Propositions de projets

1. i) **Sujet** : Arithmétique dans $\mathbb{Z}$ et dans $\mathbb{R}[X]$.
   ii) **Description** : Écrire des fonctions pour déterminer si des entiers sont premiers, et pour lister les plus petits entiers. Construire de même les solutions du “Théorème des restes chinois”. Écrire aussi une fonction déterminant si un polynôme de $\mathbb{R}[X]$ est irréductible, et adapter à ces polynômes l'algorithme de résolution du “Théorème des restes chinois”.
   $*$
   iii) **Mots clés** : Crible d'Érathostène, algorithme de Gauss étendu, polynômes irréductibles, polynômes d'interpolation de Lagranges.
   iv) **Références** : Cours de L1 et L2, wikipedia, etc.

2. i) **Sujet** : Pivot de Gauss et décomposition LR.
   ii) **Description** : Résoudre un système linéaire
   $*$
   $$Ax = b \tag{1}$$
   avec $A \in M_n(\mathbb{R})$ et $x,\, b \in \mathbb{R}^n$. Considérer d'abord le cas sans permutation et après le cas générale avec permutation. Comme ouverture, construire un algorithme pour la décomposition LR d'un matrice A comme le produit d'une matrice triangulaire inférieure L avec des 1 dans la diagonal principal et une matrice R triangulaire supérieure.
   Développer une application qu'implique une visualisation.
   iii) **Mots clés** : Pivot de Gauss, décomposition LR.
   iv) **Références** : À compléter.

3. i) **Sujet** : Algorithme de Gram-Schmidt et décomposition QR.
   ii) **Description** : Implémenter un code qui construit une base orthonormée à partir d'un ensemble de n vecteurs (vérifier qu'ils on linéairement indépendants). Une fois qu'on dispose de ce code, implémenter la décomposition QR d'une un matrice inversible A.
   $**$
   Développer une application qu'implique une visualisation.
   iii) **Mots clés** : Gram-Schmidt, décomposition QR.
   iv) **Références** : À compléter.

4. i) **Sujet** : Résolution des EDOs : Euler et Runge-Kutta.
   ii) **Description** : Considérer une équation différentielle ordinaire et implémenter les algorithmes de Euler et Runge-Kutta (d'ordre quatre) pour sa résolution. Illustrer les différences dans les résolutions en appliquant les algorithmes à différentes équations. Comparer avec l'instruction `odeint` de python.
   $**$
   Développer une application qu'implique une visualisation.
   iii) **Mots clés** : EDO, algorithme d'Euler, Runge-Kutta.
   iv) **Références** : À compléter.

5. i) **Sujet** : Le nombre π.
   ii) **Description** : De nombreuses méthodes existent pour déterminer informatiquement une valeur numérique de π la plus précise possible. Un site comme `http://www.pi314.net` décrit nombre de ces méthodes, parmi lesquelles la méthode d'*Archimède* et celle de *Nicolas de Cues* ont une interprétation géométrique, tandis que la méthode de Plouffe est particulièrement adaptée au calcul binaire (utilisé par tous les ordinateurs). Vous implémenterez certaines de ces méthodes d'évaluation numérique de π, et en comparerez les performances.
   $**$
   iii) **Mots clés** : π, suites récurrentes, Plouffe, Archimède.
   iv) **Références** : Par exemple `http://www.pi314.net`

6. i) **Sujet** : Erreurs d'arrondis.
   ii) **Description** : Les méthodes numériques sur ordinateurs sont parfois très sensibles aux erreurs d'arrondis. Cela peut provoquer des imprécisions sensibles lorsqu'un calcul requiert de nombreuses étapes successives où les erreurs d'arrondis peuvent s'ajouter (ou au contraire se compenser), comme c'était le cas dès le TD1.
   $**$
   On pourra par exemple s'appuyer sur le premier chapitre du livre “Analyse numérique et équations différentielles”, de J.-P. Demailly. On illustrera informatiquement certains des phénomènes mentionnés.
   iii) **Mots clés** : Erreurs d'arrondis, analyse numérique, mantisse.
   iv) **Références** : Par exemple “Analyse numérique et équations différentielles”, de J.-P. Demailly

7. i) **Sujet** : Ensembles de Mandelbrot et de Julia.
   ii) **Description** : Explorer la dynamique de systèmes complexes en étudiant les comportement de la suite
   $***$

$$z_{n+1} = f(z_n) \quad , \quad z_0 = 0 \tag{2}$$

   avec des $f(z) = z^2 + c$, en fonction de la valeur de c (ici $z_n, c \in \mathbb{C}$). En particulier, déterminer les points c pour lesquels la suite est bornée et ceux pour lesquels elle n'est pas bornée. Ceci définit l'ensemble de Mandelbrot. Explorer autres fonctions f(z) et la relation avec les *ensembles de Julia*.
   iii) **Mots clés** : ensemble de Mandelbrot, ensembles de Julia.
   iv) **Références** : wikipedia (par example).

8. i) **Sujet** : Trajectoires orbitales et diagrammes de phases
   ii) **Description** : Résoudre, en coordonnées sphériques, les équations de mouvement d'une particule dans le potentiel central déterminé par la loi de Newton. Reproduire les trajectoires de Kepler dans les cas des orbites “liées” et les trajectoires ouverts paraboliques et hyperboliques (comètes). Construire les diagrammes des phases de ce système. Explorer de modifications de ces équations, notamment les trajectoires d'une particule autour un trou noir de Schwarschild et (challenge !) un trou noir de Kerr.
   $***$
   Développer des visualisations, notamment des animations.
   iii) **Mots clés** : orbites kepleriennes, orbites autour un trou noir.

iv) **Références** : À compléter.

9. i) **Sujet** : Combinatoire et Séries génératrice des nombres de Catalan.

ii) **Description** : L'ordinateur permet de compter les objets satisfaisant certaines propriétés.

*** Par exemple, compter le nombre de façon d'insérer des parenthèses dans une expression donne les nombres de Catalan.

On pourra dans un premier temps demander à Python de lister et compter ces parenthésages pour calculer les premier nombres de Catalan. On étudiera ensuite une méthode mathématique qui donne, à partir d'une relation de récurrence qu'ils satisfont, l'expression explicite de ces coefficients : l'utilisation d'une "Série génératrice".

iii) **Mots clés** : Combinatoire, nombres de Catalan, séries entières.

iv) **Références** : Nombreuses références sur internet.