TP noté Maple (2h)

1. Analyse

Soit n un entier naturel. On note $\sigma(n)$ la *fonction somme des diviseurs de* n, càd la somme

$$\sigma(n) := \sum_{d|n} d$$

Exercice 1. Écrire une procédure qui prend en paramètre un entier n et qui retourne $\sigma(n)$. (Remarque : si d, n sont des entiers, taper sous Maple d mod n renvoit le reste de la division de n par d, ce qui est utile pour tester si $d|n$)

Exercice 2. Cette fonction est définie dans Maple sous le nom sigma, et appartient au package numtheory (taper donc with(numtheory) : pour le charger). Comparer les valeurs renvoyées par votre procèdure et celles provenant de la procédure Maple pour n allant de 1 à 100.

Exercice 3. Vérifier sur des exemples que si m et n sont premiers entre eux, alors

$$\sigma(mn) = \sigma(m)\sigma(n)$$

On pourra utiliser la procédure Maple *gcd* (qui renvoit le PGCD) pour vérifier que deux nombres sont bien premiers entre eux.

Exercice 4. Écrire une procédure qui trace $\sigma(n)$ en fonction de n, pour $1 \leq n \leq N$.

On rappelle que si x_n est une suite,

$$\limsup_{n\to\infty} x_n := \lim_{n\to\infty} \sup_{m\geq n} x_m$$

c'est donc la plus grande valeur au voisinage de laquelle le graphe "repasse toujours" quand n tend vers l'infini. Le théorème de Grönwall affirme que

$$\limsup_{n\to\infty} \frac{\sigma(n)}{\log(\log(n))} = e^{\gamma}$$

où γ est la constante d'Euler qu'on obtient en Maple en tapant evalf(gamma).

Exercice 5. Écrire une procédure en reprenant celle de l'exercice précédent pour vérifier graphiquement ce résultat.

On rappelle que le *n-ième nombre harmonique* H_n est défini par la somme

$$H_n := \sum_{k=1}^{n} \frac{1}{k}$$

L'hypothèse de Riemann est une célèbre conjecture. Lagarias a demontré qu'elle était équivalente à l'affirmation

$$\forall n > 1,\ \sigma(n) < H_n + \log(H_n)e^{H_n}$$

Exercice 6. Écrire une procédure qui prend en paramètre un entier n, qui renvoie 0 si cette inégalité est fausse et 1 si elle est vraie. En utilisant *rand()* la tester sur des grands entiers choisis au hasard. Note : trouver un entier pour lequel cette inégalité n'est pas vérifiée revient à prouver que l'hypothèse de Riemann est fausse, et donc à remporter un prix d'un million de dollars.

2. Algèbre

Ne pas oublier de taper au préalable : with(LinearAlgebra) :

L'application exponentielle peut être généralisée aux matrices. Si A est une matrice carré, son exponentielle est la somme infinie :

$$\exp(A) = \sum_{n=0}^{\infty} \frac{1}{n!} A^n \tag{1}$$

Celle ci est en général difficile à calculer. On s'intéresse donc à quelques cas particuliers.

Exercice 7. Soit

$$D = \begin{pmatrix} d_1 & 0 & \dots 0 & \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & d_n \end{pmatrix}$$

une matrice diagonale. On vérifie facilement que

$$\exp(D) = \begin{pmatrix} \exp(d_1) & 0 & \dots 0 & \\ 0 & \exp(d_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \exp(d_n) \end{pmatrix}$$

Écrire une procédure qui calcule l'exponentielle d'une matrice diagonale. Indication : utiliser la commande RowDimension(A) qui renvoit le nombre de lignes d'une matrice.

Exercice 8. Utiliser la commande : D := RandomMatrix(n, outputoptions = [shape = diagonal]) ; pour une valeur de n pas trop grande pour fabriquer des matrices diagonales et tester votre procédure. Comparer avec le résultat de MatrixExponential(D).

On rappelle qu'une matrice M est dite *nilpotente* si il existe $k \in \mathbb{N}$ telle que $M^k = (0)$ (où (0) est la matrice nulle). Dans ce cas, la somme (1) est finie ce qui fait qu'on peut la calculer. On vérifie facilement que k est toujours inférieur ou égal à la taille de la matrice.

Exercice 9. Écrire une procédure qui calcule l'exponentielle d'une matrice nilpotente. Indication : puisque on ne connait pas à l'avance la valeur k pour laquelle M^k s'annule, utiliser une boucle *while* pour savoir quand arrêter le calcul de la somme. Attention à ne pas gaspiller du temps de calcul en recalculant A^n à chaque étape, essayer d'utiliser les calculs déjà effectués.

Une matrice triangulaire supérieur qui n'a que des 0 sur la diagonale est nilpotente. On utilisera la commande indigeste suivante pour en fabriquer pour un n fixé au préalable (par exemple $n := 4$ ou $n := 5$) : M :=RandomMatrix(n, outputoptions = [shape = triangular[upper, unit]])-IdentityMatrix(n) ;

Exercice 10. Tester votre procédure sur quelques exemples. Comparer avec le résultat de MatrixExponential(M). Remarquer que la matrice obtenue est inversible.