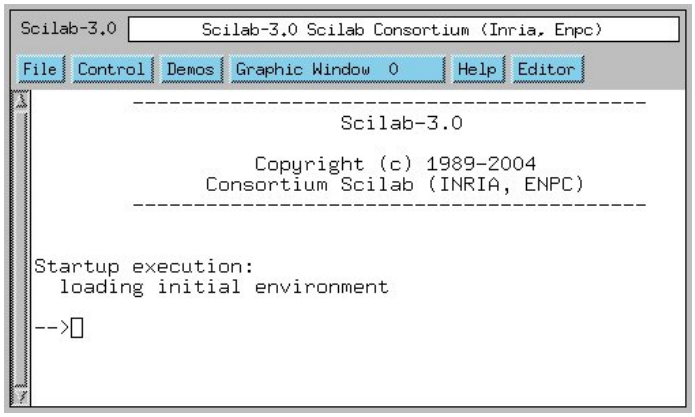
TP de Statistique (projet)

0. Lancer Scilab



1. Simulation de points aléatoire dans un carré

1.1. Taper l'instruction

```
rand()
```

puis valider. Exécuter plusieurs fois cette commande *(la touche <curseur haut> fait revenir les instructions données à Scilab, dans l'ordre de la dernière à la première)*. Qu'observe t-on ?

1.2. On va définir une suite a(n) indexée par les entiers de 1 à 1000 et prenant les valeurs successive donnée par rand() :
Taper et valider l'instruction

```
a=1:1000;for i=1:1000,a(i)=rand(),end
```

puis 'n' à la question intempestive [More (y or n) ?] posée.

Pour vérifier que la suite ainsi générée est uniformément répartie dans le segment [0,1] on peut afficher dans un graphique la suite des points de coordonnées (0,a(i)), i décrivant 1..30. Taper (et valider) l'instruction

```
xbasc();for i=1:30,plot2d(a(i),0,0,rect=[-.2,-.5,1.2,.5]),end
```

Qu'observe t-on ? Changer le nombre 30 en 200 puis en 1000. Est-ce concluant ?

Une autre méthode pour controler la répartition des valeurs des a(i) dans le segment [0,1] est de découper le segment en 20 intervalles et de calculer pour chacun de ces intervalles le nombre d'indices i tels que a(i) tombe dans l'intervalle. On affiche le résultat (les 20 nombres obtenus) sous une forme graphique qu'on appelle histogramme. La commande suivante fait tout ce travail :

```
xbasc();histplot(20,a)
```

1.3 Modifier l'instruction définissant a pour obtenir une suite de nombre aléatoire uniformément répartie dans le segment [-1,1] puis définir une suite b par la même instruction. On obtient alors une suite de couples (a(i),b(i)) qu'on va représenter graphiquement avec l'instruction

```
xbasc();for i=1:1000,plot2d(a(i),b(i),0,rect=[-1.2,-1.2,1.2,1.2]),end
```

La suite de points semble t-elle uniformément répartie dans le carré [-1,1]x[-1,1] ? Comparer avec ce qu'on obtient si on redéfinit la suite a par l'instruction

```
for i=1:1000,a(i)=-1+2*rand()^2,end
```

Changer de nouveau a en la suite demandée au début de 1.3

2. Approximation de pi

2.1 On va compter le nombre d'indices i tel que le point de coordonnées (a(i),b(i)) définit en début de 1.3 soit dans le disque de centre 0 et de rayon 1.
Essayer d'abord les instructions

```
bool2s(1<=2)
```

et

```
bool2s(2<1)
```

Ecrire une instruction dépendant de deux paramètres x et y qui rende 1 si le point (x,y) est dans le disque de centre (0,0) et de rayon 1, qui rende 0 sinon, et tester la.
Ecrire une instruction qui définisse une suite c avec c(i)=1 si le point d'indice i est dans le disque, c(i)=0 sinon.
On obtient maintenant le nombre de points dans le disque avec l'instruction

```
sum(c)
```

2.2 Si la répartition des points (a(i),b(i)), i décrivant 1,..,1000 est uniforme sur le carré, le nombre de points dans le disque de centre (0,0) et de rayon 1 doit être proche du quotient de l'aire du disque par celle du carré.
Transformer l'instruction en fin de 2.1 pour obtenir une approximation de pi.

2.3 Recommencer les instructions définissant a,b,c jusqu'à obtenir de nouvelles approximations de pi. Quelles variations observe t-on ?

3. Création d'une série de 400 valeurs approchées de pi.

On va créer une suite d(i), i décrivant les entiers de 1 à 400, contenant des valeurs approchées de pi obtenues par les instructions de 2.3
Ecrire une instruction sous forme de boucle (une instruction commençant par 'for i=1:400') répétant 2.3 et affectant à d(i) le résultat.
Calculer la moyenne des valeurs prises par les d(i) avec l'instruction

```
mean(d)
```

La valeur obtenue est elle plus proche de pi que les valeurs obtenues en 2.2 ?

4. Comparaison de la série avec une distribution normale

4.1 On va visualiser la suite d à l'aide d'un histogramme comme on la fait pour la suite a en 1.2. Cette fois on découpe le segment [2.97,3.31] en intervalles de longueur 0.02. On crée la suite x des extrémités de ces intervalles par l'instruction

```
x=2.97:0.02:3.31
```

Comme certaines valeurs de d peuvent sortir du segment [2.97,3.31] on extrait de d les seules valeurs tombant dans ce segment par l'instruction

```
de=d(2.97<=d & d<=3.31)
```

On fabrique maintenant l'histogramme par l'instruction

```
xbasc();histplot(x,de)
```

La commande histplot est telle que la suite des intervalles de [2.97,3.31] dans lesquels on compte le nombre d'occurence d'un élément de d est [x(0),x(1)],]x(1),x(2)],]x(2),x(3)],...
L'histogramme est normalisé : cela signifie que la graduation sur l'axe des y a été choisie par défaut de telle sorte que la somme des aires des rectangles vaut 1.

4.2 Pour comparer avec une loi normale ayant l'espérance et l'écart-type idéals de la série (c'est à dire correspondant à une suite d'approximation de pi de longueur infinie) on va dessiner dans la même fenêtre le graphe de la densité de la loi normale

$$f(x) = \frac{1}{s\sqrt{2\pi}}\, \mathrm{e}^{-\frac{(x-m)^2}{2s^2}}$$

où m est l'espérance (idéale) et s l'écart-type (idéal).
Que vaut m ? Définir m dans Scilab par une instruction.

On donne

$$s = \sqrt{\frac{m(4-m)}{1000}}$$

ce dont on instruit Scilab par l'instruction

```
s=sqrt(m*(4-m)/1000)
```

On définirait dans Scilab la fonction g(x)=cos(2pi*x) par la suite d'instructions

```
function y=g(x);
y=cos(2*%pi*x);
endfunction
```

Modifier cette suite d'instructions pour définir la fonction f ci-dessus(dépendant des paramètres m et s) .
(Les variables m et s devront être définies avant tout appel à la fonction f. Elles pourront éventuellement être redéfinies avant chaque appel à f.)
On trace le graphe de f dans la même fenêtre que l'histogramme par l'instruction

```
fplot2d(2.97:0.001:3.31,f)
```

Qu'observe t-on ?

5. Pourcentage de réussite

Les valeurs prises par la suite d(i), i=1..400 (définie en 3.) sont des valeurs approchées du nombre pi. On va caractériser la dispersion de cette série par son écart-type se et on va calculer le pourcentage d'indices i tel que pi soit dans l'intervalle]d(i)-se,d(i)+se]
5.1 Calculer l'écart-type effectif de la série d avec l'instruction

```
se=sqrt(variance(d))
```

5.2 Ecrire une instruction analogue à celle de 2.1 définissant une suite e(i), i=1..400, telle que pour tout i e(i)=1 si pi est dans l'intervalle]d(i)-se,d(i)+se] et 0 sinon.
Calculer la somme des valeurs prises par e puis le pourcentage d'indices cherché.
5.3 Refaire ce calcul en remplaçant se par 2*se.

*5.4 Pouvez vous écrire une fonction Scilab donnant les deux pourcentages réactualisés à chaque appel de la fonction *(voir l'aide en ligne de Scilab en appuyant sur <F1> dans la fenêtre de commande, chercher dans l'aide le mot clef 'function'.)*

6. Compte-rendu de séance

Lancez le programme Wordpad ou OpenOffice ou un éditeur html avec lequel vous allez écrire les instructions données à Scilab, recopier les résultats obtenus et écrire vos observation. Imprimer votre document quand vous aurez fini.
On peut sélectionner les instructions tapées dans la fenêtre de commande de Scilab ou les résultats fournis par Scilab et les copier ('Ctrl C' ou menu 'édition', 'copier') pour les coller (Ctrl V) dans l'éditeur que vous avez choisi pour votre compte-rendu.
Lorsqu'une instruction dans Scilab ouvre une fenêtre graphique, on peut sauvegarder le graphique dans le presse-papier (dans la fenêtre "Scilab Graphic", sélectionner le menu 'Fichier' 'Copier dans le presse papier' 'EnhMetafile') et coller le résultat dans l'éditeur. On peut aussi sauvegarder le graphique dans un fichier au format BMP (menu 'Fichier' 'Exporter') et le réutiliser plus tard pour le compte-rendu (menu 'Insertion' ou un simple "glisser-déposer" dans l'éditeur si c'est possible).

Au boulot !

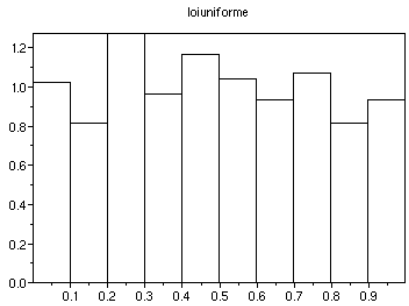
Adapté de la feuille de TP 2 de Statistiques en Licence MI/MP 2ème année, 2004-2005 "Méthode de Monte Carlo".
Page fabriquée avec l'éditeur Nvu, une commande de saisie de fenêtre au format jpg (touche 'Prt Scrn' sous Windows xp, la commande 'import' de ImageMagick sous Linux), Latex, la commande 'convert' de ImageMagick

F-X Dehon, 20 décembre 2005

Corrigé du TP2

1. Simulation de nombres aléatoires.

```
x=rand(1,1000);histplot(10,x);xtitle("loi uniforme")
```

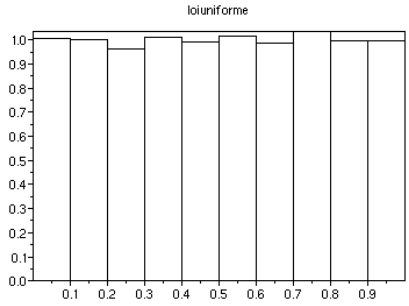


La hauteur du premier baton est donnée par le nombre d'indices i tels que x(i) est dans l'intervalle [0,0.1] divisé par le nombre total d'indices et par la largeur de l'intervalle de sorte que la somme des aires des batons vaut 1 comme l'aire délimitée par le graphe d'une densité de variable aléatoire. Idem pour les autres batons.

On peut contrôler cette affirmation avec la commande `sum(bool2s(x<=.1))/(1000*.1)` qui rend ici 1.02. De même `sum(bool2s(.2<x & x<=.3))/(1000*.1)` rend 1.26 ce qui correspond à la hauteur du 3ème batons.

La valeur attendue est 100/(1000*.1)=1 : on s'attend à ce que le nombre d'indices i tels que x(i) est dans l'intervalle [0,0.1] soit proportionnel à la largeur de l'intervalle [0,0.1] si les x(i) ont été uniformément distribués dans l'intervalle [0,1].

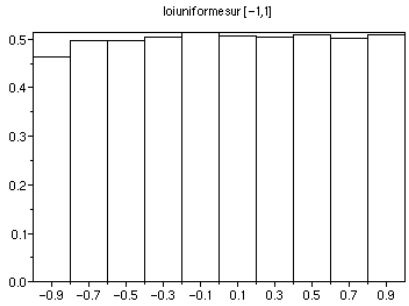
*Ce n'est pas exactement ce qu'on observe et c'est normal : le nombre d'indices i tels que x(i) est dans l'intervalle [0,0.1] suit une loi bionomiale B(1000,1/10) dont l'écart type est sqrt(1000*1/10*(1-1/10))=9.48.. . Ce nombre divisé par 100 suit donc une loi proche de la loi normale d'espérance 1 et d'écart type 0.0948.. . Cet écart-type donne un ordre de grandeur des fluctuations observées sur la hauteur des batons.
Si au lieu de simuler 1000 nbres aléatoires on en simule 10000, l'écart type associé à la hauteur d'un des 10 batons devient 0.03. Voici l'histogramme obtenu :



*

1.3 `2*rand()-1` simule une variable aléatoire de loi uniforme sur l'intervalle [-1,1]. On peut le vérifier en définissant un vecteur x comme précédemment et en traçant l'histogramme

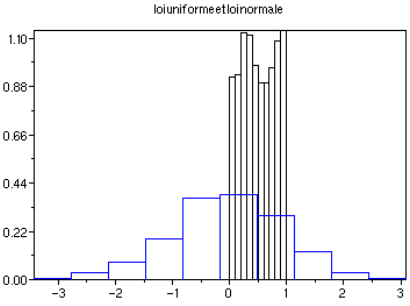
```
x=2*rand(1,10000)-1;clf();histplot(10,x);xtitle("loi uniforme sur
[-1,1]");xs2gif(0,"hist_unif3.gif")
```



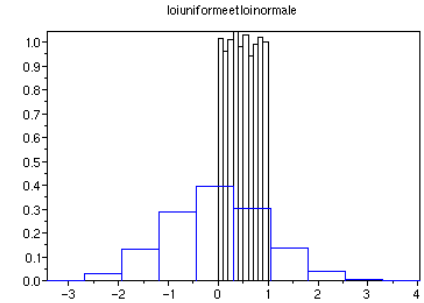
Pour a<=b, a+(b-a)*rand() simule une va de loi uniforme sur [a,b].

1.4 `rand(1,1000,'normal')` définit une matrice 1,1000 (une ligne et 1000 colonnes) dont chaque élément est obtenu par simulation d'une va de loi normale N(0,1) (espérance 0 et variance 1). Voici les histogrammes superposés de x défini par une loi uniforme et de y défini par une loi normale.

```
clf();x=rand(1,1000);histplot(10,x)
y=rand(1,1000,'normal');histplot(10,y,style=2)
xtitle("loi uniforme et loi normale")
```

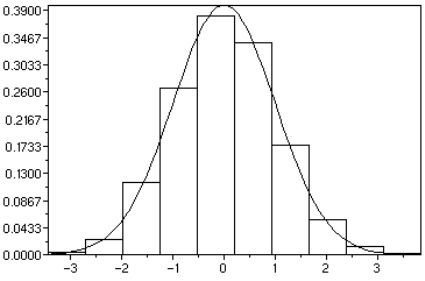


On observe que l'histogramme de x est concentré sur l'intervalle [0,1] alors que celui de y se réparti sur [-3.5,3] approximativement. Celui de y est en forme de cloche et est à peu près symétrique par rapport à 0. Comme précédemment on peut avoir une meilleure image des lois uniforme et normale en changeant 1000 en 10000 et on obtient



*En fait une va de loi normale N(0,1) est répartie sur R entier mais la probabilité qu'elle prenne des valeurs hors de l'intervalle [-4,4] est très faible ce qui fait qu'on ne l'observe pas sur l'histogramme. On peut comparer l'histogramme de y avec la densité de la loi normale

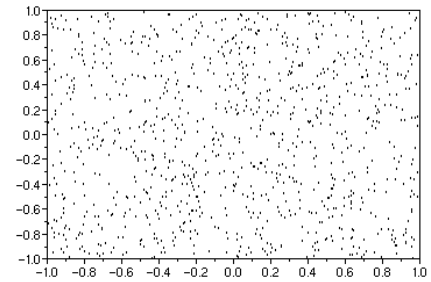
```
clf();y=rand(1,10000,'normal');histplot(10,y,style=2)
function y=f(x);y=1/sqrt(2*3.14159)*exp(-x^2/2);endfunction
fplot2d(min(y):.1:max(y),f)
xs2gif(0,"densite-norm.gif")
```



*

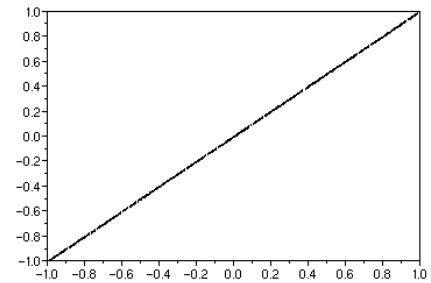
2. Simulation de points aléatoires dans un carré.

```
x=2*rand(1,1000)-1;y=2*rand(1,1000)-1;clf();plot2d(x,y,0)
```



Ca a l'air uniformément réparti.

```
clf();plot2d(x,x,0)
```



Répartition uniforme sur le segment [(0,0),(1,1)] de R^2.

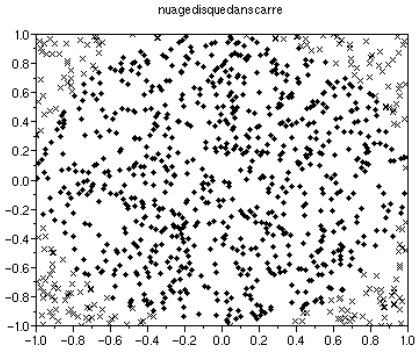
Si on change y en -y, cela ne changera pas l'allure du nuage (cela changera le nuage en son symétrique par rapport à la droite y=0) car la loi de y est symétrique par rapport à 0.

3. Distinguer les points du disque.

`z=bool2s(x^2+y^2<1)` défini un vecteur de même longueur que x et y dont la i-ème composante vaut 1 si x(i)^2+y(i)^2<1 (c'est à dire si le couple (x(i),y(i)) est un point du disque D(0,1)), 0 sinon.

```
clf();for i=1:1000;plot2d(x(i),y(i),-2*z(i)-2);end
```

Le dessin met beaucoup plus de temps à ce former que précédemment. C'est une particularité des boucles for dans scilab.



*On peut souvant ruser pour éviter une boucle for. Essayer ceci :
```
i=find(x^2+y^2<1);j=find(x^2+y^2>=1)
clf();plot2d(x(i),y(i),-4);plot2d(x(j),y(j),-2)
```
*

On s'attend à ce que la proportion de points dans le disque soit égale au rapport de l'aire du disque par l'aire du carré, c'est à dire pi/4

4. Estimation du nombre pi.

`sum(z)` rend 784, le nombre de 1 dans le vecteur z. On en déduit l'approximation sum(z)*4/1000 = 3.136 de pi. On peut simuler plusieurs fois x et y et ainsi obtenir de nouvelles approximations :

```
for i=1:3;pi(i)=sum(bool2s((2*rand(1,1000)-1)^2+
(2*rand(1,1000)-1)^2<1))*4/1000;end
pi'
```

ce qui donne

! 3.208 2.996 3.144 !

5. Propriétés de l'estimateur de pi.

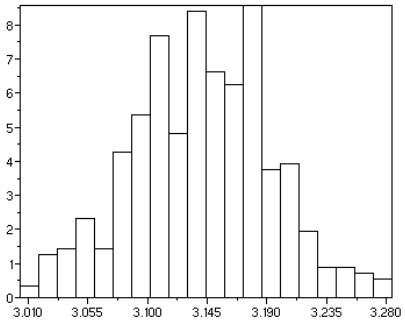
On construit un vecteur pi de longueur 400 formé de 400 simulations sucessives de pi par l'instruction

```
pi=[];for i=1:400;pi(i)=sum(bool2s((2*rand(1,1000)-1)^2+
(2*rand(1,1000)-1)^2<1))*4/1000;end
```

moyenne, écart type, min et max :

`[mean(pi),st_deviation(pi),min(pi),max(pi)]` rend ! 3.14357 0.0519110 2.972 3.28 !

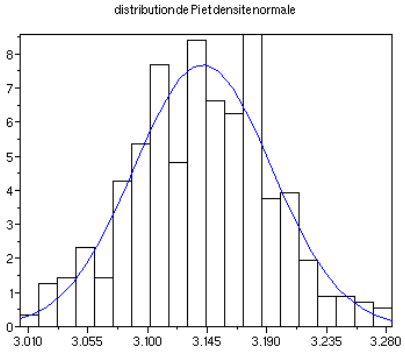
```
clf();histplot(20,pi)
```



On retrouve sur l'histogramme l'étendue (min et max) et très approximativement la moyenne. Il est difficile de dire quelque chose de l'écart-type.

5.3. Soit Z_i la variable aléatoire qui vaut 1 si le i-ème point est dans le disque, 0 sinon. Les Z_i sont indépendantes uniformément distribuées d'espérance m=pi/4 et d'écart-type sqrt(pi/4*(1-pi/4)). L'espérance de la variable aléatoire 4*(Z_1+...+Z_1000)/1000 est donc pi (linéarité de l'espérance) et sont écart-type est 4/1000*sqrt(1000*pi/4*(1-pi/4)) ce qui vaut à peu près 0.0519 . On trace la densité de la loi normale N(pi,0.0519)

```
m=3.14159;s=4/1000*sqrt(1000*m/4*(1-m/4));
function y=N(x);y=1/(sqrt(2*m)*s)*exp(-(x-m)^2/(2*s*s));endfunction
clf();fplot2d(min(pi):.01:max(pi),N)
xtitle("distribution de Pi et densite normale")
```



On observe des fluctuations des hauteurs des batons autour des valeurs prédites par le graphe de N.

6. Contrôle de la règle des 95%.

Les bornes de l'intervalle I_2sigma sont données par l'instruction

`[m-2*s,m+2*s]` qui rend ! 3.0377291 3.2454509 !

Celles de I_sigma sont données par

`[m-s,m+s]` qui rend ! 3.0896595 3.1935205 !

On compte le pourcentage d'indices i tel que pi(i) n'est pas dans I_2sigma :

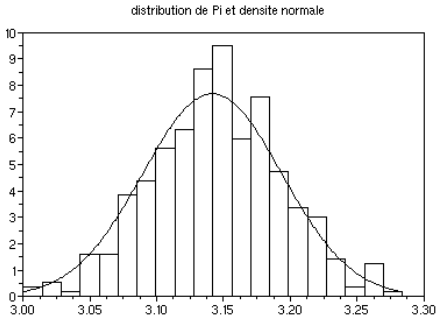
`mean(bool2s(m-2*s<pi & pi<m+2*s))` rend 0.965

`mean(bool2s(m-s<pi & pi<m+s))` rend 0.67

7. Influence de la taille de l'échantillon sur la qualité de l'estimateur.

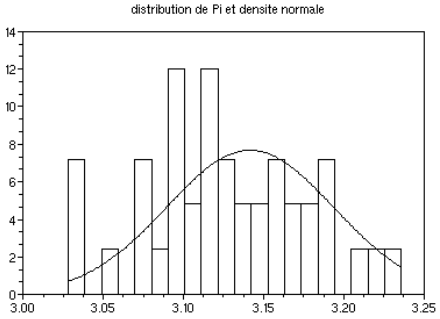
On définit successivement un vecteur pi de longeur 400 puis 40 puis 4000 :

Pi(1:400) :

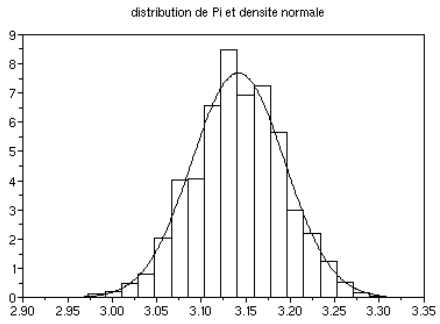


Pi(1:40) avec le script :

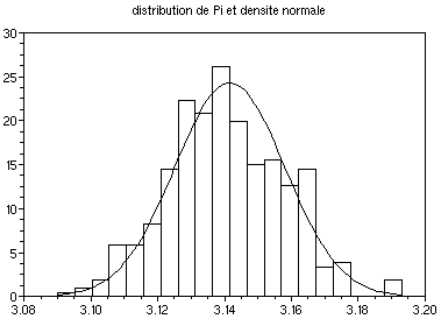
```
n1=40;//40 400 4000
n2=1000;//1000 10000
pi=[,];for
i=1:n1;x=2*rand(1,n2)-1;y=2*rand(1,n2)-1;pi(i)=4*sum(bool2s(x^2+y^2<1))/n2;end
clf();histplot(20,pi);
m=3.14159,sigma=4*sqrt(m*(4-m))/(4*sqrt(n2))
function y=n(x);y=1/(sigma*sqrt(2*m))*exp(-(x-m)^2/(2*sigma^2));endfunction
fplot2d(min(pi):.001:max(pi),n)
xtitle("distribution de Pi et densite normale")

[m-sigma,m+sigma]
[m-2*sigma,m+2*sigma]
mean(bool2s(m-sigma<pi & pi<m+sigma))
mean(bool2s(m-2*sigma<pi & pi<m+2*sigma))
```



Pi(1:4000) :



Pi(1:400), chaque valeur de Pi étant calculée avec 10000 itérations au lieu de 1000 :



Formes quadratiques et Maxima

Cf L3 Math : Algèbre et géométrie - feuille 7 ex 2 (copie) - 2007-08

Instructions exécutées avec Maxima 5.12.0 en ligne de commandes. Voir ici la version Maple.
Documentation pour Maxima, voir en particulier l'index

Définition de B2, matrice de B2 relativement à (1,X,X^2,X^3), forme quadratique sur R^4 correspondante

```
Starts dribbling to out (2007/12/13, 13:39:42).
(%i2) B2(f,g):=-f(0)*g(0)+integrate(diff(f(x),x,1)*diff(g(x),x,1),x,0,1);
(%o2) B2(f, g) := (- f(0)) g(0) + integrate(diff(f(x), x, 1) diff(g(x), x, 1), x, 0, 1)
(%i3) B2(lambda([x],x^3),lambda([x],x^3));
                                    9
(%o3)                               -
                                    5

(%i4) f:makelist(buildq([i:i],lambda([x],x^i)),i,[1,2,3]);
                          1              2              3
(%o4)         [lambda([x], x ), lambda([x], x ), lambda([x], x )]
(%i5) f:cons(lambda([x],1),f);
                            1              2              3
(%o5) [lambda([x], 1), lambda([x], x ), lambda([x], x ), lambda([x], x )]
(%i6) M:genmatrix(lambda([i,j],B2(f[i],f[j])),4,4);
                            [ - 1  0  0  0 ]
                            [              ]
                            [  0   1  1  1 ]
                            [              ]
                            [          4  3 ]
(%o6)                       [  0   1  -  - ]
                            [          3  2 ]
                            [              ]
                            [          3  9 ]
                            [  0   1  -  - ]
                            [          2  5 ]
(%i7) q:matrix([a,b,c,d]).M.transpose(matrix([a,b,c,d]));
            9 d   3 c           3 d   4 c                       2
(%o7)    d (--- + --- + b) + c (--- + --- + b) + b (d + c + b) - a
             5     2             2     3
(%i8) hessian(q/2,[a,b,c,d]);
                            [ - 1  0  0  0 ]
                            [              ]
                            [  0   1  1  1 ]
                            [              ]
                            [          4  3 ]
(%o8)                       [  0   1  -  - ]
                            [          3  2 ]
                            [              ]
                            [          3  9 ]
                            [  0   1  -  - ]
                            [          2  5 ]
(%i9) rank(M);
(%o9)                                4
(%i10) var:[a,b,c,d];
(%o10)                          [a, b, c, d]
(%i11) Gauss(quad,variable):=block([q,var,liste,i,a,l],
q:quad,var:variable,liste:[],display(q,var,liste),
while var#[] do (
 a:diff(q,var[1],2)/2,l:subst(0,var[1],diff(q,var[1])),display(a,l),
 if a=0 and l=0 then var:rest(var)
 elseif a#0 then (
  liste:append(liste,[[a,var[1]+l/(2*a)]]),q:subst(0,var[1],q)-l^2/(4*a),var:rest(var))
 else (
  i:2,while subst(0,var[i],diff(l,var[i]))=0 do i:i+1,display(var[i]),
  a:diff(q,var[i],2)/2,display(a),
  if a#0 then (
   liste:append(liste,[[a,var[i]+l/(2*a)]]),q:subst(0,var[i],q)-l^2/(4*a),var:append(rest(var,i-1-length(var)),rest(var,i)))
  else (
   a:diff(q,var[1],1,var[i],1),display(a),
   l1:subst([var[1]=0,var[i]=0],diff(q,var[1])),li:subst([var[1]=0,var[i]=0],diff(q,var[i])),display(l1,li),
   liste:append(liste,[[a/4,var[1]+var[i]+(l1+li)/a],[-a/4,var[1]-var[i]+(li-l1)/a]]),
   q:subst([var[1]=0,var[i]=0],q)-l1*li/a,
   var:append(rest(rest(var,i-1-length(var))),rest(var,i)))),
 display(q,var,liste)),
q:sum(liste[i][1]*liste[i][2]^2,i,1,length(liste)),
[liste,q])$

(%i12) q1:Gauss(q,var)[2];
              9 d   3 c           3 d   4 c                       2
       q = d (--- + --- + b) + c (--- + --- + b) + b (d + c + b) - a
               5     2             2     3

                            var = [a, b, c, d]

                                liste = []

                                  a = - 1

                                   l = 0

                9 d   3 c           3 d   4 c
         q = d (--- + --- + b) + c (--- + --- + b) + b (d + c + b)
                 5     2             2     3

                              var = [b, c, d]

                            liste = [[- 1, a]]

                                   a = 1

                                 l = 2 d + 2 c

                               2
                      (2 d + 2 c)       9 d   3 c       3 d   4 c
            q = - ------------ + d (--- + ---) + c (--- + ---)
                        4             5     2         2     3

                                 var = [c, d]

                                          2 d + 2 c
                      liste = [[- 1, a], [1, --------- + b]]
                                             2
```

```
                                    1
                                a = -
                                    3

                                l = d

                                    2
                                   d
                                q = --
                                    20

                               var = [d]

                                2 d + 2 c        1  3 d
            liste = [[- 1, a], [1, --------- + b], [-, --- + c]]
                                    2            3   2

                                    1
                                a = --
                                    20

                                l = 0

                                q = 0

                               var = []

                             2 d + 2 c        1  3 d        1
         liste = [[- 1, a], [1, --------- + b], [-, --- + c], [--, d]]
                                    2            3   2         20

                                       3 d     2
                                      (--- + c)     2
                    2 d + 2 c     2     2          d     2
(%o12)             (--------- + b)  + ---------- + -- - a
                         2                3        20
(%i13) expand(q1);
                   2                    2
                9 d                    4 c             2    2
(%o13)          ---- + 3 c d + 2 b d + ---- + 2 b c + b  - a
                 5                     3
(%i14) expand(q);
                   2                    2
                9 d                    4 c             2    2
(%o14)          ---- + 3 c d + 2 b d + ---- + 2 b c + b  - a
                 5                     3

(%i15) r:Gauss(x*y+y*z+x*z,[x,y,z]);
                             q = y z + x z + x y

                               var = [x, y, z]

                                 liste = []

                                    a = 0

                                 l = z + y

                                 var  = y
                                    2

                                   a = 0

                                   a = 1

                                  l1 = z

                                  li = z

                                        2
                                 q = - z

                                 var = [z]

                          1                   1
                liste = [[-, 2 z + y + x], [- -, x - y]]
                          4                   4

                                  a = - 1

                                   l = 0

                                   q = 0

                                 var = []

                       1                   1
             liste = [[-, 2 z + y + x], [- -, x - y], [- 1, z]]
                       4                   4

          1                   1
(%o15) [[[-, 2 z + y + x], [- -, x - y], [- 1, z]],
          4                   4
                                                    2              2
                                              (2 z + y + x)     2   (x - y)
                                              -------------- - z  - --------]
                                                    4                 4
(%i16) display(r[1],r[2]);
                       1                   1
                r  = [[-, 2 z + y + x], [- -, x - y], [- 1, z]]
                 1     4                   4

                                     2              2
                           (2 z + y + x)     2   (x - y)
                      r  = -------------- - z  - --------
                       2         4                 4

(%o16)                               done
(%i17) is(r[2]=x*y+y*z+x*z);
(%o17)                               false
(%i18) is(expand(r[2])=x*y+y*z+x*z);
(%o18)                               true
(%i20)
```

Scilab et la dynamique de l'approximation diophantienne d'un réel

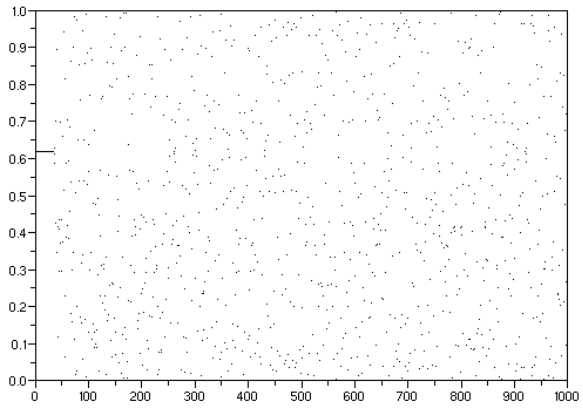
Scilab cf help('int'), etc

Etude du système dynamique x -> 1/(x-int(x)) sur [1,\infty[\Q ou de façon equivalente x \mapsto 1/x - int(1/x) sur [0,1]\Q. C'est ce dernier qu'on représente graphiquement.

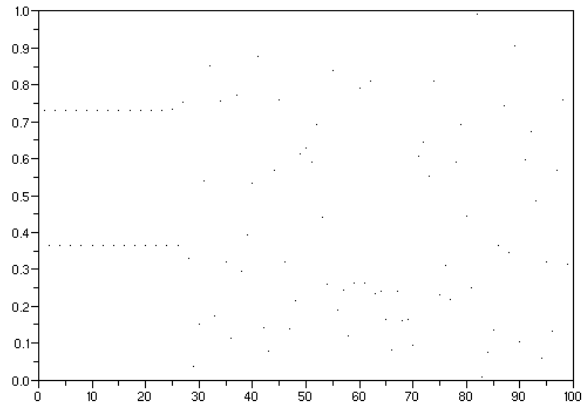
Le calcul du spectre de matrices 2x2 de la forme [k,1;1,0] ou d'un produit de matrices de cette forme sert à trouver les points fixes de l'extension à P^1(R) d'une section de f ou d'une composée de telles sections, ce qui donne des points fixes de f ou de composées de f avec elle même.

Les erreurs de calculs propres à l'approximation décimale tronquée d'un réel dans scilab conduisent à des phénomènes abérents : en prenant x=1.61, on devrait obtenir une erreur de division par 0 après 9 itérations de f (les itérés de f ne sont définis qu'en dehors des nombres rationnels). Or on obtient un phénomène périodique stable de période 4. Sauf erreur de ma part le système dynamique associé à f n'a pas d'orbite périodique stable.
En fait l'erreur dans le calcul de f(x) est multipliée par x*f(x) (>2) lors de chaque itération. Les valeurs trouvées après itérations n'ont très vite aucune pertinence.

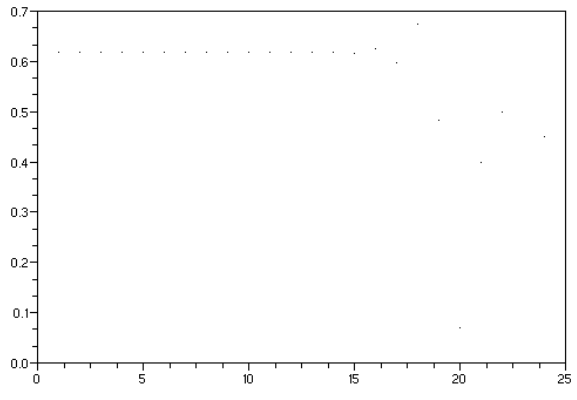
```
-->//Calcul de valeurs propres et de vecteurs propres

--> A=[1,1;1,0];[D,P]=bdiag(A)
 P  =

    0.8506508  - 0.5257311
    0.5257311    0.8506508
 D  =

    1.618034    0.
    0.        - 0.6180340

--> A*P(:,1)-D(1,1)*P(:,1) //verification
 ans  =

    0.
    0.

-->//points fixes des sections de f

--> x=[];for i=1:10;a=spec([i,1;1,0]);x(i)=a(2);end;x
 x  =

    1.618034
    2.4142136
    3.3027756
    4.236068
    5.1925824
    6.1622777
    7.1400549
    8.1231056
    9.1097722
    10.09902

--> (1+sqrt(5))/2
 ans  =

    1.618034

-->1/(sqrt(2)-1)
 ans  =

    2.4142136

--> function y=f(x);y=1/(x-int(x));endfunction

-->// Comportement des iteres de f au voisinage du premier point fixe

--> y=[];y(1)=(1+sqrt(5))/2;for i=1:1000;y(i+1)=f(y(i));end;
--> clf();plot2d(1:1000,y(1:1000).^(-1),0)
```



```
-->// Points de periode 2

-->[D,P]=bdiag([1,1;1,0]*[2,1;1,0])
 P  =

    0.8068982  - 0.3577590
    0.5906905    0.9774159
 D  =

    3.7320508    0.
    0.           0.2679492

-->a=P(1,1)/P(2,1)
 a  =

    1.3660254

-->function []=dessin(a,n);y=[];y(1)=a;for i=1:n-
1;y(i+1)=f(y(i));end;plot2d(1:n,y(1:n).^(-1),0);endfunction

-->clf();dessin(a,100)
```

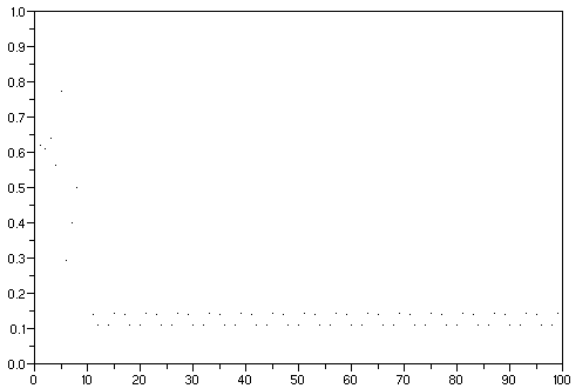


```
-->//Rq sur le spectre :

-->a=bdiag([5,1;1,0]*[3,1;1,0]*[4,1;1,0])
 a  =

    72.013886    0.
    0.         - 0.0138862

-->x=[];for i=1:100;a=spec([i,1;1,0]);x(i)=a(2);end;
-->x(find(x>72,1))
 ans  =

    72.013886

-->a(1,1)-x(find(x>72,1))
 ans  =

    1.421D-14

-->// Les iteres de f ne sont pas definis sur les nombres rationnels :

-->function i=g(a,b);i=1;while b<>0;r=a-
int(a/b)*b;a=b;b=r;i=i+1;end;endfunction

-->g(1618034,1000000) //nbre d'iterations pour une division par zero si
les calculs etaient exacts
 ans  =

    23.

-->2*log(1618034)/log(2)
 ans  =

    41.251621

-->clf();dessin(1.618034,24)
```

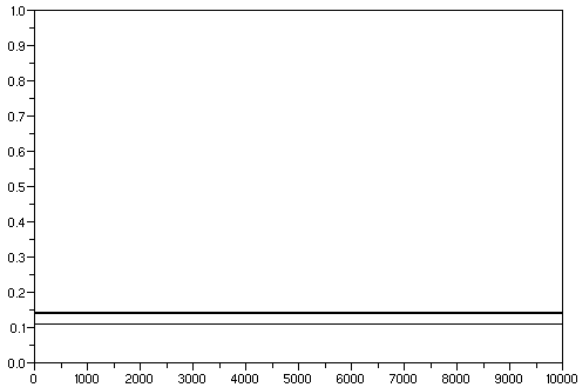


```
-->clf();dessin(1.6,24) //essai avec un nbre rationnel plus proche d'un
entier
 !--error 27
division by zero...
at line       2 of function f called by :
line     2 of function dessin called by :
clf();dessin(1.6,24)

-->g(16,10)
 ans  =

    5.

-->g(161,100)
 ans  =

    9.
```

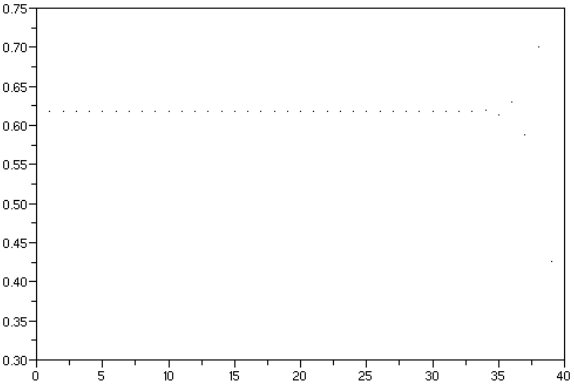
```
-->clf();dessin(1.61,100) //Y aura t-il une erreur de division par 0 ?
```



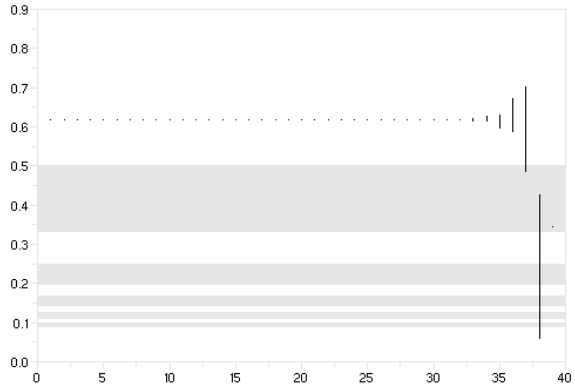
```
-->clf();dessin(1.61,10000)
```



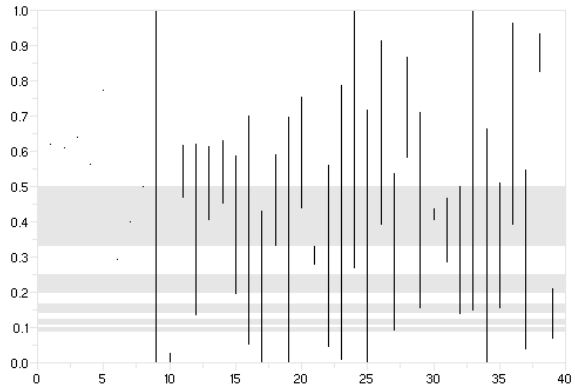
```
-->// Controle de la precision dans l'approximation decimale d'un reel par
Scilab

-->number_properties("radix")
 ans  =

    2.

-->number_properties("digits")
 ans  =

    53.

-->a=(1+sqrt(5))/2;log(a-f(a))/log(2)
 ans  =

  - 52.

-->b=2*log(a)/log(2) //nbre theorique de chiffres significatifs perdus sur
les 53 (en base 2) lors de chaque iteration
 b  =

    1.38848382726123476

-->53/b //nbre theorique d'iterations avant une erreur de l'ordre de
grandeur de a
 ans  =

    38.1711323959327444

-->clf();dessin(a,40)
```



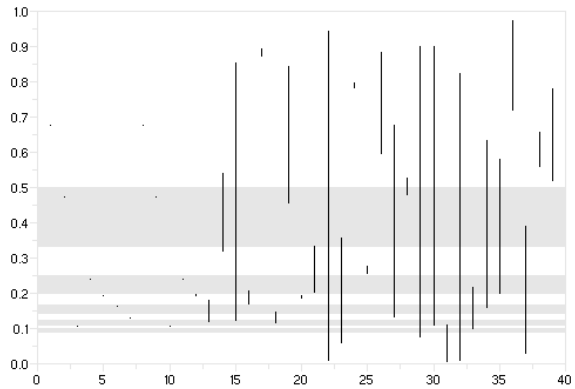
```
-->log(abs(y(37)-a))/log(2) //ecart avec a après 36 iterations
 ans  =

  - 3.58964805665926656

-->53-36*b //ecart theorique
 ans  =

    3.01458221859554953

-->// intervalle de confiance - on approxime f([a-eps,a+eps]) par [f(a-
eps),f(a+eps)] valable si [a-eps,a+eps] est inclu dans un intervalle
]k,k+1[. Si ]a-eps,a+eps[ contient un entier alors f(]a-eps,a+eps[) est
]1,+infty[

-->// Troncature decimale vecue :

-->a=(1+sqrt(5))/2;[(a+2^(-53))-a,(a+2^(-52))-a]
 ans  =

    0.    2.220D-16

-->function []=dessin(a,n);y1(1)=a-2^(-52);y2(1)=a+2^(-52);for i=1:n-
1;y1(i+1)=f(y1(i));y2(i+1)=f(y2(i));end;
xset("color",color(230,230,230));for i=1:10;xfrect(0,1/(2*i),40,1/(2*i*
(2*i+1)));end;for i=1:n;plot2d([i,i],
[1/y1(i),1/y2(i)],style=1);end;endfunction

Warning :redefining function: dessin


-->clf();dessin((1+sqrt(5))/2,40) // a comparer avec la derniere figure
ci-dessus :
```



```
-->clf();dessin(1.61,40) //a comparer avec le phenomene de periodicite
observe plus haut :
```



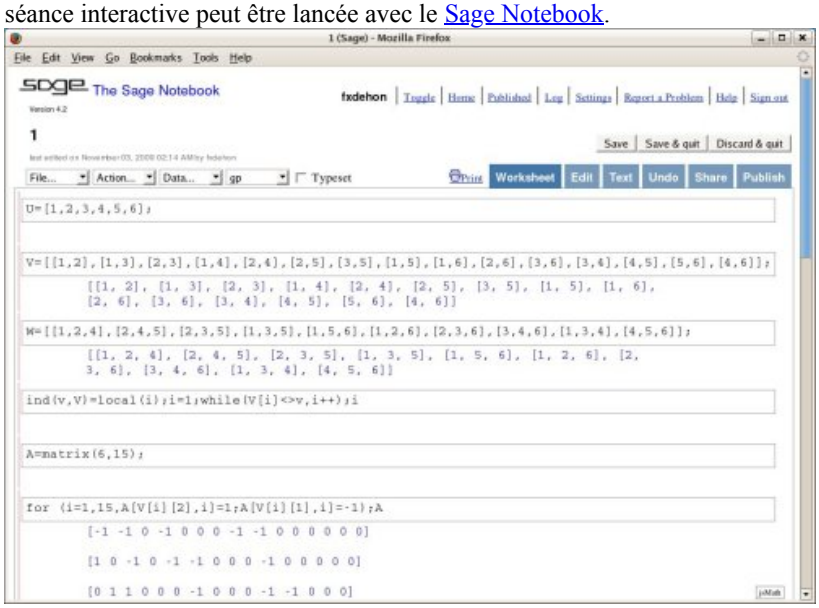
```
-->//Points de periode 7 ?

-->[D,P]=bdiag([1,1;1,0]*[2,1;1,0]*[9,1;1,0]*[4,1;1,0]*[5,1;1,0]*
[6,1;1,0]*[7,1;1,0]);clf();dessin(P(1,1)/P(2,1),40)
```



fx dehon, 24 janvier 2008

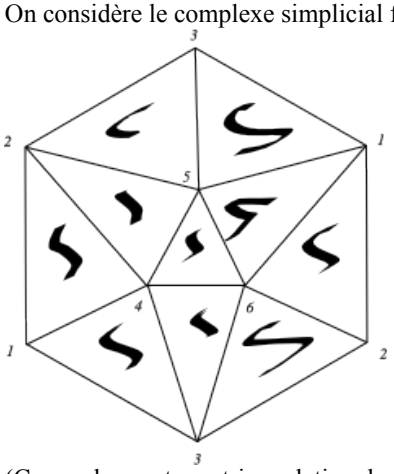
Calcul de l'homologie d'une triangulation de RP^2 avec GP/PARI

Documentation en ligne de GP/PARI , et cette page pour des manuels, tutoriels, etc. Un script gp/pari peut être exécuté avec Wims ici. Une séance interactive peut être lancée avec le Sage Notebook.



Forme normale des matrices à coefficients entiers : voir la notice sur Wikipedia ou cette page.

On considère le complexe simplicial formé des 2-simplexes et de leurs faces représenté ci-dessous :



(Ce comlexe est une triangulation du plan projectif réel.) On oriente chaque simplexe de ce complexe par la numérotation indiquée des sommets du complexe. On définit dans GP/PARI les bases des groupes des 0-chaînes, 1-chaînes et 2-chaînes et les matrices des opérateurs de bord d_1 et d_2 (cf [Munkres, Elements of Algebraic Topology, chap. 1] pour une définition de ces objets). On utilise la fonction matsnf de GP/Pari pour obtenir une forme normale des matrices des opérateurs avec les matrices de changement de bases. On en déduit immédiatement une description des groupes d'homologie du complexe simplicial.

```
                GP/PARI CALCULATOR Version 2.3.0 (released)
          i686 running linux (ix86/GMP-4.1.4 kernel) 32-bit version
        compiled: May 26 2006, gcc-4.1.0 20060304 (Red Hat 4.1.0-3)
             (readline v5.0 enabled, extended help available)

                        Copyright (C) 2000-2006 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and comes WITHOUT ANY
WARRANTY WHATSOEVER.

Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.

parisize = 4000000, primelimit = 500000

? /* Bases des 0-simplexes, 1-simplexes orientés, 2-simplexes orientés de RP^2 */
? U=[1,2,3,4,5,6];
? V=[[1,2],[1,3],[2,3],[1,4],[2,4],[2,5],[3,5],[1,5],[1,6],[2,6],[3,6],[3,4],[4,5],[5,6],[4,6]];
? W=[[1,2,4],[2,4,5],[2,3,5],[1,3,5],[1,5,6],[1,2,6],[2,3,6],[3,4,6],[1,3,4],[4,5,6]];

? /* Matrice de d_1 */
? ind(v,V)=local(i);i=1;while(V[i]<>v,i++);i /* Position de v dans V */
? A=matrix(6,15);
? for (i=1,15,A[V[i][2],i]=1;A[V[i][1],i]=-1);A /* matrice de l'opérateur de bord d_1 */
%5 =
[-1 -1 0 -1 0 0 0 -1 -1 0 0 0 0 0 0]

[1 0 -1 0 -1 -1 0 0 0 -1 0 0 0 0 0]
```

```
[0 1 1 0 0 0 -1 0 0 0 -1 -1 0 0 0]

[0 0 0 1 1 0 0 0 0 0 0 1 -1 0 -1]

[0 0 0 0 0 1 1 1 0 0 0 0 1 -1 0]

[0 0 0 0 0 0 0 0 1 1 1 0 0 1 1]

? /* rendu TeX de A */
? system("rm -f A.tex");writetex("A.tex",A);system("tex2im -r 100x100 -f gif -o A.gif A.tex");
```

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

```
? /* forme normale de Smith */
? ?matsnf
matsnf(x,{flag=0}): Smith normal form (i.e. elementary divisors) of the matrix x, expressed
as a vector d. Binary digits of flag mean 1: returns [u,v,d] where d=u*x*v, otherwise only
the diagonal d is returned, 2: allow polynomial entries, otherwise assume x is integral, 4:
removes all information corresponding to entries equal to 1 in d.

? M=matsnf(A,1);
? M[3]
[0 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 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

? /* Conclusion : le noyau de d_1 est de dim 10 ; l'image de d_1 est un facteur direct de C_0 de dimension 5 */
? writetex("M2.tex",M[2]);system("tex2im -r 100x100 -f gif -o M2.gif M2.tex");
```

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

```
? ker=vecextract(M[2],"1..10");A*ker /* Les 10 premières colonnes de M[2] forment la matrice des coordonnées d'une
base du noyau de d_1 ; vérification : */
[0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0]

? /* matrice de d_2 */
? B=matrix(15,10);B=matrix(15,10);{for (j=1,10,B[ind(vecextract(W[j],[2,3]),V),j]=1; B[ind(vecextract(W[j],
[1,3]),V),j]=-1; B[ind(vecextract(W[j],[1,2]),V),j]=1);}

? writetex("B.tex",B);system("tex2im -r 100x100 -f gif -o B.gif B.tex");
```

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

```
? BB=M[2]^(-1)*B; /* Matrice de d_2 relativement à la nouvelle base de C_1 */
? writetex("BB.tex",BB);system("tex2im -r 100x100 -f gif -o BB.gif BB.tex");
```

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

```
? /* d_2 est bien à valeurs dans ker(d_1) ! */
? BBB=vecextract(BB,"1..10","1..10"); /* 10 premières lignes et les 10 colonnes de BB */

? N=matsnf(BBB,1); /* mise sous forme normale */
? writetex("N3.tex",N[3]);system("tex2im -r 100x100 -f gif -o N3.gif N3.tex");
```

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
? /* Conclusion : ker(d_2)={0} ; im(d_2) est de dimension 10=dim(ker(d_1)) mais n'est pas un facteur direct de
ker(d_1); ker(d_1)/im(d_2)=Z/2Z */
```

F.X. Dehon, 11 décembre 2008, dehon@unice.fr

L2 ThJeux 19jan11

Un corrigé du devoir no 1 : méthode algorithmique

Soit le jeu donné par la matrice A=

```
1,-1,2,1
2,1,-3,0
0,1/2,-2,1/4
```

(3 lignes et 4 colonnes). On cherche les stratégies mixtes prudentes des joueurs 1 et 2.
Une stratégie mixte de J1 est un triplet de réels

```
x,y,z
```

vérifiant (1) x≥0, y≥0, z≥0 et x+y+z=1. Elle est prudente si elle maximise le plus mauvais gain, lequel s'exprime comme le minimum des trois colonnes du produit matriciel (x,y,z)*A=

```
x+2*y,-x+(y+1/2*z),2*x+(-3*y-2*z),x+1/4*z
```

(On a utilisé le multiplicateur de matrices de WIMS, voir la liste d'outils).
Notons f(x,y,z) le minimum des trois réels ci-dessus ; on cherche x,y,z satisfaisant les contraintes (1) et maximisant f . La fonction f est affine par morceaux sur un domaine convexe dont les coins sont les (x,y,z) égaux à

```
1,0,0
0,1,0
0,0,1
```

Les morceaux eux mêmes sont des parties convexes du plan d'équation x+y+z=1 et leurs coins sont solutions de l'équation

```
x+y+z=1
```

et de deux autres équations parmis

```
x=0
y=0
z=0
x+2*y=-x+(y+1/2*z)
x+2*y=2*x+(-3*y-2*z)
x+2*y=x+1/4*z
-x+(y+1/2*z)=2*x+(-3*y-2*z)
-x+(y+1/2*z)=x+1/4*z
2*x+(-3*y-2*z)=x+1/4*z
```

(Les 3 premières correspondent au bords du domaine, les 6 suivantes correspondent à un cas d'égalité parmis les fonctions dont f est le min.) Il y a 31 choix de 2 équations parmis les 9 ci-dessus sans redondance, ce qui donne avec l'équation de la contrainte x+y+z=1 31 systèmes de 3 équations aux inconus x,y,z, qu'on résoud avec la solveuse d'équations linéaires de WIMS.

Rq Formalisation par le calcul matriciel et utilisation de la calculatrice de matrice de WIMS :
Posons B=

```
1,1,0,0,1,-1,2,1
1,0,1,0,2,1,-3,0
1,0,0,1,0,1/2,-2,1/4
```

Posons C=

```
1,0,0
0,1,0
0,0,0
0,0,0
0,0,1
0,0,-1
0,0,0
0,0,0
```

La solution x,y,z du système d'équations (pris en exemple)

```
x+y+z=1
x=0
x+2*y=-x+(y+1/2*z)
```

est le produit matriciel [1,0,0]*(B*C)^(-1) = 0,1/3,2/3
Posons D=

```
1,0,0
0,0,0
0,0,0
0,0,0
0,1,0
0,0,1
0,-1,0
0,0,-1
```

La solution x,y,z du système d'équations

```
x+y+z=1
x+2*y=2*x+(-3*y-2*z)
-x+(y+1/2*z)=x+1/4*z
```

est le produit matriciel [1,0,0]*(B*D)^(-1) = -1/6,-5/6,2

Un système peut très bien ne pas avoir de solution ou une infinité de solutions ou une solution ne vérifiant pas les contraintes x≥0, y≥0,

z≥0. Un coin correspond forcément à une solution unique vérifiant les contraintes ; on ne retient que de telles solutions. On obtient pour x,y,z les solutions suivantes :

```
0,0,1
0,1,0
1,0,0
0,1/3,2/3
0,1/9,8/9
1/5,0,4/5
2/3,0,1/3
5/11,0,6/11
1/9,0,8/9
9/13,0,4/13
5/6,1/6,0
4/7,3/7,0
1/3,2/3,0
3/4,1/4,0
1/7,2/21,16/21
7/10,1/30,4/15
8/17,1/17,8/17
```

On note P la matrice ci dessus. Le produit P*A est la matrice des gains moyen de J1 pour chaque stratégie pure de J2 :

```
0,1/2,-2,1/4
2,1,-3,0
1,-1,2,1
2/3,2/3,-7/3,1/6
2/9,5/9,-19/9,2/9
1/5,1/5,-6/5,2/5
2/3,-1/2,2/3,3/4
5/11,-2/11,-2/11,13/22
1/9,1/3,-14/9,1/3
9/13,-7/13,10/13,10/13
7/6,-2/3,7/6,5/6
10/7,-1/7,-1/7,4/7
5/3,1/3,-4/3,1/3
5/4,-1/2,3/4,3/4
1/3,1/3,-32/21,1/3
23/30,-8/15,23/30,23/30
10/17,-3/17,-3/17,10/17
```

Le minimum de chaque ligne est la valeur de f en la stratégie mixte de J1 correspondant à cette ligne :

```
-2
-3
-1
-7/3
-19/9
-6/5
-1/2
-2/11
-14/9
-7/13
-2/3
-1/7
-4/3
-1/2
-32/21
-8/15
-3/17
```

La plus grande valeur de f est -1/7 atteinte en la ligne 12 c'est à dire pour la stratégie mixte

```
4/7,3/7,0
```

On sait que f atteint son maximum en au moins l'un des coins des morceaux sur lesquels elle est affine et que l'ensemble des points où f est maximale est l'enveloppe convexe des coins où elle est maximale. Ici elle est maximale en le seul coin (x,y,z)=(4/7,3/7,0) donc (4/7,3/7,0) est la seule stratégie mixte prudente du joueur 1.

En remplaçant la matrice A par l'opposée de sa transposée et en reprenant la méthode ci-dessus on obtient les stratégies mixtes prudentes du joueur 2.

Distribution d'une loi de va à support fini

Author	**FX D**
Date	**2017-12-04T15:21:16**
Project	**3956b162-a539-41cc-a4a8-9496897c49b0**
Location	**2017-11-17-161438.sagews**
Original file	**2017-11-17-161438.sagews**

Distribution d'une loi de va à support fini

f4 - ex.1

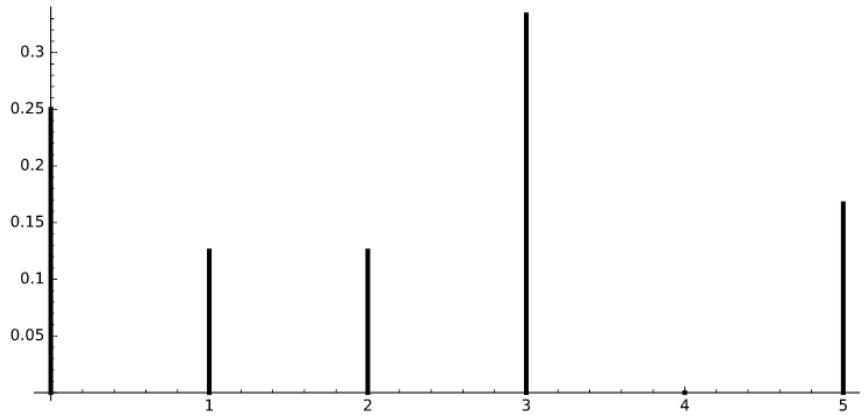
```
p=[1/4,1/8,1/8,1/3,0,1/6]
d=[[k,p[k]] for k in range(len(p))] #encodage de la distribution
print "d=",d
show(transpose(matrix(d)))
#print "(test) somme(d[1,:])=",sum(d[1,:])
```

```
d= [[0, 1/4], [1, 1/8], [2, 1/8], [3, 1/3], [4, 0], [5, 1/6]]
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ \frac{1}{4} & \frac{1}{8} & \frac{1}{8} & \frac{1}{3} & 0 & \frac{1}{6} \end{pmatrix}$$

Dessin

```
def barplot(d,t=1):
    return(sum([line([(d[k][0],0),(d[k][0],d[k][1])],thickness=t,color='black') for k in range(len(d))]))
barplot(d,3)
```



Espérance, variance, écart-type d'une va de distribution à support fini

```
def M1(d):
    return(sum([d[k][0]*d[k][1] for k in range(len(d))]))
def M2(d):
    return(sum([d[k][0]^2*d[k][1] for k in range(len(d))]))
m=M1(d);s=sqrt(M2(d)-M1(d)^2)
print "E(d)=",m,"=",m.n(digits=3)
print "sigma(d)=",s,"=",s.n(digits=3)
```

```
E(d)= 53/24 = 2.21
sigma(d)= 1/24*sqrt(1679) = 1.71
```

Simulation d'une va de loi donnée

```
transpose(matrix(d))[1,:].list()
```

```
[1/4, 1/8, 1/8, 1/3, 0, 1/6]
```

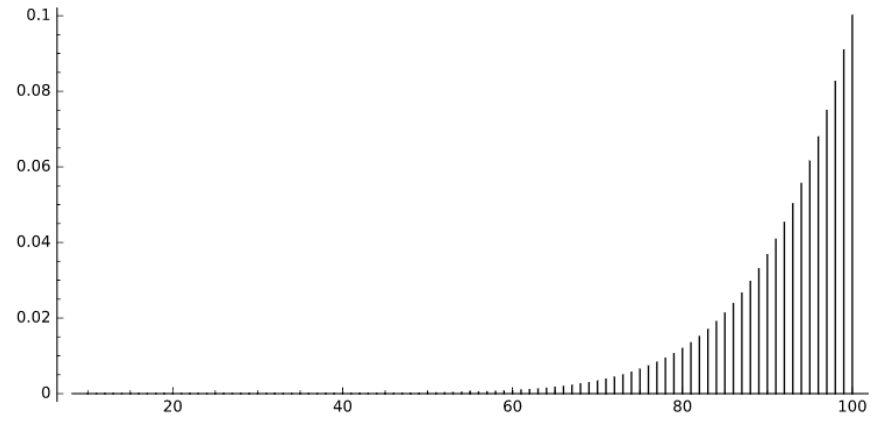
```
X = GeneralDiscreteDistribution(transpose(matrix(d))[1,:].list())
X.get_random_element()
```

```
3
```

f3-ex6

loi de max(A) où A est un ensemble de 10 éléments choisis au hasard dans {1,...,100}.

```
dmax=[[k,binomial(k-1,9)/binomial(100,10)] for k in range(10,101)]
#show(matrix(dmax).transpose())
barplot(dmax)
```

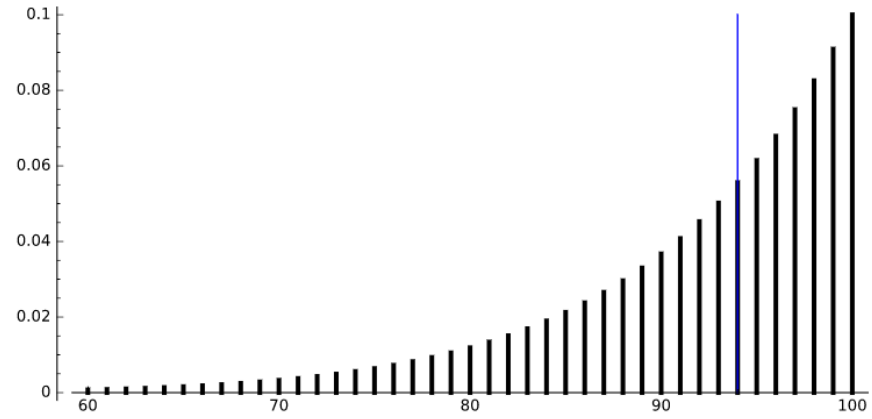


médiane ?

```
k=len(dmax)-1;c=dmax[k][1]
while c<.5:
    k=k-1;c=c+dmax[k][1]
med=dmax[k][0]
print "med=", med
print "P(X>=med)=", c.n(digits=3)
print "P(X=med)=", dmax[k][1].n(digits=3)

```

```
med= 94
P(X>=med)= 0.533
P(X=med)= 0.0556
```

```
barplot(dmax[50:],3)+line([(med,0),(med,.1)])
```

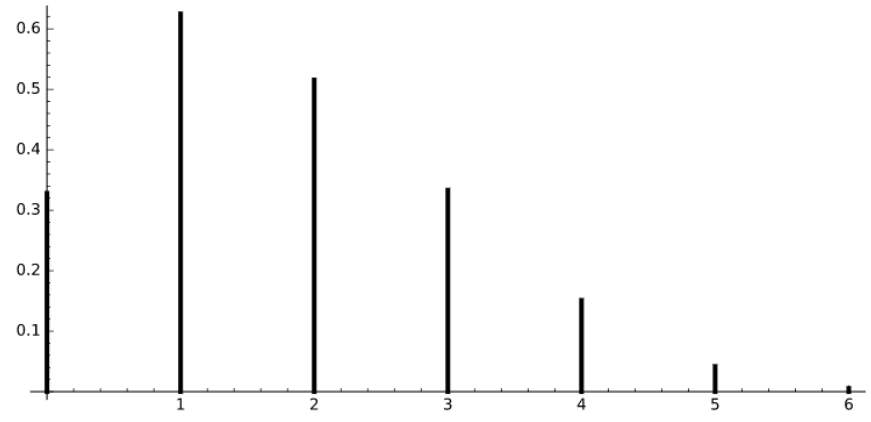


boîte à moustache ?

f3-ex7

```
d1=[[k,sum([binomial(i+1,k)*1/2^i for i in range(6)])/6] for k in range(7)]
print "Valeur(s) modale(s) ?\n"
barplot(d1,3)
```

```
Valeur(s) modale(s) ?
```

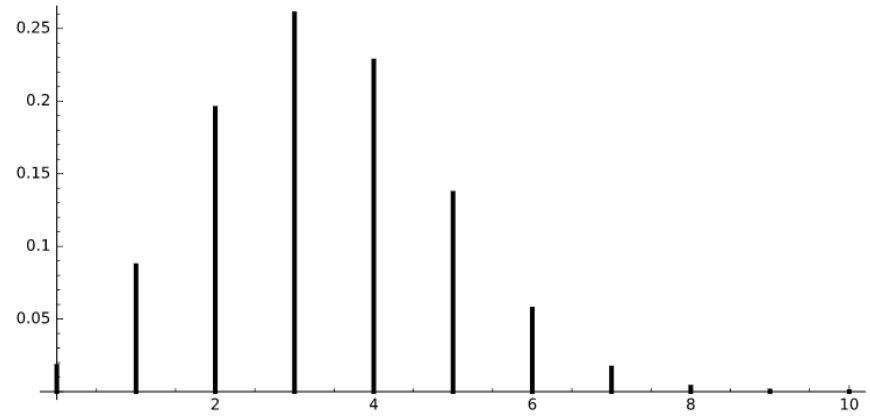


Loi binomiale

```
n=10;p=1/3
b=[[k,binomial(n,k)*p^k*(1-p)^(n-k)] for k in range(n+1)]
```

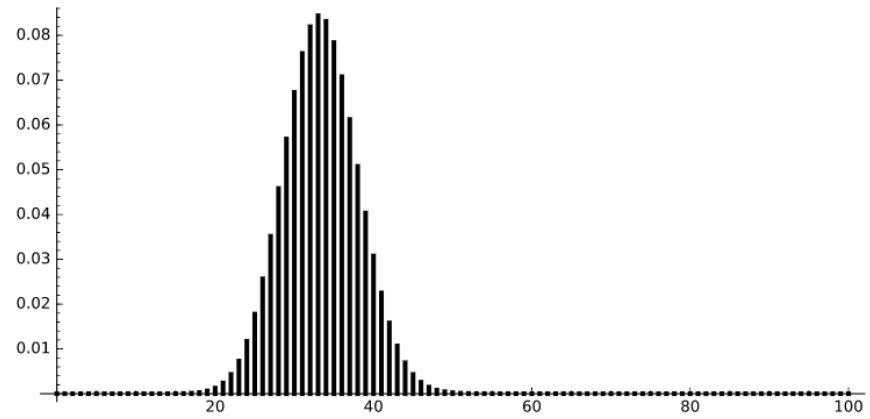
```
#transpose(matrix(b)).n(digits=2)
print "E(b)=",M1(b),"=",M1(b).n(digits=2)
print "V(b)",M2(b)-M1(b)^2
print "sigma(b)=",sqrt(M2(b)-M1(b)^2).n(digits=2)
print "La distribution est elle symétrique par rapport à sa valeur modale ?\n"
barplot(b,3)
```

```
E(b)= 10/3 = 3.3
V(b) 20/9
sigma(b)= 1.5
La distribution est elle symétrique par rapport à sa valeur modale ?
```




```
print "Même chose avec n=100,p=1/3\n"
n=100;p=1/3
b1=[[k,binomial(n,k)*p^k*(1-p)^(n-k)] for k in range(n+1)]
#transpose(matrix(b)).n(digits=2)
print "E(b1)=",M1(b1),"=",M1(b1).n(digits=2)
print "V(b1)",M2(b1)-M1(b1)^2
print "sigma(b1)=",sqrt(M2(b1)-M1(b1)^2).n(digits=2)
print "La distribution est elle symétrique par rapport à sa valeur modale ?\n"
barplot(b1,3)
```

```
Même chose avec n=100,p=1/3

E(b1)= 100/3 = 33.
V(b1) 200/9
sigma(b1)= 4.7
La distribution est elle symétrique par rapport à sa valeur modale ?
```



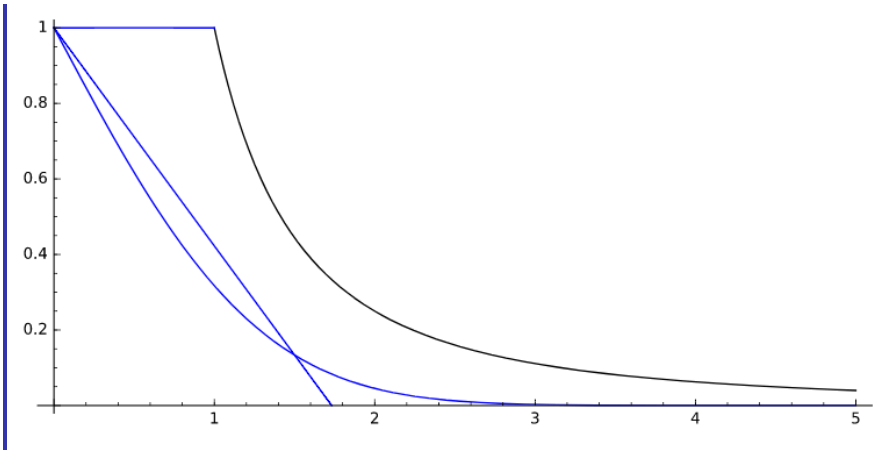

```
%md
#### Exemples de distribution de lois centrées réduites

Si $X$ est une va d'espérance $m$ et d'écart type $s\neq 0$ alors $\frac{X-m}{s}$ est d'espérance $0$ et d'écart type $1$

Qualité de l'inégalité de Tchebychev $P(|X|\geq t)\leq \frac{1}{t^2}$ pour $X$ centrée réduite
```

Exemples de distribution de lois centrées réduites

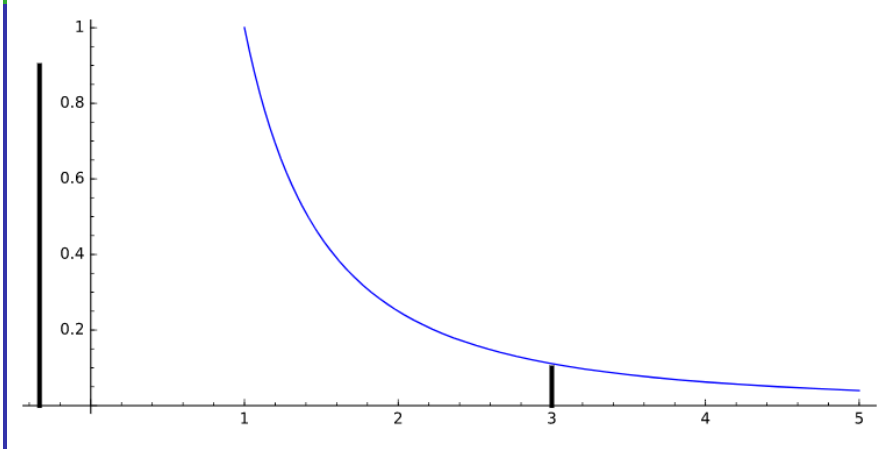
Si X est une va d'espérance m et d'écart type $s \neq 0$ alors $\frac{X-m}{s}$ est d'espérance 0 et d'écart type 1

Qualité de l'inégalité de Tchebychev $P(|X| \geq t) \leq \frac{1}{t^2}$ pour X centrée réduite

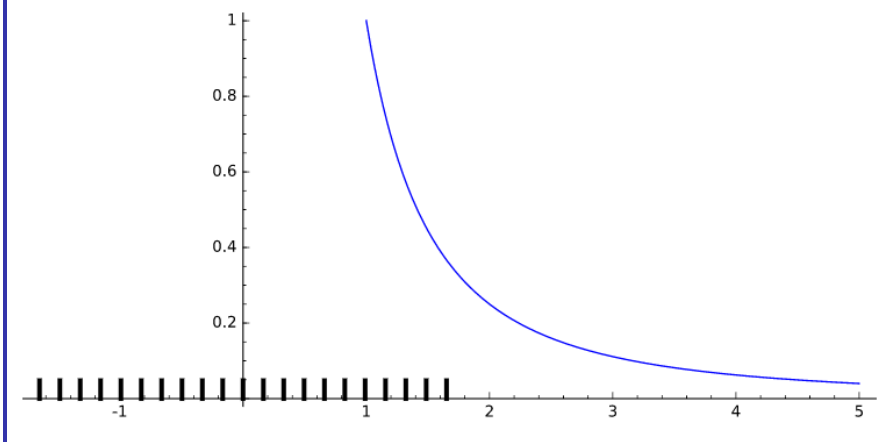
```
F(t)=1/2*erf(1/2*sqrt(2)*t) + 1/2 #F(t)=P(X<=t) pour X gaussienne centrée réduite
plot(1/t^2,t,1,5,color='black')+plot(2*(1-F(t)),t,0,5)+plot(1-t/sqrt(3),t,0,sqrt(3))+plot(1,t,0,1)
```



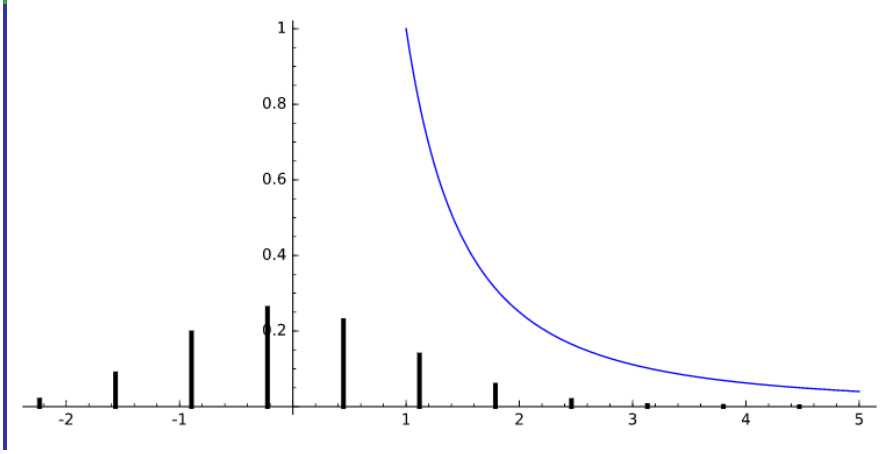

```
def denorm(d,m,s):#distribution de s*(X+m) si X est de distribution d
    return([[(l[0]+m)*s,l[1]] for l in d])

p=1/10
B=[[0,1-p],[1,p]]
barplot(denorm(B,-p,1/sqrt(p*(1-p))),3)+plot(1/t^2,t,1,5)
```




```
n=10;u=[[k,1/(2*n+1)] for k in range(1,2*n+2)]
barplot(denorm(u,-(n+1),sqrt(3)/sqrt(n*(n+1))),3)+plot(1/t^2,t,1,5)
```




```
def b(n,p): #loi binomiale
    return([[k,binomial(n,k)*p^k*(1-p)^(n-k)] for k in range(n+1)])

n=10;p=1/3
barplot(denorm(b(n,p),-n*p,1/sqrt(n*p*(1-p))),3)+plot(1/t^2,t,1,5)
```



Thème 1 - TP 1 : fonction de $\mathbb{R}$ dans $\mathbb{R}$, expression algébrique, représentation graphique, image réciproque d'un élément par la fonction.

Author	**FX D**
Date	**2017-05-06T14:30:43**
Project	**a56d8802-5740-429a-b5da-e7eb8dc5bb82**
Location	**1.sagews**
Original file	**1.sagews**

1

Thème 1 - TP 1 : fonction de $\mathbb{R}$ dans $\mathbb{R}$, expression algébrique, représentation graphique, image réciproque d'un élément par la fonction.

Application à la densité de la loi normale et ses quantiles

Sujet

On se souvient que la densité de la loi normale centrée est de la forme $f(x) = a\exp(-bx^2)$ où a et b sont tels que l'intégrale sur $\mathbb{R}$ vaut 1. Trouver la relation entre a et b puis exprimer a et b en fonction l'écart type ; donner l'expression de f avec comme paramètre l'écart type. Comment obtient on la densité associée à une espérance m au lieu de 0 ?

Dessiner le graphe de f pour différentes valeurs de l'espérance et de l'écart type.

Trouver a tel que l'intégrale de f sur [-a,a] vaut 0.95 (Cf intervalle de confiance à 95%).

Au passage : recherche de documentation, éléments de programmation en Python : déclaration - définition de variables, affectation, expressions algébriques, valeur numérique, substitution dans une expression, fonctions (voir cette page (en))

Un corrigé

Distinguer les difficultés de syntaxe (syntaxePython), de programmation (méthodes), de documentations, algorithmiques, mathématiques : faire la part des choses

1. Approche élémentaire (le moins d'astuce de programmation que possible)

```
#faire une recherche sur "sagemath intégration", menu Calculus de la feuille interactive SageMathCloud, ...
integrate(exp(-x^2),x,0,1)
integrate(exp(-x^2),x)
I=integrate(exp(-x^2),x,-infinity,infinity);I
```

```
1/2*sqrt(pi)*erf(1)
1/2*sqrt(pi)*erf(x)
sqrt(pi)
```

```
I.n?
#commande suivi de ? pour une documentation sur la commande. 1ères lettres de la commande suivi de <tab> pour listes des commandes de préfixe do
#recherche sur les mots clef "sagemath valeur numérique"
#numerical_approx? erf?
```

```
File: /projects/sage/sage-7.5/src/sage/structure/element.pyx
Signature : I.n(self, prec=None, digits=None, algorithm=None)
Docstring :
Alias for "numerical_approx()".

EXAMPLES:

   sage: (2/3).n()
   0.666666666666667
```

```
integrate(exp(-x^2),x,0,1).n(digits=3)
I.n(digits=3)
numerical_approx(I,digits=3)
```

```
0.747
1.77
1.77
```

```
reset() # supprime les variables déjà déclarées ou affectées ; la réexécution produira le même résultat
#L'intégrale comme fonction de paramètres :
var('a b');III=integrate(a*exp(-(x*b)^2),x,-infinity,infinity,hold=true);III
#sans déclaration des paramètres autres que x, la commande donnerait une erreur. Cf var? , show_identifiers()
#hold=true pour que III renvoie l'expression littérale et non celle obtenue après calcul
#sans hold=true, le calcul produit une erreur, invite à précéder le calcul de assume(b>0)
#
III.parent() #donne la structure dont III est un élément, III n'est pas une fonction
```

```
#
#Comparer avec :
reset();print
II(a,b)=integrate(a*exp(-(x*b)^2),x,-infinity,infinity,hold=true)
#cette commande déclare II comme fonction et ses arguments a,b
show_identifiers()
II
II.parent()
II(1,1)#hold=true est oublié, comparer avec integrate(exp(-(x)^2),x,-infinity,infinity,hold=true)
```

```
(a, b)
integrate(a*e^(-b^2*x^2), x, -Infinity, +Infinity)
Symbolic Ring

['a', 'II', 'b']
(a, b) |--> integrate(a*e^(-b^2*x^2), x, -Infinity, +Infinity)
Callable function ring with arguments (a, b)
sqrt(pi)
```

```
show(II) #Visualisation LaTeX de l'expression II
```

$$(a,b) \mapsto \int_{-\infty}^{+\infty} ae^{\left(-b^2x^2\right)}\,dx$$

```
assume(b>0);simplify(II)
assumptions()
#sans assume(b>0) la commande simplify(II) produit une erreur, le message d'erreur invite à utiliser simplify
#assume(b>0) reste actif pour les cellules suivantes. Pour revenir en arrière forget(b>0), pour tout oublier forget()
#cf assume? assuptions? forget?
```

```
(a, b) |--> sqrt(pi)*a/b
[b > 0]
```

```
#relation entre a et b pour que II(a,b)=1, cf recherche sur mot clef sagemath solution équation
#http://doc.sagemath.org/html/fr/tutorial/tour_algebra.html
solve(II(a,b)==1,a) #assume(b>0) implicite, sans quoi une erreur est produite
```

```
[a == b/sqrt(pi)]
```

```
#densité gaussienne centrée
a = b/sqrt(pi);f(x)=a*e^(-b^2*x^2)
f
integrate(f(x),x,-infinity,infinity)
```

```
x |--> b*e^(-b^2*x^2)/sqrt(pi)
1
```

```
#Espérance
E1=integrate(x*f(x),x,-infinity, infinity);E1
```

```
0
```

```
#Variance
E2=integrate(x^2*f(x),x,-infinity, infinity);V=E2-E1^2;V
show(V)
```

```
1/2/b^2
```

$$\frac{1}{2\,b^2}$$

```
#b en fonction de l'écart-type puis expression de f
var('s');solve(V==s^2,b)
```

```
s
[b == -1/2*sqrt(2)/s, b == 1/2*sqrt(2)/s]
```

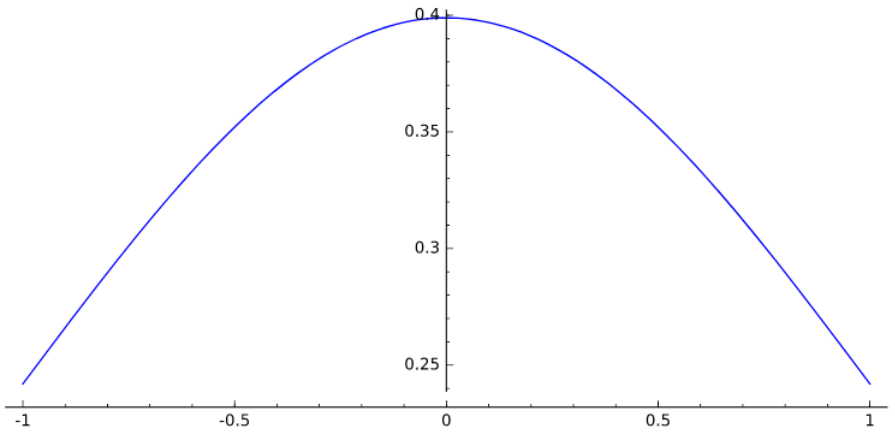
```
#substitution à la main dans l'expression de f
b = 1/2*sqrt(2)/s
f(x)=b*e^(-b^2*x^2)/sqrt(pi)
f
show(f)
```

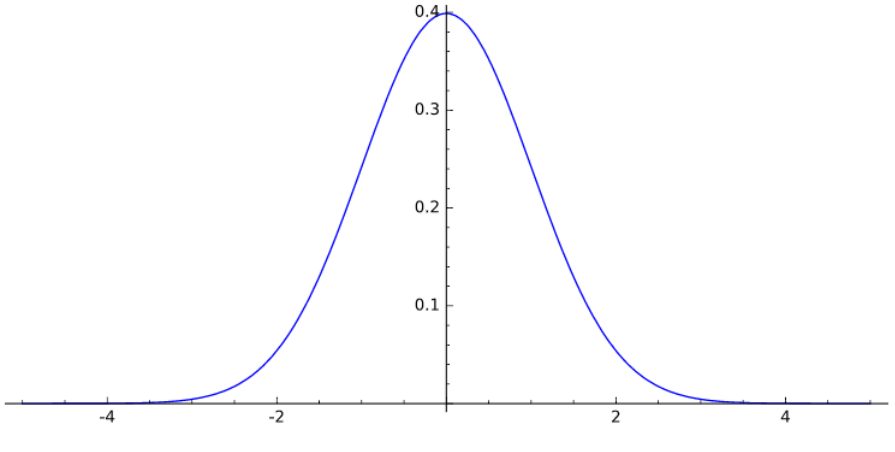
```
x |--> 1/2*sqrt(2)*e^(-1/2*x^2/s^2)/(sqrt(pi)*s)
```

$$x \mapsto \frac{\sqrt{2}e^{\left(-\frac{x^2}{2\,s^2}\right)}}{2\,\sqrt{\pi}s}$$

```
#Représentation graphique
#s=1;plot(f) produit une erreur : s=1 n'est pas pris en compte dans l'évaluation de f. La bonne méthode est f(s=1) cf "sagemath substitution exp
s=1;f(1)
f(s=1)
g(x)=f(s=1)
g
plot(g)
#google "sagemath plot" ou plot?
plot(g,-5,5)
```

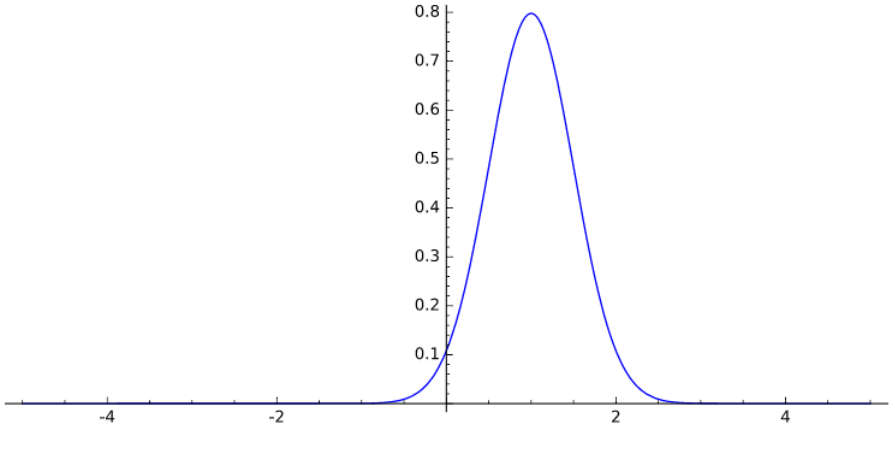
```
1/2*sqrt(2)*e^(-1/2/s^2)/(sqrt(pi)*s)
1/2*sqrt(2)*e^(-1/2*x^2)/sqrt(pi)
x |--> 1/2*sqrt(2)*e^(-1/2*x^2)/sqrt(pi)
```






```
#densité gaussienne centrée en 1, d'écart type 1/2
g(x)=f(s=1/2,x=x-1) #drole de syntaxe !
g
plot(g,-5,5)
```

```
x |--> sqrt(2)*e^(-2*(x - 1)^2)/sqrt(pi)
```




```
#a en fonction de s tel que l'intégrale de f sur [-a,a] vaut 0.95
var('a');integrate(f(x),x,-a,a)
```

```
a
erf(1/2*sqrt(2)*a/s)
```

```
var('s') #réinitialisation de s
solve(erf(1/2*sqrt(2)*a/s)==0.95,a)
```

```
s
[a == sqrt(2)*s*inverse_erf(19/20)]
```

```
#a est proportionnel à s, ça se voyait déjà sur l'équation
#valeur numérique de a pour s=1 ?
s=1;(sqrt(2)*s*inverse_erf(19/20)).n()
```

```
Error in lines 1-1
Traceback (most recent call last):
  File "/projects/sage/sage-7.5/local/lib/python2.7/site-packages/smc_sagews/sage_server.py", line 995, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in
NameError: name 'inverse_erf' is not defined

*** WARNING: Code contains non-ascii characters ***
```

```
#inverse_erf donné dans une réponse par Sage n'est pas (en mai 2017) un objet de Sage
#voir à ce sujet https://ask.sagemath.org/question/8118/evaluating-inverse-erf/
#Une version numérique de inverse_erf peut être importée de la librairie Python Scipy, cf recherche sur mots clef "python inverse erf"
from scipy.special import erfinv as inverse_erf
inverse_erf?
#On revient en arrière par la commande
#del inverse_erf
```

```
File: /projects/sage/sage-7.5/local/lib/python2.7/site-packages/scipy/special/basic.py
Signature : inverse_erf(y)
Docstring :
Inverse function for erf.
```

```
s=1;(sqrt(2)*s*inverse_erf(19/20)).n()
```

```
1.95996398454005
```

2. Résolution numérique d'une équation

```
#Méthode implémentée dans Python, cf find_root?
#Recherche d'une valeur approchée de a entre 0 et 4
find_root(erf(1/2*sqrt(2)*a)==0.95,0,4)
```

```
1.959963984540053
```

```
#Algorithme de dichotomie : recherche d'un 0 de la fonction h entre a et b à la précision prec si h change de signe entre a et b
def dicho(h,a,b,prec):
    if h(a)*h(b)>0:
        print('peut être pas de solution')
    else:
        while b-a>prec:
            i=(a+b)/2
            if h(i)*h(b)>0:b=i
            else:a=i
    return(a)

h(x)=erf(1/2*sqrt(2)*x)-0.95
dicho(h,0,4,0.001).n() #seules les 3 premières décimales après la virgule sont pertinentes !
```

```
1.95996093750000
```

```
#Vérification
show(integrate(f(s=1),x,-1.96.n(digits=3),1.96.n(digits=3),hold=true),'=',integrate(f(s=1),x,-1.96,1.96).n())
```

$$\int_{-1.96}^{1.96} \frac{\sqrt{2}e^{\left(-\frac{1}{2}x^2\right)}}{2\sqrt{\pi}}\,dx = 0.950004209703559$$

3. Fonction de répartition

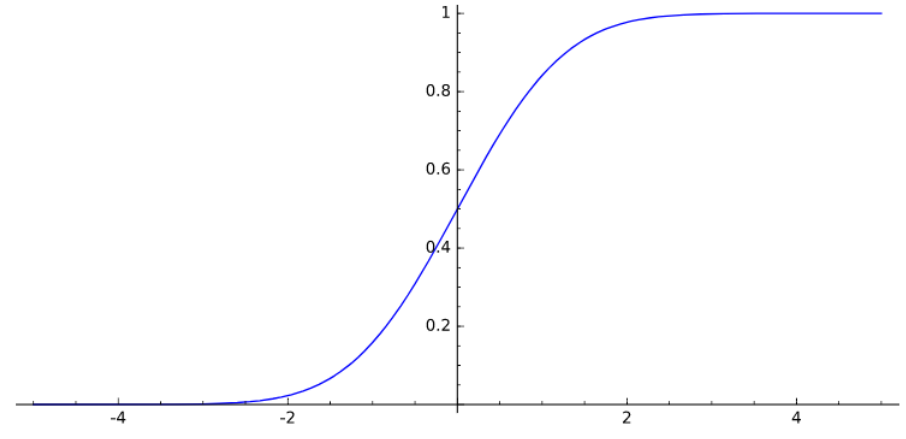
```
#Fonction de répartition de la densité gaussienne centrée réduite (s=1)
F(t)=integrate(f(s=1),x,-infinity,t);F
```

```
t |--> 1/4*sqrt(2)*(sqrt(2)*sqrt(pi)*erf(1/2*sqrt(2)*t) + sqrt(2)*sqrt(pi))/sqrt(pi)
```

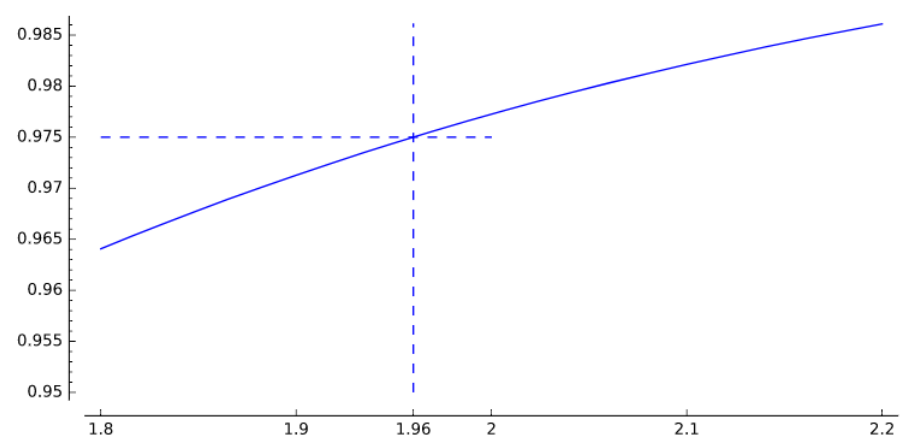
```
#Méthode simplify disponible pour F (faire F. suivi de <tab>)
F.simplify_full() #bizarement on obtient F(t) plutôt que F
F(t)=F.simplify_full();F
```

```
1/2*erf(1/2*sqrt(2)*t) + 1/2
t |--> 1/2*erf(1/2*sqrt(2)*t) + 1/2
```

```
plot(F,-5,5)
```




```
#On cherche a tel que F(a)-F(-a)=0.95 soit F(a)=0.975, d'abord sur le dessin
#1.96 est choisi "à la main" et on vérifie par le tracé des lignes en pointillés que c'est une bonne approximation
#ticks en option : voir http://doc.sagemath.org/html/en/reference/plotting/sage/plot/plot.html
plot(F,1.8,2.2)+line([(1.8,0.975),(2,0.975)],linestyle='--',ticks=[[1.8,1.9,1.96,2,2.1,2.2],None])+line([(1.96,0.95),(1.96,F(2.2))],linestyle='--')
```




```
#Méthode numérique avec find_root
find_root(F(x)==0.975,0,4)
```

```
1.959963984540053
```

```
#Méthode formelle
solve(F(x)==0.975,x) #même solution que précédemment, même pb avec inverse_erf
```

```
[x == sqrt(2)*inverse_erf(19/20)]
```

4. Programmation plus efficace. Qu'est qui change par rapport à la section 1. ?

```
show_identifiers()
reset();print;show_identifiers()
```

```
['g', 'dicho', 'V', 'F', 'h', 's', 't', 'inverse_erf', 'a', 'E1', 'E2', 'II', 'b', 'f']

[]
```

```
var('a','b')
f(x)=a*exp(-(b*x)^2);f
var('m')
assume(b != 0);I=integrate(f(x-m),x,-infinity,infinity);I #sans assume(b != 0) une erreur est produite dont le texte invite à une telle commande
```

```
(a, b)
x |--> a*e^(-b^2*x^2)
m
sqrt(pi)*a/sqrt(b^2)
```

```
assume(b>0);simplify(I)
```

```
sqrt(pi)*a/b
```

```
S=solve(I==1,a);S
```

```
[a == b/sqrt(pi)]
```

```
f=f.subs(S[0]);f
#alternative : f=f(a=S[0].rhs())
```

```
x |--> b*e^(-b^2*x^2)/sqrt(pi)
```

```
E1=integrate(x*f(x-m),x,-infinity,infinity);E1 #espérance
E2=integrate(x^2*f(x-m),x,-infinity,infinity);V=E2-E1^2;V #variance
V=V.simplify_full();V #V ne devrait pas dépendre de m !
show('V = ',V)
```

```
m
-m^2 + 1/2*(2*sqrt(pi)*m^2/b + sqrt(pi)/b^3)*b/sqrt(pi)
1/2/b^2
```

$$V = \frac{1}{2\,b^2}$$

```
var('s');B=solve(s^2==V,b);B
```

```
s
[b == -1/2*sqrt(2)/s, b == 1/2*sqrt(2)/s]
```

```
f=f.subs(B[1]);show('f_s (x) = ',f)
```

$$f\_s\ (x) = x \mapsto \frac{\sqrt{2}e^{\left(-\frac{x^2}{2\,s^2}\right)}}{2\,\sqrt{\pi}s}$$

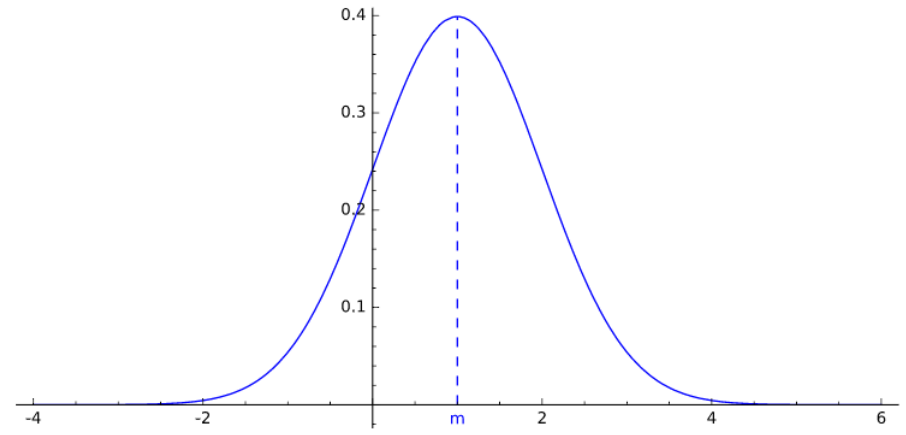
```
la='$'+latex(f)+'$'
html('<h3><div align="center"> $f_s(x)\ =\ $+la+'</div></h3>')
```

$$f_s(x) \;=\; x \mapsto \frac{\sqrt{2}e^{\left(-\frac{x^2}{2\,s^2}\right)}}{2\,\sqrt{\pi}s}$$

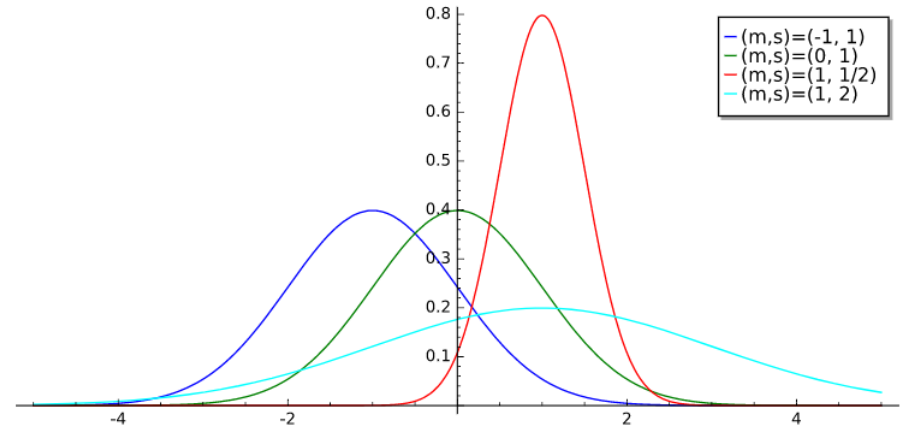
```
#densité gaussienne centrée en m
g(m,s,x)=f(x-m)
show(g)
```

$$(m, s, x) \mapsto \frac{\sqrt{2}e^{\left(-\frac{(m-x)^2}{2\,s^2}\right)}}{2\,\sqrt{\pi}\,s}$$

```
#Représentation graphique, cf "sagemath graphique", plot?
m=1;s=1
plot(g(m,s,x),x,-4,6)+line([(m,0),(m,g(m,s,m))],linestyle='--')+text("m",(m,-0.015))
```




```
c = ['blue','green','red','cyan','magenta','yellow','black']
m=[-1,0,1,1];s=[1,1,1/2,2]
p=sum(plot(g(m[k-1],s[k-1],x),x,-5,5, color=c[k-1],legend_label='(m,s)='+str((m[k-1],s[k-1]))) for k in (1..4))
show(p)
```




```
#Recherche d'un quantile pour s=1
#Méthode formelle avec la fonction de répartition
from scipy.special import erfinv as inverse_erf
F(t)=integrate(g(0,1,x),x,-infinity,t)
solq=solve(F(x)==1.95/2,x);solq
x.subs(solq) #la définition scipy de inverse_erf n'affecte pas la méthode solve
#x.subs(solq).n() produit une erreur ; on recopie à la main le texte de la solution ; peut on le faire par une commande à partir de str(solq[0])
x.subs(x == sqrt(2)*inverse_erf(19/20)).n(digits=3)
```

```
[x == sqrt(2)*inverse_erf(19/20)]
sqrt(2)*inverse_erf(19/20)
1.96
```

```
#test
%python
import platform
platform.python_version()
1/2 #le résultat est un entier dans python 2.* ! cf "python division"
1./2
2^3 #Cf la section 9.9.1 de https://docs.python.org/2/library/operator.html
2**3
```

```
'2.7.13'
0
0.5
1
8
```

```
%python3
import platform
print(platform.python_version())
print(1/2)
```

```
3.5.3
0.5
```

```
#test dans sage
from scipy.special import erfinv as inverse_erf
eval('1/2')
eval('1./2')
sage_eval('1/2')
#sage_eval('inverse_erf(1/2)') produit une erreur : inverse_erf pas reconnu
eval('inverse_erf(1/2)')
eval('inverse_erf(1./2)')
```

```
0
0.5
1/2
0.0
0.47693627620446982
```

```
eval(str(solq[0].rhs()).replace('/','*1.0/')).n()
```

```
1.95996398454005
```

```
#Algorithme de dichotomie
def dicho(h,a,b,prec):
    if h(a)*h(b)>0:
        print('peut être pas de solution')
    else:
        while b-a>prec:
            i=(a+b)/2
            if h(i)*h(b)>0:b=i
            else:a=i
    return(a)

dicho(F-0.975,0,4,0.001).n() #seules les 3 premières décimales après la virgule sont pertinentes !
```

```
1.95996093750000
```

Compléments : variables, expressions, fonctions, affectations dans Python

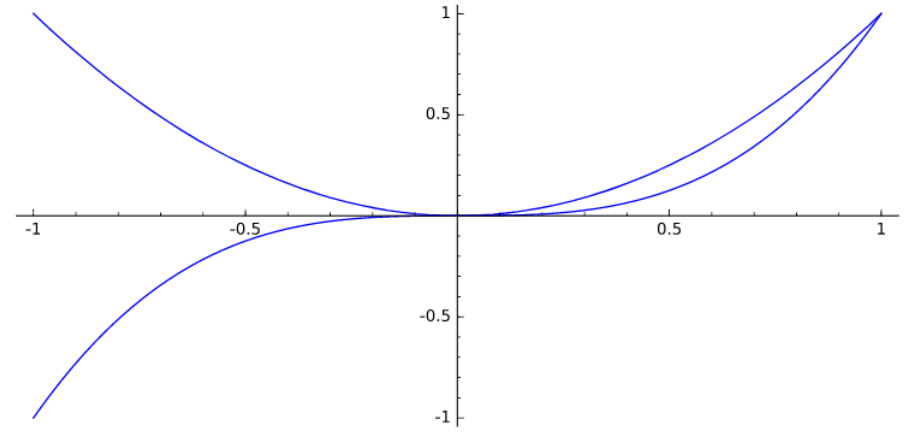
```
#Test
reset()
f=x;f #x est prédéclaré ; f=y produit une erreur si y n'est pas déclaré au préalable
x=1;f #pas rétroactif
f=x;f #x est remplacé par sa valeur
f(x)=x;f;x#x est réinitialisé !
print
f=x;f
f.parent()
f.variables()
f(x=1) #substitution de x par 1 dans l'expression f
show_identifiers() #x est omis par convention
print
g=f;g
x=1
diff(f,x)
x=var('x');diff(f)#avec x=1 produit une erreur
integrate(f)
var('y');integrate(f(x=y),y)#avec y=1 produit une erreur
print
g(x)=f;print g,',', g.parent(),',',g.variables()
#g(y)=f.subs(x=y);g
g.diff()
diff(g(y),y)
integrate(g)
integrate(g(y),y)
```

```
x
x
1
x |--> x
x

x
Symbolic Ring
(x,)
1
['f']

x
1
1
1/2*x^2
y
1/2*y^2

x |--> x , Callable function ring with argument x , (x,)
x |--> 1
1
x |--> 1/2*x^2
1/2*y^2
```

```
reset()
h=lambda x:x^2;h;h(2);h(x);diff(h(x),x);integrate(h(x),x)
#mais diff(h) produit une erreur, de même que h.variables()
```

```
print
def k(x):return x^3
k;k(x);diff(k(x),x)
show_identifiers()
plot(h)+plot(k)
```

```
 at 0x7f10bb28db18>
4
x^2
2*x
1/3*x^3


x^3
3*x^2
['k', 'h']
```




```
reset()
#y n'est pas défini, une expression contenant y produit une erreur, mais pas une affectation
g(y)=x;g;g(1) #déclare g et y
show_identifiers()
g.parent()
```

```
y |--> x
x
['g', 'y']
Callable function ring with argument y
```

```
#variables locales dans def, affectation globale
reset()
x=2
def g(y):
    z=y
    y=1
    x=1
    return(z)
show_identifiers()
g(x)
x
```

```
['g', 'x']
2
2
```

```
str(g(y))
```

```
'x'
```

```
import ast
formula = 'integrate(y+a^2,a,0,1)'
names = [
    node.id for node in ast.walk(ast.parse(formula))
    if isinstance(node, ast.Name)
]
names
```

```
['integrate', 'a', 'y', 'a']
```

```
del ast
```

generated 2017-05-06T14:30:43 on SageMathCloud

18nov19

L2 MMAG - Réponses à l'exercice 4 de la série 3 avec Sagemath

Sagemath (http://www.sagemath.org/) est un logiciel de calcul formel qui permet entre autre :

- de faire des calculs algébriques avec les vecteurs (on peut par exemple mettre en oeuvre l'algorithme de Gram-Schmidt pas à pas)
- d'expliciter une base de l'espace des solutions d'un système d'équations linéaires.

On peut exécuter un code Sagemath (une suite d'instructions) sur le site SageMathCell (https://sagecell.sagemath.org/) ou bien en ouvrant un compte et en créant une feuille de calcul sur Cocalc (https://cocalc.com/).

Exercice Soient $v_1 = (2,1,2,-4)$, $v_2 = (1,2,-2,4)$, $v_3 = (4,2,1,2)$ et $V = \mathrm{vect}(v_1,v_2,v_3) \subset \mathbb{R}^4$. Déterminer $V^\perp$, une base orthonormée de $V^\perp$ et une base orthonormée de V.

In [2]:
```
v1=vector([2,1,2,-4])
v2=vector([1,2,-2,4])
v3=vector([4,2,1,2])
```

1ère méthode : $V^\perp$ est donné par le système d'équations linéaires $< v_i, (x,y,z,t) >= 0, i = 1,2,3$. La matrice de ce système est la matrice dont les lignes sont les coordonnées des v_i.

- `M=matrix(QQ,[v1,v2,v3])` définit M comme cette matrice à coefficients dans $\mathbb{Q}$,
- `K=M.right_kernel()` définit K comme $\ker(M)$, l'espace des solutions du système $MX = 0$,
- `g=K.basis()` définit g comme la liste des vecteurs d'une base de K déterminée par Sagemath
- `g1=g[0];print g1*g1` définit g_1 comme le premier élément de cette liste et calcul $||g_1||^2 =< g_1, g_1 >$.

In [26]:
```
Vorthog=matrix(QQ,[v1,v2,v3]).right_kernel()
print Vorthog
g=Vorthog.basis()[0]
print g, g*g
```

```
Vector space of degree 4 and dimension 1 over Rational Field
Basis matrix:
[   1   -1 -5/4 -3/8]
(1, -1, -5/4, -3/8) 237/64
```

$V^\perp$ est de dimension 1 donc un seul vecteur non nul de $V^\perp$ donne une base orthogonale de $V^\perp$. Comme $V^\perp$ est de dimension 1, V est de dimension 3 et (v_1,v_2,v_3) en est une base.

Pour calculer une base orthogonale de V on part d'une base de V, ici (v_1,v_2,v_3) ; on pose $f_1 = v_1$; on cherche les combinaisons linéaires de v_1,v_2,v_3 qui sont orthogonales à f_1 en résolvant l'équation $< f_1, xv_1 + yv_2 + zv_3 >= 0$; on choisit pour f_2 une solution non triviale ; on cherche les combinaisons linéaires de v_1,v_2,v_3 qui sont orthogonales à f_1, f_2 en résolvant le système $< f_i, xv_1 + yv_2 + zv_3 >= 0, i = 1,2$; on choisit pour f_3 une solution non triviale. La matrice d'un tel système est le produit de la matrice dont les lignes sont les coordonnées des f_i avec la matrice dont les colonnes sont les coordonnées de v_1,v_2,v_3.

In [14]:
```
f=[v1]
V1=(matrix(QQ,v1)*(matrix(QQ,[v1,v2,v3]).transpose())).right_kernel()
f=f+[V1.basis()[0]]
V2=(matrix(QQ,[v1,v2])*(matrix(QQ,[v1,v2,v3]).transpose())).right_kernel()
f=f+[V2.basis()[0]]
show(f)
show([f[i]*f[i] for i in range(len(f))])
#latex(matrix(QQ,[v1,v2,v3]))
```

Out[14]:
$$\left[(2,\,1,\,2,\,-4),\,\left(1,\,0,\,-\frac{25}{4}\right),\,\left(1,\,\frac{23}{18},\,-\frac{41}{36}\right)\right]$$

Out[14]:
$$\left[25,\,\frac{641}{16},\,\frac{5093}{1296}\right]$$

2ème méthode : Orthogonalisation de Gram-Schmidt. On commence par orthogonaliser la famille (v_1, v_2, v_3) en posant

$f_1 = v_1,$

$f_2 = v_2 - \mathrm{proj}_{<f_1>}(v_2) = v_2 - \frac{<f_1,v_2>}{<f_1,f_1>} f_1,$

$f_3 = v_3 - \mathrm{proj}_{<f_1,f_2>}(v_3) = v_3 - \left(\frac{<f_1,v_3>}{<f_1,f_1>} f_1 + \frac{<f_2,v_3>}{<f_2,f_2>} f_2\right).$

In [28]:
```
f1=v1
print 'f1=',f1,',','|f1|^2=',f1*f1
f2=v2-f1*v2/(f1*f1)*f1
print 'f2=',f2,',','|f2|^2=',f2*f2
f3=v3-f1*v3/(f1*f1)*f1-f2*v3/(f2*f2)*f2
print 'f3=',f3,',','|f3|^2=',f3*f3
```

```
f1= (2, 1, 2, -4) , |f1|^2= 25
f2= (57/25, 66/25, -18/25, 36/25) , |f2|^2= 369/25
f3= (46/41, -46/41, 61/41, 42/41) , |f3|^2= 237/41
```

f_1, f_2, f_3 sont orthogonaux, non nuls et engendrent $V =< v_1, v_2, v_3 >$ donc V est de dimension 3 et $V^\perp$ est de dimension 1. On a $V^\perp =< g >$ où g complète f_1, f_2, f_3 en une base orthogonale de $\mathbb{R}^4$.

Complétion en une base orthogonale de $\mathbb{R}^4$: On cherche v qui ne soit pas dans V et alors $g = v - \mathrm{proj}_{<v_1,v_2,v_3>}(v)$ est non nul et orthogonal à v_1, v_2, v_3. On essaye successivement pour v les éléments de la base canonique de $\mathbb{R}^4$; l'un d'eux convient puisque $\mathbb{R}^4$ n'est pas inclus dans V.

In [30]:
```
v=vector([1,0,0,0]);g=v-f1*v/(f1*f1)*f1-f2*v/(f2*f2)*f2-f3*v/(f3*f3)*f3
print 'g=',g,',','|g|^2=',g*g
```

```
g= (64/237, -64/237, -80/237, -8/79) , |g|^2= 64/237
```

Commentaires : L'orthogonalisation de Gram-Schmidt fait intervenir moins d'opérations mais des numérateurs et dénominateurs très vite plus grands lors des calculs.

Algèbre linéaire avec Sagemath

F-X. Dehon dehon@unice.fr, 21 mars 2020

1. Vecteurs de $\mathbb{R}^3$, listes de tels vecteurs, liste de vecteurs dont on donne la matrice des coordonnées (relativement à la base canonique), liste des vecteurs de la base canonique.

Matrice des coordonnées d'une liste de vecteurs (relativement à la base canonique).

In [70]:
```
u=vector([1,0,2]);print 'u=',u
v=vector([-2,1,1]);print '[u,v]=',[u,v]
M=matrix([[1,1,2],[0,2,3]]).transpose();show('M=',M,LatexExpr(r"\rightsquigarrow"),M.columns())#M.transpose() est la matrice dont les colonnes sont les lignes de M
B=identity_matrix(3).columns();print 'B=',B

A=matrix([u,v]).transpose();show('Mat(u,v)=',A)
```

```
u= (1, 0, 2)
[u,v]= [(1, 0, 2), (-2, 1, 1)]
```

Out[70]:

$$\mathtt{M=}\begin{pmatrix}1 & 0\\ 1 & 2\\ 2 & 3\end{pmatrix} \rightsquigarrow [(1,\,1,\,2)\,,(0,\,2,\,3)]$$

```
B= [(1, 0, 0), (0, 1, 0), (0, 0, 1)]
```

Out[70]:

$$\mathtt{Mat(u,v)=}\begin{pmatrix}1 & -2\\ 0 & 1\\ 2 & 1\end{pmatrix}$$

2. Un avertissement (un peu technique) sur le type des objets dans Sagemath :

In [87]:
```
#Attention au type et classe des objets dans Python ou Sagemath. Une liste ou un tuple peuvent être convertis en vecteur et vice versa. On peut multiplier une matrice par un vecteur, pas par un tuple.
#Cf http://doc.sagemath.org/html/en/thematic_tutorials/tutorial-programming-python.html

u=vector([1,0,2]);v=(1,0,2)
print tuple(u),vector(v)#conversions, pas de différence dans l'affichage
print 'type(u):',type(u),'\nu.parent():',u.parent(),'\ntype(v):',type(v)
print 'v.parent():',v.parent()#produit une erreur
```

```
 (1, 0, 2) (1, 0, 2)
type(u): <type 'sage.modules.vector_integer_dense.Vector_integer_dense'>
u.parent(): Ambient free module of rank 3 over the principal ideal domain Integer Ring
type(v): <type 'tuple'>
v.parent():

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-87-ebb42e122409> in <module>()
      5 print tuple(u),vector(v)#conversions, pas de différence dans l'affichage
      6 print 'type(u):',type(u),'\nu.parent():',u.parent(),'\ntype(v):',type(v)
----> 7 print 'v.parent():',v.parent()#produit une erreur

AttributeError: 'tuple' object has no attribute 'parent'
```

3. Changement de base : formule matricielle

Rappel de cours : Soient $(f_1,\ldots,f_r)$ une famille de vecteurs de $\mathbb{R}^n$, P la matrice des coordonnées des f_i dans la base canonique. Si $(x_1,\ldots,x_r)$ est une famille de scalaires, la combinaison linéaire $x_1\,f_1+\ldots+x_r\,f_r$ à pour coordonnées dans la base canonique

$$[\sum_i x_i\,f_i] = P\begin{pmatrix}x_1\\ \vdots\\ x_r\end{pmatrix}$$

La famille $(f_1,\ldots,f_r)$ est une base si et seulement si la matrice P est inversible, ce qu'on peut établir en calculant son déterminant (pourvu que P soit une matrice carré) avec l'instruction `det(P)` ou en échelonnant P (voir plus loin "opérations sur les colonnes d'une matrice").

Supposons que $(f_1,\ldots,f_n)$ est une base de $\mathbb{R}^n$ et notons $[u]_{(f_i)}$ les coordonnées d'un vecteur $u\in\mathbb{R}^n$ dans cette base, $[u]$ les coordonnées de u dans la base canonique. On a les relations :

$$\begin{aligned}[u] &= P\,[u]_{(f_i)}\\ [u]_{(f_i)} &= P^{-1}\,[u]\end{aligned}$$

Exemple :

In [29]:
```
f1,f2,f3=((1,0,1),(1,1,2),(1,2,1));show(f1,f2,f3)
P=matrix([f1,f2,f3]).transpose();show('P =',P)
show('det(P) =',det(P))
X=P^(-1)*vector([1,0,0]);show('X:=[(1,0,0)]_(f_i) =',X)
show(html("Vérification :"));show("PX =",P*X)
```

Out[29]: $(1, 0, 1)\ (1, 1, 2)\ (1, 2, 1)$

Out[29]: $P = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

Out[29]: $\det(P) = -2$

Out[29]: $X:=[(1,0,0)]\_(f\_i) = \left(\frac{3}{2}, -1, \frac{1}{2}\right)$

Out[29]: Vérification :

Out[29]: $PX = (1, 0, 0)$

4. Application $\mathbb{R}^3 \to \mathbb{R}^2$ donnée par une expression, image d'un vecteur par une telle application, matrice d'une telle application relativement aux bases canoniques.

Expression d'une application linéaire $\mathbb{R}^3 \to \mathbb{R}^2$ donnée par sa matrice (relativement aux bases canoniques).

Matrice d'une application linéaire et changement de bases.

In [83]:
```
f(x,y,z)=(2*x+y+z,x-y-z);print 'f:',f;show('f:',f)
u=vector([1,0,2]);print 'u=',u,', f(*u)=',f(*u)# Attention à la syntaxe *u au lieu de u
B=identity_matrix(3).columns();M=matrix([f(*v) for v in B]).transpose();show('Mat(f)=',M)
N=matrix([[1,1,2],[2,0,3]])
g(x,y,z)=tuple(N*vector([x,y,z]));show('N=',N,LatexExpr(r"\rightsquigarrow"),'g:',g)#tuple() convertit un vecteur en up
let, nécessaire pour la définition de g dans Sagemath

f1,f2,f3=((1,0,1),(1,1,2),(1,2,1))
Q=matrix([f1,f2,f3]).transpose()
show('(f1,f2,f3):= ',(f1,f2,f3), "nouvelle base de ", LatexExpr(r"\mathbb{R}^3"),", Q:=Mat(f1,f2,f3) = ",Q)
u,v=((1,1),(-1,1))
P=matrix([u,v]).transpose()
show('(u,v):= ',(u,v), "nouvelle base de ", LatexExpr(r"\mathbb{R}^2"),", P:=Mat(u,v) = ",P)
N=P^(-1)*M*Q;show('Mat(f,(u,v),(f1,f2,f3)) = ',LatexExpr(r"P^{-1}"),"Mat(f)Q = ",N)
show(html("Une vérification :"));show('f(f1) = ',f(*f1),', '+str(N[0,0])+' u + '+str(N[1,0])+' v = ',N[0,0]*vector(u)+N
[1,0]*vector(v))
```

```
f: (x, y, z) |--> (2*x + y + z, x - y - z)
```

Out[83]: $f: (x, y, z) \mapsto (2\,x + y + z,\ x - y - z)$

```
u= (1, 0, 2) , f(*u)= (4, -1)
```

Out[83]: $Mat(f)= \begin{pmatrix} 2 & 1 & 1 \\ 1 & -1 & -1 \end{pmatrix}$

Out[83]: $N= \begin{pmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \end{pmatrix} \rightsquigarrow g: (x, y, z) \mapsto (x + y + 2\,z,\ 2\,x + 3\,z)$

Out[83]: $(f1,f2,f3):= ((1, 0, 1), (1, 1, 2), (1, 2, 1))\ nouvelle\ base\ de \mathbb{R}^3,\ Q:=Mat(f1,f2,f3) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

Out[83]: $(u,v):= ((1, 1), (-1, 1))\ nouvelle\ base\ de \mathbb{R}^2,\ P:=Mat(u,v) = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$

Out[83]: $Mat(f,(u,v),(f1,f2,f3)) = P^{-1}Mat(f)Q = \begin{pmatrix} \frac{3}{2} & \frac{3}{2} & \frac{3}{2} \\ -\frac{3}{2} & -\frac{7}{2} & -\frac{7}{2} \end{pmatrix}$

Out[83]: Une vérification :

Out[83]: $f(f1) = (3, 0),\ 3/2\ u + -3/2\ v = (3, 0)$

5. Opérations sur les colonnes d'une matrice, méthodes disponibles (voir https://wiki.sagemath.org/quickref?action=AttachFile&do=view&target=quickref-linalg.pdf (https://wiki.sagemath.org/quickref?action=AttachFile&do=view&target=quickref-linalg.pdf))

Ajout de a fois la colonne j de A à la colonne i : `A.add_multiple_of_column(i,j,a)` . Attention la première colonne est d'indice 0, tout comme la première ligne.

Echange des colonnes i et j : `A.swap_cols(i,j)`

Pour obtenir le résultat de ces opérations sans changer la matrice d'origine A : `A.with_added_multiple_of_column(i,j,a)` , `A.with_swapped_columns(i,j)`

Introduction de paramètres : il faut les déclarées convenablement pour les insérer dans une matrice.

Un exemple :

In [33]:
```
x1,x2,x3=ZZ['x1,x2,x3'].gens()
N=matrix([[1/3,1,2],[2,0,3]]).transpose()
M=block_matrix([[N,matrix([x1,x2,x3]).transpose()]]);show(M)
M.add_multiple_of_column(1,0,-6);show(M)
M.add_multiple_of_column(2,0,-3*x1);show(M)
M.add_multiple_of_column(2,1,(-3*x1+x2)/6);show(M)
```

Out[33]:
$$\left(\begin{array}{cc|c} \frac{1}{3} & 2 & x_1 \\ 1 & 0 & x_2 \\ 2 & 3 & x_3 \end{array}\right)$$

Out[33]:
$$\left(\begin{array}{cc|c} \frac{1}{3} & 0 & x_1 \\ 1 & -6 & x_2 \\ 2 & -9 & x_3 \end{array}\right)$$

Out[33]:
$$\left(\begin{array}{cc|c} \frac{1}{3} & 0 & 0 \\ 1 & -6 & -3x_1 + x_2 \\ 2 & -9 & -6x_1 + x_3 \end{array}\right)$$

Out[33]:
$$\left(\begin{array}{cc|c} \frac{1}{3} & 0 & 0 \\ 1 & -6 & 0 \\ 2 & -9 & -\frac{3}{2}x_1 - \frac{3}{2}x_2 + x_3 \end{array}\right)$$

L1MF2 - Une réponse à l'ex3 de la feuille 3 avec Sagemath

F-X. Dehon dehon@unice.fr, 25 mars 2020

Exercice 3 – Nous considèrons l'application linéaire $f : \mathbf{R}^4 \to \mathbf{R}^3$ définie par :

$$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3) = (x_1 + x_2 + 4x_3 + x_4, x_1 - 2x_2 - 2x_3 - x_4, 2x_1 + x_2 + 6x_3 - 2x_4)$$

Soit $\mathcal{B}_3$ la base canonique de $\mathbf{R}^3$ et $\mathcal{B}_4$ celle de $\mathbf{R}^4$.
1) Préciser l'image des vecteurs de la base $\mathcal{B}_4$. Quelle est la matrice f relativement à ces bases canoniques ?
2) Soit $u = (1, 2, 2, 1)$ et $D = \text{Vect}\,(u)$, Déterminer une base du sous-espace vectoriel $f(D)$.
3) Soit $u_1 = (1, 1, 0, 0)$, $u_2 = (0, 0, 1, 0)$, $u_3 = (1, 0, 0, 1)$ et $F = \text{Vect}\,(u_1, u_2, u_3)$. Déterminer une base du sous-espace vectoriel $f(F)$.
4) Déterminer une base de l'image de f.
5) Déterminer $\ker f$.
6) Soit G le sous-espace vectoriel de $\mathbf{R}^3$ d'équation $y_1 + y_2 + y_3 = 0$. Déterminer une base du sous-espace vectoriel $f^{-1}(G)$.
7) Notons e_1, e_2, e_3, e_4 les vecteurs de la base $\mathcal{B}_4$. Montrer que $(f(e_1), f(e_2), f(e_4))$ est une base, notée $\mathcal{B}''$, de $\mathbf{R}^3$, puis montrer que $(e_1, e_2, e_4, 2e_1 + 2e_2 - e_3)$ est une base, notée $\mathcal{B}'$, de $\mathbf{R}^4$. Déterminer la matrice de f avec comme base de départ $\mathcal{B}'$ et comme base d'arrivée $\mathcal{B}''$, c'est à dire la matrice $\mathcal{M}(f, \mathcal{B}'', \mathcal{B}')$.

On s'inspire du document "Algèbre linéaire avec Sagemath" (http://jalon.unice.fr/Members/dehon/Fichiers/alglin-21mar20.pdf/view?course_id=Cours-dehon-20200306092058) du 21 mars 2020.

Réponse : 1)

In [1]:
```
f(x1,x2,x3,x4)=(x1+x2+4*x3+x4,x1-2*x2-2*x3-x4,2*x1+x2+6*x3-2*x4);print 'f:',f;show('f:',f)
B=identity_matrix(4).columns()
for u in B:print "f"+str(u),"=",f(*u)

M=matrix([f(*v) for v in B]).transpose();show('Mat(f)=',M)
```

```
f: (x1, x2, x3, x4) |--> (x1 + x2 + 4*x3 + x4, x1 - 2*x2 - 2*x3 - x4, 2*x1 + x2 + 6*x3 - 2*x4)
```

Out[1]: $\mathtt{f}: (x_1, x_2, x_3, x_4) \mapsto (x_1 + x_2 + 4\,x_3 + x_4,\ x_1 - 2\,x_2 - 2\,x_3 - x_4,\ 2\,x_1 + x_2 + 6\,x_3 - 2\,x_4)$

```
f(1, 0, 0, 0) = (1, 1, 2)
f(0, 1, 0, 0) = (1, -2, 1)
f(0, 0, 1, 0) = (4, -2, 6)
f(0, 0, 0, 1) = (1, -1, -2)
```

Out[1]: $\mathtt{Mat(f)}= \begin{pmatrix} 1 & 1 & 4 & 1 \\ 1 & -2 & -2 & -1 \\ 2 & 1 & 6 & -2 \end{pmatrix}$

2,3,4)

In [80]:
```
u=(1,2,2,1);print 'u='+str(u)+', f'+str(u),'=',f(*u)
show(html("(f(u)) &ne; (0) est une base de f(Vect(u))=Vect(f(u))"))
u1=vector([1,1,0,0]);u2=vector([0,0,1,0]);u3=vector([1,0,0,1])
N=matrix([f(*u1),f(*u2),f(*u3)]).transpose();show('Mat(u1,u2,u3)=',N)
show(html("début échelonnage avec 1ère col : C2 - 2C1 &rarr; C2, C3 - C1 &rarr; C3"))
N.add_multiple_of_column(1,0,-2)
N.add_multiple_of_column(2,0,-1)
show(N)
show(html("échange C2 &harr; C3"))
N.swap_columns(1,2);show(N)
```

```
u=(1, 2, 2, 1), f(1, 2, 2, 1) = (12, -8, 14)
```

Out[80]: (f(u)) ≠ (0) est une base de f(Vect(u))=Vect(f(u))

Out[80]: $\mathtt{Mat(u1,u2,u3)}= \begin{pmatrix} 2 & 4 & 2 \\ -1 & -2 & 0 \\ 3 & 6 & 0 \end{pmatrix}$

Out[80]: début échelonnage avec 1ère col : C2 - 2C1 → C2, C3 - C1 → C3

Out[80]: $\begin{pmatrix} 2 & 0 & 0 \\ -1 & 0 & 1 \\ 3 & 0 & -3 \end{pmatrix}$

Out[80]: échange C2 ↔ C3

Out[80]: $\begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -3 & 0 \end{pmatrix}$

Conclusion : $f(\mathrm{Vect}(u_1,u_2,u_3)) = \mathrm{Vect}(f(u_1),f(u_2),f(u_3))$ admet pour base les vecteurs de coordonnées les deux premières colonnes de la matrice échelonnée, c'est à dire $f(u_1)$ et $f(u_3)-f(u_1)$.

De même $\mathrm{Im}(f)$ admet pour base les vecteurs dont les coordonnées sont les colonnes non nulles de la forme échelonnée de la matrice de f.

In [44]:
```
show('Mat(f)=',M)
MM=matrix(M)#on travail sur une copie de M
show(html("début échelonnage avec 1ère col : C2 - C1 &rarr; C2, C3 - 4C1 &rarr; C3, C4 - C1 &rarr; C4"))
MM.add_multiple_of_column(1,0,-1)
MM.add_multiple_of_column(2,0,-4)
MM.add_multiple_of_column(3,0,-1)
show(MM)
show(html("suite avec la 2ème col : C3 - 2C2 &rarr; C3, C4 - 2/3C2 &rarr; C4"))
MM.add_multiple_of_column(2,1,-2)
MM.add_multiple_of_column(3,1,-2/3)
show(MM)
show(html("échange C3 &harr; C4"))
MM.swap_columns(2,3);show(MM)
```

Out[44]:
$$\mathtt{Mat(f)=}\begin{pmatrix}1 & 1 & 4 & 1\\ 1 & -2 & -2 & -1\\ 2 & 1 & 6 & -2\end{pmatrix}$$

Out[44]: début échelonnage avec 1ère col : C2 - C1 → C2, C3 - 4C1 → C3, C4 - C1 → C4

Out[44]:
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 1 & -3 & -6 & -2\\ 2 & -1 & -2 & -4\end{pmatrix}$$

Out[44]: suite avec la 2ème col : C3 - 2C2 → C3, C4 - 2/3C2 → C4

Out[44]:
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 1 & -3 & 0 & 0\\ 2 & -1 & 0 & -\frac{10}{3}\end{pmatrix}$$

Out[44]: échange C3 ↔ C4

Out[44]:
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 1 & -3 & 0 & 0\\ 2 & -1 & -\frac{10}{3} & 0\end{pmatrix}$$

Conclusion : $\mathrm{Im}(f)$ admet pour base $\left(f(e_1), f(e_2)-f(e_1), f(e_4)-f(e_1)-\frac{2}{3}(f(e_2)-f(e_1))\right)$ où $(e_1,\dots,e_4)$ est la base canonique de $\mathbb{R}^4$.

5) $\mathrm{Ker}(f) = \{u\in\mathbb{R}^4,\ f(u)=0\}$ se lit aussi avec la forme échelonnée de la matrice de f : il correspond à la 4ème colonne de la matrice échelonnée donc est engendré par $e_3-4e_1-2(e_2-e1) = e_3-2e_1-2e_2 = (-2,-2,1,0)$. Vérification :

In [47]:
```
print f(-2,-2,1,0)
```

```
(0, 0, 0)
```

et on sait que $\dim(\mathrm{Ker}(f)) = \dim(\mathbb{R}^4) - \dim(\mathrm{Im}(f)) = 4-3 = 1$.

6) $G = \{(y_1,y_2,y_3)\in\mathbb{R}^3,\ y_1+y_2+y_3=0\}$ alors

$$\begin{aligned} f^{-1}(G) &= \{(x_1,x_2,x_3,x_4)\in\mathbb{R}^4,\ f(x_1,x_2,x_3,x_4)\in G\}\\ &= \{(x_1,x_2,x_3,x_4)\in\mathbb{R}^4,\ (x_1+x_2+4\,x_3+x_4)+(x_1-2\,x_2-2\,x_3-x_4)+(2\,x_1+x_2+6\,x_3-2\,x_4)=0\}\\ &= \{(x_1,x_2,x_3,x_4)\in\mathbb{R}^4,\ 4\,x_1+8\,x_3-2\,x_4=0\}\end{aligned}$$

In [59]:
```
print f(x1,x2,x3,x4);print sum(f(x1,x2,x3,x4))
#print latex(f(x1,x2,x3,x4))
```

```
(x1 + x2 + 4*x3 + x4, x1 - 2*x2 - 2*x3 - x4, 2*x1 + x2 + 6*x3 - 2*x4)
4*x1 + 8*x3 - 2*x4
```

On obtient un sytème réduit à une équation linéaire homogène d'inconnues x_1,x_2,x_3,x_4. Variable de tête x_1 et variables libres x_2,x_3,x_4 conduisent au paramétrage linéaire bijectif des solutions

$$\{(-2\,x_3+\tfrac{1}{2}\,x_4, x_2, x_3, x_4),\ x_2,x_3,x_4\in\mathbb{R}\} = \mathrm{Vect}\big((0,1,0,0),(-2,0,1,0),(\tfrac{1}{2},0,0,1)\big)$$

7) On observe que la famille $(f(e_1), f(e_2), f(e_4))$ engendre les trois vecteurs de la base de $\mathrm{Im}(f)$ donnés en (4) donc est génératrice de $\mathrm{Im}(f)$ et comme sa taille est la dimension de $\mathrm{Im}(f)$ c'en est une base.

En comparant les dimensions on observe l'égalité $\mathrm{Im}(f) = \mathbb{R}^3$; donc $(f(e_1), f(e_2), f(e_4))$ est une base de $\mathbb{R}^3$.

Rq : on peut aussi échelonner la matrice des coordonnées de $(f(e_1), f(e_2), f(e_4))$ pour conclure.

$(e_1, e_2, e_4, 2\,e_1 + 2\,e_2 - e_3)$ engendre e_1, e_2, e_3, e_4 donc $\mathbb{R}^4$ entier et comme sa taille est $4 = \dim(\mathbb{R}^4)$ c'est une base.

On reconnaît dans l'expression $2\,e_1 + 2\,e_2 - e_3$ l'opposé du générateur de $\mathrm{Ker}(f)$ donné en (5). On écrit :

$f(e_1) = 1\,f(e_1) + 0\,f(e_2) + 0\,f(e_4)$, $f(e_2) = 0\,f(e_1) + 1\,f(e_2) + 0\,f(e_4)$, $f(e_4) = 0\,f(e_1) + 0\,f(e_2) + 1\,f(e_4)$,
$f(2\,e_1 + 2\,e_2 - e_3) = 0\,f(e_1) + 0\,f(e_2) + 0\,f(e_4)$

d'où la matrice de f dans les bases $(e_1, e_2, e_4, 2\,e_1 + 2\,e_2 - e_3)$ et $(f(e_1), f(e_2), f(e_4))$:

In [81]:
```
show(identity_matrix(3).augment(matrix([0,0,0]).transpose()))
```

Out[81]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Rq : lien avec les matrices de passages :

$P = \mathrm{Mat}(f(e_1), f(e_2), f(e_4))$, $Q = \mathrm{Mat}(e_1, e_2, e_4, 2\,e_1 + 2\,e_2 - e_3)$ relativement aux bases canoniques. On a (avec les conventions du cours pour la notation $\mathrm{Mat}(f, \mathcal{B}'', \mathcal{B}')$) :

$$\mathrm{Mat}(f, (f(e_1), f(e_2), f(e_4)), (e_1, e_2, e_4, 2e_1 + 2e_2 - e_3)) = P^{-1}\mathrm{Mat}(f)Q$$

Vérification :

In [2]:
```
e1,e2,e3,e4=B#base canonique de R^4
P=matrix([f(*e1),f(*e2),f(*e4)]).transpose()
Q=matrix([e1,e2,e4,2*e1+2*e2-e3]).transpose()
show('P=',P,', Q=',Q,', Mat(f)=',M)
show('P^(-1)MQ=',P^(-1)*M*Q)
```

Out[2]:
$$\texttt{P=}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 2 & 1 & -2 \end{pmatrix}\texttt{, Q=}\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\texttt{, Mat(f)=}\begin{pmatrix} 1 & 1 & 4 & 1 \\ 1 & -2 & -2 & -1 \\ 2 & 1 & 6 & -2 \end{pmatrix}$$

Out[2]:
$$\texttt{P^(-1)MQ=}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

TD L3 Algèbre et géométrie - Feuille 2 ex. 3 avec Sagemath

F-X. Dehon - 3 oct. 2020 - dehon[@]unice.fr

Enoncé :

> **3.** Soit $E=\mathbb{R}_2[X]$ et $\langle P,Q\rangle=\int_{-1}^{+1}P(t)Q(t)dt$.
>
> **3.a.** Montrer que $(E,\langle -,-\rangle)$ est un espace euclidien de dimension 3.
> **3.b.** En appliquant le procédé de Gram-Schmidt à $(1,X,X^2)$ trouver une base orthonormée (L_0,L_1,L_2) de $(E,\langle -,-\rangle)$.
> **3.c.** Montrer que $P\mapsto u(P)=(1-X^2)P'(X)+(2X+1)P(X)$ définit un endomorphisme $u:E\to E$. Ecrire u dans la base canonique $(1,X,X^2)$.
> **3.d.** Ecrire u dans la base (L_0,L_1,L_2). En déduire la valeur de l'intégrale $\int_{-1}^{1}(u(P))^2dt$ en fonction des coefficients (a,b,c) de $P(X)=aL_0+bL_1+cL_2$.

Les éléments de $E=\mathbb{R}_2[X]$ dans l'énoncé sont ici déclarés comme des fonctions de la variable x sans précision sur son type.

In [1]:
```
a=1/sqrt(2);
print a.parent(),",",a.category()
print a,AA(a),RR(a)
```

```
Symbolic Ring , Category of elements of Symbolic Ring
1/2*sqrt(2) 0.7071067811865475? 0.707106781186548
```

Plusieurs façons de déclarer une fonction, pas strictement équivalentes :

In [2]:
```
a=1/sqrt(2)
P0(x)=x+a
Q0=lambda x:x+a
def R0(x):return(x+a)
print P0,",",Q0,",",R0
print P0==Q0,bool(P0(x)==Q0(x))
print P0==R0,bool(P0(x)==R0(x))

print P0.derivative()
try:print Q0.derivative()
except Exception as e: print "Q0.derivative() :",e
print Q0(x).derivative(x)
```

```
x |--> x + 1/2*sqrt(2) , <function <lambda> at 0x7fee6856d5f0> , <function R0 at 0x7fee6
856d668>
False True
False True
x |--> 1
Q0.derivative() : 'function' object has no attribute 'derivative'
1
```

Définition de la forme $\langle -,-\rangle$ nommée B ci-dessous, de l'application u, coordonnées et matrices

In [3]:
```
def B(P,Q):return(integrate(P(x)*Q(x),x,-1,1))
```

In [4]:
```
print B(lambda x:x, lambda x:x^3)
```

```
2/5
```

In [5]:
```
A=matrix(3,3,lambda i,j:B(lambda x:x^i,lambda x:x^j));pretty_print(A)
```

Out[5]:
$$\begin{pmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{pmatrix}$$

In [6]:
```
var('a b c')
X=matrix([a,b,c]).transpose();pretty_print(X)
print (X.transpose()*A*X)[0,0].expand()
```

Out[6]:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

```
2*a^2 + 2/3*b^2 + 4/3*a*c + 2/5*c^2
```

In [7]:
```
P(x)=a+b*x+c*x^2
print B(P,P).expand()
```

```
2*a^2 + 2/3*b^2 + 4/3*a*c + 2/5*c^2
```

In [8]:
```
def u(P):return((1-x^2)*(P.derivative(x))+(2*x+1)*P(x))
```

In [9]:
```
print u(P)(x).expand()
```

```
b*x^2 + c*x^2 + 2*a*x + b*x + 2*c*x + a + b
```

In [10]:
```
print B(u(P),u(P)).expand()
```

```
14/3*a^2 + 8*a*b + 22/5*b^2 + 20/3*a*c + 24/5*b*c + 46/15*c^2
```

In [11]:
```
def coef(P):return([P(0),P.derivative(x)(0),P.derivative(x,2)(0)/2])
```

In [12]:
```
print coef(P)
```

```
[a, b, c]
```

In [13]:
```
P1(x)=1;P2(x)=x;P3(x)=x^2
Base=[P1,P2,P3]
Mu=matrix([coef(u(e)) for e in Base]).transpose();pretty_print(Mu)
```

Out[13]:

$$\begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}$$

In [14]:
```
print (X.transpose()*Mu.transpose()*A*Mu*X)[0,0].expand()
```

```
14/3*a^2 + 8*a*b + 22/5*b^2 + 20/3*a*c + 24/5*b*c + 46/15*c^2
```

Orthonormalisation de Gram-Schmidt de $(P1, P2, P3)$***, coordonnées dans la nouvelle base***

In [15]:
```
def N(P):return(P/sqrt(B(P,P)))
def proj(P,base):return(sum(B(P,L)*L for L in base))
```

In [16]:
```
L1=N(P1);print L1(x)
```

```
1/2*sqrt(2)
```

In [17]:
```
L2p=P2-proj(P2,[L1])
L2=N(P2);print L2(x)
```

```
3/2*sqrt(2/3)*x
```

In [18]:
```
L3p=P3-proj(P3,[L1,L2])
L3=N(L3p);print L3(x)
```

```
5/4*sqrt(2/5)*(3*x^2 - 1)
```

In [19]:
```
def coefL(P):return([B(P,L) for L in [L1,L2,L3]])
```

In [20]:
```
MLu=matrix([coefL(u(L)) for L in [L1,L2,L3]]).transpose()
pretty_print(MLu)
```

Out[20]:

$$\begin{pmatrix} 1 & \frac{4}{3}\sqrt{3} & 0 \\ \frac{2}{3}\sqrt{3} & 1 & \frac{2}{3}\sqrt{5}\sqrt{3} \\ 0 & \frac{2}{15}\sqrt{5}\sqrt{3} & 1 \end{pmatrix}$$

In [21]:
```
baseL=[L1,L2,L3]
pretty_print(matrix(3,3,lambda i,j:B(baseL[i],baseL[j])))
```

Out[21]:

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

In [22]:
```
P=a*L1+b*L2+c*L3
print B(u(P),u(P)).expand()
```

```
8/5*sqrt(5)*sqrt(3)*b*c + 4*sqrt(3)*a*b + 8/3*sqrt(5)*a*c + 7/3*a^2 + 33/5*b^2 + 23/3*c^
2
```

In [23]:
```
X=matrix([a,b,c]).transpose()
print (X.transpose()*MLu.transpose()*MLu*X)[0,0].expand()
```

```
8/5*sqrt(5)*sqrt(3)*b*c + 4*sqrt(3)*a*b + 8/3*sqrt(5)*a*c + 7/3*a^2 + 33/5*b^2 + 23/3*c^
2
```

Autre définition de E dans Sagemath

In [24]:
```
print sqrt(2).parent()
print SR,",",AA
```

```
Symbolic Ring
Symbolic Ring , Algebraic Real Field
```

In [25]:
```
S.<X>=SR[] #AA[]
print 1.parent(),",",S(1).parent()
print 1==S(1)
try:print 1.derivative()
except Exception as e:print(e)
print S(1).derivative()
P=1+X+X^2;print integrate(P(x),x,-1,1)
print B(P,P)
```

```
Integer Ring , Univariate Polynomial Ring in X over Symbolic Ring
1 == 1
'sage.rings.integer.Integer' object has no attribute 'derivative'
0
8/3
22/5
```

In [26]:
```
def u(P):return((1-X^2)*(P.derivative())+(2*X+1)*P)
print u(X^2)
v(P)=(1-X^2)*(P.derivative())+(2*X+1)*P
print v(X^2),"bizarre !!"
w(P)=derivative(P);print w(X^3);print derivative(X^3)
```

```
X^2 + 2*X
2*X^3 + 1 bizarre !!
1
3*X^2
```

Scholies

Corrigé rédigé de l'exercice 3 de la feuille 6 avec Sagemath

24 avril 2021, dehon@unice.fr

Ex.3 Soient A et B les matrices de $\mathrm{M}_{6,4}(\mathbb{Z})$ définies par

```
A=matrix(QQ,[[1,0,1,-1,1,2],[-1,1,0,0,1,0],[3,-2,1,-1,2,2],[0,0,2,2,3,2]]).transpose()
B=matrix(QQ,[[2,-1,0,1,1,2],[1,0,-1,1,2,1],[2,1,0,2,3,3],[3,1,2,4,4,1]]).transpose()
#show('A=',A,'B=',B)
```

$$A=\begin{pmatrix} 1 & -1 & 3 & 0\\ 0 & 1 & -2 & 0\\ 1 & 0 & 1 & 2\\ -1 & 0 & -1 & 2\\ 1 & 1 & 2 & 3\\ 2 & 0 & 2 & 2 \end{pmatrix}, B=\begin{pmatrix} 2 & 1 & 2 & 3\\ -1 & 0 & 1 & 1\\ 0 & -1 & 0 & 2\\ 1 & 1 & 2 & 4\\ 1 & 2 & 3 & 4\\ 2 & 1 & 3 & 1 \end{pmatrix}$$

Déterminer avec les méthodes de niveau 1 et 2 la dimension, une base et une équation de $\mathrm{Im}(A)\cap\mathrm{Im}(B)$ vu comme sous-$\mathbb{Q}$-espace vectoriel de $\mathbb{Q}^6$.
Combien de méthodes pouvez vous concevoir ?

Déterminer ces mêmes objets vus comme sous-$\mathbb{Z}$-modules de $\mathbb{Z}^6$.

Comparer avec la réponse donnée par l'instruction (de niveau 3) :

```
print matrix(ZZ,A).column_space().intersection(matrix(ZZ,B).column_space())
```

ou par

```
V=VectorSpace(QQ,6);VA=V.submodule(A.columns());VB=V.submodule(B.columns())
print VA.intersection(VB)
```

Méthode sur $\mathbb{Z}$:

- On détermine une base du sous-module $\mathrm{Sol}\subset\mathbb{Z}^4\times\mathbb{Z}^4$ formé des solutions de l'équation $AX=BY$ d'inconnue X,Y. Pour cela on détermine la matrice de l'application $(X,Y)\mapsto AX-BY$.
- On détermine la matrice M de l'application $\mathrm{Sol}\to\mathbb{Z}^6$, $(X,Y)\mapsto AX$.
- On détermine une base et une équation de $\mathrm{Im}(M)$ de façon standard via la forme de Smith de M.

Les base et système d'équations sur $\mathbb{Z}$ sont aussi des base et système d'équations sur $\mathbb{Q}$!

Méthode Sagemath pour les matrices : cf la Quick Reference Card Linear Algebra

In [1]:
```
A=matrix(ZZ,[[1,0,1,-1,1,2],[-1,1,0,0,1,0],[3,-2,1,-1,2,2],[0,0,2,2,3,2]]).transpose()
B=matrix(ZZ,[[2,-1,0,1,1,2],[1,0,-1,1,2,1],[2,1,0,2,3,3],[3,1,2,4,4,1]]).transpose()
show('A=',A,', B=',B)
```

Out[1]:

$$\mathtt{A=}\begin{pmatrix} 1 & -1 & 3 & 0\\ 0 & 1 & -2 & 0\\ 1 & 0 & 1 & 2\\ -1 & 0 & -1 & 2\\ 1 & 1 & 2 & 3\\ 2 & 0 & 2 & 2 \end{pmatrix}, \mathtt{B=}\begin{pmatrix} 2 & 1 & 2 & 3\\ -1 & 0 & 1 & 1\\ 0 & -1 & 0 & 2\\ 1 & 1 & 2 & 4\\ 1 & 2 & 3 & 4\\ 2 & 1 & 3 & 1 \end{pmatrix}$$

In [2]:
```
N=A.augment(B);show(N)
```

Out[2]:

$$\begin{pmatrix} 1 & -1 & 3 & 0 & 2 & 1 & 2 & 3\\ 0 & 1 & -2 & 0 & -1 & 0 & 1 & 1\\ 1 & 0 & 1 & 2 & 0 & -1 & 0 & 2\\ -1 & 0 & -1 & 2 & 1 & 1 & 2 & 4\\ 1 & 1 & 2 & 3 & 1 & 2 & 3 & 4\\ 2 & 0 & 2 & 2 & 2 & 1 & 3 & 1 \end{pmatrix}$$

In [3]:
```
S=N.smith_form();DN=S[0];QN=S[2];show('D=',DN,', Q=',QN)
```

Out[3]:

$$\mathtt{D=}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \mathtt{Q=}\begin{pmatrix} 23 & -122 & -107 & 126 & 10 & -10 & 135 & -35\\ -50 & 264 & 232 & -275 & -20 & 21 & -285 & 76\\ -11 & 59 & 52 & -61 & -5 & 5 & -66 & 17\\ 10 & -52 & -46 & 54 & 4 & -4 & 56 & -15\\ -32 & 167 & 146 & -174 & -12 & 13 & -178 & 48\\ 32 & -167 & -147 & 173 & 13 & -14 & 183 & -48\\ -4 & 21 & 19 & -21 & -2 & 2 & -26 & 6\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$\mathrm{Ker}(D)=\mathrm{Ker}(PNQ)$ est engendré par les deux derniers vecteurs de la base canonique de $\mathbb{Z}^8$.
Donc $\mathrm{Ker}(N)=Q\mathrm{Ker}(D)$ a pour base l'image par Q de ces deux vecteurs i.e. les deux dernières colonnes de Q.
L'application $(X,Y)\mapsto AX, \mathrm{Ker}(N)\to\mathbb{Z}^6$ à pour matrice relativement à cette base de $\mathrm{Ker}(N)$ et à la base canonique de $\mathbb{Z}^6$ l'image par $(A|0)$ des deux dernières colonnes de Q.

In [4]:
```
AA=A.augment(matrix(ZZ,6,4,lambda i,j:0))
show(AA,QN.matrix_from_columns([6,7]))
```

Out[4]:

$$\begin{pmatrix} 1 & -1 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 135 & -35 \\ -285 & 76 \\ -66 & 17 \\ 56 & -15 \\ -178 & 48 \\ 183 & -48 \\ -26 & 6 \\ 1 & 0 \end{pmatrix}$$

In [5]:
```
M=AA*QN.matrix_from_columns([6,7]);show(M)
```

Out[5]:

$$\begin{pmatrix} 222 & -60 \\ -153 & 42 \\ 181 & -48 \\ 43 & -12 \\ -114 & 30 \\ 250 & -66 \end{pmatrix}$$

On calcule maintenant la forme de Smith de M.
Si $PMQ = D$ alors $\mathrm{Im}(M) = P^{-1}\mathrm{Im}(D)$ a pour base l'image par P^{-1} d'une base de $\mathrm{Im}(D)$ (adaptée à la base de $\mathbb{Z}^6$ formée par les vecteurs colonnes de P^{-1}) et pour équation $PX = 0 \bmod d_1\mathbb{Z} \times \cdots \times d_6\mathbb{Z}$ (d'inconnue X) où $Y = 0 \bmod d_1\mathbb{Z} \times \cdots \times d_6\mathbb{Z}$ est une équation de $\mathrm{Im}(D)$.

Observons que si les deux vecteurs colonnes de M sont non liés, ils forment déjà une base de $\mathrm{Im}(M)$, non adaptée cependant à une base de $\mathbb{Z}^6$ a priori.

In [6]:
```
T=M.smith_form();DT=T[0];PT=T[1];show('D=',DT,', P=',PT,', P⁻¹=',PT^(-1))
```

Out[6]:

$$\text{D=}\begin{pmatrix} 1 & 0 \\ 0 & 6 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \text{ P=}\begin{pmatrix} 0 & 0 & 3 & 0 & 0 & -2 \\ 0 & 0 & 2 & 0 & -20 & -11 \\ 0 & 0 & -3 & 1 & 0 & 2 \\ 0 & 1 & -37 & 0 & 100 & 73 \\ 1 & 0 & 118 & 0 & -280 & -214 \\ 0 & 0 & 4 & 0 & -9 & -7 \end{pmatrix}, \text{ P}^{-1}\text{=}\begin{pmatrix} -30 & -14 & 0 & 0 & 1 & 0 \\ 9 & 5 & 0 & 1 & 0 & 0 \\ -41 & -18 & 0 & 0 & 0 & 40 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 30 & 13 & 0 & 0 & 0 & -29 \\ -62 & -27 & 0 & 0 & 0 & 60 \end{pmatrix}$$

In [7]:
```
show('base adaptée de Im(M) :',PT^(-1)*DT)
```

Out[7]:

$$\text{base adaptée de Im(M) :} \begin{pmatrix} -30 & -84 \\ 9 & 30 \\ -41 & -108 \\ 1 & 0 \\ 30 & 78 \\ -62 & -162 \end{pmatrix}$$

In [8]:
```
show("Système d'équation de Im(M) :")
X=var(['x'+str(i) for i in range(1,7)])#;print(X)
d=[1,6,0,0,0,0]
for i in range(6):
    show((PT*vector(X))[i],' = 0 mod',LatexExpr('~'),d[i])
```

Out[8]:

Système d'équation de Im(M) :

$$3\,x_{3} - 2\,x_{6} = 0 \bmod 1$$

$$2\,x_{3} - 20\,x_{5} - 11\,x_{6} = 0 \bmod 6$$

$$-3\,x_{3} + x_{4} + 2\,x_{6} = 0 \bmod 0$$

$$x_{2} - 37\,x_{3} + 100\,x_{5} + 73\,x_{6} = 0 \bmod 0$$

$$x_{1} + 118\,x_{3} - 280\,x_{5} - 214\,x_{6} = 0 \bmod 0$$

$$4\,x_{3} - 9\,x_{5} - 7\,x_{6} = 0 \bmod 0$$

La première équation est toujours satisfaite (elle équivaut à $3x_3-2x_6 \in \mathbb{Z}$). Une équation y=0 mod 0 équivaut à y=0.
D'où le système sous forme standard :

In [9]:
```
show((PT*vector(X))[1]," = 0 mod",LatexExpr('~'),d[1])
for i in range(2,6):
    show((PT*vector(X))[i]," = 0")
```

Out[9]:
$$2\,x_{3} - 20\,x_{5} - 11\,x_{6} \;=\; 0 \;\mathrm{mod}\; 6$$

Out[9]:
$$-3\,x_{3} + x_{4} + 2\,x_{6} \;=\; 0$$

Out[9]:
$$x_{2} - 37\,x_{3} + 100\,x_{5} + 73\,x_{6} \;=\; 0$$

Out[9]:
$$x_{1} + 118\,x_{3} - 280\,x_{5} - 214\,x_{6} \;=\; 0$$

Out[9]:
$$4\,x_{3} - 9\,x_{5} - 7\,x_{6} \;=\; 0$$

Rq : on doit avoir $\mathrm{Im}(M) \subset \mathrm{Im}(A)$ c'est à dire $Y = AX$ d'inconnue X a une solution pour chaque colonne Y de M.
Idem en remplaçant A par B.

Autre méthode :

Chercher d'abord un système d'équation de $\mathrm{Im}(A)$ et un système d'équation de $\mathrm{Im}(B)$.
La concaténation des deux systèmes est un système d'équations de $\mathrm{Im}(A) \cap \mathrm{Im}(B)$.
La résolution de ce système donne une base de $\mathrm{Im}(A) \cap \mathrm{Im}(B)$.

In [10]:
```
DA,PA=A.smith_form()[0:2];show(DA,PA)
```

Out[10]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 3 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & -8 & 0 & -4 & 5 \\ 0 & 0 & -3 & 1 & 0 & 2 \\ 1 & 1 & 1 & 0 & 0 & -1 \end{pmatrix}$$

In [11]:
```
show("Equation de Im(A) :")
show((PA*vector(X))[3]," = 0 mod ",DA[3,3])
for i in range(4,6):
    show((PA*vector(X))[i]," = 0")
```

Out[11]:
$$\mathrm{Equation\ de\ Im(A)\ :}$$

Out[11]:
$$2\,x_{1} - 8\,x_{3} - 4\,x_{5} + 5\,x_{6} \;=\; 0 \;\mathrm{mod}6$$

Out[11]:
$$-3\,x_{3} + x_{4} + 2\,x_{6} \;=\; 0$$

Out[11]:
$$x_{1} + x_{2} + x_{3} - x_{6} \;=\; 0$$

In [12]:
```
DB,PB=B.smith_form()[0:2];show(DB,PB)
```

Out[12]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 2 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -3 & -2 & 1 & 1 & 1 & 1 \\ 20 & 11 & -2 & -15 & 0 & -7 \end{pmatrix}$$

In [13]:
```
show("Equation de Im(B) :")
for i in range(4,6):
    show((PB*vector(X))[i]," = 0")
```

Out[13]:
$$\mathrm{Equation\ de\ Im(B)\ :}$$

Out[13]:
$$-3\,x_{1} - 2\,x_{2} + x_{3} + x_{4} + x_{5} + x_{6} \;=\; 0$$

Out[13]:
$$20\,x_{1} + 11\,x_{2} - 2\,x_{3} - 15\,x_{4} - 7\,x_{6} \;=\; 0$$

In [14]:
```
show("Equation de Im(A)∩Im(B) :")
```

```
show((PA*vector(X))[3]," = 0 mod ",DA[3,3])
for i in range(4,6):
    show((PA*vector(X))[i]," = 0")
for i in range(4,6):
    show((PB*vector(X))[i]," = 0")
```

Out[14]:

$$\text{Equation de Im(A)}\cap\text{Im(B) :}$$

$$2\,x_1 - 8\,x_3 - 4\,x_5 + 5\,x_6 \;=\; 0 \;\mathrm{mod}6$$

$$-3\,x_3 + x_4 + 2\,x_6 \;=\; 0$$

$$x_1 + x_2 + x_3 - x_6 \;=\; 0$$

$$-3\,x_1 - 2\,x_2 + x_3 + x_4 + x_5 + x_6 \;=\; 0$$

$$20\,x_1 + 11\,x_2 - 2\,x_3 - 15\,x_4 - 7\,x_6 \;=\; 0$$

Pour obtenir une base des solutions de ce système on introduit une inconnue supplémentaire k et on change la première équation en $2x_1 - 8x_3 - 4x_5 + 5x_6 + 6k = 0$.
On forme la matrice E du système, on déduit une base de Ker(E) de la forme de Smith de E.
Im(A)∩Im(B) est alors l'image de la projection $\mathrm{Ker}(E) \to \mathbb{Z}^6$, $(x_1, \ldots, x_6, k) \mapsto (x_1, \ldots, x_6)$.
On déduit une base de Im(A)∩Im(B) de la forme de Smith de la matrice de cette projection. La matrice est extraite de la matrice des coordonnées des vecteurs de la base de Ker(E) qu'on a calculée en retenant les 6 premières lignes. Si le rang de cette matrice reste égal au nombre de colonnes, les colonnes de la matrice forment déjà une base de l'image, c'est à dire de Im(A)∩Im(B).

In [15]:
```
E=(PA[3:6].stack(PB[4:6])).augment(matrix(5,1,[6,0,0,0,0]))
#E=block_matrix(1,2,[block_matrix(2,1,[PA[3:6],PB[4:6]]),matrix(5,1,[6,0,0,0,0])])
show(E)
DE,QE=E.smith_form()[0:3:2];show(DE,QE)
```

Out[15]:

$$\begin{pmatrix} 2 & 0 & -8 & 0 & -4 & 5 & 6 \\ 0 & 0 & -3 & 1 & 0 & 2 & 0 \\ 1 & 1 & 1 & 0 & 0 & -1 & 0 \\ -3 & -2 & 1 & 1 & 1 & 1 & 0 \\ 20 & 11 & -2 & -15 & 0 & -7 & 0 \end{pmatrix}$$

Out[15]:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 99 & 304 & -890 & 89 & -29 & 5340 & -138 \\ -121 & -376 & 1103 & -110 & 36 & -6618 & 171 \\ 5 & 12 & -32 & 3 & -1 & 192 & -5 \\ 51 & 156 & -458 & 46 & -15 & 2748 & -71 \\ 17 & 53 & -155 & 16 & -5 & 930 & -24 \\ -18 & -60 & 181 & -18 & 6 & -1086 & 28 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

In [16]:
```
EE=QE[0:6,5:7];show(EE," de rang ",LatexExpr('~'),EE.rank())
```

Out[16]:

$$\begin{pmatrix} 5340 & -138 \\ -6618 & 171 \\ 192 & -5 \\ 2748 & -71 \\ 930 & -24 \\ -1086 & 28 \end{pmatrix} \text{ de rang } 2$$

Les bases et systèmes d'équations obtenues par les deux méthodes n'ont aucune raison d'être identique. On peut tester la pertinence des réponses en vérifiant que la base obtenue avec la première méthode est solution du système d'équations obtenue avec la deuxième méthode et inversement.

In [17]:
```
show((PT[1]*EE)%6)
show(PT[2:6]*EE)
```

Out[17]:

$$(0,\ 0)$$

Out[17]:

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

Générateur du groupe multiplicatif de Fp

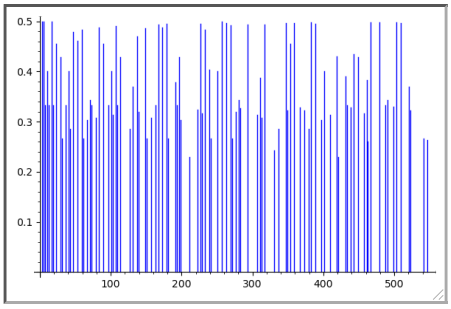
F-X. Dehon, mars 2022, mise à jour 15 mars 2023

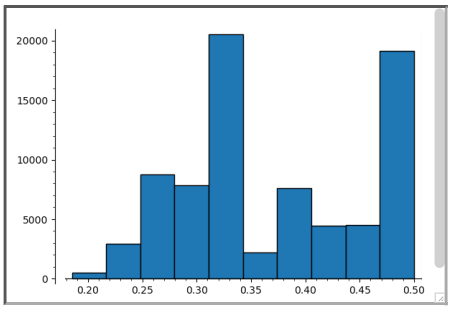
On sait $(\mathbb{Z}/n\mathbb{Z})^\times$ est cyclique si n est premier (mais pas seulement si). On cherche alors un générateur ; il y en a $\varphi(n-1)$.

Si on prend un nombre au hasard dans $\{1,\ldots,n-1\}$ il a une probabilité $\frac{\varphi(n-1)}{n-1}$ d'être premier avec $n-1$. A quoi ressemble $\frac{\varphi(n-1)}{n-1}$ pour n décrivant les premiers nombres premiers ? Essai sur jhub[1]

```
P=[p for p in primes(3,1000000)]
l=[euler_phi(p-1)/(p-1) for p in P]
print('eff=',len(l))
print('min=',min(l).n(digits=2))
print('max=',max(l).n(digits=2))
print('moy=',mean(l).n(digits=2))
show(sum(line([(p,0),(p,euler_phi(p-1)/(p-1))]) for p in P[:100]))
show(histogram(l))
```

↝ eff= 78497, min=0.19 , max=0.50, moy= 0.37





https://en.wikipedia.org/wiki/Euler's_totient_function

Certains des nombres pris au hasard dans $\{1,\ldots,n-1\}$ sont de mauvais candidats : 1 est d'ordre 1, $n-1$ d'ordre 2, $4=2^2$ ne peut pas être générateur sinon 2 serait d'ordre $2(n-1)$, idem pour 3^2 etc. Peut être que les mauvais candidats sont en proportion négligeable lorsque n est très grand ?
Quelle est la "probabilité" que 2 soit un carré ?

2 est il un bon choix ? Mieux que 3, 5, 6 ?

```
n=10000
print(sum([IntegerModRing(p)(2).multiplicative_order()==p-1 for p in P[:n]])/n.n(digits=2))
print(sum([IntegerModRing(p)(3).multiplicative_order()==p-1 for p in P[1:n]])/(n-1).n(digits=2))
print(sum([IntegerModRing(p)(5).multiplicative_order()==p-1 for p in P[2:n]])/(n-2)
print(sum([IntegerModRing(p)(6).multiplicative_order()==p-1 for p in P[2:n]])/(n-2).n(digits=2))
```

↝ 0.38 , 0.38 , 0.40 , 0.38

```
n=100000;print(sum([IntegerModRing(p)(5).multiplicative_order()==p-1 for p in P[2:n]])/(n-2).n(digits=3))
```

↝ 0.393

```
n=10000;print(mean([euler_phi(p-1)/(p-1) for p in P[2:n]]).n(digits=2))
```

↝ 0.37

1. https://math.unice.fr/jhub ↩

Temps Factorisation des entiers sur la machine math13 (jhub) suivant algorithme choisi

```
top - 15:20:48 up 110 days,  4:13,  3 users,  load average: 8.68, 8.47, 8.03
Tasks: 646 total,   9 running, 637 sleeping,   0 stopped,   0 zombie
%Cpu(s): 17.1 us,  0.7 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 772660.4 total,  45620.5 free,  30436.0 used, 696603.9 buff/cache
MiB Swap:   8192.0 total,   8179.0 free,     13.0 used. 737266.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
3642831 gscarel+  20   0 6984840   6.4g  15032 R 100.3   0.8   2872:24 LassoBe+
3615778 gscarel+  20   0 4606104   4.1g  15000 R 100.0   0.5   4075:45 LassoBe+
3649254 gscarel+  20   0 5381216   4.9g  15000 R 100.0   0.6   2473:35 LassoBe+
3649313 gscarel+  20   0 3568524   3.1g  15108 R 100.0   0.4   2471:04 LassoBe+
3649360 gscarel+  20   0 4219440   3.8g  14828 R 100.0   0.5   2470:07 LassoBe+
3940230 tlaporte  20   0  588996 587156   6504 R 100.0   0.1   1282:13 skeleton
3952310 dehon     20   0  191.6g 297104  93784 R 100.0   0.0  10:57.61 python3
3953140 tlaporte  20   0   43.9g   2.7g 125988 R 100.0   0.4  12:15.62 python
3641530 tlaporte  20   0   43.9g   3.3g 744256 S  41.7   0.4   1495:51 python
3641560 root      20   0       0      0      0 S   6.0   0.0 130:29.11 nv_queue
3641567 root      20   0       0      0      0 S   3.3   0.0  97:49.26 nv_queue
2392436 cpakzad   20   0   19468   7716   4128 S   1.3   0.0 637:37.43 htop
   1250 root      20   0 3414916  23908   6696 S   1.0   0.0   1601:45 contain+
     12 root      20   0       0      0      0 I   0.3   0.0 208:59.31 rcu_sch+
3641558 root     -51   0       0      0      0 S   0.3   0.0   4:46.99 irq/218+
3952028 dehon     20   0 6827100 160908  30252 S   0.3   0.0   0:37.76 python
3953079 dehon     20   0   18476   4740   3460 R   0.3   0.0   0:05.52 top
```

In [1]:
```
#Nbres à factoriser
P=[];K=30
for i in range(K):
    p=next_prime(10^i);pp=next_prime(p);n=p*pp
    P.append([p,pp,n])
```

In [2]:
```
#mesure temps en secondes de factorisation de n suivant algo f
import time
def temps(f,n):
    t=time.time()
    p=f(n)
    tt=time.time()-t
    return(tt,n,p,n%p)
```

In [7]:
```
#liste log temps factorisation des entiers avec facteurs premiers de l'ordre de 10^k, 1≤k≤K
def ltemps(f,K):
    l=[]
    for i in [1..K]:
        l.append([i,ln(temps(f,P[i][2])[0])])
    return(l)
```

In [5]:
```
#Algorithme de Pollard - Brent
#Utilisé en 1981 en ~2h pour la 1ère factorisation du nbre de Fermat F8=1+2^(2^8) ≈ 10^77 avec un premier facteur
def f3(n):
    def f(x):return((x^2+1)%n)
    x0=11
    i,p,x,y=(0,1,x0,f(x0))
    while gcd(x-y,n)==1:
        if p<=i:i,p,x,y=(i,p+1,x,f(y))
        else:i,p,x,y=(i+p,1,y,f(y))
    return(gcd(x-y,n))
```

In [11]:
```
(ln(2)/ln(10)*2**8).n(digits=3)
```

Out[11]:
```
77.1
```

In [6]:
```
#Algorithme natif Sagemath
#factor()?
def f4(n):
    return(factor(n)[0][0])
```

In [12]:
```
lt3=ltemps(f3,15)
```

In [13]:
```
print([16,ln(temps(f3,P[16][2])[0])])
```

```
[16, 4.4092517762998344]
```

In [16]:
```
lt3.append([16, 4.4092517762998344])
```

In [18]:
```
lt3.append([17,ln(temps(f3,P[17][2])[0])])
```

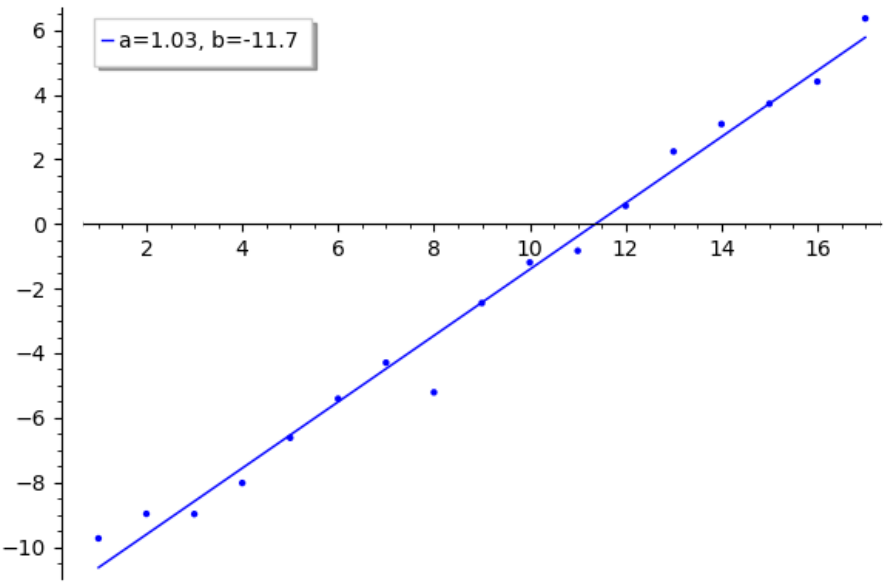
In [19]:
```
print(lt3)
```

```
[[1, -9.735809226153815], [2, -8.974475951076858], [3, -8.982037423777435], [4, -8.020124094525494], [5, -6.62175
6427008436], [6, -5.410929145807663], [7, -4.294719026744006], [8, -5.213407232794956], [9, -2.4478845904948003],
[10, -1.1916757533528017], [11, -0.8304535203251717], [12, 0.5608372434109334], [13, 2.244226548341697], [14, 3.0
92249065955822], [15, 3.7241116685572075], [16, 4.4092517762998344], [17, 6.364685368521412]]
```

In [20]:
```
#lt3=[[1, -9.735809226153815], [2, -8.974475951076858], [3, -8.982037423777435], [4, -8.020124094525494], [5, -6.
```

In [14]:
```
def coef(l):
    var('a b')
    model(x)=a*x+b
    sol=find_fit(l,model)
    return(sol[0].rhs(),sol[1].rhs())
```

In [26]:
```
l=[[lt3,'blue']]#,[lt4[10:],'grey'],[lt4[:15],'black']]
dessin=plot([])
for i in l:
    a,b=coef(i[0])
    dessin+=plot(a*x+b,x,xmin=i[0][0][0],xmax=i[0][-1][0],color=i[1],legend_label='a='+str(a.n(digits=3))+', b='+
    dessin+=list_plot(i[0],color=i[1])
show(dessin)
```



In [27]:
```
#n du TP RSA
n=313297851918307610502911324433872045797703258998425302542562974 63
f=factor(n)
print(f)
```

```
12345678998745632485 51 * 3214523698745962578373 * 7894521369756421254781
```

In [29]:
```
p=123456789987456324855 1
print((ln(n)/ln(10)).n(digits=3),(ln(p)/ln(10)).n(digits=3))
```

```
64.5 21.1
```

Addendum au TP4 : polynôme annulateur de $\sqrt{2}+\sqrt{3}+\sqrt{5}$

27 avril 2022 - F-X. Dehon

In [1]:
```
banner()
```

```
| SageMath version 9.5, Release Date: 2022-01-30                     |
| Create a "Sage Worksheet" file for the notebook interface.         |
| Enhanced for CoCalc.                                               |
| Using Python 3.9.9. Type "help()" for help.                        |
```

In [2]:
```
z=sqrt(2)+sqrt(3)+sqrt(5);print(z,', ',z.parent())
```

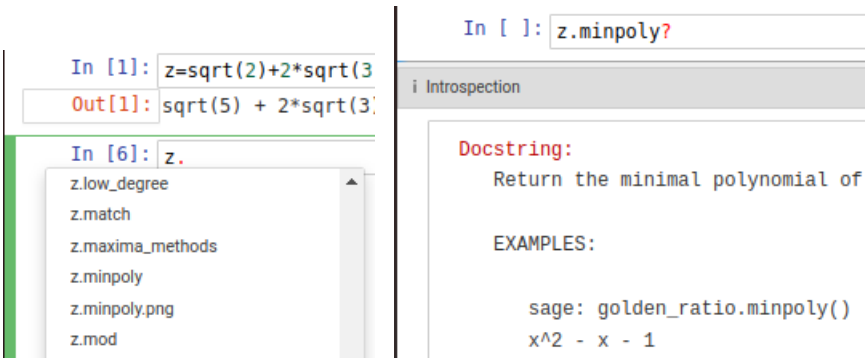
```
sqrt(5) + sqrt(3) + sqrt(2) ,  Symbolic Ring
```

Construire un polynôme annulateur de z à coefficients dans $\mathbb{Q}$

Avant toute chose, définir une stratégie en veillant à ce qu'elle soit implémentable dans Sagemath.

Rq : Sagemath peut il de lui même fournir le polynôme minimal de z ? Faire une recherche sur "Sagemath minimal polynomial".

Dans Cocalc on peut obtenir la liste des méthodes qui s'appliquent à z (préalablement défini) en entrant dans une cellule `z.` suivi de \<tab>. On y voit minpoly au nom évocateur. On peut ensuite obtenir de l'aide sur l'intruction en entrant `z.minpoly()?` dans une cellule (pourvu que z soit préalablement défini) :



In [3]:
```
P=z.minpoly()
show(P)
show(P(z))
```

Out[3]:

$$x^8 - 40x^6 + 352x^4 - 960x^2 + 576$$

Out[3]:

$$\left(\sqrt{5}+\sqrt{3}+\sqrt{2}\right)^8 - 40\left(\sqrt{5}+\sqrt{3}+\sqrt{2}\right)^6 + 352\left(\sqrt{5}+\sqrt{3}+\sqrt{2}\right)^4 - 960\left(\sqrt{5}+\sqrt{3}+\sqrt{2}\right)^2 + 576$$

On s'attendait à obtenir 0. En faisant une recherche sur "Sagemath simplify radical expression" on trouve comme méthodes `P(z).simplify_full` , `P(z).expand()` , également dans la liste des méthodes disponibles mentionnée plus haut.

In [4]:
```
print(P(z).simplify_full())
print(P(z).expand())
```

```
0
0
```

`z.minpoly()` est une instruction de niveau 3 qu'on peut utiliser par curiosité mais qui ne répond pas à notre attente.

Stratégie 1.

Du point de vue mathématique on observe que les puissances de z sont toutes combinaison linéaire à coefficients dans $\mathbb{Q}$ ($\mathbb{Z}$ en fait) de 1, $\sqrt{2}$, $\sqrt{3}$, $\sqrt{5}$, $\sqrt{6}$, $\sqrt{10}$, $\sqrt{15}$, $\sqrt{30}$.

Toute famille de plus de n vecteurs d'un $\mathbb{Q}$-espace vectoriel admettant une famille génératrice de n vecteurs est forcément liée. Donc la famille $(1,z,\ldots,z^8)$ est liée.

On veut une relation non triviale explicite, ce qui donnera un polynôme annulateur non nul de z.

Pour cela on forme la matrice M des coordonnées des z^k, $0\leq k\leq 8$, relativement à la famille génératrice $(1,\sqrt{2},\ldots,\sqrt{30})$ et on cherche une relation non triviale entre les colonnes.

Pour cela on échelonne M suivant les colonnes ou bien on calcule sa forme de Smith.

Du point de vue Sagemath se pose la difficulté d'extraire les coordonnées de z^k à partir de l'expression de z^k.

D'abord à la main :

In [5]:
```
for k in range(9):print(k,' ',(z^k).expand())
```

```
0   1
1   sqrt(5) + sqrt(3) + sqrt(2)
2   2*sqrt(5)*sqrt(3) + 2*sqrt(5)*sqrt(2) + 2*sqrt(3)*sqrt(2) + 10
3   6*sqrt(5)*sqrt(3)*sqrt(2) + 20*sqrt(5) + 24*sqrt(3) + 26*sqrt(2)
4   56*sqrt(5)*sqrt(3) + 64*sqrt(5)*sqrt(2) + 80*sqrt(3)*sqrt(2) + 224
5   200*sqrt(5)*sqrt(3)*sqrt(2) + 520*sqrt(5) + 664*sqrt(3) + 784*sqrt(2)
6   1584*sqrt(5)*sqrt(3) + 1904*sqrt(5)*sqrt(2) + 2448*sqrt(3)*sqrt(2) + 6160
7   5936*sqrt(5)*sqrt(3)*sqrt(2) + 14720*sqrt(5) + 18976*sqrt(3) + 23024*sqrt(2)
8   45568*sqrt(5)*sqrt(3) + 55552*sqrt(5)*sqrt(2) + 71680*sqrt(3)*sqrt(2) + 176576
```

In [6]:
```
M=matrix(ZZ,8,9,{(0,0):1})
M[0]=[1,0,10,0,224,0,6160,0,176576]
for i in [1..3]:M[i,1]=1
for i in [4..6]:M[i,2]=2
d={1:26,2:24,3:20,7:6}
for i in d.keys():M[i,3]=d[i]
d={4:80,5:64,6:56}
for i in d.keys():M[i,4]=d[i]
d={1:784,2:664,3:520,7:200}
for i in d.keys():M[i,5]=d[i]
d={4:2448,5:1904,6:1584}
for i in d.keys():M[i,6]=d[i]
d={1:23024,2:18976,3:14720,7:5936}
for i in d.keys():M[i,7]=d[i]
d={4:71680,5:55552,6:45568}
for i in d.keys():M[i,8]=d[i]
show('M=',M,M.parent())
```

Out[6]:

$$\mathtt{M=}\begin{pmatrix} 1 & 0 & 10 & 0 & 224 & 0 & 6160 & 0 & 176576 \\ 0 & 1 & 0 & 26 & 0 & 784 & 0 & 23024 & 0 \\ 0 & 1 & 0 & 24 & 0 & 664 & 0 & 18976 & 0 \\ 0 & 1 & 0 & 20 & 0 & 520 & 0 & 14720 & 0 \\ 0 & 0 & 2 & 0 & 80 & 0 & 2448 & 0 & 71680 \\ 0 & 0 & 2 & 0 & 64 & 0 & 1904 & 0 & 55552 \\ 0 & 0 & 2 & 0 & 56 & 0 & 1584 & 0 & 45568 \\ 0 & 0 & 0 & 6 & 0 & 200 & 0 & 5936 & 0 \end{pmatrix} \mathrm{Mat}_{8\times 9}(\mathbf{Z})$$

En cherchant bien on découvre qu'on peut extraire les coefficients de l'expression normalisée de z^k avec Sagemath :

In [7]:
```
z=sqrt(2)+sqrt(3)+sqrt(5)
b=[sqrt(2),sqrt(3),sqrt(5),sqrt(6),sqrt(10),sqrt(15),sqrt(30)]
w=SR.wild(0)
l=[]
for k in [0..8]:
    zz=(z^k).expand().maxima_methods().rootscontract()
    l+=[[zz.subs(w==0)]+[zz.coefficient(i) for i in b]]
MM=column_matrix(l);show('MM=',MM,MM.parent())
```

Out[7]:

$$\texttt{MM=}\begin{pmatrix} 1 & 0 & 10 & 0 & 224 & 0 & 6160 & 0 & 176576 \\ 0 & 1 & 0 & 26 & 0 & 784 & 0 & 23024 & 0 \\ 0 & 1 & 0 & 24 & 0 & 664 & 0 & 18976 & 0 \\ 0 & 1 & 0 & 20 & 0 & 520 & 0 & 14720 & 0 \\ 0 & 0 & 2 & 0 & 80 & 0 & 2448 & 0 & 71680 \\ 0 & 0 & 2 & 0 & 64 & 0 & 1904 & 0 & 55552 \\ 0 & 0 & 2 & 0 & 56 & 0 & 1584 & 0 & 45568 \\ 0 & 0 & 0 & 6 & 0 & 200 & 0 & 5936 & 0 \end{pmatrix} \mathrm{Mat}_{8\times 9}(\mathrm{SR})$$

On calcule maintenant une relation entre les colonnes de M

In [8]:
```
# pourvu que les coefficients de M soit de type convenable et pas Symbolic Ring !
D,P,Q=M.smith_form();show(D);show(Q)
```

Out[8]:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 576 & 0 \end{pmatrix}$$

Out[8]:

$$\begin{pmatrix} 0 & 1 & -40368 & -8074 & 1208 & 129184 & -96 & 0 & -576 \\ 1 & 0 & 534 & -2 & 0 & 968 & 0 & 2544 & 0 \\ 0 & 0 & 67205 & 13441 & -1948 & -215056 & 632 & 0 & 960 \\ 0 & 0 & -311 & 0 & 0 & -588 & 0 & -1520 & 0 \\ 0 & 0 & -24640 & -4928 & 705 & 78848 & -312 & 0 & -352 \\ 0 & 0 & 39 & 0 & 0 & 77 & 0 & 194 & 0 \\ 0 & 0 & 2800 & 560 & -80 & -8960 & 39 & 0 & 40 \\ 0 & 0 & -1 & 0 & 0 & -2 & 0 & -5 & 0 \\ 0 & 0 & -70 & -14 & 2 & 224 & -1 & 0 & -1 \end{pmatrix}$$

La dernière colonne de Q donne une relation sur les colonnes de M :

In [9]:
```
show(M*Q.column(8))
```

Out[9]:

$$(0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)$$

On convertit la dernière colonne de Q en un polynôme :

In [10]:
```
P=QQ['X'](list(Q.column(8)))
show('P=',P,' ','de type : ',P.parent())
show('P(z)=',P(z).expand())
```

Out[10]:

$$\texttt{P=}\ -X^8+40X^6-352X^4+960X^2-576\ \texttt{de type :}\ \mathbf{Q}[X]$$

Out[10]:

$$\texttt{P(z)=0}$$

Un modèle algébrique pur pour $\mathbb{Q}[\sqrt{(2)},\sqrt{(3)},\sqrt{(5)}]\subset\mathbb{R}$

In [11]:
```
X,Y,Z=QQ['X,Y,Z'].gens()
z=X+Y+Z
zz=(z^2)%(X^2-2)%(Y^2-3)%(Z^2-5)
print(zz)
```

```
2*X*Y + 2*X*Z + 2*Y*Z + 10
```

In [12]:
```
[zz.constant_coefficient()]+[zz.monomial_coefficient(i) for i in [X,Y,Z,X*Y,X*Z,Y*Z]]#Et XYZ ?!
```

Out[12]: [10, 0, 0, 0, 2, 2, 2]

In [13]:
```
def zk(z,k):
    return((((z^k)%(X^2-2))%(Y^2-3))%(Z^2-5))
```

In [14]:
```
M=column_matrix([[zk(z,k).constant_coefficient()]+[zk(z,k).monomial_coefficient(i) for i in [X,Y,Z,X*Y,X*Z,
show('M=',M,M.parent())
```

Out[14]:

$$\mathtt{M=}\begin{pmatrix} 1 & 0 & 10 & 0 & 224 & 0 & 6160 & 0 & 176576 \\ 0 & 1 & 0 & 26 & 0 & 784 & 0 & 23024 & 0 \\ 0 & 1 & 0 & 24 & 0 & 664 & 0 & 18976 & 0 \\ 0 & 1 & 0 & 20 & 0 & 520 & 0 & 14720 & 0 \\ 0 & 0 & 2 & 0 & 80 & 0 & 2448 & 0 & 71680 \\ 0 & 0 & 2 & 0 & 64 & 0 & 1904 & 0 & 55552 \\ 0 & 0 & 2 & 0 & 56 & 0 & 1584 & 0 & 45568 \\ 0 & 0 & 0 & 6 & 0 & 200 & 0 & 5936 & 0 \end{pmatrix} \mathrm{Mat}_{8\times 9}(\mathbf{Q})$$

Stratégie 2.

On observe que la multiplication par z laisse stable le $\mathbb{Q}$ -espace vectoriel engendré par 1, √2, √3, √5, √6, √10, √15, √30. On détermine la matrice M de la multiplication par z : x↦zx relativement à la famille génératrice (1, √2, √3, √5, √6, √10, √15, √30) et on observe qu'un polynôme annulateur de cette matrice annule z.
Le théorème de Cayley-Hamilton nous donne un tel polynôme annulateur. Pour le calculer on peut déclarer X comme générateur Sagemath de $\mathbb{Q}[X]$, former la matrice M-XI et calculer son déterminant avec la fonction det() ou via la forme de Smith de M-XI. Sagemath dispose aussi de l'instruction magique M.charpoly().

Mettre en place la stratégie 2

In [0]:
```
```

L2ande - Corrigé du TP maximisation d'une fonction sous contraintes

9 mai 2022

In [1]:
```
banner()
```

```
┌────────────────────────────────────────────────────────────────────┐
│ SageMath version 9.4, Release Date: 2021-08-22                     │
│ Using Python 3.9.5. Type "help()" for help.                        │
└────────────────────────────────────────────────────────────────────┘
```

Partie 1 : maximisation d'une fonction $\mathbb{R}\to\mathbb{R}$ sur un intervalle [a,b]

1.1. Lecture des points d'intérêt sur le graphe

Habituellement on forme le **tableau de variation de la fonction** (qui est une allure du graphe très épuré avec les points d'intérêt explicités : extrémités de l'intervalle, points de changement de variation de la fonction).
Le graphe de la fonction[1] donne une représentation visuelle du tableau de variation. Ne manque qu'un traqueur des points sur le graphe pour avoir une approximation numérique à la précision souhaitée des points d'intérêt.

[1] *Représentation approchée et non démontrée a priori : on ne connaît pas la certification de l'algorithme Sagemath qui dessine le graphe !*

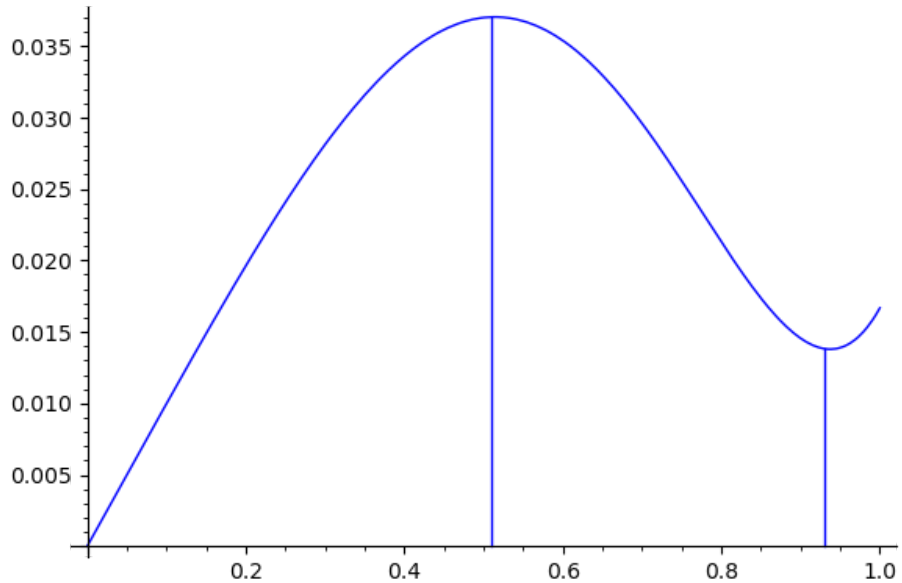
✍️ Exemple : $g : [0,1] \to \mathbb{R}, x \mapsto \frac{1}{6}x^6 - \frac{1}{4}x^4 + \frac{1}{10}x$. Quels sont les points d'intérêt lus sur le graphe ci-dessous ? (Donner une valeur numérique approximative.)
Donner une valeur à 10^{-3} près du x où g atteint son maximum en vous aidant du dessin interactif ci-dessous et en redéfinissant $a, b, ymin$ à mesure.

In [20]:
```
@interact
def _(a=input_box(0.n(digits=4), width=10),b=input_box(1, width=10),ymin=input_box(0, width=20)):
    var('x')
    @interact
    def _(expr=1/6*x^6-1/4*x^4+1/10*x,x0=slider(a,b,(b-a)/100,default = (a*2/3+b*1/3))):
        g(x)=expr(x=x)
        p1=plot(g(x),(x,a,b))
        p2=line([(x0,ymin),(x0,g(x0))])
        show(p1+p2+text(str((x0.n(digits=5),g(x0).n(digits=6))),(b,g(x0))))
```

```
Interactive function <function _ at 0x7fb1623e5310> with 3 widgets
  a: EvalText(value='0.0000', description='…
```

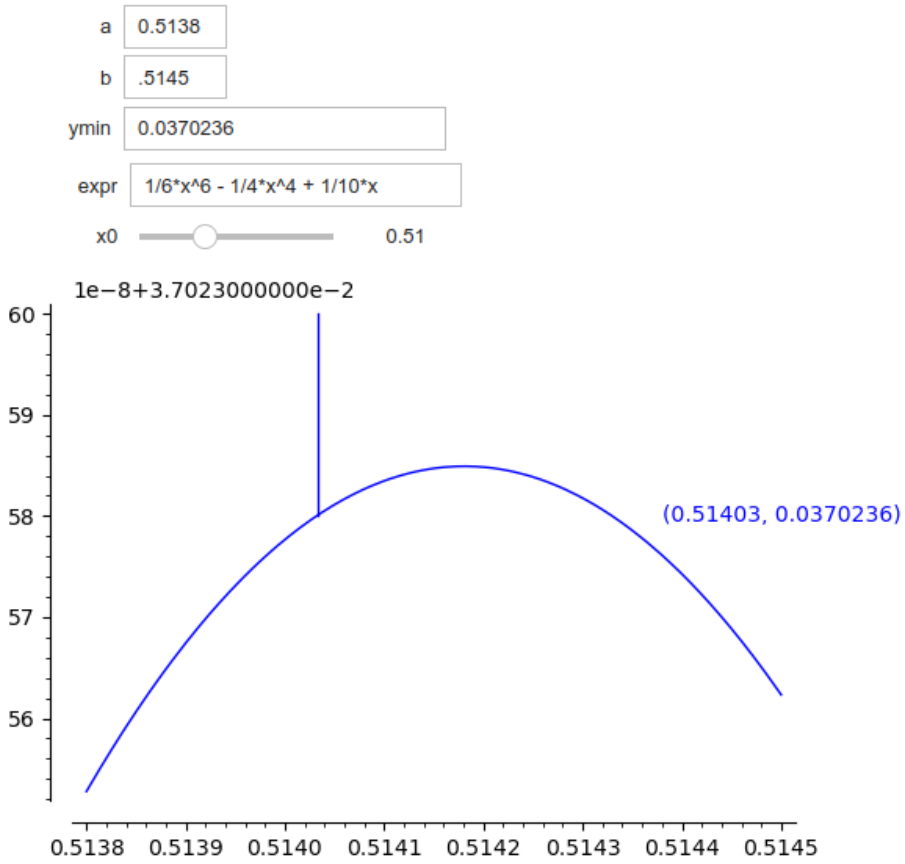
Réponse :
D'abord les points d'intérêt du graphe de g sur $[0,1]$ lu sur le dessin :

In [16]:
```
g(x)=1/6*x^6-1/4*x^4+1/10*x
ymin=0;x0=0.51;x1=0.93
show(plot(g(x),(x,0,1))+line([(x0,ymin),(x0,g(x0))])+line([(x1,ymin),(x1,g(x1))]))
```



Les points d'intérêt sont les bornes de l'intervalle d'étude 0 et 1 et les points de changement de variation de g : 0.51 et 0.93 à peu près.
Sur le dessin le maximum est atteint en le premier point de changement de variation ≈ 0.51.

Voici ce qu'on obtient en ajustant progressivement a et b à la main. Il apparaît que le maximum est atteint en un x entre 0.5138 et 0.5145 dont 0.5138 est une valeur à 10^{-3} près du x où g atteint son maximum.



1.2. Calcul des valeurs numériques des points d'intérêt

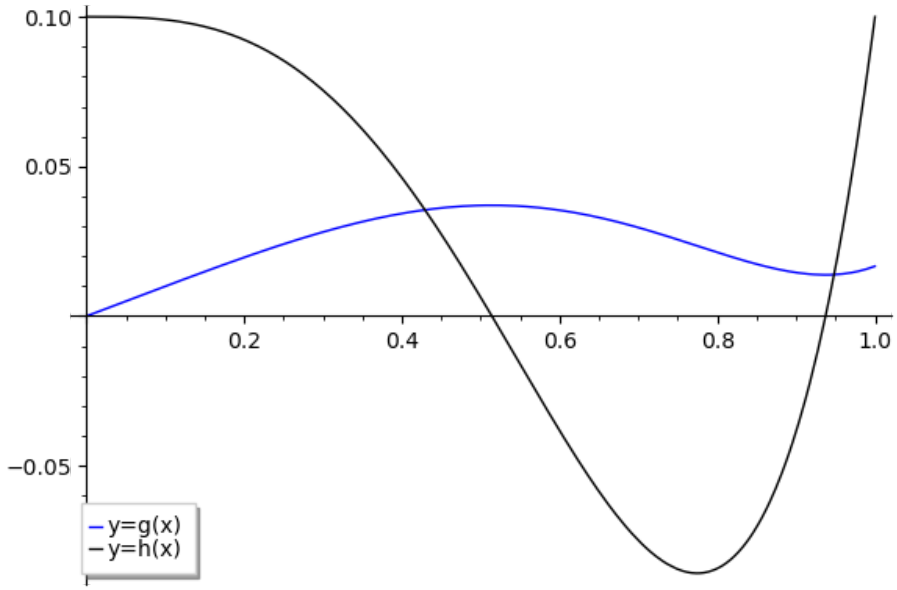
✍ Calculer avec Sagemath les valeurs numériques des points où la fonction $h(x) = g'(x) = x^5 - x^3 + 1/10$ s'annule. Comparer avec le graphe de h et les points d'intérêt du graphe de g.

✍ Tracer le graphe de g sur l'intervalle $[-1.4\,,\,1.2]$ et comparer les points d'intérêt lus sur le dessin avec ceux trouvés numériquement ci-dessus.

Réponse :

In [5]:
```
g(x)=1/6*x^6-1/4*x^4+1/10*x
h(x)=x^5-x^3+1/10
sol=solve(h(x)==0,x,to_poly_solve='force')
for i in range(len(sol)):print(str(i)+'. ',sol[i].rhs().n(digits=3))
show(plot(g(x),(x,0,1),legend_label="y=g(x)")+plot(h(x),(x,0,1),color='black',legend_label="y=h(x)"))
```

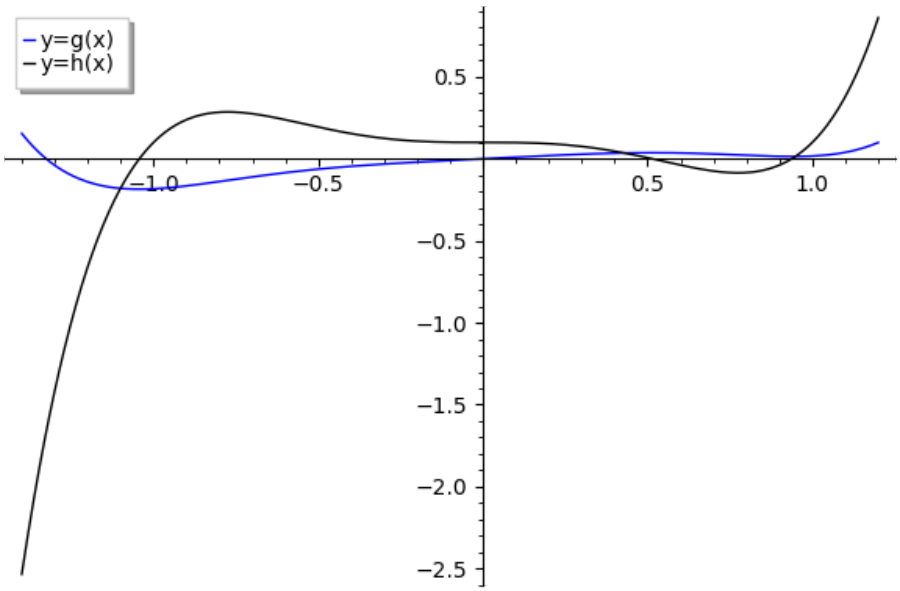
```
0.  -1.04
1.  0.514
2.  0.937
3.  -0.204 - 0.397*I
4.  -0.204 + 0.397*I
```



Les solutions 3 et 4 font apparaître des nombres complexes, hors sujet. La solution 0 n'est pas dans l'intervalle $[0,1]$. Restent les solutions 1 et 2, qui correspondent aux points où le graphe de h coupe l'axe des x sur le dessin et aux points de changement de variation de g.

Graphes de g et h sur $[-1.4\,,\,1.2]$:

In [6]:
```
show(plot(g(x),(x,-1.4,1.2),legend_label="y=g(x)")+plot(h(x),(x,-1.4,1.2),color='black',legend_label="y=h(x)")
```



On voit cette fois sur le dessin le point d'annulation $x \approx -1.04$ de h.
Sur l'intervalle $[-1.4\,,\,1.2]$ le maximum apparaît maintenant en l'extrémité 1.2 de l'intervalle.

1.3. Maximisation de $g(x) = \frac{1}{6}x^6 - \frac{1}{4}x^4 + \frac{1}{10}x$ sous la contrainte $x \geq a$ puis $x \in [a,b]$ avec les multiplicateurs de Lagrange (conditions KKT)

Si $g : [a,+\infty[\to \mathbb{R}$ est maximal en x alors nécessairement le gradient de g en x (c'est à dire la dérivée de g en x) est nul ou bien $x = a$ et le gradient pointe vers l'extérieur de $[a,+\infty[$ (c'est à dire est négatif).

On réécrit la contrainte sous la forme $a - x \leq 0$ puis on forme le Lagrangien $L(x,k) = g(x) - k \times (a-x)$ et son gradient par rapport à x : $\nabla_x L(x,k) = \frac{\partial}{\partial x}(g(x) - k \times (a-x))$. Les points d'intérêt sont les (x,k) satisfaisant les conditions KKT, c'est à dire annulant en même temps $\nabla_x L$ et $k \times (a-x)$. On retient ceux pour lesquels $k \geq 0$. Si on en trouve plusieurs, on compare la valeur de g en les x trouvés (pourvu qu'ils soient en nombre fini). Si on sait que g atteint son maximum sur le domaine $[a,+\infty[$ alors les x trouvés pour lesquels g est maximal sont des maximums de g.

✍️ Refaire les calculs et la discussion pour les contraintes simultanée $(x \geq 0 \text{ et } x \leq 1)$. Cette fois g atteint un maximum sous les

contraintes $x \geq 0$ et $x \leq 1$: g est continue et le domaine $[0,1]$ est fermé et borné ("compact"). On a cependant également besoin de croire que Sagemath rend tous les points d'intérêt sans omission pour conclure la discussion.

✍ Comment adapte t-on la discussion pour la recherche du minimum de g sous contraintes $x \in [a,b]$?

✍ Former le Lagrangien et déterminer les points d'intérêt (les x satisfaisant les conditions KKT) pour les problèmes de maximisation et de minimisation de g sur l'intervalle $[0.8\,,\,1.2]$

Réponse :

In [17]:
```
a=0;b=1
g(x)=1/6*x^6-1/4*x^4+1/10*x
var('k,l')
s=solve([diff(g(x)-k*(a-x)-l*(x-b),x)==0,k*(a-x)==0,l*(b-x)==0],[x,k,l])
for i in range(len(s)):
    print(i+1,s[i])
```

```
1 [x == -1.043122035360069, k == 0.0, l == 0.0]
2 [x == 0.9373188405797102, k == 0.0, l == 0.0]
3 [x == (-0.2041887771764046 - 0.3965088043695821*I), k == 0.0, l == 0.0]
4 [x == (-0.2041887771764046 + 0.3965088043695821*I), k == 0.0, l == 0.0]
5 [x == 0.5141808475141808, k == 0.0, l == 0.0]
6 [x == 0, k == (-1/10), l == 0]
7 [x == 1, k == 0, l == (1/10)]
```

Les solutions 3 et 4 sont hors sujet.
La solution 1 n'est pas dans l'intervalle $[0,1]$.
Les solutions 2 et 5 sont des points d'annulation du gradient (de la dérivée) à l'intérieur du domaine sous contrainte.
La solution 6 est le bord inférieur du domaine sous contrainte, disqualifiée car $k < 0$.
La solution 7 est le bord supérieur du domaine ; elle est qualifiée : $k, l \geq 0$.
Il y a donc trois candidats qualifiés : ceux des solutions 2, 5 et 7. On les distingue en calculant g en ces points :

In [18]:
```
for i in [2,5,7]:
    x=s[i-1][0].rhs() # Attention à l'indexation !
    print(str(i)+'. x=',x.n(digits=3),'⇝ g(x)=',g(x).n(digits=3))
```

```
2. x= 0.937 ⇝ g(x)= 0.0138
5. x= 0.514 ⇝ g(x)= 0.0370
7. x= 1.00 ⇝ g(x)= 0.0167
```

On trouve que g est maximal en $x \approx 0.514$ (solution 5).

Si on cherchait le minimum de g, les candidats seraient les solutions avec x dans le domaine sous contrainte et $k, l \leq 0$ cette fois, donc de nouveau les solutions 2, 5 plus la solution 6.
A nouveau on distingue ces candidats en calculant la valeur de g.

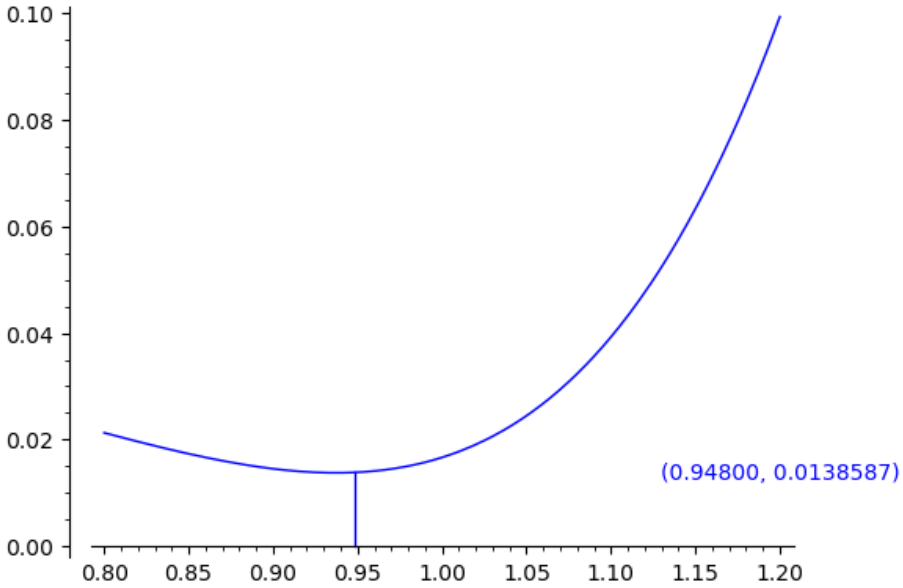
Sur l'intervalle $[0.8\,,\,1.2]$ maintenant :

In [19]:
```
a=0.8;b=1.2
g(x)=1/6*x^6-1/4*x^4+1/10*x
var('k,l')
s=solve([diff(g(x)-k*(a-x)-l*(x-b),x)==0,k*(a-x)==0,l*(b-x)==0],[x,k,l])
for i in range(len(s)):
    print(i+1,s[i])
```

```
1 [x == -1.043122035360069, k == 0.0, l == 0.0]
2 [x == 0.9373188405797102, k == 0.0, l == 0.0]
3 [x == (-0.2041887771764046 - 0.3965088043695821*I), k == 0.0, l == 0.0]
4 [x == (-0.2041887771764046 + 0.3965088043695821*I), k == 0.0, l == 0.0]
5 [x == 0.5141808475141808, k == 0.0, l == 0.0]
6 [x == (4/5), k == (527/6250), l == 0]
7 [x == (6/5), k == 0, l == (5377/6250)]
```

Pour le maximum de g les candidats qualifiés sont les solutions 2, 6,7 qu'on distingue en évaluant g en ces points. Pour le minimum de g le seul candidat qualifié est la solution 2.

Ceci est cohérent avec ce qu'on voit sur le dessin :



In [21]:
```
for i in [2,6,7]:
    x=s[i-1][0].rhs() # Attention à l'indexation !
    print(str(i)+'. x=',x.n(digits=3),'⇝ g(x)=',g(x).n(digits=3))
```

```
2. x= 0.937 ⇝ g(x)= 0.0138
6. x= 0.800 ⇝ g(x)= 0.0213
7. x= 1.20 ⇝ g(x)= 0.0993
```

L2ande - corrigé du TP maximisation d'une fonction sous contraintes

9 mai 2022

Partie 2 : maximisation d'une fonction $\mathbb{R}^2 \to \mathbb{R}$ sur un domaine défini par un ensemble d'inégalité.

On considère la fonction d'utilité et les contraintes de l'exercice 1 de la feuille de TD 2 :

$U(x,y) = (x+2)(x+3y)$ sous les contraintes $x \geq 0$, $y \geq 0$, $5x + 3y \leq r$ (contrainte de budget, $r \geq 0$ paramètre fixé).

On veut déterminer le maximum de U sous les contraintes pour $r = 20$ et $r = 5$.

Alternativement On veut déterminer le minimum de la dépense $D(x,y) = 5x + 3y$ sous les contraintes $x \geq 0$, $y \geq 0$, $U(x,y) \geq u_0$ pour $u_0 = 10$ et $u_0 = 20$ (contrainte de satisfaction).

Méthode 1 : On distingue l'intérieur du domaine et son bord dont on donne un paramétrage

Les points d'intérêt sont les points à l'intérieur du domaine où le gradient de U s'annule et les points d'intérêt de la restriction de U au bord du domaine.

On définit U par `U(x,y)=(x+2)*(x+3*y)` ; on calcule son gradient par `(diff(U(x,y),x),diff(U(x,y),y))` qu'on affiche avec `print()` ; on cherche les (x,y) annulant les deux composantes du gradient par `solve([diff(U(x,y),x)==0,diff(U(x,y),y)==0],[x,y])`.
On dessine le graphe de U sur le domaine $[0,2] \times [0,2]$ (par exemple) par `show(plot3d(U,[0,1],[0,1]))` mais on peut faire mieux : voir plus bas.

Pour déterminer les points d'intérêt de la restriction de U au bord du domaine donné par les contraintes on a besoin d'un paramétrage de ce dernier. Le bord du domaine s'obtient en saturant l'une des contraintes, une par une ; il y a donc autant de morceaux que de contraintes.

Exemple avec la contrainte de budget pour $r = 5$:

In [1]:
```
r=5
var('x,y')
solve([5*x+3*y==r],[x,y])
```

Out[1]:
```
[[x == -3/5*r1 + 1, y == r1]]
```

On obtient ainsi pour le morceau donné par $5x + 3y = 5$ le paramétrage suivant : $(x(t), y(t)) = (1 - \frac{3}{5}t, t)$, d'où la restriction de U au morceau : $t \mapsto U(1 - \frac{3}{5}t, t)$ qu'on étudiera sous les contraintes $x(t) \geq 0$, $y(t) \geq 0$ comme dans la première partie du TP pour en chercher le maximum. On fait de même avec les autres contraintes saturées :

- $y = 0$ qu'on paramétrise par $(x(t), y(t)) = (t, 0)$ sous la contrainte de budget $5x(t) + 3y(t) \leq 5$,
- $x = 0$ qu'on paramétrise par $(x(t), y(t)) = (0, t)$ sous la contrainte de budget $5x(t) + 3y(t) \leq 5$.

✍ $r = 5$. Déterminer par cette méthode les points d'intérêt de U dans le domaine et sélectionner celui où U atteint son maximum.

Réponse :

In [6]:
```
var('x,y')
U(x,y)=(x+2)*(x+3*y)
print(solve([diff(U(x,y),x)==0,diff(U(x,y),y)==0],[x,y]))
```

```
[
[x == -2, y == (2/3)]
]
```

Le gradient s'annule, mais pas dans le domaine sous contraintes.
On étudie la retriction de U aux trois morceaux du bord.

Morceau 1. (contrainte de budget saturée) :

In [42]:
```
x(t)=-3*t/5+1
y(t)=t
solve([x(t)>=0,y(t)>=0],t)
```

Out[42]:
```
[[0 < t, t < (5/3)], [t == 0], [t == (5/3)]]
```

In [47]:
```
a,b=0,5/3
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= -3/25*(12*t + 5)*(t - 5)
∂U/∂t= -72/25*t + 33/5
Candidats :
0.  2.29 ⇝ U= 10.6
t= 0 ⇝ U= 3.00
t= 5/3 ⇝ U= 10.0
```

La dérivée de $t \mapsto U(x(t), y(t))$ ne s'annule pas en un t tel que $x(t), y(t) \geq 0$. Les seuls points d'intérêt sont donc les extrémités du domaine $[0\,,\,\frac{5}{3}]$ pour t, qu'on distingue en évaluant $U(x(t), y(t))$ en ces points. Le maximum sur ce premier morceau est atteint en $t = \frac{5}{3}$ soit $(x, y) = (0, \frac{5}{3})$ et U vaut 10 en ce point.

Morceau $y = 0$:

In [48]:
```
x(t)=t
y(t)=0
solve([x(t)>=0,5*x(t)+3*y(t)<=5],t)
```

Out[48]:
```
[[0 < t, t < 1], [t == 0], [t == 1]]
```

In [49]:
```
a,b=0,1
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= (t + 2)*t
∂U/∂t= 2*t + 2
Candidats :
0.  -1.00 ⇝ U= -1.00
t= 0 ⇝ U= 0.000
t= 1 ⇝ U= 3.00
```

Points d'intérêt $t = 0$ ou $t = 1$, maximum de U atteint en $t = 1$ mais la valeur est inférieur au max de u sur le premier morceau.

Morceau $x = 0$:

In [51]:
```
x(t)=0
y(t)=t
solve([y(t)>=0,5*x(t)+3*y(t)<=5],t)
```

Out[51]:
```
[[0 < t, t < (5/3)], [t == 0], [t == (5/3)]]
```

In [52]:
```
a,b=0,5/3
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= 6*t
∂U/∂t= 6
Candidats :
t= 0 ⇝ U= 0.000
t= 5/3 ⇝ U= 10.0
```

Points d'intérêt $t = 0$ et $t = 5/3$; le max de U est atteint en $t = 5/3$; on retrouve le point $(x, y) = (0, \frac{5}{3})$ du premier morceau.

Conclusion : pour $r = 5$, U atteint son maximum dans le domaine en $(x, y) = (0, \frac{5}{3})$

✍️ Faites de même pour $r = 20$.

Réponse :

Le gradient de U ne s'annule toujours pas dans le domaine.
Le bord du domaine change. Morceau 1. :

In [54]:
```
r=20
var('x,y')
solve([5*x+3*y==r],[x,y])
```

Out[54]:
```
[[x == -3/5*r3 + 4, y == r3]]
```

In [8]:
```
x(t)=-3*t/5+4
y(t)=t
solve([x(t)>=0,y(t)>=0],t)
```

Out[8]:
```
[[0 < t, t < (20/3)], [t == 0], [t == (20/3)]]
```

In [9]:
```
a,b=0,20/3
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= -12/25*(3*t + 5)*(t - 10)
∂U/∂t= -72/25*t + 12
Candidats :
0.  4.17 ⇝ U= 49.0
t= 0 ⇝ U= 24.0
t= 20/3 ⇝ U= 40.0
```

On trouve cette fois un maximum en $t \approx 4.17$ qui correspond à :

In [59]:
```
t=sol[0].rhs()
print('x,y=',x(t),',',y(t))
```

```
x,y= 3/2 , 25/6
```

Il faut encore étudier la restriction de U aux morceaux $x = 0$ et $y = 0$ pour conclure.

In [12]:
```
x(t)=t
y(t)=0
solve([x(t)>=0,5*x(t)+3*y(t)<=20],t)
```

Out[12]:
```
[[0 < t, t < 4], [t == 0], [t == 4]]
```

In [18]:
```
print(U)
```

```
(x, y) |--> (x + 3*y)*(x + 2)
```

In [14]:
```
a,b=0,4
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= (t + 2)*t
∂U/∂t= 2*t + 2
Candidats :
0.  -1.00 ⇝ U= -1.00
t= 0 ⇝ U= 0.000
t= 4 ⇝ U= 24.0
```

In [15]:
```
x(t)=0
y(t)=t
solve([y(t)>=0,5*x(t)+3*y(t)<=20],t)
```

Out[15]:
```
[[0 < t, t < (20/3)], [t == 0], [t == (20/3)]]
```

In [16]:
```
a,b=0,20/3
f(t)=U(x(t),y(t));print('U(x(t),y(t))=',f(t))
h(t)=diff(f(t),t);print('∂U/∂t=',h(t))
sol=solve(h==0,t,to_poly_solve='force')
print('Candidats :')
for i in range(len(sol)):
    t=sol[i].rhs()
    print(str(i)+'. ',t.n(digits=3),'⇝ U=',f(t).n(digits=3))
for t in [a,b]:
    print('t=',t,'⇝ U=',f(t).n(digits=3))
```

```
U(x(t),y(t))= 6*t
∂U/∂t= 6
Candidats :
t= 0 ⇝ U= 0.000
t= 20/3 ⇝ U= 40.0
```

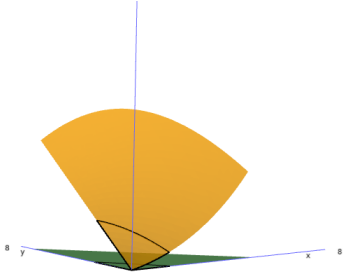
Conclusion : le maximum de U sur les morceaux $y=0$ et $x=0$ n'atteint pas le maximum de U sur le morceau 1. Le maximum de U sur le domaine est donc 49 atteint en le point $(x,y)=(\frac{3}{2},\frac{25}{6})$ situé sur la courbe de budget $5x+3y=r=20$.

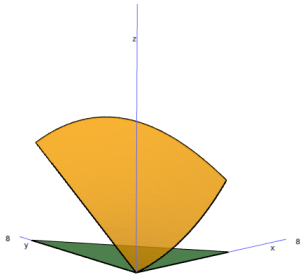
✍️ On représente ci-dessous pour $r=20$ le graphe de U au dessus du domaine, le bord du domaine et le graphe (en noir) de la restriction de U au bord.
Retrouvez vous sur le dessin les points d'intérêt ? Le point où U atteint son maximum ?

In [20]:
```
#https://ask.sagemath.org/question/10555/plotting-regions-in-3d/
var('z');r=20;c(x,y)=5*x+3*y;U(x,y)=(x+2)*(x+3*y);u=40
xmin,xmax=(0,8)
ymin,ymax=(0,8)
zmin,zmax=(0,80)

A=line3d([(xmin,0,0),(xmax,0,0),(xmin,0,0)])+line3d([(0,ymin,0),(0,ymax,0),(0,ymin,0)])+line3d([(0,0,zmin),(
T=text3d('x', (xmax*3/4, -.5, -.5))+text3d('y', (-.5, ymax*3/4, -.5))+text3d('z', (-.5, -.5,zmax*3/4))+text3
#B=implicit_plot3d(c==r,(x,0,r/5),(y,0,r/3),(z,0,zmax),color='green', opacity=0.6)#plan c==r en vert
#B=B.add_condition(lambda x,y,z: z<= U(x,y))
P = implicit_plot3d(z==U(x,y),(x,xmin,xmax),(y,ymin,ymax),(z,zmin,zmax),color='orange', opacity=.8)#graphe d
P=P.add_condition(lambda x,y,z: c(x,y)<=r)
#I=implicit_plot3d(z==u,(x,0,xmax),(y,0,ymax),(z,0,zmax),color='lightblue', opacity=0.5)+text3d('z='+str(u),
domaine = implicit_plot3d(z==0, (x, 0, r/5), (y, 0, r/3), (z, zmin-.1, zmax),color='lightgreen')
domaine = domaine.add_condition(lambda x,y,z: 5*x+3*y<= r)

@interact
def _(r=input_box(20, width=8),b=slider([0..2],default=2)):
    G=P+A+T+domaine
    L1=line([(0,0),(r/5,0)], thickness=2,color='black')
    G1=parametric_plot3d([x,0,U(x,0)],(0,r/5), thickness=2,color='black')
    L2=line([(0,0),(0,r/3)], thickness=2,color='black')
    G2=parametric_plot3d([0,y,U(0,y)],(0,r/3), thickness=2,color='black')
    L3=line([(0,r/3),(r/5,0)], thickness=2,color='black')
    G3=parametric_plot3d([x,(r-5*x)/3,U(x,(r-5*x)/3)],(0,r/5), thickness=2,color='black')
    if b>0:G+=L1+L2+L3
    if b>1:G+=G1+G2+G3
    show(G,frame=false,aspect_ratio=[1,1,.1],viewpoint=[[-0.9927,0.086,0.0841],91.71])#,viewer='threejs'
```

```
Interactive function <function _ at 0x7f56b4b538b0> with 2 widgets
  r: EvalText(value='20', description='r', …
```

Le dessin des graphes de U (en orange) et de la restriction de U au bord (courbe en noir) pour $r=5$ et $r=20$ confirme les conclusions ci-dessus : le maximum est atteint en l'extrémité $x=0$ de la courbe de budget $5x+3y=r$ pour $r=5$, à l'intérieur de cette courbe pour $r=20$.





Méthode 2 via les multiplicateurs de Lagrange et les conditions KKT

On réécrit les contraintes : $c_i(x,y)\leq 0$ pour $i=1,2,3$, où $c_1(x,y)=-x$, $c_2(x,y)=-y$ et $c_3(x,y)=5x+3y-r$.

Le Lagrangien s'écrit $L(x,y,k,l,m)=U(x,y)-k\times c_1(x,y)-l\times c_2(x,y)-m\times c_3(x,y)$.

Les points d'intérêt sont les (x,y,k,l,m) annulant à la fois le gradiant $\nabla_{(x,y)}L=\left(\frac{\partial}{\partial x}L(x,y,k,l,m),\ \frac{\partial}{\partial y}L(x,y,k,l,m)\right)$ et $k\times c_1(x,y),\ l\times c_2(x,y),\ m\times c_3(x,y)$.
On retient ceux pour lesquels les contraintes sont satisfaites ($c_i(x,y)\leq 0$) et $k,l,m\geq 0$.

Rq : La convention habituelle pour le Lagrangien est d'écrire
$L(x,y,k,l,m) = -U(x,y) + k \times c_1(x,y) + l \times c_2(x,y) + m \times c_3(x,y)$ ce qui ne change pas la discussion.

Si U est continue et le domaine déterminé par les contraintes $c_i(x,y) \leq 0$ est fermé borné alors U atteint son maximum sous contrainte. Si aucun des points d'intérêt n'est omis par Sagemath, on peut mener la discussion jusqu'à sa conclusion.

✍ Quels seraient les points d'intérêt si on cherchait le minimum plutôt que le maximum ?

Réponse : les solutions pour lesquels les contraintes sont staisfaites et $k, l, m \leq 0$.

✍ Faire les calculs et la discussion pour la fonction d'utilité U et les contraintes rappelées ci-dessus pour $r = 20$ puis pour $r = 5$. Retrouvez vous les points d'intérêt de la première méthode ?

Réponse : On détermine les candidats en résolvant les conditions KKT. Pour chaque solution on écrit x, y, k, l, m, la valeur de la contrainte de budget et U.

In [36]:
```
for r in [5,20]:
    print('r=',r)
    var('x,y,k,l,m')
    U(x,y)=(x+2)*(x+3*y)
    c(x,y)=5*x+3*y-r
    sol=solve([diff(U(x,y)-k*(-x)-l*(-y)-m*c(x,y),x)==0,diff(U(x,y)-k*(-x)-l*(-y)-m*c(x,y),y)==0,k*(-x)==0,l
    ns=1
    print('no. x y k l m c(x,y) ⇝ U(x,y) :')
    for s in sol:
        x,y,k,l,m=(s[i].rhs().n(digits=2) for i in range(5))
        print(str(ns)+'. ',x,y,k,l,m,c(x,y),'⇝ U=',U(x,y))
        ns=ns+1
    print()
```

```
r= 5
no. x y k l m c(x,y) ⇝ U(x,y) :
1.  -2.0 0.67 0.00 0.00 0.00 -13. ⇝ U= 0.00
2.  -1.0 0.00 0.00 -3.0 0.00 -10. ⇝ U= -1.0
3.  0.00 0.00 -2.0 -6.0 0.00 -5 ⇝ U= 0
4.  -0.38 2.3 0.00 0.00 1.6 0.00 ⇝ U= 11.
5.  0.00 1.7 3.0 0.00 2.0 0.00 ⇝ U= 10.
6.  1.0 0.00 0.00 -6.6 0.80 0.00 ⇝ U= 3.0

r= 20
no. x y k l m c(x,y) ⇝ U(x,y) :
1.  -2.0 0.67 0.00 0.00 0.00 -28. ⇝ U= 0.00
2.  -1.0 0.00 0.00 -3.0 0.00 -25. ⇝ U= -1.0
3.  0.00 0.00 -2.0 -6.0 0.00 -20 ⇝ U= 0
4.  1.5 4.2 0.00 0.00 3.5 0.00 ⇝ U= 49.
5.  0.00 6.7 -12. 0.00 2.0 0.00 ⇝ U= 40.
6.  4.0 0.00 0.00 -12. 2.0 0.00 ⇝ U= 24.
```

Pour $r = 5$ les solutions 1,2,4 sont hors domaine. Le seul candidat qualifié ($k, l, m \geq 0$ et contraintes sur x, y satisfaites) est la solution 5. Deux contraintes sont alors saturées : $x = 0$ et la contrainte de budget, ce qui correspond à $k, m \neq 0$. On retrouve la solution de la méthode 1 (ouf !).

Pour $r = 20$ le seul candidat qualifié est la solution 4.

✍ Chercher les (x, y) minimisant la dépense $5x + 3y$ sous les contraintes $x \geq 0$, $y \geq 0$, $U(x,y) \geq u_0$ pour $u_0 = 10$ et $u_0 = 20$. Cf la discussion sur Slack pour l'ex.2 de la feuille de TD 2.

In [ ]:
```
```

Détermination de $\mathbb{Z}/16\mathbb{Z}^\times$

8-10, 13 mars 2023

1. Ad hoc pour $\mathbb{Z}/16\mathbb{Z}^\times$

https://sagecell.sagemath...

```
print((3^3)%16, mod(3^3,16),3^(-1)%16,mod(3^(-1),16))
l=[[a,b,c,d] for a in range(2) for b in range(4) for c in range(4)\
   for d in range(2) if ((-1)^a*3^b*5^c*7^d)%16==1]
print(l,"\n")
c=[[2,0,0,0],[0,4,0,0],[0,0,4,0],[0,0,0,2]]
#c=([C.list() for C in diagonal_matrix([2,4,4,2]).columns()])
#c=([C.list() for C in 8*identity_matrix(4)).columns()])
M=column_matrix(ZZ,l+c);show(M);show(" ")
D,P,Q=M.smith_form() # A.smith_form?
show(D,P^(-1));show(" ")
a,b,c,d=(P^(-1)).column(3)
print(mod((-1)^a*3^(b)*5^c*7^d,16))
```

```
11 11 11 11
[[0, 0, 0, 0], [0, 1, 3, 1], [0, 2, 2, 0], [0, 3, 1, 1], [1, 0, 2, 1], [1, 1, 1, 0], [1, 2, 0, 1], [1, 3, 3, 0]]
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 4 & 0 & 0 \\ 0 & 3 & 2 & 1 & 2 & 1 & 0 & 3 & 0 & 0 & 4 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
5
```

Help | Powered by SageMath

2. $\mathbb{Z}/80\mathbb{Z}^\times$

Liste de générateurs avec leur ordre, on enlève les multiples.
Puis construction de la liste des familles de coefficients modulo l'ordre des générateurs, mais pb de mémoire : `log(prod(dict[i] for i in dict),10).n(digits=3)` $\approx 10^{8.4}$. Fonctionne bien pour $\mathbb{Z}/p\mathbb{Z}^\times$ avec p premier ($\rightsquigarrow$ un seul générateur).

Prendre tous les éléments du groupe et retenir toutes les familles de coefficients modulo l'ordre du groupe donnerait $32^{32} \approx 10^{48}$ familles de taille 32.

https://sagecell.sagemath...

```
n=80 #n=31
#l=range(n);print(l) #crible
#l=[i for i in l if i%2!=0];print(l)
#l=[i for i in l if i%5!=0];print(l)
#print(len(l))
#print()
l=[mod(i,n) for i in range(n) if gcd(i,n)==1];print(len(l),l)
ll=[];dict={mod(1,n):1};l.remove(mod(1,n))
while l!=[] and l[0] not in ll:
    a=l[0];k=1;aa=a;del l[0]
    while aa!=mod(1,n):
        aa*=a;k+=1;
        if aa in l:l.remove(aa)
        if aa in ll:ll.remove(aa)
    dict[a]=k
    ll.append(a)
print(ll)
```

```
print(dict)
print(dict[ll[0]])

def f(m):
    return(prod(ll[i]^m[i] for i in range(len(m))))

lm=[[]]
for i in ll:
    print(i)
    lm=[m+[j] for j in range(dict[i]) for m in lm]
lm=[m for m in lm if f(m)==mod(1,n)]
print(len(lm),lm)
```

```
32 [1, 3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39, 41, 43, 47, 49, 51, 53, 57, 59, 61,
16 [3, 7, 11, 13, 17, 19, 21, 29, 31, 39, 43, 47, 53, 57, 71, 79]
{1: 1, 3: 4, 7: 4, 11: 4, 13: 4, 17: 4, 19: 4, 21: 4, 29: 4, 31: 2, 39: 2, 43: 4, 47: 4, 53: 4, 57:
3 1
7 4
11 16
13 64
17 256
19 1024
21 4096
29 16384
31 65536
39 131072
43 262144
47 1048576
53 4194304
57 16777216
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
MemoryError:

During handling of the above exception, another exception occurred:

MemoryError                               Traceback (most recent call last)
Input In [1], in <cell line: 25>()
     25 for i in ll:
     26     print(i,len(lm))
---> 27     lm=[m+[j] for j in range(dict[i]) for m in lm]
     28 lm=[m for m in lm if f(m)==mod(Integer(1),n)]
     29 print(len(lm),lm)

MemoryError:
```

Help | Powered by SageMath

3. liste plus petite de générateurs puis de relations

SageMathCell

```
n=80 #n=31
l0=[mod(i,n) for i in range(2,n) if gcd(i,n)==1]
l=l0;a=mod(1,n);ll=[a];l1=[a]
print("a:",a)
print("l:",len(l),l)
print("ll:",len(ll),ll)
print("l1:",len(l1),l1)

while l!=[]:
    a=l[0]
    l10=[i for i in l1]
    for b in l10:
        c=a*b;
        if c not in l1:l1+=[c]
        if c in l:l.remove(c)
    ll.append(a)
    print()
    print("a:",a)
    print("l:",len(l),l)
    print("ll:",len(ll),ll)
    print("l1:",len(l1),l1)
```

```
a: 1
l: 31 [3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39, 41, 43, 47, 49, 51, 53, 57, 59, 61, 63, 67, 69, 71, 73, 77, 79]
ll: 1 [1]
l1: 1 [1]

a: 3
l: 30 [7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39, 41, 43, 47, 49, 51, 53, 57, 59, 61, 63, 67, 69, 71, 73, 77, 79]
ll: 2 [1, 3]
l1: 2 [1, 3]

a: 7
l: 28 [9, 11, 13, 17, 19, 23, 27, 29, 31, 33, 37, 39, 41, 43, 47, 49, 51, 53, 57, 59, 61, 63, 67, 69, 71, 73, 77, 79]
ll: 3 [1, 3, 7]
l1: 4 [1, 3, 7, 21]

a: 9
l: 24 [11, 13, 17, 19, 23, 31, 33, 37, 39, 41, 43, 47, 49, 51, 53, 57, 59, 61, 67, 69, 71, 73, 77, 79]
ll: 4 [1, 3, 7, 9]
l1: 8 [1, 3, 7, 21, 9, 27, 63, 29]

a: 11
l: 16 [13, 17, 23, 31, 37, 39, 41, 43, 47, 49, 51, 59, 61, 67, 69, 73]
ll: 5 [1, 3, 7, 9, 11]
l1: 16 [1, 3, 7, 21, 9, 27, 63, 29, 11, 33, 77, 71, 19, 57, 53, 79]

a: 13
l: 8 [17, 23, 47, 51, 59, 61, 69, 73]
ll: 6 [1, 3, 7, 9, 11, 13]
l1: 24 [1, 3, 7, 21, 9, 27, 63, 29, 11, 33, 77, 71, 19, 57, 53, 79, 13, 39, 37, 31, 41, 43, 49, 67]

a: 17
l: 0 []
ll: 7 [1, 3, 7, 9, 11, 13, 17]
l1: 32 [1, 3, 7, 21, 9, 27, 63, 29, 11, 33, 77, 71, 19, 57, 53, 79, 13, 39, 37, 31, 41, 43, 49, 67, 17, 51, 73, 59, 61, 23, 69, 47]
```

Help | Powered by SageMath

Avec dictionnaires l1 et rel des relations

Ajout d'un élement, concaténation (merge), test d'égalité, deep copy : voir documentation Python dictionnary

SageMathCell

```
n=31 #n=31
l0=[mod(i,n) for i in range(2,n) if gcd(i,n)==1]
l=l0;a=mod(1,n);ll=[a];l1={a:{}};rel=[{a:1}]
print("a:",a)
print("l:",len(l),l)
print("l1:",len(l1),l1)
print("ll:",len(ll),ll)
print("rel:",len(rel),rel)
print()
while l!=[]:# and l[0] not in ll:
    a=l[0];ll.append(a);print("a:",a)
    k=0
    while k<len(l1):
        b=list(l1)[k]
        c=a*b;
        if c not in l1:
            if a in l1[b]:
                l1[c]={**l1[b]};l1[c][a]+=1
                print(a,"*",b,"=",c,",",c,"not in l1 et",a,"in l1[",b,"] ⇝ l1[",c,"]:",l1[c])
            else:
                l1[c]={**l1[b],**{a:1}}
                print(a,"*",b,"=",c,",",c,"not in l1 et",a,"not in l1[",b,"] ⇝ l1[",c,"]:",l1[c])
        else:
            relab={**l1[b]}
            if a in relab:relab[a]+=1
            else:relab[a]=1
            rel.append({**{i:relab[i] for i in relab if i not in l1[c]},\
                 **{i:-l1[c][i] for i in l1[c] if i not in relab},\
                 **{i:relab[i]-l1[c][i] for i in relab if i in l1[c]}})
            print(a,"*",b,"=",c,"et",c,"in l1 ⇝ rel+=",rel[-1])
        if c in l:l.remove(c)
        k+=1
    print("l:",len(l),l)
    print("l1:",len(l1),l1)
    print("ll:",len(ll),ll)
    print("rel:",len(rel),rel)
    print()

del ll[0],rel[0] #1 enlevé des générateurs et relations
dictll={ll[i]:i for i in range(len(ll))}
M=matrix(ZZ,len(ll),len(rel),{(dictll[i],j):rel[j][i] for j in range(len(rel)) for i in rel[j]})
show("M:",M)
s=M.smith_form()
show(s[0][0:len(ll),0:len(ll)])
```

```
a: 1
l: 29 [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
l1: 1 {1: {}}
ll: 1 [1]
rel: 1 [{1: 1}]

a: 2
2 * 1 = 2 , 2 not in l1 et 2 not in l1[ 1 ] ↝ l1[ 2 ]: {2: 1}
2 * 2 = 4 , 4 not in l1 et 2 in l1[ 2 ] ↝ l1[ 4 ]: {2: 2}
2 * 4 = 8 , 8 not in l1 et 2 in l1[ 4 ] ↝ l1[ 8 ]: {2: 3}
2 * 8 = 16 , 16 not in l1 et 2 in l1[ 8 ] ↝ l1[ 16 ]: {2: 4}
2 * 16 = 1 et 1 in l1 ↝ rel+= {2: 5}
l: 25 [3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
l1: 5 {1: {}, 2: {2: 1}, 4: {2: 2}, 8: {2: 3}, 16: {2: 4}}
ll: 2 [1, 2]
rel: 2 [{1: 1}, {2: 5}]

a: 3
3 * 1 = 3 , 3 not in l1 et 3 not in l1[ 1 ] ↝ l1[ 3 ]: {3: 1}
3 * 2 = 6 , 6 not in l1 et 3 not in l1[ 2 ] ↝ l1[ 6 ]: {2: 1, 3: 1}
3 * 4 = 12 , 12 not in l1 et 3 not in l1[ 4 ] ↝ l1[ 12 ]: {2: 2, 3: 1}
3 * 8 = 24 , 24 not in l1 et 3 not in l1[ 8 ] ↝ l1[ 24 ]: {2: 3, 3: 1}
3 * 16 = 17 , 17 not in l1 et 3 not in l1[ 16 ] ↝ l1[ 17 ]: {2: 4, 3: 1}
3 * 3 = 9 , 9 not in l1 et 3 in l1[ 3 ] ↝ l1[ 9 ]: {3: 2}
3 * 6 = 18 , 18 not in l1 et 3 in l1[ 6 ] ↝ l1[ 18 ]: {2: 1, 3: 2}
3 * 12 = 5 , 5 not in l1 et 3 in l1[ 12 ] ↝ l1[ 5 ]: {2: 2, 3: 2}
3 * 24 = 10 , 10 not in l1 et 3 in l1[ 24 ] ↝ l1[ 10 ]: {2: 3, 3: 2}
3 * 17 = 20 , 20 not in l1 et 3 in l1[ 17 ] ↝ l1[ 20 ]: {2: 4, 3: 2}
3 * 9 = 27 , 27 not in l1 et 3 in l1[ 9 ] ↝ l1[ 27 ]: {3: 3}
3 * 18 = 23 , 23 not in l1 et 3 in l1[ 18 ] ↝ l1[ 23 ]: {2: 1, 3: 3}
3 * 5 = 15 , 15 not in l1 et 3 in l1[ 5 ] ↝ l1[ 15 ]: {2: 2, 3: 3}
3 * 10 = 30 , 30 not in l1 et 3 in l1[ 10 ] ↝ l1[ 30 ]: {2: 3, 3: 3}
3 * 20 = 29 , 29 not in l1 et 3 in l1[ 20 ] ↝ l1[ 29 ]: {2: 4, 3: 3}
3 * 27 = 19 , 19 not in l1 et 3 in l1[ 27 ] ↝ l1[ 19 ]: {3: 4}
3 * 23 = 7 , 7 not in l1 et 3 in l1[ 23 ] ↝ l1[ 7 ]: {2: 1, 3: 4}
3 * 15 = 14 , 14 not in l1 et 3 in l1[ 15 ] ↝ l1[ 14 ]: {2: 2, 3: 4}
3 * 30 = 28 , 28 not in l1 et 3 in l1[ 30 ] ↝ l1[ 28 ]: {2: 3, 3: 4}
3 * 29 = 25 , 25 not in l1 et 3 in l1[ 29 ] ↝ l1[ 25 ]: {2: 4, 3: 4}
3 * 19 = 26 , 26 not in l1 et 3 in l1[ 19 ] ↝ l1[ 26 ]: {3: 5}
3 * 7 = 21 , 21 not in l1 et 3 in l1[ 7 ] ↝ l1[ 21 ]: {2: 1, 3: 5}
3 * 14 = 11 , 11 not in l1 et 3 in l1[ 14 ] ↝ l1[ 11 ]: {2: 2, 3: 5}
3 * 28 = 22 , 22 not in l1 et 3 in l1[ 28 ] ↝ l1[ 22 ]: {2: 3, 3: 5}
3 * 25 = 13 , 13 not in l1 et 3 in l1[ 25 ] ↝ l1[ 13 ]: {2: 4, 3: 5}
3 * 26 = 16 et 16 in l1 ↝ rel+= {3: 6, 2: -4}
3 * 21 = 1 et 1 in l1 ↝ rel+= {2: 1, 3: 6}
3 * 11 = 2 et 2 in l1 ↝ rel+= {3: 6, 2: 1}
3 * 22 = 4 et 4 in l1 ↝ rel+= {3: 6, 2: 1}
3 * 13 = 8 et 8 in l1 ↝ rel+= {3: 6, 2: 1}
l: 0 []
l1: 30 {1: {}, 2: {2: 1}, 4: {2: 2}, 8: {2: 3}, 16: {2: 4}, 3: {3: 1}, 6: {2: 1, 3: 1}, 12: {2: 2, 3: 1}, 24: {2: 3, 3
ll: 3 [1, 2, 3]
rel: 7 [{1: 1}, {2: 5}, {3: 6, 2: -4}, {2: 1, 3: 6}, {3: 6, 2: 1}, {3: 6, 2: 1}, {3: 6, 2: 1}]
```

$$\mathbf{M:}\begin{pmatrix}5 & -4 & 1 & 1 & 1 & 1\\ 0 & 6 & 6 & 6 & 6 & 6\end{pmatrix}$$

$$\begin{pmatrix}1 & 0\\ 0 & 30\end{pmatrix}$$

Help | Powered by SageMath

4. Script abouti

SageMathCell

```
n=80 #n=31
print("n:",n);print()
l=[mod(i,n) for i in range(2,n) if gcd(i,n)==1]
a=mod(1,n);ll=[a];l1={a:{}};rel=[{a:1}]
while l!=[]:
    a=l[0];ll.append(a);k=0
    while k<len(l1):
        b=list(l1)[k];c=a*b;
        if c not in l1:
            if a in l1[b]:l1[c]={**l1[b]};l1[c][a]+=1
            else:l1[c]={**l1[b],**{a:1}}
        else:
            relab={**l1[b]}
            if a in relab:relab[a]+=1
            else:relab[a]=1
            nrel={**{i:relab[i] for i in relab if i not in l1[c]},\
                 **{i:-l1[c][i] for i in l1[c] if i not in relab},\
                 **{i:relab[i]-l1[c][i] for i in relab if i in l1[c]}}
            nrel={i:nrel[i] for i in nrel if nrel[i]!=0}
```

```
            if nrel not in rel:rel.append(nrel) # and nrel!={}
        if c in l:l.remove(c)
        k+=1

del ll[0],rel[0] #1 enlevé des générateurs et relations
print("ll:",len(ll),ll)
print("rel:",len(rel),rel)
print()

dictll={ll[i]:i for i in range(len(ll))}
M=matrix(ZZ,len(ll),len(rel),{(dictll[i],j):rel[j][i] for j in range(len(rel)) for i in rel[j]})
show("M:",M)
s=M.smith_form()
show(s[0][0:len(ll),0:len(ll)])
```

```
n: 80

ll: 3 [3, 7, 11]
rel: 6 [{3: 4}, {7: 4}, {11: 2, 3: -2, 7: -2}, {3: 2, 11: 2, 7: -2}, {7: 2, 11: 2, 3: -2}, {3: 2, 7: 2, 11: 2}]
```

$$\mathbf{M:}\begin{pmatrix} 4 & 0 & -2 & 2 & -2 & 2 \\ 0 & 4 & -2 & -2 & 2 & 2 \\ 0 & 0 & 2 & 2 & 2 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

Help | Powered by SageMath

5. SageMathCell embeded

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>SageMathCell gp des unités</title>
    <script  type="text/javascript" src="https://sagecell.sagemath.org/static/embedded_sagecell.js"></script>
    <style type="text/css">.sagecell_output th, .sagecell_output td {border: none}</style>


<script>
sagecell.makeSagecell({inputLocation:  'div.mycella',
});
    // Make the div with id 'mycell' a Sage cell
    sagecell.makeSagecell({inputLocation:  'div.mycell',//inputLocation: '#mycell'
                           template:       sagecell.templates.minimal,
                           hide: ["evalButton"],
                           editor:"textarea",
                           autoeval: true,
                           });
    // Make *any* div with class 'compute' a Sage cell
    sagecell.makeSagecell({inputLocation: 'div.compute',
        autoeval:true,
        editor:"textarea",
        linked: true,
        });
</script>


  </head>
  <body>
  <h1>TP ℤ/nℤ<sup>x</sup></h1>
  <i>F-X. Dehon - 10 mars 2023</i>
  <br><br>

<!---
<div style="cursor: text; inset #aaa;border: 2px inset #AAA; font: -webkit-small-control; height: 1000px;
overflow: auto; padding: 2px; resize: both;" class="jop-noMdConv">
--->
<div class="compute"><script type="text/x-sage">
n=80 #n=31
print("n:",n);print()
l=[mod(i,n) for i in range(2,n) if gcd(i,n)==1]
a=mod(1,n);ll=[a];l1={a:{}};rel=[{a:1}]
while l!=[]:
    a=l[0];ll.append(a);k=0
    while k<len(l1):
        b=list(l1)[k];c=a*b;
        if c not in l1:
            if a in l1[b]:l1[c]={**l1[b]};l1[c][a]+=1
```

```
            else:l1[c]={**l1[b],**{a:1}}
        else:
            relab={**l1[b]}
            if a in relab:relab[a]+=1
            else:relab[a]=1
            nrel={**{i:relab[i] for i in relab if i not in l1[c]},\
                 **{i:-l1[c][i] for i in l1[c] if i not in relab},\
                 **{i:relab[i]-l1[c][i] for i in relab if i in l1[c]}}
            nrel={i:nrel[i] for i in nrel if nrel[i]!=0}
            if nrel not in rel:rel.append(nrel) # and nrel!={}
        if c in l:l.remove(c)
        k+=1

del ll[0],rel[0] #1 enlevé des générateurs et relations
print("ll:",len(ll),ll)
print("rel:",len(rel),rel)
print()

dictll={ll[i]:i for i in range(len(ll))}
M=matrix(ZZ,len(ll),len(rel),{(dictll[i],j):rel[j][i] for j in range(len(rel)) for i in rel[j]})
show("M:",M)
s=M.smith_form()
show(s[0][0:len(ll),0:len(ll)])
</script></div>
<!---</div>--->

<h4>Calculs liés</h4>
<div class="compute"><script type="text/x-sage">
banner()
show(s[1])
</script></div>

</body>
</html>
```

SageMathCell gp des unités.html

TP $\mathbb{Z}/n\mathbb{Z}^\times$

F-X. Dehon - 10 mars 2023

```
n=80 #n=31
print("n:",n);print()
l=[mod(i,n) for i in range(2,n) if gcd(i,n)==1]
a=mod(1,n);ll=[a];l1={a:{}};rel=[{a:1}]
while l!=[]:
    a=l[0];ll.append(a);k=0
    while k<len(l1):
        b=list(l1)[k];c=a*b;
        if c not in l1:
            if a in l1[b]:l1[c]={**l1[b]};l1[c][a]+=1
            else:l1[c]={**l1[b],**{a:1}}
        else:
            relab={**l1[b]}
            if a in relab:relab[a]+=1
            else:relab[a]=1
            nrel={**{i:relab[i] for i in relab if i not in l1[c]},\
                 **{i:-l1[c][i] for i in l1[c] if i not in relab},\
                 **{i:relab[i]-l1[c][i] for i in relab if i in l1[c]}}
            nrel={i:nrel[i] for i in nrel if nrel[i]!=0}
            if nrel not in rel:rel.append(nrel) # and nrel!={}
        if c in l:l.remove(c)
        k+=1

del ll[0],rel[0] #1 enlevé des générateurs et relations
print("ll:",len(ll),ll)
print("rel:",len(rel),rel)
print()

dictll={ll[i]:i for i in range(len(ll))}
M=matrix(ZZ,len(ll),len(rel),{(dictll[i],j):rel[j][i] for j in range(len(rel)) for i in rel[j]})
show("M:",M)
s=M.smith_form()
show(s[0][0:len(ll),0:len(ll)])
```

Evaluate

```
n: 80

ll: 3 [3, 7, 11]
rel: 6 [{3: 4}, {7: 4}, {11: 2, 3: -2, 7: -2}, {3: 2, 11: 2, 7: -2}, {7: 2, 11: 2, 3: -2}, {3: 2, 7: 2, 11: 2}]
```

$$M: \begin{pmatrix} 4 & 0 & -2 & 2 & -2 & 2 \\ 0 & 4 & -2 & -2 & 2 & 2 \\ 0 & 0 & 2 & 2 & 2 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

Help | Powered by SageMath

Calculs liés

```
banner()
show(s[1])
```

Evaluate

```
SageMath version 9.7, Release Date: 2022-09-19
Using Python 3.10.6. Type "help()" for help.
```

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

Help | Powered by SageMath

Kernel: SageMath 10.3

Morphisme induit en homologie par la digonale de la sphère S^2

Stage M1 - 24 juin, 1-4 juil 2024

In [3]:
```
# Cf
https://doc.sagemath.org/html/en/reference/topology/sage/topology/simplicial_complex_examp
les.html#
K=simplicial_complexes.Sphere(2)
print(K, ',\033[1m dim :\033[0m', K.dimension(), ',\033[1m nbre simplexes maximaux
:\033[0m',len(K.maximal_faces()))
print('\033[1m'+'Simplexes :\033[0m',K.cells())
print('\033[1m'+'Simplexes maximaux :\033[0m',K.maximal_faces())
```

Out[3]:
```
Minimal triangulation of the 2-sphere , dim : 2 , nbre simplexes maximaux : 4
Simplexes : {-1: {()}, 0: {(3,), (2,), (1,), (0,)}, 1: {(2, 3), (1, 2), (0, 3), (0, 1), (0,
2), (1, 3)}, 2: {(1, 2, 3), (0, 1, 3), (0, 2, 3), (0, 1, 2)}}
Simplexes maximaux : {(1, 2, 3), (0, 1, 3), (0, 2, 3), (0, 1, 2)}
```

In [4]:
```
print(K.homology(reduced=False)) # homologie non-réduite à coefficient dans ZZ, cf
K.homology? ou help(K.homology) une fois K défini.
```

Out[4]:
```
{0: Z, 1: 0, 2: Z}
```

In [5]:
```
help(K.homology)
```

Out[5]:
```
Help on method homology in module sage.topology.cell_complex:

homology(dim=None, base_ring=Integer Ring, subcomplex=None, generators=False,
cohomology=False, algorithm='pari', verbose=False, reduced=True, **kwds) method of
sage.topology.simplicial_complex_examples.UniqueSimplicialComplex_with_category instance
    The (reduced) homology of this cell complex.

    :param dim: If None, then return the homology in every
       dimension.  If ``dim`` is an integer or list, return the
       homology in the given dimensions.  (Actually, if ``dim`` is
       a list, return the homology in the range from ``min(dim)``
       to ``max(dim)``.)
    :type dim: integer or list of integers or None; optional,
       default None
    :param base_ring: commutative ring, must be ZZ or a field.
    :type base_ring: optional, default ZZ
    :param subcomplex: a subcomplex of this simplicial complex.
       Compute homology relative to this subcomplex.
    :type subcomplex: optional, default empty
    :param generators: If ``True``, return generators for the homology
       groups along with the groups.
    :type generators: boolean; optional, default False
    :param cohomology: If True, compute cohomology rather than homology.
    :type cohomology: boolean; optional, default False
    :param algorithm: The options are 'auto', 'dhsw', or 'pari'.
       See below for a description of what they mean.
    :type algorithm: string; optional, default 'pari'
    :param verbose: If True, print some messages as the homology is
       computed.
    :type verbose: boolean; optional, default False
    :param reduced: If ``True``, return the reduced homology.
    :type reduced: boolean; optional, default ``True``

    ALGORITHM:

    Compute the chain complex of ``self`` and compute its homology
    groups.  To do this: over a field, just compute ranks and
    nullities, thus obtaining dimensions of the homology groups as
    vector spaces.  Over the integers, compute Smith normal form
    of the boundary matrices defining the chain complex according
    to the value of ``algorithm``.  If ``algorithm`` is
    ``'auto'``, then for each relatively small matrix, use the
    standard Sage method, which calls the Pari package.  For any
    large matrix, reduce it using the Dumas, Heckenbach, Saunders,
    and Welker elimination algorithm [DHSW2003]_: see
    :func:`~sage.homology.matrix_utils.dhsw_snf` for details.

    ``'no_chomp'`` is a synonym for ``'auto'``, maintained for
    backward-compatibility.
```

L'homologie s'obtient en explicitant une $\mathbb{Z}$-base adaptée du sous-groupe des bords dans le sous-groupe des cycles.

Ce niveau de détail est il implémenté dans Sagemath ?

In [6]:
```
# Produit
# Cf
https://doc.sagemath.org/html/en/reference/topology/sage/topology/simplicial_complex_morphism.html
K2=K.product(K, rename_vertices=False, is_mutable=False)
print('K×K:',K2, ',dim :', K2.dimension());print()
print(K2.maximal_faces())
# Quel genre de produit est-ce ?
```

Out[6]:
```
K×K: Simplicial complex with 16 vertices and 96 facets ,dim : 4

{((0, 0), (1, 0), (1, 1), (2, 1), (2, 2)), ((1, 0), (1, 1), (2, 1), (2, 3), (3, 3)), ((1,
1), (2, 1), (2, 2), (2, 3), (3, 3)), ((0, 1), (0, 2), (0, 3), (1, 3), (2, 3)), ((0, 0), (1,
0), (1, 2), (1, 3), (3, 3)), ((0, 0), (0, 2), (1, 2), (2, 2), (2, 3)), ((0, 0), (1, 0), (1,
2), (1, 3), (2, 3)), ((0, 0), (2, 0), (2, 1), (2, 2), (3, 2)), ((0, 0), (2, 0), (2, 2), (3,
2), (3, 3)), ((0, 0), (1, 0), (1, 2), (2, 2), (2, 3)), ((0, 0), (0, 1), (1, 1), (1, 3), (3,
3)), ((1, 0), (2, 0), (2, 1), (3, 1), (3, 2)), ((1, 0), (1, 2), (1, 3), (2, 3), (3, 3)),
((0, 0), (0, 2), (1, 2), (1, 3), (2, 3)), ((0, 0), (0, 2), (1, 2), (3, 2), (3, 3)), ((0, 0),
(2, 0), (2, 1), (2, 3), (3, 3)), ((1, 0), (1, 1), (1, 3), (2, 3), (3, 3)), ((1, 0), (2, 0),
(2, 1), (2, 3), (3, 3)), ((1, 1), (1, 2), (2, 2), (3, 2), (3, 3)), ((0, 0), (1, 0), (3, 0),
(3, 1), (3, 3)), ((0, 0), (0, 1), (2, 1), (2, 2), (3, 2)), ((0, 0), (0, 1), (0, 2), (1, 2),
(2, 2)), ((1, 0), (1, 1), (2, 1), (3, 1), (3, 2)), ((0, 0), (1, 0), (1, 1), (1, 2), (2, 2)),
((0, 0), (1, 0), (3, 0), (3, 2), (3, 3)), ((0, 0), (2, 0), (3, 0), (3, 2), (3, 3)), ((1, 1),
(2, 1), (3, 1), (3, 2), (3, 3)), ((0, 0), (0, 1), (2, 1), (2, 3), (3, 3)), ((0, 0), (0, 2),
(0, 3), (1, 3), (2, 3)), ((1, 0), (2, 0), (3, 0), (3, 2), (3, 3)), ((0, 0), (0, 1), (0, 2),
(1, 2), (3, 2)), ((0, 0), (0, 2), (1, 2), (1, 3), (3, 3)), ((0, 0), (1, 0), (1, 1), (1, 2),
(3, 2)), ((0, 0), (1, 0), (2, 0), (2, 1), (2, 2)), ((0, 0), (2, 0), (2, 1), (3, 1), (3, 3)),
((0, 0), (1, 0), (2, 0), (2, 1), (2, 3)), ((0, 1), (0, 2), (1, 2), (1, 3), (2, 3)), ((1, 0),
(1, 1), (2, 1), (3, 1), (3, 3)), ((1, 0), (1, 2), (2, 2), (3, 2), (3, 3)), ((0, 0), (1, 0),
(1, 1), (1, 3), (2, 3)), ((0, 1), (0, 2), (0, 3), (1, 3), (3, 3)), ((0, 1), (1, 1), (3, 1),
(3, 2), (3, 3)), ((0, 0), (1, 0), (1, 1), (3, 1), (3, 3)), ((0, 0), (1, 0), (1, 2), (3, 2),
(3, 3)), ((0, 1), (1, 1), (2, 1), (2, 2), (2, 3)), ((1, 1), (2, 1), (2, 2), (3, 2), (3, 3)),
((0, 0), (1, 0), (1, 1), (3, 1), (3, 2)), ((1, 0), (1, 2), (2, 2), (2, 3), (3, 3)), ((0, 0),
(0, 1), (1, 1), (2, 1), (2, 3)), ((0, 0), (0, 1), (1, 1), (1, 2), (3, 2)), ((0, 1), (2, 1),
(2, 2), (2, 3), (3, 3)), ((0, 0), (0, 1), (2, 1), (3, 1), (3, 3)), ((1, 0), (2, 0), (2, 1),
(3, 1), (3, 3)), ((0, 0), (0, 1), (0, 3), (2, 3), (3, 3)), ((0, 1), (1, 1), (1, 2), (1, 3),
(3, 3)), ((1, 0), (1, 1), (1, 2), (2, 2), (3, 2)), ((0, 0), (0, 1), (0, 2), (2, 2), (3, 2)),
((0, 0), (0, 1), (1, 1), (1, 2), (2, 2)), ((0, 0), (0, 1), (2, 1), (3, 1), (3, 2)), ((0, 1),
(0, 2), (2, 2), (3, 2), (3, 3)), ((0, 1), (0, 2), (1, 2), (1, 3), (3, 3)), ((0, 0), (1, 0),
(1, 1), (1, 3), (3, 3)), ((0, 0), (1, 0), (2, 0), (2, 2), (2, 3)), ((1, 1), (1, 2), (2, 2),
(2, 3), (3, 3)), ((0, 0), (1, 0), (3, 0), (3, 1), (3, 2)), ((0, 1), (0, 2), (2, 2), (2, 3),
(3, 3)), ((0, 0), (0, 2), (0, 3), (1, 3), (3, 3)), ((0, 0), (2, 0), (3, 0), (3, 1), (3, 2)),
((0, 0), (0, 1), (1, 1), (2, 1), (2, 2)), ((1, 0), (1, 1), (2, 1), (2, 2), (3, 2)), ((0, 0),
(0, 2), (0, 3), (2, 3), (3, 3)), ((0, 0), (0, 1), (1, 1), (1, 3), (2, 3)), ((1, 1), (1, 2),
(1, 3), (2, 3), (3, 3)), ((0, 0), (2, 0), (2, 1), (3, 1), (3, 2)), ((0, 0), (1, 0), (1, 1),
(2, 1), (2, 3)), ((0, 0), (0, 1), (0, 3), (1, 3), (3, 3)), ((0, 0), (2, 0), (3, 0), (3, 1),
(3, 3)), ((1, 0), (2, 0), (2, 2), (2, 3), (3, 3)), ((0, 0), (2, 0), (2, 2), (2, 3), (3, 3)),
((0, 0), (0, 1), (0, 3), (1, 3), (2, 3)), ((1, 0), (2, 0), (3, 0), (3, 1), (3, 3)), ((0, 1),
(0, 2), (1, 2), (2, 2), (2, 3)), ((0, 0), (0, 1), (1, 1), (3, 1), (3, 2)), ((0, 1), (2, 1),
(3, 1), (3, 2), (3, 3)), ((0, 1), (1, 1), (1, 2), (2, 2), (2, 3)), ((1, 0), (2, 0), (2, 1),
(2, 2), (3, 2)), ((0, 1), (0, 2), (1, 2), (3, 2), (3, 3)), ((0, 0), (0, 2), (2, 2), (3, 2),
(3, 3)), ((0, 1), (1, 1), (1, 2), (3, 2), (3, 3)), ((1, 0), (2, 0), (3, 0), (3, 1), (3, 2)),
((1, 0), (2, 0), (2, 2), (3, 2), (3, 3)), ((0, 0), (0, 1), (1, 1), (3, 1), (3, 3)), ((0, 1),
(1, 1), (1, 2), (1, 3), (2, 3)), ((0, 1), (2, 1), (2, 2), (3, 2), (3, 3)), ((0, 0), (0, 2),
(2, 2), (2, 3), (3, 3)), ((0, 1), (0, 2), (0, 3), (2, 3), (3, 3))}
```

In [7]:
```
print(K2.homology(reduced=False))
```

Out[7]:
```
{0: Z, 1: 0, 2: Z x Z, 3: 0, 4: Z}
```

"96 simplexes maximaux" semble excessif pour un modèle combinatoire de $\mathrm{S}^2 \times \mathrm{S}^2$. En comparaison, le produit de K par lui même dans la catégorie des complexes simpliciaux a 16 simplexes maximaux.

Détermination dans Sagemath du produit catégoriel de K par lui-même ?

In [8]:
```
# Cf
https://doc.sagemath.org/html/en/reference/topology/sage/topology/simplicial_complex_morphism.html
d=Hom(K,K.product(K, is_mutable=False)).diagonal_morphism()
print(d)
#d=Hom(K,K.product(K, rename_vertices=False, is_mutable=False)).diagonal_morphism()
produit une erreur ; il faudrait définir ce d à la main.
```

Out[8]:
```
Simplicial complex morphism:
  From: Minimal triangulation of the 2-sphere
  To:   Simplicial complex with 16 vertices and 96 facets
  Defn: 0 |--> L0R0
        1 |--> L1R1
        2 |--> L2R2
        3 |--> L3R3
```

$\mathrm{H}_n(d)\colon \mathrm{H}_n(\mathrm{S}^2) \to \mathrm{H}_n(\mathrm{S}^2 \times \mathrm{S}^2)$ pour $n = 0, 1, 2, 3, 4$. Seul morphisme non trivial pour $n = 2$, de Z dans $\mathrm{Z} \times \mathrm{Z}$, décrit par une matrice à déterminer.

In [9]:
```
d.induced_homology_morphism()
# n'est défini dans Sagemath que pour l'homologie à coefficients dans un corps, par défaut QQ
```

Out[9]:
```
Graded vector space morphism:
  From: Homology module of Minimal triangulation of the 2-sphere over Rational Field
  To:   Homology module of Simplicial complex with 16 vertices and 96 facets over Rational Field
  Defn: induced by:
          Simplicial complex morphism:
            From: Minimal triangulation of the 2-sphere
            To:   Simplicial complex with 16 vertices and 96 facets
            Defn: 0 |--> L0R0
                  1 |--> L1R1
                  2 |--> L2R2
                  3 |--> L3R3
```

On reste sur sa faim.

Pour aller plus loin : https://doc.sagemath.org/html/en/reference/homology/sage/homology/homology_morphism.html

Méme chose avec la bouteille de Klein

In [10]:
```
L=simplicial_complexes.KleinBottle()
print(L,', dim :',dim(L),', nbre simpl. max.:',len(L.maximal_faces()))
print(L.maximal_faces())
print('Homologie :',L.homology())
```

Out[10]:
```
Minimal triangulation of the Klein bottle , dim : 2 , nbre simpl. max.: 16
{(1, 3, 5), (3, 4, 7), (2, 3, 7), (0, 2, 5), (0, 2, 4), (1, 5, 7), (0, 5, 6), (0, 1, 6), (2, 4, 6), (2, 5, 7), (0, 1, 4), (1, 4, 7), (3, 5, 6), (3, 4, 6), (1, 2, 3), (1, 2, 6)}
Homologie : {0: 0, 1: Z x C2, 2: 0}
```

In [11]:
```
L2=L.product(L, is_mutable=False)
print(L2, ', dim :',dim(L2))
```

Out[11]:
```
Simplicial complex with 64 vertices and 1536 facets , dim : 4
```

Trop de simplexes ⇝ le calcul de l'homologie sera long, trop long pour Cocalc.

Le calcul sur jhub rend :

```
print(L2.homology()) #Ce n'est pas rapide !
#{0: 0, 1: Z x Z x C2 x C2, 2: Z x C2 x C2 x C2, 3: C2, 4: 0}
```

En comparaison le produit catégoriel de L par lui même a $(\text{nbre de simpl. max. de L})^2 = 256$ simplexes maximaux.

Cette fois le morphisme induit en homologie par la diagonale d est (mal ?) décrit par

$\mathrm{H}_1(d)\colon \mathrm{Z} \times \mathrm{Z}/2 \to \mathrm{Z} \times \mathrm{Z} \times \mathrm{Z}/2 \times \mathrm{Z}/2.$

Notion de matrice pour une telle application ?

Périodes

Kernel: SageMath 10.3

Stage 2nde Factorisation des entiers - 25 juin 2024

Recherche de cycle pour une suite $(u_0, u_{n+1} = f(u_n))$

In [3]:
```
sage.
misc.banner.banner()
```

```
SageMath version 10.3, Release Date: 2024-03-19
Create a "Sage Worksheet" file for the notebook interface.
Enhanced for CoCalc.
Using Python 3.11.1. Type "help()" for help.
```

In [4]:
```
def
per_1(f,u0):
    l=[u0];j=1;x=f(u0)
    while x not in l:
        l.append(x);x=f(
x);j+=1
    i=l.index(x)
    return(i,j-i)
```

In [5]:
```
def per_2(f,u0):
    i,p,x,y=0,1,u0,f(u0)
    while x!=y:
        if p<=i:
            i,p,x,y=i,p+1,x,f(y)
        else:
            i,p,x,y=i+p,1,y,f(y)
    return(i,p)
```

In [6]:
```
f=lambda x:(10*x)
%14
print('algo 1:',per_1(f,85))
print('algo 2:',per_2(f,85))
```

```
algo 1: (1, 6) algo 2: (7, 6)
```

In [7]:
```
def
l_cycles(n,f):#n>0
    li=[0..n-1];lc=[];#print(li,lc)
    while li!=[]:
        i=li.pop(0);l=[i];x=f(i)%
n
        while x not in l and
 x in li:
            l.append(x);li.remove(
x);x=f(x)%n
        if x in l:
            l=l[l.index(x)
:];l.sort()
            if l not in lc:
lc.append(l)
        #print(li,lc)
    return(lc)
```

In [8]:
```
f=lambda x:x^2+
1
print(l_cycles(13,f))
```

```
[[0, 1, 2, 5], [10], [4]]
```

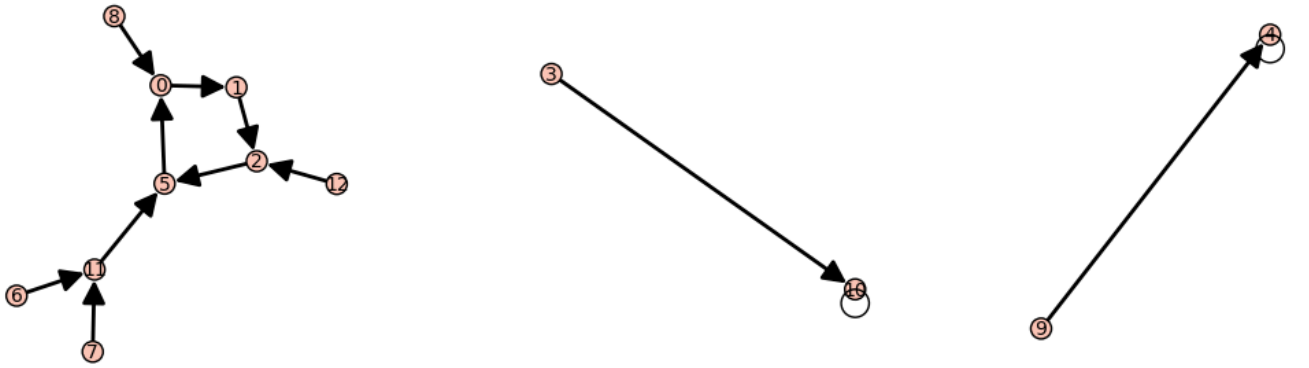
In [9]:
```
for
n in range(1,100):
    print(n,l_cycles(n,f))
```

```
1 [[0]] 2 [[0, 1]] 3 [[2]] 4 [[1, 2]] 5 [[0, 1, 2]] 6 [[2, 5]] 7 [[5], [3]] 8 [[2, 5]] 9 [[2, 5, 8]] 10 [[0, 1, 2,
5, 6, 7]] 11 [[4, 6]] 12 [[2, 5]] 13 [[0, 1, 2, 5], [10], [4]] 14 [[5, 12], [3, 10]] 15 [[2, 5, 11]] 16 [[5, 10]]
17 [[2, 5, 9, 10, 14, 16]] 18 [[2, 5, 8, 11, 14, 17]] 19 [[12], [8]] 20 [[1, 2, 5, 6, 10, 17]] 21 [[5], [17]] 22
[[4, 17], [6, 15]] 23 [[9, 13]] 24 [[2, 5]] 25 [[1, 2, 5]] 26 [[0, 1, 2, 5], [10, 23], [4, 17], [13, 14, 15, 18]]
27 [[2, 5, 26], [17, 20, 23], [8, 11, 14]] 28 [[5, 26], [10, 17]] 29 [[7, 21]] 30 [[2, 5, 11, 17, 20, 26]] 31
[[26], [3, 8, 10], [6]] 32 [[5, 26]] 33 [[17, 26]] 34 [[2, 5, 10, 26, 31, 33], [9, 14, 16, 19, 22, 27]] 35 [[5,
```

```
12, 26], [10, 17, 31]] 36 [[2, 5, 14, 17, 26, 29]] 37 [[11], [27], [8, 28]] 38 [[12, 31], [8, 27]] 39 [[2, 5, 14,
26], [23], [17]] 40 [[2, 5, 10, 21, 26, 37]] 41 [[0, 1, 2, 5, 21, 26, 32]] 42 [[5, 26], [17, 38]] 43 [[7], [37],
[18, 24]] 44 [[17, 26], [6, 37]] 45 [[2, 5, 26], [11, 32, 35], [17, 20, 41]] 46 [[13, 32], [9, 36]] 47 [[8, 17,
18, 43], [3, 7, 10]] 48 [[5, 26]] 49 [[5, 12, 26, 33, 40, 47], [3, 10], [17, 45], [24, 38], [19], [31]] 50 [[1, 2,
5, 26, 27, 30]] 51 [[2, 5, 14, 26, 44, 50]] 52 [[1, 2, 5, 26], [10, 49], [17, 30], [13, 14, 18, 41]] 53 [[14, 38]]
54 [[2, 5, 26, 29, 32, 53], [17, 20, 23, 44, 47, 50], [8, 11, 14, 35, 38, 41]] 55 [[6, 15, 17, 26, 37, 50]] 56
[[5, 26], [10, 45]] 57 [[50], [8]] 58 [[7, 50], [21, 36]] 59 [[16, 21, 29]] 60 [[2, 5, 17, 26, 41, 50]] 61 [[4, 6,
17, 20, 28, 35, 37, 43, 46, 53], [48], [14]] 62 [[26, 57], [3, 8, 10, 34, 39, 41], [6, 37]] 63 [[5, 26, 47], [17,
38, 59]] 64 [[26, 37]] 65 [[0, 1, 2, 5, 15, 26, 27, 31, 40, 41, 52, 57], [10, 36, 62], [17, 30, 56]] 66 [[17, 26],
[50, 59]] 67 [[11, 55], [18, 34, 57], [30], [38]] 68 [[2, 5, 10, 26, 33, 65], [9, 14, 22, 50, 53, 61]] 69 [[32,
59]] 70 [[5, 12, 26, 40, 47, 61], [10, 17, 31, 45, 52, 66]] 71 [[4, 6, 16, 17, 20, 21, 28, 37, 44, 46, 58], [3,
10, 30, 49, 59], [31, 39]] 72 [[2, 5, 26, 29, 50, 53]] 73 [[5, 20, 26, 36, 56, 71], [65], [9]] 74 [[11, 48], [27,
64], [8, 65], [28, 45]] 75 [[2, 5, 26]] 76 [[50, 69], [46, 65]] 77 [[26, 61], [17, 59]] 78 [[2, 5, 26, 53], [23,
62], [17, 56], [14, 41, 44, 65]] 79 [[26, 45, 51, 74], [6, 19, 20, 27, 37, 46, 63], [12, 66], [56], [24]] 80 [[5,
10, 21, 26, 37, 42]] 81 [[2, 5, 26, 29, 32, 53, 56, 59, 80], [17, 20, 23, 44, 47, 50, 71, 74, 77], [8, 11, 14, 35,
38, 41, 62, 65, 68]] 82 [[0, 1, 2, 5, 21, 26, 32, 41, 42, 43, 46, 62, 67, 73]] 83 [[4, 17, 22, 41, 70]] 84 [[5,
26], [17, 38]] 85 [[2, 5, 10, 16, 26, 82], [22, 27, 31, 36, 50, 60], [56, 61, 65, 67, 70, 77]] 86 [[7, 50], [37,
80], [24, 61], [18, 67]] 87 [[50, 65]] 88 [[26, 61], [37, 50]] 89 [[2, 5, 10, 12, 22, 26, 40, 45, 54, 56, 68, 69,
86, 88], [17, 23, 85], [9, 50, 82]] 90 [[2, 5, 26, 47, 50, 71], [11, 32, 35, 56, 77, 80], [17, 20, 41, 62, 65,
86]] 91 [[5, 26, 40, 54], [10], [17], [82], [31, 52, 66, 80], [75]] 92 [[13, 78], [9, 82]] 93 [[26], [8, 41, 65],
[68]] 94 [[18, 43, 55, 64], [3, 7, 10, 50, 54, 57], [8, 17, 65, 90]] 95 [[12, 31, 50], [27, 46, 65]] 96 [[5, 26]]
97 [[5, 26, 95], [7, 50, 54, 76], [62], [36]] 98 [[5, 26, 61, 82, 89, 96], [3, 10], [17, 94], [12, 33, 40, 47, 54,
75], [24, 87], [19, 68], [31, 80], [38, 73], [45, 66], [52, 59]] 99 [[17, 26, 50, 59, 83, 92]]
```

In [10]:
```
n=13
f=lambda x:(x^2+1)%n
M=DiGraph(loops=True)
for i in range(n):
    M.add_edges([(i,f(i)
)])
```

In [11]:
```
M.show(figsize=(10))
```



In [12]:
```
def dic_rho(n,f):
    li=[0..n-1];dic={}
    while li!=[]:
        i=li.pop(0);l=[i];x=f(i)%
n
        while x not in l and
 x in li:
            l.append(x);li.remove(
x);x=f(x)%n
        I=len(l)
        if x in l:
            i=l.index(x);p
=I-i
            for j in range(i):dic[l[j]]=[i-j
,p]
            for j in range(i,I):dic[l[j]]=[0
,p]
            #print(x,l,li,dic)
        else:
            i=dic[x][0];p
=dic[x][1]
            for j in range(I):dic[l[j]]=[I-j
+i,p]
            #print(x,l,li,dic)
    return(dic)
```

In [13]:
```
f=lambda x:x^2+
1
print('n=13 :',dic_rho(13,f))
print('n=17 :',dic_rho(17,f))
```

```
n=13 : {0: [0, 4], 1: [0, 4], 2: [0, 4], 5: [0, 4], 3: [1, 1], 10: [0, 1], 4: [0, 1], 6: [2, 4], 11: [1, 4], 7:
[2, 4], 8: [1, 4], 9: [1, 1], 12: [1, 4]} n=17 : {0: [2, 6], 1: [1, 6], 2: [0, 6], 5: [0, 6], 9: [0, 6], 14: [0,
```

```
6], 10: [0, 6], 16: [0, 6], 3: [1, 6], 4: [3, 6], 6: [2, 6], 7: [1, 6], 8: [1, 6], 11: [2, 6], 12: [1, 6], 13: [3,
6], 15: [1, 6]}
```

In [14]:
```
def
dic_pollard(p,q,f):
    dp=dic_rho(p,f);dq=dic_rho(q,f);dn=dic_rho
(p*q,f);dic={}

    for i in range(p*q):
        if sum(dp[i%p])
<sum(dq[i%q]):dic[i]=dp[i%p]+dn[i]
        else:dic[i]=dq[i%q]+dn[i]
    return(dic)
```

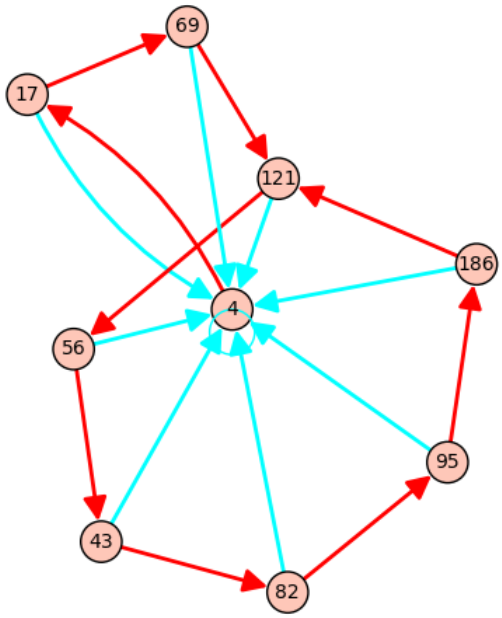
In [15]:
```
p,q=13,17
dp=dic_pollard(p,q,f)
print(dp)
```

```
{0: [0, 4, 2, 12], 1: [0, 4, 1, 12], 2: [0, 4, 0, 12], 3: [1, 1, 1, 6], 4: [0, 1, 3, 6], 5: [0, 4, 0, 12], 6: [2,
4, 2, 12], 7: [2, 4, 2, 12], 8: [1, 4, 1, 12], 9: [1, 1, 1, 6], 10: [0, 1, 0, 6], 11: [1, 4, 2, 12], 12: [1, 4, 1,
12], 13: [0, 4, 3, 12], 14: [0, 4, 0, 12], 15: [0, 4, 1, 12], 16: [1, 1, 1, 6], 17: [0, 1, 2, 6], 18: [0, 4, 1,
12], 19: [0, 6, 2, 12], 20: [2, 4, 2, 12], 21: [1, 4, 3, 12], 22: [1, 1, 1, 6], 23: [0, 1, 2, 6], 24: [1, 4, 1,
12], 25: [1, 4, 1, 12], 26: [0, 4, 0, 12], 27: [0, 4, 0, 12], 28: [0, 4, 2, 12], 29: [1, 1, 1, 6], 30: [0, 1, 3,
6], 31: [0, 4, 0, 12], 32: [2, 4, 2, 12], 33: [0, 6, 2, 12], 34: [1, 4, 2, 12], 35: [1, 1, 1, 6], 36: [0, 1, 0,
6], 37: [1, 4, 1, 12], 38: [1, 4, 3, 12], 39: [0, 4, 0, 12], 40: [0, 4, 2, 12], 41: [0, 4, 1, 12], 42: [1, 1, 1,
6], 43: [0, 1, 0, 6], 44: [0, 4, 0, 12], 45: [2, 4, 2, 12], 46: [2, 4, 2, 12], 47: [1, 4, 3, 12], 48: [1, 1, 1,
6], 49: [0, 1, 1, 6], 50: [1, 4, 1, 12], 51: [1, 4, 2, 12], 52: [0, 4, 1, 12], 53: [0, 4, 0, 12], 54: [0, 4, 1,
12], 55: [1, 1, 3, 6], 56: [0, 1, 0, 6], 57: [0, 4, 2, 12], 58: [2, 4, 2, 12], 59: [2, 4, 2, 12], 60: [1, 4, 1,
12], 61: [1, 1, 1, 6], 62: [0, 1, 2, 6], 63: [1, 4, 1, 12], 64: [1, 4, 3, 12], 65: [0, 4, 0, 12], 66: [0, 4, 1,
12], 67: [0, 4, 0, 12], 68: [1, 1, 2, 6], 69: [0, 1, 1, 6], 70: [0, 4, 0, 12], 71: [2, 4, 2, 12], 72: [2, 4, 3,
12], 73: [1, 4, 1, 12], 74: [1, 1, 2, 6], 75: [0, 1, 1, 6], 76: [1, 4, 1, 12], 77: [1, 4, 1, 12], 78: [0, 4, 0,
12], 79: [0, 4, 2, 12], 80: [0, 4, 1, 12], 81: [1, 1, 3, 6], 82: [0, 1, 0, 6], 83: [0, 4, 1, 12], 84: [0, 6, 2,
12], 85: [2, 4, 2, 12], 86: [1, 4, 1, 12], 87: [1, 1, 1, 6], 88: [0, 1, 1, 6], 89: [1, 4, 3, 12], 90: [1, 4, 1,
12], 91: [0, 4, 2, 12], 92: [0, 4, 1, 12], 93: [0, 4, 1, 12], 94: [1, 1, 1, 6], 95: [0, 1, 0, 6], 96: [0, 4, 2,
12], 97: [2, 4, 2, 12], 98: [2, 4, 3, 12], 99: [1, 4, 1, 12], 100: [1, 1, 1, 6], 101: [0, 1, 0, 6], 102: [1, 4, 2,
12], 103: [1, 4, 1, 12], 104: [0, 4, 0, 12], 105: [0, 4, 1, 12], 106: [0, 4, 3, 12], 107: [1, 1, 1, 6], 108: [0,
1, 2, 6], 109: [0, 4, 1, 12], 110: [2, 4, 2, 12], 111: [0, 6, 2, 12], 112: [1, 4, 1, 12], 113: [1, 1, 2, 6], 114:
[0, 1, 1, 6], 115: [1, 4, 3, 12], 116: [1, 4, 1, 12], 117: [0, 4, 1, 12], 118: [0, 4, 0, 12], 119: [0, 4, 2, 12],
120: [1, 1, 1, 6], 121: [0, 1, 0, 6], 122: [0, 4, 1, 12], 123: [2, 4, 3, 12], 124: [0, 6, 2, 12], 125: [1, 4, 2,
12], 126: [1, 1, 1, 6], 127: [0, 1, 1, 6], 128: [1, 4, 1, 12], 129: [1, 4, 1, 12], 130: [0, 4, 2, 12], 131: [0, 4,
1, 12], 132: [0, 4, 3, 12], 133: [1, 1, 1, 6], 134: [0, 1, 1, 6], 135: [0, 4, 0, 12], 136: [2, 4, 2, 12], 137: [2,
4, 2, 12], 138: [1, 4, 1, 12], 139: [1, 1, 1, 6], 140: [0, 1, 3, 6], 141: [1, 4, 1, 12], 142: [1, 4, 2, 12], 143:
[0, 4, 1, 12], 144: [0, 4, 1, 12], 145: [0, 4, 0, 12], 146: [1, 1, 1, 6], 147: [0, 1, 2, 6], 148: [0, 4, 1, 12],
149: [2, 4, 3, 12], 150: [0, 6, 2, 12], 151: [1, 4, 1, 12], 152: [1, 1, 1, 6], 153: [0, 1, 2, 6], 154: [1, 4, 1,
12], 155: [1, 4, 1, 12], 156: [0, 4, 1, 12], 157: [0, 4, 3, 12], 158: [0, 4, 0, 12], 159: [1, 1, 2, 6], 160: [0,
1, 1, 6], 161: [0, 4, 1, 12], 162: [0, 6, 2, 12], 163: [0, 6, 2, 12], 164: [1, 4, 2, 12], 165: [1, 1, 1, 6], 166:
[0, 1, 3, 6], 167: [1, 4, 1, 12], 168: [1, 4, 1, 12], 169: [0, 4, 0, 12], 170: [0, 4, 2, 12], 171: [0, 4, 1, 12],
172: [1, 1, 1, 6], 173: [0, 1, 1, 6], 174: [0, 4, 3, 12], 175: [0, 6, 2, 12], 176: [2, 4, 2, 12], 177: [1, 4, 1,
12], 178: [1, 1, 1, 6], 179: [0, 1, 0, 6], 180: [1, 4, 1, 12], 181: [1, 4, 2, 12], 182: [0, 4, 1, 12], 183: [0, 4,
3, 12], 184: [0, 4, 0, 12], 185: [1, 1, 1, 6], 186: [0, 1, 0, 6], 187: [0, 4, 2, 12], 188: [2, 4, 2, 12], 189: [0,
6, 2, 12], 190: [1, 4, 1, 12], 191: [1, 1, 3, 6], 192: [0, 1, 0, 6], 193: [1, 4, 2, 12], 194: [1, 4, 1, 12], 195:
[0, 4, 1, 12], 196: [0, 4, 0, 12], 197: [0, 4, 0, 12], 198: [1, 1, 2, 6], 199: [0, 1, 1, 6], 200: [0, 4, 3, 12],
201: [0, 6, 2, 12], 202: [2, 4, 2, 12], 203: [1, 4, 1, 12], 204: [1, 1, 2, 6], 205: [0, 1, 1, 6], 206: [1, 4, 1,
12], 207: [1, 4, 1, 12], 208: [0, 4, 3, 12], 209: [0, 4, 0, 12], 210: [0, 4, 2, 12], 211: [1, 1, 1, 6], 212: [0,
1, 1, 6], 213: [0, 4, 0, 12], 214: [0, 6, 2, 12], 215: [2, 4, 2, 12], 216: [1, 4, 1, 12], 217: [1, 1, 3, 6], 218:
[0, 1, 0, 6], 219: [1, 4, 1, 12], 220: [1, 4, 1, 12]}
```

In [16]:
```
x=4
print(str(x)+':',dp[x])
```

```
4: [0, 1, 3, 6]
```

In [17]:
```
x=4
G=DiGraph(loops=True,multiedges=True)
for i in range(sum(dp[
x][2:])):
    G.add_edges([(x,f(x)
%(p*q))])
    G.add_edges([(x,f(x)
%p,"p")])
    x=f(x)%(p*
q)

show(G.plot(vertex_size=300,color_by_label=True))
```



In [18]:
```
x=4
print(str(x)+':',dp[x])
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p*
q);print(y,'\t',end='')
print()
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p);
print(y,'\t',end='')
```

```
4: [0, 1, 3, 6] 4 17 69 121 56 43 82 95 186 121 4 4 4 4 4 4 4 4 4 4
```

In [19]:
```
x=11
print(str(x)+':',dp[x])
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p*
q);print(y,'\t',end='')
print()
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p);
print(y,'\t',end='')
```

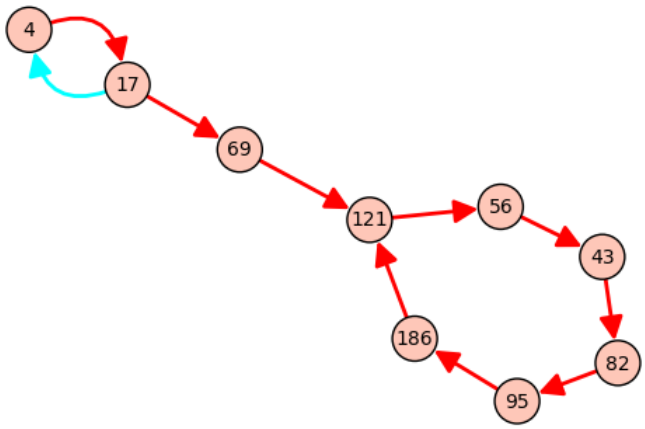
```
11: [1, 4, 2, 12] 11 122 78 118 2 5 26 14 197 135 104 209 145 31 78 11 5 0 1 2 5 0 1 2 5 0 1 2 5 0
```

In [20]:
```
def
G(x,dp,p,q,f):

    G=DiGraph(loops=True,
multiedges=True)
    y=x
    for i in range(sum(dp
[x][2:])):
        G.add_edges([(y,f(y)
%(p*q))])
        y=f(y)%(p
*q)
    y=x
    for i in range(dp[x
][0]):
        y=f(y)%(p
*q)
    z=y
    for i in range(dp[x
][1]):
        y=f(y)%(p
*q)
    G.add_edges([(y,z,"p")
```

```
])
    return(G)
```

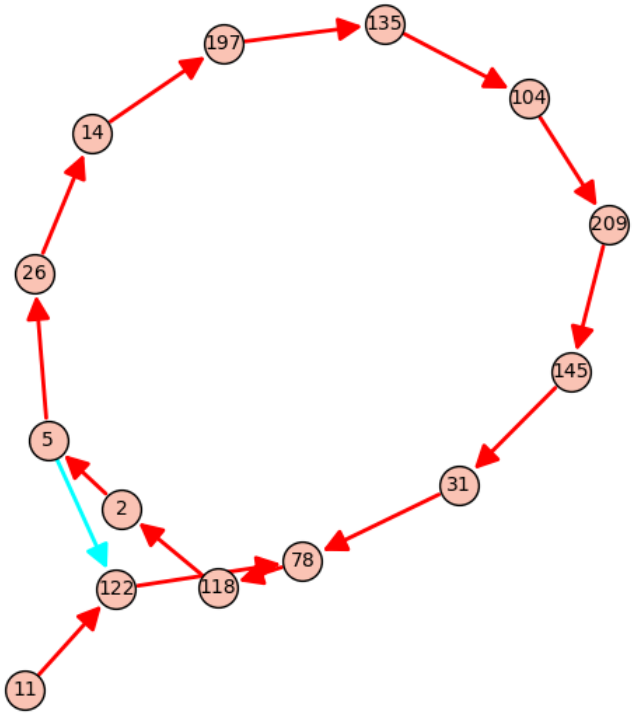
In [21]:

```
x=4
show(G(x,dp,p,q,
f).plot(figsize=5,color_by_label=True))#,vertex_size=300
```



In [22]:

```
x=11
show(G(x,dp,p,q,
f).plot(figsize=7,color_by_label=True,vertex_size=400))#,vertex_size=300
```



Si on a bien suivi, **on observe :** 122-5 a un facteur commun avec n=13*17. De fait :

In [23]:

```
gcd(
122-5,13*17)
```

13

In [24]:

```
p,
q=13,17
f=lambda x:2*x
dp=dic_pollard(p,q,f)
print(dp)
```

```
{0: [0, 1, 0, 1], 1: [0, 8, 0, 24], 2: [0, 8, 0, 24], 3: [0, 8, 0, 24], 4: [0, 8, 0, 24], 5: [0, 8, 0, 24], 6: [0,
8, 0, 24], 7: [0, 8, 0, 24], 8: [0, 8, 0, 24], 9: [0, 8, 0, 24], 10: [0, 8, 0, 24], 11: [0, 8, 0, 24], 12: [0, 8,
0, 24], 13: [0, 1, 0, 8], 14: [0, 8, 0, 24], 15: [0, 8, 0, 24], 16: [0, 8, 0, 24], 17: [0, 1, 0, 12], 18: [0, 8,
0, 24], 19: [0, 8, 0, 24], 20: [0, 8, 0, 24], 21: [0, 8, 0, 24], 22: [0, 8, 0, 24], 23: [0, 8, 0, 24], 24: [0, 8,
0, 24], 25: [0, 8, 0, 24], 26: [0, 1, 0, 8], 27: [0, 8, 0, 24], 28: [0, 8, 0, 24], 29: [0, 8, 0, 24], 30: [0, 8,
0, 24], 31: [0, 8, 0, 24], 32: [0, 8, 0, 24], 33: [0, 8, 0, 24], 34: [0, 1, 0, 12], 35: [0, 8, 0, 24], 36: [0, 8,
0, 24], 37: [0, 8, 0, 24], 38: [0, 8, 0, 24], 39: [0, 1, 0, 8], 40: [0, 8, 0, 24], 41: [0, 8, 0, 24], 42: [0, 8,
0, 24], 43: [0, 8, 0, 24], 44: [0, 8, 0, 24], 45: [0, 8, 0, 24], 46: [0, 8, 0, 24], 47: [0, 8, 0, 24], 48: [0, 8,
0, 24], 49: [0, 8, 0, 24], 50: [0, 8, 0, 24], 51: [0, 1, 0, 12], 52: [0, 1, 0, 8], 53: [0, 8, 0, 24], 54: [0, 8,
```

```
0, 24], 55: [0, 8, 0, 24], 56: [0, 8, 0, 24], 57: [0, 8, 0, 24], 58: [0, 8, 0, 24], 59: [0, 8, 0, 24], 60: [0, 8,
0, 24], 61: [0, 8, 0, 24], 62: [0, 8, 0, 24], 63: [0, 8, 0, 24], 64: [0, 8, 0, 24], 65: [0, 1, 0, 8], 66: [0, 8,
0, 24], 67: [0, 8, 0, 24], 68: [0, 1, 0, 12], 69: [0, 8, 0, 24], 70: [0, 8, 0, 24], 71: [0, 8, 0, 24], 72: [0, 8,
0, 24], 73: [0, 8, 0, 24], 74: [0, 8, 0, 24], 75: [0, 8, 0, 24], 76: [0, 8, 0, 24], 77: [0, 8, 0, 24], 78: [0, 1,
0, 8], 79: [0, 8, 0, 24], 80: [0, 8, 0, 24], 81: [0, 8, 0, 24], 82: [0, 8, 0, 24], 83: [0, 8, 0, 24], 84: [0, 8,
0, 24], 85: [0, 1, 0, 12], 86: [0, 8, 0, 24], 87: [0, 8, 0, 24], 88: [0, 8, 0, 24], 89: [0, 8, 0, 24], 90: [0, 8,
0, 24], 91: [0, 1, 0, 8], 92: [0, 8, 0, 24], 93: [0, 8, 0, 24], 94: [0, 8, 0, 24], 95: [0, 8, 0, 24], 96: [0, 8,
0, 24], 97: [0, 8, 0, 24], 98: [0, 8, 0, 24], 99: [0, 8, 0, 24], 100: [0, 8, 0, 24], 101: [0, 8, 0, 24], 102: [0,
1, 0, 12], 103: [0, 8, 0, 24], 104: [0, 1, 0, 8], 105: [0, 8, 0, 24], 106: [0, 8, 0, 24], 107: [0, 8, 0, 24], 108:
[0, 8, 0, 24], 109: [0, 8, 0, 24], 110: [0, 8, 0, 24], 111: [0, 8, 0, 24], 112: [0, 8, 0, 24], 113: [0, 8, 0, 24],
114: [0, 8, 0, 24], 115: [0, 8, 0, 24], 116: [0, 8, 0, 24], 117: [0, 1, 0, 8], 118: [0, 8, 0, 24], 119: [0, 1, 0,
12], 120: [0, 8, 0, 24], 121: [0, 8, 0, 24], 122: [0, 8, 0, 24], 123: [0, 8, 0, 24], 124: [0, 8, 0, 24], 125: [0,
8, 0, 24], 126: [0, 8, 0, 24], 127: [0, 8, 0, 24], 128: [0, 8, 0, 24], 129: [0, 8, 0, 24], 130: [0, 1, 0, 8], 131:
[0, 8, 0, 24], 132: [0, 8, 0, 24], 133: [0, 8, 0, 24], 134: [0, 8, 0, 24], 135: [0, 8, 0, 24], 136: [0, 1, 0, 12],
137: [0, 8, 0, 24], 138: [0, 8, 0, 24], 139: [0, 8, 0, 24], 140: [0, 8, 0, 24], 141: [0, 8, 0, 24], 142: [0, 8, 0,
24], 143: [0, 1, 0, 8], 144: [0, 8, 0, 24], 145: [0, 8, 0, 24], 146: [0, 8, 0, 24], 147: [0, 8, 0, 24], 148: [0,
8, 0, 24], 149: [0, 8, 0, 24], 150: [0, 8, 0, 24], 151: [0, 8, 0, 24], 152: [0, 8, 0, 24], 153: [0, 1, 0, 12],
154: [0, 8, 0, 24], 155: [0, 8, 0, 24], 156: [0, 1, 0, 8], 157: [0, 8, 0, 24], 158: [0, 8, 0, 24], 159: [0, 8, 0,
24], 160: [0, 8, 0, 24], 161: [0, 8, 0, 24], 162: [0, 8, 0, 24], 163: [0, 8, 0, 24], 164: [0, 8, 0, 24], 165: [0,
8, 0, 24], 166: [0, 8, 0, 24], 167: [0, 8, 0, 24], 168: [0, 8, 0, 24], 169: [0, 1, 0, 8], 170: [0, 1, 0, 12], 171:
[0, 8, 0, 24], 172: [0, 8, 0, 24], 173: [0, 8, 0, 24], 174: [0, 8, 0, 24], 175: [0, 8, 0, 24], 176: [0, 8, 0, 24],
177: [0, 8, 0, 24], 178: [0, 8, 0, 24], 179: [0, 8, 0, 24], 180: [0, 8, 0, 24], 181: [0, 8, 0, 24], 182: [0, 1, 0,
8], 183: [0, 8, 0, 24], 184: [0, 8, 0, 24], 185: [0, 8, 0, 24], 186: [0, 8, 0, 24], 187: [0, 1, 0, 12], 188: [0,
8, 0, 24], 189: [0, 8, 0, 24], 190: [0, 8, 0, 24], 191: [0, 8, 0, 24], 192: [0, 8, 0, 24], 193: [0, 8, 0, 24],
194: [0, 8, 0, 24], 195: [0, 1, 0, 8], 196: [0, 8, 0, 24], 197: [0, 8, 0, 24], 198: [0, 8, 0, 24], 199: [0, 8, 0,
24], 200: [0, 8, 0, 24], 201: [0, 8, 0, 24], 202: [0, 8, 0, 24], 203: [0, 8, 0, 24], 204: [0, 1, 0, 12], 205: [0,
8, 0, 24], 206: [0, 8, 0, 24], 207: [0, 8, 0, 24], 208: [0, 1, 0, 8], 209: [0, 8, 0, 24], 210: [0, 8, 0, 24], 211:
[0, 8, 0, 24], 212: [0, 8, 0, 24], 213: [0, 8, 0, 24], 214: [0, 8, 0, 24], 215: [0, 8, 0, 24], 216: [0, 8, 0, 24],
217: [0, 8, 0, 24], 218: [0, 8, 0, 24], 219: [0, 8, 0, 24], 220: [0, 8, 0, 24]}
```

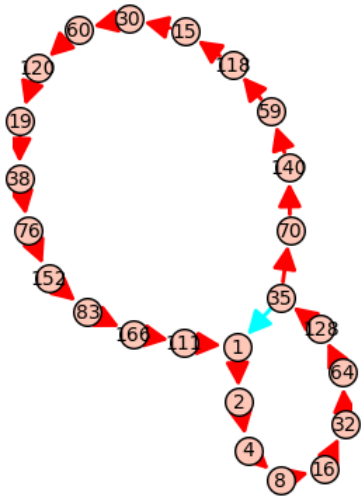
In [25]:

```
x=1
print(str(x)+':',dp[x])
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p*
q);print(y,'\t',end='')
print()
y=x;print(y,'\t',end
='')
for i in range(sum(dp[
x][2:])):
    y=f(y)%(p);
print(y,'\t',end='')
```

```
1: [0, 8, 0, 24] 1 2 4 8 16 32 64 128 35 70 140 59 118 15 30 60 120 19 38 76 152 83 166 111 1 1 2 4 8 3 6 12 11 9
5 10 7 1 2 4 8 3 6 12 11 9 5 10 7 1
```

In [26]:

```
show(
G(x,dp,p,q,f)
.plot(figsize=5,color_by_label=True
))#,vertex_size=300
```



On observe : 35-1 a un facteur commun avec 13*17, mais l'observation a demandée plus d'itérations (8) qu'avec f:x↦x^2+1 (5 itérations)

In [0]: