Tutoriel Maxima pour enseigner les mathématiques en BTS

Thomas BRÉLIVET

12 juin 2014

Document en cours d'élaboration.

Pour toute question ou remarque, envoyez un email : thomas.brelivet@ac-creteil.fr

Table des matières

Chapitre 1

Première prise en main de Maxima

Maxima (http ://maxima.sourceforge.net/) est un logiciel de calcul formel, descendant du logiciel Macsyma développé dès 1968 au MIT (institut de recherche et université américaine). Il est disponible sous linux, Mac OS X et Windows. C'est un logiciel libre distribué sous licence GNU GPL, programmé dans le langage Lisp.

Ce logiciel possède une interface graphique qui est wxmaxima. C'est cette interface graphique avec maxima que l'on utilise ici.

Le logiciel wxmaxima se présente graphiquement en deux parties : la zone de menu et la zone d'exécution des commandes. La zone de menu, comme beaucoup de logiciels comporte les sous menus « Fichiers », « Edition » et « Aide ». Dans la zone d'exécution des commandes, on peut écrire du texte comme ce que vous êtes en train de lire et aussi bien sûr exécuter des commandes !

Les commandes sont toujours précédées d'une flêche lorsqu'elles sont vides ou de %i et un nombre entouré par des parenthèses lorsquelles contiennent une commande qui a été exécutée. Si la commande (ci-dessous) n'a pas été exécutée, il faut mettre le curseur dessus et appuyer simultanément les touches Majuscule et entrée (Maj+Entrée).

```
(%i1)  2+5;
```

(%o1) 7

La lettre i signifie input (entrée) et le o output (sortie). Le pourcentage sert à dire que i1 est un nom, c'est le nom de la première entrée. On peut les utiliser dans la suite des commandes. Un poucentage seul est le nom du dernier résultat obtenu.

```
(%i2)  %;
```

(%o2) 7

```
(%i3)  %o1;
```

(%o3) 7

```
(%i4)  %i1;
```

(%o4) 7

Le point virgule « ; » marque la fin de la commande à éxécuter. À la place du point virgule, on peut mettre un dollar pour ne pas afficher le résultat. Cela peut être utile si l'on ne souhaite pas faire apparaître de résultats intermédiaires.

```
(%i5)  2+5$
```

```
(%i6)  %i5+2;
```

(%o6) 9

On peut aussi écrire plusieurs commandes dans une même cellule :

```
(%i7)  2+7;
       7/8;
```

(%o7) 9

(%o8) $\frac{7}{8}$

Si on éxécute plusieurs fois une commande, la numérotation augmente et certains numéros disparaissent. On peut remettre à jour la numérotation en utilisant "Menu/Maxima/Redémarrer Maxima" puis "Menu/Cell/Réévaluer toutes les cellules". Avec le clavier on peut utiliser : Alt+M+R puis CTRL+R.

Chapitre 2

Les nombres

2.1 Opérations sur les nombres

2.1.1 Somme et différence

```
(%i1)  2+3;
       2-3;
```

(%o1) 5

(%o2) -1

2.1.2 Produit et division

```
(%i3)  5*45;
```

(%o3) 225

```
(%i4)  -63/49;
```

(%o4) $-\frac{9}{7}$

2.1.3 Arrondis

Si on veut un arrondi, on utilise la commande float :

```
(%i5)  float(%);
```

(%o5) -1.285714285714286

On peut l'avoir directement si on écrit un des nombres sous forme décimale :

```
(%i6)  -63.0/49;
       -63/49.0;
```

(%o6) -1.285714285714286

(%o7) -1.285714285714286

2.2 Des fonctions sur les entiers

2.2.1 Factorisation

```
(%i8)  factor(6);
       factor(45454566780);
```

(%o8) $2\,3$

(%o9) $2^2\,3^2\,5\,7^2\,19\,271241$

2.2.2 Racine carrée

La racine est un entier

```
(%i10) sqrt(4);
```

(%o10) 2

La racine est non entière et le résultat donne la racine carrée :

```
(%i11) sqrt(2);
```

(%o11) $\sqrt{2}$

Les racine est non entière et on obtient une valeur approchée :

```
(%i12) sqrt(2.0);
```

(%o12) 1.414213562373095

Valeur exacte écrite sous forme de puissance

```
(%i13) sqrt(8);
```

(%o13) $2^{\frac{3}{2}}$

Valeur approchée

```
(%i14) sqrt(8.0);
```

(%o14) 2.82842712474619

2.2.3 Puissances

```
(%i15) 2^3;
```

(%o15) 8

```
(%i16) 45^3;
```

(%o16) 91125

2.2.4 Valeur absolue

Valeur absolue :

```
(%i17) abs(-4);
```

(%o17) 4

2.2.5 Factorielle

La factorielle d'un nombre

```
(%i18) 3!;
```

(%o18) 6

```
(%i19) 50!;
```

(%o19) 30414093201713378043612608166064768844377641568960512000000000000

2.3 Représentation des nombres

La notation scientifique

```
(%i20) sqrt(23.0)^2-23;
```

(%o20) $-3.5527136788005009\,10^{-15}$

Si l'on souhaite avoir d'avantage de précision, on indique la précision souhaitée, par exemple avec 100 chiffres après la décimale :

```
(%i21) fpprec:100;
```

(%o21) 100

On utilise ensuite la commande bfloat :

```
(%i22) bfloat(sqrt(4700000));
```

(%o22) $2.16794833886787994189896244 80[43 digits]41853792930268502274725809 43b3$

où b3 signifie que le résultat est multiplié par 10^3 (il y a 43 chiffres qui ne sont pas affichés, ils sont remplacés par [43 digits]). On peut aussi écrire la commande en une seule ligne :

```
(%i23) sqrt(8), bfloat, fpprec:50;
```

(%o23) $2.8284271247461900976033774484193961571393437505 39b0$

Chapitre 3

Les variables

3.1 Les variables

Déclaration d'une variable

```
(%i1)  a:2;
```

(%o1) 2

Maintenant la variable a, contient la valeur 2 :

```
(%i2)  a;
```

(%o2) 2

Si l'on souhaite désaffecter la variable a :

```
(%i3)  kill(a);
```

(%o3) *done*

```
(%i4)  a;
```

(%o4) a

On peut développer :

```
(%i5)  expand((a+1)^2);
```

(%o5) $a^2 + 2\,a + 1$

On peut aussi factoriser :

```
(%i6)  factor(a^2+5*a+6);
```

(%o6) $(a + 2)\,(a + 3)$

Les identités remarquables avec la commande expand :

```
(%i7)  expand((a+b)^2);
```

(%o7) $b^2 + 2\,a\,b + a^2$

```
(%i8)  expand((a-b)^2);
```

(%o8) $b^2 - 2\,a\,b + a^2$

```
(%i9)  expand((a-b)*(a+b));
```

(%o9) $a^2 - b^2$

Les identités remarquables avec la commande factor :

```
(%i10) factor(a^2+2*a*b+b^2);
```

(%o10) $(b + a)^2$

```
(%i11) factor(a^2-2*a*b+b^2);
```

(%o11) $(b - a)^2$

```
(%i12) factor(a^2-b^2);
```

(%o12) $-(b - a)\,(b + a)$

Après utilisation de declare(a,mainvar) ; afin d'avoir la variable a en premier :

```
(%i13) declare(a,mainvar);
```

(%o13) *done*

```
(%i14) expand((a+b)^2);
       expand((a-b)^2);
       expand((a-b)*(a+b));
```

(%o14) $a^2 + 2\,b\,a + b^2$

(%o15) $a^2 - 2\,b\,a + b^2$

(%o16) $a^2 - b^2$

```
(%i17) factor(a^2+2*a*b+b^2);
       factor(a^2-2*a*b+b^2);
       factor(a^2-b^2);
```

(%o17) $(a + b)^2$

(%o18) $(a - b)^2$

(%o19) $(a - b)\,(a + b)$

3.2 Les constantes

Le nombre PI

```
(%i20) %pi;
```

(%o20) π

```
(%i21) float(%pi);
```

(%o21) 3.141592653589793

Le nombre e

```
(%i22) %e;
```

(%o22) e

```
(%i23) float(%e);
```

(%o23) 2.718281828459045

3.3 Substitutions de variables

Il peut parfois être utilise de remplacer une variable par une expression ou par une valeur. La fonction subst permet de réaliser cette action :

```
(%i24) subst(x^2, a,(a+1)^2);
```

(%o24) $\left(x^2+1\right)^2$

```
(%i25) subst(2, a,(a+1)^2);
```

(%o25) 9

Chapitre 4

Les équations

4.1 Les équations

4.1.1 Définition d'une équation

On commence par définir une équation et on lui donne un nom :

```
(%i1)  equation1:x^2-1=0;
```

(%o1) $x^2 - 1 = 0$

4.1.2 Résolution exacte d'une équation

On résout l'équation suivante, où l'inconnue est x (dans le menu : Menu/Equations/Résoudre...) :

```
(%i2)  solve(equation1,x);
```

(%o2) $[x = -1, x = 1]$

On peut aussi l'écrire directement :

```
(%i3)  solve(x^2-1=0,x);
```

(%o3) $[x = -1, x = 1]$

4.1.3 Utilisation des solutions obtenues

Les résultats précédents sont donnés sous forme d'équation $x = \ldots$.

Pour obtenir la première équation, on écrit :

```
(%i4)  solve(x^2-1=0,x)[1];
```

(%o4) $x = -1$

Pour la deuxième :

```
(%i5)  solve(x^2-1=0,x)[2];
```

(%o5) $x = 1$

Pour récupérer le membre de droite, on utilise rhs (right hand side) :

```
(%i6)  rhs(solve(x^2-1=0,x)[1]);
       rhs(solve(x^2-1=0,x)[2]);
```

(%o6) -1

(%o7) 1

Si on souhaite récupérer la variable, on utilise lhs (left hand side) :

```
(%i8)  lhs(solve(x^2-1=0,x)[1]);
       lhs(solve(x^2-1=0,x)[2]);
```

(%o8) x

(%o9) x

Pour vérifier qu'un nombre annule une expression :

```
(%i10) subst(-1, x, x^2-1);
```

(%o10) 0

4.1.4 Résolution approchée d'une équation

Si une solution exacte de l'équation n'existe pas, on obtient l'équation :

```
(%i11) solve(x^7-2*x+5=0,x);
```

(%o11) $[0 = x^7 - 2\,x + 5]$

Dans ce cas on utilise une méthode de résolution numérique, il faut donner un intervalle où la fonction change une seule fois de signe.

```
(%i12) find_root(x^7-2*x+5=0, x, -2, 0);
```

(%o12) -1.337973214895086

En changeant l'intervalle, on peut obtenir toutes les solutions :

```
(%i13) find_root(x^5+20*x^2+2*x-1=0, x, -10, 10);
       find_root(x^5+20*x^2+2*x-1=0, x, -10, -1);
       find_root(x^5+20*x^2+2*x-1=0, x, -1, 0);
```

(%o13) 0.17910867241289

(%o14) -2.673637481699788

(%o15) -0.27931420183824

4.1.5 Résolution d'équations polynomiales

Si l'équation est polynomiale, on dispose de realroots :

```
(%i16) realroots(x^5+20*x^2+2*x-1);
```

(%o16) $[x = -\frac{89712387}{33554432}, x = -\frac{9372229}{33554432}, x = \frac{6009889}{33554432}]$

Pour obtenir les solutions réelles approchées, on utilise float :

```
(%i17) float(realroots(x^5+20*x^2+2*x-1));
```

(%o17) $[x = -2.673637479543686, x = -0.27931419014931, x = 0.17910864949226]$

Par contre solve ne trouve pas ces solutions !

```
(%i18) solve(x^5+20*x^2+2*x-1=0, x);
```

(%o18) $[0 = x^5 + 20\,x^2 + 2\,x - 1]$

4.2 Les inéquations

Pour les inéquations, il existe le package solve_rat_ineq qui fonctionne pour les expressions rationnelles.

```
(%i19) load(solve_rat_ineq)$
```

```
(%i20) solve_rat_ineq((2*x+3)*(x)>0);
```

(%o20) $[[x < -\frac{3}{2}], [x > 0]]$

```
(%i21) solve_rat_ineq((x-1)^2*(x+1)^2>0);
```

(%o21) $[[x < -1], [x > -1, x < 1], [x > 1]]$

Chapitre 5

Les fonctions

5.1 Définition d'une fonction

(%i1)
```
f(x):=x^2;
```

(%o1) $\mathrm{f}\,(x) := x^2$

Avec define :

(%i2)
```
define(g(x), x^3-1);
```

(%o2) $\mathrm{g}\,(x) := x^3 - 1$

5.2 Tableau de valeurs d'une fonction

On utilise les listes :

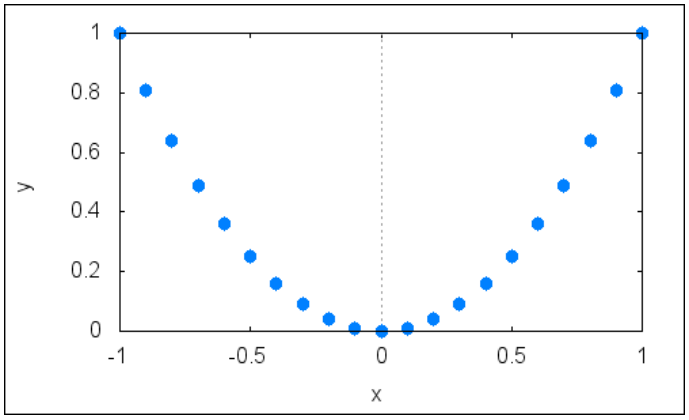
(%i3)
```
valeursdex:makelist(-1+k/10,k,0, 20);
imagesdesvaleursdex:map(f,valeursdex);
```

(%o3) $[-1, -\frac{9}{10}, -\frac{4}{5}, -\frac{7}{10}, -\frac{3}{5}, -\frac{1}{2}, -\frac{2}{5}, -\frac{3}{10}, -\frac{1}{5}, -\frac{1}{10}, 0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1]$

(%o4) $[1, \frac{81}{100}, \frac{16}{25}, \frac{49}{100}, \frac{9}{25}, \frac{1}{4}, \frac{4}{25}, \frac{9}{100}, \frac{1}{25}, \frac{1}{100}, 0, \frac{1}{100}, \frac{1}{25}, \frac{9}{100}, \frac{4}{25}, \frac{1}{4}, \frac{9}{25}, \frac{49}{100}, \frac{16}{25}, \frac{81}{100}, 1]$

(%i5)
```
wxplot2d([discrete, valeursdex,imagesdesvaleursdex],[style, points]);
```

(%t5)

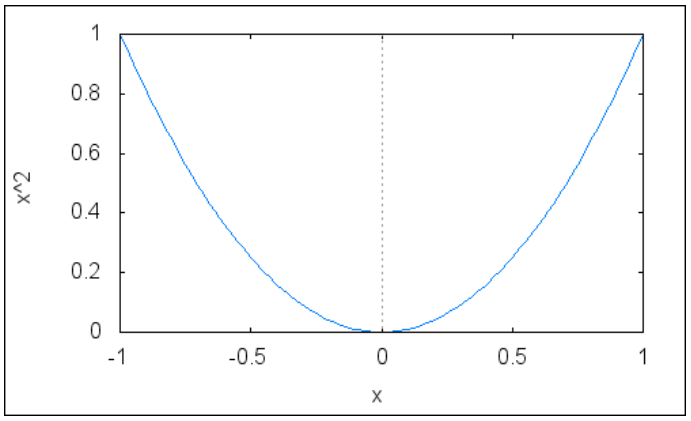


(%o5)

5.3 Courbe représentative d'une fonction

On donne l'expression de f(x) et l'intervalle sur lequel on souhaite représenter la fonction. L'intervalle des coordonnées se calcule automatiquement.

```
(%i6)  wxplot2d(f(x),[x,-1,1]);
```

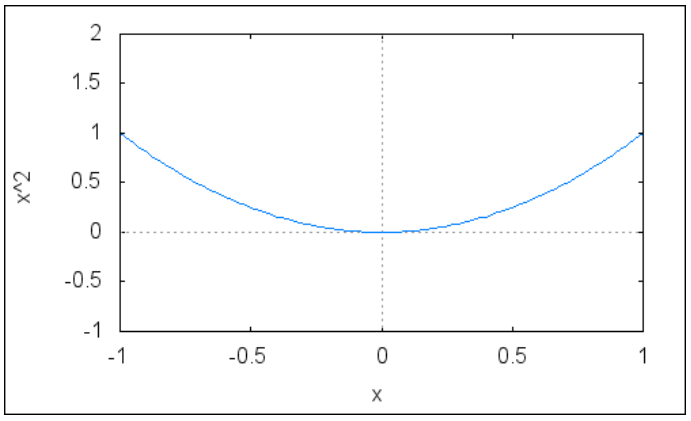


(%t6)

(%o6)

On peut aussi donner cet intervalle.

```
(%i7)  wxplot2d(f(x),[x,-1,1], [y, -1,2]);
```

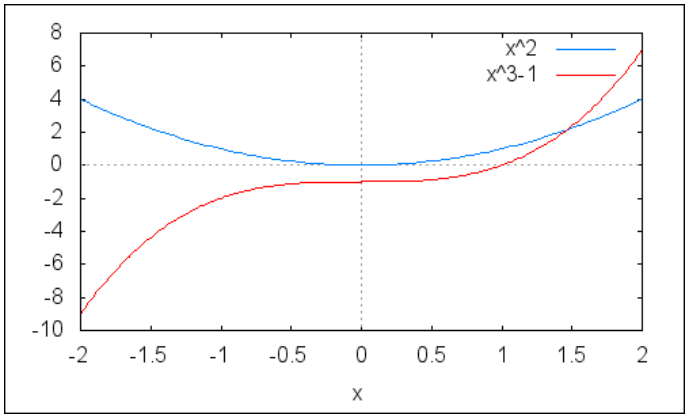


(%t7)

(%o7)

Pour tracer plusieurs courbes à la fois, on donne la liste des expressions des fonctions à représenter.

```
(%i8)  wxplot2d([f(x),g(x)],[x,-2,2]);
```



(%t8)

(%o8)

5.4 Dérivée d'une fonction

```
(%i9)  define(f(x),x^20+x^15);
       diff(f(x),x);
       define(df(x),diff(f(x),x));
```

(%o9) $\mathrm{f}\left(x\right) := x^{20} + x^{15}$

(%o10) $20\,x^{19} + 15\,x^{14}$

(%o11) $\mathrm{df}\left(x\right) := 20\,x^{19} + 15\,x^{14}$

Dérivée seconde :

```
(%i12) diff(f(x),x,2);
```

(%o12) $380\,x^{18} + 210\,x^{13}$

Dérivées d'ordre supérieur :

```
(%i13) diff(f(x),x,5);
```

(%o13) $1860480\,x^{15} + 360360\,x^{10}$

Remarque : pour définir la dérivée, il faut utiliser define et non := Voici deux exemples illustrant une différence :

```
(%i14) g(x):=x^2;
       h(x):=x^3;
```

(%o14) $\mathrm{g}\left(x\right) := x^2$

(%o15) $\mathrm{h}\left(x\right) := x^3$

```
(%i16) dg(x):=diff(g(x),x);
       define(dh(x), diff(h(x),x));
```

(%o16) $\mathrm{dg}\left(x\right) := \mathrm{diff}\left(\mathrm{g}\left(x\right), x\right)$

(%o17) $\mathrm{dh}\left(x\right) := 3\,x^2$

Jusqu'ici pas de différence, en effet :

```
(%i18) dg(x);
       dh(x);
```

(%o18) $2\,x$

(%o19) $3\,x^2$

mais pour le calcul en une valeur := donne une erreur :

```
(%i20) dg(2);
       dh(2);
```

$diff$: $secondargumentmustbeavariable; found"2\#0"$: $dg(x = 2) - -anerror.Todebugthistry$: $debugmode(true)$;

(%o21) 12

Lorsque l'on demande dg(2), Maxima exécute la commande diff(g(2), 2) ce qui n'est pas possible car le deuxième paramètre de diff doit-être une variable. On dérive par rapport à une variable.

5.5 Limites d'une fonction

En plus l'infini

```
(%i22) limit(f(x),x,inf);
```

(%o22) ∞

Lorsque le signe n'est pas précisé, c'est plus l'infini. Lorsque la limite est indéfinie et infinie en valeur absolue, on obtient le mot "infinity" :

```
(%i23) limit(1/x,x,0);
```

(%o23) $infinity$

En 0^+ et 0^- :

```
(%i24) limit(1/x, x, 0, plus);
```

(%o24) ∞

```
(%i25) limit(1/x, x, 0, minus);
```

(%o25) $-\infty$

En moins l'infini

```
(%i26) limit(f(x),x,minf);
       limit(f(x),x,-inf);
```

(%o26) ∞

(%o27) ∞

En une valeur

```
(%i28) limit(f(x),x,2);
```

(%o28) 1081344

Limite indéfinie

```
(%i29) limit(sin(x), x, inf);
```

(%o29) ind

5.6 Fonctions affines

```
(%i30) kill(a)$
       kill(b)$
```

```
(%i32) h(x):=a*x+b;
```

(%o32) $\mathrm{h}\left(x\right) := a\,x + b$

Zéro d'une fonction affine

```
(%i33) solve(h(x)=0,x);
```

(%o33) $[x = -\frac{b}{a}]$

Dérivée d'une fonction affine

```
(%i34) diff(h(x),x);
```

(%o34) a

Limites

```
(%i35) limit(f(x),x,-inf);
       limit(f(x),x,x[0]);
       limit(f(x),x,inf);
```

(%o35) ∞

(%o36) $x_0^{20} + x_0^{15}$

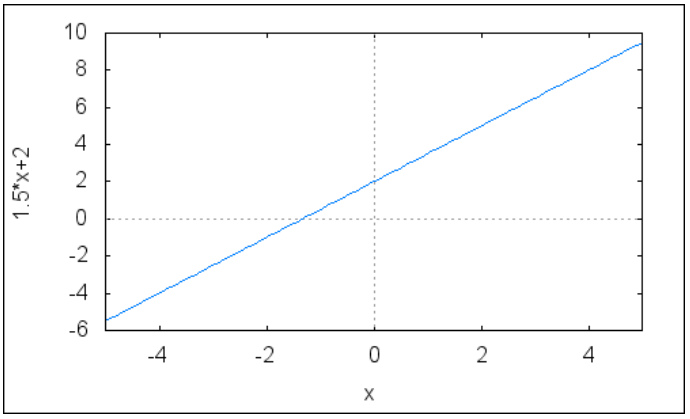
(%o37) ∞

Courbe d'une fonction affine : droite

```
(%i38) a:1.5$
       b:2$
       h(x);
```

(%o40) $1.5\,x + 2$

```
(%i41) wxplot2d([h(x)], [x,-5,5])$
```

(%t41) 

5.7 Fonctions du second degré

```
(%i42) kill(a)$
       kill(b)$
       kill(c)$
```

```
(%i45) f(x):=a*x^2+b*x+c;
```

(%o45) $\mathrm{f}\left(x\right) := a\,x^2 + b\,x + c$

Zéros d'un polynôme du second degré

```
(%i46) solutions:solve(f(x)=0,x);
```

(%o46) $[x = -\frac{\sqrt{b^2 - 4\,a\,c} + b}{2\,a}, x = \frac{\sqrt{b^2 - 4\,a\,c} - b}{2\,a}]$

Attention cette expression est valable pour toutes les valeurs réelles de Δ. Si Delta est négatif le nombre %i apparaît, voir les exemples suivants. Dans le cas où Delta = b^2-4*a*c est positif :

```
(%i47) subst(1, c,subst(4, b, subst(1, a, solutions)));
```

(%o47) $[x = -\frac{2\,\sqrt{3} + 4}{2}, x = \frac{2\,\sqrt{3} - 4}{2}]$

Dans le cas où Delta = b^2-4*a*c est nul :

```
(%i48) subst(1, c,subst(2, b, subst(1, a, solutions)));
```

(%o48) $[x = -1, x = -1]$

Dans le cas où Delta = b^2-4*a*c est négatif :

```
(%i49) subst(4, c,subst(2, b, subst(1, a, solutions)));
```

(%o49) $[x = -\frac{2\,\sqrt{3}\,i + 2}{2}, x = \frac{2\,\sqrt{3}\,i - 2}{2}]$

Dérivée d'une fonction polynôme

```
(%i50) define(df(x),diff(f(x),x));
```

(%o50) $\mathrm{df}\left(x\right) := 2\,a\,x + b$

Zéro de la dérivée

```
(%i51) solve(df(x)=0,x);
```

(%o51) $[x = -\frac{b}{2\,a}]$

```
(%i52) a:1.5$
       b:2$
       c:-1$
       f(x);
```

(%o55) $1.5\,x^2 + 2\,x - 1$

Limites

```
(%i56) limit(f(x),x,-inf);
       limit(f(x),x,x[0]);
       limit(f(x),x,inf);
```

(%o56) ∞

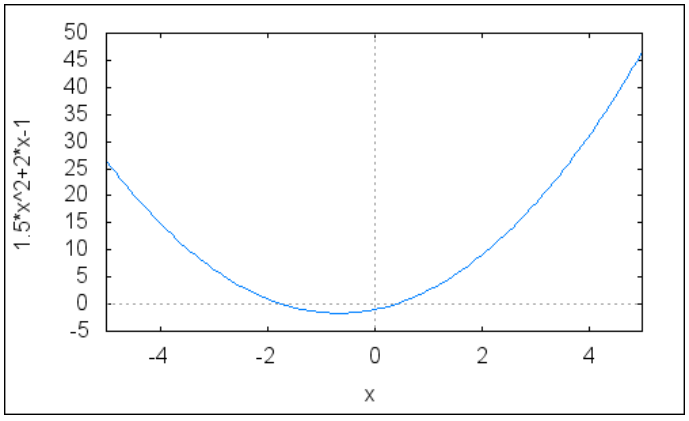
(%o57) $1.5\,x_0^2 + 2\,x_0 - 1$

(%o58) ∞

Courbe d'une fonction du second degré

```
(%i59) wxplot2d([f(x)], [x,-5,5])$
```

(%t59)



5.8 Fonction exponentielle

Définition

```
(%i60) define(f(x),exp(x));
```

(%o60) $\mathrm{f}\,(x) := e^x$

Valeurs particulières

```
(%i61) exp(0);
```

(%o61) 1

```
(%i62) exp(1);
```

(%o62) e

Dérivée

```
(%i63) define(df(x),diff(f(x),x));
```

(%o63) $\mathrm{df}\,(x) := e^x$

Limites

```
(%i64) limit(f(x),x,-inf);
       limit(f(x),x,x[0]);
       limit(f(x),x,inf);
```

(%o64) 0

(%o65) e^{x_0}

(%o66) ∞

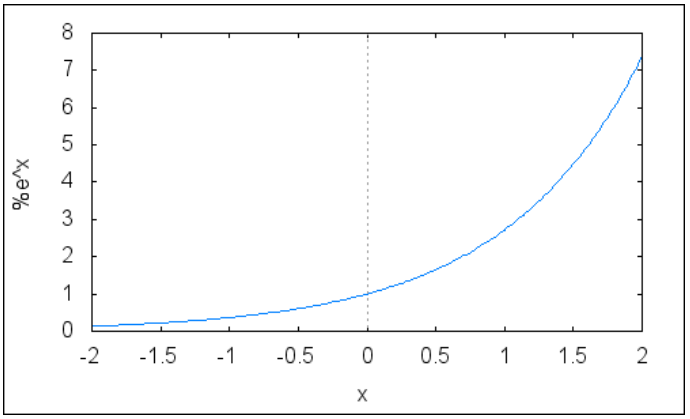
```
(%i67) limit(exp(x)/x,x,inf);
```

(%o67) ∞

Courbe

```
(%i68) wxplot2d([f(x)], [x,-2,2])$
```

(%t68)

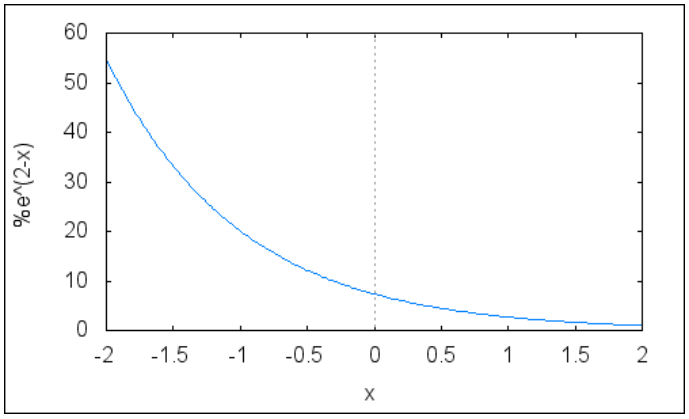


Composée avec une fonction affine

```
(%i69) kill(a)$
       kill(b)$
       define(f(x),exp(a*x+b));
       define(df(x),diff(f(x),x));
       a:-1$ b:2$
       wxplot2d([f(x)], [x,-2,2])$
```

(%o71) $\mathrm{f}\,(x) := e^{a\,x+b}$

(%o72) $\mathrm{df}\,(x) := a\,e^{a\,x+b}$



(%t75)

Relations fonctionnelles

```
(%i76) kill(a)$
       kill(b)$
```

```
(%i78) exp(a)*exp(b);
```

(%o78) e^{b+a}

```
(%i79) 1/exp(a);
```

(%o79) e^{-a}

```
(%i80) exp(a)/exp(b);
```

(%o80) e^{a-b}

```
(%i81) (exp(a))^b;
```

(%o81) $e^{a\,b}$

```
(%i82) sqrt(exp(a));
```

(%o82) $e^{\frac{a}{2}}$

5.9 Fonction logarithme népérien

Definition

```
(%i83) define(f(x),log(x));
```

(%o83) $\mathrm{f}\,(x) := \log\,(x)$

Valeurs particulières

```
(%i84) log(1);
       log(1.0);
```

(%o84) 0

(%o85) 0.0

```
(%i86) log(%e);
       log(2.718);
```

(%o86) 1

(%o87) 0.99989631572895

Dérivée

```
(%i88) define(df(x),diff(f(x),x));
```

(%o88) $\mathrm{df}\,(x) := \frac{1}{x}$

Limites

```
(%i89) limit(f(x),x,-inf);
       limit(f(x),x,3);
       limit(f(x),x,inf);
```

(%o89) $infinity$

(%o90) $\log\,(3)$

(%o91) ∞

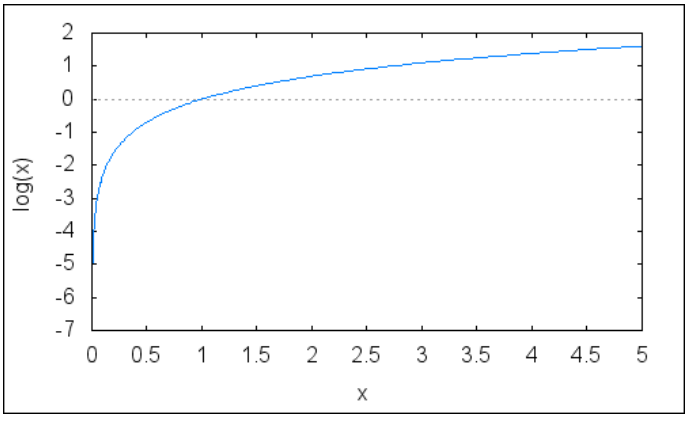
```
(%i92) limit(log(x)/x,x,inf);
```

(%o92) 0

Courbe

```
(%i93) wxplot2d([f(x)], [x,0.001,5])$
```

(%t93)

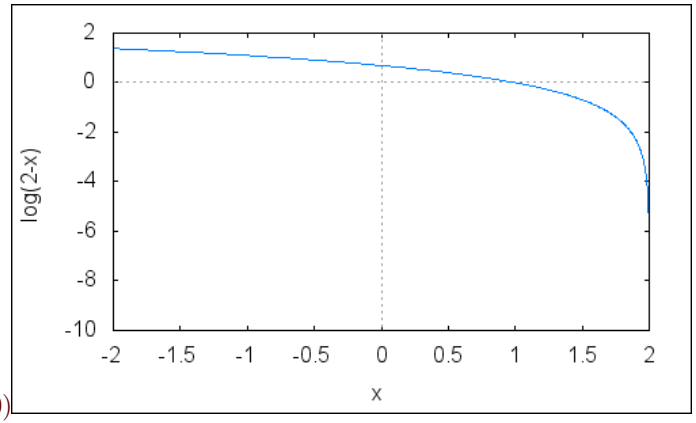


Composée avec une fonction affine

```
(%i94) kill(a)$
       kill(b)$
       define(f(x),log(a*x+b));
       define(df(x),diff(f(x),x));
       a:-1$ b:2$
       wxplot2d([f(x)], [x,-2,1.9999])$
```

(%o96) $\mathrm{f}\,(x) := \log\,(a\,x + b)$

(%o97) $\mathrm{df}\,(x) := \frac{a}{a\,x + b}$



(%t100)

Logarithme et exponentielle

```
(%i101)log(exp(x));
```

(%o101) x

```
(%i102)exp(log(x));
```

(%o102) x

Relations fonctionnelles

```
(%i103)kill(a)$
       kill(b)$
```

```
(%i105)logcontract(log(a)+log(b));
```

(%o105) $\log(a\,b)$

```
(%i106)logcontract(-log(a));
```

(%o106) $-\log(a)$

```
(%i107)logcontract(log(a)-log(b));
```

(%o107) $\log\left(\frac{a}{b}\right)$

```
(%i108)log(a^b);
```

(%o108) $\log(a)\ b$

```
(%i109)log(sqrt(a));
```

(%o109) $\frac{\log(a)}{2}$

5.10 Fonctions trigonométriques

Définition

```
(%i110)define(f(x),cos(x));
       define(g(x),sin(x));
```

(%o110) $\mathrm{f}(x) := \cos(x)$

(%o111) $g(x) := \sin(x)$

Valeurs particulières entre 0 **et** $2 * \pi$

```
(%i112) valeursdex:sort(append(makelist(k*%pi/6,k,0, 12),[%pi/4, 3*%pi/4, 5*%pi/4, 7*%pi/4]));
        valeursdecosx:map(f,valeursdex);
        valeursdesinx:map(g,valeursdex);
```

(%o112) $[0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{3\pi}{4}, \frac{5\pi}{6}, \pi, \frac{7\pi}{6}, \frac{5\pi}{4}, \frac{4\pi}{3}, \frac{3\pi}{2}, \frac{5\pi}{3}, \frac{7\pi}{4}, \frac{11\pi}{6}, 2\pi]$

(%o113) $[1, \frac{\sqrt{3}}{2}, \frac{1}{\sqrt{2}}, \frac{1}{2}, 0, -\frac{1}{2}, -\frac{1}{\sqrt{2}}, -\frac{\sqrt{3}}{2}, -1, -\frac{\sqrt{3}}{2}, -\frac{1}{\sqrt{2}}, -\frac{1}{2}, 0, \frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{\sqrt{3}}{2}, 1]$

(%o114) $[0, \frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{\sqrt{3}}{2}, 1, \frac{\sqrt{3}}{2}, \frac{1}{\sqrt{2}}, \frac{1}{2}, 0, -\frac{1}{2}, -\frac{1}{\sqrt{2}}, -\frac{\sqrt{3}}{2}, -1, -\frac{\sqrt{3}}{2}, -\frac{1}{\sqrt{2}}, -\frac{1}{2}, 0]$

Dérivées

```
(%i115) define(df(x),diff(f(x),x));
        define(dg(x),diff(g(x),x));
```

(%o115) $df(x) := -\sin(x)$

(%o116) $dg(x) := \cos(x)$

Limites

L'affichage de ind veut dire indéfinie, la limite n'existe pas !

```
(%i117) limit(f(x),x,-inf);
        limit(f(x),x,3);
        limit(f(x),x,inf);
        limit(g(x),x,-inf);
        limit(g(x),x,3);
        limit(g(x),x,inf);
```

(%o117) *ind*

(%o118) $\cos(3)$

(%o119) *ind*

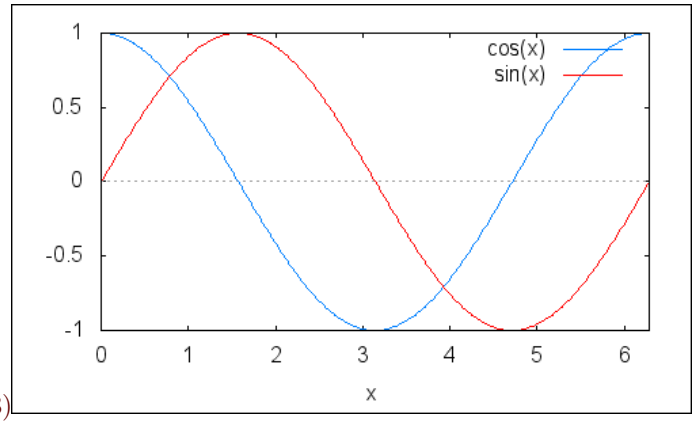
(%o120) *ind*

(%o121) $\sin(3)$

(%o122) *ind*

Courbes

```
(%i123) wxplot2d([f(x),g(x)], [x,0,2*%pi])$
```



(%t123)

Composée avec une fonction affine

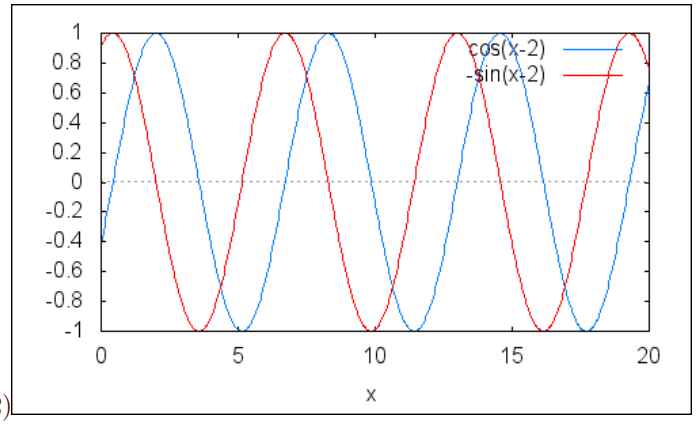
```
(%i124)kill(a)$
       kill(b)$
       define(f(x),cos(a*x+b));
       define(df(x),diff(f(x),x));
       define(g(x),sin(a*x+b));
       define(dg(x),diff(f(x),x));
       a:-1$ b:2$
       wxplot2d([f(x),g(x)], [x,0,20])$
```

(%o126) $\mathrm{f}\left(x\right) := \cos\left(a\,x+b\right)$

(%o127) $\mathrm{df}\left(x\right) := -a\sin\left(a\,x+b\right)$

(%o128) $\mathrm{g}\left(x\right) := \sin\left(a\,x+b\right)$

(%o129) $\mathrm{dg}\left(x\right) := -a\sin\left(a\,x+b\right)$



(%t132)

Relations

```
(%i133)kill(a)$
       kill(b)$

(%i135)trigexpand(cos(a+b));
```

(%o135) $\cos\left(a\right)\,\cos\left(b\right)-\sin\left(a\right)\,\sin\left(b\right)$

```
(%i136)trigexpand(cos(a-b));
```

(%o136) $\sin\left(a\right)\,\sin\left(b\right)+\cos\left(a\right)\,\cos\left(b\right)$

```
(%i137)trigexpand(sin(a+b));
```

(%o137) $\cos(a)\,\sin(b)+\sin(a)\,\cos(b)$

```
(%i138)trigexpand(sin(a-b));
```

(%o138) $\sin(a)\,\cos(b)-\cos(a)\,\sin(b)$

5.11 Primitives et intégrales d'une fonction

5.11.1 Primitives

```
(%i139)kill(all)$
```

```
(%i1)  integrate(a*x+b, x);
```

(%o1) $\frac{a\,x^2}{2}+b\,x$

```
(%i2)  integrate(a*x^2+b*x+c,x);
```

(%o2) $\frac{a\,x^3}{3}+\frac{b\,x^2}{2}+c\,x$

```
(%i3)  integrate(exp(a*x+b),x);
```

(%o3) $\frac{e^{a\,x+b}}{a}$

```
(%i4)  integrate(log(a*x+b),x);
```

(%o4) $\frac{(a\,x+b)\,\log(a\,x+b)-a\,x-b}{a}$

```
(%i5)  integrate(cos(a*x+b),x);
```

(%o5) $\frac{\sin(a\,x+b)}{a}$

```
(%i6)  integrate(sin(a*x+b),x);
```

(%o6) $-\frac{\cos(a\,x+b)}{a}$

5.11.2 Intégrales

```
(%i7)  integrate(2*x+3, x, 0, 1);
```

(%o7) 4

```
(%i8)  integrate(2*x^2+4*x-5,x, 1, 4);
```

(%o8) 57

```
(%i9)  integrate(exp(-x+3),x, 3, 5);
```

(%o9) $1-e^{-2}$

```
(%i10) integrate(log(x-5),x, 6, 7);
```

(%o10) $2\log(2) - 1$

```
(%i11) integrate(cos(2*x),x, 0, %pi/3);
```

(%o11) $\frac{\sqrt{3}}{4}$

```
(%i12) integrate(sin(2*x),x, 0, %pi/3);
```

(%o12) $\frac{3}{4}$

5.12 Développements limités

On note N l'ordre des développements que l'on veut étudier.

```
(%i13) N:4;
```

(%o13) 4

La fonction exponentielle :

```
(%i14) taylor(exp(x), x, 0, N);
```

(%o14)/T/ $x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots$

La fonction inverse :

```
(%i15) taylor(1/(1+x), x, 0, N);
```

(%o15)/T/ $x + x^2 - x^3 + x^4 + \ldots$

La fonction logarithme népérien :

```
(%i16) taylor(log(1+x), x, 0, N);
```

(%o16)/T/ $\frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$

La fonction sinus :

```
(%i17) taylor(sin(x), x, 0, N);
```

(%o17)/T/ $\frac{x^3}{6} + \ldots$

La fonction cosinus :

```
(%i18) taylor(cos(x), x, 0, N);
```

(%o18)/T/ $\frac{x^2}{2} + \frac{x^4}{24} + \ldots$

La fonction puissance :

```
(%i19) taylor((1+x)^alpha, x, 0, N);
```

(%o19)/T/ $\alpha\, x + \frac{\left(\alpha^2 - \alpha\right)\, x^2}{2} + \frac{\left(\alpha^3 - 3\,\alpha^2 + 2\,\alpha\right)\, x^3}{6} + \frac{\left(\alpha^4 - 6\,\alpha^3 + 11\,\alpha^2 - 6\,\alpha\right)\, x^4}{24} + \ldots$

Chapitre 6

Les suites

6.1 Définition d'une suite

Suites définies explicitement, comme une fonction de n :

```
(%i1)  a(n):=n^2;
```

(%o1) $\mathrm{a}\,(n) := n^2$

ou comme une liste infinie :

```
(%i2)  a[n]:=n^2;
```

(%o2) $a_n := n^2$

Suites définies par une relation de récurrence, comme une fonction de n :

```
(%i3)  a(n):= if n = 0 then 4 else 2*a(n-1)-3$
       a(110);
```

(%o4) 1298074214633706907132624082305027

ou comme une liste infinie :

```
(%i5)  a[n]:= if n = 0 then 4 else 2*a[n-1]-3$
       a[110];
```

(%o6) 1298074214633706907132624082305027

6.2 Expression explicite d'une suite définie par récurrence

On charge le module « solve_rec » :

```
(%i7)  load("solve_rec");
```

(%o7) $C : /PROGRA\,2/MAXIMA\,1.0{-}2/share/maxima/5.28.0{-}2/share/solve\_rec/solve\_rec.mac$

```
(%i8)  solve_rec(u[n]=2*u[n-1]-3, u[n]);
```

(%o8) $u_n = \%k_1\,2^n - 3\,2^n + 3$

Avec un terme initial :

```
(%i9)  solve_rec(u[n]=2*u[n-1]-3, u[n], u[0]=4);
```

(%o9) $u_n = 2^n + 3$

On peut définir une suite de cette façon :

```
(%i10) define(v[n], rhs(%o9));
```

(%o10) $v_n := 2^n + 3$

```
(%i11) v[9];
```

(%o11) 515

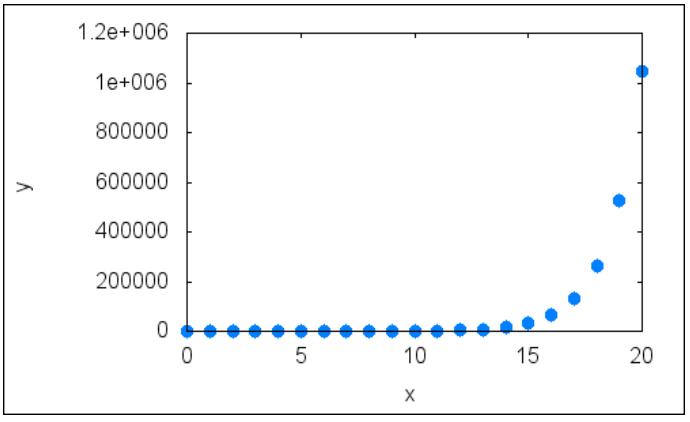
6.3 Représentation géométrique d'une suite

```
(%i12) entiers:makelist(n,n,0, 20);
       termesdelasuite:map(a,entiers);
       wxplot2d([discrete, entiers,termesdelasuite],[style, points]);
```

(%o12) $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$

(%o13) $[4, 5, 7, 11, 19, 35, 67, 131, 259, 515, 1027, 2051, 4099, 8195, 16387, 32771, 65539, 131075, 262147, 524291, 1048579]$

(%t14)



(%o14)

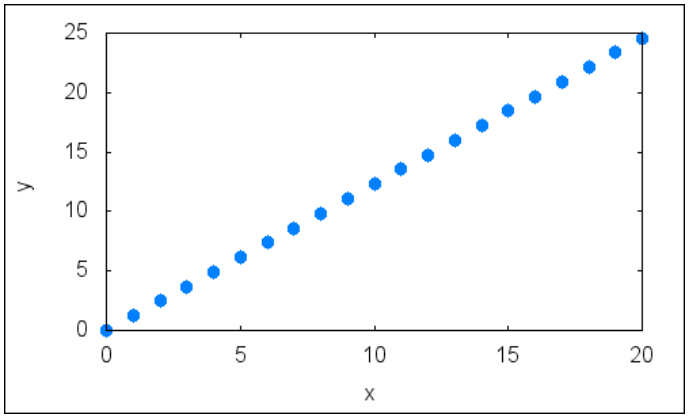
6.4 Suites arithmétiques

```
(%i15) r:1.23$
       a(n):= if n < 1 then 0 else a(n-1)+r$
       a(110);
```

(%o17) 135.3000000000001

```
(%i18) entiers:makelist(n,n,0, 20)$
       termesdelasuite:map(a,entiers)$
       wxplot2d([discrete, entiers,termesdelasuite],[style, points])$
```

(%t20)



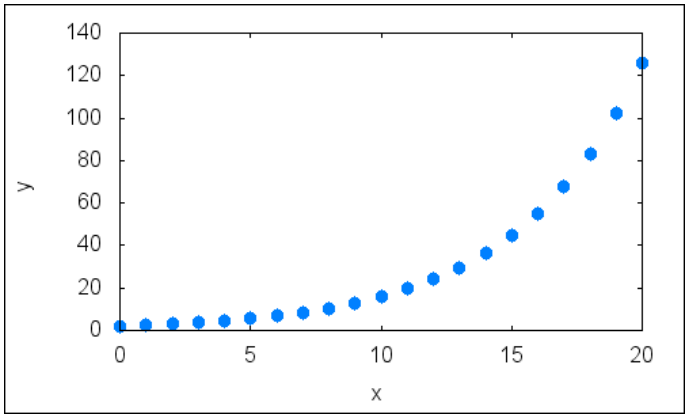
6.5 Suites géométriques

```
(%i21) q:1.23$
       a(n):= if n < 1 then 2 else a(n-1)*q$
       a(10);
```

(%o23) 15.85189219210378

```
(%i24) entiers:makelist(n,n,0, 20)$
       termesdelasuite:map(a,entiers)$
       wxplot2d([discrete, entiers,termesdelasuite],[style, points])$
```

(%t26)



6.6 Limite d'une suite

```
(%i27) a(n):=n^2;
       limit(a(n),n,inf);
```

(%o27) $\mathrm{a}\left(n\right):=n^2$

(%o28) ∞

```
(%i29) a(n):=1/n^2;
       limit(a(n),n,inf);
```

(%o29) $\mathrm{a}\left(n\right):=\frac{1}{n^2}$

(%o30) 0

6.7 Somme d'une suite

On utilise la commande sum :

```
(%i31) sum(a(k), k, 1, 10);
```

(%o31) $\frac{1968329}{1270080}$

6.8 Le module functs

```
(%i32) load("functs");
```

(%o32) $C : /PROGRA\ 2/MAXIMA\ 1.0{-}2/share/maxima/5.28.0{-}2/share/simplification/functs.mac$

Suites arithmétiques arithmetic(a, r, n) retourne le n-ième terme de la suite arithmétique de raison r et de premier terme a :

```
(%i33) arithmetic(1,2,3);
```

(%o33) 5

geometric(a, q, n) retourne le n-ième terme de la suite arithmétique de raison r et de premier terme a :

```
(%i34) geometric(1,2,3);
```

(%o34) 4

Somme des termes des suites arithmétiques et géométriques :

```
(%i35) arithsum(1,2,3);
```

(%o35) 9

```
(%i36) geosum(1,2,3);
```

(%o36) 7

Chapitre 7

Les nombres complexes

7.1 Forme algébrique d'un nombre complexe

Définition d'un nombre complexe

```
(%i1)  z1:2+3*%i;
```

(%o1) $3\,i+2$

```
(%i2)  z2:5+3*%i;
```

(%o2) $3\,i+5$

Somme et soustraction de deux nombres complexes

```
(%i3)  z1+z2;
       z1-z2;
```

(%o3) $6\,i+7$

(%o4) -3

Produit de deux nombres complexes

Il faut développer l'expression avec expand si l'on souhaite avoir la forme algébrique :

```
(%i5)  z1*z2;
       expand(z1*z2);
```

(%o5) $(3\,i+2)\,(3\,i+5)$

(%o6) $21\,i+1$

Pour avoir la forme algébrique d'un nombre complexe, on utilise la fonction rectform :

```
(%i7)  z1/z2;
       rectform(z1/z2);
```

(%o7) $$\frac{3\,i+2}{3\,i+5}$$

(%o8) $$\frac{9\,i}{34}+\frac{19}{34}$$

Une nouvelle identité

```
(%i9)  declare(a,mainvar)$
       expand((a-b*%i)*(a+b*%i));
```

(%o10) $a^2 + b^2$

Partie réelle

```
(%i11) realpart(z1);
```

(%o11) 2

Partie imaginaire

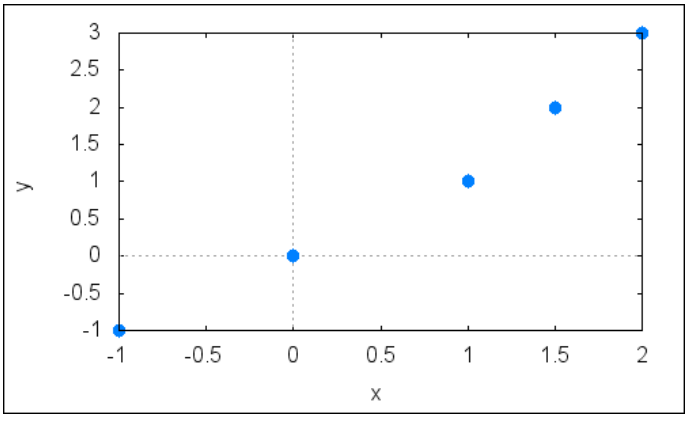
```
(%i12) imagpart(z1);
```

(%o12) 3

7.2 Représentation géométrique d'un nombre complexe

Définition d'une fonction utilisée pour placer les images des nombres complexes dans le plan complexe :

```
(%i13) plotcomplex(Liste):=wxplot2d([discrete, makelist(realpart(Liste[k]), k, 1, length(Liste)),
       makelist(imagpart(Liste[k]), k, 1, length(Liste))],[style, points])$

(%i14) plotcomplex([1+%i, 2+3*%i, -1-%i, 1.5+2*%i, 0]);
```



(%t14)

(%o14)

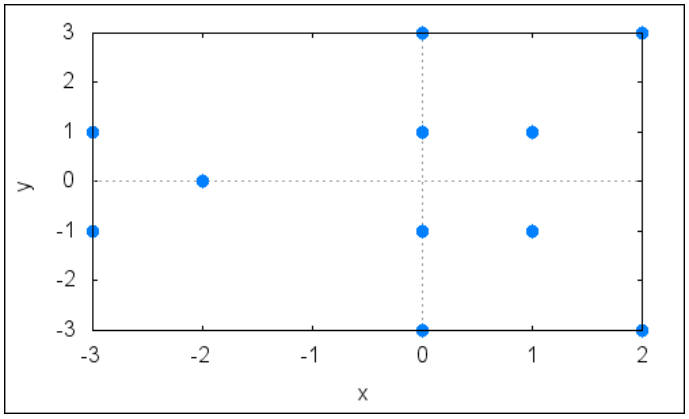
7.3 Nombre complexe conjugué

```
(%i15) conjugate(1+%i);
```

(%o15) $1 - i$

Représentation géométrique d'un conjugué

```
(%i16) liste:[1+%i, 2+3*%i, %i, -3+%i, 3*%i, -2]$
       liste:append(liste, makelist(conjugate(liste[k]), k, 1, length(liste)))$
       plotcomplex(liste);
```



(%t18)

(%o18)

```
(%i19) conjugate[liste];
```

(%o19) $conjugate_{[i+1,3\,i+2,i,i-3,3\,i,-2,1-i,2-3\,i,-i,-i-3,-3\,i,-2]}$

7.4 Forme trigonométrique d'un nombre complexe

Définition d'un nombre complexe

```
(%i20) z:1+%i;
```

(%o20) $i+1$

Le module

```
(%i21) cabs(z);
```

(%o21) $\sqrt{2}$

Un argument

```
(%i22) carg(z);
```

(%o22) $\frac{\pi}{4}$

Définition de la forme trigonométrique

```
(%i23) formetrigo(z):=[cabs(z), carg(z)]$
```

```
(%i24) formetrigo(z);
```

(%o24) $[\sqrt{2}, \frac{\pi}{4}]$

```
(%i25) formetrigo(1+%i*sqrt(3));
```

(%o25) $[2, \frac{\pi}{3}]$

7.5 Forme exponentielle

```
(%i26) polarform(z);
```

(%o26) $\sqrt{2}\,e^{\frac{i\,\pi}{4}}$

```
(%i27) rectform(%);
```

(%o27) $i+1$

```
(%i28) z:exp(%i*theta);
```

(%o28) $e^{i\,\theta}$

Définition de $e^{i\theta}$

```
(%i29) rectform(z);
```

(%o29) $i\sin(\theta)+\cos(\theta)$

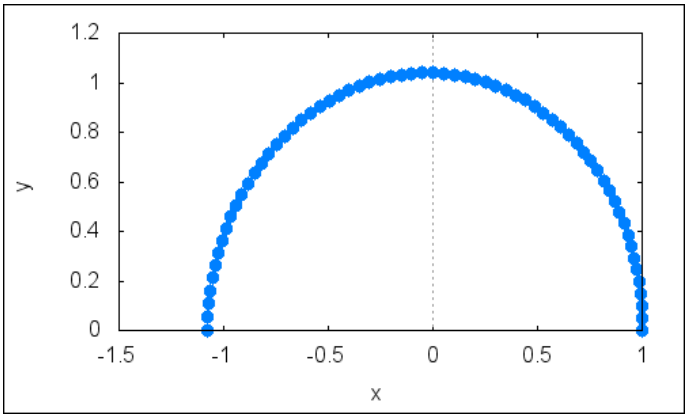
Cas particulier : $e^{i\pi}+1=0$

```
(%i30) exp(%i*%pi);
```

(%o30) -1

Illustration de l'égalité précédente et $\lim_{n\to+\infty}\left(1+\frac{z}{n}\right)^n=e^z$

```
(%i31) N:64$
       liste:makelist((1+%i*%pi/N)^k, k, 0, N)$
       plotcomplex(liste);
```



(%t33)

(%o33)

Propriétés de la forme exponentielle

```
(%i34) exp(%i*theta[1])*exp(%i*theta[2]);
```

(%o34) $e^{i\,\theta_2+i\,\theta_1}$

```
(%i35) 1/exp(%i*theta);
```

(%o35) $e^{-i\,\theta}$

```
(%i36) exp(%i*theta[1])/exp(%i*theta[2]);
```

(%o36) $e^{i\,\theta_1 - i\,\theta_2}$

```
(%i37) exp(%i*theta)^n;
```

(%o37) $e^{i\,n\,\theta}$

Formule de Moivre : $(i\sin(\theta)+\cos(\theta))^n = i\sin(n\theta)+cos(n\theta)$

```
(%i38) (cos(theta)+%i*sin(theta))^n;
```

(%o38) $(i\sin(\theta)+\cos(\theta))^n$

```
(%i39) trigrat((exp(%i*theta))^n);
```

(%o39) $i\sin(n\,\theta)+\cos(n\,\theta)$

En une seule commande

```
(%i40) trigrat((cos(theta)+%i*sin(theta))^n);
```

(%o40) $i\sin(n\,\theta)+\cos(n\,\theta)$

Formules d'Euler

```
(%i41) exponentialize(cos(theta));
```

(%o41) $$\frac{e^{i\,\theta}+e^{-i\,\theta}}{2}$$

```
(%i42) exponentialize(sin(theta));
```

(%o42) $$-\frac{i\,\left(e^{i\,\theta}-e^{-i\,\theta}\right)}{2}$$

7.6 Équations du second degré à coefficients constants

Maxima ne fait pas la distinction entre delta positif ou négatif :

```
(%i43) kill(z)$
```

```
(%i44) solutions:solve(a*z^2+b*z+c=0, z);
```

(%o44) $$[z=-\frac{\sqrt{b^2-4\,c\,a}+b}{2\,a},z=\frac{\sqrt{b^2-4\,c\,a}-b}{2\,a}]$$

```
(%i45) solve(z^2+z-1=0,z);
```

(%o45) $$[z=-\frac{\sqrt{5}+1}{2},z=\frac{\sqrt{5}-1}{2}]$$

```
(%i46) solve(z^2+2*z+1=0,z);
```

(%o46) $[z=-1]$

```
(%i47) solve(z^2+z+1=0,z);
```

(%o47) $$[z=-\frac{\sqrt{3}\,i+1}{2},z=\frac{\sqrt{3}\,i-1}{2}]$$

Pour vérifier qu'un nombre complexe annule une expression :

```
(%i48) subst(-1, z, z^2+z+1);
```

(%o48) 1

Chapitre 8

Listes et statistiques descriptives

8.1 Listes

Définition d'une liste :

```
(%i1)  L:[30, 14, 15, 16, 14, 15, 18, 19, 1 , 6 ,18 ];
```

(%o1) $[30, 14, 15, 16, 14, 15, 18, 19, 1, 6, 18]$

Longueur d'une liste :

```
(%i2)  length(L);
```

(%o2) 11

Concaténer deux listes :

```
(%i3)  append(L,[0, 2, 3]);
```

(%o3) $[30, 14, 15, 16, 14, 15, 18, 19, 1, 6, 18, 0, 2, 3]$

Ordonner la liste :

```
(%i4)  sort(L);
```

(%o4) $[1, 6, 14, 14, 15, 15, 16, 18, 18, 19, 30]$

Créer un tableau de valeur d'une fonction :

```
(%i5)  f(x):=x^2+1;
```

(%o5) $\mathrm{f}\,(x) := x^2 + 1$

On crée une liste de nombres régulièrement espacés, par exemple de 0,1 :

```
(%i6)  nombres:makelist(k/10,k,0,10);
```

(%o6) $[0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1]$

On crée la liste des images par f de ces nombres :

```
(%i7)  fnombres:f(nombres);
```

(%o7) $[1, \frac{101}{100}, \frac{26}{25}, \frac{109}{100}, \frac{29}{25}, \frac{5}{4}, \frac{34}{25}, \frac{149}{100}, \frac{41}{25}, \frac{181}{100}, 2]$

Si on souhaite avoir les valeurs approchées :

```
(%i8)  anombres:float(fnombres);
```

(%o8) [1.0, 1.01, 1.04, 1.09, 1.16, 1.25, 1.36, 1.49, 1.64, 1.81, 2.0]

8.2 Statistiques descriptives

```
(%i9)  load(descriptive);
```

(%o9) $C : /PROGRA\ 2/MAXIMA\ 1.0{-}2/share/maxima/5.28.0{-}2/share/descriptive/descriptive.mac$

Valeur moyenne de L :

```
(%i10) mean(L);
```

(%o10) $\frac{166}{11}$

```
(%i11) float(mean(L));
```

(%o11) 15.09090909090909

Écart-type :

```
(%i12) std(L);
```

(%o12) $\frac{2\sqrt{1482}}{11}$

```
(%i13) float(std(L));
```

(%o13) 6.999409656334603

Minimum de L :

```
(%i14) mini(L);
```

(%o14) 1

Macimum de L :

```
(%i15) maxi(L);
```

(%o15) 30

Étendue :

```
(%i16) range(L);
```

(%o16) 29

Médiane :

```
(%i17) median(L);
```

(%o17) 15

Quantiles :

```
(%i18) quantile(L,1/4);
```

(%o18) 14

```
(%i19) quantile(L,2/4);
```

(%o19) 15

```
(%i20) quantile(L,3/4);
```

(%o20) 18

```
(%i21) quantile(L,1/10);
```

(%o21) 6

8.3 Listes avec pondérations

Liste des valeurs :

```
(%i22) L:[12,13,15,16];
```

(%o22) $[12, 13, 15, 16]$

Liste des effectifs ou des fréquences :

```
(%i23) F:[1,2,3,4];
```

(%o23) $[1, 2, 3, 4]$

```
(%i24) moyenne(L_1, L_2):=block ( [n:length(L_1)],

                  sum(L_1[k]*L_2[k],k,1,n)/sum(L_2[k],k,1,n)
               )$

(%i25) variance(L_1, L_2):=block ( [n:length(L_1)],

                  moyenne(L_1^2,L_2)-moyenne(L_1, L_2)^2
               )$

(%i26) ecartype(L_1, L_2):=sqrt(variance(L_1, L_2))$

(%i27) moyenne(L,F);
       variance(L,F);
       ecartype(L,F);
```

(%o27) $\frac{147}{10}$

(%o28) $\frac{201}{100}$

(%o29) $\frac{\sqrt{201}}{10}$

Chapitre 9

Probabilités

9.1 Simulations

9.1.1 Simulation du lancer de pièce et loi de Bernoulli

On utilise la commande random(2), pour avoir 0 ou 1 de façon aléatoire et équiprobable :

```
(%i1)  random(2);
       random(2);
       random(2);
```

(%o1) 0

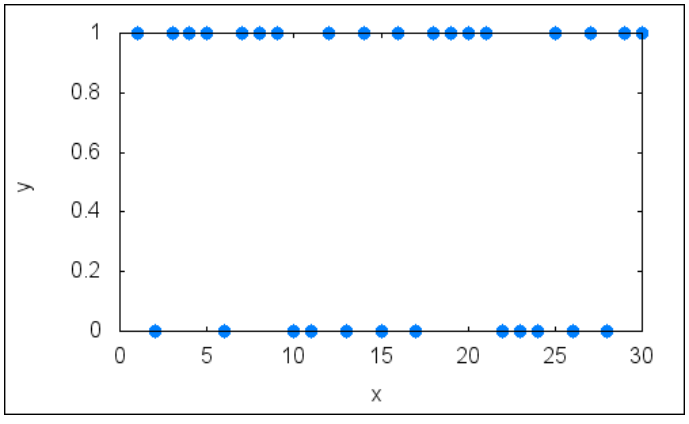
(%o2) 0

(%o3) 0

Liste de simulations du lancer de pièce

```
(%i4)  Liste:[]$
       N:30$
       for i from 1 thru N do Liste:append(Liste,[random(2)])$
       Liste;
       entiers:makelist(n,n,1,N)$
       wxplot2d([discrete, entiers, Liste], [style, points]);
```

(%o7) $[1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1]$



(%t9)

(%o9)

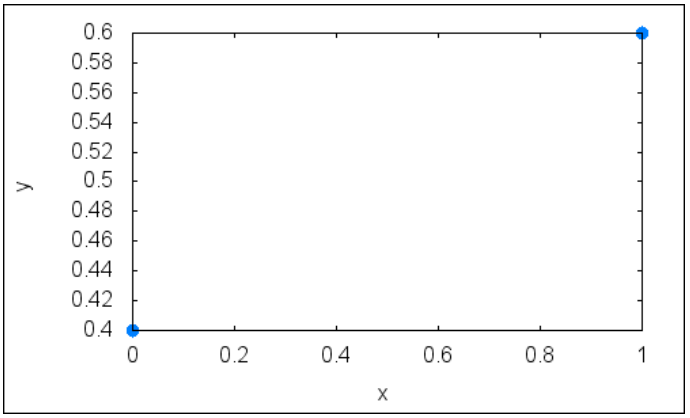
Affichage des fréquences

```
(%i10) zeroUn:[0, 1]$
       f1:sum(Liste[k],k, 1, N)/N$
       f0:1-f1$
       float(f0);
       float(f1);
       ListeFrequences:[f0, f1]$
       wxplot2d([discrete, zeroUn, ListeFrequences], [style, points]);
```

(%o13) 0.4

(%o14) 0.6

(%t16)



(%o16)

La commande random(1.0) permet d'obtenir un nombre de l'intervalle $[0;1[$.

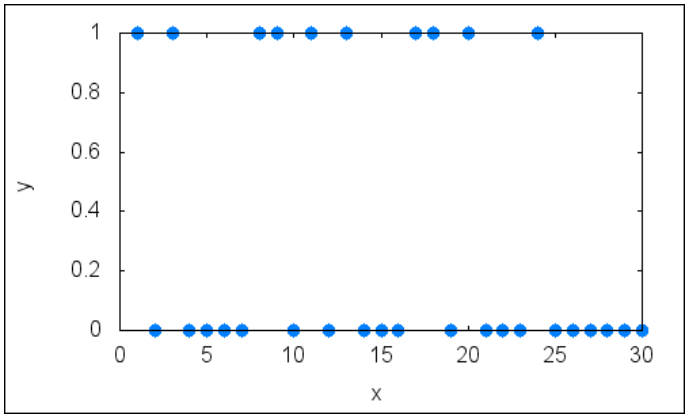
Pour obtenir un lancer de pièce truquée, où la probabilité d'obtenir le 1 est p, on utilise la commande ci-dessous :

```
(%i17) p:0.3$
       random(1.0)+ p;
```

(%o18) 1.152705109085575

```
(%i19) Liste:[]$
       N:30$
       for i from 1 thru N do Liste:append(Liste,[floor(random(1.0)+p)])$
       Liste;
       entiers:makelist(n,n,1,N)$
       wxplot2d([discrete, entiers, Liste], [style, points]);
```

(%o22) $[1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$



(%t24)

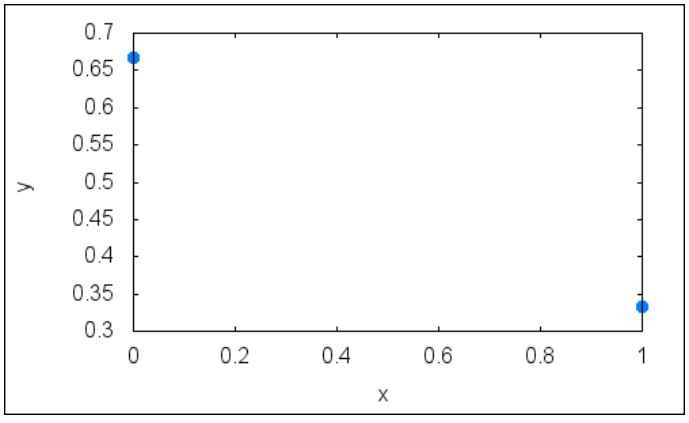
(%o24)

Affichage des fréquences

```
(%i25) zeroUn:[0, 1]$
       f1:sum(Liste[k],k, 1, N)/N$
       f0:1-f1$
       float(f0);
       float(f1);
       ListeFrequences:[f0, f1]$
       wxplot2d([discrete, zeroUn, ListeFrequences], [style, points]);
```

(%o28) 0.66666666666667

(%o29) 0.33333333333333



(%t31)

(%o31)

9.1.2 Simulation du lancer de dé

Simulation du lancer de dé

```
(%i32) random(7);
       random(7);
       random(7);
       random(7);
       random(7);
       random(7);
       random(7);
```

(%o32) 0

(%o33) 3

(%o34) 5

(%o35) 1

(%o36) 0

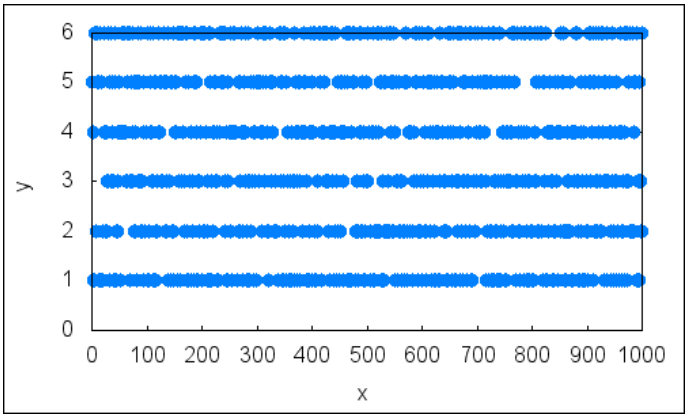
(%o37) 4

(%o38) 3

Liste de simulation du lancer de dé

```
(%i39) Liste:[]$
       N:1000$
       for i from 1 thru N do Liste:append(Liste,[random(6)+1])$
       Liste$
       entiers:makelist(n,n,1,N)$
       wxplot2d([discrete, entiers, Liste], [style, points], [y,0,6])$
```

(%t44)



Transformation d'une liste en un ensemble

```
(%i45) set:setify(Liste);
       for s in set do print (s);
```

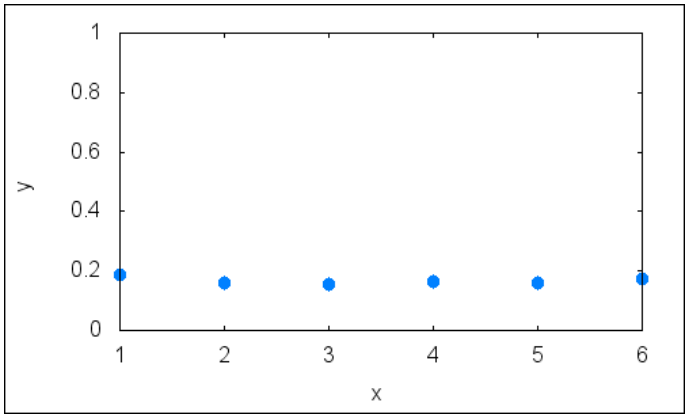
(%o45) $1, 2, 3, 4, 5, 6123456$

(%o46) *done*

Calcul des fréquences

```
(%i47) listeFrequences:[]$
       for element in set do (
           s:0,
           for i step 1 from 1 thru length(Liste) do (
               if element=Liste[i] then s:s+1
           ),
           listeFrequences:append(listeFrequences, [s/length(Liste)])
       )$
       float(listeFrequences);
       entiers:makelist(n,n,1,6)$
       wxplot2d([discrete, entiers, listeFrequences], [style, points], [y,0,1]);
```

(%o49) $[0.185, 0.159, 0.157, 0.165, 0.162, 0.172]$



(%t51)

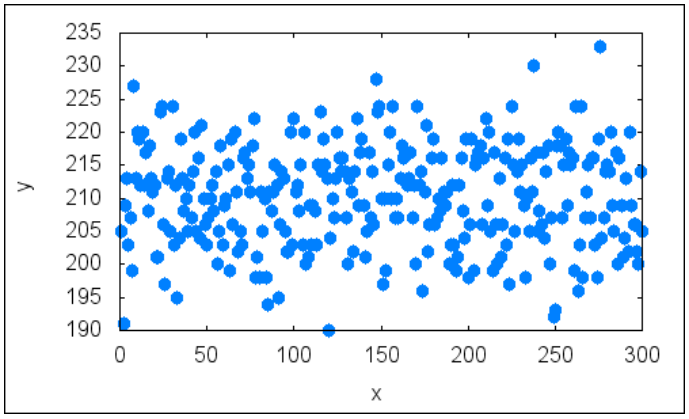
(%o51)

9.1.3 Simulation de la loi binomiale

```
(%i52) nombreSucces(N,p):=block([],
           somme:0,
           for i from 1 thru N do (
               somme:somme+floor(random(1.0)+p)
               ),
           somme
       )$

(%i53) nombreSucces(30,0.7);
```

(%o53) 26

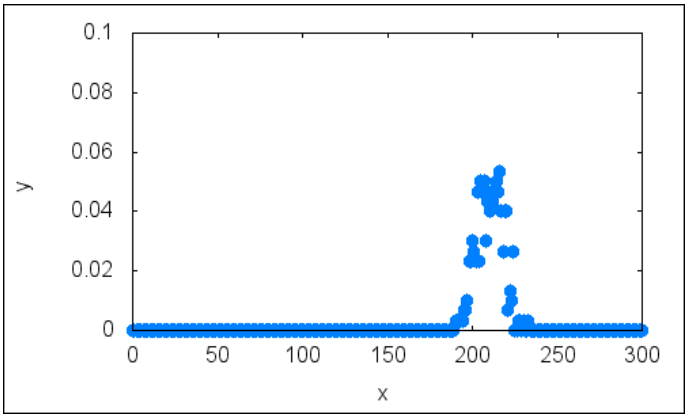
```
(%i54) Liste:[]$
       N:300$
       p:0.7$
       for i from 1 thru N do Liste:append(Liste,[nombreSucces(N,p)])$
       entiers:makelist(n,n,1,N)$
       wxplot2d([discrete, entiers, Liste], [style, points]);
```



(%t59)

(%o59)

```
(%i60) listeFrequences:[]$
       listeValeurs:makelist(i,i,0,N)$
       for element in setify(listeValeurs) do (
           s:0,
           for i step 1 from 1 thru length(Liste) do (
               if element=Liste[i] then s:s+1
           ),
           listeFrequences:append(listeFrequences, [s/length(Liste)])
       )$
       wxplot2d([discrete, listeValeurs, listeFrequences], [style, points],[y,0,0.1]);
```



(%t63)

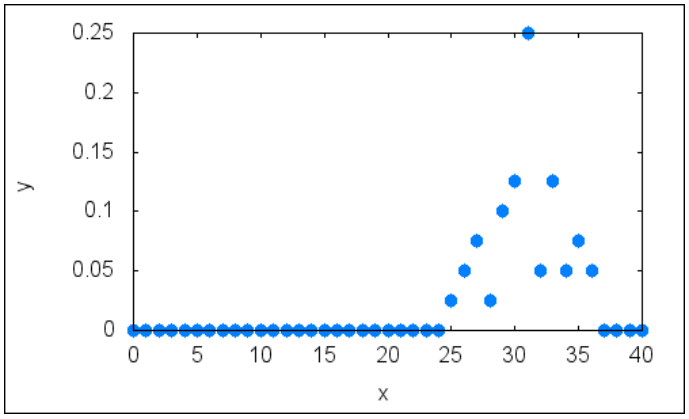
(%o63)

Définition de la fonction simulation de la loi binomiale, elle retourne une liste :

```
(%i64) declare(N, integer)$
       assume(N>0)$
       simulationLoiBinomiale(N,p):=block([],
           Liste:[],
           for i from 1 thru N do Liste:append(Liste,[nombreSucces(N,p)]),
           listeFrequences:[],
           listeValeurs:makelist(i,i,0,N),
           for element in setify(listeValeurs) do (
               s:0,
               for i step 1 from 1 thru length(Liste) do (
                   if element=Liste[i] then s:s+1
                   ),
               listeFrequences:append(listeFrequences, [s/length(Liste)])
               ),
           listeFrequences
       )$
```

Définition de la fonction affichage d'une simulation de la loi binomiale :

```
(%i67) afficheSimulationLoiBinomiale(N,p):=block([],
           listeFrequences:simulationLoiBinomiale(N,p),
           wxplot2d([discrete, makelist(i,i,0,N), listeFrequences], [style, points])
       )$

(%i68) afficheSimulationLoiBinomiale(40,0.8);
```



(%t68)

(%o68)

9.1.4 Simulation de la loi uniforme

Tirage aléatoire d'un nombre de l'intervalle $[0;1[$

```
(%i69) random(1.0);
```

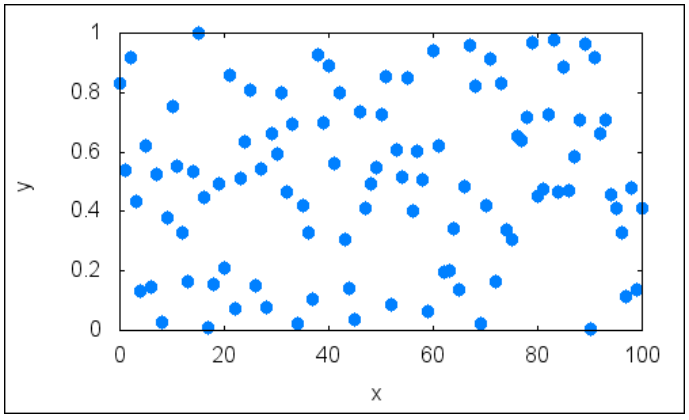
(%o69) 0.95659153069339

Création d'une liste de nombres aléatoires de l'intervalle $[0;1[$

```
(%i70) N:100$
       Liste:[]$
       for i from 0 thru N do Liste:append(Liste,[random(1.0)])$
       Liste$
```

Affichage de la liste précédente

```
(%i74) entiers:makelist(n,n,0,N)$
       wxplot2d([discrete, entiers, Liste], [style, points],[y, 0, 1]);
```



(%t75)

(%o75)

9.2 Distributions

```
(%i76) load(distrib)$
```

9.2.1 Schéma de Bernoulli

À retenir :

pdf : probability density function, c'est la fonction densité de probabilité

cdf : cumulative density function, c'est la fonction de répartition pdf_bernoulli(1,p) donne la valeur de $P(X = 1)$ où X suit la loi de Bernoulli de paramètre p :

```
(%i77) assume(0<p,p<1)$
       pdf_bernoulli(1,p);
       pdf_bernoulli(0,p);
```

(%o78) 0.7

(%o79) 0.3

Représentation graphique

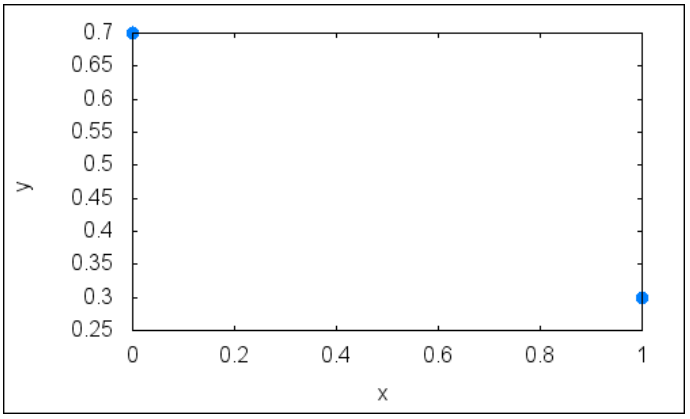
```
(%i80) p:0.3;
       zeroUn:[0, 1]$
       f1:p$
       f0:1-f1$
       float(f0);
       float(f1);
       ListeFrequences:[f0, f1]$
       wxplot2d([discrete, zeroUn, ListeFrequences], [style, points]);
```

(%o80) 0.3

(%o84) 0.7

(%o85) 0.3

(%t87)



(%o87)

9.2.2 Loi binomiale

pdf_binomial(k,n,p) ; donne la valeur de $P(X = k)$ où X suit la loi binomiale de paramètres n et p :

```
(%i88) pdf_binomial(5,7,1/6);
```

(%o88) $\frac{175}{93312}$

cdf_binomial(k,n,p) ; donne la valeur de $P(X \leq k)$ où X suit la loi binomiale de paramètres n et p :

```
(%i89) cdf_binomial(5,7,1/6);
```

(%o89) $\frac{7775}{7776}$

Calcul de $P(a \leq X \leq b)$

```
(%i90) n:20$
       p:0.3$
       a:2$
       b:5$
       cdf_binomial(b,n,p)-cdf_binomial(a-1,n,p);
```

(%o94) 0.40873356967328

Coefficient binomial

```
(%i95) binomial (5, 2);
```

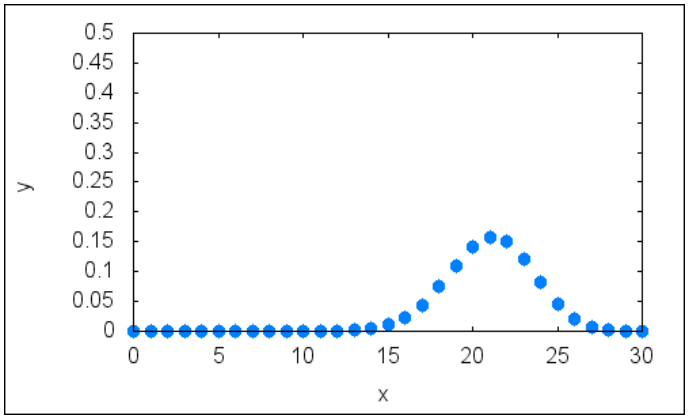
(%o95) 10

Représentation graphique de la loi binomiale

```
(%i96) N:30;
       entiers:makelist(n,n,0,N)$
       listeBinomiale:makelist(pdf_binomial(n,N,0.7),n,0,N)$
       wxplot2d([discrete, entiers, listeBinomiale], [style, points], [y,0,0.5]);
```

(%o96) 30

(%t99)

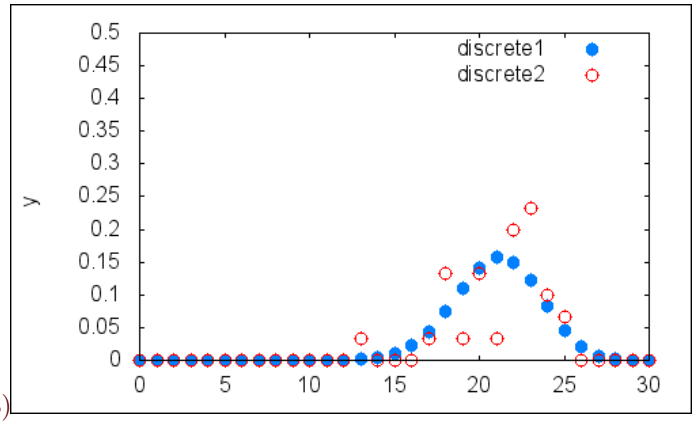


(%o99)

Comparaison avec une simulation

```
(%i100)N:30;
       entiers:makelist(n,n,0,N)$
       listeBinomiale:makelist(pdf_binomial(n,N,0.7),n,0,N)$
       wxplot2d([[discrete, entiers, listeBinomiale], [discrete, entiers, simulationLoiBinomial
```

(%o100)30



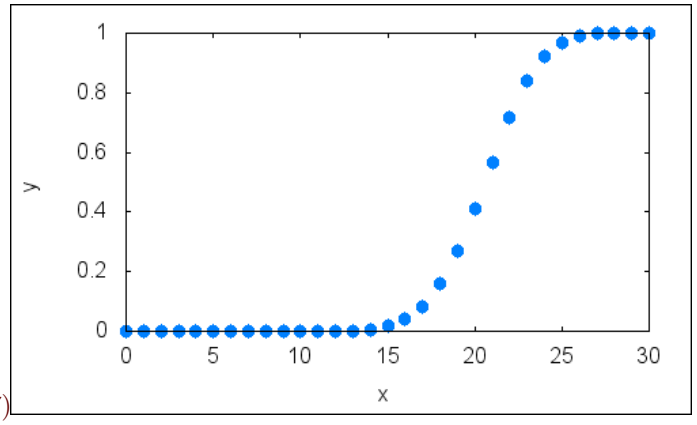
(%t103)

(%o103)

Représentation graphique de la fonction de répartition de loi binomiale

```
(%i104)N:30;
       entiers:makelist(n,n,0,N)$
       listeBinomiale:makelist(cdf_binomial(n,N,0.7),n,0,N)$
       wxplot2d([discrete, entiers, listeBinomiale], [style, points], [y,0,1]);
```

(%o104)30



(%t107)

(%o107)

9.2.3 Loi de Poisson

float(pdf_poisson(k,lambda)) ; donne une valeur approchée de $P(X = k) = \frac{\lambda^k}{k!}e^(-\lambda)$:

```
(%i108)float(pdf_poisson(3,95/100));
```

(%o108)0.055263680830717

float(cdf_poisson(k,lambda)) ; donne une valeur approchée de $P(X \leq k)$:

```
(%i109)float(cdf_poisson(3,95/100));
```

(%o109)0.98392556340084

Calcul de $P(a \leq X \leq b)$

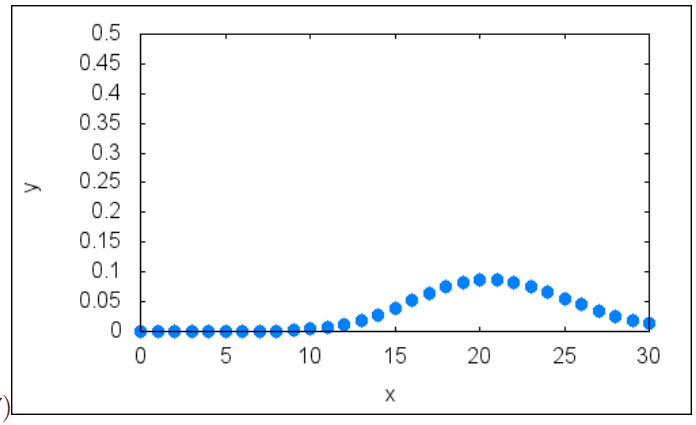
```
(%i110)lambda:7.3$
       a:2$
       b:5$
       cdf_poisson(b,lambda)-cdf_poisson(a-1,lambda);
```

(%o113)0.258436232947

Représentation graphique de la loi de Poisson

```
(%i114)N:30;
       entiers:makelist(n,n,0,N)$
       listePoisson:makelist(pdf_poisson(n,21),n,0,N)$
       wxplot2d([discrete, entiers, listePoisson], [style, points], [y,0,0.5]);
```

(%o114)30



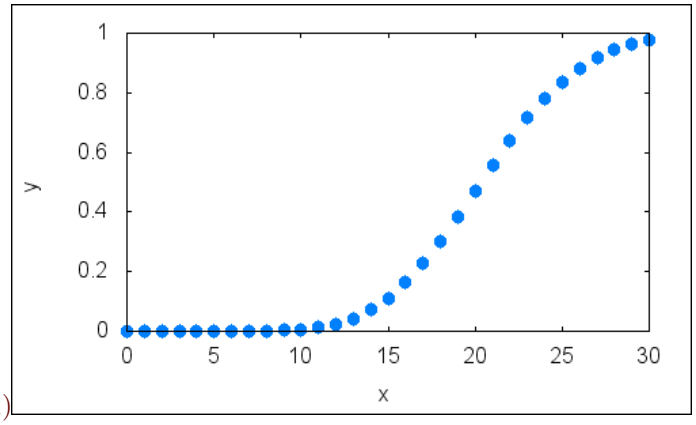
(%t117)

(%o117)

Représentation graphique de la fonction de répartition de loi de Poisson

```
(%i118)N:30;
       entiers:makelist(n,n,0,N)$
       listePoisson:makelist(cdf_poisson(n,21),n,0,N)$
       wxplot2d([discrete, entiers, listePoisson], [style, points], [y,0,1]);
```

(%o118)30



(%t121)

(%o121)

9.2.4 Loi normale

float(pdf_normal(k,m,s)) ; donne une valeur approchée de la fonction densité de la loi normale d'espérance m et d'écart-type s :

```
(%i122)float(pdf_normal(95/100,0,1));
```

(%o122)0.25405905646919

float(cdf_normal(k,m,s)) ; donne une valeur approchée de la fonction répartition de la loi normale d'espérance m et d'écart-type s :

```
(%i123)float(cdf_normal(95/100,0,1));
```

(%o123)0.82894387369152

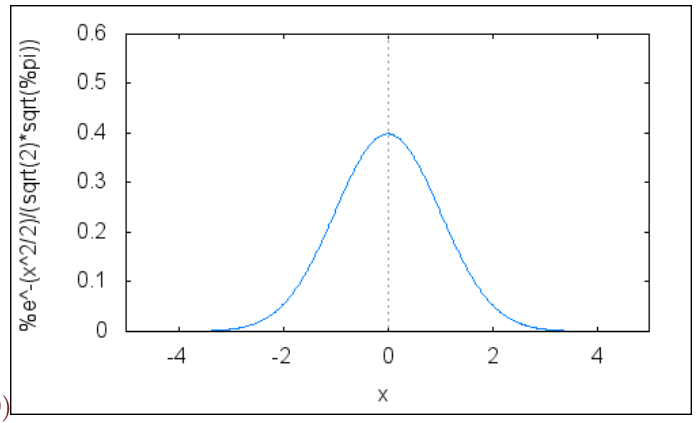
Calcul de $P(a \leq X \leq b)$

```
(%i124)m:0.95$
       s:1.5$
       a:2$
       b:5$
       float(cdf_normal(b,m,s)-cdf_normal(a,m,s));
```

(%o128)0.23849667842003

Représentation graphique de la loi normale Loi normale centrée réduite

```
(%i129)wxplot2d(pdf_normal(x,0,1),[x,-5,5], [y,0,0.6]);
```

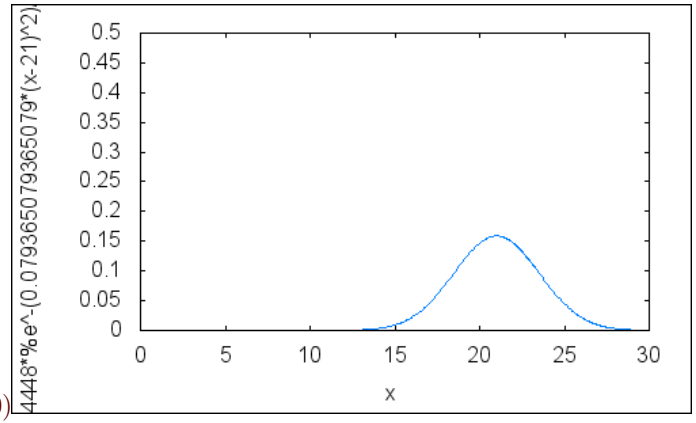


(%t129)

(%o129)

Loi normale de paramètres 21 et 6.3 (variance)

```
(%i130)wxplot2d(pdf_normal(x,21,sqrt(6.3)),[x,0,30], [y,0,0.5]);
```

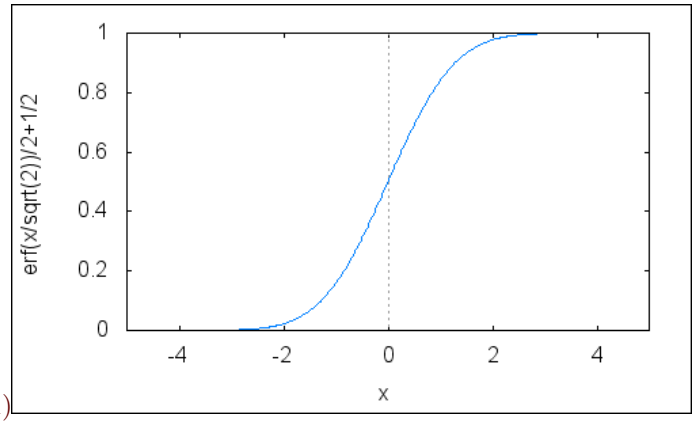


(%t130)

(%o130)

Représentation graphique de la fonction de répartition de la loi normale Fonction de répartition de la loi normale centrée réduite

```
(%i131) wxplot2d(cdf_normal(x,0,1),[x,-5,5], [y,0,1]);
```

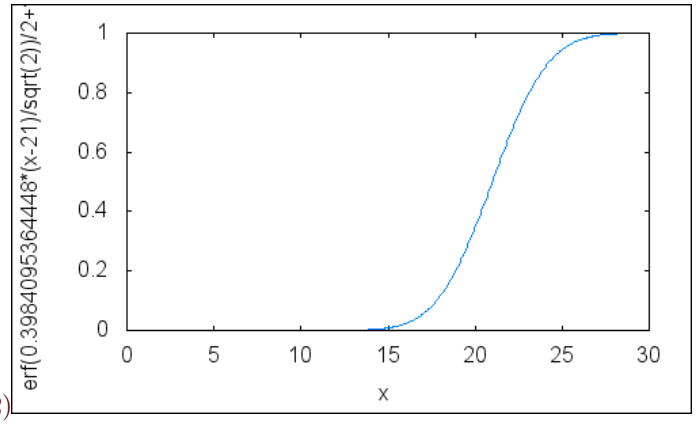


(%t131)

(%o131)

F

```
(%i132) wxplot2d(cdf_normal(x,21,sqrt(6.3)),[x,0,30], [y,0,1]);
```



(%t132)

(%o132)

9.2.5 Approximation d'une loi binomiale par une loi normale

```
(%i133)N:30;
       p:0.7;
       N*p;
       N*(1-p);
       entiers:makelist(n,n,0,N)$
       listeBinomiale:makelist(pdf_binomial(n,N,p),n,0,N)$
       wxplot2d([[discrete, entiers, listeBinomiale], pdf_normal(x,N*p,sqrt(N*p*(1-p)))],
                   [x,0,N],
                   [style, points, lines],
                   [y,0,0.2],
                   [legend, "Loi binomiale", "Loi normale"],
                   [xlabel, "x"],


                   [ylabel, "y"]
       );
```

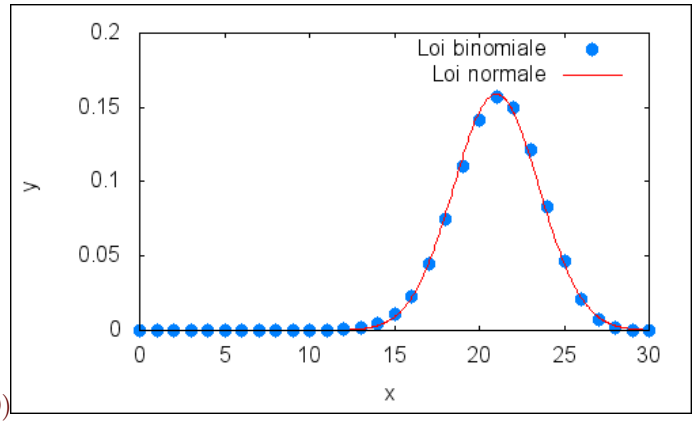
(%o133) 30

(%o134) 0.7

(%o135) 21.0

(%o136) 9.000000000000002

(%t139)



(%o139)

9.2.6 Approximation d'une loi binomiale par une loi de Poisson

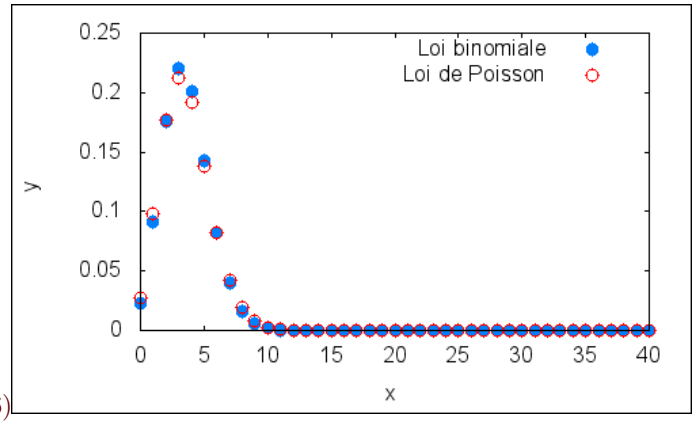
```
(%i140) N:40;
        p:0.09;
        N*p*(1-p);
        entiers:makelist(n,n,0,N)$
        listeBinomiale:makelist(pdf_binomial(n,N,p),n,0,N)$
        listePoisson:makelist(pdf_poisson(n,N*p),n,0,N)$
        wxplot2d([[discrete, entiers, listeBinomiale], [discrete, entiers, listePoisson]],
                    [x,0,N],
                    [style, points, points],
                    [y,0,0.25],
                    [legend, "Loi binomiale", "Loi de Poisson"],
                    [xlabel, "x"],


                    [ylabel, "y"]
        );
```

(%o140) 40

(%o141) 0.09

(%o142) 3.276

(%t146)



(%o146)

Chapitre 10

Algorithmique et programmation

10.1 Boucle Tant que

Boucle << tant que >> (<< while >>)

```
(%i1)  n:0$
       s:0$
       while n < 10 do (
           n : n + 1,
           s : s + n
       )$
       s;
```

(%o4) 55

10.2 Boucle Pour

Boucle << pour >> (<< for >>), le pas est égal à 1 par défaut :

```
(%i5)  s:0$
       for n from 0 thru 10 do (
           s : s + n
       )$
       s;
```

(%o7) 55

On peut écrire la ligne for en une seule ligne :

```
(%i8)  s:0$
       for n from 0 thru 10 do s : s + n$
       s;
```

(%o10) 55

Avec un pas de 2

```
(%i11) s:0$
       for n step 2 from 0 thru 10 do s : s + n$
       s;
```

(%o13) 30

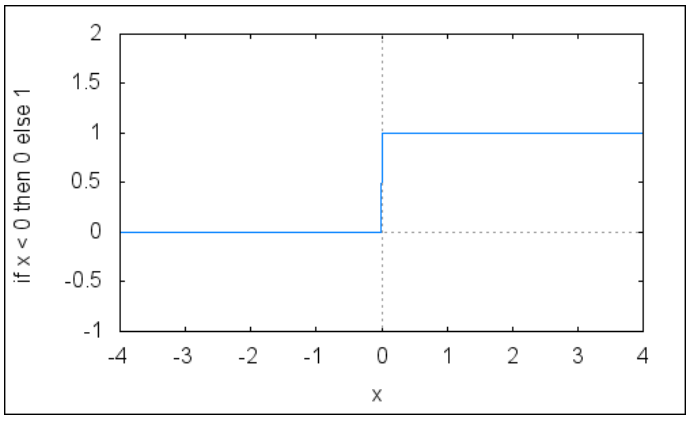
10.3 Test

Test : si, alors, sinon (if, then, else)

```
(%i14) f(x):=if x < 0 then 0 else 1;
```

(%o14) $\mathrm{f}\,(x) := \mathrm{if} x < 0 \mathrm{then} 0 \mathrm{else} 1$

```
(%i15) wxplot2d(f(x),[x, -4,4], [y, -1, 2]);
```



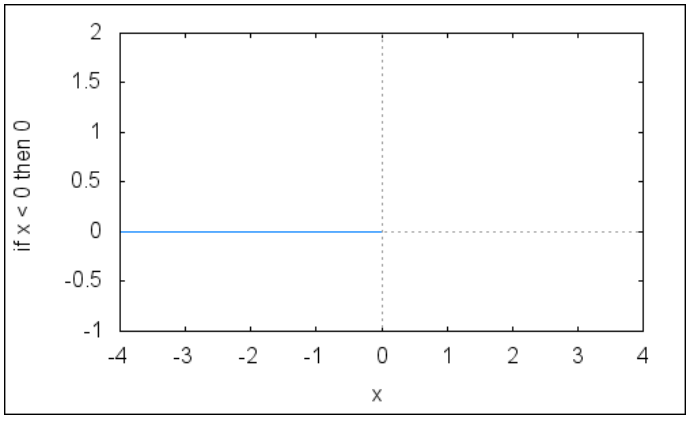
(%t15)

(%o15)

Sans le else :

```
(%i16) f(x):=if x < 0 then 0;
       wxplot2d(f(x),[x, -4,4], [y, -1, 2]);
```

(%o16) $\mathrm{f}\,(x) := \mathrm{if} x < 0 \mathrm{then} 0 plot2d : expressionevaluatestononnumericvaluesomewhereinplottingrange.$



(%t17)

(%o17)

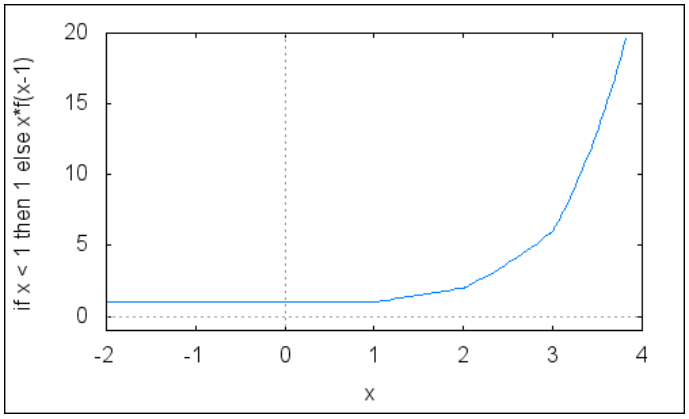
Récursivité définition de la factorielle (pour x entier)

```
(%i18) f(x):= if x < 1 then 1 else x*f(x-1);
```

(%o18) $\mathrm{f}\,(x) := \mathrm{if} x < 1 \mathrm{then} 1 \mathrm{else} x\,\mathrm{f}\,(x-1)$

```
(%i19) wxplot2d(f(x),[x, -2,4], [y, -1, 20]);
```

$plot2d : somevalueswereclipped.$



(%t19)

(%o19)

10.4 Logique (utile pour les tests)

Utilisation de vrai et faux

```
(%i20) true and flase;
```

(%o20) *flase*

```
(%i21) true or flase;
```

(%o21) *true*

Avec des parenthèses : associativité

```
(%i22) (true and flase) or true;
```

(%o22) *true*

Avec des variables

```
(%i23) x:true$
       y:false$
       x and y;
       x or y;
       kill(x, y);
```

(%o25) *false*

(%o26) *true*

(%o27) *done*

10.5 Structure d'une fonction/procédure

Définition d'une fonction (une valeur est retournée par un return)

```
nom(paramètres d'entrées):=block([paramètres locaux],
        block d'instructions 1,
        /* Commentaires */
        block d'instructions 2,
        return(valeur)
    );
```

Exemple : définition de la factorielle (pour x entier).

```
(%i28) g(x) := block([temp],
           temp:1,
           while x > 1 do(
               temp:x*temp,
               x:x-1),
           return(temp))$

(%i29) g(5);
```

(%o29) 120

Définition d'une procédure (la dernière valeur est retournée)

```
nom(paramètres d'entrées):=block([paramètres locaux],
        block d'instructions 1,
        /* Commentaires */
        block d'instructions 2
    );
```

Exemple : définition de la factorielle (pour x entier).

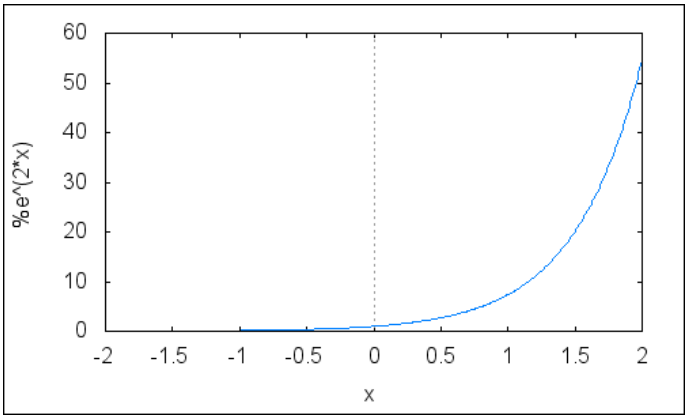
```
(%i30) g(x) := block([temp],
           temp:1,
           while x > 1 do(
               temp:x*temp,
               x:x-1),
           temp)$

(%i31) g(5);
```

(%o31) 120

Un procédure peut être utilisée si l'on souhaite par exemple afficher une courbe et que l'on a rien à retourner.

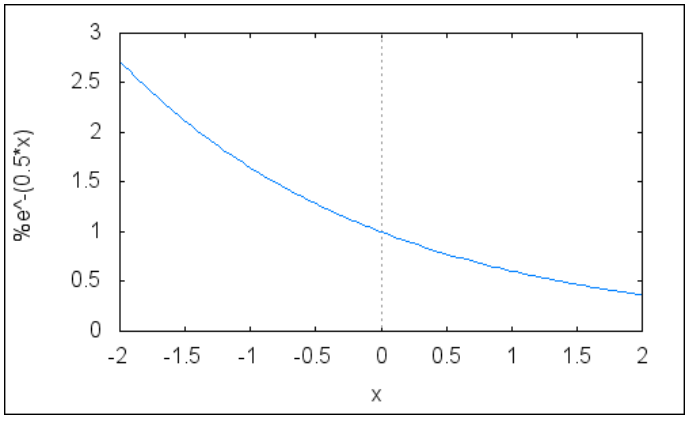
```
(%i32) courbeExponentielle(n) := block([],
           wxplot2d(exp(n*x), [x, -2, 2])
       )$

(%i33) courbeExponentielle(2);
```

(%t33)



(%o33)

```
(%i34) courbeExponentielle(-0.5);
```



(%t34)

(%o34)

10.6 Affichage de chaînes de caractères.

```
(%i35) test(x,y):=block([],
           if x > y then print(x, "est plus grand que ", y)
                    else print(x, "n'est pas plus grand que", y),
           return("Commande exécutée"))$
```

```
(%i36) test(4,3);
```

$4 est plus grand que 3$

(%o36) $Commande exécutée$

Il y a aussi la fonction display qui permet d'afficher le contenu d'une variable.

```
(%i37) x:4;
       display(x);
```

(%o37) $4x = 4$

(%o38) *done*

10.7 Utilisations d'hypothèses

Pour oublier les hypothèses ou les affectations de y :

```
(%i39) kill(y);
```

(%o39) *done*

```
(%i40) assume(y > 0);
```

(%o40) $[y > 0]$

```
(%i41) test(y^2+1,-y);
```

$y^2 + 1 est plus grand que - y$

(%o41) *Commandeexcute*

Pour oublier l'hypothèse :

```
(%i42) forget(y>0);
```

(%o42) $[y > 0]$

```
(%i43) test(y^2+1,-y);
```

(%o43) *Commandeexcute*

```
(%i44) assume(y < 0);
       test(y^2+1,-y);
```

(%o44) $[y < 0]$

(%o45) *Commandeexcute*

```
(%i46) facts();
```

(%o46) $[0 > y]$

```
(%i47) forget(all);
```

(%o47) $[true]$

```
(%i48) facts();
```

(%o48) $[0 > y]$

10.8 Types de variables

Si n est un entier, on peut écrire :

```
(%i49) declare(n, integer);
       assume(n>0);
```

(%o49) *done*

(%o50) $[redundant]$

Cela peut changer des calculs :

```
(%i51) cos(n*%pi);
```

(%o51) 1

Si on oublie tout :

```
(%i52) kill(all);
```

(%o0) *done*

```
(%i1)  cos(n*%pi);
```

(%o1) $\cos(\pi\, n)$

10.9 Commentaires

```
(%i2)  /* Ceci est un commentaire,
          la somme de deux entiers ! */
       1+2;
(%o2)  3
```

Chapitre 11

Équations différentielles

11.1 Équations différentielles du premier ordre

On définit une équation différentielle et son équation homogène :

```
(%i1)  equadiff:'diff(y,x)+y=exp(x);
       equadiff0:lhs(equadiff)=0;
```

(%o1) $\frac{d}{d\,x}\,y+y=e^{x}$

(%o2) $\frac{d}{d\,x}\,y+y=0$

Menu/Équations/Résoudre une équation différentielle...

```
(%i3)  ode2(equadiff, y, x);
       ode2(equadiff0, y, x);
```

(%o3) $y=e^{-x}\left(\frac{e^{2\,x}}{2}+\%c\right)$

(%o4) $y=\%c\,e^{-x}$

où la fonction inconnue est notée y, sa dérivée 'diff(y,x) et %c désigne un réel quelconque. On peut développer le résultat précédent en utilisant % et expand (% permet de récupérer le dernier résultat calculé) :

```
(%i5)  expand(%);
```

(%o5) $y=\%c\,e^{-x}$

Si par exemple on a la condition initiale y(0)=1 : Menu/Équations/Condition initiele (1)...

```
(%i6)  ic1(%, x=0, y=1);
```

(%o6) $y=e^{-x}$

On peut développer :

```
(%i7)  expand(%);
```

(%o7) $y=e^{-x}$

On peut vérifier que l'on obtient bien une solution de l'équation différentielle. La commande suivante remplace y par son expression dans l'équation différentielle, diff permet de calculer les dérivées.

```
(%i8)  ev(equadiff, %o6, diff);
```

(%o8) $0 = e^x$

On a l'égalité donc c'est bien une solution de l'équation différentielle.

11.2 Équations différentielles du second ordre

Menu/Équations/Résoudre une équation différentielle...

```
(%i9)  ode2('diff(y,x,2)+4*'diff(y,x)+4*y=1-exp(-x), y, x);
```

(%o9) $y = \frac{e^{-x}\,(e^x-4)}{4} + (\%k2\,x + \%k1)\,e^{-2\,x}$

où la fonction inconnue est notée y, sa dérivée 'diff(y,x), sa dérivée seconde 'diff(y,x,2) et %k1 et %k2 désignent deux réels quelconques. On peut développer

```
(%i10) expand(%o9);
```

(%o10) $y = -e^{-x} + \%k2\,x\,e^{-2\,x} + \%k1\,e^{-2\,x} + \frac{1}{4}$

Si on a les conditions initiales y(0)=1 et y'(0)=2 : Menu/Équations/Condition initiele (2)...

```
(%i11) ic2(%o9, x=0, y=1, 'diff(y,x)=2);
```

(%o11) $y = \frac{e^{-x}\,(e^x-4)}{4} + \left(\frac{9\,x}{2} + \frac{7}{4}\right)\,e^{-2\,x}$

Si on a les conditions initiales y(0)=1 et y(1)=3 : Menu/Équations/Problème aux limites...

```
(%i12) bc2(%o9,x = 0,y = 1,x = 1,y = 3);
```

(%o12) $y = \frac{e^{-x}\,(e^x-4)}{4} + \left(\frac{(11\,e^2+4\,e-7)\,x}{4} + \frac{7}{4}\right)\,e^{-2\,x}$

11.3 Exemple : masse - ressort - frottement visqueux

```
(%i13) equation:m*'diff(y,x,2)+c*'diff(y,x)+r*y=1;
```

(%o13) $m\left(\frac{d^2}{d\,x^2}\,y\right) + c\left(\frac{d}{d\,x}\,y\right) + r\,y = 1$

Pour delta positif où m est la masse, c est le coefficient de frottement, r la raideur, et le second membre de l'équation différentielle est donné par une force externe appliquée à la masse

```
(%i14) forget(facts());
       assume(c**2-4*m*r>0);
       ode2(equation, y, x);
```

(%o14) [[]]

(%o15) $[c^2 > 4\,m\,r]$

(%o16) $y = \%k1\, e^{\frac{\left(\sqrt{\frac{c^2}{m^2}-\frac{4\,r}{m}}-\frac{c}{m}\right)\,x}{2}} + \%k2\, e^{\frac{\left(-\sqrt{\frac{c^2}{m^2}-\frac{4\,r}{m}}-\frac{c}{m}\right)\,x}{2}} + \frac{1}{r}$

Pour delta nul

```
(%i17) forget(facts());
       assume(equal(c**2-4*m*r,0));
       ode2(equation, y, x);
```

(%o17) $[[c^2 > 4\,m\,r]]$

(%o18) $[\mathrm{equal}\left(c^2, 4\,m\,r\right)]$

(%o19) $y = \left(\%k2\,x + \%k1\right)\, e^{-\frac{c\,x}{2\,m}} + \frac{4\,m}{c^2}$

Pour delta négatif

```
(%i20) forget(facts());
       assume(c**2-4*m*r<0);
       ode2(equation, y, x);
```

(%o20) $[[\mathrm{equal}\left(c^2, 4\,m\,r\right)]]$

(%o21) $[4\,m\,r > c^2]$

(%o22) $y = e^{-\frac{c\,x}{2\,m}}\left(\%k1\,\sin\left(\frac{\sqrt{\frac{4\,r}{m}-\frac{c^2}{m^2}}\,x}{2}\right) + \%k2\,\cos\left(\frac{\sqrt{\frac{4\,r}{m}-\frac{c^2}{m^2}}\,x}{2}\right)\right) + \frac{1}{r}$

Chapitre 12

Séries de Fourier

Pour le paragraphe « Utilisation du package piecewise », il faut utiliser une bibliothèque, pour la récupérer, cliquer deux fois sur le lien ci-contre : lien Piecewise.

Enregistrez ce fichier dans le répertoire de votre choix et n'oubliez pas de donner le chemin complet lors de l'utilisation :

```
load("C:/chemin-complet/pw.mac");
```

12.1 Définition de la fonction périodique

On suppose que l'on se donne une fonction périodique de période T, définie sur $\mathbb{R}$ et qu'une seule expression est donnée sur l'intervalle $[a;b]$ de longueur T. Pour les fonctions définies par morceaux sur l'intervalle $[a;b]$, le lecteur est renvoyé à la section sur le package piecewise. La notation g désignera la fonction définie sur $[a;b]$. La notation f désignera la fonction définie sur $\mathbb{R}$.

```
(%i1)  a:-%pi;
       b:%pi;
       T:b-a;
       %omega:2*%pi/T;
       g(t):=t;
       f(t):=g(t-(b-a)*floor((t-a)/(b-a)));
```

(%o1) $-\pi$

(%o2) π

(%o3) $2\,\pi$

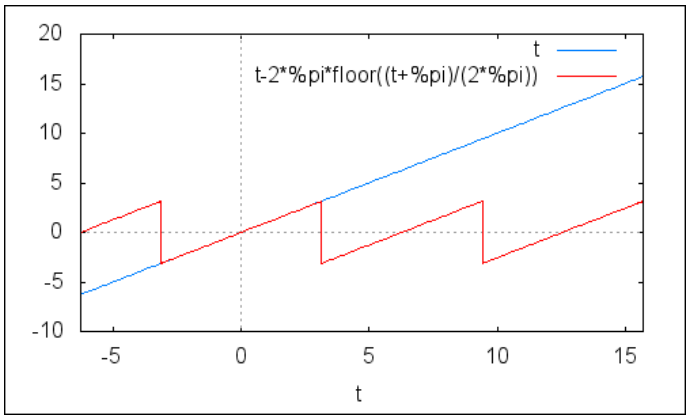
(%o4) 1

(%o5) $\mathrm{g}\left(t\right):=t$

(%o6) $\mathrm{f}\left(t\right):=\mathrm{g}\left(t-\left(b-a\right)\,\mathrm{floor}\left(\frac{t-a}{b-a}\right)\right)$

12.2 Courbe de la fonction étudiée

```
(%i7)  wxplot2d([g(t), f(t)], [t,a-b,3*b-2*a])$
```



(%t7)

12.3 Calcul des coefficients de Fourier

12.3.1 Calcul de la moyenne sur une période : a_0

```
(%i8)  a0:1/T*integrate(g(t),t,a,b);
```

(%o8) 0

12.3.2 Calcul des coefficients a_n et b_n

```
(%i9)  (declare(n,integer), n>0);
```

(%o9) $n > 0$

```
(%i10) an:2/T*integrate(g(t)*cos(n*%omega*t),t,a,b);
       define(a(n),an);
```

(%o10) 0

(%o11) $\mathrm{a}\left(n\right) := 0$

```
(%i12) bn:2/T*integrate(g(t)*sin(n*%omega*t),t,a,b);
       define(b(n),bn);
```

(%o12) $-\dfrac{2\left(-1\right)^{n}}{n}$

(%o13) $\mathrm{b}\left(n\right) := -\dfrac{2\left(-1\right)^{n}}{n}$

N désigne le nombre de termes affichés. La fonction makelist permet de créer la liste des entiers de 1 à N et map permet de caculer les images par la fonction a (ou b) des nombres de la liste précédente.

```
(%i14) N:4$
       map('a, makelist(i,i,1,N));
       map('b, makelist(i,i,1,N));
```

(%o15) $[0, 0, 0, 0]$

(%o16) $[2, -1, \frac{2}{3}, -\frac{1}{2}]$

12.4 Somme de Fourier partielle

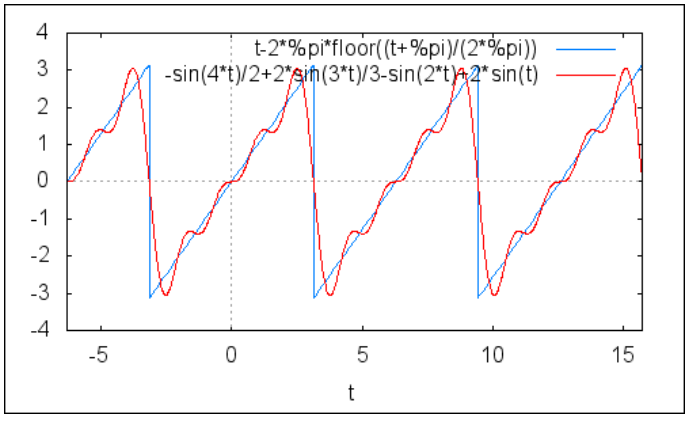
On désigne par $\mathbb{N}$ le nombre de termes de la somme de série de Fourier partielle :

```
(%i17) Sf(t,N):=a0+sum(a(n)*cos(n*%omega*t)+b(n)*sin(n*%omega*t), n, 1, N);
```

(%o17) $\mathrm{Sf}\,(t, N) := a0 + \sum_{n=1}^{N} \mathrm{a}\,(n)\,\cos\,(n\,\omega\,t) + \mathrm{b}\,(n)\,\sin\,(n\,\omega\,t)$

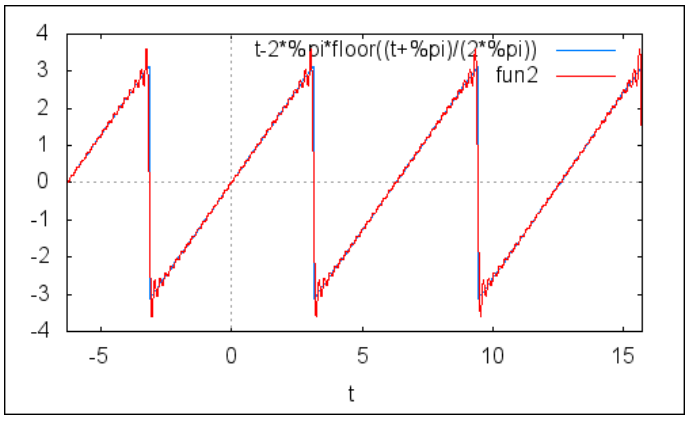
```
(%i18) wxplot2d([f(t),Sf(t,4)], [t,a-b,3*b-2*a])$
```

(%t18)




```
(%i19) wxplot2d([f(t),Sf(t,30)], [t,a-b,3*b-2*a])$
```

(%t19)



12.5 Spectre

Le spectre de f est donné par la suite (A_n), où $A_n = \sqrt{a_n^2 + b_n^2}$, pour $n \geq 1$.

```
(%i20) define(A(n), sqrt(a(n)^2+b(n)^2));
```

(%o20) $\mathrm{A}\,(n) := \dfrac{2}{|n|}$

12.6 Valeur efficace et formule de Parseval

La valeur efficace est donnée par : $f_{\text{eff}}^2 = \dfrac{1}{T}\displaystyle\int_a^{T+a} f^2(t)4dt$.

La formule de Parseval est : $f_{\text{eff}}^2 = a_0^2 + \dfrac{1}{2}\displaystyle\sum_{n=1}^{+\infty}(a_n^2 + b_n^2)$.

La valeur efficace est approchée par : $P(N) = a_0^2 + \dfrac{1}{2}\displaystyle\sum_{n=1}^{N}(a_n^2 + b_n^2)$.

```
(%i21) P(N):=a0^2+1/2*sum(a(n)^2+b(n)^2, n, 1, N);
```

(%o21) $\mathrm{P}\,(N) := a0^2 + \dfrac{1}{2}\displaystyle\sum_{n=1}^{N} \mathrm{a}\,(n)^2 + \mathrm{b}\,(n)^2$

```
(%i22) N:10;
       for n:1 thru N step 1 do print("P(",n,")=", float(P(n)), "  ;  ");
```

(%o22) $10P(1) = 2.0; P(2) = 2.5; P(3) = 2.722222222222222; P(4) = 2.847222222222222; P(5) = 2.927222222222222; P(6) = 2.982777777777778; P(7) = 3.02359410430839; P(8) = 3.05484410430839; P(9) = 3.079535462333082; P(10) = 3.099535462333082;$

(%o23) *done*

```
(%i24) Peff:float(1/T*integrate(f(t)**2,t,a,b));
```

(%o24) 3.289868133696452

Erreur relative

```
(%i25) for n:1 thru N step 1 do print("err(",n,")=", float((Peff-P(n))/Peff), "  ;  ");
```

$err(1) = 0.39207289814597; err(2) = 0.24009112268247; err(3) = 0.17254366692091; err(4) = 0.13454822305503; err(5) = 0.11023113898087; err(6) = 0.093344275040481; err(7) = 0.080937599492439; err(8) = 0.07143873852597; err(9) = 0.063933465663575; err(10) = 0.057854194645034;$

(%o25) *done*

12.7 Utilisation du package piecewise

Package à utiliser pour les fonctions définie par morceaux sur une période.

```
(%i26) kill(all);
```

(%o0) *done*

```
(%i1)  load("maxima-BTS_V_1/pw.mac");
```

(%o1) $maxima - BTS\_V\_1/pw.mac$

On suppose que la fonction g est donnée sur les intervalles \$left[a_1;b_1\$right], \$ldtos, \$left[a_r;b_r\$right] par les expressions $g_1(x), \$ldots, $g\_r(x)$. Dans Maxima et avec la bibliothèque pw, on peut définir la fonction g directement par :

define($g(t)$, piecewise([a_1, $g\_1(t)$, b_1, a_2, $g\_2(t)$, b_2, \ldots, a_r, $g\_r(t)$, b_r,],t)) ;

```
(%i2)  a:0;
       b:4*%pi;
       T:b-a;
       %omega:2*%pi/T;
       define(g(t), piecewise([a, %pi-t, %pi, 0, b/2, 1, b],t));
       f(t):=g(t-(b-a)*floor((t-a)/(b-a)));
       wxplot2d([f(t)], [t,a-b,3*b-2*a],[y,-0.5,3.5])$
```

(%o2) 0

(%o3) $4\,\pi$

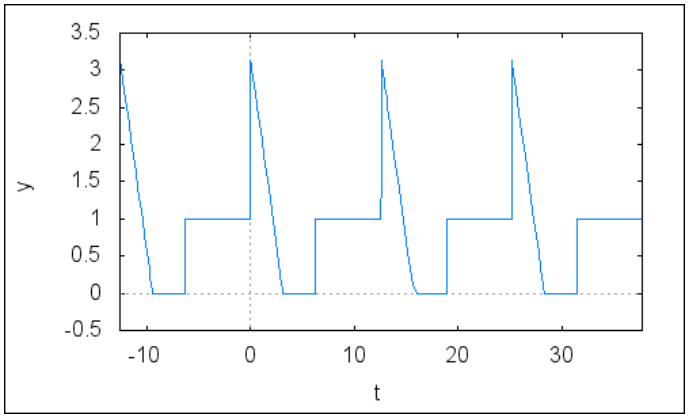
(%o4) $4\,\pi$

(%o5) $\frac{1}{2}$

(%o6) $$\mathrm{g}\left(t\right) := \frac{\left(\pi - t\right)\,\left(\mathrm{signum}\left(t\right) - \mathrm{signum}\left(t - \pi\right)\right)}{2} + \frac{\mathrm{signum}\left(t - 2\,\pi\right) - \mathrm{signum}\left(t - 4\,\pi\right)}{2}$$

(%o7) $$\mathrm{f}\left(t\right) := \mathrm{g}\left(t - \left(b - a\right)\,\mathrm{floor}\left(\frac{t - a}{b - a}\right)\right)$$

(%t8)



La fonction pwint est la fonction integrate utilisable sur les fonctions définies par pw, c'est l'intégrale d'une fonction définie par morceaux.

```
(%i9)  a0:radcan(1/T*pwint(g(t), t, a, b));
```

(%o9) $\frac{\pi + 4}{8}$

```
(%i10) declare(n, integer);
       assume(n>0);
```

(%o10) *done*

(%o11) $[n > 0]$

La fonction radcan, sert à simplifier l'expression obtenue (simplification canonique).

```
(%i12) an:radcan(2/T*pwint(g(t)*cos(n*%omega*t), t, a, b));
       define(a(n),an);
```

(%o12) $$-\frac{2\cos\left(\frac{\pi\,n}{2}\right) - 2}{\pi\,n^2}$$

(%o13) $\mathrm{a}\left(n\right) := -\frac{2\cos\left(\frac{\pi\,n}{2}\right)-2}{\pi\,n^2}$

```
(%i14) bn:radcan(2/T*pwint(g(t)*sin(n*%omega*t), t, a, b));
       define(b(n),bn);
```

(%o14) $-\frac{2\sin\left(\frac{\pi\,n}{2}\right)-n\left(-1\right)^n+\left(1-\pi\right)\,n}{\pi\,n^2}$

(%o15) $\mathrm{b}\left(n\right) := -\frac{2\sin\left(\frac{\pi\,n}{2}\right)-n\left(-1\right)^n+\left(1-\pi\right)\,n}{\pi\,n^2}$

```
(%i16) N:4$
       map('a, makelist(i,i,1,N));
       map('b, makelist(i,i,1,N));
```

(%o17) $[\frac{2}{\pi},\frac{1}{\pi},\frac{2}{9\,\pi},0]$

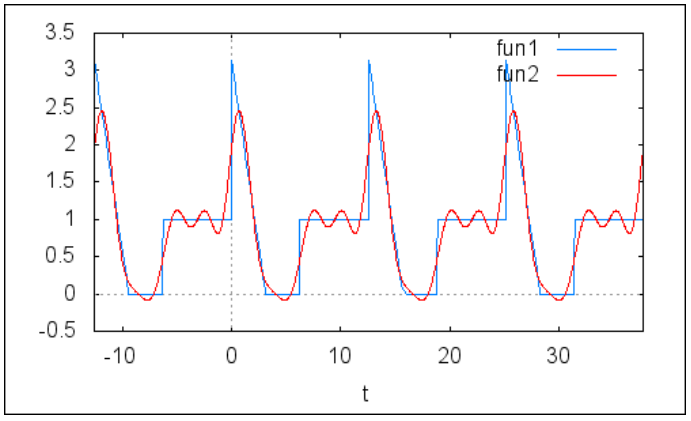
(%o18) $[-\frac{4-\pi}{\pi},-\frac{2\,\left(1-\pi\right)-2}{4\,\pi},-\frac{3\,\left(1-\pi\right)+1}{9\,\pi},-\frac{4\,\left(1-\pi\right)-4}{16\,\pi}]$

```
(%i19) Sf(t,N):=a0+sum(a(n)*cos(n*%omega*t)+b(n)*sin(n*%omega*t), n, 1, N);
```

(%o19) $\mathrm{Sf}\left(t,N\right) := a0+\sum_{n=1}^{N}\mathrm{a}\left(n\right)\,\cos\left(n\,\omega\,t\right)+\mathrm{b}\left(n\right)\,\sin\left(n\,\omega\,t\right)$

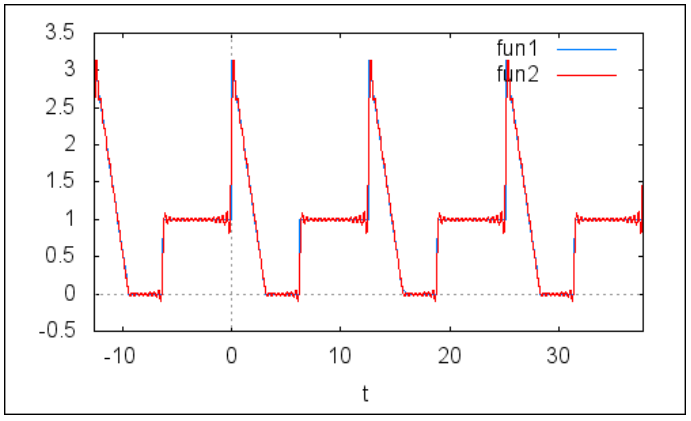
```
(%i20) wxplot2d([f(t),Sf(t,5)], [t,a-b,3*b-2*a])$
```

(%t20)




```
(%i21) wxplot2d([f(t),Sf(t,30)], [t,a-b,3*b-2*a])$
```

(%t21)



Chapitre 13

Transformée de Laplace

13.1 Définition de l'échelon unité

Échelon unité : $v(t) = 0$ si $t < 0$ et 1 si $t > 0$;

Dans Maxima c'est la fonction u suivante qui est définie : $u(t) = 0$ si $t <= 0$ et 1 si $t > 0$;

Dans la pratique en mathématiques, c'est la fonction v que l'on utilise mais avec maxima c'est avec la fonction u que cela fonctionne correctement !

```
(%i1)  u(t):=unit_step(t);
       u(-1);
       u(0);
       u(1);
```

(%o1) $\mathrm{u}\,(t) := \mathrm{unit\_step}\,(t)$

(%o2) 0

(%o3) 0

(%o4) 1

La fonction v peut être définie de la façon suivante :

```
(%i5)  v(t):=unit_step(t)+kron_delta(t,0);
       v(-1);
       v(0);
       v(1);
```

(%o5) $\mathrm{v}\,(t) := \mathrm{unit\_step}\,(t) + \mathrm{kron\_delta}\,(t, 0)$

(%o6) 0

(%o7) 1

(%o8) 1

Les calculs sur la transformée de Laplace ne changent pas quel que soit le choix d'une des deux définitions précédentes.

13.2 Calculs de transformées de Laplace

Menu/Calculs/Transformée de Laplace...

```
(%i9)  laplace(u(t), t, p);
```

(%o9) $\frac{1}{p}$

```
(%i10) laplace(t*u(t), t, p);
```

(%o10) $\frac{1}{p^2}$

Ci-dessous, $\Gamma(n+1)$ désigne $n!$.

```
(%i11) laplace(t^n*u(t), t, p);
```

$Isn + 1positive, negative, orzero?positive;$

(%o11) $\gamma\left(n+1\right)\,p^{-n-1}$

Retard $u(t-1)$

```
(%i12) laplace(u(t-tau), t, p);
```

(%o12) $\frac{e^{-p\,\tau}}{p}$

Exponentielle

```
(%i13) laplace(exp(-a*t), t, p);
```

(%o13) $\frac{1}{p+a}$

Fonctions trigonométriques

```
(%i14) laplace(cos(omega*t)*u(t), t, p);
```

(%o14) $\frac{p}{p^2+\omega^2}$

```
(%i15) laplace(sin(omega*t)*u(t), t, p);
```

(%o15) $\frac{\omega}{p^2+\omega^2}$

Propriétés : les deux premières ne sont par reconnues...

```
(%i16) laplace(f(alpha*t)*u(alpha*t), t, p);
```

(%o16) $\mathrm{laplace}\left(\mathrm{f}\left(\alpha\,t\right)\,\mathrm{unit\_step}\left(\alpha\,t\right),t,p\right)$

```
(%i17) laplace(f(t-tau), t, p);
```

(%o17) $\mathrm{laplace}\left(\mathrm{f}\left(t-\tau\right),t,p\right)$

```
(%i18) laplace(exp(-a*t)*f(t), t, p);
```

$Isp + apositive, negative, orzero?positive;$

(%o18) $\left.\mathrm{laplace}\left(\mathrm{f}\left(t\right),t,p\right)\right|_{p=p+a}$

```
(%i19) laplace(diff(f(t), t),t,p);
```

(%o19) $p\,\mathrm{laplace}\left(\mathrm{f}\left(t\right),t,p\right)-\mathrm{f}\left(0\right)$

```
(%i20) laplace(diff(f(t), t, 2),t,p);
```

$$(\%o20)\quad -\left.\frac{d}{d\,t}\,\mathrm{f}\,(t)\right|_{t=0}+p^2\,\mathrm{laplace}\,(\mathrm{f}\,(t)\,,t,p)-\mathrm{f}\,(0)\;p$$

```
(%i21) laplace(-t*f(t), t, p);
```

$$(\%o21)\quad \frac{d}{d\,p}\,\mathrm{laplace}\,(\mathrm{f}\,(t)\,,t,p)$$

```
(%i22) laplace(integrate(f(s), s, 0, t), t, p);
```

$$(\%o22)\quad \frac{\mathrm{laplace}\,(\mathrm{f}\,(t)\,,t,p)}{p}$$

13.3 Calculs de transformées inverse de Laplace

Menu/Calculs/Transformée inverse de Laplace... Attention, il faut multiplier les résultats par $u(t)$.

```
(%i23) ilt(1/p, p, t);
```

(%o23) 1

Si le dénominateur est du second degré avec un discriminant strictement positif :

```
(%i24) ilt((p+1)/(p^2+4*p-1), p, t);
```

$$(\%o24)\quad e^{-2\,t}\left(\cosh\left(\sqrt{5}\,t\right)-\frac{\sinh\left(\sqrt{5}\,t\right)}{\sqrt{5}}\right)$$

Des fonctions trigonométriques hyperboliques ! Pour éviter cela, on utilise exponentialize :

```
(%i25) exponentialize(ilt((p+1)/(p^2+4*p-1), p, t));
```

$$(\%o25)\quad e^{-2\,t}\left(\frac{e^{\sqrt{5}\,t}+e^{-\sqrt{5}\,t}}{2}-\frac{e^{\sqrt{5}\,t}-e^{-\sqrt{5}\,t}}{2\,\sqrt{5}}\right)$$

On peut aussi développer !

```
(%i26) expand(exponentialize(ilt((p+1)/(p^2+4*p-1), p, t)));
```

$$(\%o26)\quad -\frac{e^{\sqrt{5}\,t-2\,t}}{2\,\sqrt{5}}+\frac{e^{\sqrt{5}\,t-2\,t}}{2}+\frac{e^{-\sqrt{5}\,t-2\,t}}{2\,\sqrt{5}}+\frac{e^{-\sqrt{5}\,t-2\,t}}{2}$$

Si le dénominateur est du second degré avec un discriminant nul :

```
(%i27) ilt((p+1)/(p^2+2*p+1), p, t);
```

$$(\%o27)\quad e^{-t}$$

Rien à modifier ! Si le dénominateur est du second degré avec un discriminant strictement négatif :

```
(%i28) ilt((p+1)/(p^2+p+1), p, t);
```

$$(\%o28)\quad e^{-\frac{t}{2}}\left(\frac{\sin\left(\frac{\sqrt{3}\,t}{2}\right)}{\sqrt{3}}+\cos\left(\frac{\sqrt{3}\,t}{2}\right)\right)$$

Attention : "ilt" (inverse laplace transform) ne fonctionne que pour les fonctions rationnelles pour lesquelles le dénominateur est factorisable en produits de facteurs linéaires et quadratiques (dont on sait caluler la valeur exacte des racines). Cela pose des problèmes si l'on veut travailler avec des retards. Il faut dans ce cas utiliser un autre logiciel ou trouver une autre solution !

13.4 Résolutions d'équations différentielles avec la transformée de Laplace

– Résolution directe : Menu/Equations/Résoudre une équation différentielle avec Laplace...

```
(%i29) desolve(['diff(y(t), t, 2) + 5*'diff(y(t), t) + 4*y(t) = u(t)],[y(t)]);
```

(%o29) $\mathrm{y}\left(t\right)=-\frac{e^{-4\,t}\,\left(4\,\left(\frac{d}{d\,t}\,\mathrm{y}\left(t\right)\left|_{t=0}\right.\right)+4\,\mathrm{y}\left(0\right)-1\right)}{12}+\frac{e^{-t}\,\left(\frac{d}{d\,t}\,\mathrm{y}\left(t\right)\left|_{t=0}\right.+4\,\mathrm{y}\left(0\right)-1\right)}{3}+\frac{1}{4}$

Contrairement à ode2 les paramètres restants sont les valeurs de y(0) et y'(0) et non les deux réels quelconques notés %k1 et %k2 dans maxima. Pour faciliter la lecture des résultats on pose les conditions initiales f(0) = y0 et f'(0) = yp0 :

```
(%i30) atvalue(y(t), t=0, y0);
       atvalue('diff(y(t), t), t= 0, yp0);
```

(%o30) $y0$

(%o31) $yp0$

On affiche le changement :

```
(%i32) ''%i19;
```

(%o32) $p\,\mathrm{laplace}\left(\mathrm{f}\left(t\right),t,p\right)-\mathrm{f}\left(0\right)$

On développe :

```
(%i33) expand(%o22);
```

(%o33) $\frac{\mathrm{laplace}\left(\mathrm{f}\left(t\right),t,p\right)}{p}$

Avec les conditions initiales : y(0)=1 et y'(0)=2 :

```
(%i34) subst(1,y0,%o23);
```

(%o34) 1

```
(%i35) subst(2,yp0,%o24);
```

(%o35) $e^{-2\,t}\,\left(\cosh\left(\sqrt{5}\,t\right)-\frac{\sinh\left(\sqrt{5}\,t\right)}{\sqrt{5}}\right)$

– Par étapes :

```
(%i36) equation: 'diff(y(t), t, 2) + 5*'diff(y(t), t) + 4*y(t) = u(t);
```

(%o36) $\frac{d^2}{d\,t^2}\,\mathrm{y}\left(t\right)+5\,\left(\frac{d}{d\,t}\,\mathrm{y}\left(t\right)\right)+4\,\mathrm{y}\left(t\right)=\mathrm{unit\_step}\left(t\right)$

```
(%i37) equationtransformee:laplace(equation,t,p);
```

(%o37) $-yp0-p\,y0+5\,\left(p\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)-y0\right)+p^2\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)+4\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)=\frac{1}{p}$

On impose des conditions initiales :

```
(%i38) atvalue(y(t), t=0, 1);
       atvalue('diff(y(t), t), t= 0, 2);
       equationtransformee:laplace(equation,t,p);
```

(%o38) 1

(%o39) 2

(%o40) $5\,\left(p\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)-1\right)+p^2\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)+4\,\text{laplace}\left(\text{y}\left(t\right),t,p\right)-p-2=\frac{1}{p}$

On résout l'équation linéaire :

```
(%i41) equationsolution: solve(equationtransformee, 'laplace(y(t), t, p));
```

(%o41) $[\text{laplace}\left(\text{y}\left(t\right),t,p\right)=\frac{p^2+7\,p+1}{p^3+5\,p^2+4\,p}]$

On donne directement l'original. Si map et lambda paraissent obscurs, on peut faire un copier coller !

```
(%i42) map( lambda( [eq], ilt(eq, p, t)), equationsolution);
```

(%o42) $[\text{y}\left(t\right)=\frac{5\,e^{-t}}{3}-\frac{11\,e^{-4\,t}}{12}+\frac{1}{4}]$

On peut aussi décomposer le calcul en passant par la décomposition en éléments simples.

```
(%i43) map( lambda( [expression], partfrac(expression, p)), equationsolution);
```

(%o43) $[\text{laplace}\left(\text{y}\left(t\right),t,p\right)=-\frac{11}{12\,\left(p+4\right)}+\frac{5}{3\,\left(p+1\right)}+\frac{1}{4\,p}]$

```
(%i44) map( lambda( [expression], ilt(expression, p, t)), %);
```

(%o44) $[\text{y}\left(t\right)=\frac{5\,e^{-t}}{3}-\frac{11\,e^{-4\,t}}{12}+\frac{1}{4}]$

L'ordre des termes a été modifié !

Chapitre 14

Transformée en z

Pour ce chapitre, il faut utiliser deux bibliothèques, pour les récupérer, cliquer deux fois sur les liens ci-contre : lien 1 z-transform et lien 2 supplement-z-transform.

Enregistrez ces fichier dans le répertoire de votre choix et n'oubliez pas de donner le chemin complet lord de l'utilisation :

```
load("C:/chemin-complet/z-transform.mac");
```

```
load("C:/chemin-complet/supplement-z-transform.mac");
```

14.1 Introduction

Définition des fonctions tz (transformée en z), tiz (transformée inverse en z) tiz retourne une liste, les premiers termes sont donnés lorsqu'ils sont nuls, le dernier terme de la liste (disons d'ordre r dans la liste) donne l'expression du signal discret en fonction de n pour n plus grand ou égal à r. On peut obtenir ce terme générique directement avec la fonction tizgen. En utilisant la fonction plotliste, on peut représenter le signal discret.

```
(%i1)  load("z-transform.mac")$
       load("supplement-z-transform.mac")$
```

14.2 Transformée en z

Des exemples :

Dirac (impulsion unité discrète) et sa trasformée

```
(%i3)  dirac(n):=kron_delta(n,0)$
       tz(dirac(n));
```

(%o4) 1

Dirac retardé de k et sa transformée

```
(%i5)  d(n,k):=kron_delta(n,k)$
       tz(d(n,k));
```

(%o6) $\frac{1}{z^k}$

Échelon unité discret

```
(%i7)  u(n):=unit_step(n)$
       tz(u(n));
```

$$(\%o8)\quad \frac{z}{z-1}$$

Rampe unité causale discrète : n

```
(%i9)  tz(n*u(n));
       factor(ev(''(tz(n*u(n))),diff));
       tz(n);
```

$$(\%o9)\quad -z\left(\frac{d}{d\,z}\,\frac{z}{z-1}\right)$$

$$(\%o10)\quad \frac{z}{(z-1)^2}$$

$$(\%o11)\quad \frac{z}{(z-1)^2}$$

Carré

```
(%i12) tz(n^2*u(n));
       factor(ev(''(tz(n^2*u(n))),diff));
       tz(n^2);
```

$$(\%o12)\quad -z\left(\frac{d}{d\,z}\left(-z\left(\frac{d}{d\,z}\,\frac{z}{z-1}\right)\right)\right)$$

$$(\%o13)\quad \frac{z\,(z+1)}{(z-1)^3}$$

$$(\%o14)\quad \frac{z\,(z+1)}{(z-1)^3}$$

Exponentiel

```
(%i15) tz(a^n*u(n));
       factor(ev(''(tz(a^n*u(n))),nouns));
       tz(a^n);
```

$$(\%o15)\quad \text{substitute}\left(\frac{z}{a},z,\frac{z}{z-1}\right)$$

$$(\%o16)\quad \frac{z}{z-a}$$

$$(\%o17)\quad \frac{z}{z-a}$$

```
(%i18) tz(b^(n-1)*u(n-1));
```

$$(\%o18)\quad \frac{1}{z-b}$$

```
(%i19) tz(exp(a*n));
```

$$(\%o19)\quad \frac{z}{z-e^a}$$

Trigonométrie

```
(%i20) tz(sin(a*n));
       tz(cos(a*n));
```

$$(\%o20)\quad \frac{\sin(a)\ z}{z^2-2\cos(a)\ z+1}$$

$$(\%o21)\quad \frac{z\ (z-\cos(a))}{z^2-2\cos(a)\ z+1}$$

```
(%i22) factor(ev(''(tz(b^n*cos(a*n))),nouns));
       factor(ev(''(tz(b^n*sin(a*n))),nouns));
```

$$(\%o22)\quad \frac{z\ (z-\cos(a)\ b)}{z^2-2\cos(a)\ b\,z+b^2}$$

$$(\%o23)\quad \frac{\sin(a)\ b\,z}{z^2-2\cos(a)\ b\,z+b^2}$$

Des propriétés Linéarité

```
(%i24) tz(lambda*x(n)+mu*y(n));
```

$$(\%o24)\quad \mathrm{z\_transform}\left(\mathrm{x}\left(n\right),n,z\right)\ lambda+\mu\,\mathrm{z\_transform}\left(\mathrm{y}\left(n\right),n,z\right)$$

Retard

```
(%i25) tz(x(n-2)*u(n-2));
       y(n):=n^2*u(n);
       factor(ev(''(tz(y(n-2))),diff));
```

$$(\%o25)\quad \frac{\mathrm{z\_transform}\left(\mathrm{x}\left(n\right),n,z\right)}{z^2}$$

$$(\%o26)\quad \mathrm{y}\left(n\right):=n^2\,\mathrm{u}\left(n\right)$$

$$(\%o27)\quad \frac{z+1}{\left(z-1\right)^3 z}$$

L'avance ne fonctionne pas encore !

```
(%i28) tz(x(n+3)*unit_step(n+3));
       y(n):=n^2*u(n);
       ev(''(tz(y(n+2))), diff);
       tz(expand((n+2)^2));
```

$$(\%o28)\quad \mathrm{z\_transform}\left(\mathrm{x}\left(n\right),n,z\right)\ z^3$$

$$(\%o29)\quad \mathrm{y}\left(n\right):=n^2\,\mathrm{u}\left(n\right)$$

$$(\%o30)\quad -z^3\left(-z\left(\frac{2\,z}{\left(z-1\right)^3}-\frac{2}{\left(z-1\right)^2}\right)+\frac{z}{\left(z-1\right)^2}-\frac{1}{z-1}\right)$$

$$(\%o31)\quad \frac{z\ (z+1)}{\left(z-1\right)^3}+\frac{4\,z}{z-1}+\frac{4\,z}{\left(z-1\right)^2}$$

Multiplication par a^n

```
(%i32) tz(a^n*x(n));
       y(n):=n^2*u(n);
       tz(a^n*y(n));
       factor(ev(''(tz(a^n*y(n))),nouns));
```

$$(\%o32)\quad \mathrm{substitute}\left(\frac{z}{a},z,\mathrm{z\_transform}\left(\mathrm{x}\left(n\right),n,z\right)\right)$$

(%o33) $\mathrm{y}\left(n\right) := n^2\,\mathrm{u}\left(n\right)$

(%o34) $\mathrm{substitute}\left(\frac{z}{a}, z, -z\,\left(\frac{d}{d\,z}\,\left(-z\,\left(\frac{d}{d\,z}\,\frac{z}{z-1}\right)\right)\right)\right)$

(%o35) $\frac{a\,z\,\left(z+a\right)}{\left(z-a\right)^3}$

14.3 Transformée en z inverse

Le premier terme donne la valeur du signal discret pour n=0. Le dernier terme est l'expression du terme général du signal.

```
(%i36) tiz(z/(z-1));
```

(%o36) $[1, 1]$

```
(%i37) tiz(z/(z-1)^2);
```

(%o37) $[0, n]$

```
(%i38) tiz((z*(z+1))/(z-1)^3);
```

(%o38) $[0, n^2]$

```
(%i39) tiz(z/(z-b));
```

(%o39) $[1, b^n]$

```
(%i40) tiz(z/(z-%e^a));
```

(%o40) $[1, e^{a\,n}]$

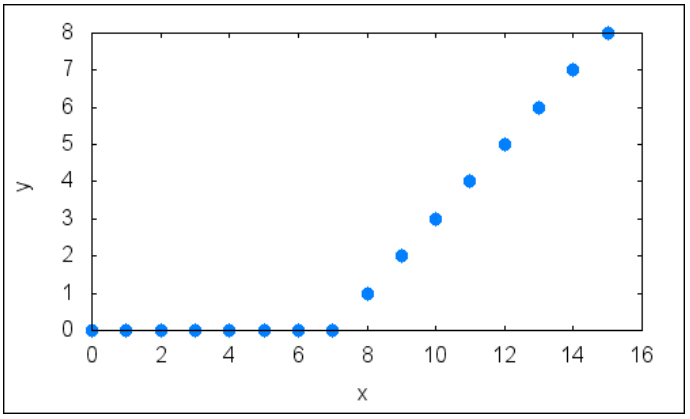
```
(%i41) tiz(z^(6)/(z-1)^2);
```

(%o41) $[\infty, n+5]$

```
(%i42) define(f(z),z^(-6)/(z-1)^2);
       signal:tiz(f(z));
       plotliste(signal,15);
```

(%o42) $\mathrm{f}\left(z\right) := \frac{1}{\left(z-1\right)^2\,z^6}$

(%o43) $[0, 0, 0, 0, 0, 0, 0, n-7]$



(%t44)

(%o44)

14.4 Transformée en z à partir d'une liste

```
(%i45) tzliste([1.1,1.2,1.3,1.4,1.5*n]);
```

$$(\%o45)\quad \frac{1.5\left(\frac{4\,z}{z-1}+\frac{z}{(z-1)^2}\right)}{z^4}+\frac{1.2}{z}+\frac{1.3}{z^2}+\frac{1.4}{z^3}+1.1$$

```
(%i46) tzliste([0,0,n-2]);
```

$$(\%o46)\quad \frac{1}{(z-1)^2\,z}$$

```
(%i47) factor(tzliste([n+1]));
```

$$(\%o47)\quad \frac{z^2}{(z-1)^2}$$

14.5 Exemples d'utilisations

```
(%i48) tz(n^2+3*n+2);
```

$$(\%o48)\quad \frac{z\,(z+1)}{(z-1)^3}+\frac{2\,z}{z-1}+\frac{3\,z}{(z-1)^2}$$

```
(%i49) factor(ev(''(tz(n*2^n)),diff, nouns));
```

$$(\%o49)\quad \frac{2\,z}{(z-2)^2}$$

```
(%i50) factor(tz(n-3));
       factor(tzliste([0, 0, 0, n-3]));
```

$$(\%o50)\quad -\frac{z\,(3\,z-4)}{(z-1)^2}$$

$$(\%o51)\quad \frac{1}{(z-1)^2\,z^2}$$

```
(%i52) factor(tz(expand((n+2)^2)));
       factor(z^3*(z+1)/(z-1)^3-z);
```

$$(\%o52)\quad \frac{z\left(4\,z^2-3\,z+1\right)}{(z-1)^3}$$

$$(\%o53)\quad \frac{z\left(4\,z^2-3\,z+1\right)}{(z-1)^3}$$

```
(%i54) tiz(2*z/((z-1)*(z-3)));
```

$$(\%o54)\quad [0,3^n-1]$$

```
(%i55) tiz(3/(z-2));
```

$$(\%o55)\quad [0,3\,2^{n-1}]$$

```
(%i56) residue(3/(z*(z-2)),z,2);
```

$$(\%o56)\quad \frac{3}{2}$$

```
(%i57) tiz(z/(z-1));
```

(%o57) $[1, 1]$

```
(%i58) tiz(1/(z-1));
```

(%o58) $[0, 1]$

```
(%i59) tiz(1/(z^5*(z-1)));
```

(%o59) $[0, 0, 0, 0, 0, 0, 1]$

```
(%i60) residue(1/(z*(z-1)),z,1);
```

(%o60) 1

```
(%i61) tiz((2*z^2)/(11*z^2-21*z+10));
```

(%o61) $[\frac{2}{11}, 2 - 2\,11^{-n-1}\,10^{n+1}]$

14.6 Résolution d'équations récurrentes avec la transformée en z

```
(%i62) kill(x, y);
```

(%o62) *done*

```
(%i63) equation : 11*y(n)-10*y(n-1)*unit_step(n-1) = 2*unit_step(n);
```

(%o63) $11\,\mathrm{y}\,(n) - 10\,\mathrm{unit\_step}\,(n-1)\,\mathrm{y}\,(n-1) = 2\,\mathrm{unit\_step}\,(n)$

```
(%i64) equationtransformee:tz(equation);
```

(%o64) $11\,\mathrm{z\_transform}\,(\mathrm{y}\,(n), n, z) - \frac{10\,\mathrm{z\_transform}\,(\mathrm{y}\,(n), n, z)}{z} = \frac{2\,z}{z-1}$

```
(%i65) solve(equationtransformee, ''tz(y(n)))[1];
```

(%o65) $\mathrm{z\_transform}\,(\mathrm{y}\,(n), n, z) = \frac{2\,z^2}{11\,z^2 - 21\,z + 10}$

```
(%i66) tiz(rhs(%));
```

(%o66) $[\frac{2}{11}, 2 - 2\,11^{-n-1}\,10^{n+1}]$