TP Maple 2 | Nombres et équations

Ce chapitre expose les différents types de nombres sous Maple et quelques principes de calcul formel et approché les concernant. Il se termine par un paragraphe sur la commande **solve** *qui permet la résolution formelle des équations.*

1. Les nombres

Dans ce chapitre, nous exposerons la gestion des nombres (entiers, rationnels, réels, complexes) par Maple. Les commandes les plus importantes seront consignées dans des tableaux récapitulatifs.

1.1. Les entiers et l'arithmétique

Maple peut manipuler des nombres comportant un très grand nombre de chiffres mais néanmoins limité. Le nombre maximal de chiffres, *maxdigits*, dépend du matétiel utilisé. On y aura accès par la commande **kernelopts(maxdigits)**.

```
> kernelopts(maxdigits);

                                268435448
```

Voici un aperçu des différentes opérations possibles sur les nombres entiers :

► *La fonction factorielle* : pour tout $n \in \mathbb{N}$, $n!$ se calcule au moyen de la commande **factorial** ou à l'aide du symbole **!**.

```
> factorial(12);

                                479001600

> 12!;

                                479001600
```

► *Division euclidienne*[1] : on utilise les commandes **irem** et **iquo** pour calculer le reste et le quotient de la division euclidienne[2].

```
> irem(23,3);

                                    2

> iquo(23,3);

                                    7
```

► *Test de primalité* : la fonction **isprime** permet de tester si un entier est premier[3] ou non.

1. Rappelons rapidement le principe de la division euclidienne : pour tous entiers n et $m \neq 0$, il existe un unique couple d'entiers (q,r) vérifiant $n = mq + r$ *et* $0 \leqslant r < m$; q s'appelle le quotient dans la division euclidienne de n par m et r le reste dans la division euclidienne de n par m. Par exemple, $23 = 7 \times 3 + 2$.
2. **irem** et **iquo** sont les abréviations respectives de *integer remainder* et *integer quotient*.
3. Un nombre entier est dit premier s'il est ditinct de 1 et si ses seuls diviseurs sont 1 et lui-même.

```
> isprime(23);
```

$$\textit{true}$$

► *Décomposition en produit de facteurs premiers*[4] : la fonction **ifactor** calcule la décomposition en produit de facteur(s) premier(s) d'un entier donné.

```
> ifactor(561);
```

$$(3)(11)(17)$$

On a donc $561 = 3 \times 11 \times 17$.

Arithmétique élémentaire sous Maple

- **irem(n,m)** : renvoie le reste de n dans la division euclidienne par m.
- **iquo(n,m)** : renvoie le quotient de n dans la division euclidienne par m.
- **igcd(n,m)** : renvoie $\text{pgcd}(n,m)$.
- **ilcm(n,m)** : renvoie $\text{ppcm}(n,m)$.
- **ithprime(n)** : renvoie le n-ième nombre premier.
- **isprime(n)** : renvoie *true* si n est premier, *false* sinon.
- **ifactor(n)** : renvoie la décomposition en produit de facteur(s) premier(s) de n.

1.2. Les nombres rationnels

Maple manipule les nombres rationnels formellement, c'est-à-dire comme des quotients d'entiers et non pas des approximations décimales. En cas de calcul sur des nombres rationnels, le logiciel renverra une écrire du résultat sous forme irréductible, ie sous la forme d'une fraction *simplifiée*.

```
> (1/2-1/3)*(66/78);
```

$$\frac{11}{78}$$

4. Il s'agit du théorème fondamental de l'Arithmétique : tout nombre entier distinct de 1 se décompose en produit de facteurs premiers.

1.3. Nombres remarquables

Au-delà des nombres rationnels, Maple sait manipuler formellement certains nombres irrationnels.

Les radicaux

C'est le cas des racines carrées de nombres rationnels. La racine carrée d'un nombre r s'obtient par la commande [5] **sqrt(r)**.

```
> x:=sqrt(2):x^2;
```

$$2$$

Le nombre irrationnel [6] $\sqrt{2}$ est traité par le logiciel comme l'unique solution positive de l'équation $x^2 = 2$ et non pas comme une approximation décimale. Il faudra parfois forcer la main au logiciel pour obtenir certaines simplifications pour nous évidentes.

```
> sqrt(10)/sqrt(2);
```

$$\frac{1}{2}\sqrt{(10)}\sqrt{(2)}$$

```
> simplify(%);
```

$$\sqrt{5}$$

La commande **simplify** permet de telles simplifications lors de calculs symboliques. On pourra également utiliser les commandes **radnormal** et **rationalize** permettant des simplifications.

```
> a:=(7+5*sqrt(2))^(1/3): radnormal(a);
```

$$1+\sqrt{2}$$

```
> rationalize(1/(sqrt(2)+sqrt(3)+sqrt(5)));
```

$$-\left(\frac{1}{12}\left(-\sqrt{2}-\sqrt{3}+\sqrt{5}\right)\right)\sqrt{2}\sqrt{3}$$

5. **sqrt** comme abréviation de *square root*, *racine carrée* in French.
6. Voir le cours de Mathématique pour la preuve (classique) de l'irrationnalité de $\sqrt{2}$.

Simplifications des radicaux

► **simplify** : permet de symplifier des expressions radicalaires (entre autres) ; pour les expressions rebelles e, on pourra utiliser **simplify(e,radical)** ou **simplify(e,sqrt)** (plus efficace s'il n'y a que des racines carrées).

► **rationalize** : simplifie des expressions où apparaissent des radicaux au dénominateur.

► **radnormal** : simplifie des expressions où apparaissent des radicaux ; pour éviter des radicaux au dénominateur, on l'utilisera avec l'option 'rationalized'.

Les constantes prédéfinies

Il faudra prendre garde à la syntaxe : le nombre π s'obtient par **Pi** et la base e de l'exponentielle par **exp(1)**. Le lecteur tira les enseignements de l'exemple qui suit...

```
> cos(pi);
```

$$\cos(\pi)$$

```
> cos(Pi);
```

$$-1$$

```
> ln(e);
```

$$\ell\mathrm{n}(e)$$

```
> ln(exp(1));
```

$$1$$

1.4. Les nombres décimaux

On appelle nombre décimal tout nombre rationnel qui est le quotient d'un entier relatif par une puissance entière de 10. Les décimaux sont notés sous Maple à l'anglo-saxonne : avec un point **.** au lieu de la virgule **,** française. Ces nombres sont qualifiés de *flottants*[7] et leur type est *float*.

7. C'est en fait la virgule qui est flottante !

```
> 1.3-1.5;
```

$$-0.2$$

Il y a trois écritures possibles d'un décimal au moyen des puissances de 10 :

- *la notation dite scientifique* : au moyen du symbole e ou E (indifféremment) ;
- *les puissances usuelles de 10* ;
- *la fonction* **Float**[8].

Par exemple, le décimal 0.0012 pourra être défini par

$$1.2e-3,\ 1.2E-3,\ 1.2*10^{-3} \text{ ou encore } \textbf{Float(12,-4)}.$$

On retiendra la règle suivante :

Calcul formel vs. évaluation décimale

- Le mélange dans une expression de nombres sous forme de fraction et sous forme décimale entraîne l'écriture du résultat sous forme décimale ;
- Lorsqu'une fonction est appelée avec un argument sous forme décimale, le résultat retourné sera sous forme décimale ;
- En revanche, priroité est donnée au calcul symbolique sur le calcul numérique.

De quoi éclairer ce qui suit :

8. Les arguments de cette fonctions doivent être des entiers sous peine d'erreurs.

```
> 1/2+0.25;
```

$$0.75$$

```
> 1/2+1/4;
```

$$\frac{3}{4}$$

```
> sqrt(0.25);
```

$$0.5000000000$$

```
> sqrt(1/4);
```

$$\frac{1}{2}$$

```
> 1.024+Pi;
```

$$1.024+\pi$$

Signalons que l'on peut convertir un nombre décimal d donné en fraction irréductible en utilisant la commande **convert(d,rational)**.

```
> convert(3.1415927,rational);
```

$$\frac{126003}{40108}$$

Ecriture fractionnaire d'un nombre décimal

convert(d,rational) : donne la forme fractionnaire irréductible du nombre décimal d.

1.5. Les nombres réels

Nous avons déjà remarqué que Maple effectue *en priorité* des calculs exacts sur les fractions. Il faudra donc utiliser une commande spécifique afin d'obtenir des valeurs approchées sous forme décimale.

```
> sqrt(2);
```

$$\sqrt{2}$$

C'est la fonction **evalf** qui permet d'obtenir des valeurs approchées sous forme décimale [9] avec un nombre de chiffres significatifs contenu dans la variable *globale* **Digits**. La valeur par défaut de **Digits** est 10. Attention, si l'on modifie cette variable, la nouvelle valeur sera utilisée pour toutes les évaluations qui suivront dans la feuille de calcul.

```
> evalf(sqrt(2));
```

1.414213562

```
> Digits:=20:evalf(sqrt(2));
```

1.4142135623730950488

On peut obtenir le même résultat en spécifiant le nombre de chiffres significatifs voulu en second argument de **evalf**. Cette méthode a l'avantage de ne changer que *localement* la valeur de **Digits**.

```
> evalf(sqrt(2),20);
```

1.4142135623730950488

Rappelons pour finir que la présence d'un nombre sous forme décimale en argument d'une fonction forcera Maple à écrire le résultat sous forme décimale [10].

```
> sqrt(2.1);
```

1.449137674

9. Le nom **evalf** est une contraction de *floating evaluation*.
10. Avec toutefois priorité au calcul symbolique.

Manipulation des flottants

- **Digits := n** : ixe à n le nombre de chiffres significatifs lors de l'appel de **evalf**.
- **evalf(x)** : renvoie une valeur décimale approchée de x avec **Digits** chiffres significatifs.
- **evalf(x,n)** : renvoie une valeur décimale approchée de x avec n chiffres significatifs.
- **floor(x)** : partie entière de x.
- **round(x)** : entier le plus proche de x.

1.6. Les nombres complexes

Sous Maple, tout nombre complexe z s'écrit sous forme algébrique $z = a + b * I$ où le symbole **I** désigne le nombre noté i dans le cours de Mathématiques. Ce symbole est *réservé* : au même titre que **Pi**, on ne peut pas changer sa signification sous peine d'erreur. Maple effectue les opérations algébriques usuelles sur les nombres complexes. La multiplication sera codée explicitement par le signe *.

```
> (3 + I)*(2*I + 3);
```

$$7 + 9I$$

Dans le cas où Maple ne retourne pas une écriture sous forme algébrique, on pourra lui forcer la main en utilisant la fonction **evalc**.

```
> (sqrt(2) + I)*(1 + I);
```

$$(\sqrt{2} + I)(1 + I)$$

```
> evalc((sqrt(2) + I)*(1 + I));
```

$$\sqrt{2} - 1 + I(1 + \sqrt{2})$$

On peut également définir un nombre complexe sous forme trigonométrique à l'aide de la fonction **polar**. Le passage d'une forme algébrique à une forme trigonométrique s'effectue au moyen des commandes **argument** et **abs** calculant respectivement un argument appartenant à $[-\pi, \pi]$ et le module d'un nombre complexe.

```
> evalc(polar(1,Pi/3));
```

$$\frac{1}{2}+\frac{1}{2}I\sqrt{3}$$

Manipulation des nombres complexes

- **z^n** : renvoie z^n.
- $\mathbf{z_1} * \mathbf{z_2}$: renvoie $z_1 \times z_2$.
- **abs(z)** : renvoie $|z|$.
- **argument(z)** : renvoie un argument de z. Attention, sous Maple, 0 est d'argument 0 !
- **Re(z)** : renvoie $\mathrm{Re}(z)$.
- **Im(z)** : renvoie $\mathrm{Im}(z)$.
- **polar(r,θ)** : renvoie la valeur $re^{i\theta}$.
- **conjugate(z)** : renvoie $\overline{z}$.

1.7. La commande assume

Il est parfois important de faire des hypothèses sur les inconnues que l'on manipule. Commençons par un exemple.

```
> restart: z:=x+I*y: Re(z);
```

$$\mathrm{Re}(x+Iy)$$

Le logiciel ne sait pas calculer $\mathrm{Re}(z)$ pour la même raison que nous : les nombres x et y sont-ils réels ? Pour indiquer à Maple que x et y sont réels, on utilisera la commande **assume**.

```
> restart: assume(x,real,y,real): z:=x+I*y: Re(z);
```

$$x\tilde{}$$

On remarquera que le logiciel affuble x d'un tilde ~ afin que l'on se souvienne que des hypothèses ont été faites sur x. On retiendra la syntaxe suivante.

La commande assume

assume(x, type) ou **assume(hypothèses sur** x **)**.

```
> restart: expand(ln(x*y));
```

$$\ell\mathrm{n}(xy)$$

```
> assume(x>0,y>0): expand(ln(x*y));
```

$$\ell\mathrm{n}(x\tilde{}) + \ell\mathrm{n}(y\tilde{})$$

1.8. Exercices

Exercice 1.

Pour tout entier naturel n, on pose

$$u_n = \frac{1}{\binom{2n}{n}}.$$

1. Simplifier, pour tout er naturel n,

$$v_n = \frac{u_{n+1}}{u_n}.$$

2. En déduire la limite de v_n quand n tend vers $+\infty$.

Exercice 2.

Simplifier

$$x = \sqrt[5]{13452\sqrt{2} - 19024}.$$

Exercice 3.

Combien y-a-t-il de zéros à la fin de l'écriture décimale de 300! ? On pourra utiliser la commande **ifactor**.

Exercice 4.

Simplifier les expressions suivantes :

$$z_1 = \frac{1 + i\tan(\alpha)}{1 - i\tan(\alpha)} \quad \text{et} \quad z_2 = \frac{(1+i)^n}{(1-i)^{n-2}}.$$

où $\alpha \in \mathbb{R}$ et $n \in \mathbb{Z}$.

Exercice 5.

On souhaite retrouver au moyen de Maple la célèbre formule de Machin :

$$\frac{\pi}{4} = 4\arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right).$$

1. On pose

$$z = (1-i)(5+i)^4.$$

1.a. Ecrire z sous forme algébrique.

1.b. Calculer un argument de z au moyen de la fonction arctan.

1.c. En déduire que

$$\frac{\pi}{4} = 4\arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right) \ [2\pi]$$

2. Conclure en utilisant **evalf**.

Exercice 6.

En vous aidant de Maple, trouver une simplification de la somme :

$$\arctan(2) + \arctan(3) + \arctan(2+\sqrt{3}).$$

2. Résolution formelle des équations

Commençons par remarquer qu'une équation est considérée par Maple comme une *expression* dont le type est simplement = .

```
> eq:=3*x+2=3+2*x: whattype(eq), op(eq);
```

‘=‘, $3x+2, 3+2x$

Toute équation admet deux opérandes : ses premier et second membres. On pourra donc y accèder respectivement par les commandes **op(1,eq)** et **op(2,eq)**. Le logiciel dispose cependant de deux raccourcis permettant ces extractions : **lhs(eq)** et **rhs(eq)**.

```
> op(1,eq), op(2,eq), is(rhs(eq)=op(2,eq));
```

3x+2, 3+2x, true

Extraction des opérandes d'une égalité (ou inégalité)

► **lhs(E)** : extrait le membre de gauche de E. C'est un raccourci de **op(1,E)** (*lhs* signifie *left hand side*).

► **rhs(E)** : extrait le membre de droite de E. C'est un raccourci de **op(2,E)** (*rhs* signifie *right hand side*).

Dans ce paragraphe, nous étudierons la résolution de certaines équations. Nous nous bornerons aux équations (allant même jusqu'aux systèmes d'équations, voire aux inéquations !) de la forme $f(x) = a$ où f est une fonction usuelle et a un nombre. Lorsque f est un polynôme, on parlera d'équation algébrique. Le cas des systèmes linéaires sera à peine évoqué car sera traité dans un chapitre ultérieur. Notre approche balancera entre le calcul exact des solutions et leur calcul numérique.

2.1. La commande solve

Le logiciel est capable de résoudre exactement certains types d'équation. C'est la commande **solve** qui permet la résolution formelle d'une équation donnée.

Résolution formelle de $p = 0$: la commande **solve**

solve(p=0, *nom de l'inconnue de résolution*)

On peut également enregistrer l'équation[11] $p = 0$ dans une variable *equ* et valider la ligne de commandes suivante : **solve(*equ*)**. Dans le cas où figurent plusieurs variables dans l'équation $p = 0$, il faudra préciser l'inconnue choisie pour la résolution.

```
> solve(3*x+2=3+2*x);
```

$$1$$

```
> equ:=a*x+1=2+x: solve(equ,a), solve(equ,x);
```

$$\frac{1+x}{x}, \quad \frac{1}{a-1}$$

Finissons par remarquer que la commande **assume** n'est pas prise en compte par **solve**.

11. Cette écriture est en fait superflue. Ecrire **solve(p)** au lieu de **solve(p=0)** est accepté par le logiciel.

```
> equ:=z^2+1=0: assume(z,real): solve(equ);
```

$$-I, I$$

2.2. Résultats possibles d'une commande solve

Cas où il y a plusieurs solutions

Dans le cas de figure où l'équation admet plusieurs solutions, le logiciel les renvoie sous forme d'une séquence [12]. On pourra accéder à ces solutions en enregistrant le résultat de la commande **solve** dans une variable *sol* et en spécifiant *sol*[1], *sol*[2], etc.

```
> solve(x^2-5*x+6=0);
```

$$3,\ 2$$

```
> sol:=solve(x^2-5*x+6=0): sol[1];
```

$$3$$

Cas d'une infinité de solutions

Lorsque le nombre de solutions dépasse le contenu de la variable globale **_MaxSols**, le logiciel se contente de renvoyer une seule solution. On peut alors le forcer à rechercher toutes les solutions en changeant le valeur de la variable globale **_EnvAllSolutions**.

```
> solve(cos(x)=0);
```

$$\frac{1}{2}\pi$$

```
> _EnvAllSolutions:=true: solve(cos(x)=0);
```

$$\frac{1}{2}\pi + \pi\_Z1$$

On retiendra que les symboles **_Zn** et **_Bm** désignent respectivement $\mathbb{Z}$ et $\{0,1\}$, les indices dépassant le nombre 1 dès que ces ensembles apparaissent à plusieurs reprises.

12. Une séquence est une suite d'expressions séparées par des virgules.

Aucune réponse

Quand le logiciel n'a aucune réponse, il répondra à l'utilisateur par... rien avec le message *Warning, solutions may have been lost* ! Lorsqu'il n'y a aucune solution, le logiciel ne reverra rien.

```
> solve(cos(sqrt(x))=x^2);
Warning, solutions may have been lost

> solve(x=x+1);
```

Aucune réponse explicite mais un peu plus que rien...

Dans certains cas le logiciel, sans résoudre explicitement l'équation, renvoie l'expression **RootOf(...)** (signifiant *racines de...*)

```
> solve(exp(x)=ln(x));
```

$$e^{\text{RootOf}\left(\_Z - e^{e^{-Z}}\right)}$$

Cela ne nous avance guère... La commande **RootOf** n'est pas une fonction qui calcule les racines mais une expression qui représente formellement celle-ci. Elle peut avoir une certaine utilité pour effectuer des calculs formels dans manipuler les solutions exactes. On l'utilise alors avec la commande **alias** qui permet de créer des abréviations et la commande **evala** qui effectue des calculs dans la plus petite extension possible du corps de base.

```
> alias( a = RootOf(x^3+2*x-1) ): evala(a^3);
```

$$1-2a$$

Cependant, la commande **allvalues** permet d'exprimer à l'aide de radicaux (lorsque cela est possible !) les racines d'un polynôme :

```
> RootOf(x^3-5*x+2): allvalues(%);
```

$$\sqrt{2}-1,\ 2,\ -1-\sqrt{2}$$

2.3. Cas d'un système d'équations

La syntaxe sera la suivante :

Résolution formelle d'un système d'équations

solve({*équation*$_1$, …, *équation*$_n$}, {*inconnue*$_1$, …, *inconnue*$_m$})

```
> solve({x^3+y=0,x+y+z=0},{y,z});
```

$$\{y = -x^3,\ z = -x + x^3\}$$

2.4. Comment extraire les solutions ?

Partons de l'exemple suivant :

```
> equ := {a+b = 1, 2*a-3*b = 3}; solve(equ), a+b;
```

$$\left\{b = -\frac{1}{5}, a = \frac{6}{5}\right\}, a+b$$

On voit ici que le logiciel résout l'équation sans affecter les variables a et b puisqu'il ne retourne aucun résultat numérique pour $a+b$. Il y a cependant des situations où nous aurons besoin de l'expression des solutions pour continuer nos calculs. Dans ce cas, on pourra utiliser la commande **assign**.

```
> equ := {a+b = 1, 2*a-3*b = 3}; solve(equ);
```

$$\left\{b = -\frac{1}{5}, a = \frac{6}{5}\right\}$$

```
> sol:=solve(equ): assign(sol): b+a;
```

$$1$$

Mais cette technique a un inconvénient majeur : les expressions définies en fonction de a et b seront alors automatiquement évaluées pour les valeurs de a et b trouvées par la commande **solve** ce qui peut s'avérer problématique. On préférera donc utiliser de nouvelles variables pour enregistrer les solutions trouvées, on conservera ainsi a et b comme des inconnues (ie des variables libres). On utilisera pour cela la substitution.

```
> equ := {a+b = 1, 2*a-3*b = 3}; sol:=solve(equ);
```

$$sol := \left\{b = -\frac{1}{5}, a = \frac{6}{5}\right\}$$

```
> a1,b1:=subs(sol,a),subs(sol,b);
```

$$a1, b1 := -\frac{1}{5}, \frac{6}{5}$$

Dans le cas où l'équation admet plusieurs solutions, la commande **solve** les renvoie sous la forme d'une séquence, ie une énumération d'expressions séparées par des virgules. Pour une séquence sauvegardée dans une variable S, on accède au n-ième terme par la commande $S[n]$. Par exemple,

```
> S:=a,23,67,rt: S[3], S[2]+S[4];
```

$$67, 23 + rt$$

Voici un exemple avec la commande **solve**.

```
> equ := z^2+z+1 = 0; sol:=solve(equ,z);
```

$$sol := -\frac{1}{2} + I\frac{\sqrt{3}}{2}, -\frac{1}{2} - I\frac{\sqrt{3}}{2}$$

```
> sol[1]+sol[2], sol[1]*sol[2];
```

$$-1, 1$$

La commande assign

sol :=solve({*équation*$_1$, …, *équation*$_n$}, {*inconnue*$_1$, …, *inconnue*$_m$})
assign(sol) ;

2.5. Inéquations

La commande **solve** permet également la résolution des équations et des systèmes d'inéquations. La syntaxe est exactement celle des équations, nous n'y reviendrons pas. Les ensembles de solutions seront exprimés de la manière suivante : **RealRange(open(*a*),*b*)** signifie $]a,b]$, on écrira **RealRange(*a*,*b*)** pour le segment $[a,b]$, etc.

```
> solve(x^3+4*x+1<0);
```

$$\text{RealRange}\left(-\text{infinity},\text{Open}\left(-\frac{1}{6}(108+12\sqrt{849})^{1/3}+\frac{8}{(108+12\sqrt{849})^{1/3}}\right)\right)$$

2.6. Exercices

Exercice 7.

Dans $\mathbb{C}$ résoudre l'équation suivante

$$z^2+\bar{z}+iz=0.$$

Montrer que les points dont les affixes sont solutions forment un triangle rectangle.

Exercice 8.

On cherche à résoudre avec Maple l'équation suivante

$$\text{Re}(z^3)=\text{Im}(z^3)$$

d'inconnue complexe z.

1. Quel est le résultat donné par la commande **solve** ?

2. Résoudre cette équation en écrivant z sous forme algébrique.

3. Résoudre cette équation en écrivant z sous forme trigonométrique.

4. Résoudre l'équation

$$\text{Re}(z^5)=\text{Im}(z^5)$$

d'inconnue complexe z (cette question est indépendante des précédentes).

Exercice 9.

Soit (**E**) l'équation $x^3-5x+2=0$.

1. Résoudre exactement (**E**) au moyen de la commande **solve**.

2. Essayons à présent de *comprendre* comment Maple obtient ce résultat. Le logiciel applique la méthode de *Cardan*.

2.a. Soient deux réels u et v. Posons $x = u + v$. Prouver que *si*

$$u^3 + v^3 = -2 \text{ et } uv = \frac{5}{3}$$

alors x est solution de (**E**).

2.b. Résoudre l'équation

$$y^2 + 2y + \frac{125}{27} = 0.$$

2.c. En déduire les solutions de (**E**).

3. Résolution approchée : la commande fsolve

Une résolution n'étant pas toujours possible, le logiciel possède une commande de calcul approché des solutions d'une équation : **fsolve**, comme *floating solve*.

Résolution numérique d'une équation

fsolve(p=0, *inconnue de résolution*)

```
> eq:=x^3-x+1=0: fsolve(eq);
```

$$-1.324717957$$

Le nombre de chiffres significatifs peut être préciser au moyen de la variable globale **Digits** (voir page 8).

```
> eq:=x^3-x+1=0:Digits:=15: fsolve(eq);
```

$$-1.32471795724475$$

En cas de zéros multiples, la commande **fsolve** les renvoie sous forme d'une séquence. On pourra alors accéder à chacune d'entre-elles en utilisant la syntaxe des séquences.

```
> eq:=x^3-12*x+1=0: sol:=fsolve(eq);
```

$$\text{sol} := -3.505039725,\ 0.08338164256,\ 3.421658082$$

```
> sol[2];
```

$$0.08338164256$$

Il va de soit que le logiciel ne peut afficher une infinité de valeurs approchées. Lorsque l'équation en admettra une infinité, le logiciel ne donnera la valeur approchée que de l'une d'entre-elles.

```
> eq:=cos(x)=0: sol:=fsolve(eq);
```

$$sol := 1.570796327$$

Par défaut, les solutions renvoyées par la commande **fsolve** sont réelles. On pourra forcer le logiciel à retourner des valeurs approchées des racines complexes au moyen de l'option *complex*. Attention toutefois, cette option doit absolumment figuere en troisième position dans **fsolve**, on fera donc systématiquement figurer x (ou plus généralement l'inconnue choisie) en deuxième argument.

```
> fsolve(x^3+3*x-2=0,x,complex);
```

$$-0.2980358190 - 1.807339494I,\ -0.2980358190 + 1.807339494I,\ 0.5960716380$$