Les polyèdres réguliers convexes

Marc Lorenzi - 3 juillet 2018

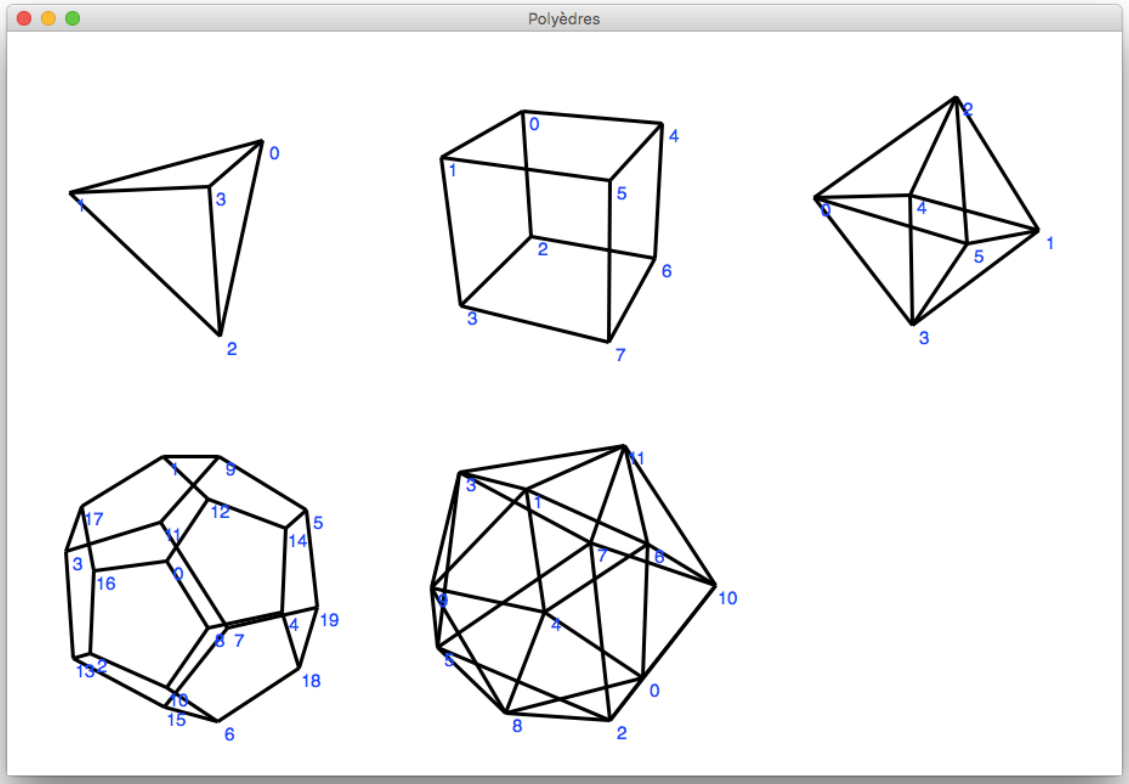
In [1]:
```
import matplotlib.pyplot as plt
from sympy import *
import math
%matplotlib inline
#init_printing()
```

In [2]:
```
plt.rcParams['figure.figsize'] = (6, 6)
```

Dans ce notebook, nous allons explorer quelques propriétés des **solides platoniciens** : le tétraèdre, le cube, l'octaèdre, l'icosaèdre et le dodécaèdre. Voici quelques questions auxquelles nous allons nous intéresser :

- Qu'est-ce qu'un polyèdre régulier ?
- Pourquoi les polyèdres réguliers sont-ils au nombre de 5 ?
- Comment dessiner un tel objet, le projeter joliment sur le plan ?
- Quelles sont les grandeurs caractéristiques de ces objets ?

Il n'y a rien de plus triste qu'un objet 3D que l'on ne peut pas faire tourner. Comme il n'est pas vraiment simple d'animer des graphiques `matplotlib` ou d'interagir sur eux, un petit programme (`polyedres.py`) est joint à ce notebook. Il affiche les 5 polyèdres réguliers et permet de les faire tourner à la souris.



Pour l'exécuter, ouvrez un terminal et entrez la commande `python3 polyedres.py` . Pour faire tourner un polyèdre, maintenez le bouton gauche de la souris enfocé et déplacez-la. Si les couleurs ne vous conviennent pas, ou que la taille des graphiques est trop grande ou trop petite, les premières lignes du fichier contiennent quelques paramètres que vous pouvez modifier. Faites des essais.

Maintenant, nous allons pouvoir travailler avec nos polyèdres sous les yeux. Leurs sommets sont numérotés, j'utiliserai ces mêmes numéros au cours de l'exposé.

1. Le cube

1.1 C'est quoi un polyèdre ?

Commençons par le plus facile, le cube. Cela nous permettra de mettre en place toute la machinerie nécessaire pour parler des autres polyèdres. Le cube a 8 sommets, 12 arêtes et 6 faces. De façon générale, un polyèdre $\mathcal{P}$ a un ensemble fini de **sommets** et A un ensemble fini d'**arêtes**, qui sont des segments reliant certains de ses sommets. Chaque arête est donc caractérisée par une paire $\{x, y\}$ de sommets distincts : ses extrémités. On impose que deux arêtes ne peuvent avoir qu'une de leurs extrémités en commun.

Deux sommets distincts sont dits **adjacents** lorsqu'ils sont les extrémités d'une même arête. De même, deux arêtes distinctes sont dites **adjacentes** lorsqu'elles ont un sommet en commun.

Soit $n \geq 2$. Un cycle de longueur n dans le polyèdre $\mathcal{P}$ est une suite de sommets $(s_0, s_1, \ldots, s_n)$ telle que

- les $s_i, 0 \leq i \leq n-1$ sont distincts deux à deux.
- $s_n = s_0$.
- Pour $0 \leq i \leq n-1$, $\{s_i, s_{i+1}\}$ est une arête de $\mathcal{P}$.

Par exemple, dans le cube de la fenêtre "Polyèdres", $(6, 4, 5, 7, 6)$ est un cycle. $(1, 3, 7, 5, 3)$ en est un autre. Et $(5, 4, 6, 2, 3, 1, 5)$ en est un troisième.

Les points des deux premiers cycles sont dans un même plan. Les arêtes qui les relient partage ce plan en deux parties, l'une bornée et l'autre pas. La partie bornée est appelée une **face**. Une face est donc caractérisée par une liste de sommets coplanaires (ou une liste d'arêtes, si on préfère).

Pour éviter des polyèdres pathologiques, nous imposons deux conditions supplémentaires :

- Deux faces ne peuvent s'intersecter que sur une arête commune. Lorsque c'est le cas, on dit que les faces sont **adjacentes**.
- Deux faces quelconques ne sont jamais dans un même plan.

1.2 Représenter un polyèdre en Python

Un polyèdre, nous l'avons dit, est caractérisé par ses sommets, des points de l'espace, ses arêtes, des paires de sommets, et ses faces, elles-mêmes définies par des listes de sommets. Pour plus de maniabilité dans la suite, nous allons définir une classe `Polyedre`. Cette classe contient juste un **constructeur**, la méthode `__init__`. Un objet de la classe `Polyedre` comporte un champ `sommets`, un champ `aretes` et un champ `faces`.

Dans un "vrai" programme Python, nous rajouterions dans la classe `Polyedre` des **méthodes**. Pour des raisons évidentes de présentation, nous ne procéderons pas ainsi dans ce notebook.

In [3]:
```
class Polyedre:

    def __init__(self, sommets, aretes, faces):
        self.sommets = sommets
        self.aretes = aretes
        self.faces = faces
```

Évidemment, il est facile de déterminer les sommets du cube. Trop facile ! Mais attendez donc que nous nous attaquions au dodécaèdre ... Nous allons prendre les 8 points $(\pm 1, \pm 1, \pm 1)$. Remarquez que deux sommets sont reliés par une arête si et seulement si il diffèrent par une unique corrdonnée. Voici donc le cube :

In [4]:
```
cube = Polyedre(
    [(-1,-1,-1),(-1,-1,1),(-1,1,-1),(-1,1,1),(1,-1,-1),(1,-1,1),(1,1,-
    [(0,1),(0,2),(0,4),(1,3),(1,5),(2,3),(2,6),(3,7),(4,5),(4,6),(5,7)
    [[0,1,5,4],[0,2,6,4],[0,1,3,2],[2,6,7,3],[4,5,7,6],[1,5,7,3]]
)
```

In [5]:
```
print(cube.sommets)
```

```
[(-1, -1, -1), (-1, -1, 1), (-1, 1, -1), (-1, 1, 1), (1, -1, -1), (1
, -1, 1), (1, 1, -1), (1, 1, 1)]
```

In [6]:
```
print(cube.aretes)
```

```
[(0, 1), (0, 2), (0, 4), (1, 3), (1, 5), (2, 3), (2, 6), (3, 7), (4,
5), (4, 6), (5, 7), (6, 7)]
```

In [7]:
```
print(cube.faces)
```

```
[[0, 1, 5, 4], [0, 2, 6, 4], [0, 1, 3, 2], [2, 6, 7, 3], [4, 5, 7, 6
], [1, 5, 7, 3]]
```

Maintenant, il faudrait le dessiner, ce cube.

2. Projection centrale

2.1 C'est quoi ?

Regardez la fenêtre "Polyèdres" : les faces éloignées des polyèdres sont plus petites que les faces proches. Le programmeur n'a donc pas fait une bête projection orthogonale pour afficher les objets. Nous utiliser la **projection centrale**, qui permet justement de donner un effet de perspective.

Soit $\Omega \in \mathbb{R}^3$. Soit $\mathcal{P}$ un plan affine. Soit $M \in \mathbb{R}^3$, $M \neq \Omega$. Sauf pas de chance (parallélisme !), la droite $\mathcal{D}$ qui passe par Ω et M coupe le plan $\mathcal{P}$ en un point M'. Le point M' est le projeté de M sur $\mathcal{P}$ par la projection centrale de centre Ω. Il nous reste à calculer M'.

Dans la suite, nous allons nous placer dans un cas particulier : $\Omega = (0, 0, z_0)$ où $z_0 \in \mathbb{R}$, et $\mathcal{P}$ est le plan d'équation $z = z_1$, où $z_1 \neq z_0$.

Pourquoi choisir un tel plan ? Pour coller avec l'écran de notre ordinateur. Les axes Ox et Oy sont les axes de coordonnées de l'écran. L'axe Oz part de l'écran et pointe vers l'observateur Ω (le type assis devant son écran). L'objet à projeter est quelque part dans l'espace, disons derrière l'écran.

2.2 Calcul du projeté

Soit $M = (x, y, z) \in \mathbb{R}^3$. Soit $M' = (x', y', z_1)$ son projeté central sur le plan $\mathcal{P}$. Il existe $\mu \in \mathbb{R}$ tel que $\overrightarrow{\Omega M'} = \mu \overrightarrow{\Omega M}$. En termes de coordonnées :

$$x' = \mu x, y' = \mu y, z_1 - z_0 = \mu(z - z_0)$$

Ainsi :

$$x' = \frac{z_1 - z_0}{z - z_0} x, y' = \frac{z_1 - z_0}{z - z_0} y$$

In [8]:
```
def projection(p, z0, z1):
    x, y, z = p
    mu = (z1 - z0) / (z - z0)
    return (mu * x, mu * y)
```

In [9]:
```
projection((1, 2, 3), 10, 5)
```

Out[9]:
```
(0.7142857142857143, 1.4285714285714286)
```

Exercice : Refaire le calcul pour un point Ω et un plan $\mathcal{P}$ quelconques.

2.3 Projeter et dessiner un polyèdre

Projeter un polyèdre, c'est projeter ses sommets.

In [10]:
```
def projection_poly(P, z0, z1):
    s = []
    for p in P.sommets:
        p = projection(p, z0, z1)
        s.append(p)
    return s
```

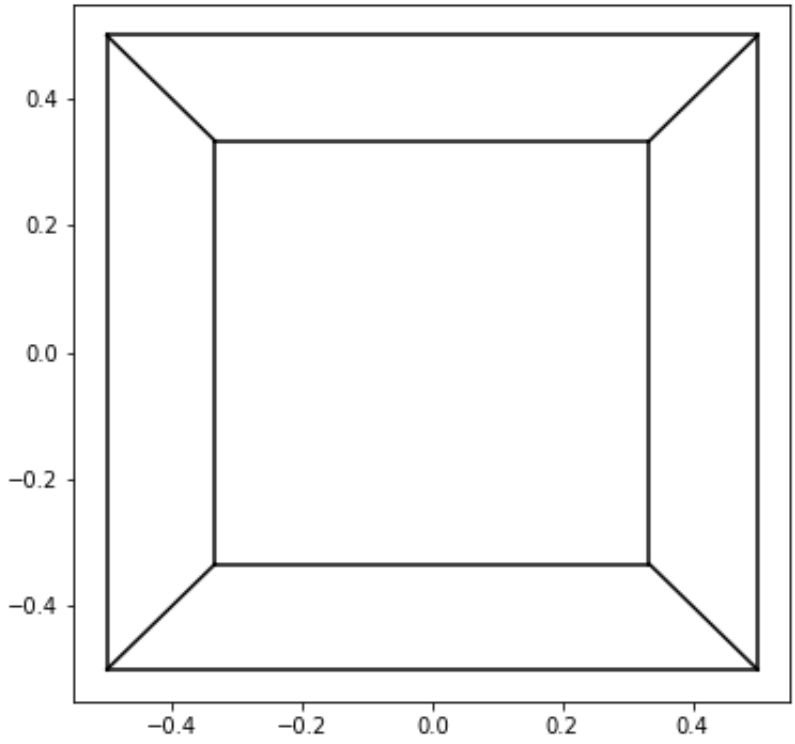
In [11]:
```
projection_poly(cube, 5,3)
```

Out[11]:
```
[(-0.3333333333333333, -0.3333333333333333),
 (-0.5, -0.5),
 (-0.3333333333333333, 0.3333333333333333),
 (-0.5, 0.5),
 (0.3333333333333333, -0.3333333333333333),
 (0.5, -0.5),
 (0.3333333333333333, 0.3333333333333333),
 (0.5, 0.5)]
```

Dessiner un polyèdre, c'est dessiner les projections de ses arêtes. Il s'avère que la projection centrale conserve les droites (sauf celles qui passent par Ω, qui se transforment en points, ou en l'ensemble vide si elles sont parallèles au plan). Pour tracer les arêtes il nous suffira donc de tracer les segments qui relient les projections des sommets du polyèdre.

In [12]:
```
def dessiner(P, z0, z1):
    s = projection_poly(P, z0, z1)
    for (i, j) in P.aretes:
        a, b = s[i]
        c, d = s[j]
        plt.plot((a, c), (b, d), 'k')
```

In [13]:
```
dessiner(cube, 5, 3)
```



Nous voilà un peu satisfaits mais pas complètement. La projection centrale fonctionne bien, mais notre cube est trop "droit", trop "parallèle" aux axes de coordonnées. Il faudrait le faire un peu tourner pour mieux le voir.

2.4 Rotations

Soit f la rotation de $\mathbb{R}^3$ d'axe la droite vectorielle D orienté par un vecteur unitaire ω, et d'angle θ. La formule d'Euler-Rodrigues nous donne, pour tout vecteur u de l'espace :

$$f(u) = u\cos\theta + (1-\cos\theta) < \omega, u > \omega + \sin\theta\, \omega \wedge u$$

où les crochets représentent le produit scalaire et $\wedge$ est le produit vectoriel.

Posant $\omega = (\alpha, \beta, \gamma)$ et $u = (x, y, z)$ il est alors facile de calculer les coordonnées de $f(u)$. C'est ce que fait la fonction `rotation` . Dans cette fonction, le vecteur ω n'est pas nécessairement unitaire : la fonction se charge de le normaliser.

In [14]:
```
def rotation(u, omega, theta):
    alpha, beta, gamma = omega
    r = math.sqrt(alpha ** 2 + beta **2 + gamma **2)
    alpha, beta, gamma = (alpha / r, beta / r, gamma / r)
    x, y, z = u
    c = math.cos(theta)
    s = math.sin(theta)
    ps = alpha * x + beta * y + gamma * z
    X = c * x + (1 - c) * ps * alpha + s * (beta * z - gamma * y)
    Y = c * y + (1 - c) * ps * beta + s * (gamma * x - alpha * z)
    Z = c * z + (1 - c) * ps * gamma + s * (alpha * y - beta * x)
    return (X, Y, Z)
```

L'astuce du jour : Que se passe-t-il dans la fenêtre "Polyèdres" lorsqu'on déplace la souris et qu'un polyèdre se met à tourner ?

- Le programme détermine le déplacement $\vec{v}$ de la souris dans le plan xOy.
- Il en déduit un axe de rotation qui est la droite du plan xOy dirigée par le vecteur $\vec{\omega}$ directement orthogonal à $\vec{v}$.
- De quel angle tourner ? D'un angle proportionnel à distance dont la souris s'est déplacée. La constante de proportionnalité est choisie pour que les choses aient l'air "naturelles".

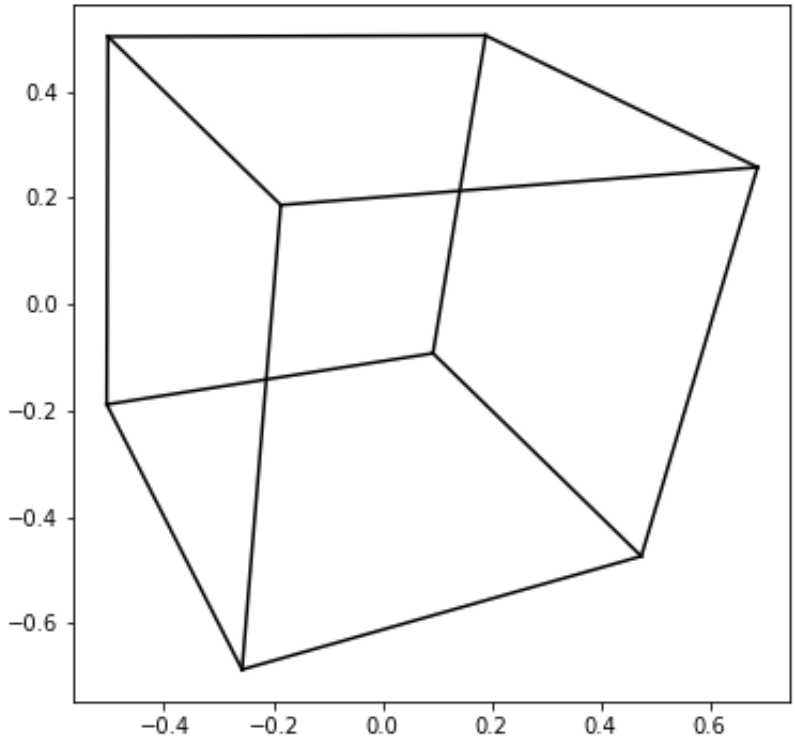
Exercice : Dans la fenêtre "Polyèdres", positionnez le cube de telle sorte que la face $(3, 7, 6, 2)$ soit en haut, l'arête $(6, 7)$ vers l'avant. Puis faites tourner le cube de 90° par rapport à l'axe Oz, de telle sorte que la face $(3, 7, 6, 2)$ soit gauche, l'arête $(6, 7)$ toujours vers l'avant.

Moralité : on n'a pas vraiment besoin des rotations d'axe Oz.

Faisons maintenant tourner tout un polyèdre. La fonction `rotation_poly` prend un polyèdre en paramètre, ainsi qu'un vecteur unitaire et un angle. Elle renvoie l'image du polyèdre par la rotation correspondante.

In [15]:
```
def rotation_poly(P, omega, theta):
    s = [rotation(p, omega, theta) for p in P.sommets]
    return Polyedre(s, P.aretes, P.faces)
```

In [16]:
```
dessiner(rotation_poly(cube, (1, 1, 0), 0.7), 5, 3)
```



Nous voilà satisfaits. On aimerait animer tout cela mais comme je l'ai expliqué au début, `matplotlib` n'est pas l'outil rêvé pour le faire.

NB : Je ne dis pas qu'on ne peut pas faire d'animations avec `matplotlib` . Je dis que je préfère utiliser autre chose (le module `tkinter` , en l'occurence).

3. Cinq polyèdres réguliers ?

Pourquoi 5, et pas 4 ou 12 ? Nous allons montrer qu'il existe au plus 5 polyèdres réguliers. Mais avant tout, qu'appelle-t-on polyèdre **régulier** ? Un polyèdre est régulier lorsque :

- Toutes ses faces sont des polygones réguliers "identiques" : même nombre d'arêtes, même taille.
- Tous ses sommets appartiennent au même nombre de faces (ou d'arêtes, cela revient au même).

Nous allons aussi supposer nos polyèdres **convexes** : si l'on supprime l'hypothèse de convexité, il existe trois polyèdres supplémentaires : les polyèdres de **Kepler-Poinsot**.

Proposition : Il existe au plus 5 polyèdres réguliers convexes.

Démonstration : Je passerai sous silence quelques points techniques ...

Soit $\mathcal{P}$ un polyèdre convexe. Supposons que les faces de notre polyèdre ont p arêtes. Supposons que chaque sommet appartient à q faces (ou à q arêtes, cela revient au même). Pour le cube, par exemple, $p = 4$ et $q = 3$.

Que valent les angles entre deux arêtes adjacentes ? Prenez un papier et un crayon, dessinez un polygone régulier ayant p arêtes. Ce polygone a un centre O. Dessinez des rayons partant du centre et allant aux sommets de ce polygone : vous venez de dessiner p triangles isocèles. Les angles "au centre" de ces triangles isocèles valent $\frac{2\pi}{p}$. Il vous est alors facile d'en déduire que les deux angles égaux de chaque triangle valent $\frac{1}{2}(\pi - \frac{2\pi}{p})$, puis d'en déduire que l'angle entre deux arêtes du polygone est $\pi - \frac{2\pi}{p}$.

Prenons maintenant un sommmet s de notre polyèdre. Il y a q arêtes issues de s, et l'angle entre deux arêtes successives est $\pi - \frac{2\pi}{p}$. La somme de ces angles est $q(\pi - \frac{2\pi}{p})$. Là je ne vais pas entrer dans les détails, mais notre hypothèse de convexité du polyèdre et le fait que deux faces ne soient pas dans le même plan entraînent que cette somme est strictement inférieure à 2π. Faites un dessin pour essayer de vous en convaincre.

On a donc $q(\pi - \frac{2\pi}{p}) < 2\pi$, ou encore $q(1 - \frac{2}{p}) < 2$, c'est à dire

$$\frac{1}{p} + \frac{1}{q} > \frac{1}{2}$$

Autre façon d'écrire cela : $(p-2)(q-2) < 4$. Sachant que $p \geq 3$ et $q \geq 3$, il n'y a qu'un nombre très restreint de possibilités.

- $(p-2)(q-2) = 1$: on a alors $p = q = 3$.
- $(p-2)(q-2) = 2$: on a alors $p = 4, q = 3$ ou $p = 3, q = 4$.
- $(p-2)(q-2) = 3$: on a alors $p = 5, q = 3$ ou $p = 3, q = 5$.

Cela fait donc 5 possibilités : il y a au plus 5 polyèdres réguliers.

Et maintenant ? Il nous reste à prouver qu'il y a au moins 5 polyèdres réguliers, que tous les couples (p, q) trouvés ci-dessus correspondent à un vrai objet. Comment ? En les construisant un par un. Nous en avons déjà un, le cube, qui correspond au couple $(p, q) = (4, 3)$. Les quatre autres, vous les avez sous les yeux, mais nous allons montrer que nos sens ne nous trompent pas :-).

4. Le tétraèdre et l'octaèdre

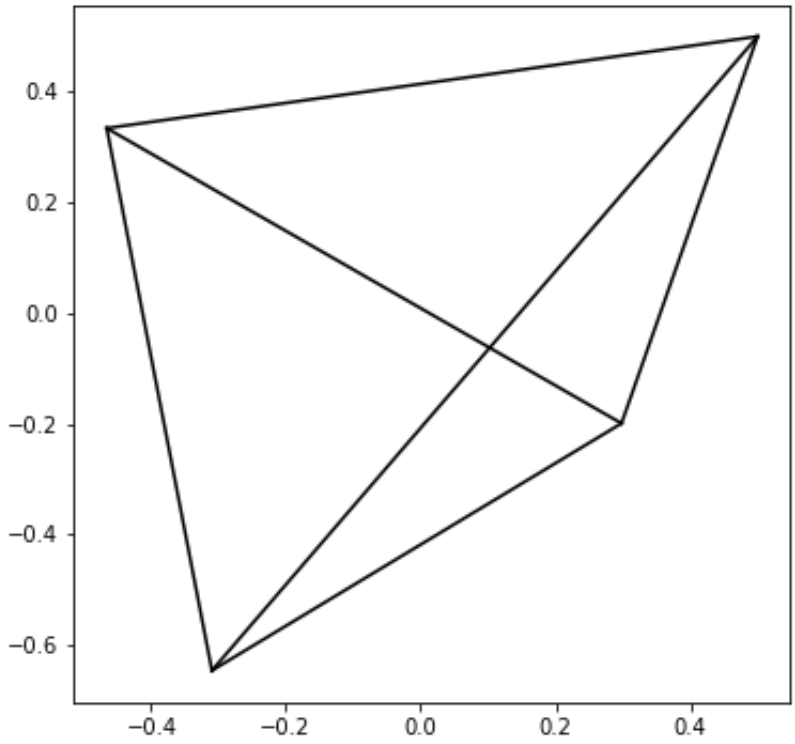
4.1 Le tétraèdre

Prenons les sommets 1,2,4,7 du cube. Relions-les par des arêtes. Ces arêtes forment des diagonales des faces du cube, elles ont donc toutes la même longueur. Nous obtenons là un polyèdre régulier à 4 sommets et 4 arêtes. Ses faces sont des triangles dont les côtés sont égaux, donc des triangles dont tous les angles sont égaux ! Et de chaque sommet partent 3 arêtes. On tient un polyèdre régulier, caractérisé par le couple $(p, q) = (3, 3)$: le **tétraèdre**.

Aparté : Oui, je sais, il faudrait prouver que les arêtes ne s'intersectent qu'en leurs extrémités, que le tétraèdre est convexe, que les faces ne sont pas coplanaires, etc. Libre à vous de le faire :-).

In [17]:
```
tetraedre = Polyedre(
    [(1,1,1),(1,-1,-1),(-1,1,-1),(-1,-1,1)],
    [(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)],
    [[0,1,3],[0,1,2],[0,2,3],[1,2,3]]
)
```

In [18]:
```
dessiner(rotation_poly(tetraedre, (1, 1, 1), 0.3), 5, 3)
```

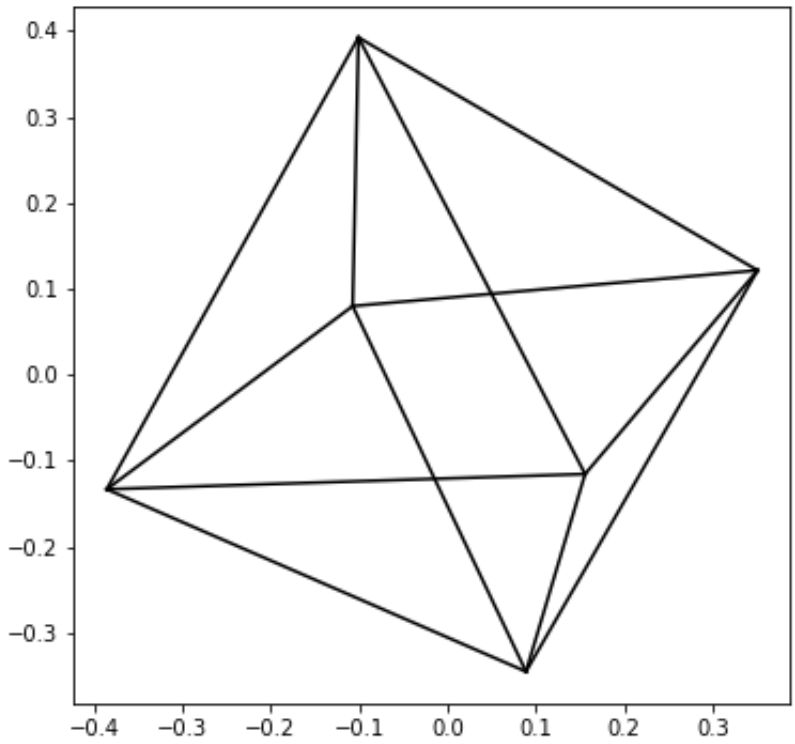


4.2 L'octaèdre

Encore plus facile à construire ! Prenons pour sommets les 6 points $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$ et $(0, 0, \pm 1)$. Et relions deux sommets lorsqu'ils n'ont pas les mêmes coordonnées nulles. Nous obtenons un polyèdre régulier, caractérisé par $(p, q) = (3, 4)$: l'**octaèdre**.

In [19]:
```
octaedre = Polyedre(
    [(-1,0,0),(1,0,0),(0,-1,0),(0,1,0),(0,0,-1),(0,0,1)],
    [(0,2),(0,3),(0,4),(0,5),(1,2),(1,3),(1,4),(1,5),(2,4),(4,3),(3,5)
    [[2,0,4],[2,1,4],[2,0,5],[2,1,5],[3,0,4],[3,0,5],[3,1,5],[3,1,4]]
)
```

In [20]:
```
dessiner(rotation_poly(octaedre, (1, 1, 1), 0.5), 5, 3)
```



Il est facile de montrer à la main que l'on a bien un polyèdre régulier : les 8 faces de l'octaèdre sont des triangles équilatéraux de même taille.

5. L'icosaèdre

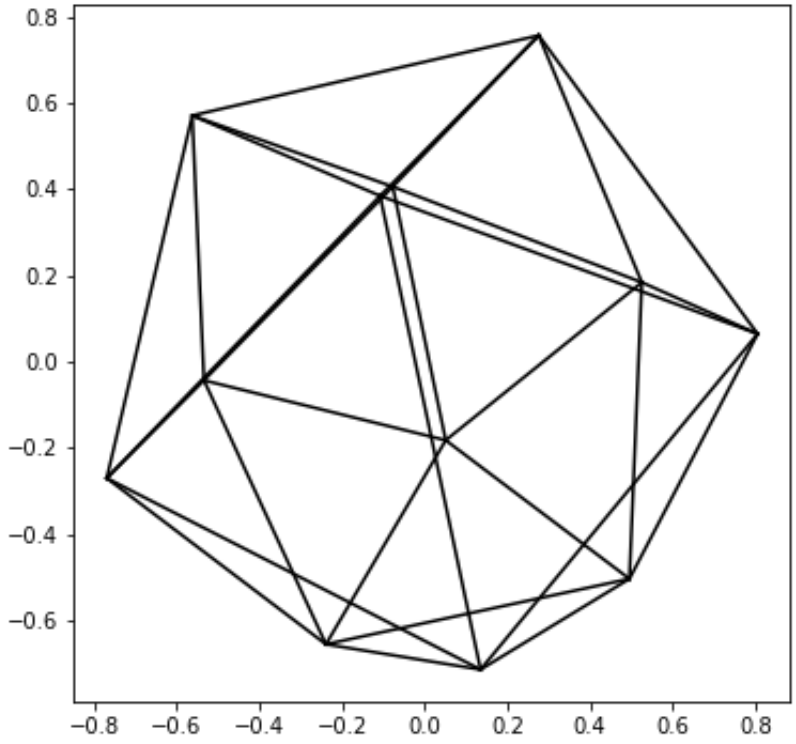
Voici maintenant une formule magique pour l'icosaèdre.

In [21]:
```
def icosaedre():
    phi = (1 + sqrt(5)) / 2
    sommets = [(0,-1,-phi),(0,-1,phi),(0,1,-phi),(0,1,phi),
               (-1,-phi,0),(-1,phi,0),(1,-phi,0),(1,phi,0),
               (-phi,0,-1),(-phi,0,1),(phi,0,-1),(phi,0,1)]
    aretes = [(0,2),(0,8),(0,4),(0,6),(0,10),
               (2,8),(8,4),(4,6),(6,10),(10,2),
               (3,1),(3,9),(3,5),(3,7),(3,11),
               (1,9),(9,5),(5,7),(7,11),(11,1),
               (1,4),(4,9),(9,8),(8,5),(5,2),(2,7),(7,10),(10,11),(11
    faces = [[1,11,6],[1,6,4],[1,4,9],[1,9,3],[1,3,11],[2,10,0],[2,0,8
             [7,10,11],[10,11,6],[10,0,6],[6,4,0],[0,8,4],[4,9,8],[8,5
    return Polyedre(sommets, aretes,faces)
```

In [22]:
```
icosaedre().sommets
```

Out[22]:
```
[(0, -1, -sqrt(5)/2 - 1/2),
 (0, -1, 1/2 + sqrt(5)/2),
 (0, 1, -sqrt(5)/2 - 1/2),
 (0, 1, 1/2 + sqrt(5)/2),
 (-1, -sqrt(5)/2 - 1/2, 0),
 (-1, 1/2 + sqrt(5)/2, 0),
 (1, -sqrt(5)/2 - 1/2, 0),
 (1, 1/2 + sqrt(5)/2, 0),
 (-sqrt(5)/2 - 1/2, 0, -1),
 (-sqrt(5)/2 - 1/2, 0, 1),
 (1/2 + sqrt(5)/2, 0, -1),
 (1/2 + sqrt(5)/2, 0, 1)]
```

In [23]:
```
dessiner(rotation_poly(icosaedre(), (1, 0, 1), 0.3), 5, 3)
```



Il nous faut maintenant vérifier que l'icosaèdre est un polygone régulier. Comme ses faces sont des triangles, il n'y a qu'à montrer que toutes ses arêtes ont la même longueur. Mais non, nous n'allons pas le faire à la main, car Python et `sympy` sont nos amis.

Voici une fonction qui calcule la longueur de l'arête d'extrémités les sommets i et j du polyèdre $\mathcal{P}$.

In [24]:
```
def longueur_arete(P, i, j):
    x = Matrix(P.sommets[i])
    y = Matrix(P.sommets[j])
    return simplify((y - x).norm())
```

In [25]:
```
longueur_arete(icosaedre(), 0, 2)
```

Out[25]: 2

Et voici une fonction qui calcule les longueurs de toutes les arêtes du polyèdre.

In [26]:
```
def longueurs_aretes(P):
    return [longueur_arete(P, i, j) for (i, j) in P.aretes]
```

In [27]:
```
print(longueurs_aretes(icosaedre()))
```

```
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
, 2, 2, 2, 2, 2, 2, 2]
```

Eh oui, l'icosaèdre est bien un polyèdre régulier. Il est caractérisé par le couple $(p, q) = (3, 5)$: ses faces ont trois côtés, et chaque sommet est adjacent à 5 faces.

Histoire de nous préparer à la suite, vérifions que les angles entre deux arêtes sont tous les mêmes. Je sais, ici c'est inutile puisque un triangle qui a trois côtés égaux a obligatoirement ses trois angles égaux. Mais préparons-nous à affronter le dodécaèdre ...

In [28]:
```
def angle(u, v):
    return acos(u.dot(v) / (u.norm()*v.norm()))
```

La fonction `angle_aretes` calcule l'écart angulaire entre les arêtes adjacentes (i, j) et (i, k) dans le polyèdre $\mathcal{P}$.

In [29]:
```
def angle_aretes(P, i, j, k):
    x = Matrix(P.sommets[i])
    y = Matrix(P.sommets[j])
    z = Matrix(P.sommets[k])
    return simplify(angle(y - x, z - x))
```

Testons ?

In [30]:
```
angle_aretes(icosaedre(), 0, 2, 10)
```

Out[30]:
```
pi/3
```

C'est ce à quoi on s'attendait. Et maintenant, écrivons une fonction qui calcule tous les angles entre tous les couples d'arêtes adjacentes.

In [31]:
```
def angles_aretes(P):
    s = []
    for f in P.faces:
        n = len(f)
        for i in range(n - 2):
                s.append(angle_aretes(P, f[i+1], f[i], f[i+2]))
        s.append(angle_aretes(P, f[n-1], f[n-2], f[0]))
    return s
```

In [32]:
```
print(angles_aretes(icosaedre()))
```

```
[pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, p
i/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/
3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3, pi/3,
pi/3, pi/3, pi/3, pi/3, pi/3, pi/3]
```

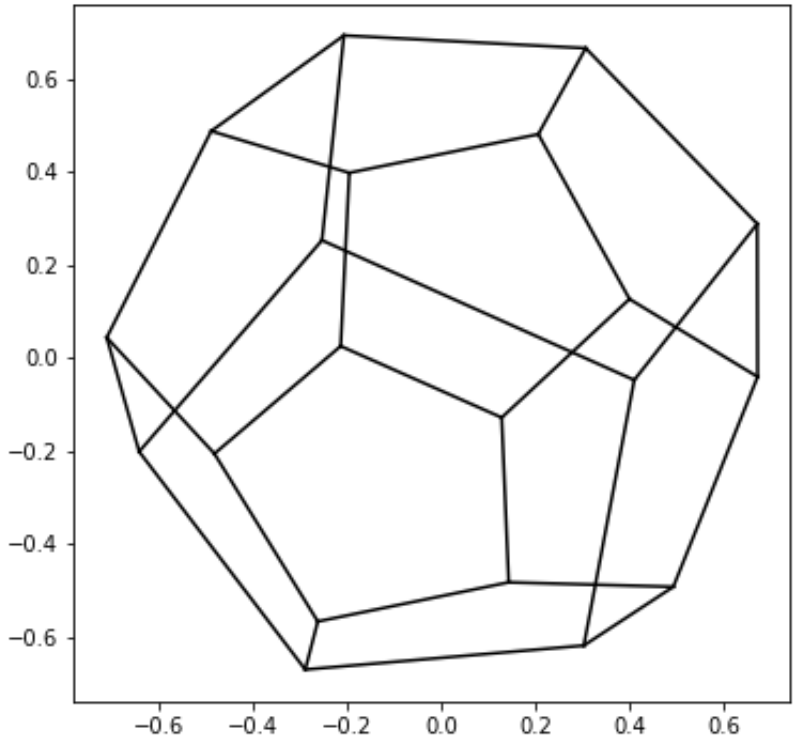
Maintenant on en est sûrs : l'icosaèdre est un polyèdre régulier.

6. Le dodécaèdre

Le meilleur pour la fin : le dodécaèdre.

In [33]:
```
def dodecaedre():
    phi = (1 + sqrt(5)) / 2
    sommets = [(-1,-1,-1),(-1,-1,1),(-1,1,-1),(-1,1,1),(1,-1,-1),(1,-1
              (0,-1/phi,-phi),(0,-1/phi,phi),(0,1/phi,-phi),(0,1/phi,pl
              (-1/phi,-phi,0),(-1/phi,phi,0),(1/phi,-phi,0),(1/phi,phi
              (-phi,0,-1/phi),(-phi,0,1/phi),(phi,0,-1/phi),(phi,0,1/pl
    aretes = [(1,9),(9,5),(5,14),(14,12),(12,1),
                (5,19),(19,7),(7,11),(11,9),(11,3),(3,17),(17,1),(17,1
                (3,13),(13,2),(2,16),(7,15),(15,13),(15,6),(6,10),(10,
                (8,4),(4,14),(18,4),(18,6),(18,19)]
    faces = [[5,19,18,4,14],[5,14,12,1,9],[5,9,11,7,19],[0,12,1,17,16]
             [7,19,18,6,15],[6,18,4,8,10],[0,8,4,14,12],[0,8,10,2,16],
    return Polyedre(sommets, aretes, faces)
```

In [34]:
```
dessiner(rotation_poly(dodecaedre(), (1, 1, 0), 0.6), 5, 3)
```



In [35]:
```
print(longueurs_aretes(dodecaedre()))
```

```
[-1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5
), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt
(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sq
rt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 +
sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1
+ sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -1 + sqrt(5), -
1 + sqrt(5)]
```

Imaginez un pentagone ayant 5 côtés de même longueur. Ses angles sont-ils tous égaux ? Eh non, un pentagone n'est pas "rigide" comme un triangle, on peut l'aplatir, le déformer. Il nous faut donc vérifier que tous les angles entre des arêtes du dodécaèdre sont égaux.

In [36]:
```
print(angles_aretes(dodecaedre()))
```

```
[acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1
/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4
+ 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5
)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sq
rt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos
(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4),
acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/
4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4
+ 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5
)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sq
rt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos
(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4),
acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/
4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4
+ 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5
)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sq
rt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4), acos(-sqrt(5)/4 + 1/4)]
```

Pas beaux les angles, mais égaux. En fait on en a parlé plus haut : ces angles valent $\pi - \frac{2\pi}{5} = \frac{3\pi}{5}$. Et de 5, le dodécaèdre est un polyèdre régulier, caractérisé par le couple $(p, q) = (5, 3)$.

In [37]:
```
print(math.acos(-math.sqrt(5)/4+1/4))
print(3*math.pi/5)
```

```
1.8849555921538759
1.8849555921538759
```

Nous venons donc de prouver avec `sympy` le

Théorème : Il y a au moins 5 polyèdres réguliers convexes.

7. Bilan

Nous avons obtenu tous les polyèdres réguliers convexes. Ils sont caractérisés par le couple (p, q) où

- p est le nombre d'arêtes par face.
- q est le nombre d'arêtes issues de chaque sommet.

Ces polyèdres sont :

1. Le tétraèdre : $(p, q) = (3, 3)$, 4 sommets, 6 arêtes, 4 faces.
2. Le cube : $(p, q) = (4, 3)$, 8 sommets, 12 arêtes, 6 faces.
3. L'octaèdre : $(p, q) = (3, 4)$, 6 sommets, 12 arêtes, 8 faces.
4. Le dodécaèdre : $(p, q) = (5, 3)$, 20 sommets, 30 arêtes, 12 faces.
5. L'icosaèdre : $(p, q) = (3, 5)$, 12 sommets, 30 arêtes, 20 faces.

Et ensuite ? Il y aurait sans doute à parler de :

- la formule d'Euler : $S - A + F = 2$, où S, A, F sont les nombres de sommets d'arêtes et de faces.
- la "dualité" entre certains couples de polyèdres : que se passe-til lorsqu'on échange p et q ? En quoi l'octaèdre et le cube sont ils liés ? Et le dodécaèdre et l'icosaèdre ?
- Les groupes d'isométries des polyèdres. Combien d'éléments ? Sous-groupes ? Etc.
- Grandeurs géométriques des polyèdres : volume, surface, etc.

Bref, de beaucoup de choses. Alors arrêtons-nous là pour aujourd'hui.

Remarque finale : J'ai, tout au long du notebook, parlé des polyèdres, en disant qu'un polyèdre **a** des sommets, des arêtes, etc., que ses faces vérifient ce ci ou cela, et qu'en plus on a telle ou telle condition. Mais **C'est quoi** un polyèdre ? Je conseille au lecteur curieux l'excellent livre d'Imre Lakatos : Proofs and Refutations. Je n'en dirai pas plus :-).

In [ ]: