Maple à l'oral de mathématique

MP⋆

Guillaume CONNAN

Lycée Clemenceau

Dernière mise à jour : 19 mai 2014 à 16:54

Sommaire

Sommaire

- 30 minutes de préparation puis 30 minutes de présentation ;
- bien écouter l'examinateur - dialogue ;
- prendre l'initiative d'utiliser Maple pour des calculs compliqués mais savoir rester circonspect sur certaines réponses ;
- être conscient que quelques établissements font l'impasse sur Maple ;
- ne pas espérer découvrir des fonctions en lisant l'aide pendant la préparation ;

- 30 minutes de préparation puis 30 minutes de présentation ;
- bien écouter l'examinateur - dialogue ;
- prendre l'initiative d'utiliser Maple pour des calculs compliqués mais savoir rester circonspect sur certaines réponses ;
- être conscient que quelques établissements font l'impasse sur Maple ;
- ne pas espérer découvrir des fonctions en lisant l'aide pendant la préparation ;

- 30 minutes de préparation puis 30 minutes de présentation ;
- bien écouter l'examinateur - dialogue ;
- prendre l'initiative d'utiliser Maple pour des calculs compliqués mais savoir rester circonspect sur certaines réponses ;
- être conscient que quelques établissements font l'impasse sur Maple ;
- ne pas espérer découvrir des fonctions en lisant l'aide pendant la préparation ;

- 30 minutes de préparation puis 30 minutes de présentation ;
- bien écouter l'examinateur - dialogue ;
- prendre l'initiative d'utiliser Maple pour des calculs compliqués mais savoir rester circonspect sur certaines réponses ;
- être conscient que quelques établissements font l'impasse sur Maple ;
- ne pas espérer découvrir des fonctions en lisant l'aide pendant la préparation ;

- 30 minutes de préparation puis 30 minutes de présentation ;
- bien écouter l'examinateur - dialogue ;
- prendre l'initiative d'utiliser Maple pour des calculs compliqués mais savoir rester circonspect sur certaines réponses ;
- être conscient que quelques établissements font l'impasse sur Maple ;
- ne pas espérer découvrir des fonctions en lisant l'aide pendant la préparation ;

- N'avoir recours aux procédures que lorsque cela reste sous contrôle : inutile souvent ;
- ne pas réinventer la roue : utiliser au mieux les fonctions Maple ;
- ne pas oublier `seq` et éviter les copier-coller ;
- distinguer fonction et expression. Penser éventuellement à `unapply`. Ne pas oublier `f := x -> ...` ;
- savoir indiquer qu'une variable est entière, complexe, etc. (avec `assume` par exemple) ;
- connaître les fonctions de simplification, conversion, transformation ;

Points mal maîtrisés

construction de matrices avec fonction « définissante » des coefficients ;

```
>> M := n -> Matrix(n, n, (i,j) -> if i - j mod n = 1 then i else 0; fi);
```

```
>> M(6);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \end{bmatrix}$$

Points mal maîtrisés

construction de matrices avec fonction « définissante » des coefficients ;

```
>> M := n -> Matrix(n, n, (i,j) -> if i - j mod n = 1 then i else 0; fi);
```

```
>> M(6);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \end{bmatrix}$$

Points mal maîtrisés

construction de matrices avec fonction « définissante » des coefficients ;

```
>> M := n -> Matrix(n, n, (i,j) -> if i - j mod n = 1 then i else 0; fi);
```

```
>> M(6);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \end{bmatrix}$$

Points mal maîtrisés

construction de matrices avec fonction « définissante » des coefficients ;

```
>> M := n -> Matrix(n, n, (i,j) -> if i - j mod n = 1 then i else 0; fi);
```

```
>> M(6);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 \end{bmatrix}$$

Points mal maîtrisés

savoir obtenir des valeurs approchées des racines d'une équation sans que l'affichage d'un seul résultat garantisse l'unicité de la solution.

```
>> f := t -> exp(-0.1*t) * (0.7*sin(10*t) + 0.8*cos(10*t));
```

```
>> fsolve(f(x));
                     0.2289626326
```

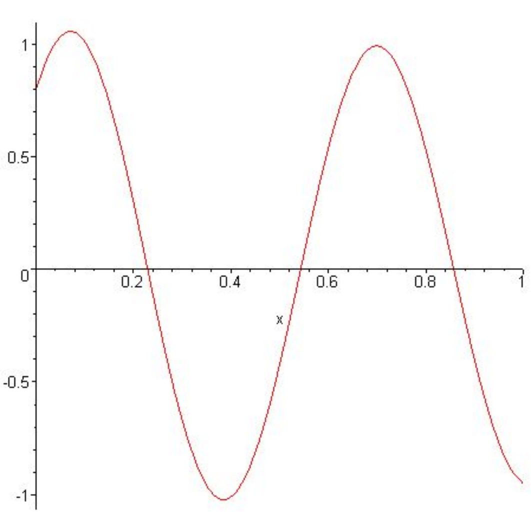
Points mal maîtrisés

savoir obtenir des valeurs approchées des racines d'une équation sans que l'affichage d'un seul résultat garantisse l'unicité de la solution.

```
>> f := t -> exp(-0.1*t) * (0.7*sin(10*t) + 0.8*cos(10*t));
```

```
>> fsolve(f(x));
                    0.2289626326
```

Points mal maîtrisés



Points mal maîtrisés

```
>> seq(fsolve(f(x), x = k), k in [0.2,0.5,0.8] );
                 0.2289626326, 0.5431218980, 0.8572811634
```

Points mal maîtrisés

Il faut savoir superposer divers types de graphiques :

```
>> P := plot(cos(x), x = -3 .. 3, color = green):
>> R := plot([seq([x, cos(x)],x in seq(k*Pi/6 , k = -5 .. 5))], color = blue,style = point, symbol =
     cross):
```

```
>> plots[display](P,R);
```

Points mal maîtrisés

Il faut savoir superposer divers types de graphiques :

```
>> P := plot(cos(x), x = -3 .. 3, color = green):
>> R := plot([seq([x, cos(x)],x in seq(k*Pi/6 , k = -5 .. 5))], color = blue,style = point, symbol =
     cross):
```

```
>> plots[display](P,R);
```

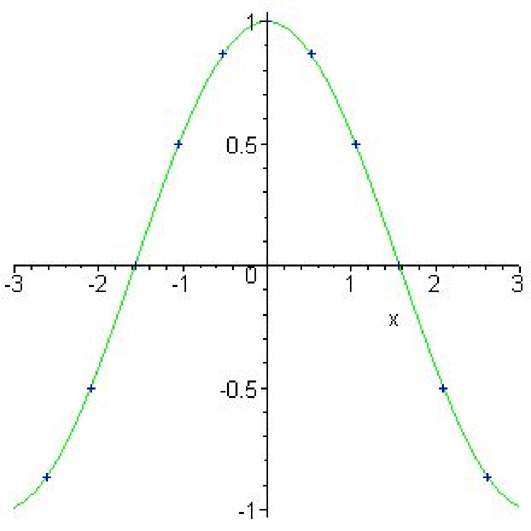
Points mal maîtrisés

Il faut savoir superposer divers types de graphiques :

```
>> P := plot(cos(x), x = -3 .. 3, color = green):
>> R := plot([seq([x, cos(x)],x in seq(k*Pi/6 , k = -5 .. 5))], color = blue,style = point, symbol =
     cross):
```

```
>> plots[display](P,R);
```

Points mal maîtrisés



Points mal maîtrisés

Dans le cas particulier des équations différentielles, beaucoup ne savent pas visualiser le graphe d'une solution lorsque le logiciel n'en donne pas une solution exacte.

Points mal maîtrisés

```
>> eq := diff(u(t),t) = (u(t))^2 + cos(u(t)) + sinh(cosh(u(t)));
>> dsolve({eq,u(0)=1},u(t));
```

```
u(t) = RootOf(t-Int(1/(_a^2+cos(_a)+sinh(cosh(_a))),_a = 0 .. _Z)+Int(1/(_a^2+cos(_a)+sinh(cosh(_a))),
      _a = 0 .. 1))
```

Points mal maîtrisés

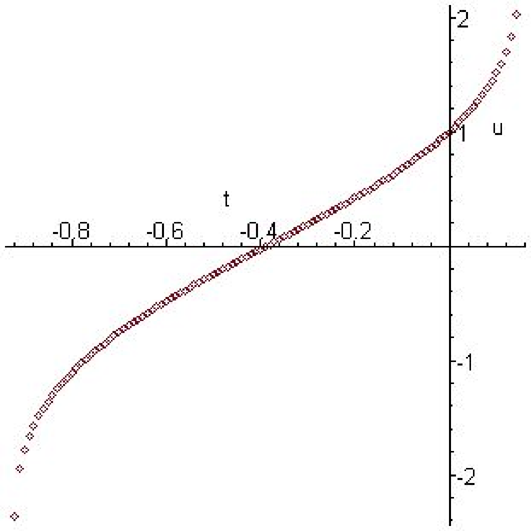
```
>> eq := diff(u(t),t) = (u(t))^2 + cos(u(t)) + sinh(cosh(u(t)));
>> dsolve({eq,u(0)=1},u(t));
```

```
u(t) = RootOf(t-Int(1/(_a^2+cos(_a)+sinh(cosh(_a))),_a = 0 .. _Z)+Int(1/(_a^2+cos(_a)+sinh(cosh(_a))),
      _a = 0 .. 1))
```

points mal maîtrisés

```
>> S := dsolve({eq,u(0)=1},u(t), type=numeric);
>> plots[odeplot](S,[t,u(t)],-1..1,style=point);
```

Points mal maîtrisés



```
Warning, cannot evaluate the solution further right of .14882443, probably a singularity

Warning, cannot evaluate the solution further left of -.92151725, probably a singularity
```

Points mal maîtrisés

Proscrire l'ouverture et l'usage simultanés des librairies `linalg` et `LinearAlgebra`.

Points mal maîtrisés

Connaître les inconvénients et avantages respectifs des commandes `sum` et add.

The add and mul commands are related to the seq command, which is related to the for-loop construct.

In either form, the add and mul commands are generally more efficient than the for-loop versions because the for-loop versions construct many intermediate sums and products.

Note that the limits m and n must evaluate to numerical constants, that is, integers, fractions, or floating point numbers. If you want to compute a formula for a symbolic sum or product in terms of a symbolic limit, use the sum and product commands. As a special case, m may evaluate to infinity, or n may evaluate to -infinity. If m is greater than n, add returns 0 and mul returns 1.

Points mal maîtrisés

Connaître les inconvénients et avantages respectifs des commandes `sum` et add.

The add and mul commands are related to the seq command, which is related to the for-loop construct.

In either form, the add and mul commands are generally more efficient than the for-loop versions because the for-loop versions construct many intermediate sums and products.

Note that the limits m and n must evaluate to numerical constants, that is, integers, fractions, or floating point numbers. If you want to compute a formula for a symbolic sum or product in terms of a symbolic limit, use the sum and product commands. As a special case, m may evaluate to infinity, or n may evaluate to -infinity. If m is greater than n, add returns 0 and mul returns 1.

Points mal maîtrisés

```
>> add( i^2, i=0..n );
Error, unable to execute add
>> sum( i^2, i=0..n );
                              3               2
              1/3 (n + 1)  - 1/2 (n + 1)  + 1/6 n + 1/6
```

```
>> M:=k -> Matrix([[k,0],[0,k]]);
>> add(M(k),k=1..5);
                           [15     0]
                           [        ]
                           [ 0    15]
```

Points mal maîtrisés

```
>> add( i^2, i=0..n );
Error, unable to execute add
>> sum( i^2, i=0..n );
                          3               2
             1/3 (n + 1)  - 1/2 (n + 1)  + 1/6 n + 1/6
```

```
>> M:=k -> Matrix([[k,0],[0,k]]);
>> add(M(k),k=1..5);
                              [15     0]
                              [        ]
                              [ 0    15]
```

Savoir calculer le quotient, le reste dans une division euclidienne dans $\mathbf{Z}$, dans $\mathbf{Q}[X]$

```
>> irem(19,5);
                                        4
>> iquo(19,5);
                                        3
>> 19 mod 5;
                                        4
```

Savoir calculer le quotient, le reste dans une division euclidienne dans $\mathbf{Z}$, dans $\mathbf{Q}[X]$

```
>> irem(19,5);
                                        4
>> iquo(19,5);
                                        3
>> 19 mod 5;
                                        4
```

```
>> divide(x^3 + x + 1, x^2 + x + 1);
                                false
>> quo(x^3 + x + 1, x^2 + x + 1,x);
                                x - 1
>> r1:=rem(x^3 + x + 1, x^2 + x + 1, x, 'q1');
                             r1 := 2 + x
>> q1;
                                x - 1
>> a:=(x^2 + x + 1)*q1 + r1;
                  a := (x^2  + x + 1) (x - 1) + 2 + x
>> simplify(a);
                              x^3  + x + 1
```

Savoir tester qu'un entier est premier, savoir travailler modulo n ;

```
ifactor, isprime, ithprime, igcd, ilcm
```

Savoir tester qu'un entier est premier, savoir travailler modulo n ;

```
ifactor, isprime, ithprime, igcd, ilcm
```

```
>> modp(12, 7)
               5
>> 12 mod 7
               5
>> mods(12, 7)
              -2
>> 1/3 mod 7
               5
>> 5*3 mod 7
               1
>> 5 &^ 1000 mod 100
              25
```

```
>> with(numtheory);
[GIgcd, bigomega, cfrac, cfracpol, cyclotomic, divisors, factorEQ, factorset, fermat, imagunit, index,
 integral_basis, invcfrac, invphi, iscyclotomic, issqrfree, ithrational, jacobi, kronecker,
 lambda, legendre, mcombine,mersenne, migcdex, minkowski, mipolys, mlog, mobius, mroot, msqrt,
 nearestp, nthconver, nthdenom, nthnumer, nthpow, order, pdexpand, phi, pi, pprimroot, primroot,
 quadres, rootsunity, safeprime, sigma, sq2factor,sum2sqr, tau, thue]
```

Savoir factoriser (dans $\mathbf{Q}[X]$ et éventuellement dans une extension simple suggérée par l'énoncé), développer, ordonner un polynôme ;

```
>> factor(x^4 + 1);
                              x^4 + 1
>> factor(x^4 + 1) mod 2;
                              x^4 + 1
>> Factor(x^4 + 1) mod 2;
                              (x + 1)^4
>> Quo(x^2 + 1,x + 1,x) mod 2;
                              x + 1
>> Rem(x^2 + 1,x + 1,x) mod 2;
                              0
```

```
>> Expand((x + 1)^2) mod 2;
                                x^2 + 1
>> expand((x+1)^2);
                                x^2 + x + 1
```

```
>> f := x*(x+1)+y*(x+1);
>> g := collect(f, x);
                              g := x^2 + (1 + y)x + y
>> coeff(g, x, 1);
                              1 + y
>> coeffs(g,x);
                              y, 1, 1 + y
>> PolynomialTools[CoefficientList](g,x)
                              [y, 1 + y, 1]
```

```
>> with(PolynomialTools);
[AnnihilatingPolynomial, CoefficientList, CoefficientVector, FromCoefficientList,
      FromCoefficientVector, GcdFreeBasis, GreatestFactorialFactorization, Hurwitz, IsSelfReciprocal,
      MinimalPolynomial, PDEToPolynomial, PolynomialToPDE, ShiftEquivalent, ShiftlessDecomposition,
      Shorten, Shorter, Sort, Split, Splits,SquareFreePart, Translate]
```

Savoir calculer le pgcd de deux entiers, de deux polynômes ;

```
>> igcd(36,24);
                                12
>> gcd(x^3 + 1, x^2 + 1);
                                1
>> Gcd(x^3 + 1, x^2 + 1) mod 2;
                                x + 1
```

Savoir calculer le pgcd de deux entiers, de deux polynômes ;

```
>> igcd(36,24);
                                12
>> gcd(x^3 + 1, x^2 + 1);
                                1
>> Gcd(x^3 + 1, x^2 + 1) mod 2;
                                x + 1
```

Savoir obtenir un couple donnant la relation de Bézout ;

```
>> igcdex(14,27,'u','v');
                                   1
>> u;
                                   2
>> v;
                                   -1
>> 1/14 mod 27;
                                   2
>> 14*u + 27*v;
                                   1
```

Savoir obtenir un couple donnant la relation de Bézout ;

```
>> igcdex(14,27,'u','v');
                              1
>> u;
                              2
>> v;
                              -1
>> 1/14 mod 27;
                              2
>> 14*u + 27*v;
                              1
```

```
>> gcdex(x^3 + 1, x^2 + 1, x, 'U', 'V');
                                                      1

>> U;
                                                  x/2 + 1/2

>> V;
                                                  2
                                            -1/2 x  - 1/2 x + 1/2
>> Gcdex(x^3 + 1, x^2 + 1, x, 'U', 'V') mod 2;
                                                    x + 1

>> U;
                                                      1

>> V;
                                                      x
```

Savoir déterminer les racines d'une équation (algébrique ou non) de façon exacte, de façon approchée ; savoir déterminer une valeur approchée d'une racine localisée dans un intervalle ;

```
>> solve( (a^2*c^2 - 4*b^2)/b = a^6*b - 4*a^3*b, c );
                                            3            3
                                        b (a  - 2)   b (a  - 2)
                                        ----------, - ----------
                                            a            a

>> solve( {(a^2*c^2 - 4*b^2)/b = a^6*b - 4*a^3*b}, [c] );
                                           3                   3
                                       b (a  - 2)          b (a  - 2)
                                 [[c = ----------], [c = - ----------]]
                                           a                   a
```

Savoir déterminer les racines d'une équation (algébrique ou non) de façon exacte, de façon approchée ; savoir déterminer une valeur approchée d'une racine localisée dans un intervalle ;

```
>> solve( (a^2*c^2 - 4*b^2)/b = a^6*b - 4*a^3*b, c );
                                              3            3
                                          b (a  - 2)    b (a  - 2)
                                          ----------, - ----------
                                              a            a

>> solve( {(a^2*c^2 - 4*b^2)/b = a^6*b - 4*a^3*b}, [c] );
                                             3                  3
                                         b (a  - 2)         b (a  - 2)
                                   [[c = ----------], [c = - ----------]]
                                             a                  a
```

```
>> solve( {x + y < 10, x^2 = 9}, [x, y] );
                                 [[x = -3, y < 13], [x = 3, y < 7]]
```

```
>> solve(x^2=a,x) assuming a::negative;
                                         1/2            1/2
                                     (-a)    I, -I (-a)
```

```
>> solve( {x + y < 10, x^2 = 9}, [x, y] );
                                   [[x = -3, y < 13], [x = 3, y < 7]]
```

```
>> solve(x^2=a,x) assuming a::negative;
                                                 1/2          1/2
                                            (-a)    I, -I (-a)
```

RootOf

Un coup d'œil dans les extensions de corps...

```
>> alias( a = RootOf(z^4 - z^2 + 1) );
>> a^4;
                                     4
                                    a

>> evala(a^4);
                                   2
                                  a  - 1

>> evala(a^6);
                                    -1

>> evala(1/a);
                                   3
                                 -a  + a

>> allvalues(a);
          1/2 1/2             1/2 1/2             1/2 1/2            1/2 1/2
 (2 + 2 I 3   )       (2 - 2 I 3   )       (2 + 2 I 3   )    (2 - 2 I 3   )
 -----------------, - -----------------, - -----------------, -----------------
         2                   2                   2                  2
```

RootOf

Un coup d'œil dans les extensions de corps...

```
>> alias( a = RootOf(z^4 - z^2 + 1) );
>> a^4;
                                                4
                                               a

>> evala(a^4);
                                              2
                                             a  - 1

>> evala(a^6);
                                               -1

>> evala(1/a);
                                              3
                                           -a  + a

>> allvalues(a);
                  1/2 1/2             1/2 1/2             1/2 1/2            1/2 1/2
          (2 + 2 I 3   )      (2 - 2 I 3   )      (2 + 2 I 3   )     (2 - 2 I 3   )
          -----------------, - -----------------, - -----------------, -----------------
                  2                   2                   2                  2
```

RootOf

```
>> factor(x^4 - x^2 + 1);
                                                      4    2
                                                     x  - x  + 1
>> factor(x^4 - x^2 + 1, a);
                                     3            3
                                   (a  - a - x) (a  - a + x) (x + a) (-x + a)
>> e := factor(x^4 - x^2 + 1, a);
                                     3            3
                              e := (a  - a - x) (a  - a + x) (x + a) (-x + a)
>> expand(e);
                              6  2    8      4  2      6      2  2    4    4
                            -a  x  + a  + 2 a  x  - 2 a  - 2 x  a  + a  + x
>> c := collect(e,x);
                             4       3     2    2   2     3     2  2
                      c := x  + (-(a  - a)  - a ) x  + (a  - a)  a
>> evala(c);
                                        4    2
                                       x  - x  + 1
```

Savoir décomposer une fraction rationnelle en éléments simples dans $\mathbf{Q}(X)$ (éventuellement dans une extension simple de $\mathbf{Q}$ suggérée par l'énoncé).

```
>>  f := (x-2) / ((x+1)^2 * (x-1)^2) :
>> convert(f,parfrac,x) ;
                                   3             1           1            1
                             - ---------- - --------- + --------- - ----------
                                        2   2 (x + 1)   2 (x - 1)            2
                               4 (x + 1)                            4 (x - 1)
```

Savoir décomposer une fraction rationnelle en éléments simples dans **Q**(X) (éventuellement dans une extension simple de **Q** suggérée par l'énoncé).

```
>>  f := (x-2) / ((x+1)^2 * (x-1)^2) :
>> convert(f,parfrac,x) ;
                                3            1           1           1
                           - ---------- - --------- + --------- - ----------
                                      2   2 (x + 1)   2 (x - 1)            2
                             4 (x + 1)                            4 (x - 1)
```

```
>> g := (x-2) / ((x+a)^2 * (x-b)^2) :
>> convert(g,parfrac,x) ;
             b - 2                -b + 4 + a           -a - 2              b - 4 - a
        ----------------- + ---------------- + ----------------- + ----------------
               2        2              3                  2        2                3
        (b + a)  (x - b)    (b + a)  (x - b)   (x + a)  (b + a)    (x + a) (b + a)
```

```
>> h := (x-2) / ((x^2+1)^2 * (x-1)^2) :
>> convert(h,parfrac,x) ;
                               -3 x - 2        3           1         -2 x - 1
                              ---------- + --------- - ---------- + -----------
                                  2        4 (x - 1)            2       2     2
                              4 (x   + 1)              4 (x - 1)    2 (x   + 1)
```

```
>> convert(h,parfrac,x,I) ;
                 3/8 + 3/8 I   1/8 + 1/4 I   3/8 - 3/8 I   1/8 - 1/4 I       3           1
               - ----------- + ----------- + ----------- + ----------- + --------- - ----------
                    x + I               2      -x + I               2     4 (x - 1)            2
                                 (-x + I)                    (x + I)                 4 (x - 1)
```

```
>> h := (x-2) / ((x^2+1)^2 * (x-1)^2) :
>> convert(h,parfrac,x) ;
 -3 x - 2        3           1         -2 x - 1
---------- + --------- - ---------- + -----------
    2        4 (x - 1)            2       2     2
4 (x  + 1)               4 (x - 1)    2 (x  + 1)
```

```
>> convert(h,parfrac,x,I) ;
  3/8 + 3/8 I   1/8 + 1/4 I   3/8 - 3/8 I   1/8 - 1/4 I       3           1
- ----------- + ----------- + ----------- + ----------- + --------- - ----------
     x + I               2      -x + I               2    4 (x - 1)            2
                 (-x + I)                     (x + I)               4 (x - 1)
```

```
>> k := (4*x^3-6*x^2-2)/(x^4-2*x^3-2*x+4);
                                     3      2
                                  4 x  - 6 x  - 2
                            k := -------------------
                                   4      3
                                  x  - 2 x  - 2 x + 4
>> convert(k,parfrac,x);
                                      2
                                   3 x       1
                                  ------ + -----
                                   3       x - 2
                                  x  - 2
```

```
>> convert(k,parfrac,x,2^(1/3));
```

$$\frac{\left(2^{2/3}-1\right)\sqrt[3]{2}}{\left(-2+\sqrt[3]{2}\right)\left(-x+\sqrt[3]{2}\right)}-6\,\frac{1}{\left(-2+\sqrt[3]{2}\right)\left(2^{2/3}+2\sqrt[3]{2}+4\right)(x-2)}+\frac{-\sqrt[3]{2}-2x}{-2^{2/3}-x\sqrt[3]{2}-x^2}$$

```
>> k := (4*x^3-6*x^2-2)/(x^4-2*x^3-2*x+4);
                                                  3      2
                                               4 x  - 6 x  - 2
                                          k := ------------------
                                                4      3
                                               x  - 2 x  - 2 x + 4
>> convert(k,parfrac,x);
                                                  2
                                               3 x       1
                                               ------ + -----
                                                3       x - 2
                                               x  - 2
```

```
>> convert(k,parfrac,x,2^(1/3));
```

$$\frac{\left(2^{2/3}-1\right)\sqrt[3]{2}}{\left(-2+\sqrt[3]{2}\right)\left(-x+\sqrt[3]{2}\right)}-6\,\frac{1}{\left(-2+\sqrt[3]{2}\right)\left(2^{2/3}+2\,\sqrt[3]{2}+4\right)(x-2)}+\frac{-\sqrt[3]{2}-2\,x}{-2^{2/3}-x\,\sqrt[3]{2}-x^2}$$

Savoir calculer des produits matriciels, créer une matrice diagonale et a fortiori la matrice identité, former la transposée ;

```
>> with(LinearAlgebra);
with(LinearAlgebra);
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix,BidiagonalForm, BilinearForm,
    CARE, CharacteristicMatrix,CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation,
    ColumnSpace, CompanionMatrix, CompressedSparseForm, ConditionNumber,ConstantMatrix,
    ConstantVector, Copy, CreatePermutation, CrossProduct,DARE, DeleteColumn, DeleteRow, Determinant
    , Diagonal, DiagonalMatrix,Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues
    ,Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm,FromCompressedSparseForm, FromSplitForm,
    GaussianElimination,GenerateEquations, GenerateMatrix, Generic, GetResultDataType,GetResultShape
    , GivensRotationMatrix, GramSchmidt, HankelMatrix,HermiteForm, HermitianTranspose,
    HessenbergForm, HilbertMatrix,HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite,
    IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm,KroneckerProduct, LA_Main,
    LUDecomposition, LeastSquares, LinearSolve,LyapunovSolve, Map, Map2, MatrixAdd,
    MatrixExponential, MatrixFunction,MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower,
    MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor,Modular, Multiply,
    NoUserValue, Norm, Normalize, NullSpace,OuterProductMatrix, Permanent, Pivot, PopovForm,
    ProjectionMatrix,QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm,
    ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace,ScalarMatrix, ScalarMultiply,
    ScalarVector, SchurForm, SingularValues,SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix
    , SubVector,SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace,Transpose,
    TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd,VectorAngle, VectorMatrixMultiply,
    VectorNorm, VectorScalarMultiply,ZeroMatrix, ZeroVector, Zip]
```

```
>> M := <<1|2|3> , <4|5|6>>;
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := <<1,4> | <2,5> | <3,6>>;
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := Matrix(2,3,[1,2,3,4,5,6]);
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := Matrix(2,3, (i,j) -> j + 3*(i - 1));
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := <<1|2|3> , <4|5|6>>;
                                    [1    2    3]
                               M := [           ]
                                    [4    5    6]
```

```
>> M := <<1,4> | <2,5> | <3,6>>;
                                    [1    2    3]
                               M := [           ]
                                    [4    5    6]
```

```
>> M := Matrix(2,3,[1,2,3,4,5,6]);
                                    [1    2    3]
                               M := [           ]
                                    [4    5    6]
```

```
>> M := Matrix(2,3, (i,j) -> j + 3*(i - 1));
                                    [1    2    3]
                               M := [           ]
                                    [4    5    6]
```

```
>> M := <<1|2|3> , <4|5|6>>;
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := <<1,4> | <2,5> | <3,6>>;
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := Matrix(2,3,[1,2,3,4,5,6]);
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := Matrix(2,3, (i,j) -> j + 3*(i - 1));
                         [1    2    3]
                    M := [           ]
                         [4    5    6]
```

```
>> M := <<1|2|3> , <4|5|6>>;
                                  [1    2    3]
                             M := [           ]
                                  [4    5    6]
```

```
>> M := <<1,4> | <2,5> | <3,6>>;
                                  [1    2    3]
                             M := [           ]
                                  [4    5    6]
```

```
>> M := Matrix(2,3,[1,2,3,4,5,6]);
                                  [1    2    3]
                             M := [           ]
                                  [4    5    6]
```

```
>> M := Matrix(2,3, (i,j) -> j + 3*(i - 1));
                                  [1    2    3]
                             M := [           ]
                                  [4    5    6]
```

```
>>  M := RandomMatrix(3);
                                  [14     99    -20]
                                  [                ]
                             M := [16     60    -25]
                                  [                ]
                                  [ 9    -95     51]
>>  N := RandomMatrix(3);
                                 [ 42     12    -68]
                                 [                 ]
                            N := [ 18    -62    -67]
                                 [                 ]
                                 [-59    -33     22]
>> M . N;
                            [ 3550    -5310    -8025]
                            [                       ]
                            [ 3227    -2703    -5658]
                            [                       ]
                            [-4341     4315     6875]
```

```
>> Eye := IdentityMatrix(3);
                                  [1    0    0]
                                  [           ]
                           Eye := [0    1    0]
                                  [           ]
                                  [0    0    1]
```

```
>> P := DiagonalMatrix([1,2,3]);
                                  [1    0    0]
                                  [           ]
                             P := [0    2    0]
                                  [           ]
                                  [0    0    3]
```

```
>> Eye := IdentityMatrix(3);
                        [1    0    0]
                        [           ]
                 Eye := [0    1    0]
                        [           ]
                        [0    0    1]
```

```
>> P := DiagonalMatrix([1,2,3]);
                        [1    0    0]
                        [           ]
                   P := [0    2    0]
                        [           ]
                        [0    0    3]
```

```
>> M := RandomMatrix(3);
                                   [27    99     92]
                                   [               ]
                              M := [ 8    29    -31]
                                   [               ]
                                   [69    44     67]
```

```
>> Transpose(M);
                                   [27      8    69]
                                   [               ]
                                   [99     29    44]
                                   [               ]
                                   [92    -31    67]
```

```
>> M := RandomMatrix(3);
                                     [27    99     92]
                                     [               ]
                                M := [ 8    29    -31]
                                     [               ]
                                     [69    44     67]
```

```
>> Transpose(M);
                                   [27      8    69]
                                   [               ]
                                   [99     29    44]
                                   [               ]
                                   [92    -31    67]
```

Savoir calculer le rang, le noyau ou l'image (en obtenant une base de ces sous-espaces) ;

```
>> M := Matrix(3,3, (i,j) -> j + 3*(i - 1));
                                   [1    2    3]
                                   [           ]
                              M := [4    5    6]
                                   [           ]
                                   [7    8    9]
>> Rank(M);
                                       2
>> NullSpace(M);
                                     [ 1]
                                     [  ]
                                    {[-2]}
                                     [  ]
                                     [ 1]
>> ColumnSpace(M);
                                   [ 1]  [0]
                                   [  ]  [ ]
                                  [[ 0], [1]]
                                   [  ]  [ ]
                                   [-1]  [2]
```

Calcul modulaire

On peut calculer dans $\mathfrak{M}_n(\mathbb{Z}/p\mathbb{Z})$ mais attention aux conflits.

```
>> with(LinearAlgebra[Modular]);
[AddMultiple, Adjoint, BackwardSubstitute, Basis, CharacteristicPolynomial,ChineseRemainder, Copy
     , Create, Determinant, Fill, ForwardSubstitute, Identity, IntegerCharacteristicPolynomial,
     IntegerDeterminant,IntegerLinearSolve, Inverse, LUApply, LUDecomposition, LinearSolve, MatBasis,
      MatGcd, MatrixPower, Mod,Multiply, Permute, Random, Rank, RankProfile, RowEchelonTransform,
     RowReduce, Swap, Transpose, ZigZag]
```

```
>> M := Matrix(3,3,(i,j)->rand());
                  [790383660726    133373034524    418188244309]
                  [                                            ]
             M := [300701008180    114772485963    698490191419]
                  [                                            ]
                  [692395119931    968558890305    386332534580]
>> Mod(26,M,integer[]);
                              [10     2    17]
                              [              ]
                              [14     5     7]
                              [              ]
                              [21    19    18]
>> Determinant(26,M);
                                     17
>> with(LinearAlgebra):
>> Determinant(M);
                   -362098780918597362137312644293052871

>> Determinant(M) mod 26;
                                     17
```

Savoir calculer le déterminant, éventuellement l'inverse, la comatrice (ou sa transposée) d'une matrice carrée ;

```
>> M := RandomMatrix(3);
                                      [-23    -18    -92]
                                      [                 ]
                                 M := [-73      7     91]
                                      [                 ]
                                      [-34    -64     48]
>> Determinant(M);
                                          -600780
>> 1/M;
                                [-308     -1688      497   ]
                                [-----    ------    ------]
                                [30039    150195    300390]
                                [                          ]
                                [ -41      1058     -8809 ]
                                [-----    ------    ------]
                                [60078    150195    600780]
                                [                          ]
                                [-491      43        295   ]
                                [-----    -----     ------]
                                [60078    30039     120156]
>> M . (1/M);
                                       [1    0    0]
                                       [           ]
                                       [0    1    0]
                                       [           ]
                                       [0    0    1]
```

```
>> M := RandomMatrix(3);
                                     [97     98     36]
                                     [                ]
                                M := [ 9    -95    -26]
                                     [                ]
                                     [30     31     73]
>> A := Adjoint(M);
                                [-6129    -6038       872]
                                [                        ]
                           A := [-1437     6001      2846]
                                [                        ]
                                [ 3129      -67    -10097]
>> M . A;
                           [-622695          0          0]
                           [                             ]
                           [      0    -622695          0]
                           [                             ]
                           [      0          0    -622695]

>> Determinant(M);
                                       -622695
```

Savoir calculer le polynôme caractéristique d'une matrice carrée, ses valeurs propres, ses vecteurs propres ;

```
>> M := <<1|1|0>, <0|1|3>, <0|0|2>>;
                                   [1    1    0]
                                   [           ]
                              M := [0    1    3]
                                   [           ]
                                   [0    0    2]
>> CharacteristicPolynomial(M,X);
                               3      2
                              X  - 4 X  + 5 X - 2
>> solve(%,X);
                                    2, 1, 1
>> Eigenvalues(M);
                                      [2]
                                      [ ]
                                      [1]
                                      [ ]
                                      [1]
```

```
>> vals ,Vecs := Eigenvectors(M);
                                        [1]  [1    0    3]
                                        [ ]  [           ]
                          vals, Vecs := [1], [0    0    3]
                                        [ ]  [           ]
                                        [2]  [0    0    1]
```

```
>> M . Vecs[1..-1,3] = vals[3] . Vecs[1..-1,3];
                                      [6]   [6]
                                      [ ]   [ ]
                                      [6] = [6]
                                      [ ]   [ ]
                                      [2]   [2]
```

```
>> vals ,Vecs := Eigenvectors(M);
                                      [1]  [1    0    3]
                                      [ ]  [           ]
                        vals, Vecs := [1], [0    0    3]
                                      [ ]  [           ]
                                      [2]  [0    0    1]
```

```
>> M . Vecs[1..-1,3] = vals[3] . Vecs[1..-1,3];
                                    [6]   [6]
                                    [ ]   [ ]
                                    [6] = [6]
                                    [ ]   [ ]
                                    [2]   [2]
```

Savoir résoudre une équation d'inconnue matricielle (après l'avoir transformée en un ensemble d'équations scalaires d'inconnues les coefficients) ;

```
>>  solve({x-y = b, a*x+b*y = 3}, [x, y]) ;
                                    2
                                   b   + 3          a b - 3
                             [[x = ------, y = - -------]]
                                   a + b             a + b
>> M := <<1|-1|b>, <a|b|3>>;
                                       [1    -1    b]
                                  M := [            ]
                                       [a    b     3]
>> LinearSolve(M);
                                      [   2        ]
                                      [  b   + 3   ]
                                      [  ------    ]
                                      [  a + b     ]
                                      [            ]
                                      [    a b - 3]
                                      [- -------]
                                      [     a + b ]
```

```
>>  solve({x - y + z = 1, 2*x - 4*y + z = 3}, [x, y, z]) ;
                    [[x = 2 + 3 y, y = y, z = -1 - 2 y]]
>> M := <<1|-1|1|1>, <2|-4|1|3>>;
                                [1    -1    1    1]
                           M := [                 ]
                                [2    -4    1    3]
>> LinearSolve(M,free='k');
                                 [2 + 3 k[2] ]
                                 [           ]
                                 [   k[2]    ]
                                 [           ]
                                 [-1 - 2 k[2]]
```

Savoir calculer le produit scalaire, le produit vectoriel de deux vecteurs de $\boldsymbol{R}^3$.

```
>> U := <1,2,3,x>;
                                        [1]
                                        [ ]
                                        [2]
                                   U := [ ]
                                        [3]
                                        [ ]
                                        [x]
>> V := <4,3,x,1>:
>> Transpose(U) . V;
                                   10 + 4 x
>> DotProduct(U,V);
                                             _
                                 10 + 3 x + x
>> DotProduct(U,V,conjugate=false);
                                   10 + 4 x
```

```
>>  V1 := <1, 2, 3>;
                                       [1]
                                       [ ]
                                 V1 := [2]
                                       [ ]
                                       [3]
>> V2 := <2,3,4>;
                                       [2]
                                       [ ]
                                 V2 := [3]
                                       [ ]
                                       [4]
>> W := V1 &x V2;
                                      [-1]
                                      [  ]
                                 W := [ 2]
                                      [  ]
                                      [-1]
> map(V -> Transpose(W) . V , [V1,V2]);
                                  [0, 0]
```

Savoir composer des fonctions (ou des opérateurs), calculer des dérivées d'ordre supérieur à un ;

```
>> f := x -> exp(x);
                              f := x -> exp(x)
>> g := x -> x^2;
                                           2
                                g := x -> x
>> h := f@g;
                                  h := f@g
>> k := g@f;
                                  k := g@f
>> diff(h(x),x);
                                          2
                                2 x exp(x )
>> diff(k(x),x);
                                          2
                                  2 exp(x)
>> D(h)(x);
                                          2
                                2 x exp(x )
>> D(k)(x);
                                          2
                                  2 exp(x)
>> (D@@2)(h)(x);
                                 2   2           2
                          4 exp(x ) x  + 2 exp(x )
>> diff(h(x),x$2);
                                 2   2           2
                          4 exp(x ) x  + 2 exp(x )
```

Savoir calculer un développement limité, savoir extraire la partie régulière d'un tel développement ;

```
>> series(h(x),x);
                                2        4      6
                           1 + x  + 1/2 x  + O(x )
>> series(h(x),x,12);
                  2        4        6         8          10      12
             1 + x  + 1/2 x  + 1/6 x  + 1/24 x  + 1/120 x   + O(x  )
>> convert(series(h(x),x,12),polynom);
                      2        4        6         8          10
                 1 + x  + 1/2 x  + 1/6 x  + 1/24 x  + 1/120 x
>> P := unapply(convert(series(h(x),x,12),polynom),x);
                           2        4        6         8          10
            P := x -> 1 + x  + 1/2 x  + 1/6 x  + 1/24 x  + 1/120 x
>> P(3);
                                     18749
                                     -----
                                      20
```

Savoir calculer une intégrale de façon exacte, de façon approchée, faire un changement de variable ou une intégration par parties ;

```
>> int(exp(-x^2),x=0..infinity);
                                           1/2
                                         Pi
                                         -----
                                           2

>> int(exp(-x^2),x=0..1);
                                                1/2
                                   1/2 erf(1) Pi

>> evalf(int(exp(-x^2),x=0..1));
                                       0.7468241330
```

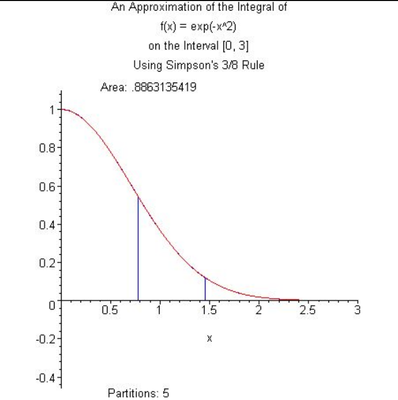
```
>> series(erf(x),x,2);
                                      2           3
                                    ----- x + O(x )
                                      1/2
                                    Pi
>> diff(erf(x),x);
                                              2
                                      2 exp(-x )
                                      ----------
                                          1/2
                                        Pi
```

```
>> int(exp(-x^2),x=0..infinity);
                                        1/2
                                      Pi
                                      -----
                                        2

>> int(exp(-x^2),x=0..1);
                                             1/2
                               1/2 erf(1) Pi

>> evalf(int(exp(-x^2),x=0..1));
                                  0.7468241330
```

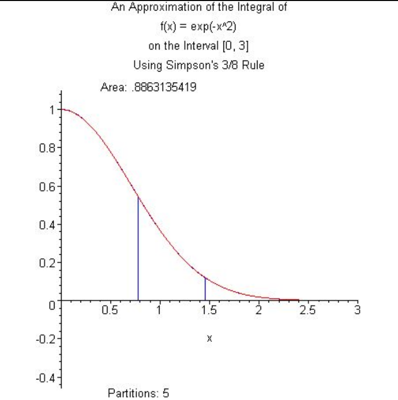
```
>> series(erf(x),x,2);
                                 2           3
                               ----- x + O(x )
                                 1/2
                               Pi
>> diff(erf(x),x);
                                         2
                                 2 exp(-x )
                                 ----------
                                     1/2
                                   Pi
```

```
>> with(Student[Calculus1]);
  [AntiderivativePlot, AntiderivativeTutor, ApproximateInt,ApproximateIntTutor, ArcLength,
        ArcLengthTutor, Asymptotes,Clear, CriticalPoints, CurveAnalysisTutor, DerivativePlot,
        DerivativeTutor, DiffTutor, ExtremePoints, FunctionAverage,FunctionAverageTutor, FunctionChart
        , FunctionPlot, GetMessage,GetNumProblems, GetProblem, Hint, InflectionPoints, IntTutor,
        Integrand, InversePlot, InverseTutor, LimitTutor,MeanValueTheorem, MeanValueTheoremTutor,
        NewtonQuotient,NewtonsMethod, NewtonsMethodTutor, PointInterpolation,RiemannSum, RollesTheorem
        , Roots, Rule, Show, ShowIncomplete,ShowSteps, Summand, SurfaceOfRevolution,
        SurfaceOfRevolutionTutor, Tangent, TangentSecantTutor,TangentTutor, TaylorApproximation,
        TaylorApproximationTutor,Understand, Undo, VolumeOfRevolution, VolumeOfRevolutionTutor,
        WhatProblem]
```

```
>> ApproximateInt(4*sqrt(1-x^2), 0..1, output=animation, partition=random[1.0], refinement=random,
     subpartition=width, iterations=50, showpoints=false);
```



```
>> ApproximateInt(4*sqrt(1-x^2), 0..1, output=animation, partition=random[1.0], refinement=random,
     subpartition=width, iterations=50, showpoints=false);
```



```
>> with(student);
[D, Diff, Doubleint, Int, Limit, Lineint, Product, Sum, Tripleint, changevar,completesquare, distance,
      equate, integrand, intercept, intparts, leftbox,leftsum, makeproc, middlebox, middlesum,
     midpoint, powsubs, rightbox,rightsum, showtangent, simpson, slope, summand, trapezoid]
```

```
>>  intparts(Int(exp(x)*x,x = 1 .. A),x);
                                              A
                                             /
                                            |
                      exp(A) A - exp(1) -   |   exp(x) dx
                                            |
                                           /
                                             1
```

```
>> changevar(t=F(x),Int(G(t),t),x);
                             /
                            |          /d      \
                            |  G(F(x)) |-- F(x)| dx
                            |          \dx     /
                           /
```

```
>> with(student);
[D, Diff, Doubleint, Int, Limit, Lineint, Product, Sum, Tripleint, changevar,completesquare, distance,
      equate, integrand, intercept, intparts, leftbox,leftsum, makeproc, middlebox, middlesum,
      midpoint, powsubs, rightbox,rightsum, showtangent, simpson, slope, summand, trapezoid]
```

```
>>  intparts(Int(exp(x)*x,x = 1 .. A),x);
                                             A
                                            /
                                           |
                 exp(A) A - exp(1) -       |   exp(x) dx
                                           |
                                          /
                                            1
```

```
>> changevar(t=F(x),Int(G(t),t),x);
                         /
                        |          /d      \
                        |  G(F(x)) |-- F(x)| dx
                        |          \dx     /
                       /
```

```
>> with(student);
[D, Diff, Doubleint, Int, Limit, Lineint, Product, Sum, Tripleint, changevar,completesquare, distance,
       equate, integrand, intercept, intparts, leftbox,leftsum, makeproc, middlebox, middlesum,
      midpoint, powsubs, rightbox,rightsum, showtangent, simpson, slope, summand, trapezoid]
```

```
>>  intparts(Int(exp(x)*x,x = 1 .. A),x);
                                               A
                                              /
                                             |
                     exp(A) A - exp(1) -     |   exp(x) dx
                                             |
                                            /
                                              1
```

```
>> changevar(t=F(x),Int(G(t),t),x);
                               /
                              |            /d      \
                              |  G(F(x)) |-- F(x)| dx
                              |            \dx     /
                             /
```

```
>> f := x -> arctan( sqrt( (1 - x) / (1 + x) ) );
                                                1 - x
                        f := x -> arctan(sqrt(-----))
                                                x + 1

>> intparts(Int(x * f(x),x = 0 .. 1), f(x));
                          1 /    1      1 - x   \  2
                           /   |- ----- - --------| x
                          |    |  x + 1          2|
                          |    \         (x + 1) /
                        - |    ------------------------ dx
                          |      /1 - x\1/2 /    1 - x\
                          |    4 |-----|    |1 + -----|
                          |      \x + 1/    \    x + 1/
                         /
                           0
>> I1 := simplify(intparts(Int(x * f(x),x = 0 .. 1), f(x)));
                                  1
                                 /        2
                                |        x
                      I1 := 1/4 | ------------ dx
                                |      2     1/2
                               /    (-x  + 1)
                                 0
```

```
>> I1 := simplify(intparts(Int(x * f(x),x = 0 .. 1), f(x)));
                                  1
                                 /        2
                                |        x
                      I1 := 1/4 | ------------ dx
                                |      2     1/2
                               /    (-x  + 1)
                                0
>> changevar(x = sin(u), I1, u);
                               Pi
                              ----
                               2
                             /            2
                            |       sin(u)  cos(u)
                       1/4  |      ----------------- du
                            |               2     1/2
                           /       (-sin(u)  + 1)
                             0

>> I2 := simplify(changevar(x = sin(u), I1, u));
                                      Pi
                                     ----
                                      2
                                     /
                                    |          2
                         I2 := 1/4  |     sin(u)  du
                                    |
                                   /
                                    0
```

```
>> I2 := simplify(changevar(x = sin(u), I1, u));
                                         Pi
                                        ----
                                         2
                                       /
                                      |         2
                        I2 := 1/4     |   sin(u)   du
                                      |
                                     /
                                       0
>> combine(I2,trig);
                                 Pi
                                ----
                                 2
                               /
                              |
                      1/4     |    1/2 - 1/2 cos(2 u) du
                              |
                             /
                               0
```

Comprendre pourquoi le logiciel n'affiche pas toujours une limite explicite, ou le résultat d'un calcul d'intégrale, par manque d'information sur la nature d'un paramètre introduit : savoir préciser à quelle partie de $\mathbf{R}$ il appartient (entier, réel positif…)

```
>> int(exp(-a * x^2), x = 0 .. infinity);
                                        1/2      1/2
                                      Pi    erf(a    x)
                        lim       1/2 -----------------
                   x -> infinity               1/2
                                              a

>> assume(a > 0); int(exp(-a * x^2), x = 0 .. infinity);
                                     1/2
                                   Pi
                                  -------
                                      1/2
                                  2 a~
```

```
>> I1 := int(exp(-sqrt(x^3)),x);
          /       /  3 1/2\2/3         3 1/2
          |       |(x )   |           (x )                        3 1/2
          |     x |-------|    exp(- -------) WhittakerM(1/3, 5/6, (x )   )
          |       |  3/2  |              2
          |       \ x     /
I1 := 2/3 |9/10 -----------------------------------------------------------
          |                                  3 1/6
          \                                (x )

             /  3 1/2\2/3         3 1/2                              \
             |(x )   |           (x )                        3 1/2 |
           x |-------|    exp(- -------) WhittakerM(4/3, 5/6, (x )   )|
             |  3/2  |              2                                 |
             \ x     /                                                |   /
     + 3/2 -----------------------------------------------------------|  /
                                         3 2/3                        | /
                                       (x )                           /

    /  3 1/2\2/3
    |(x )   |
    |-------|
    |  3/2  |
    \ x     /
```

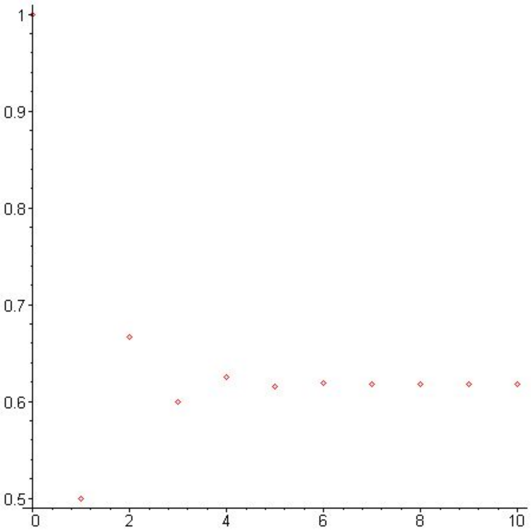
```
>> I1 := int(exp(-sqrt(x^3)),x):
>> series(I1,x);
                   (3/2) (2/3)            (3/2) (5/3)   5/2
        (9/10 csgn(x      )      - 3/2 csgn(x      )     ) x
x + 2/3 -----------------------------------------------------
                                   (3/2) 2/3
                             csgn(x      )

                       (3/2) (5/3)            (3/2) (8/3)   4
           (-9/16 csgn(x      )      + 3/4 csgn(x      )     ) x       (11/2)
     + 2/3 ----------------------------------------------------- + O(x      )
                                     (3/2) 2/3
                               csgn(x
```

```
>> assume(x>0);
>> series(I1,x);
                           5/2      4
                       2 x~       x~         (11/2)
                 x~ - ------- + --- + O(x~        )
                          5        8

>> int(series(exp(-sqrt(x^3)),x),x);
                         5/2      4      11/2
                     2 x~       x~     x~             7
               x~ - ------- + --- - ------ + O(x~ )
                        5        8      33
```

Savoir expliciter les premiers termes (de façon exacte ou approchée) d'une suite numérique ou d'une suite de fonctions, en particulier lorsqu'elle est définie par récurrence ;
Savoir visualiser sur un même schéma les premiers termes d'une suite de fonctions ;

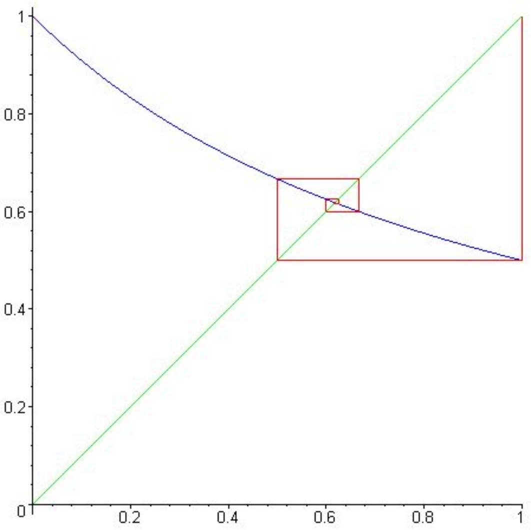
```
>> g := x -> 1 / (1 + x):
>> u := (f,n) -> f(u(f,n - 1)):
>> u(g,0) := 1:
>> seq(u(g,k), k = 0 .. 10);
                                         13   21   34   55   89
           1, 1/2, 2/3, 3/5, 5/8, 8/13, --, --, --, --, ---
                                         21   34   55   89   144
>> seq(evalf(u(g,k)), k = 0 .. 10);
  1., 0.5000000000, 0.6666666667, 0.6000000000, 0.6250000000, 0.6153846154, 0.6190476190,
        0.6176470588, 0.6181818182, 0.6179775281, 0.6180555556
```

```
>> plot([seq([k,u(g,k)], k = 0 .. 10)], style = point);
```

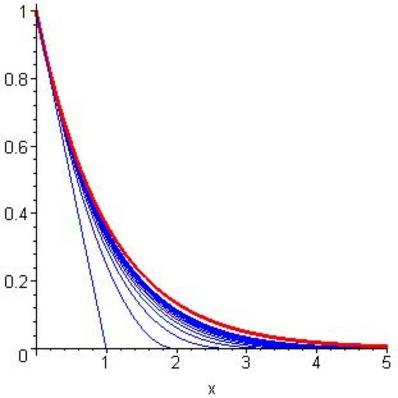


```
>> escargot := proc(f,n,xM)
    local avance, P1,P2,P3;
    avance := k -> ([u(f,k),u(f,k)],[u(f,k),u(f,k + 1)]):
    P1 := plot([seq(avance(k), k = 0 .. n)], color = red):
    P2 := plot(x, x = 0 .. xM, color = green):
    P3 := plot(f(x), x = 0 .. xM, color = blue):
    plots[display](P1,P2,P3, scaling = constrained);
  end:
```

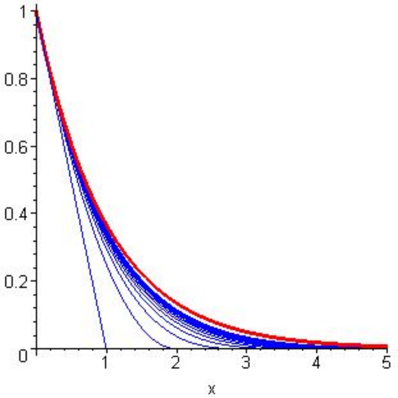
```
>> escargot(g,10,1);
```



```
>> f := (n,x) -> piecewise(x < n, (1 - x/n)^n, 0):
>> S := seq(plot([f(k,x),exp(-x)],x=0..5,color=[blue,red],thickness=[1,3]),k=1..15):
>> plots[display]([S]); # ou plots[display]([S], insequence = true) pour une animation
```

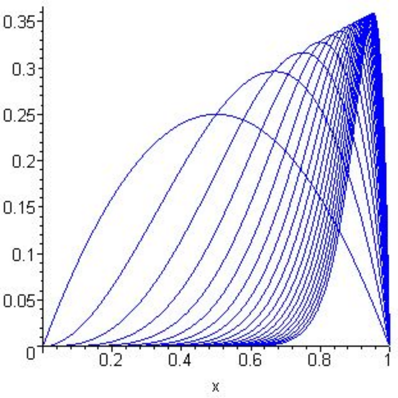


```
>> f := (n,x) -> piecewise(x < n, (1 - x/n)^n, 0):
>> S := seq(plot([f(k,x),exp(-x)],x=0..5,color=[blue,red],thickness=[1,3]),k=1..15):
>> plots[display]([S]); # ou plots[display]([S], insequence = true) pour une animation
```

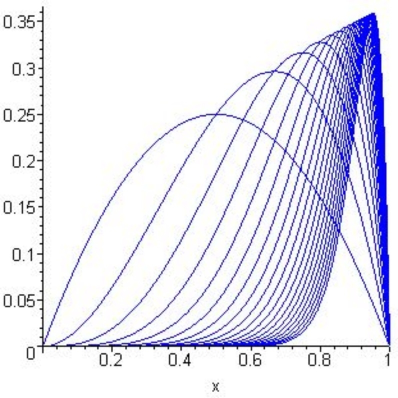


```
>> seq(evalf(f(100*k,2)),k=0..10);
  0., 0.1326195559, 0.1339796749, 0.1344320455, 0.1346580429,0.1347935812, 0.1348839150, 0.1349484269,
         0.1349968041,0.1350344268, 0.1350645224
>> exp(-2.0);
                          0.1353352832
>> seq(evalf(f(10000,x)-exp(-x)), x=0..5);
  0., -0.0000183948, -0.0000270679, -0.00002240362, -0.00001465056, -0.000008419977
```

```
>> f := (n,x) -> n * x^n * (1 - x):
>> S := seq(plot(f(k,x),x=0..1,color=blue),k=1..20):
>> plots[display]([S]);
```



```
>> f := (n,x) -> n * x^n * (1 - x):
>> S := seq(plot(f(k,x),x=0..1,color=blue),k=1..20):
>> plots[display]([S]);
```



```
>> maxf := solve(diff(f(n,x),x) = 0,x);

                                            n
                                maxf := -----
                                          n + 1
```

```
>> limit( f(n,maxf), n = infinity );
                              exp(-1)
```

```
>> maxf := solve(diff(f(n,x),x) = 0,x);

                    n
          maxf := -----
                  n + 1
```

```
>> limit( f(n,maxf), n = infinity );
                     exp(-1)
```

```
>> u := n -> (4*n + 3) / ( (2*n + 1)^2 * (2*n + 2)^2 );
                                 4 n + 3
                  u := n -> ---------------------
                                   2           2
                          (2 n + 1)  (2 n + 2)

>> sum(u(n), n = 0 .. infinity);

                                  2
                                Pi
                                ---
                                12
```

```
>> u := (n,x) -> x^(2*n + 2) / ( n*(n + 1)*(2*n + 1) );
                                        (2 n + 2)
                                       x
                  u := (n, x) -> -------------------
                                 n (n + 1) (2 n + 1)

>> series(u(n,x),n=infinity,2);

                              1          n 2
                            O(----) (| x | )
                               3
                              n

>> assume(x > 0, x < 1); simplify( sum( u(n,x), n = 1..infinity ) );
      2                                          2
  3 x~  + 2 x~ ln(1 - x~) - 2 x~ ln(1 + x~) - x~  ln(1 - x~)

            2
         - x~  ln(1 + x~) - ln(1 - x~) - ln(1 + x~)

>> coulditbe(1 - x < 0);
                                 false
```

Exercice 1

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0, \pi]$.

1. *Représentation sur $[-15, 15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi} (f(x))^2 \, dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{T}\int_{[a,a+T]} f^2(x) \, dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{T}\int_{[a,a+T]} (a_n\cos(n\omega x) + b_n\sin(n\omega x))^2(x) \, dx$.*

   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f) = f_{moy}^2 + \sum_{n=1}^{+\infty} \mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2 + \sum_{n=1}^{n_0} \mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0; \frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.*

Exercice 1

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0, \pi]$.

1. *Représentation sur $[-15, 15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}(f(x))^2 \, dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{T}\int_{[a,a+T]} f^2(x) \, dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{T}\int_{[a,a+T]} (a_n \cos(n\omega x) + b_n \sin(n\omega x))^2(x) \, dx$.*

   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f) = f_{moy}^2 + \sum_{n=1}^{+\infty} \mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2 + \sum_{n=1}^{n_0} \mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0; \frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.*

Exercice 1

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0, \pi]$.

1. *Représentation sur $[-15, 15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}(f(x))^2\ dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{T}\int_{[a,a+T]} f^2(x)\ dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{T}\int_{[a,a+T]}(a_n\cos(n\omega x) + b_n\sin(n\omega x))^2(x)\ dx$.*

   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f) = f_{moy}^2 + \sum_{n=1}^{+\infty}\mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2 + \sum_{n=1}^{n_0}\mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0; \frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.*

Exercice 1

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0, \pi]$.

1. *Représentation sur $[-15, 15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}(f(x))^2 \, dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{T}\int_{[a,a+T]} f^2(x) \, dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{T}\int_{[a,a+T]} (a_n \cos(n\omega x) + b_n \sin(n\omega x))^2(x) \, dx$.*

   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f) = f_{moy}^2 + \sum_{n=1}^{+\infty} \mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2 + \sum_{n=1}^{n_0} \mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0; \frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.*

Exercice 1

Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x) = x$ pour tout x appartenant à $[0, \pi]$.

1. *Représentation sur $[-15, 15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}(f(x))^2 \ dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s) = \frac{1}{T}\int_{[a,a+T]} f^2(x) \ dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s) = \frac{1}{T}\int_{[a,a+T]} (a_n \cos(n\omega x) + b_n \sin(n\omega x))^2(x) \ dx$.*

   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f) = f_{moy}^2 + \sum_{n=1}^{+\infty} \mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2 + \sum_{n=1}^{n_0} \mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t) = 1$ pour tout $t \in \left]0; \frac{1}{2}\right[$ et $f(0) = f(1/2) = 0$ et visualisez un résultat bien connu.*

Exercice 1

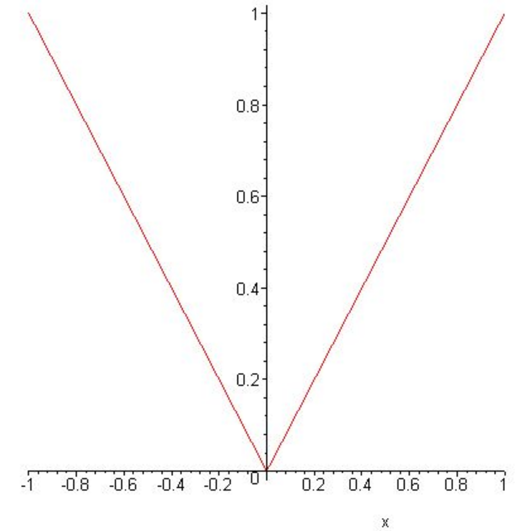
Soit f la fonction paire, 2π-périodique et définie sur $\mathbb{R}$ par $f(x)=x$ pour tout x appartenant à $[0,\pi]$.

1. *Représentation sur $[-15,15]$*
2. *Valeur moyenne de la fonction et valeur efficace sur une période (i.e. $\sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}(f(x))^2 \ dx}$).*
3. *Coefficients du développement en série de* Fourier
4. *L'énergie d'un signal est donnée par $\mathcal{E}(s)=\frac{1}{T}\int_{[a,a+T]} f^2(x)\ dx$ et l'énergie de l'harmonique de rang n par $\mathcal{E}_n(s)=\frac{1}{T}\int_{[a,a+T]}(a_n\cos(n\omega x)+b_n\sin(n\omega x))^2(x)\ dx$.*

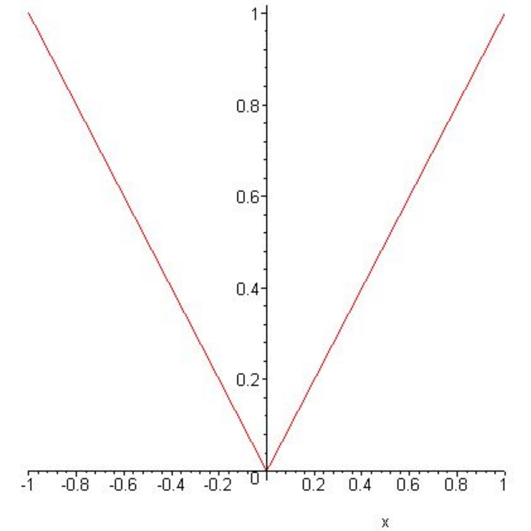
   La formule de Bessel-Parseval *donne donc $\mathcal{E}(f)=f_{moy}^2+\sum_{n=1}^{+\infty}\mathcal{E}_n$*

   Déterminez le plus petit entier n_0 telle que $f_{moy}^2+\sum_{n=1}^{n_0}\mathcal{E}_n$ soit égal à 99,9% de l'énergie du signal.
5. *Refaites les calculs avec la fonction impaire, de période 1, et telle que $f(t)=1$ pour tout $t\in\left]0;\frac{1}{2}\right[$ et $f(0)=f(1/2)=0$ et visualisez un résultat bien connu.*

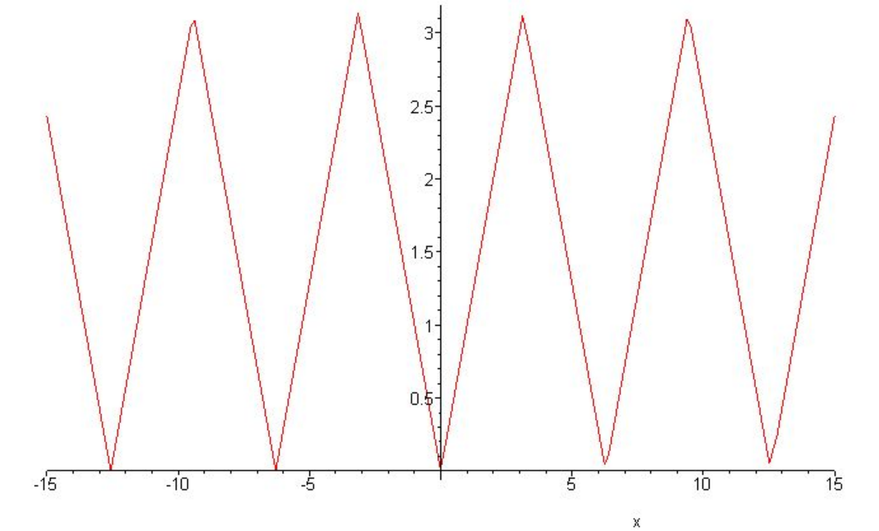
```
>> f := x -> piecewise(x<0,-x,x);
>> plot(f(x),x=-1..1,discont=true);
```



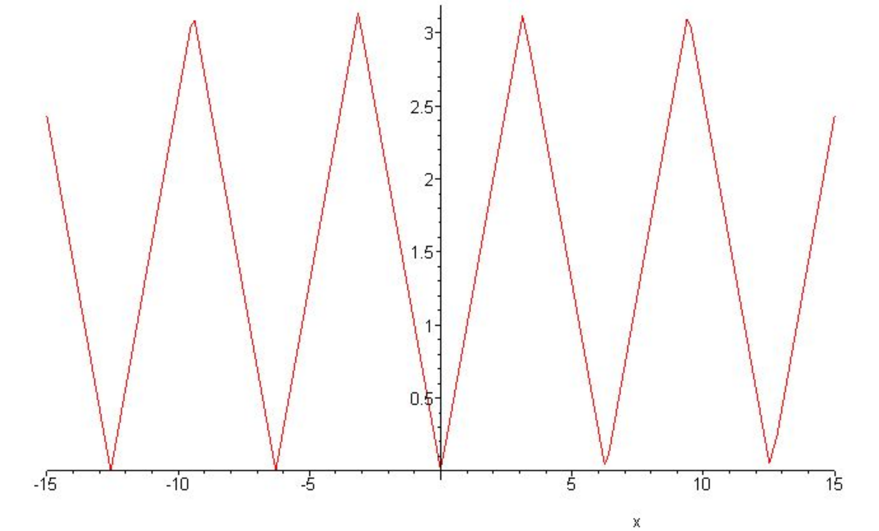
```
>> f := x -> piecewise(x<0,-x,x);
>> plot(f(x),x=-1..1,discont=true);
```



```
>> T := 2*Pi:
>> x0 := -Pi:
>> signal := x -> f(x - floor((x - x0)/T) * T);
>> plot(signal(x),x=-15..15);
```



```
>> T := 2*Pi:
>> x0 := -Pi:
>> signal := x -> f(x - floor((x - x0)/T) * T);
>> plot(signal(x),x=-15..15);
```



```
>> omega := 2*Pi/T;
                                  omega := 1
>> ValMoy := (1/(T))*int(f(x),x = -T/2..T/2);
                                            Pi
                                ValMoy := ----
                                            2
>> f_eff := sqrt((1/T)*(int(f(x)^2, x = -T/2..T/2)));
                                          1/2
                                         3    Pi
                               f_eff := -------
                                            3
>> a:= k -> (2/T)*int(f(x)*cos(k*omega*x), x = -T/2..T/2);
                                  T/2
                                 /
                                |
             a := k -> 2/T      |      f(x) cos(k omega x) dx
                                |
                               /
                                 - T/2
>> b:= k -> (2/T)*int(f(x)*sin(k*omega*x), x = -T/2..T/2);
                                  T/2
                                 /
                                |
             b := k -> 2/T      |      f(x) sin(k omega x) dx
                                |
                               /
                                 - T/2
```

```
>> a(k);b(k);
                  2 (cos(Pi k) + sin(Pi k) Pi k - 1)
                  ----------------------------------
                                    2
                                Pi k

                                  0
```

```
>> assume(j,integer);
>> a(j);
                                   j~
                         2 ((-1)   - 1)
                         --------------
                                     2
                                 Pi j~
```

```
>> assume(p,even);
>> a(p);
                                  0

>> assume(q,odd);
>> a(q);
                                   4
                                - ------
                                       2
                                  Pi q~
```

```
>> a(k);b(k);
                  2 (cos(Pi k) + sin(Pi k) Pi k - 1)
                  ----------------------------------
                                    2
                                Pi k

                                  0
```

```
>> assume(j,integer);
>> a(j);
                                   j~
                            2 ((-1)   - 1)
                            --------------
                                     2
                                Pi j~
```

```
>> assume(p,even);
>> a(p);
                                  0

>> assume(q,odd);
>> a(q);
                                   4
                               - ------
                                      2
                                 Pi q~
```

```
>> a(k);b(k);
                  2 (cos(Pi k) + sin(Pi k) Pi k - 1)
                  ----------------------------------
                                    2
                                Pi k

                                  0
```

```
>> assume(j,integer);
>> a(j);
                                     j~
                            2 ((-1)   - 1)
                            --------------
                                       2
                                Pi j~
```

```
>> assume(p,even);
>> a(p);
                                  0

>> assume(q,odd);
>> a(q);
                                   4
                               - ------
                                      2
                                 Pi q~
```

```
>> s := (n,x) -> ValMoy + sum(a(k)*cos(k*omega*x)+b(k)*sin(k*omega*x),k=1..n);

  s := (n, x) ->

                  /  n                                               \
                  |-----                                             |
                  | \                                                |
        ValMoy +  |  )   (a(k) cos(k omega x) + b(k) sin(k omega x))|
                  | /                                                |
                  |-----                                             |
                  \k = 1                                             /
```

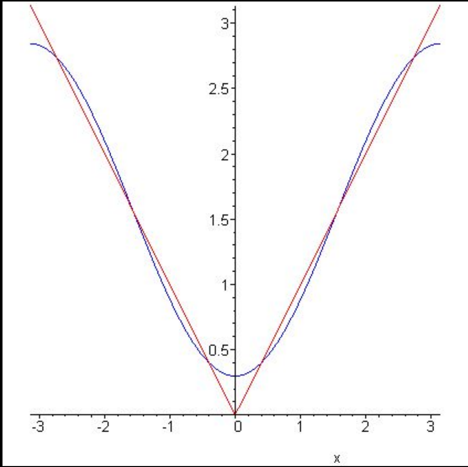
```
>> s(5,x);
              Pi    4 cos(x)        cos(3 x)        cos(5 x)
             ---- - -------- - 4/9 -------- - 4/25 --------
              2        Pi              Pi               Pi
```

```
>> s := (n,x) -> ValMoy + sum(a(k)*cos(k*omega*x)+b(k)*sin(k*omega*x),k=1..n);

  s := (n, x) ->

                 /  n                                            \
                 |-----                                          |
                 | \                                             |
       ValMoy +  |  )   (a(k) cos(k omega x) + b(k) sin(k omega x))|
                 | /                                             |
                 |-----                                          |
                 \k = 1                                          /
```

```
>> s(5,x);
             Pi    4 cos(x)       cos(3 x)        cos(5 x)
            ---- - -------- - 4/9 -------- - 4/25 --------
             2        Pi             Pi              Pi
```

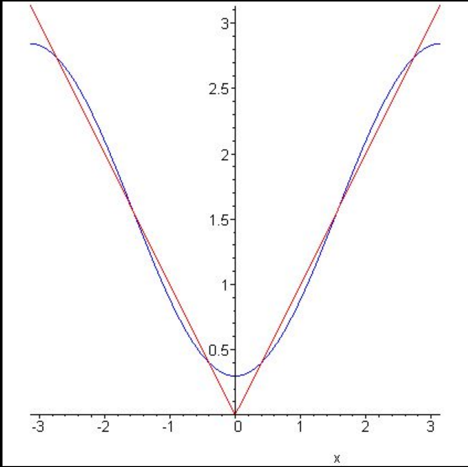
```
>> conv:=proc(n,xmin,xmax)
     local F,i,S;
     S := seq(plot([signal(x),s(i,x)],x=xmin..xmax,color=[red,blue]),i=1..n);
     plots[display]([S],insequence=true);
   end:
```

```
>> conv(20,-T/2,T/2);
```



```
>> conv:=proc(n,xmin,xmax)
     local F,i,S;
     S := seq(plot([signal(x),s(i,x)],x=xmin..xmax,color=[red,blue]),i=1..n);
     plots[display]([S],insequence=true);
   end:
```

```
>> conv(20,-T/2,T/2);
```



```
>> seuil := proc(taux)
     local azero,valeur,seuil,k;
     azero := ValMoy;
     valeur := evalf(azero^2);
     seuil := evalf(taux*(f_eff)^2);
     k:=0;
     while valeur < seuil do
       k:=k+1;
       valeur := valeur + 0.5*evalf(a(k)^2+b(k)^2);
     od;
     RETURN(k)
   end:
```

```
>> seuil(0.9999);

                              7
```

```
>> seuil := proc(taux)
    local azero,valeur,seuil,k;
    azero := ValMoy;
    valeur := evalf(azero^2);
    seuil := evalf(taux*(f_eff)^2);
    k:=0;
    while valeur < seuil do
      k:=k+1;
      valeur := valeur + 0.5*evalf(a(k)^2+b(k)^2);
    od;
    RETURN(k)
  end:
```

```
>> seuil(0.9999);

                              7
```

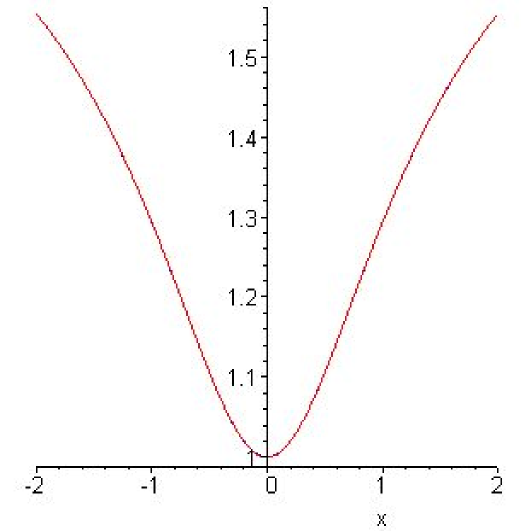
Savoir résoudre une équation différentielle, un système d'équations différentielles, avec ou sans conditions initiales ;
Savoir récupérer une fonction solution et la tracer ;
Savoir tracer directement le graphe d'une solution obtenue par résolution numérique.

1. Résoudre l'équation différentielle

$$(E) : (1 + x2)y' + xy - 2x = 0$$

2. Déterminer la solution particulière φ de (E) dont la courbe passe par $A(0,1)$. et tracer cette courbe sur $[-20,20]$.
3. Tracer la courbe précédente et la courbe obtenue au moyen de l'option `numeric`.
4. Donner un développement limité de φ à l'ordre 12 au voisinage de 0.

```
>> with(plots):
>> eq := (1+x^2)*diff(y(x),x) + x*y(x) - 2*x = 0:
>> dsolve(eq,y(x));

                             _C1
            y(x) = 2 + -----------
                              2 1/2
                       (1 + x )

>> s := dsolve({eq,y(0)=1},y(x)):
>> a := rhs(s);

                            1
             a := 2 - -----------
                             2 1/2
                      (1 + x )

>> A := plot(a,x=-2 .. 2):
>> b := dsolve({eq,y(0)=1},y(x),numeric):
>> B := odeplot(b,x=-2..2,color=blue):
>> display(A,B);
```


1.5
1.4
1.3
1.2
1.1
1
-2
-1
0
1
2
x

On considère le système différentiel

$$(S) : \begin{cases} x' & = & 3x + y + \mathrm{e}^t \\ y' & = & 2x + 2y + \mathrm{e}^{2t} \end{cases}$$

1. Résoudre ce système.
2. Tracer la courbe correspondant au problème de Cauchy (S) et $x(0) = 1, y(0) = 2$.
3. Préciser l'allure de la courbe quand t tend vers $-\infty$ et quand t tend vers $+\infty$.

On considère le système différentiel

$$(S) : \begin{cases} x' = 3x + y + \mathrm{e}^{t} \\ y' = 2x + 2y + \mathrm{e}^{2t} \end{cases}$$

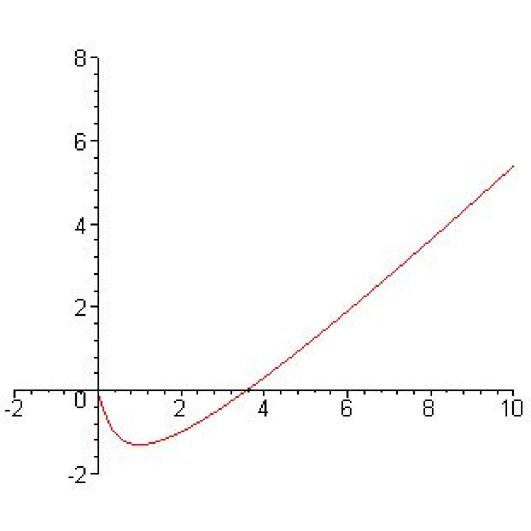
1. Résoudre ce système.
2. Tracer la courbe correspondant au problème de Cauchy (S) et $x(0) = 1, y(0) = 2$.
3. Préciser l'allure de la courbe quand t tend vers $-\infty$ et quand t tend vers $+\infty$.

On considère le système différentiel

$$(S) : \begin{cases} x' = 3x + y + e^t \\ y' = 2x + 2y + e^{2t} \end{cases}$$

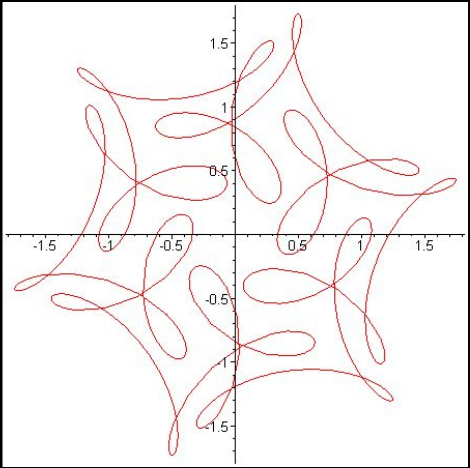
1. Résoudre ce système.
2. Tracer la courbe correspondant au problème de Cauchy (S) et $x(0) = 1, y(0) = 2$.
3. Préciser l'allure de la courbe quand t tend vers $-\infty$ et quand t tend vers $+\infty$.

```
>> sys:=diff(x(t),t)=3*x(t)+y(t)+exp(t),diff(y(t),t)=2*x(t)+2*y(t)+exp(2*t):
>> S:=dsolve({sys,x(0)=2,y(0)=-1},{x(t),y(t)});
  S := {x(t) =

                        25
        10/9 exp(t) + -- exp(4 t) + 1/3 exp(t) t - 1/2 exp(2 t), y(t)
                        18

                            25
         = -26/9 exp(t) + -- exp(4 t) + 1/2 exp(2 t) - 2/3 exp(t) t}
                            18
>> X:=rhs(S[1]):Y:=rhs(S[2]):
>> A := plot([X,Y,t=-10..10],view=[-2..10,-2..8],numpoints=500,scaling=constrained):
>> limit(X,t=-infinity);limit(Y,t=-infinity);
                                   0

                                   0
>> limit((2*X+2*Y+exp(2*t))/(3*X+Y+exp(t)),t=-infinity);
                                   -2
>> limit(X,t=infinity);limit(Y,t=infinity);
                                infinity
                                infinity
>> limit(Y/X,t=infinity);
                                   1
>> limit(Y-X,t=infinity);
                                infinity
>> display(A,scaling=constrained);
```

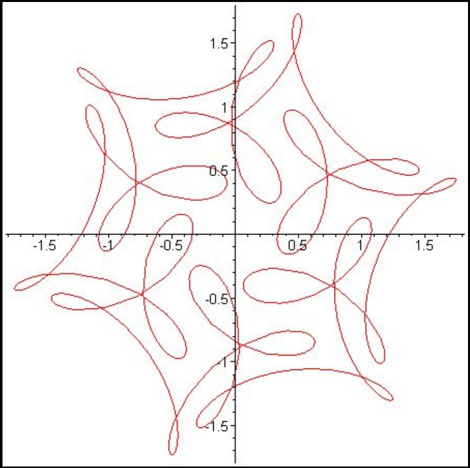

8
6
4
2
0
-2
-2
2
4
6
8
10

Savoir tracer une courbe du plan, définie par une équation cartésienne (de façon implicite), ou par un paramétrage, peut-être en coordonnées polaires, et gérer les discontinuités ;

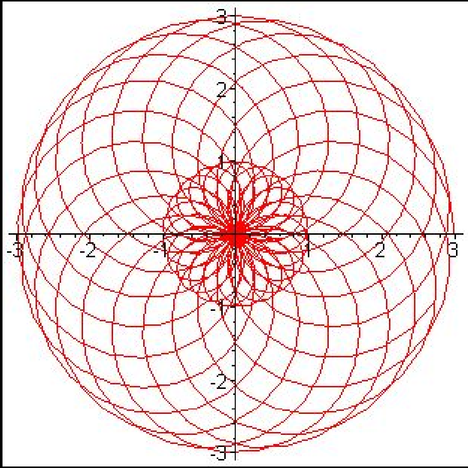
```
>> x := t -> cos(t) + cos(7*t)/2 + sin(17*t)/3:
>> y := t -> sin(t) + sin(7*t)/2 + cos(17*t)/3:
>> plot([x(t),y(t), t = 0 .. 2*Pi]);
```



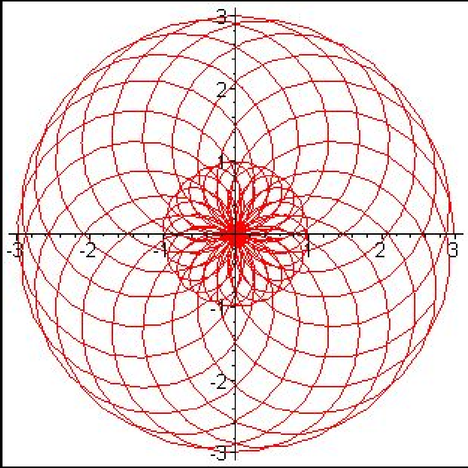
```
>> x := t -> cos(t) + cos(7*t)/2 + sin(17*t)/3:
>> y := t -> sin(t) + sin(7*t)/2 + cos(17*t)/3:
>> plot([x(t),y(t), t = 0 .. 2*Pi]);
```



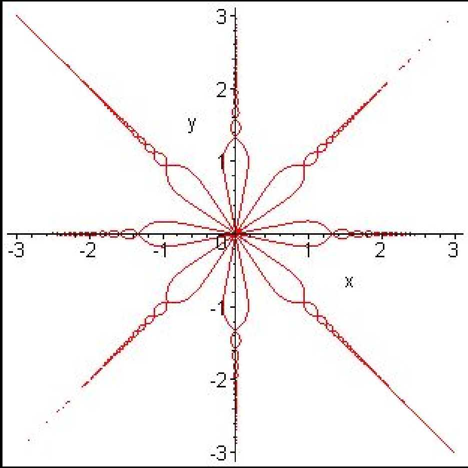
```
>> n := 20/19: p := t -> 1 + 2*cos(n*t):
>> plot(p(t),t=0..36*n*Pi,coords=polar,scaling=constrained);
```



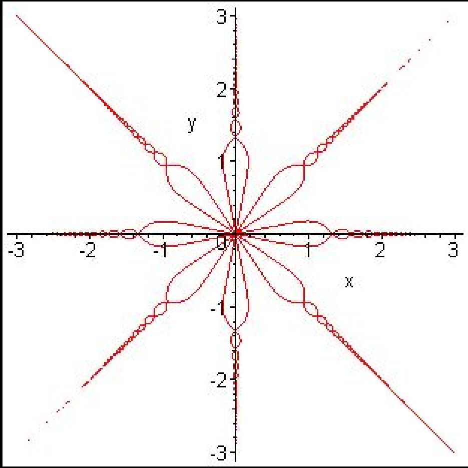
```
>> n := 20/19: p := t -> 1 + 2*cos(n*t):
>> plot(p(t),t=0..36*n*Pi,coords=polar,scaling=constrained);
```



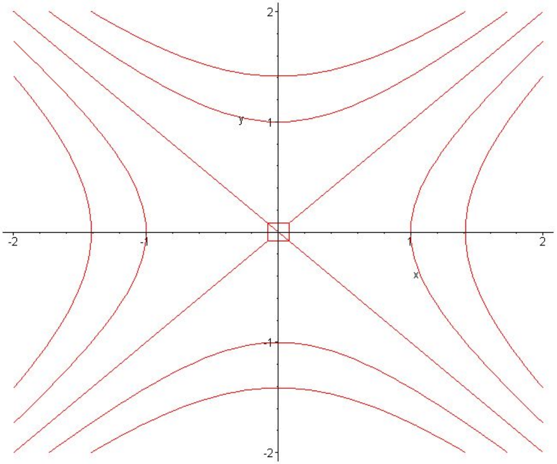
```
>> p := (x,y) -> abs(cos((x + I * y)^4)) - 1:
>> implicitplot(p(x,y),x=-3..3,y=-3..3, numpoints=100000);
```



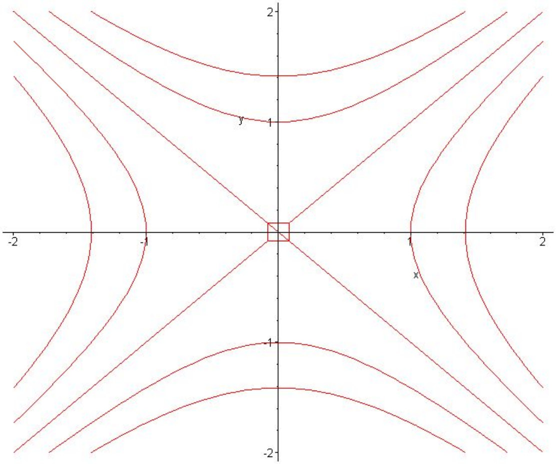
```
>> p := (x,y) -> abs(cos((x + I * y)^4)) - 1:
>> implicitplot(p(x,y),x=-3..3,y=-3..3, numpoints=100000);
```



```
>> p := t -> x^2 - y^2 - t:
>> plots[implicitplot]({seq(p(t), t = -2..2)}, x = -2..2, y = -2..2);
```

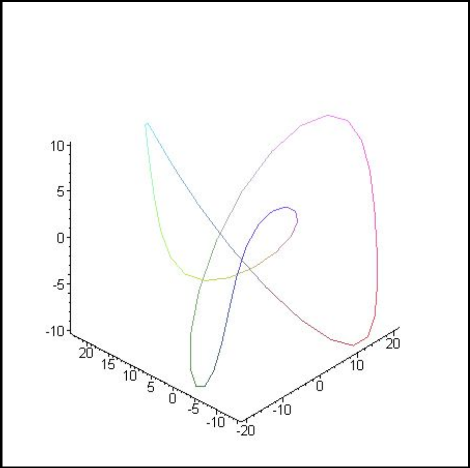


```
>> p := t -> x^2 - y^2 - t:
>> plots[implicitplot]({seq(p(t), t = -2..2)}, x = -2..2, y = -2..2);
```

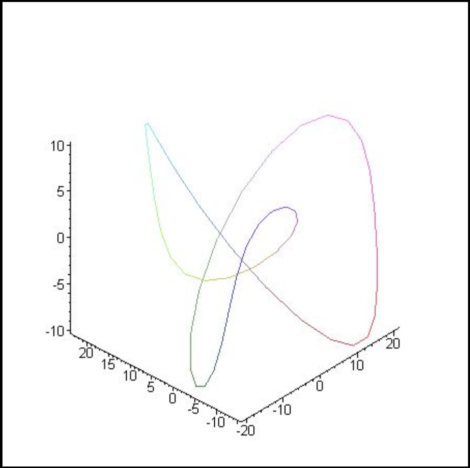


Savoir tracer une courbe paramétrée de l'espace ;

```
>> noeud := [-10*cos(t)-2*cos(5*t)+15*sin(2*t), -15*cos(2*t)+10*sin(t)-2*sin(5*t), 10*cos(3*t), t = 0
     .. 2*Pi]:
>> spacecurve(noeud);
```



```
>> noeud := [-10*cos(t)-2*cos(5*t)+15*sin(2*t), -15*cos(2*t)+10*sin(t)-2*sin(5*t), 10*cos(3*t), t = 0
       .. 2*Pi]:
>> spacecurve(noeud);
```



Savoir tracer une surface définie par un paramétrage, ou par une équation cartésienne.

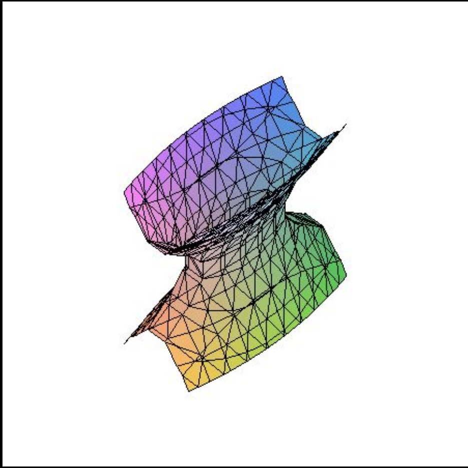
Soit $(\mathcal{S})$ la surface d'équation $\dfrac{x^2}{9}+y^2-\dfrac{z^2}{4}=1$

1. Nature de $(\mathcal{S})$? Donner un paramétrage de $(\mathcal{S})$. Tracer $(\mathcal{S})$ avec Maple.
2. Déterminer l'intersection de $(\mathcal{S})$ avec le plan d'équation $z=\alpha$.

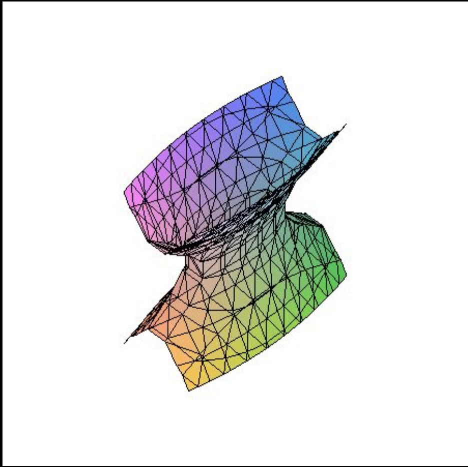
Soit $(\mathcal{S})$ la surface d'équation $\dfrac{x^2}{9} + y^2 - \dfrac{z^2}{4} = 1$

1. Nature de $(\mathcal{S})$? Donner un paramétrage de $(\mathcal{S})$. Tracer $(\mathcal{S})$ avec Maple.
2. Déterminer l'intersection de $(\mathcal{S})$ avec le plan d'équation $z = \alpha$.

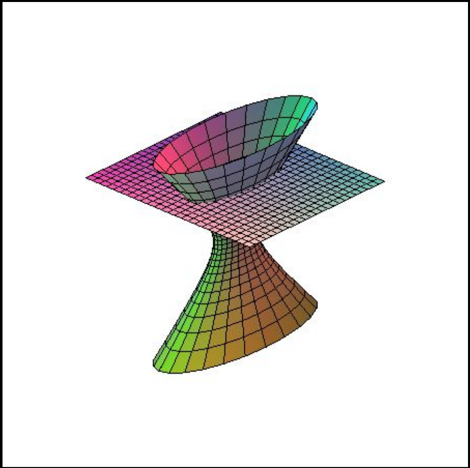
```
>> implicitplot3d(x^2/9+y^2-z^2/4=1, x=-5..5, y=-5..5, z=-5..5);
```



```
>> implicitplot3d(x^2/9+y^2-z^2/4=1, x=-5..5, y=-5..5, z=-5..5);
```



```
>> H1 := plot3d([3*cos(u)*cosh(v),sin(u)*cosh(v),2*sinh(v)], u=0..2*Pi, v=-2..2):
>> P := plot3d([x,y,4], x=-10..10, y=-10..10):
>> display3d({H1, P},scaling=constrained);
```



```
>> H1 := plot3d([3*cos(u)*cosh(v),sin(u)*cosh(v),2*sinh(v)], u=0..2*Pi, v=-2..2):
>> P := plot3d([x,y,4], x=-10..10, y=-10..10):
>> display3d({H1, P},scaling=constrained);
```