WxMaxima

1. DE QUOI S'AGIT-IL ?

WxMaxima est un logiciel de calcul **formel**, c'est à dire qu'il est capable de faire du calcul littéral, et pas seulement numérique. On peut donc résoudre des équations, même celles ayant des coefficients qui ne sont pas numériques, on peut factoriser et développer des expressions, calculer une dérivée ou une primitive d'une fonction. Et bien sûr, plein d'autres choses sont faisables, mais dans cette introduction à l'utilisation de ce logiciel, on s'en tiendra là.

Ce logiciel est assez retord à manipuler, on va donc épurer au maximum. L'utilité, pour des élèves de lycée, sera non pas de faire le travail à leur place, mais de donner le résultat des problèmes demandés. Ceci a un double but : vérifier que le travail qu'on a fait est juste, et sinon, si on « sèche », la réponse du logiciel peut donner une piste de solution.

Ce n'est donc absolument pas l'outil miracle qui fait le devoir à la place de l'élève, loin de là, d'autant plus que parfois, il faut arranger le résultat donné par la machine (donc comprendre ce qu'on fait !), et que dans tous les cas, dans un devoir, ce n'est pas tant le résultat qui compte que la manière d'y arriver, la démarche, qui doit être explicitée dans la copie ; et ça, WxMaxima ne vous le fera pas !

Sinon, bien entendu, c'est un merveilleux outil de conjecture : quand on flaire une piste dans un raisonnement, si cette piste demande beaucoup de calculs pour être explorée, on peut la valider (ou non) très vite et éviter de perdre du temps inutilement.

2. PREMIERS CONTACTS

2.1. POUR QUELS ORDINATEURS ?

Tous ! WxMaxima existe sur les plateformes Windows, Linux et MacOsX. C'est un programme relativement léger qui tourne bien sur toutes les machines, même sur les petites configurations (par exemple sur mon eeepc 701).

On trouvera le logiciel à l'adresse suivante :

http://andrejv.github.com/wxmaxima/

C'est le site de l'auteur : on y trouvera bien sûr la dernière version à jour. On peut aller voir aussi le site : http://michel.gosse.free.fr/ pour de la documentation et des exemples (le site n'est plus maintenu, il ne faut pas y télécharger le logiciel). Dans la doc, on téléchargera impérativement « aide mémoire pour Maxima » de Vincent Obaton.

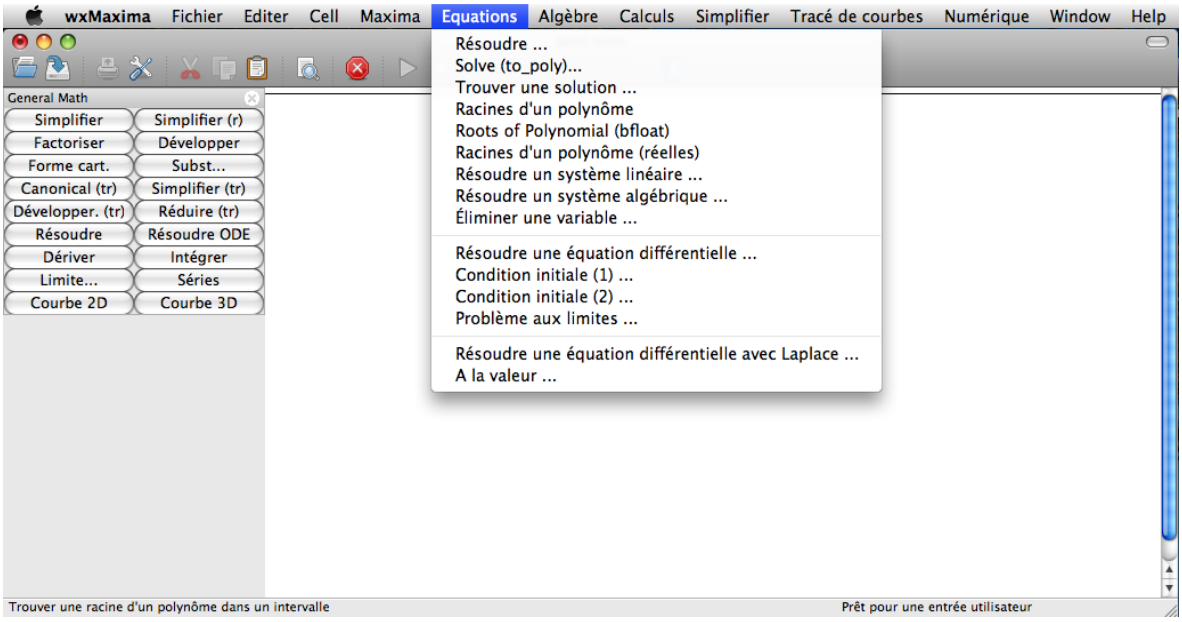
A savoir : le programme est en fait constitué de deux parties :

- Maxima, qui est le moteur de calcul. On peut l'utiliser seul, mais en mode « console », à savoir une fenêtre de texte où on tape les commandes : un peu brut pour des débutants…
- WxMaxima est une interface graphique qui enjolive Maxima, comporte des boutons et menus pour appliquer des commandes simples sans connaître la syntaxe, et donne les résultats sous forme Wysiwyg.

Pratiquement, l'utilisateur ne voit que WxMaxima (et c'est tant mieux !)

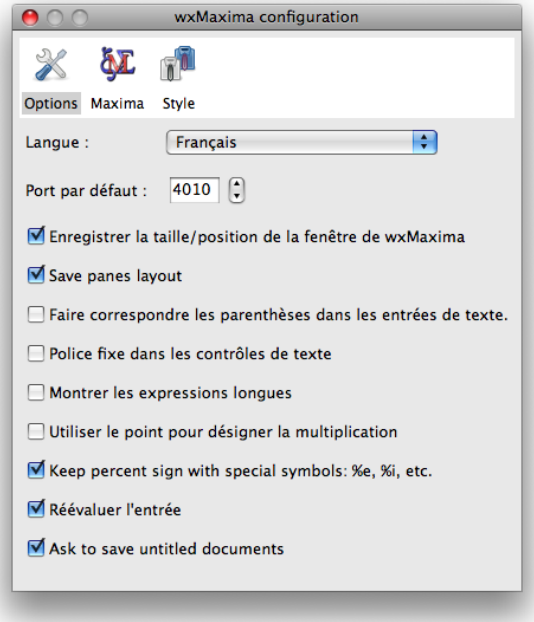
2.2.L'INTERFACE DU LOGICIEL, CONFIGURATION

On lance WxMaxima simplement en double-cliquant son icône, et on tombe sur une fenêtre qui va ressembler à ça :



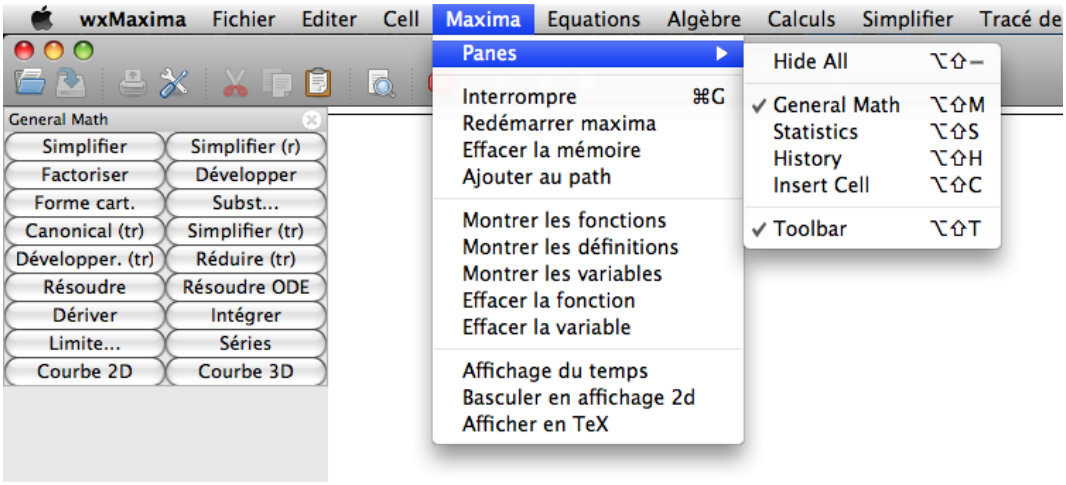
On note la barre d'outils en haut, des boutons permettant d'effectuer des opérations mathématiques (ici à gauche, mais vous les mettez où vous voulez), et une feuille blanche : l'espace de travail où vous aller rentrer vos commandes. Les menus sont assez fournis et pas toujours bien traduits, mais au niveau où vous êtes, l'anglais ne vous pose plus de problèmes, isn't it ?

Avant toutes choses, il faut configurer le logiciel : cliquez sur le bouton représentant un tournevis et une clef dans la barre d'outils. On obtient la fenêtre suivante, que vous allez reproduire comme indiqué ici (ne changez pas le port par défaut, laissez le comme il est sur votre machine) :



Vous pouvez ensuite mettre en place ou escamoter les barres de boutons. Je vous conseille de laisser la barre « general maths » et d'enlever les autres, pas trop utiles pour ce qu'on va faire.

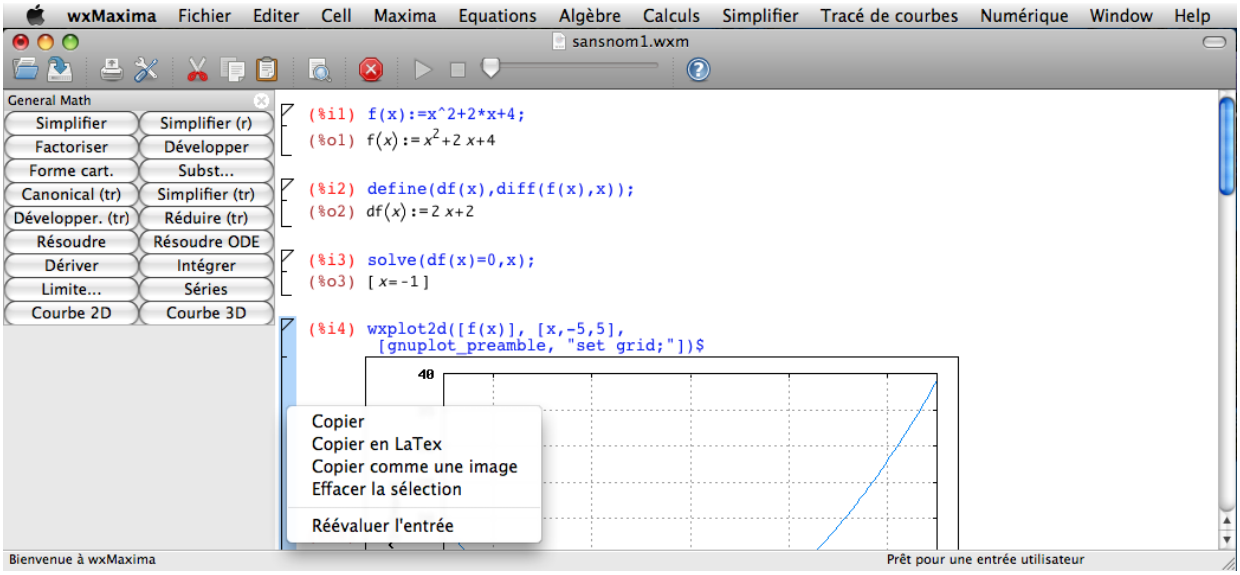
Il suffit d'aller dans le menu « Maxima → panes » :



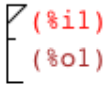
On note aussi qu'on peut escamoter la barre d'outils (Toolbar) : elle n'est pas franchement utile.

2.3. UTILISATION DE BASE

Pour initier un calcul, il suffit de cliquer dans la fenêtre, de taper sa formule (ici on définit une fonction *f* et une suite de calculs) et d'appuyer sur la touche entrée pour la valider.



WxMaxima note (%i1) l'entrée 1 (« i » comme input, entrée en anglais), et (%o1) la sortie 1 (« o » comme output). Chaque commande, avec son entrée (ce qu'on tape) et sa sortie (réponse Maxima) est délimitée par des crochets :



Si on clique sur le petit triangle du haut, WxMaxima escamote la réponse, si on clique dans la partie inférieur du crochet, tout le crochet se grise, et on peut faire un clic droit et voir apparaître un menu, ou bien appuyer sur la touche « retour arrière » pour effacer cette entrée, si on a fait n'importe quoi.

Si on se trompe dans une entrée, il suffit de double-cliquer dedans, de rectifier l'erreur et d'appuyer sur la touche « entrée » pour valider. La sortie est modifiée en conséquence.

Si on a oublié une étape dans un processus de calcul, il faut cliquer au dessus de la ligne où on veut insérer une commande (un trait horizontal apparaît à cet endroit), on tape sa commande et on appuie sur « entrée ». Par contre, il va falloir recalculer tout l'ensemble de la feuille de calcul. Pour ceci, on sélectionne tout (ctrl+A ou pomme+A), les crochets se grisent, on fait un clic droit sur un crochet grisé et dans le menu, on sélectionne « réévaluer l'entrée » : toute la feuille de calcul se remet à jour en tenant compte des modifications (voir figure page précédente).

Il y a aussi une technique plus simple pour tout recalculer : faire ctrl+R ou pomme+R (Mac).

Nous verrons au paragraphe « bugs » quelques soucis qu'on peut rencontrer avec WxMaxima. Maintenant, place aux exemples.

3. EXEMPLES

3.1. MANIPULATION D'EXPRESSIONS

Voyons un exemple de base :

```
(%i1)  A:(2*x+3)**2-5*(x-2)*(2*x+3);
(%o1)  (2 x+3)^2-5(x-2)(2 x+3)

(%i2)  ratsimp(A);
(%o2)  -6 x^2+17 x+39

(%i3)  factor(A);
(%o3)  -(2 x+3)(3 x-13)
```

On note plusieurs choses :

- On ne met pas « A = … ». On utilise l'assignation « : ». Pour Maxima, « = » correspond à l'égalité mathématique. Le « : » correspond à « A prend la valeur… ».
- Il faut mettre tous les opérateurs, notamment les multiplications (« * », et pas « x »), contrairement à d'autres logiciels qui utilisent les notations implicites (ex : GeoGebra, les calculettes graphiques).
- Pour les puissances, taper ^ suivi d'un espace avant de taper l'exposant. Si cette frappe marche mal (comme sur certaines versions Mac), taper ** au lieu de ^.
- Une fois qu'on a rentré notre expression, on peut la développer / réduire avec la commande « ratsimp », ou plus simplement, en cliquant sur le bouton « simplifier ».
- On peut aussi aisément la factoriser avec la commande « factor » ou le bouton « factoriser »
- Vous pourriez aussi rentrer une expression B, et l'additionner, la multiplier ou je ne sais quoi d'autre avec l'expression A.

Voilà, maintenant, vous êtes capables d'épater le petit frère ou la petite sœur qui galère en 3e !

3.2.ÉTUDE DE FONCTION (DÉBUTANT)

C'est le morceau le plus important de ce document, car c'est ce qui servira le plus à un lycéen.

On va donner l'exemple exécuté dans WxMaxima et ensuite le commenter. Faites très attention à la syntaxe utilisée.

```
(%i1) f(x):=(-x**2-2*x-1)/(x-3);
```

$$(\%o1)\quad f(x):=\frac{-x^2-2x-1}{x-3}$$

```
(%i2) factor(f(x));
```

$$(\%o2)\quad -\frac{(x+1)^2}{x-3}$$

```
(%i3) define(df(x),diff(f(x),x));
```

$$(\%o3)\quad df(x):=\frac{-2x-2}{x-3}-\frac{-x^2-2x-1}{(x-3)^2}$$

```
(%i4) ratsimp(df(x));
```

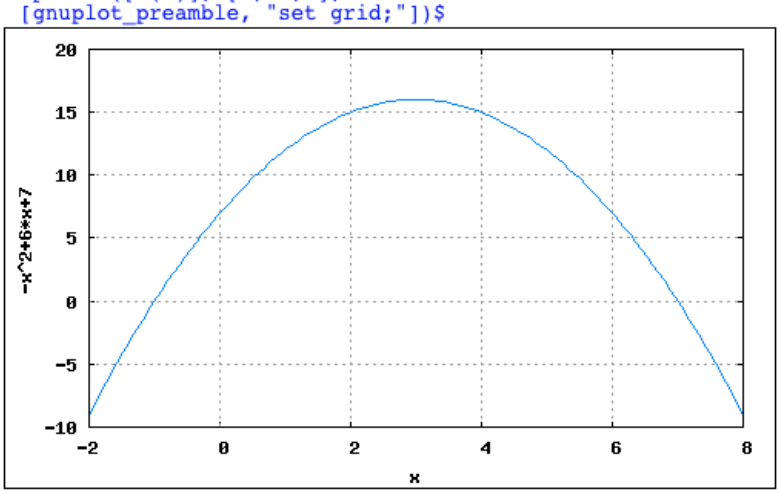
$$(\%o4)\quad -\frac{x^2-6x-7}{x^2-6x+9}$$

```
(%i5) factor(%o4);
```

$$(\%o5)\quad -\frac{(x-7)(x+1)}{(x-3)^2}$$

```
(%i6) n(x):=num(%o4);
(%o6) n(x):=num(%o4)
(%i7) n(x);
(%o7) -x^2+6*x+7
(%i8) wxplot2d([n(x)], [x,-2,8],
  [gnuplot_preamble, "set grid;"])$
```

(%t8)




```
(%i9) solve(n(x)=0,x);
(%o9) [x=7,x=-1]
```

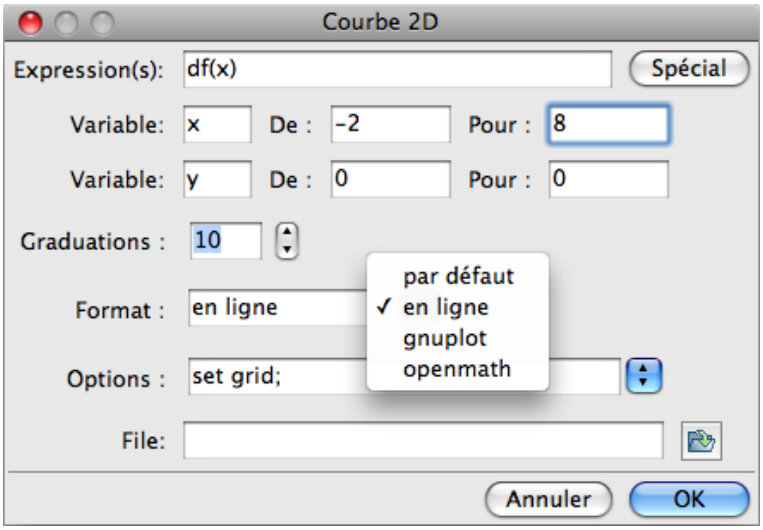
Commentaires :

- Définition de la fonction : il faut utiliser une assignation un peu bizarre : « f(x) := … ». Si on ne fait pas ça, la fonction n'est pas reconnue comme telle. Soyez très vigilants !
- On peut ensuite factoriser cette fonction pour le fun (commande %i2). Si rien ne bouge, c'est que ce n'est pas factorisable.

- La commande %i3 est très importante : elle permet de définir la dérivée de notre fonction. On utilisera impérativement la commande « define » (voir §4.2) pour définir cette dérivée. df(x) est le nom donné à notre dérivée, et la commande « diff(f(x),x) » indique qu'il faut calculer la dérivée de la fonction f(x) relativement à la variable x (Maxima sait traiter des fonctions de plusieurs variables, l'horreur totale !). Si vous devez dériver une fonction C(q), vous écrirez « diff(C(q),q) », tout simplement.
- %i4 permet d'avoir une vision plus « civilisée » de notre dérivée. A noter qu'on aurait pu intégrer cette commande « ratsimp » dans la commande précédente, ce qui aurait donné :

define(df(x), ratsimp(diff(f(x),x)))

- En %i5, une petite factorisation nous donne de précieuses indications sur cette dérivée. A noter qu'on fait référence au résultat %o4 pour appliquer la commande.
- Ici, seul le numérateur de la dérivée nous sera utile : on peut l'isoler avec la commande « num », car il peut être plus pratique de travailler ensuite uniquement sur ce numérateur. La commande %i6 permet ceci. %i7 permet d'afficher le résultat.
- %i8 est une commande de graphique, qu'on ne tapera pas bêtement comme ça : on utilisera le bouton « courbe 2D » et on remplira la boite de dialogue comme suit :

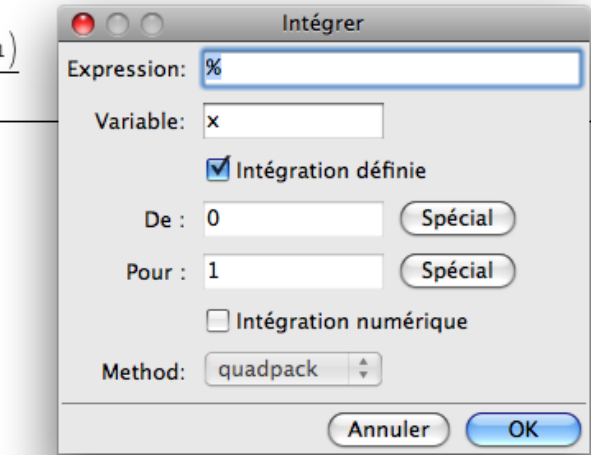


- Utiliser le format « en ligne » pour avoir le graphique dans la feuille de calcul, sinon choisir « gnuplot » (on obtiendra une fenêtre annexe). L'option « set grid » permet d'avoir un quadrillage. On peut tracer plusieurs courbes sur le même graphique : dans la case « expressions », taper les expressions désirées en les séparant par une virgule.
  Ex : « df(x),f(x),n(x) » va tracer ces 3 courbes.
- La commande %i9 quant à elle va permettre de résoudre l'équation « dérivée = 0 ». On n'utilise ici que le numérateur, mais la commande aurait marché avec df(x). La commande est « solve », et comme pour la dérivée, il faut préciser quelle est la variable qu'on cherche. On obtient donc les valeurs qui annulent la dérivée. A noter qu'ici, cette commande est redondante avec celle qui permet de factoriser la dérivée, car dans la forme factorisée, on lit directement les solution de l'équation df(x) = 0 (voir %o5). Mais bon, vu les efforts à déployer dans les deux cas, on peut se le permettre, non ?

3.3.ÉTUDE DE FONCTION (AVANCÉ)

On va aussi pouvoir calculer des limites et intégrer nos fonctions (chic !). Voyons ça sur un autre exemple :

```
(%i1)  f(x):=log(x+1)+exp(2*x-1);
(%o1)  f(x):=log(x+1)+exp(2*x-1)

(%i2)  f(x);
(%o2)  log(x+1)+%e^(2*x-1)

(%i10) define(F(x),integrate(f(x), x));
(%o10) F(x):=(x+1)log(x+1)+%e^(2*x-1)/2-x-1

(%i14) integrate(f(x), x, 0, 1);
(%o14) (%e^-1(4 %e log(2)+%e^2-2 %e-1))/2

(%i12) limit(f(x), x, -1, plus);
(%o12) -∞

(%i13) limit(f(x), x, inf);
(%o13) ∞
```



- On commence par définir notre fonction. A noter que la fonction « ln » s'appelle ici « log ». Désolé pour ce contretemps, mais c'est monnaie courante dans les logiciels de calcul et de programmation. Quand on fait afficher notre fonction (commande %i2), on note que l'exponentielle est précédée de % : « %e ». C'est pareil pour π « %pi », et « %i » pour les complexes. On se reportera à l'aide mémoire cité au §2.1 pour les autres fonctions usuelles.
- Pour intégrer, le plus simple est d'aller dans le menu « calculs > integrate », et on obtient la fenêtre ci-dessus. Il faut renseigner l'expression (ici, le % indique que c'est le dernier résultat calculé), la variable, et les bornes si on veut une intégrale définie entre bornes. On cochera aussi dans ce cas la case « intégrale définie ». Si on veut seulement une primitive, ne pas cocher cette case, c'est tout !
  La commande %i10 a été obtenue avec cette fenêtre, et on l'a ensuite éditée et « habillée » avec la commande « define » pour obtenir une « vraie » primitive. On note que la syntaxe est la même que pour la dérivée, en remplaçant « diff » par « integrate ».
  La commande %i14 est une intégrale entre les bornes 0 et 1.
- Les commandes %i12 et %i13 donnent des limites de notre fonction. Idem, on les obtient à partir du menu « calculs > limit » et on remplit la fenêtre. Le bouton « special » donne accès aux bornes infinies, et on peut demander la limite à droite ou à gauche (ex de %i12, où on spécifie « plus », qui donne la limite à droite).

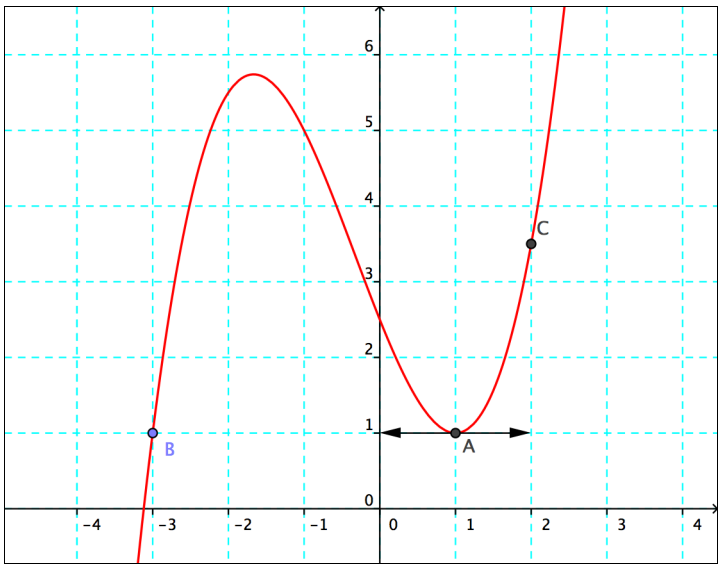
3.4. DÉTERMINER UNE FONCTION AVEC PARAMÈTRES

Un exercice fréquent consiste à donner l'énoncé d'une fonction dépendant de plusieurs paramètres, de donner des infos graphiques permettant de calculer des images et des nombres dérivés et d'en faire des équations permettant de calculer nos paramètres.

Ex : soit la fonction *f* définie de la manière suivante :

$$f(x) = a\,x^3 + b\,x^2 + c\,x + d$$

D'après le graphique suivant, déterminez les images de -3, 1 et 2 par *f*, ainsi que le nombre dérivé en 1. Déduisez-en les coefficients *a*, *b*, *c et d*.



Voyons ce que ça donne avec WxMaxima. (on note qu'il faut savoir lire le graphique…)

```
(%i1)  f(x):=a*x**3+b*x**2+c*x+d;
(%o1)  f(x):= a x^3 + b x^2 + c x + d

(%i2)  define(df(x),diff(f(x),x));
(%o2)  df(x):= 3 a x^2 + 2 b x + c

(%i3)  e1:f(-3)=1;
(%o3)  d - 3 c + 9 b - 27 a = 1

(%i4)  e2:f(1)=1;
(%o4)  d + c + b + a = 1

(%i5)  e3:df(1)=0;
(%o5)  c + 2 b + 3 a = 0

(%i6)  e4: f(2)=7/2;
(%o6)  d + 2 c + 4 b + 8 a = 7/2

(%i7)  solve([e1,e2,e3,e4],[a,b,c,d]);
(%o7)  [ [ a = 1/2, b = 1/2, c = -5/2, d = 5/2 ] ]
```

- Les lignes %i1 et %i2 sont classiques : on définit la fonction et sa dérivée. Notez que le fait que la fonction dépende de paramètres non numériques ne gêne absolument pas le logiciel. On répète ici : la dérivée est **obligatoirement** définie avec la fonction « define », sinon la suite ne marche pas.
- Les lignes %i3 à %i6 définissent 4 équations nommées e1 à e4. On notera la syntaxe très spéciale : « e1:f(-3)=1 ». On a d'abord une assignation avec « : », et ensuite une égalité. Ceci montre la nécessité de l'assignation : on remplit la variable e1 avec une égalité !
- On note qu'à chaque fois, la réponse %o3 à %o6 donne l'équation correspondante.
- L'entrée %i7 a déjà été vue aussi, mais sous une forme plus simple. C'est la commande « solve », mais ici, on voit la syntaxe avec plusieurs équations à plusieurs inconnues. On obtient la solution à notre problème instantanément.

C'est plus rapide qu'à la main, non ?

3.5. RÉSOLUTION D'ÉQUATIONS

```
(%i1) e1:x-2*y=3;
(%o1) x-2 y=3

(%i2) e2:-3*x+4*y=-1;
(%o2) 4 y-3 x=-1

(%i3) solve([e1,e2],[x,y]);
(%o3) [[x=-5,y=-4]]

(%i4) e3:log(x+2)=3;
(%o4) log(x+2)=3

(%i5) solve(e3,x);
(%o5) [x=%e^3-2]

(%i6) e4:log(exp(x+2)-3)=4;
(%o6) log(%e^(x+2)-3)=4

(%i7) solve(e4,x);
(%o7) []

(%i8) find_root(e4,x,1,10);
(%o8) 2.053490449705934
```

- Les deux premières commandes définissent un système de deux équations à deux inconnues qu'on résoud avec la troisième commande.
- %i4 et %i5 montrent qu'on peut résoudre aussi des équations avec des logarithmes (penser à mettre « log » et pas « ln »…)
- Par contre, l'équation e4 met la commande « solve » en échec. Dans ce cas, c'est que généralement, on ne peut pas trouver une solution algébrique au problème, on ne peut trouver qu'une valeur numérique approchée. On utilise alors la commande « find_root ». Il faut alors préciser deux bornes entre lesquelles on est susceptible de trouver la solution (ici, ce sont les valeurs 1 et 10).

4. « BUGS » ET ASTUCES DE SYNTAXE

4.1. ASSIGNATION

On fera très attention à la syntaxe d'assignation. Ne pas utiliser le « = » mais le « : » pour une variable ou une expression, ou le « := » pour définir une fonction. Si une commande ne marche pas, il est très probable qu'on a mal écrit une commande d'assignation un peu plus haut dans les calculs.

4.2. DÉFINITION D'UNE DÉRIVÉE

On peut faire plus simple que ce qu'on a vu au §3.2 pour définir la dérivée, et c'est notamment ce que fera le menu « dériver » :

```
(%i1)  f(x):=log(x**2+4);
(%o1)  f(x):=log(x^2+4)

(%i2)  df(x):=diff(f(x),x);
(%o2)  df(x):=diff(f(x),x)

(%i3)  df(x);
(%o3)  2*x/(x^2+4)

(%i4)  df(3);
diff : second argument must be a variable; found  3
#0: df(x=3)
 -- an error.  To debug this try debugmode(true);
```

Le problème, c'est que cette commande (%i2) nous donne bien la dérivée, mais celle-ci est inutilisable dans les calculs (ex de %i4), ce qui est dommage. On utilisera donc la commande « define » pour calculer la dérivée, on obtiendra ainsi une « vraie » dérivée, utilisable comme une fonction dans des calculs.

4.3. SIMPLIFICATION D'EXPRESSIONS

Il faut user et abuser de la commande « ratsimp » pour simplifier des expressions qui semblent compliquées. Vous pouvez aussi cliquer sur le bouton « simplifier », ça tape la commande à votre place. Attention dans ce cas, WxMaxima applique la simplification sur le dernier résultat affiché. Des fois, ça ne marche pas. Explications :

```
(%i5)  f(x):=(-x**2-2*x-1)/(x-3);
(%o5)  f(x):=(-x^2-2*x-1)/(x-3)

(%i6)  df(x):=diff(f(x),x);
(%o6)  df(x):=diff(f(x),x)

(%i7)  ratsimp(%);
(%o7)  df(x):=diff(f(x),x)

(%i8)  df(x);
(%o8)  (-2*x-2)/(x-3)-(-x^2-2*x-1)/(x-3)^2

(%i9)  ratsimp(%);
(%o9)  -(x^2-6*x-7)/(x^2-6*x+9)
```

Le « ratsimp » de la ligne %i7 s'applique à la sortie %o6 qui n'est pas une expression simplifiable. Il faut d'abord afficher df(x) (%i8), et ensuite appliquer la commande « ratsimp ».