Travaux dirigés avec SAGE (partie I)

Math 2 — Année 2010-2011

Sommaire

1 Prise en main

Connectez-vous à l'adresse `http://sage-math.univ-lyon1.fr`*. Après vous être identifié, cliquez sur le lien "New Worksheet" : une feuille de calcul vierge apparaît, que vous nommerez par exemple "TD1". En fin de séance, n'oubliez pas de sauvegarder votre travail.*

1.1 Feuille de calcul

Elle se présente sous la forme de cellules dans lesquelles vous taperez vos commandes. Entrez par exemple l'expression :

```
cos(pi/12)
```

puis validez par les touches `<MAJ><ENTREE>`.
La valeur de l'expression apparaît sous forme littérale (`sqrt` désigne la racine carrée). Pour avoir une approximation numérique avec 10 chiffres décimaux, entrez dans la cellule suivante :

```
cos(pi/12).n(digits=10)
```

puis validez.
A tout moment, on peut :
– insérer une nouvelle cellule : placer la souris entre deux cellules et cliquer sur la ligne bleue qui apparaît,
– supprimer une cellule : effacer d'abord son contenu puis presser la touche <DELETE>.
On peut aussi valider une cellule par <CTRL><ENTREE>, ce qui a pour effet d'insérer une cellule vide juste en-dessous.
Les boutons du haut proposent certaines actions, comme "Interrupt" pour interrompre le calcul en cours, "Restart" pour effacer de la mémoire tous les calculs effectués, "Save" pour enregistrer la feuille.

1.2 Assignation

Pour des calculs un peu longs, il est pratique de nommer les expressions intermédiaires par l'instruction d'*assignation*, de la forme :

`nom = expression`

Dans une nouvelle cellule, tapez (puis validez) :

```
u = sqrt(5)+sqrt(3) ; v = sqrt(5)-sqrt(3)
y = u/v + v/u ; y
```

Notez au passage que dans une cellule :
– on peut saisir plusieurs lignes,
– sur chaque ligne, on peut saisir plusieurs instructions séparées par des points-virgules,
– le résultat affiché est celui de la dernière ligne, ici la valeur de `y`.

Obtenez un affichage plus sympathique de `y` avec la commande :

```
y.show()
```

Pour SAGE, les expressions mathématiques sont des *objets* au sens informatique du terme. Tout objet a un *type* bien défini (par exemple Integer, Rational etc.), des *attributs* (i.e. données internes) et des *méthodes* (i.e. fonctions qui peuvent lui être appliquées). Pour faire appel à une méthode d'un objet, la syntaxe est de la forme :

`objet.methode(`*paramètres éventuels*`)`

Par exemple, ci-dessus on a fait appel à la méthode `show()` de l'objet `y`.

Remarque.— Certaines méthodes peuvent également être appelées sous une forme plus classique : `methode(objet,` *paramètres éventuels*`)`. Dans l'exemple précédent, la commande `show(y)` a le même effet que `y.show()`.

Pour afficher le type de `y` et l'ensemble auquel il appartient, vous pouvez entrer la commande :

```
type(y) ; parent(y)
```

1.3 Aide en ligne

SAGE contient de nombreuses fonctions et méthodes prédéfinies, qu'on peut trouver grâce à l'aide intégrée au logiciel.

Cherchons à simplifier l'expression de `y`. Dans une cellule, tapez :

```
y.simp
```

suivi de la touche <TAB> ce qui fait apparaître la liste des méthodes applicables à `y` et commençant par `simp`. Cliquez sur la méthode `simplify_radical`, complétez la cellule par un couple de parenthèses et validez :

```
y.simplify_radical()
```

Le résultat est 8.

Auto-complétion des mots : comme vous l'avez vu, en tapant le début d'une commande suivie de la touche `<TAB>`, on fait apparaître les méthodes correspondantes. Si vous tapez :

```
y.
```

suivi de la touche `<TAB>`, toutes les méthodes applicables à `y` sont proposées (sauf si la liste est trop longue).

Aide sur une méthode : utilisez le point d'interrogation. Par exemple, tapez :

```
y.simplify_radical?
```

puis la touche `<TAB>` pour faire apparaître la documentation sur `simplify_radical`, avec des exemples d'utilisation.

1.4 Expressions symboliques

SAGE permet de faire du *calcul formel*, c'est-à-dire manipuler des expressions mathématiques contenant des symboles.
Essayez cet exemple (le signe `#` indique le début d'un commentaire ; pour la suite de ce TD, il sera inutile de taper les commentaires) :

```
var('x,y')    # déclare deux variables symboliques x et y
z = sin(x^2/y) ; z
```

Pour remplacer `x` et `y` dans `z`, on utilise la syntaxe de *substitution* :

```
z(x=-1,y=2)
```

ou la méthode `subs_expr` :

```
z.subs_expr(x==-1,y==2)
```

On peut aussi définir une fonction `f` :

```
var('x,y')
f(x,y) = sin(x^2/y)
f
```

et calculer ensuite :

```
f(-1,2)
```

Remarque.— Ne pas confondre l'expression `z` et la fonction `f` :
- à partir de l'expression `z`, il serait possible d'obtenir la fonction correspondante `f` par l'instruction :

```
f = z.function(x,y)
```

- inversement, à partir de la fonction `f` on obtient l'expression `z` en écrivant tout simplement :

```
z = f(x,y)
```

1.5 Types composés

1.5.1 Définitions

SAGE utilise le langage de programmation Python, qui lui-même possède trois types de données composés qui vous seront utiles :

1°) *n-uplet* : plusieurs expressions séparées par des virgules, et entourées d'un couple de parenthèses (en général facultatives), par exemple :

```
v = (x,y,z) ; v
```

2°) *liste* : se note avec des crochets, par exemple :

```
L = [1,4,2,8,5,7] ; L
```

Faites afficher le type de `v` et de `L`.

L'accès à un élément d'un n-uplet ou d'une liste se fait en précisant son indice (numéroté à partir de 0). Evaluez par exemple `L[0]`. Le nombre d'éléments de L s'obtient avec `len(L)`. On peut modifier les éléments d'une liste, mais pas ceux d'un n-uplet. Essayez par exemple :

```
L[0] = -1 ; L
```

et vérifiez que la même instruction appliquée à `v` provoque une erreur.

3°) *dictionnaire* : c'est un ensemble de couples de la forme `clef:valeur`, par exemple :

```
d = {4 : 8, 1 : 25}
```

L'accès à un élément se fait en précisant la clef entre crochet. Evaluez par exemple `d[4]`.

Que renvoie la commande suivante ?

```
var('x,y') ; z = sin(x^2/y)
dico = {x : -1, y : 2}
z(dico)
```

(les dictionnaires sont une autre manière d'opérer des substitutions).

1.5.2 Construction des listes

Une liste d'entiers successifs s'obtient avec la fonction `range`. Testez les commandes suivantes :

```
range(10) ; range(3,13) ; range(3,13,2)
```

```
[3..12]
```

(la construction `[a..b]` est équivalente à `range(a,b+1)`).

On peut aussi construire des listes par *compréhension* :

```
[1/n^2 for n in range(1,11)]
```

Remarques :
– Les n-uplets, listes et dictionnaires peuvent contenir des objets de tous types. Par exemple :

```
M = [1.0, x, 'math 2', range(4)] ; M
```

– Certaines commandes prennent des listes (ou bien des n-uplets) en paramètres : c'est par exemple le cas de `solve` (voir §2) ou de `plot3d` (voir §5.1).

1.6 Programmation

Il est possible de programmer ses propres fonctions et utiliser des tests et des boucles. La syntaxe utilisée est celle du langage Python, avec ses structures de contrôle `if, for, while` et certains mots-clefs.

Voici juste un exemple : programmez la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par $f(x) = x$ si $x \in [-1, 1]$, $f(x) = \frac{1}{x}$ sinon. Solution :

```
def f(x) :
     if  -1 <= x <= 1 :
          return x
     else :
          return 1/x
```

Notez la présence des deux-points, ainsi que l'indentation des lignes qui doit être rigoureuse car elle délimite chaque bloc d'instructions.
Après avoir défini cette fonction, tracez son graphe sur l'intervalle $[-4, 4]$ par la commande `plot(f,-4,4)`.

On teste si deux objets sont égaux (resp. différents) avec l'opérateur `==` (resp. `!=`). Essayez :

```
f(0) == 0 ; f(1) != 1
```

2 Exercice résolu

Soit à calculer la valeur de $z = x^5 + y^5$ sachant que les nombres x, y vérifient $x + y = 1$ et $xy = -3$.

On commence par mettre en équations le problème :

```
var('x,y')                  # déclaration des symboles x et y
z = x^5+y^5                 # expression formelle de z
eqs = [x+y==1, x*y==-3]     # liste d'équations
vars = [x,y]                # liste d'inconnues
```

puis on résout les équations avec la commande `solve` :

```
s = solve(eqs,vars) ; s
```

qui rend les solutions sous forme d'une liste de listes, ou bien par :

```
s = solve(eqs,vars,solution_dict=True) ; s
```

qui rend les solutions sous forme d'une liste de dictionnaires. Cette deuxième manière est plus facile à exploiter puisqu'on obtient directement la valeur de z par substitution (voir §1.5.1) :

```
z(s[0])      # valeur de z pour le premier couple de solutions
```

Simplifiez le résultat (on trouve 61). Calculez de même z avec le second couple de solutions.

3 Vecteurs et matrices

3.1 Construction, opérations élémentaires

Un vecteur de $\mathbb{R}^n$ est défini en donnant la liste de ses composantes. Entrez par exemple :

```
u = vector([1,3,2]) ; v = vector([2,5,4])
u ; v ; 2*u-v
```

Notez que SAGE affiche les vecteurs horizontalement. Calculez le produit scalaire $\vec{u} \cdot \vec{v}$ et le produit vectoriel $\vec{u} \wedge \vec{v}$:

```
u.dot_product(v) ; u.cross_product(v)
```

ainsi que la norme euclidienne de $\vec{u}$:

```
u.norm(2)
```

On accède aux composantes d'un vecteur en précisant l'indice entre crochets. ATTENTION : les indices sont numérotés à partir de 0 :

```
u[0] ; u[1] ; u[2]
```

La commande `list(u)` donne la liste des composantes de $\vec{u}$.

Une matrice à n lignes et p colonnes est définie en donnant ses dimensions et la liste de ses coefficients :

```
M = matrix(2,3,[1,3,2,2,5,4]) ; M; show(M) ; M.list()
```

On peut aussi définir M en donnant une liste de listes :

```
M = matrix([[1,3,2],[2,5,4]]) ; M
```

ou encore la liste de ses vecteurs-lignes :

```
M = matrix([u,v]) ; M
```

L'accès à un coefficient de la matrice s'obtient en précisant ses indices de ligne et de colonne entre crochets (numérotés à partir de 0) :

```
M[0,0]
```

Le produit matriciel se traite comme la multiplication ordinaire. Entrez par exemple :

```
A = matrix(3,3,[1/i for i in range(1,10)]) ; A
```

puis :

```
A*u ; A*A
```

ainsi que :

```
det(A) ; A^-1 ; A*A^-1 == identity_matrix(3)
```

3.2 Exercice

On considère les vecteurs de $\mathbb{R}^3$:

$$\vec{u} = \begin{pmatrix} a-b-c \\ 2a \\ 2a \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 2b \\ b-a-c \\ 2b \end{pmatrix} \quad \vec{w} = \begin{pmatrix} 2c \\ 2c \\ c-a-b \end{pmatrix}$$

où a, b, c sont trois nombres réels.

1. Calculer le produit mixte $(\vec{u}, \vec{v}, \vec{w})$ et simplifier son expression (utiliser la méthode `factor()`).
2. En déduire que le parallélépipède engendré par ces trois vecteurs a le même volume que le cube de côté $|a+b+c|$.

4 Fonctions d'une variable

Définissez la fonction $f(x) = \frac{1}{1+x^2}$ par la commande :

```
var('x')
f(x) = 1/(1+x^2)
```

Calculez $f(2)$ de deux façons (notez la différence d'affichage) :

```
f(2) ; f(2.0)
```

Démontrez que f est une fonction paire :

```
bool(f(x)==f(-x))
```

4.1 Limite, dérivée, intégrale, développement limité

Entrez les commandes suivantes :

```
limit(f(x),x=infinity)              # limite
```

```
diff(f(x),x) ; diff(f(x),x,2)       # dérivées première et seconde
```

```
integrate(f(x),x)                   # primitive
```

```
integrate(f(x),x,0,1)               # intégrale définie
```

```
taylor(f(x),x,0,6)                  # développement limité en 0, à l'ordre 6
```

4.2 Graphe

Calculez le graphe de f sur l'intervalle $[-4, 4]$:

```
g1 = plot(f(x),x,-4,4,color='blue') ; show(g1)
```

ainsi que celui de la droite tangente au graphe en $x = 2$:

```
t = taylor(f(x),x,2,1)
g2 = plot(t,x,-4,4,color='green')
```

puis tracez ces deux graphes sur un même dessin grâce à l'opérateur de superposition `+` :

```
show(g1+g2)
```

4.3 Exercices

1. Calculer $\lim_{x\to 0} \left(\frac{1}{\sin^2(x)} - \frac{1}{x^2}\right)$ et $\lim_{x\to+\infty} \sqrt{x+\sqrt{x}} - \sqrt{x}$.
2. Définir la fonction $f(x) = x^{\frac{1}{x}}$. Calculer $\lim_{x\to 0_+} f(x)$ par la commande :

   ```
   limit(f(x),x=0,dir='plus')
   ```

   Calculer $\lim_{x\to+\infty} f(x)$. Tracer le graphe de f sur l'intervalle $]0, 10]$. Calculer $f''(1)$.
3. Donner un développement limité de $\sin(\sinh(x)) - \sinh(\sin(x))$ en $x = 0$, à l'ordre 15.

5 Fonctions de deux variables

Définissez la fonction de deux variables $f(x, y) = (3x + 4y)\,\mathrm{e}^{-(x^2+y^2)}$ par la commande :

```
var('x,y')
f(x,y) = (3*x+4*y)*exp(-(x^2+y^2))
```

Pour SAGE, l'expression `f(x,y)` et la fonction `f` sont deux objets différents (voir remarque §1.4). Vérifiez-le en entrant la commande :

```
f(x,y) ; f
```

5.1 Graphe en 3D

Tracez le graphe de $f(x, y)$ pour $-2 \leqslant x \leqslant 2$ et $-2 \leqslant y \leqslant 2$:

```
G1 = plot3d(f(x,y),(x,-2,2),(y,-2,2)) ; show(G1)
```

En cliquant sur le dessin, vous pouvez faire tourner la surface et l'observer sous différents angles.

5.2 Dérivées partielles, gradient

Calculez les dérivées partielles $\frac{\partial f}{\partial x}(x, y)$, $\frac{\partial f}{\partial y}(x, y)$, et vérifiez que $\frac{\partial^2 f}{\partial x \partial y}(x, y) = \frac{\partial^2 f}{\partial y \partial x}(x, y)$:

```
diff(f(x,y),x) ; diff(f(x,y),y) ; diff(f(x,y),x,y)-diff(f(x,y),y,x)
```

Vous pouvez obtenir le vecteur gradient $\overrightarrow{\nabla} f(x, y)$ avec la commande :

```
df = f(x,y).gradient() ; df
```

5.3 Courbes de niveau

Tracez la courbe de niveau définie par $f(x, y) = 1$:

```
implicit_plot(f(x,y)==1,(x,-2,2),(y,-2,2))
```

Il est possible de tracer plusieurs courbes de niveaux, par exemple 20, avec la commande :

```
g1 = contour_plot(f(x,y),(x,-2,2),(y,-2,2),contours=20,fill=False,cmap='spectral')
show(g1)
```

Tracez aussi le champ de gradient par la commande :

```
g2 = plot_vector_field(df,(x,-2,2),(y,-2,2))
show(g2)
```

puis les deux dessins superposés, dans un repère orthonormé :

```
show(g1+g2,aspect_ratio=1)
```

Que constatez-vous ?

5.4 Points critiques

Pour trouver les points critiques, entrez la commande :

```
s = solve([df[0]==0,df[1]==0],[x,y],solution_dict=True) ; s
```

Il y a donc deux points critiques qui sont :

```
[(s[i][x],s[i][y]) for i in range(2)]
```

Pour étudier leur nature, calculez la matrice hessienne de f en (x, y) :

```
H = f(x,y).hessian() ; H
```

que l'on peut simplifier un peu :

```
H = H.factor() ; show(H)
```

Déterminons la nature du premier point critique. Pour cela, entrez :

```
H1 = H(s[0]) ; det(H1) ; H1[0,0]
```

Qu'en déduisez-vous ? Etudiez de la même façon la nature du second point critique.

5.5 Exercices

1. Calculer le développement de taylor au point $(1, 1)$, à l'ordre 2, de la fonction f précédente.
2. Trouver l'équation du plan tangent au graphe de f au point $(1, \frac{1}{5}, f(1, \frac{1}{5}))$ et le tracer en superposition avec le graphe de f (utiliser `implicit_plot3d`).
3. On rappelle que l'aire d'un triangle de côtés x, y, z et de demi-périmètre $p = \frac{1}{2}(x+y+z)$ est donnée par :
$$A = \sqrt{p(p-x)(p-y)(p-z)}$$
   (a) On suppose que $p = \frac{3}{2}$. Calculer z en fonction de x et y.
   (b) En déduire l'expression de A en fonction de x et y.
   (c) Tracer le graphe de A pour $0 \leqslant x \leqslant \frac{3}{2}$ et $0 \leqslant y \leqslant \frac{3}{2}$.
   (d) Tracer quelques courbes de niveau de A.
   (e) Montrer que $(1, 1)$ est point critique de A et que c'est un maximum.
   (f) En déduire qu'à périmètre donné, le triangle ayant la plus grande aire est le triangle équilatéral.
4. En procédant comme dans l'exercice 3, montrer qu'à surface donnée, le parallélépipède rectangle de plus grand volume est le cube.