
15 print("b=",b)
16
17 def magicsquare(e,a,b): #la fonction magicsquare prend comme
       arguments les entiers e, a et b
18
19     #on définit des entiers c, d, f, g, h, i qui n'ont pas
       encore de valeur asscociée
20     c = int()
21     d = int()
22     f = int()
23     g = int()
24     h = int()
25     i = int()
26
27     #la constante magique de notre carré est égale à 3 fois la
       valeur de e
28     M = 3*e
29     print("M=",M)
30
31     #on vérifie d'abord que a, b et e sont distincts deux à
       deux
32     if a!=b and a!=e and b!=e:
33         if M>(a+b): #si M est strictement supérieur à a+b, on
       assigne à c la valeur M-(a+b), et on l'affiche
34             c = M-a-b
35             print("c=",c)
```

```
36              if M>(b+e): #on vérifie que la valeur de M est
    strictement supérieure à la somme de b et e
37                  h = M-b-e
38                  print("h=",h)
39                  #on continue de vérifier ainsi que les valeurs
    des composantes de notre carré magique sont strictement
    positives
40                  if M>(a+e):
41                      i= M-a-e
42                      print("i=",i)
43                      if M>(c+e):
44                          g = M-c-e
45                          print("g=",g)
46                          if M>(c+i):
47                              f = M-c-i
48                              print("f=",f)
49                              if M>(e+f):
50                                  d = M-e-f
51                                  print("d=",d)#si cette étape
    est atteinte, le carré magique est complet, et on peut l'
    imprimer en entier
52
53                                  row1=[a,b,c]
54                                  row2=[d,e,f]
55                                  row3=[g,h,i]
56
57                                  print(row1)
58                                  print(row2)
59                                  print(row3)
60
61                                  #on ajoute finalement les
    lignes de notre carré à la liste L
62                                  L.append(row1)
63                                  L.append(row2)
64                                  L.append(row3)
65                                  L.append("/")
66
67                                  #cette commande nous permet de
    sauvegarder la liste L dans un autre fichier
68                                  with open('MS', 'wb') as
    MagicSquares:
69                                      pickle.dump(L, MagicSquares
    )
70
71                              else: #si M est inférieur ou égal à
     e+f, on ne peut pas contruire un carré magique valide
72                                  print("Cannot create a Magic
    Square")
73                      #de la même manière, si une des
    conditions requises ci-dessus ne peut être remplie, on
    retourne un message d'erreur
74                      else:
75                          print("Cannot create a Magic Square
    ")
```

```
76                      else:
77                          print("Cannot create a Magic Square")
78                  else:
79                      print("Cannot create a Magic Square")
80              else:
81                  print("Cannot create a Magic Square")
82          else:
83              print("Cannot create a Magic Square")
84      else:
85          print("Cannot create a Magic Square")
86
87 #cette commande nous permet de charger la liste L préalablement
       enregistrée afin de l'imprimer si nécessaire
88 with open('MS', 'rb') as f:
89      L = pickle.load(f)
```

Ainsi, à la ligne 4 de l'algorithme, nous définissons une fonction que nous appelons "magicsquare". Au sein de celle-ci, nous définissons nos trois variables e, a et b, que nous associons à trois entiers déterminés aléatoirement entre 1 et 1000. Les lignes 14 à 19 servent simplement à préciser que les autres variables avec lesquelles nous allons travailler sont elles aussi des nombres entiers. À la ligne 21, nous définissons la constante magique du carré comme étant égale à 3 fois la valeur de e, comme nous avons préalablement vu que cela est nécessairement le cas pour tout carré magique d'ordre 3.

Par la suite, nous établissons les conditions devant être respectées afin de former un carré magique à partir de nos variables. Ici, nous procédons étape par étape, en soustrayant successivement les valeurs des entiers déjà à notre disposition jusqu'à ce que les 9 valeurs constituant notre carré magique puissent être déduites. Dans un premier temps, à la ligne 21, nous vérifions que les entiers a, b et e sont distincts deux à deux. Si cette condition n'est déjà pas vérifiée, l'algorithme prend fin en affichant un message d'erreur. Autrement, nous continuons en vérifiant que la somme des entiers a et b est bien strictement inférieure à la somme magique devant être atteindre. Lorsque c'est le cas, nous assignons la valeur de la différence entre M, a et b à c. De la même manière, nous vérifions ensuite successivement que la somme des entiers b et e, a et e, c et e, c et i, et enfin e et f est à chaque fois inférieure à la somme magique M. Auquel cas il est une fois de plus évident que nous ne pourrons additionner un nouvel entier à cette somme déjà supérieure à M pour atteindre la valeur de M. Le raisonnement "en cascade" que nous appliquons ici requiert alors que chacune des conditions préalablement posées soit respectée, afin de trouver les nouvelles valeurs inconnues des entiers c, d, f, g, h, et i. Si ce n'est pas le cas pour une de ces conditions, le programme prend systématiquement fin et affiche à l'utilisateur qu'un carré magique ne peut être créé. Les lignes 43 à 56 de l'algorithme retranscrivent cet aspect du programme, puisqu'elles consistent à imprimer le message `"Cannot create a magic square"` si la condition établie au sein du "if-statement" correspondant ne peut être remplie.

Remarque 2.3. Dans cet algorithme, nous faisons mention de l'utilisation du module Python `pickle`. Les commandes utilisées en lien avec ce module ne seront plus ajoutées dans les autres algorithmes de ce rapport, mais il est à noter que le recours y a été systématique afin d'enregister les diverses listes d'objets magiques complétés.

2.1.3 Visualisation

Voici quelques exemples de carrés magiques d'ordre 3 obtenus grâce à notre programme Python donné au sein de la section 2.1.2.

63	31	38
19	44	69
50	57	25

FIGURE 1 – Carré magique d'ordre 3 et de somme magique $M = 152$.

250	506	444
594	400	206
356	294	550

FIGURE 2 – Carré magique d'ordre 3 et de somme magique $M = 1200$.

2.1.4 Carré magique normal d'ordre 3

À présent, notre objectif est de construire un carré magique normal d'ordre 3. Par la définition 2.1 un carré magique 3×3 est constitué des entiers allant de 1 à 9 inclus, et a pour somme magique $M = 15$.

Dès lors, nous pouvons exploiter les résultats obtenus grâce à SageMath afin de trouver un tel carré magique. On sait qu'un carré magique d'ordre 3 a pour somme magique 15 si et seulement si son élément central e est égal à 5 (Prop. 2.2). Il s'agira alors de tester différentes combinaisons linéaires de a et b, qui ajoutées à $5e$ donnent un carré magique non seulement à entrées strictement positives et distinctes, mais également composé des premiers entiers de 1 à 9. Voici des exemples de combinaisons linéaires testées au moyen du logiciel SageMath dans le but de trouver un carré magique normal :

```
a = W[0]
b = W[1]
a+2*b+5*e
```

```
(1, 2, 12, 16, 5, -6, -2, 8, 9)
```

```
2*a+3*b+5*e
```

```
(2, 3, 10, 13, 5, -3, 0, 7, 8)
```

```
3*a+4*b+5*e
```

```
(3, 4, 8, 10, 5, 0, 2, 6, 7)
```

```
4*a+6*b+5*e
```

```
(4, 6, 5, 6, 5, 4, 5, 4, 6)
```

```
6*a+7*b+5*e
```

```
(6, 7, 2, 1, 5, 9, 8, 3, 4)
```

On remarque alors que lorsque l'on fixe $a = 6, b = 7$ et $e = 5$, on obtient effectivement un carré magique normal d'ordre 3, dont on peut donner une visualisation :

6	7	2
1	5	9
8	3	4

FIGURE 3 – Carré magique normal d'ordre 3.

De la même manière, nous sommes en mesure de trouver toutes les configurations possibles pour les carrés magiques normaux d'ordre 3 :

```
8*a+1*b+5*e
    (8, 1, 6, 3, 5, 7, 4, 9, 2)

8*a+3*b+5*e
    (8, 3, 4, 1, 5, 9, 6, 7, 2)

6*a+1*b+5*e
    (6, 1, 8, 7, 5, 3, 2, 9, 4)

4*a+3*b+5*e
    (4, 3, 8, 9, 5, 1, 2, 7, 6)

4*a+9*b+5*e
    (4, 9, 2, 3, 5, 7, 8, 1, 6)

2*a+7*b+5*e
    (2, 7, 6, 9, 5, 1, 4, 3, 8)

2*a+9*b+5*e
    (2, 9, 4, 7, 5, 3, 6, 1, 8)
```

Finalement, nous pouvons représenter toutes les autres permutations possibles pour le carré magique d'ordre 3 :

8	1	6	8	3	4	6	1	8	4	3	8
3	5	7	1	5	9	7	5	3	9	5	1
4	9	2	6	7	2	2	9	4	2	7	6

4	9	2	2	7	6	2	9	4
3	5	7	9	5	1	7	5	3
8	1	6	4	3	8	6	1	8

FIGURE 4 – Permutations des carrés magiques normaux d'ordre 3.

2.2 Carrés magiques 4×4

2.2.1 Paramétrisation

Soient $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p \in \mathbb{N}$ des nombres entiers distincts.
Nous cherchons à construire un carré magique d'ordre 4 que nous définissons

de façon générale de la manière suivante :

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

avec M la constante magique du carré, définie telle que :

$$\begin{cases} a+b+c+d &= M, \\ e+f+g+h &= M, \\ i+j+k+l &= M, \\ m+n+o+p &= M, \\ a+e+i+m &= M, \\ b+f+j+n &= M, \\ c+g+k+o &= M, \\ d+h+l+p &= M, \\ a+f+k+p &= M, \\ m+j+g+d &= M. \end{cases}$$

Ces relations d'égalité entre les différentes lignes, colonnes et diagonales du carré magique d'ordre 4 peuvent se visualiser de la manière suivante :

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Afin d'être résolu, ce système d'équations linéaires peut également être

exprimé sous forme matricielle :

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{=:P} \underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \\ m \\ n \\ o \\ p \end{pmatrix}}_{=:x} = \begin{pmatrix} M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \end{pmatrix}.$$

De la même manière qu'au sein de la section 2.1.1 sur les carrés magiques d'ordre 3, nous trouvons au moyen de SageMath qu'une base du noyau de la matrice P est :

$$W := \left\langle \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ -2 \\ 0 \\ -1 \\ -2 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ -2 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ -1 \\ 0 \\ 1 \\ 0 \\ 1 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \right\rangle.$$

De plus, nous obtenons comme solution :

$$x = \begin{pmatrix} -1 \\ -1 \\ 2 \\ 1 \\ 0 \\ 2 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \times M.$$

En analysant simultanément les lignes et les colonnes données par les vecteurs colonnes composant la base W, on remarque que la première ligne se compose uniquement de 0 en dehors de l'élément en première position, où se trouve un 1. Or cette position correspond à celle de l'élément a dans le premier vecteur colonne. Cela nous permet alors de dire que a est une variable libre de notre système.

On peut faire un constat similaire au niveau des seconde, troisième, cinquième, sixième, septième, et neuvième lignes. Nous pouvons donc encore avancer que b, c, e, f, g et i sont les autres variables libres de notre système d'équations linéaires.

Ainsi, tout carré magique de taille 4×4 peut être défini uniquement à partir des variables entières a, b, c, e, f, g et i et d'une constante magique M donnée. Cela nous permet de conclure que tout carré magique d'ordre 4 peut s'écrire comme suit :

a	b	c	M−(a + b + c)
e	f	g	M−(e + f + g)
i	$-M+$ 2a + b + c + e − g + i	$2M-$ 2a − b − c − e − f − i	f + g − i
M−(a + e + i)	$2M-$ 2a−2b− c − e − f + g − i	$-M+$ 2a + b + e + f − g + i	$-M+$ a + b + c + e + i

Cette paramétrisation peut être trouvée à la fois à partir de calculs élémentaires reposant sur la définition d'un carré magique et de sa somme M, ainsi qu'à partir des résultats obtenus avec SageMath. En effet, on observe dans le vecteur solution x qu'à la douzième ligne, correspondant à la variable l, se trouve un 0. Ainsi on peut choisir de ne pas exprimer cette variable en fonction de notre somme magique M. De plus, on remarque que la douzième ligne de la base W, elle aussi correspondant à la variable l, est la suivante :

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{pmatrix}.$$

Or, rappelons que la première colonne est associée à la première variable libre de notre système (ici, a), et ainsi de suite. Cela nous permet finalement d'écrire $l = f + g - i$, et de dresser la paramétrisation ci-dessus.

En outre, en analysant la paramétrisation finale trouvée, deux nouvelles relations entre les entiers constituant un carré magique d'ordre 4 peuvent être données. En effet, comme $l = f + g - i$, on a :

$$l + i = f + g.$$

Il en est de même pour les entiers e, h, j et k, c'est-à-dire qu'on a :

$$e + h = j + k.$$

Nous obtenons également des relations identiques par réfléxion et rotation du carré, ce qui donne :

$$b + n = g + k,$$

$$c + o = f + j.$$

Ces relations d'égalité peuvent alors se visualiser de la manière suivante :

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

2.2.2 Implémentation dans Python

Grâce à la paramétrisation complète du carré magique d'ordre 4 obtenue, nous sommes en mesure d'implémenter un algorithme dans Python qui nous permettrait de construire des carrés magiques 4×4 à partir d'entiers aléatoirement choisis. Ces entiers joueraient le rôle de nos variables libres précédemment déterminées, ainsi que celui de notre somme magique M.

Le programme Python en question se trouve ci-dessous :

```
import random
import math

#on définit la constante magique de notre carré comme étant un entier aléatoirement choisi entre 1 et 600
M = int(random.randint(1,600))
print("M=",M)

#on définit les varaibles libres de notre carré comme étant des entiers positifs choisis aléatoirement entre 1 et 300
a = int(random.randint(1,300))
print("a=",a)

b = int(random.randint(1,300))
print("b=",b)

c = int(random.randint(1,300))
print("c=",c)

e = int(random.randint(1,300))
print("e=",e)

f = int(random.randint(1,300))
print("f=",f)

g = int(random.randint(1,300))
print("g=",g)
```

```
26
27 i = int(random.randint(1,300))
28 print("i=",i)
29
30 def magicsquare(M,a,b,c,e,f,g,i): #la fonction magicsquare
       prend comme arguments les entiers M, a, b, e, f, g, et i
31
32     #on définit des entiers d, h, j, k, l, m, n, o, p qui n'ont
        pas encore de valeur asscociée
33     d = int()
34     h = int()
35     j = int()
36     k = int()
37     l = int()
38     m = int()
39     n = int()
40     o = int()
41     p = int()
42
43     #on vérifie d'abord que a, b, c, e, f, g et i sont
       distincts deux à deux
44     if a!=b and b!=c and a!=c and a!=e and a!=f and a!=g and a
       !=i and b!=e and b!=f and b!=g and b!=i and c!=e and c!=f
       and c!=g and c!=i and e!=f and e!=g and e!=i and f!=g and f
       !=i and g!=i:
45         if M>(a+b+c): #si M est strictement supérieur à a+b+c,
       on assigne à d la valeur M-(a+b+c), et on l'affiche
46             d = M-a-b-c
47             print("d=",d)
48             if M>(e+f+g): #on vérifie que la valeur de M est
       strictement supérieure à la somme de e, f et g
49                 h = M-e-f-g
50                 print("h=",h)
51                 #on continue de vérifier ainsi que les valeurs
       des composantes de notre carré magique sont strictement
       positives
52                 if M>(a+e+i):
53                     m= M-a-e-i
54                     print("m=",m)
55                     if M>(d+g+m):
56                         j = M-d-g-m
57                         print("j=",j)
58                         if M>(b+f+j):
59                             n = M-b-f-j
60                             print("n=",n)
61                             if M>(f+g-i) and (f+g-i)>0:
62                                 l = f+g-i
63                                 print("l=",l)
64                                 if M>(i+j+l):
65                                     k = M-i-j-l
66                                     print("k=",k)
67                                     if M>(c+g+k):
68                                         o = M-c-g-k
69                                         print("o=",o)
```

```
70                                          if M>(m+n+o):
71                                              p = M-m-n-o
72                                              print("p=",p) #si
    cette étape est atteinte, le carré magique est complet, et
    on peut l'imprimer en entier
73
74                                              row1=[a,b,c,d]
75                                              row2=[e,f,g,h]
76                                              row3=[i,j,k,l]
77                                              row4=[m,n,o,p]
78
79                                              print(row1)
80                                              print(row2)
81                                              print(row3)
82                                              print(row4)
83
84                                          else: #si M est infé
    rieur ou égal à m+n+o, on ne peut pas contruire un carré
    magique valide
85                                              print("Cannot
    create a Magic Square")
86                                      #de la même manière, si une
     des conditions requises ci-dessus ne peut être remplie, on
     retourne un message d'erreur
87                                      else:
88                                          print("Cannot create a
    Magic Square")
89                                  else:
90                                      print("Cannot create a
    Magic Square")
91                              else:
92                                  print("Cannot create a Magic
    Square")
93                          else:
94                              print("Cannot create a Magic Square
    ")
95                      else:
96                          print("Cannot create a Magic Square")
97                  else:
98                      print("Cannot create a Magic Square")
99              else:
100                 print("Cannot create a Magic Square")
101         else:
102             print("Cannot create a Magic Square")
103     else:
104         print("Cannot create a Magic Square")
```

En premier lieu, des lignes 5 à 28, nous définissons les variables libres entières aléatoires avec lesquelles nous allons travailler. Les entiers a, b, c, e, f, g et i sont tous compris entre 1 et 300, tandis que la somme magique de notre futur carré est aléatoirement choisie entre 1 et 600. Un tel choix a été fait de sorte à augmenter nos chances de trouver un carré magique viable, les entiers le constituant ainsi que leurs sommes devant nécessairement être inférieurs

à la valeur de M.

À la ligne 30, nous définissons une fonction appelée `magicsquare`, prenant comme arguments les variables libres et la somme magique M précédemment déclarées. Cette fonction servira à déterminer si un carré magique 4×4 est constructible à partir des variables libres et de la somme M aléatoirement données, ainsi qu'à donner ce carré magique lorsque cela est possible. Avant toute chose, nous devons déclarer les variables liées de notre système en tant qu'entiers positifs, ce que nous faisons des lignes 33 à 43. Une fois les entiers d, h, j, k, l, m, n, o et p déclarés, nous commençons par vérifier que les variables libres a, b, c, e, f, g sont toutes distinctes deux à deux. Si ce n'est pas le cas, nous arrêtons d'ores et déjà le programme, et affichons un message d'erreur indiquant qu'un tel carré magique ne peut être construit (ligne 103). Si les variables libres sont effectivement distinctes entre elles, nous continuons, ligne 45, en vérifiant que la somme magique M est strictement supérieure à la somme de a, b et c, auquel cas, nous attribuons à d, ligne 46, la valeur de $M - (a + b + c)$. Ainsi, cela nous garantit que d est un entier positif dont la valeur correspond bien à celle donnée dans la paramétrisation d'un carré magique d'ordre 4. Si jamais la somme $a + b + c$ est supérieure ou égale à M, nous retournons le message d'erreur `"Cannot create a Magic Square"` (ligne 101) et mettons fin au processus algorithmique en cours. De la même manière, nous vérifions successivement, des lignes 48 à 60, que les sommes des entiers $e + f + g$, $a + e + i$, $d + g + m$ et $b + f + j$ sont elles aussi strictement inférieures à M, auquel cas on assigne leur valeur à la variable leur étant associée dans la paramétrisation que nous avons donnée. Autrement, nous retournons à chaque fois un message d'erreur avant de mettre fin à l'algorithme.

Si chacune de ces conditions a pu être vérifiée avec succès, nous nous chargeons alors, à la ligne 61, de vérifier que l'entier l, égal à $f + g - i$ par définition, peut être trouvé. Pour ce faire, nous nous assurons que la valeur de $f + g - i$ est d'une part strictement positive, et d'autre part strictement inférieure à M, car dans le cas contraire, nous ne pourrions évidemment pas sommer l avec les autres entiers positifs du carré magique afin d'atteindre exactement la valeur M. Dans le cas où ces conditions peuvent être vérifiées avec succès, nous finissons par contrôler que les sommes $i + j + l$, $c + g + k$ et $m + n + o$ sont toutes inférieures à la valeur de M, et comme avant, nous arrêtons le programme si une de ces conditions n'est pas remplie.

Finalement, lorsque tous les entiers du carré magique ont pu être trouvés en répondant aux conditions énoncées, nous imprimons le carré correspondant, ce à quoi correspondent les lignes 74 à 82 du programme présenté.

2.2.3 Visualisation

Voici à présent quelques exemples de carrés magiques d'ordre 4 ayant pu être obtenus grâce au programme Python présenté dans la partie 2.2.2.

109	153	198	59
259	31	93	136
103	319	76	21
48	16	152	303

FIGURE 5 – Carré magique d'ordre 4 et de somme magique $M = 519$.

81	77	33	146
64	94	59	120
88	28	156	65
104	138	89	6

FIGURE 6 – Carré magique d'ordre 4 et de somme magique $M = 337$.

3 Objets magiques

Dans cette nouvelle section, nous cherchons désormais à étendre le raisonnement développé dans le cadre de l'étude des carrés magiques à de nouvelles figures géométriques. Nous nous attacherons donc à définir des objets magiques autres que des carrés, et prendrons à chaque fois le soin de définir ces objets d'un point de vue géométrique, ainsi que de préciser les conditions considérées nous permettant d'en faire un objet bel et bien magique.

Ainsi, tous les objets magiques que nous définissons par la suite sont constitués d'entiers strictement positifs, et sont également caractérisés par une somme magique M.

L'aspect "magique" d'une construction géométrique se base sur des conditions similaires, voire identiques à celles nécessaires à la construction d'un carré magique. De manière générale, cela se traduit donc par des sommes d'éléments égales entre elles, à l'instar des carrés magiques précédemment étudiés.

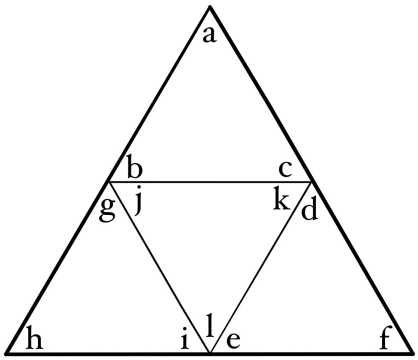
3.1 Tétraèdres magiques

Les tétraèdres magiques sont des tétraèdres dont chaque face et chaque sommet sont constitués de trois entiers dont la somme est à chaque fois égale.

Nous cherchons dans un premier temps à en donner une paramétrisation exacte et concrète.

3.1.1 Paramétrisation

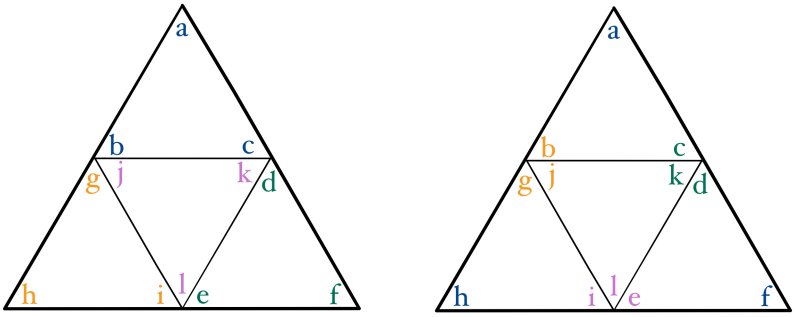
Soient $a, b, c, d, e, f, g, h, i, j, k, l \in \mathbb{N}$ des nombres entiers distincts. On représente de manière générale et systématique un tétraèdre magique comme suit :



avec M la constante magique du tétraèdre, définie telle que :

$$\begin{cases} a+b+c &= M, \\ d+e+f &= M, \\ g+h+i &= M, \\ j+k+l &= M, \\ a+f+h &= M, \\ b+g+j &= M, \\ c+d+k &= M, \\ e+i+l &= M. \end{cases}$$

Ces relations d'égalité entre les différents sommets et faces d'un tétraèdre magique peuvent être visualisées de la manière suivante :



Dans chacune de ces figures, les sommes des entiers d'une même couleur sont toutes égales entre elles.

Dans le but de résoudre le système d'équations linéaires donné ci-dessus, nous l'exprimons sous forme matricielle :

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}}_{=:P} \underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \end{pmatrix}}_{=:x} = \begin{pmatrix} M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \end{pmatrix}.$$

Le logiciel SageMath nous donne alors comme base du noyau de P la base W suivante :

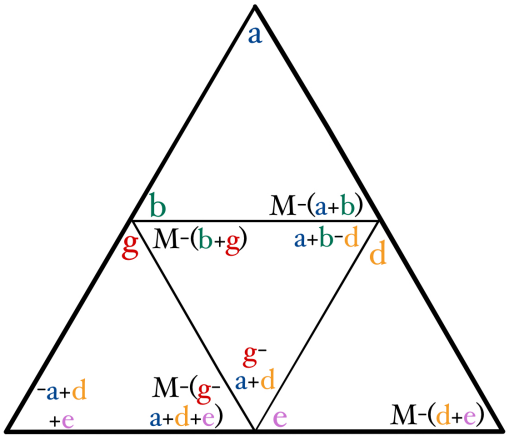
$$W := \langle \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \rangle.$$

On trouve également comme solution du système :

$$x = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \times M.$$

En analysant les lignes et le colonnes formées par les vecteurs de la base W de la même manière que nous l'avons fait pour le carré magique d'ordre 4, nous en venons à la conclusion que les variables libres du système d'équations linéaires d'un tétraèdre magique sont a, b, d, e, g.

Finalement, nous pouvons construire tout tétraèdre magique à partir des entiers a, b, d, e, g et d'une somme magique M. On en conclut alors la paramétrisation complète d'un tétraèdre magique :



3.1.2 Implémentation dans Python

Maintenant que nous avons trouvé la paramétrisation générale de tout tétraèdre magique, nous mettons en place un programme nous permettant de déterminer s'il est possible de construire un tétraèdre magique à partir de 5 variables libres a, b, d, e et g, choisies aléatoirement.

Ci-dessous se trouve l'implémentation dans Python du programme dont il est ici question :

```
1  import random
2  import math
3
4  #on déclare la somme magique du tétraèdre
5  M = int(random.randint(1,600))
6  print("M=",M)
7
8  #on déclare les variables libres du tétraèdre comme des entiers
       positifs aléatoires compris entre 1 et 300
9  a = int(random.randint(1,300))
10 print("a=",a)
11
12 b = int(random.randint(1,300))
13 print("b=",b)
14
15 d = int(random.randint(1,300))
16 print("d=",d)
```

```
17
18 e = int(random.randint(1,300))
19 print("e=",e)
20
21 g = int(random.randint(1,300))
22 print("g=",g)
23
24 def magic_tetrahedron(M,a,b,d,e,g):
25     #on déclare les variables liées du tétraèdre comme étant
       des entiers positifs
26     c = int()
27     f = int()
28     h = int()
29     i = int()
30     j = int()
31     k = int()
32     l = int()
33
34     if a!=b and a!=d and a!=e and a!=g and b!=d and b!=e and b
       !=g and d!=e and d!=g and e!=g:
35         if M>(a+b): #on vérifie que les variables liées sont
           des entiers strictement positifs dont la somme est
           strictement inférieure à M, autrement on arrête le
           programme
36             c = M-(a+b)
37             print("c=",c)
38             if M>(d+e):
39                 f = M-(d+e)
40                 print("f=",f)
41                 if M>(b+g):
42                     j = M-(b+g)
43                     print("j=",j)
44                     if M>(g-a+d+e):
45                         i = M-(g-a+d+e)
46                         print("i=",i)
47                         #maintenant on vérifie que les dernië
           res entrées du tétraèdre seront positives non nulles et
           strictement inférieures à la valeur de M
48                         if M>(g-a+d) and (g-a+d)>0:
49                             l = g-a+d
50                             print("l=",l)
51                             if M>(-a+d+e) and (-a+d+e)>0:
52                                 h = -a+d+e
53                                 print("h=",h)
54                                 if M>(a+b-d) and (a+b-d)>0:
55                                     k = a+b-d
56                                     print("k=",k)
57
58                                     #une fois tous les entiers
           nécessaires à la construction d'un tétraèdre magique de
           somme M trouvés, nous imprimons celui-ci
59                                     row1 = [a]
60                                     row2 = [b,c]
61                                     row3 = [g,j,k,d]
```

```
62                                      row4 = [h,i,l,e,f]
63
64                                      print(row1)
65                                      print(row2)
66                                      print(row3)
67                                      print(row4)
68
69                                  else:
70                                      print("Cannot create a
    Magic Tetrahedron")
71                              else:
72                                  print("Cannot create a Magic
    Tetrahedron")
73                          else:
74                              print("Cannot create a Magic
    Tetrahedron")
75                      else:
76                          print("Cannot create a Magic
    Tetrahedron")
77                  else:
78                      print("Cannot create a Magic Tetrahedron")
79              else:
80                  print("Cannot create a Magic Tetrahedron")
81          else:
82              print("Cannot create a Magic Tetrahedron")
83      else:
84          print("Cannot create a Magic Tetrahedron")
```

Dans un premier temps, des lignes 5 à 22, nous déclarons la somme magique M du tétraèdre, ainsi que ses 5 variables libres : a, b, d, e et g. Nous avons choisi de définir M comme étant un nombre entier aléatoirement choisi entre 1 et 600, tandis que les autres variables aléatoires sont comprises entre 1 et 300. Ce choix nous permet effectivement de maximiser nos chances d'obtenir aléatoirement une somme magique M numériquement plus grande que les entiers constituant le tétraèdre, condition *sine qua non* pour construire correctement tout objet magique.

Dès lors, à la ligne 24, nous définissons la fonction `magic_tetrahedron`, qui prend comme arguments les entiers M, a, b, d, e, g. Au sein de cette fonction, après avoir déclaré les variables liées de notre système en tant qu'entiers naturels (lignes 26 à 32), nous implémentons successivement les conditions devant être respectées afin de former un tétraèdre magique. Ainsi, nous vérifions d'abord à la ligne 34 que les variables libres précédemment déclarées sont toutes distinctes. Dans le cas où cela ne serait pas vrai, le programme retourne un message d'erreur avant de prendre fin, à la ligne 83. En revanche, si cette condition est vérifiée, le programme suit son cours, et à la ligne 35, il se charge de vérifier que la valeur de M est strictement supérieure à la somme des entiers a et b. Si cela est le cas, la valeur de cette somme est attribuée à la variable c, autrement l'algorithme prend fin avec un message d'erreur : `Cannot create a Magic Tetrahedron`. Nous procédons ensuite de la même

manière avec les sommes $d+e$, $b+g$ et $g-a+d+e$ (lignes 38 à 46).

Ces étapes nous garantissent que les entiers du tétraèdre sont tous strictement positifs et répondent bien aux conditions données par la paramétrisation générale d'un tétraèdre magique.

Finalement, nous veillons à ce que les valeurs des sommes $g-a+d$, $-a+d+e$ et $a+b-d$ soient toujours strictement inférieures à la somme magique M, mais également strictement positives, et lorsque c'est le cas, nous attribuons leurs valeurs respectivement aux entiers l (ligne 49), h (ligne 52) et k (ligne 55). Une fois de plus, ces conditions sont posées afin de répondre à la paramétrisation que nous avons trouvée, ainsi que la définition même d'un tétraèdre magique.

Si toutes les conditions successivement posées sont remplies, le programme donne alors le tétraèdre magique final obtenu, et imprime les entiers le composant (lignes 59 à 67). Cependant, si une seule de ces requêtes n'aboutit pas, le programme s'arrête directement, ce qui explique l'implémentation de "else-statements" consécutifs des lignes 69 à 84.

3.1.3 Visualisation

Voici finalement un tétraèdre magique trouvé au moyen du programme Python présenté précédemment :

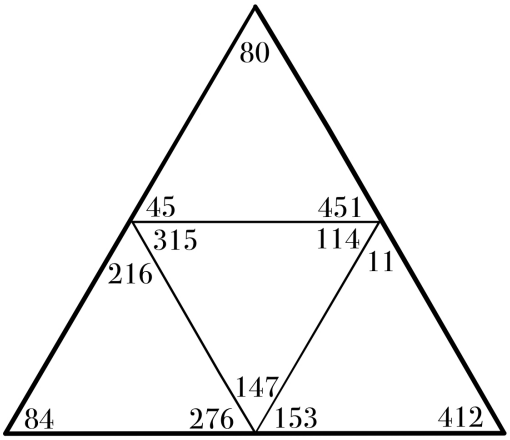


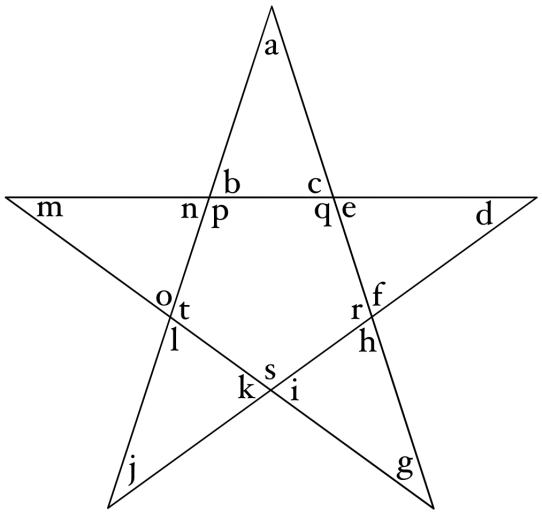
FIGURE 7 – Tétraèdre magique de somme $M = 576$.

3.2 Étoiles magiques

Les étoiles magiques sont des pyramides à base pentagonale dont la somme des entiers constituant chaque face et chaque sommet est égale.

3.2.1 Paramétrisation

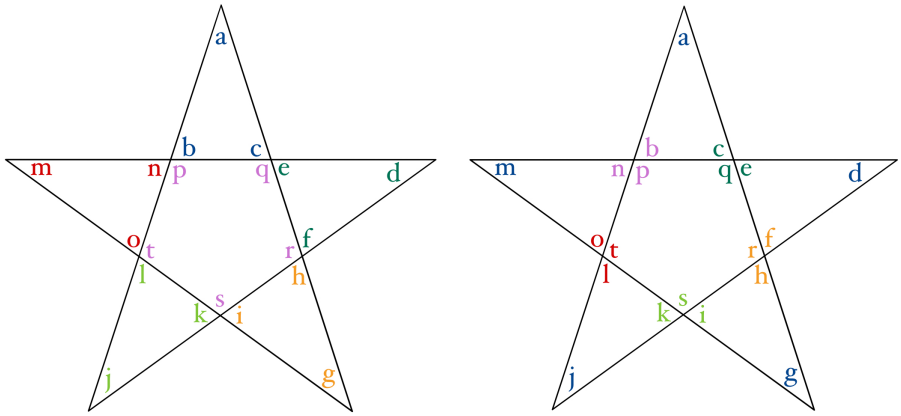
Soient $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t \in \mathbb{N}$ des entiers naturels distincts. On définit une étoile magique comme suit :



avec M sa constante magique, définie telle que :

$$\begin{cases} a+b+c & = & M, \\ d+e+f & = & M, \\ g+h+i & = & M, \\ j+k+l & = & M, \\ m+n+o & = & M, \\ b+n+p & = & M, \\ c+e+q & = & M, \\ f+h+r & = & M, \\ i+k+s & = & M, \\ l+o+t & = & M, \\ p+q+r+s+t & = & M, \\ a+d+g+j+m & = & M. \end{cases}$$

Ces égalités entre les différentes sommes des entiers d'une étoile magique peuvent se visualiser de la manière suivante :



Dans chacune de ces figures, on a que toutes les sommes des entiers d'une même couleur sont égales entre elles.

Afin de résoudre le système d'équation linéaires définissant une étoile magique, nous exprimons celui-ci au moyen de matrices :

$$\underbrace{\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}}_{=:P}
\underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \\ m \\ n \\ o \\ p \\ q \\ r \\ s \\ t \end{pmatrix}}_{=:x}
= \begin{pmatrix} M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \\ M \end{pmatrix}.$$

Le logiciel SageMath nous donne alors la base W suivante comme base

du noyau de la matrice P :

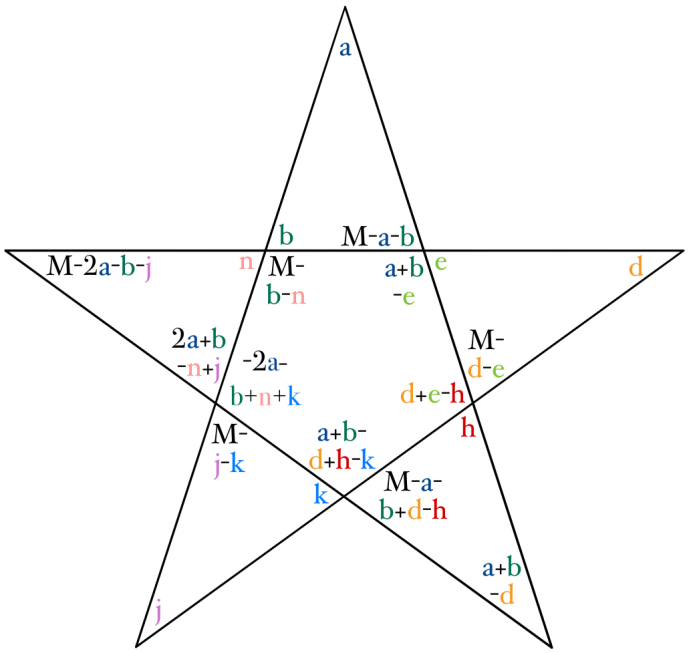
$$W := \langle \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ -2 \\ 0 \\ 2 \\ 0 \\ 1 \\ 0 \\ 1 \\ -2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rangle.$$

De plus, on obtient comme solution :

$$x = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T \times M.$$

En suivant le même raisonnement que pour les autres objets magiques précédemment considérés, on a que les variables libres de notre système sont les entiers a, b, d, e, h, j, k, n.

Nous en concluons alors que toute étoile magique peut être construite à partir des entiers positifs a, b, d, e, h, j, k, n ainsi que d'une somme magique M. Cela nous permet de dresser la paramétrisation finale d'une étoile magique :



3.2.2 Implémentation dans Python

La paramétrisation que nous venons de trouver nous permet à présent de mettre en place un programme Python construisant des étoiles magiques :

```
1  import random
2  import math
3
4  #on déclare les variables libres du système d'équations liné
       aires comme des entiers aléatoires entre 1 et 300
5  a = int(random.randint(1,300))
6  print("a=",a)
7
8  b = int(random.randint(1,300))
9  print("b=",b)
10
11 d = int(random.randint(1,300))
12 print("d=",d)
13
14 e = int(random.randint(1,300))
15 print("e=",e)
16
17 h = int(random.randint(1,300))
18 print("h=",h)
19
20 j = int(random.randint(1,300))
21 print("j=",j)
22
23 k = int(random.randint(1,300))
24 print("k=",k)
```

```
25
26 n = int(random.randint(1,300))
27 print("n=",n)
28
29 #on déclare la somme magique de la future étoile comme un
       entier aléatoire entre 1 et 800, permettant d'avoir plus de
        chances de trouver une somme M plus grande que les
       variables libres
30 M = int(random.randint(1,800))
31 print("M=",M)
32
33 #on définit la fonction magic_star prenant comme argument les
       variables libres ainsi que la somme magique M
34 def magic_star(a,b,d,e,h,j,k,n,M):
35     #on déclare les variables liées du système de notre étoile
       comme des entiers
36     c = int()
37     f = int()
38     g = int()
39     i = int()
40     l = int()
41     m = int()
42     o = int()
43     p = int()
44     q = int()
45     r = int()
46     s = int()
47     t = int()
48
49     #on vérifie d'abord que les variables libres précédemment d
       éfinies sont toutes disctinctes deux à deux
50     if a!=b and a!=d and a!=e and a!=h and a!=j and a!=k and a
       !=n and b!=d and b!=e and b!=h and b!=j and b!=k and b!=n
       and d!=e and d!=h and d!=j and d!=k and d!=n and e!=h and e
       !=j and e!=k and e!=n and h!=j and h!=k and h!=n and j!=k
       and j!=n and k!=n:
51         #on vérifie que la valeur de M est strictement supé
       rieure à la somme de a et b
52         if M>(a+b):
53             c = M-(a+b) #on assigne la valeur de M-(a+b) à la
       variable c
54             print("c=",c) #on imprime la valeur de c
55             #on vérifie ensuite que la valeur de M est
       strictement suéprieure à la somme de d et e
56             if M>(d+e):
57                 f = M-(d+e) #on assigne à f la valeur de M-(d+e
       )
58                 print("f=",f) #on imprime la variable f
59                 #on continue ainsi de suite avec les autres
       entiers de l'étoile magique
60                 if M>(j+k):
61                     l = M-(j+k)
62                     print("l=",l)
63                     if M>(f+h):
```

```
64                          r = M-(f+h)
65                          print("r=",r)
66                          if M>(b+n):
67                               p = M-(b+n)
68                               print("p=",p)
69                               if M>(c+e):
70                                    q = M-(c+e)
71                                    print("q=",q)
72                                    if M>(a+b-d+h):
73                                         i = M-(a+b-d+h)
74                                         print("i=",i)
75                                         if M>(h+i):
76                                              g = M-(h+i)
77                                              print("g=",g)
78                                              if M>(a+d+g+j):
79                                                   m = M-(a+d+g+j)
80                                                   print("m=",m)
81                                                   if M>(m+n):
82                                                        o = M-(m+n)
83                                                        print("o=",o)
84                                                        if M>(l+o):
85                                                             t = M-(l+o)
86                                                             print("t=",
      t)
87                                                             if M>(i+k):
88                                                                  s = M-(
      i+k)
89                                                                  print("
      s=",s) #si nous atteignons cette étape, l'étoile magique
      est complète, et on l'imprime
90
91                                                                  row1 =
      [a]
92                                                                  row2 =
      [b,c]
93                                                                  row3 =
      [m,n,p,q,e,d]
94                                                                  row4 =
      [o,t,r,f]
95                                                                  row5 =
      [l,s,h]
96                                                                  row6 =
      [k,i]
97                                                                  row7 =
      [j,g]
98
99                                                                  print(
      row1)
100                                                                 print(
      row2)
101                                                                 print(
      row3)
102                                                                 print(
      row4)
```

```
103                                                                 print(
    row5)
104                                                                 print(
    row6)
105                                                                 print(
    row7)
106
107                                                             else: #si M
     est inférieur à i+k, l'étoile magique ne peut être complét
    ée, et on retourne un message d'erreur
108                                                                 print("
    Cannot create a Magic Star")
109                                                         #de la même
    manière, si une des conditions énoncées auparavant ne peut
    pas être remplie, on arrête l'algorithme
110                                                         else:
111                                                             print("
    Cannot create a Magic Star")
112                                                 else:
113                                                     print("Cannot
    create a Magic Star")
114                                             else:
115                                                 print("Cannot
    create a Magic Star")
116                                         else:
117                                             print("Cannot create a
    Magic Star")
118                                     else:
119                                         print("Cannot create a
    Magic Star")
120                                 else:
121                                     print("Cannot create a Magic
    Star")
122                             else:
123                                 print("Cannot create a Magic Star")
124                         else:
125                             print("Cannot create a Magic Star")
126                 else:
127                     print("Cannot create a Magic Star")
128             else:
129                 print("Cannot create a Magic Star")
130         else:
131             print("Cannot create a Magic Star")
132     else:
133         print("Cannot create a Magic Star")
```

Ainsi, l'algorithme ici implémenté permet de déterminer si une étoile magique est constructible à partir de variables libres a, b, d, e, h, j, k, n et d'une somme magique M aléatoirement choisies. Si une telle construction est possible, l'algorithme nous donne alors l'étoile magique correspondante, sinon, l'algorithme retourne un message d'erreur.

L'implémentation de ce programme suit la même logique que ceux précédemment introduits. Le raisonnement consiste simplement à trouver suc-

cessivement les entiers constituant l'étoile, et ce, en suivant les conditions données au sein de la section 3.2.1. Si une de ces conditions n'est pas remplie, le programme prend fin.

3.2.3 Visualisation

Se trouve finalement ci-après un exemple d'étoile magique obtenu grâce au programme Python que nous avons implémenté.

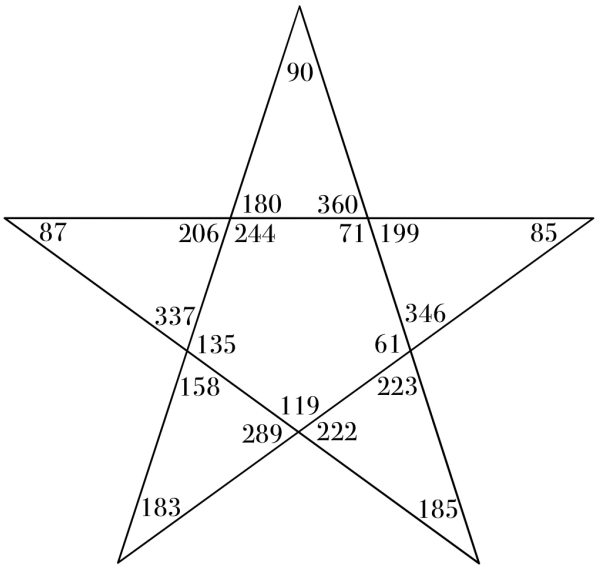


FIGURE 8 – Étoile magique de somme $M = 630$.

3.3 Tétraèdres magiques de Sierpiński

Au sein de cette section, nous étudions un objet magique que nous avons choisi d'appeler "Tétraèdre de Sierpiński". Ce nom s'inspire du triangle de Sierpiński, objet fractal, que nous avons dû adapter.

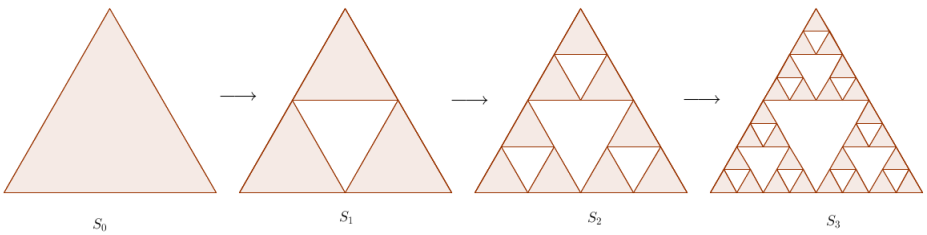


FIGURE 9 – Processus de construction du triangle de Sierpiński [3].

L'idée de notre adaptation nous est en effet venue après avoir traité le cas des tétraèdres magiques simples. En premier lieu, nous avions choisi de considérer un triangle de Sierpiński de dimension 3 (S_2 dans la figure 9). Cependant, en posant les conditions nécessaires pour rendre cet objet "magique", cela revenait à avoir à chaque fois les mêmes entiers dans les 3 principaux triangles.

Ainsi, nous avons décidé de "combler" les espaces laissés vides dans le triangle de Sierpiński original, ce qui résolvait notre problème, et revenait à considérer, avec $a_n, b_n, c_n, d_n, e_n, f_n, g_n, h_n, i_n, j_n, k_n, l_n \in \mathbb{N}$, $n \in \{1, 2, 3, 4\}$, l'objet suivant :

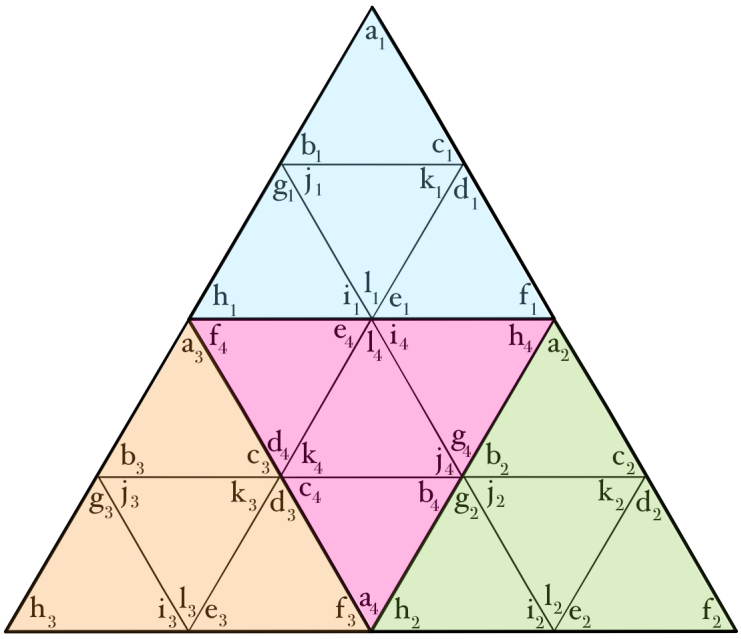


FIGURE 10 – Tétraèdre magique de Sierpiński.

On remarque alors que chaque face de ce tétraèdre est de nouveau un tétraèdre. Ce nouvel objet géométrique peut dès lors être rendu magique, et ce, en posant quelques conditions sur les entiers le constituant. Nous les précisons dans la section 3.3.1 qui suit.

3.3.1 Paramétrisation

Les conditions considérées afin de rendre un tétraèdre de Sierpiński "magique" sont très similaires à celles d'un tétraèdre magique simple.

Soient $a_n, b_n, c_n, d_n, e_n, f_n, g_n, h_n, i_n, j_n, k_n, l_n \in \mathbb{N}$, $n \in \{1, 2, 3, 4\}$, des entiers distincts, et soit M la constante magique de notre tétraèdre de Sierpiński. On a alors :

$$\begin{cases} a_n + b_n + c_n &= M, \\ d_n + e_n + f_n &= M, \\ g_n + h_n + i_n &= M, \\ j_n + k_n + l_n &= M, \\ a_n + f_n + h_n &= M, \\ b_n + j_n + g_n &= M, \\ c_n + k_n + d_n &= M, \\ i_n + l_n + e_n &= M, \\ a_1 + f_2 + h_3 &= M, \\ a_2 + f_1 + h_4 &= M, \\ a_3 + f_4 + h_1 &= M, \\ a_4 + f_3 + h_2 &= M. \end{cases} \qquad \forall n \in \{1, 2, 3, 4\}$$

Afin de trouver les solutions de ce système d'équations linéaires, nous l'exprimons en termes de matrices. Cependant, de par leurs tailles, celles-ci ne peuvent pas être représentées ici.

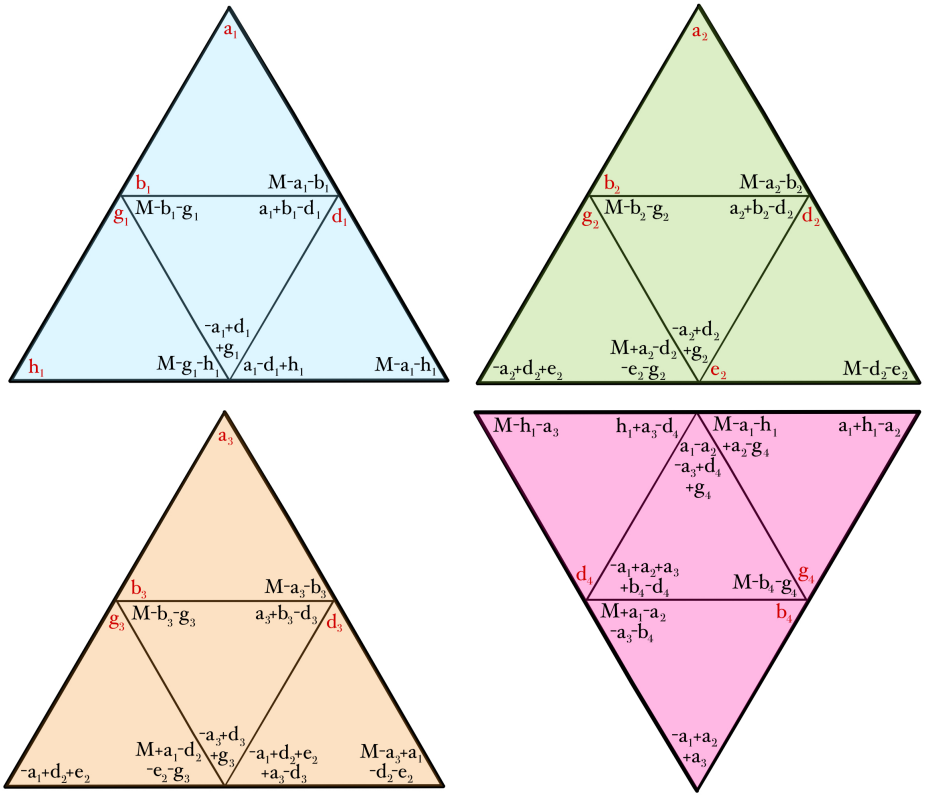
Le logiciel SageMath nous donne la matrice suivante en tant que noyau de la matrice associée à notre système d'équations :

$$\begin{pmatrix}
1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

En analysant les lignes et colonnes de cette matrice de la même manière que pour les autres objets magiques, nous pouvons conclure que les variables libres de notre système sont :

$$a_1, b_1, d_1, g_1, h_1, a_2, b_2, d_2, e_2, g_2, a_3, b_3, d_3, g_3, b_4, d_4, g_4.$$

Finalement, nous dressons la paramétrisation suivante :



Remarque 3.1. Les tétraèdres présentés individuellement ici peuvent être réunis afin de correspondre à la figure 3.3 d'un tétraèdre de Sierpiński.

3.3.2 Implémentation dans Python

Étant donnée la longueur du programme créé dans le but de construire des tétraèdres de Sierpiński, son implémentation dans ce rapport est impossible. Le lecteur est alors invité à consulter le répertoire GitHub suivant : `https://github.com/doubavitch/tetra_sierpinski.git`

3.3.3 Visualisation

Finalement, l'algorithme mis en place nous permet d'obtenir le tétraèdre magique de Sierpiński suivant :

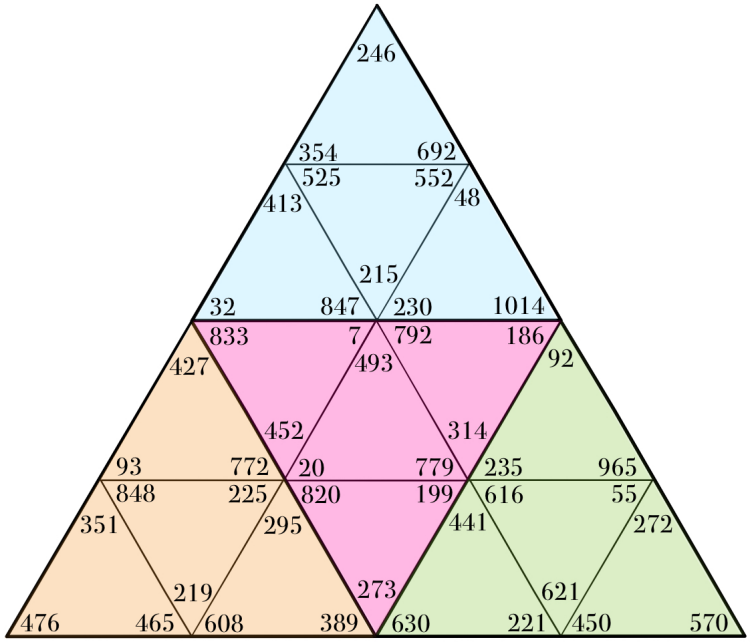


FIGURE 11 – Tétraèdre magique de Sierpiński de somme $M = 1292$.

4 Conclusion

Ainsi, nous avons réussi à donner une paramétrisation complète de chacun des objets magiques définis au cours de ce rapport, et avons pu donner un exemple, accompagné d'une visualisation, dans chaque cas. Par ailleurs, tous les objets magiques que nous avons choisi d'étudier ont pu être associés à un système d'équations linéaires soluble, rendant leur construction possible.

Cependant, les résultats présentés sont bien évidemment perfectibles. Par exemple, nous aurions encore pu prendre le soin de modifier les algorithmes donnés afin de nous assurer que les entiers composant la version complète de notre objet magique sont bien tous distincts.

Enfin, s'il y a une chose que nous avons retenue du Parker Square [4], c'est que dans ce domaine, il faut essayer, même si l'on n'est pas sûr de pouvoir trouver une solution. C'est ce que nous avons en particulier fait en choisissant de construire les tétraèdres de Sierpiński, et qui s'est révélé finalement fructueux.

Références

[1] Christian BOYER, `http://www.multimagie.com`.

[2] Jacques Sesiano, *Les carrés magiques dans les pays islamiques*, Presses polytechniques et universitaires romandes, 2004.

[3] Blogdemaths, https://blogdemaths.wordpress.com/2013/07/16/sierpinski-et-pascal-sont-dans-un-triangle/

[4] Matt Parker, *The Parker Square*, Numberphile, https://www.youtube.com/watch?v=aOT_bG-vWyg.