1 Généralités

1.1 feuille de calcul

• « ; » exécute une commande en affichant le résultat. Par exemple `1+2/3;` affichera comme résultat $\frac{5}{3}$
• «$ » exécute une commande sans afficher le résultat. Par exemple `a:2 $`
• « % » rappelle le dernier calcul effectué
•`? plot2d` affiche l'aide en ligne sur l'instruction `plot2d`. Pour ouvrir une nouvelle fenêtre d'aide, mieux vaut saisir `plot2d` suivi de l'appui sur la touche F1.
• `example(expand)` affiche des exemples d'utilisation de l'instruction `expand`
• `kill(all)` réinitialise le système

1.2 opérateurs

• les quatre opérations usuelles `+ , - , * , /`
• opérateur « ^ » élévation à une puissance. `x^3` est x^3
• opérateur « # » non égal à (ou différent de)
• opérateurs de comparaison `= , < , <= , > , >=`
• opérateur « : » d'affectation.
`a:3` donne la valeur 3 à la variable a.
• opérateur «:= » pour définir une fonction.
• opérateur « = » indique une équation dans Maxima.
• opérateur « ! » factorielle d'un entier naturel,
par exemple $5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$.

1.3 constantes

• `%pi` désigne $\pi \approx 3,14159$.
• `%e` désigne $e = \exp(1) \approx 2,7183$.
• `%i` est l'imaginaire pur de module 1, d'argument $\pi/2$.
• `true` valeur booléenne «vrai».
• `false` valeur booléenne «faux».
• `inf` désigne $+\infty$.
• `minf` désigne $-\infty$.
• `%gamma` constante d'Euler-Mascheroni qui est la limite de la suite de terme général $\left(\sum_{k=1}^{n} \frac{1}{k}\right) - \ln n$.

1.4 listes

Une liste est un type de données, qui tient compte de l'ordre, accepte les répétitions d'éléments et est délimité par les caractères [et]. Voici quelques fonctions importantes concernant les listes :
• `L:makelist(k^2,k,0,9)` permet de créer la liste des carrés des 10 premiers naturels, k prenant toutes les valeurs entières de 0 jusqu'à 9.
• `L[2]:5` remplace le 2ème élément de la liste L par 5.
• `length(L)` donne le nombre d'éléments de la liste L.
• `first(L) ; second(L) ; last(L)` renvoient respectivement le premier, le second, le dernier élément de L.
• `member(x,L)` vaut `true` si x appartient à la liste L (`false` sinon).
• `append([a,1,3],[2,7])` regroupe les deux listes en une seule liste $[a, 1, 3, 2, 7]$ par concaténation.
• `join(l,m)` crée une nouvelle liste constituée des éléments des listes l et m, intercalés. La liste obtenue est $[l[1], m[1], l[2], m[2], l[3], m[3], \ldots]$.
• `sort(L)` permet de ranger les éléments de la liste L par ordre croissant.
• `L:f(L)` permet d'appliquer la fonction f à tous les éléments de la liste L.
• `listify(E)` retourne une liste formée des éléments de l'ensemble E lorsque E est un ensemble.

2 Calculs élémentaires

2.1 nombres entiers et arithmétique

Soit a et b deux entiers. Soit n et p deux entiers naturels.
• `divide(a,b)` division euclidienne de a par b.
Le résultat est une liste dont le premier élément est le quotient et le second élément le reste.
• `divisors(a)` ensemble des diviseurs positifs de a.
• `divsum(a)` somme des diviseurs positifs de a.
• `mod(a,b)` reste de la division euclidienne de a par b.
• `gcd(a,b)` pgcd de a et b.
• `lcm(a,b)` ppcm de a et b.
• `primep(p)` teste si p est premier.
• `p:prev_prime(n)` donne le nombre premier p qui vient juste avant n, avec $p < n$ et $n \geq 3$.
• `next_prime(n)` donne le nombre premier qui vient juste après n avec $p > n$.
• `factor(n)` décompose n en produit de facteurs premiers.
• `ifactors(n)` décompose n en produit de facteurs premiers en affichant le résultat sous forme de liste.
• `factorial(7)` retourne 7!=5040.
• `binomial(n,p)` est le coefficient binomial $\binom{n}{p}$.
• `permutation(n,p)` est le nombre d'arrangements A_n^p de p éléments pris parmi n.
• `random(n)` renvoie un entier naturel, choisi au hasard entre 0 et $n-1$ lorsque $n \in \mathbb{N}^*$.

2.2 fonctions usuelles

• `abs(x)` valeur absolue de x.
• `floor(x)` partie entière de x.
• `sqrt(x)` racine carrée de x.
• `sin(x) , cos(x) , tan(x)`
• `exp(x) , log(x)` *Attention* : log désigne la fonction **logarithme népérien**.
• Fonctions trigonométriques circulaires réciproques :
`asin(x) , acos(x) , atan(x)`
• Fonctions trigonométriques hyperboliques :
`sinh(x) , cosh(x) , tanh(x)`
• Fonctions hyperboliques récirpoques :
`asinh(x) , acsh(x) ,atanh(x)`

2.3 valeurs approchées

- `float(x)` fournit une valeur décimale approchée de x.
- `bfloat(x)` donne une valeur approchée de x en notation scientifique.
- `fpprec:20` fixe la précision de la valeur approchée donnée par `bfloat` (20 chiffres affichés au lieu de 16 par défaut).

2.4 trigonométrie

- `trigexpand(a)` développe l'expression trigonométrique a en utilisant les formules d'addition de cos et sin. Par exemple, `trigexpand(cos(x+y))` renvoie $\cos x \cos y - \sin x \sin y$.
- `trigreduce(a)` permet de linéariser un polynôme trigonométrique a. Par exemple, `trigreduce(sin(x)^3)` renvoie $\dfrac{3\sin x - \sin(3x)}{4}$.
- `trigsimp(a)` simplifie l'expression trigonométrique a en utilisant la relation $\cos^2 t + \sin^2 t = 1$ et en remplaçant $\tan t$ par $\dfrac{\sin t}{\cos t}$.
- `load(ntrig)` charge le paquetage permettant d'obtenir les valeurs exactes de $\sin x$, $\cos x$ et $\tan x$ lorsque x est un multiple de $\pi/10$.

2.5 nombres complexes

Soit z un nombre complexe.
- `realpart(z)` partie réelle de z.
- `imagpart(z)` partie imaginaire de z.
- `conjugate(z)` conjugué de z.
- `abs(z)` module de z.
- `carg(z)` argument de z (dans $]-\pi, \pi]$).
- `rectform(z)` écrit z sous forme algébrique.
- `polarform(z)` écrit z sous forme exponentielle.
- `exponentialize(cos(x))` convertit les fonctions trigonométriques en exponentielles (avec les formules d'Euler).

3 Polynômes à une indéterminée

Soit P et Q deux polynômes.
- `P:x^4-4*x^2-5` définit le polynôme $P = X^4 - 4X^2 - 5$.
- `expand(P)` développe P.
- `factor(P)` factorise P dans $\mathbb{R}[X]$.
- `gfactor(P)` factorise parfois P dans $\mathbb{C}[X]$.
- `solve(P,x)` retourne les racines complexes de P.
- `allroots(P)` calcule des valeurs approchées des racines complexes de P.
- `ratcoef(P,x^3)` renvoie le coefficient du terme en X^3 de P.
- `divide(P,Q,x)` calcule le quotient et le reste de la divison de P par Q. Le résultat est une liste dont le premier élément est le quotient et le second élément le reste.
- `partfrac(P/Q,x)` décompose la fonction rationnelle P/Q (de la variable x) en éléments simples.
- `ratsimp(expr)` simplifie l'expression `expr` (en écrivant tout sur le même dénominateur).
- `subst(4,x,P)` calcule la valeur de P quand $X = 4$.
- `subst(1/z,x,expr)` remplace x par $1/z$ dans l'expression `expr`.

4 Fonctions réelles

4.1 définir une fonction

- `f(x):=x^2+2*x-3`
- `define(f(x),x^2+2*x-3)`
- `f:lambda([x],x^2+2*x-3)`

4.2 courbes représentatives

Pour afficher les courbes $\mathcal{C}_f$ et $\mathcal{C}_g$ sur le même graphique, dans la fenêtre $[x_1, x_2] \times [y_1, y_2]$, on entre :
- `plot2d([f(x),g(x)],[x,x1,x2],[y,y1,y2])`

Sur un exemple, avec plus d'options (grille, graduations choisies, axes du repère apparents) :
- `plot2d([atan(x),x],[x,-4,10],[y,-1.5,1.5], grid2d, [xtics,2],[ytics,1],[axes,solid])`

4.3 limites

- `limit(sin(x)/x,x,0)` limite en 0
- `limit(1/x,x,0,plus)` limite à droite en 0
- `limit(1/x,x,0,minus)` limite à gauche en 0
- `limit(x*exp(x),x,minf)` limite en $-\infty$

4.4 dérivation et développements limités

- `diff(f(x),x)` calcule la dérivée $f'(x)$.
- `define(g(x),diff(f(x),x))` définit la fonction g comme étant la dérivée de f.
- `diff(f(x),x,2)` calcule $f''(x)$, dérivée seconde.
- `taylor(log(x),x,2,3)` affiche le développement limité de la fonction ln au voisinage de 2, à l'ordre 3.
- `y=expand(taylor(f(x),x,a,1))` permet d'obtenir l'équation réduite de la tangente à $\mathcal{C}_f$ au point $A(a, f(a))$

4.5 intégration

- `integrate(f(x),x)` calcule une primitive de la fonction f
- `integrate(f(x),x,a,b)` calcule l'intégrale $\displaystyle\int_a^b f(x)\,\mathrm{d}x$
- `quad_qags(1/log(x),x,2,3)` fournit une approximation de l'intégrale $\displaystyle\int_2^3 \frac{1}{\ln x}\,\mathrm{d}x$.

- Pour détailler un *changement de variable* dans une intégrale :
- `a:'integrate(sqrt(x)/(1+x),x,1,3)$ changevar(a,u=sqrt(x),u,x)`

4.6 fonctions de deux variables

- Définir une fonction de plusieurs variables :
• `f(x,y):=x^ 3*log(y)`

- Surface représentative en 3D :
• `plot3d(f(x,y), [x,-4,4], [y,0,6],[grid,40,40], [plot_format,gnuplot])$`

- Dérivées partielles d'ordre 1, d'ordre 2 :
• `diff(f(x,y),x,1)` renvoie $\frac{\partial f}{\partial x}(x,y)$
• `diff(f(x,y),x,1,y,1)` renvoie $\frac{\partial^2 f}{\partial y \partial x}(x,y)$

- Développement limité à l'ordre 2 de f au vosiniage de $a=(2,3)$:
• `taylor(f(x,y),[x,y],[2,3],2)` ou encore
• `taylor(f(2+k,3+l),[k,l],[0,0],2)`

4.7 séries de Fourier

Soit $f:\mathbb{R}\to\mathbb{R}$ une fonction $T-$périodique telle que pour tout $x\in[-T/2\,,\,T/2]$, $f(x)=g(x)$.
• `load(fourie)$ fourier(g(x),x,T/2)` retourne la liste des coefficients de Fourier trigonométriques de f.

5 Équations

5.1 résolution d'équations

Résolution exacte dans l'ensemble $\mathbb{C}$ des complexes :
• `solve(x^ 2+x=1,x)`
Résolution approchée dans $\mathbb{R}$:
• `find_root(x^ 5=1+x,x,1,2)` solution dans $[1,2]$
Racines d'un polynôme :
• `allroots(x^ 5=1+x)` trouve des valeurs approchées de toutes les solutions (réelles et complexes) de l'équation polynomiale $x^5=1+x$.

5.2 systèmes linéaires

Pour résoudre le système $\begin{cases} 3x+2y=1 \\ x-y=2 \end{cases}$
• `s1:[3*x+2*y=1,x-y=2]`
• `linsolve(s1,[x,y])`

5.3 équations différentielles

Pour résoudre l'équation différentielle $y''+w^2\,y=\sin t$, on définit d'abord l'équation :
• `eqn:'diff(y,t,2)+w^ 2*y=sin(t)`
On la résout :
• `sol:ode2(eqn,y,t)`
Pour trouver la solution satisfaisant aux conditions initiales $y(0)=1$ et $y'(0)=-1$, on entre :
• `ic2(sol,t=0,y=1,diff(y,t)=-1)`
Pour trouver la solution satisfaisant aux conditions $y(0)=1$ et $y(1)=0$, on entre :
• `bc2(sol,t=0,y=1,t=1,y=0)`
• `rhs(sol)` saisit le membre de droite de l'égalité `sol` obtenue ci-dessus.

5.4 système différentiel linéaire

Pour résoudre le système différentiel linéaire homogène $\begin{cases} x'=x+8y \\ y'=y+2x \end{cases}$, on définit d'abord les équations :
• `eqn_1:'diff(x(t),t)=x(t)+8*y(t)$`
`eqn_2:'diff(y(t),t)=2*x(t)+y(t)$`
On fait résoudre le système :
• `desolve([eqn_1,eqn_2],[x(t),y(t)])`
Avec les conditions initiales $x(0)=2$ et $y(0)=3$:
• `atvalue(x(t),t=0,2)$ atvalue(y(t),t=0,3)$`
`desolve([eqn_1,eqn_2],[x(t),y(t)]`

6 Suites et séries

6.1 définir une suite récurrente

Pour définir la suite (u_n) par
$u_0=1$ et $\forall n\in\mathbb{N},\ u_{n+1}=\frac{u_n}{1+n\,u_n}$, on entre :
• `u[n]:=if n=0 then 1`
`else u[n-1]/(1+(n-1)*u[n-1])`
• `L:makelist(u[k],k,0,10)` créée la liste liste L des 11 premiers termes de la suite (u_n).

6.2 représenter graphiquement une suite

• `plot2d([discrete,L],[style,points])` place dans un repère les points $M_k(k,u_k)$ de la suite ci-dessus.

- Dans le cas d'une suite récurrente du type $u_{n+1}=f(u_n)$, pour représenter sur un même graphique la fonction $f:x\mapsto\sqrt{1+x}$, la droite d'équation $y=x$ et les 10 premiers points de coordonnées $(u_k\,,\,f(u_k))$ sachant que $u_0=1/2$:
• `staircase(sqrt(1+x),1/2,10,[x,0,2],[y,0,2])`

6.3 expliciter le terme de rang n

Pour obtenir le terme de rang n de la suite de Fibonacci (v_n) définie par
$v_0=0\,,\ v_1=1$ et $\forall n\in\mathbb{N},\ v_{n+2}=v_n+v_{n+1}$:

• `load(solve_rec) $`
`solve_rec(v[n+2]=v[n]+v[n+1],v[n],v[0]=0,v[1]=1)`

6.4 sommes et produits finis

• `sum(1/k^ 2,k,1,10)` calcule la somme des inverses des carrés des entiers compris entre 1 et 10.
• `product(sqrt(k),k,1,10)` calcule le produit des racines carrées des entiers compris entre 1 et 10.

6.5 somme d'une série convergente

On sait que la série $\sum \frac{1}{n^2}$ est convergente. Sa somme est notée $\sum_{n=1}^{+\infty} \frac{1}{n^2}$. On peut demander sa valeur exacte comme suit :

- `load(simplify_sum)$ S:sum(1/k^2,k,1,inf)$ simplify_sum(S);`

7 Matrices

7.1 création d'une matrice

- En entrant chaque coefficient

on définit la matrice $A = \begin{pmatrix} -1 & 2 & 0 \\ 2 & 2 & -3 \\ -2 & 2 & 1 \end{pmatrix}$ ligne par ligne de la façon suivante :

- `A:matrix([-1,2,0],[2,2,-3],[-2,2,1])`

- À l'aide d'une fonction

- `M:genmatrix(lambda([i,j],i+j),3,2)` permet de créer la matrice $M \in \mathfrak{M}_{3,2}(\mathbb{R})$ dont le terme général est $m_{i,j} = i + j$.

- Extraction de ligne ou de colonne

- `A[i]` renvoie la $i-$ème ligne de A.
- `col(A,j)` renvoie la $j-$ème colonne de A.

7.2 des matrices particulières

- `zeromatrix(5,3)` retourne la matrice nulle à 5 lignes et 3 colonnes.
- `ident(5)` fournit la matrice identité I_5.
- `diag_matrix(a,b,c)` est la matrice diagonale

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$$

7.3 opérations sur les matrices

Soit A et B deux matrices de même taille.

- `A+B` somme des matrices A et B.
- `3*A` produit de la matrice A par le réel 3.
- `A.B` produit des matrices A et B.
- `transpose(A)` transposée de la matrice A.
- `rank(A)` rang de la matrice A.
- `A^^3` matrice A élevée à la puissance 3 si A est carrée.
- `invert(A)` inverse A^{-1} de la matrice carrée A
- `determinant(A)` déterminant de A si A est carrée.
- `mat_trace(A)` trace de A si A est carrée.

7.4 matrices et applications linéaires

Soit f une application linéaire de $\mathbb{R}^p$ dans $\mathbb{R}^n$ dont la matrice dans les bases canoniques est A de taille $n \times p$.

- `nullspace(A)` retourne une base du noyau de f.
- `columnspace(A)` retourne une base de l'image de f.

7.5 réduction des matrices carrées

- `factor(charpoly(A,x))` renvoie le polynôme caractéristique de la matrice A sous forme factorisée.
- `eigenvalues(A)` renvoie la liste des valeurs propres de A suivies de leurs multiplicités respectives.
- `eigenvectors(A)` renvoie une base de chaque sous-espace propre de A.

8 Programmation

8.1 syntaxe d'une procédure

```
nom(paramètres en entrée):= block([variables locales],
<instruction 1>,<instruction 2>, ...
/* ----Commentaire--- */     )$
```

Exemple d'une procédure qui additionne deux nombres.

- `somme(a,b):=block([c], c:a+b, return(c))`

8.2 structure conditionnelle

```
• if (condition1)
then (<instruction1> , <instruction2>)
else if (condition2) then (<instruction3>)
else (<instruction4> , <instruction5>)
```

8.3 structures itératives

Boucle **For** et affichage de la table de 7 :

```
• for k from 1 thru 10 do
( print("7 fois",k,"égale",7*k) )
```

Boucle **While** et affichage de la table de 7 :

```
• k:1 $ while k<11 do
( print("7 fois",k,"égale",7*k) , k:k+1 )
```

8.4 utilisation d'hypothèses

$\sqrt{y^2}$ n'est pas toujours égal à y :

- `assume(y<0)$ is(y=sqrt(y^2)); forget(y<0)$`

8.5 déclaration de types

- `declare(n,integer)$ cos(2*n*%pi)`