Appropriation de cours

À l'aide du fascicule « *Initiation à la programmation MATLAB* » qui vous a été fourni, familiarisez-vous avec le logiciel MATLAB et sa syntaxe. Une fois à l'aise avec ce nouveau langage, écrivez des programmes répondant aux problèmes présentés dans ce document.

Travail à rendre

Par groupes de **2 ou 3 étudiants**, vous devrez rendre pour le **vendredi 6 mai 2022 (18h00)** des programmes MATLAB répondant, au choix :

- soit aux **3 problèmes** proposés et à l'**une des questions bonus**,
- soit à **2 problèmes** parmi les 3 proposés ainsi que leurs **deux questions bonus**.

Ce travail sera noté sur 20, et le barème favorisera l'utilisation de méthodes matricielles en lieu et place de boucles. À titre indicatif, un problème ou une question bonus, lorsqu'elle est traitée d'un point de vue matriciel, ne prend pas plus de 15 lignes.

Consignes de rendu

Vous rendrez des fichiers au format texte avec l'extension *.m* et en respectant la nomenclature suivante :

- le programme répondant au problème 1 devra s'appeler `montecarlo.m`,
- le programme répondant au problème 2 devra s'appeler `pagerank.m`,
- le programme répondant au problème 3 devra s'appeler `fractale.m`.

La ou les questions bonus seront traitées dans le même fichier que le problème correspondant, vous ferez en sorte de séparer les deux parties du code en utilisant une ligne commentée :

```
% Question bonus
```

Vous rendrez vos deux ou trois fichiers dans **un seule archive** au format **ZIP**, dont le nom de fichier fera apparaître les noms de famille de tous les étudiants du groupe.

Attention. Le non-respect de ces consignes entraînera des points de pénalité.

Problème 1 – Méthode de Monte–Carlo

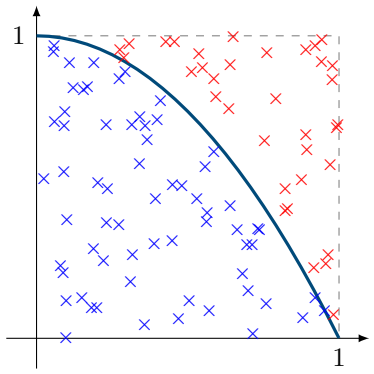
La *méthode de Monte–Carlo* désigne, dans le contexte du calcul intégral, une méthode probabiliste donnant une valeur approchée de l'intégrale d'une fonction. Illustrons-la dans le cas de :

$$I = \int_0^1 (1 - x^2) \mathrm{d}x.$$

Son principe repose sur le fait que la valeur de I s'interprète géométriquement comme l'aire du domaine $\mathcal{D}$ délimité par la courbe $y = 1 - x^2$ au dessus du segment $[0, 1]$. Si l'on tire aléatoirement (de manière indépendante et selon une loi uniforme) des points (x_i, y_i) dans le carré unité $\mathcal{C} = [0, 1]^2$, la loi des grand nombres nous indique que lorsque le nombre de tirages tend vers $+\infty$, la proportion de ces points tombant dans $\mathcal{D}$ parmi tous les points tirés devrait se rapprocher du quotient

$$\frac{\text{aire}(\mathcal{D})}{\text{aire}(\mathcal{C})} = I.$$

Ainsi en simulant un très grand nombre de lancers et en regardant la proportion de points tombés sous la courbe, on peut obtenir numériquement une approximation plus ou moins bonne [1] de la valeur de I.



$$I \approx \frac{65}{100} = 0.65$$

Illustration pour un échantillon de 100 points.

Problème. Implémenter dans MATLAB la méthode de Monte-Carlo sur l'exemple donné en prenant $N \geqslant 1000$ points. En plus de la valeur approchée obtenue, faire apparaître une figure similaire à celle présentée ci-dessus.

1. En réalité, pour une intégrale simple, cette méthode n'est pas très performante par rapport aux autres méthodes numériques que vous avez rencontrées jusque là car elle converge en général moins vite vers la valeur exacte. En revanche elle s'avère bien plus compétitive lorsqu'il s'agit de calculer des intégrales doubles (ou triples, ou plus ...) sur le même principe.

Indications

Il n'y a pas besoin d'utiliser de boucles pour cet exercice. Commencez par simuler aléatoirement une matrice

$$\begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_N \\ y_1 & y_2 & y_3 & \cdots & y_N \end{pmatrix}$$

de réels compris entre 0 et 1, puis utilisez le masquage logique pour détecter les indices i pour lesquels le point (x_i, y_i) se situe sous la courbe $y = f(x)$. Cela vous donnera un vecteur

$$\begin{pmatrix} t_1 & t_2 & t_3 & \cdots & t_N \end{pmatrix}$$

formé de 0 et de 1 à partir duquel vous pourrez facilement :

- calculer le nombre de points compris dans le domaine $\mathcal{D}$,
- sélectionner les points à afficher en rouge ou en bleu sur le graphique demandé.

Pour aller plus loin ...

Pour écrire mathématiquement les choses décrites ci-dessus, supposons que l'on ait une suite (infinie) $((x_n, y_n))_{n\geqslant 1}$ de points tirés aléatoirement dans le carré $[0,1]^2$ (les tirages étant tous indépendants et suivant une loi uniforme). Si on pose

$$S(n) = \operatorname{card}\{1 \leqslant k \leqslant n \mid (x_k, y_k) \in \mathcal{D}\}$$

(c'est-à-dire que $S(n)$ représente le nombre de points appartenant à $\mathcal{D}$ parmi les n premiers points tirés), alors nous affirmons que

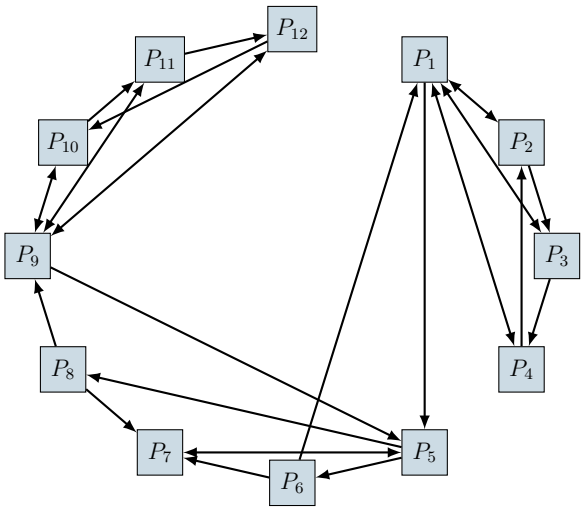
$$\lim_{n\to\infty} \frac{S(n)}{n} = I.$$

Ici, nous pouvons vérifier expérimentalement si cette affirmation est crédible car la valeur de l'intégrale I peut se calculer de manière théorique sans aucun problème.

Question bonus. Sur une nouvelle figure, dessiner un graphique faisant apparaître la droite $y = I$ (pour la valeur exacte de I que vous aurez calculée à la main) ainsi que l'évolution de $S(n)/n$ en fonction de n (pour n compris entre 1 et N).

Problème 2 – Une approche de l'algorithme *PageRank*

L'ensemble des pages web disponibles sur l'Internet peut être représenté mathématiquement par un immense *graphe*, dans lequel chaque sommet est une page web P_i, et dans lequel on ajoute un flèche de P_i vers P_j si la page P_i contient un lien hypertexte vers la page P_j. Ci-dessous, un exemple [1] d'un tel graphe pour 12 pages fictives $P_1, P_2, \dots, P_{12}$, ainsi que la *matrice d'adjacence* A associée (c'est la matrice $(a_{i,j})$ telle que $a_{i,j}$ vaut 1 si la page P_i envoie sur la page P_j, et 0 sinon).



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

L'algorithme *PageRank* [2] permet de quantifier l'importance d'une page web en se basant sur le postulat suivant : plus cette page est importante, plus nombreuses seront les autres page web à pointer vers elle via des liens hypertextes. Dans l'exemple ci-dessus, on peut par exemple se dire naïvement que la page P_1, référencée par 4 autres pages, est plus importante que la page P_2, référencée par seulement 2 autres pages. En poussant l'argument plus loin, on pourrait de plus considérer qu'une référence depuis une page importante doit compter plus qu'une référence depuis une page moins importante : par exemple P_3 est référencée par les pages P_1 et P_2, mais comme P_1 est plus importante, on peut considérer que le lien de P_1 vers P_3 a plus de poids que le lien de P_2 vers P_3. Le calcul devient alors assez compliqué ...

Pour simplifier le problème, l'algorithme *PageRank* choisit une approche *probabiliste*. Imaginons un visiteur qui navigue de page en page, en choisissant à chaque fois de cliquer au hasard sur l'un des liens présents. Pour simplifier les calculs, on ne tient pas compte des pages déjà visitées (c'est-à-dire que le visiteur peut retourner sur une page déjà parcourue), et on imagine que le choix se fait de manière équiprobable. Notons X_n le vecteur de taille 12, dont la i-ème composante représente la probabilité de se retrouver sur la page i après n clics. Si le visiteur commence sur la page P_1, on aura alors pour premiers

1. Exemple tiré de l'article « *Comment Google classe les pages web* » de M. Eisermann sur *Images des Mathématiques* (CNRS). `https://images.math.cnrs.fr/Comment-Google-classe-les-pages-web.html`
2. Algorithme inventé Larry PAGE, cofondateur de Google, et encore partiellement utilisé par ce moteur de recherche (en complément d'autres méthodes plus sophistiquées)

vecteurs :

$$X_0 = \begin{pmatrix} \overset{1}{1} & \overset{2}{0} & \overset{3}{0} & \overset{4}{0} & \overset{5}{0} & \overset{6}{0} & \overset{7}{0} & \overset{8}{0} & \overset{9}{0} & \overset{10}{0} & \overset{11}{0} & \overset{12}{0} \end{pmatrix},$$

$$X_1 = \begin{pmatrix} 0 & 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$X_2 = \begin{pmatrix} 0.375 & 0.125 & 0.125 & 0.125 & 0 & 0.083 & 0.083 & 0.083 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

En continuant suffisamment longtemps, il semblerait que cette suite $(X_n)_{n\geqslant 0}$ converge vers un vecteur :

$$X_\infty = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{pmatrix}.$$

La i-ème composante x_i de ce vecteur limite peut alors s'interpréter comme la probabilité que l'utilisateur se retrouve sur la page P_j après être parti de la page P_1 et s'être promené infiniment longtemps sur la toile. C'est cette valeur que l'on retiendra ici comme indicateur *PageRank* de la page P_i, car elle traduit justement le fait que la page P_i est souvent visitée, car référencée par des pages elles-mêmes souvent visitées, et ainsi de suite. Bien sûr, il faudrait en toute rigueur s'assurer que la valeur de cet indicateur ne dépend pas de la page dont on a choisi de partir.

Problème. Établir le classement de ces 12 pages web suivant l'algorithme *PageRank* (simplifié) présenté ci-dessus.

Indications

Commencez par construire la matrice $P = (p_{i,j}) \in \mathcal{M}_{12}(\mathbb{R})$ telle que $p_{i,j}$ soit égal à la probabilité de cliquer sur un lien menant vers la page P_j lorsqu'on se trouve sur la page P_i. Comme on a choisi un modèle équiprobable, cette probabilité est soit égale à $1/n_i$ (avec n_i le nombre de pages différentes vers lesquelles pointe P_i) si jamais la page P_i pointe vers la page P_j, soit à zéro s'il n'y a pas de lien de la page P_i vers la page P_j. L'intérêt de passer par cette matrice est double :

1) il est possible dans MATLAB de calculer facilement P à partir de la matrice A, sans utiliser de boucle,
2) pour tout $n \in \mathbb{N}$, on aura alors la relation

$$X_{n+1} = X_n \times P$$

(vous pouvez essayer de vous en convaincre avec un arbre de probabilités).

Comme MATLAB ne calcule qu'avec un nombre fini de décimales, la convergence théorique de la suite $(X_n)_{n\geqslant 0}$ va se traduire par le fait que la suite $(X_n)_{n\geqslant 0}$ de valeurs que vous allez calculer numériquement sera constante à partir d'un certain rang. En première approximation, vous pourrez émettre l'hypothèse que dès que vous avez atteint un stade où $X_{n+1} = X_n$, alors vous avez atteint votre vecteur limite X_∞.

Pour aller plus loin ...

Cet algorithme comporte une faille majeure : si jamais une page web ne possède aucun lien vers une autre page, notre visiteur imaginaire va rester bloqué indéfiniment sur cette page ... Pour cela, le véritable algorithme *PageRank* introduit la variation suivante. Plutôt que de systématiquement cliquer sur un lien, on décide qu'à chaque étape :

— avec probabilité $c \in]0, 1[$, le visiteur abandonne sa navigation en cours et retourne sur l'une des 12 pages choisie au hasard (il peut éventuellement recommencer sur la page où il se trouvait),

— avec probabilité $1 - c$, le visiteur suit un lien au hasard sur la page en cours (comme il le faisait dans la méthode précédente).

Dans les faits, cela revient simplement à remplacer les probabilités $p_{i,j}$ introduites ci-dessus par les nouvelles probabilités $p'_{i,j}$ définies par :

$$p'_{i,j} = c \times \frac{1}{12} + (1 - c) \times p_{i,j}.$$

Question bonus. Déterminer ce que devient le classement des 12 pages précédentes pour cette nouvelle version de l'algorithme (en prenant $c = 0{,}15$).

Problème 3 – Création d'une image de fractale

Étant donné un nombre complexe $c \in \mathbb{C}$, considérons la suite $(z_n)_{n\geqslant 0}$ définie par :

$$\begin{cases} z_0 = 0, \\ z_{n+1} = z_n^2 + c. \end{cases}$$

On s'attend à ce que le comportement de cette suite (variations, limites, ...) dépende éventuellement de la valeur de c qui aura été choisie. Justement, on appelle *ensemble de* MANDELBROT[1] l'ensemble des complexes $c \in \mathbb{C}$ pour lesquels la suite $(z_n)_{n\geqslant 0}$ est **bornée**.

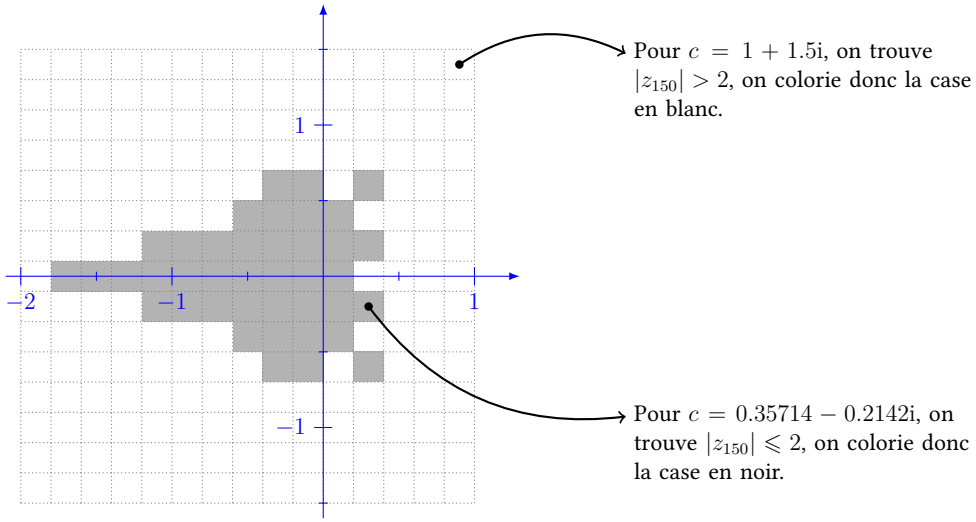
Problème. À l'aide du procédé décrit ci-dessous et de la commande `imshow` de MATLAB, créez puis affichez une image représentant l'ensemble de MANDELBROT dans le plan complexe.

Pour dessiner l'ensemble de MANDELBROT, on va créer une image de taille d pixels par d pixels représentant la région du plan complexe :

$$E = \{a + \mathrm{i}b \mid -2 \leqslant a \leqslant 1,\ -1.5 \leqslant b \leqslant 1.5\} \subset \mathbb{C}.$$

Chaque pixel de cette image correspondra naturellement à un nombre complexe $c \in E$. Pour cette valeur de c, on calculera le 150-ième terme z_{150} de la suite (z_n) définie ci-dessus, puis nous ferons l'approximation suivante :

- si $|z_{150}| > 2$, on supposera que la suite $(z_n)_{n\geqslant 0}$ n'est pas bornée et on coloriera la case en blanc,
- si $|z_{150}| \leqslant 2$, on supposera que la suite $(z_n)_{n\geqslant 0}$ est bornée et on coloriera la case en noir. [2]



1. En hommage au mathématicien polono-franco-américain Benoît MANDELBROT (1924–2010) à qui on associe la découverte des *fractales*, notamment au travers de l'étude de cette suite.

2. Ces choix en apparence arbitraires se basent sur des théorèmes rigoureux qui nous garantissent qu'on n'est pas trop loin de la vérité, et que donc la partie de E colorié en noir sera une bonne approximation de l'ensemble de MANDELBROT.

Indications

1) Pour quadriller E, vous pouvez commencer par créez deux subdivisions à pas constant :
   - $(a_1, a_2, \dots, a_d)$ une subdivision de l'ensemble $[-2, 1]$ avec $a_1 = -2$ et $a_d = 1$,
   - $(b_1, b_2, \dots, b_d)$ une subdivision de l'ensemble $[-1.5, 1.5]$ avec $b_1 = 1.5$ et $b_d = -1.5$.

   Utilisez ensuite ces subdivisions pour créer la matrice $C = (c_{i,j}) = (a_i + \mathrm{i}\, b_j) \in \mathcal{M}_d(\mathbb{C})$.

   Remarque. *Cette étape là ne nécessite pas de boucle !*
2) Vous pouvez, dans une même matrice Z_n de taille $d \times d$, calculer simultanément le n-ième terme de la suite $(z_n)_{n\geqslant 0}$ pour chacune des valeurs de c contenue dans la matrice C construite précédemment. Une fois arrivé au 150^{e} terme, vous n'aurez plus qu'à utiliser un masquage logique pour créer l'image voulue.

Exemple. Pour $d = 4$.

- On crée la matrice C contenant les 16 valeurs de c que l'on va tester :

$$C = \begin{pmatrix} -2+1.5\mathrm{i} & -1+1.5\mathrm{i} & 0+1.5\mathrm{i} & 1+1.5\mathrm{i} \\ -2+0.5\mathrm{i} & -1+0.5\mathrm{i} & 0+0.5\mathrm{i} & 1+0.5\mathrm{i} \\ -2-0.5\mathrm{i} & -1-0.5\mathrm{i} & 0-0.5\mathrm{i} & 1-0.5\mathrm{i} \\ -2-1.5\mathrm{i} & -1-1.5\mathrm{i} & 0-1.5\mathrm{i} & 1-1.5\mathrm{i} \end{pmatrix}.$$

- On crée la matrice Z_0 contenant la valeur de z_0 pour chacun de ces c :

$$Z_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

- On opère sur Z_0 de sorte à obtenir les différentes valeurs de $z_1 = z_0^2 + c$:

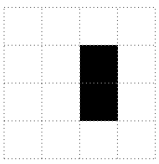
$$Z_1 = \begin{pmatrix} -2+1.5\mathrm{i} & -1+1.5\mathrm{i} & 0+1.5\mathrm{i} & 1+1.5\mathrm{i} \\ -2+0.5\mathrm{i} & -1+0.5\mathrm{i} & 0+0.5\mathrm{i} & 1+0.5\mathrm{i} \\ -2-0.5\mathrm{i} & -1-0.5\mathrm{i} & 0-0.5\mathrm{i} & 1-0.5\mathrm{i} \\ -2-1.5\mathrm{i} & -1-1.5\mathrm{i} & 0-1.5\mathrm{i} & 1-1.5\mathrm{i} \end{pmatrix}.$$

- Puis, on opère sur Z_1 de sorte à obtenir les différentes valeurs de $z_2 = z_1^2 + c$:

$$Z_2 = \begin{pmatrix} -0.25-4.5i & -2.25-1.5i & -2.25+1.5i & -0.25+4.5i \\ 1.75-1.5i & -0.25-0.5i & -0.25+0.5i & 1.75+1.5i \\ 1.75+1.5i & -0.25+0.5i & -0.25-0.5i & 1.75-1.5i \\ -0.25+4.5i & -2.25+1.5i & -2.25-1.5i & -0.25-4.5i \end{pmatrix}.$$

- On continue ainsi jusqu'à obtenir les différentes valeurs de z_{150}.
- On crée une image en posant 255 là où on a trouvé $|z_{150}| > 2$ et 0 dans les autres cas :

$$\begin{pmatrix} 255 & 255 & 255 & 255 \\ 255 & 255 & 0 & 255 \\ 255 & 255 & 0 & 255 \\ 255 & 255 & 255 & 255 \end{pmatrix} \quad \longleftrightarrow \quad$$



Ici on obtient une image de taille 4px par 4px, ce qui n'est pas très intéressant. L'intérêt sera donc d'effectuer ce même algorithme avec une valeur de d bien plus grande.

Pour aller plus loin ...

Pour rendre l'image plus belle, on peut modifier légèrement l'algorithme présenté ci-dessus de la façon suivante. Dès qu'on trouve $|z_k| > 2$ (pour une valeur de c donnée), au lieu d'affecter au pixel correspondant la valeur 255 (blanc), on lui associe une valeur d'autant plus petite que k est grand. On peut par exemple utiliser une formule du type $255 - 2k$ (mais attention à bien générer des valeurs entre 0 et 255). L'idée est qu'un pixel sera d'autant plus sombre que la suite correspondante mettra du temps pour dépasser (en module) la valeur 2 (le cas critique étant un point complètement noir si la suite ne dépasse jamais la valeur 2). Cela aura pour effet de créer des ombres et dégradés au bord de la figure obtenue précédemment.

Question bonus. Sur une nouvelle figure, dessiner à nouveau la fractale de MANDELBROT en ajoutant cette fois ces effets d'ombre.