TP n^{o}1 : initiation à MATLAB

Les objectifs de ce premier TP étaient multiples, à commencer par l'étude d'algorithmes simples codés en langage MATLAB. On détaille ci-dessous les principaux défauts rencontrés dans les codes et les rapports, tant au niveau du fond que de la forme. Ce rapport doit servir de base pour les prochains TP.

1 Étude d'une suite remarquable

Le premier exercice consiste à implémenter le calcul des premiers termes d'une suite et de conjecturer son comportement global. La suite concernée est définie par la relation de récurrence :

$$\forall\, n \in \mathbb{N},\ u_{n+1} = \begin{cases} 3u_n + 1, & \text{si } u_n \text{ est impair,} \\ \dfrac{u_n}{2}, & \text{sinon.} \end{cases}$$

Comme toute suite récurrente, il convient de se donner les premiers termes. La récurrence étant ainsi à un pas, la donnée de u_0 suffit. Il était précisé dans l'énoncé $u_0 \in \mathbb{N}$. Les cas $u_0 \in \mathbb{R}\backslash\mathbb{N}$ et $u_0 \in \mathbb{Z}_-$ sont possibles mais les conclusions sont différentes. Il faut donc en premier lieu **tester la valeur entrée par l'utilisateur**. On pouvait par exemple redemander une valeur pour u_0 dans les cas $u_0 < 0$ et $E(u_0) \neq u_0$.

La démarche consiste ici à écrire une fonction `grelons.m` pour construire la suite à partir d'un u_0 donné et d'un nombre d'itérations maximal. Une fois implémentée, cette fonction permet d'établir la conjecture suivante :

Conjecture 1

Quel que soit $u_0 \in \mathbb{N}$, la suite (u_n) est périodique à partir d'un certain rang.

Pour $u_0 = 0$, la suite est constante à 0. Pour $u_0 \in \mathbb{N}^*$, on constate que la suite atteint un cycle $4-2-1$ (*cf.* Fig. 1). Il n'est toutefois pas possible de faire un lien entre l'indice auquel on atteint le cycle et le terme initial ni entre la valeur du maximum et le premier terme (*cf.* Fig.2).

Des études ont été menées sur cette suite. Sans que l'on ait réussi à le prouver mathématiquement, tous les premiers termes de 1 à 10^{12} ont abouti au cycle infernal [1].

Cependant, dans le cas (improbable) où la suite divergerait, il est préférable de conserver un test de précaution sur le nombre d'itérations.

Pour le confort de l'utilisateur, il est nécessaire de créer un script qui utilise la fonction `grelons.m`, où il est possible de fixer le premier terme (voire le nombre d'itérations), et que l'on peut relancer autant de fois qu'on le souhaite. Il faut donc prévoir une possibilité de sortie du programme pour l'utilisateur.

De manière générale, il faut annoncer le calcul qui va être lancé, expliquer à l'utilisateur ce qu'il est censé entrer au clavier et surtout **expliquer les affichages**. [1]

Il est préférable d'illustrer les résultats par des figures, surtout pour une convergence de suite.

1. Les sorties `ans = ...` ne sont pas très explicites, surtout lorsqu'il y a trois ou quatre variables en jeu.

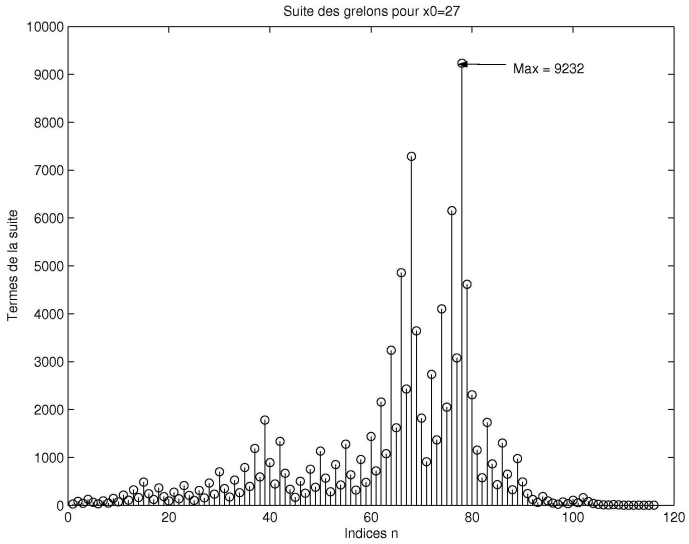
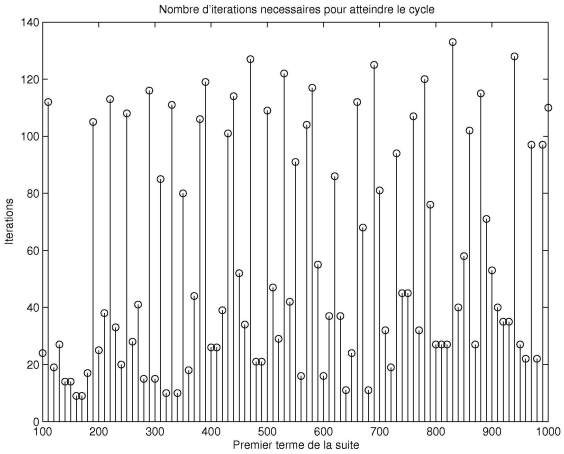
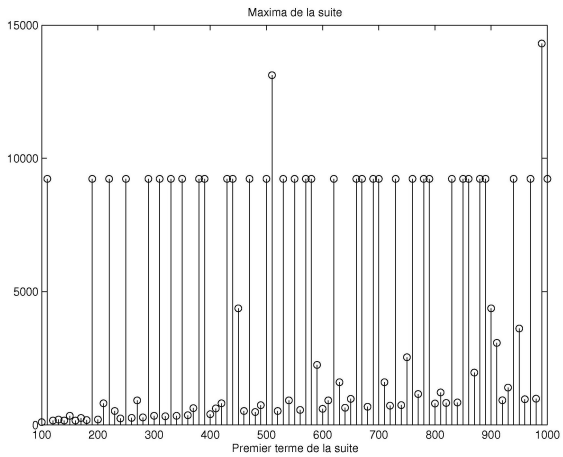


FIGURE 1 – Evolution de la suite d'origine 27



(a) Itération à laquelle on atteint le cycle



(b) Maximum de la suite

FIGURE 2 – Influence du premier terme

2 Recherche des zéros d'une fonction

On s'intéresse dans cette partie à la résolution de l'équation $f(x) = 0$ par deux algorithmes différents pour une fonction f continue sur un compact.

2.1 Méthode de dichotomie

La méthode de dichotomie est basée sur le principe des valeurs intermédiaires : si le produit $f(a)f(b) < 0$ avec f continue, alors nécessairement la fonction s'annule (au moins une fois) sur $]a, b[$. La méthode consiste alors à découper en deux l'intervalle d'étude à chaque itération. L'algorithme proposé est le suivant :

Algorithme 1 Algorithme de dichotomie

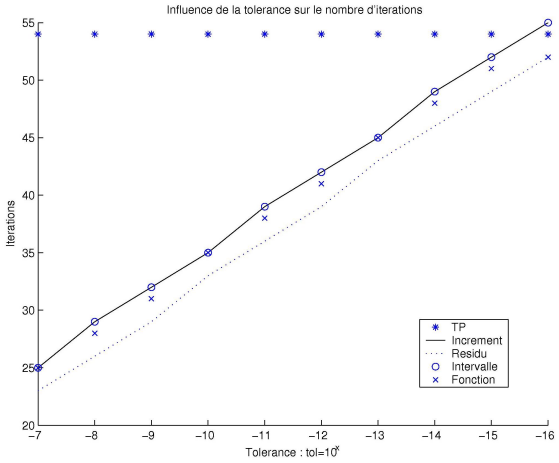
Paramètres d'entrées : f, a, b
f : fonction continue sur un intervalle I
a et b : deux réels de I tels que $f(a)f(b) < 0$
Paramètres de sorties : x, n
x : approximation du zéro de la fonction f
n : nombre d'itérations nécessaires pour atteindre x

1: $a_0 \leftarrow a$, $b_0 \leftarrow b$
2: $x_0 \leftarrow \dfrac{a+b}{2}$
3: **Tant que** $x_n > a_n$ **et** $x_n < b_n$ **Faire**
4: **Si** $f(a_n)f(x_n) > 0$ **alors**
5: $a_{n+1} \leftarrow x_n$, $b_{n+1} \leftarrow b_n$
6: $x_{n+1} \leftarrow \dfrac{x_n + b_n}{2}$
7: **Sinon**
8: $a_{n+1} \leftarrow a_n$, $b_{n+1} \leftarrow x_n$
9: $x_{n+1} \leftarrow \dfrac{x_n + a_n}{2}$
10: **Fin Si**
11: **Fin Tant que**
12: $x \leftarrow x_n$

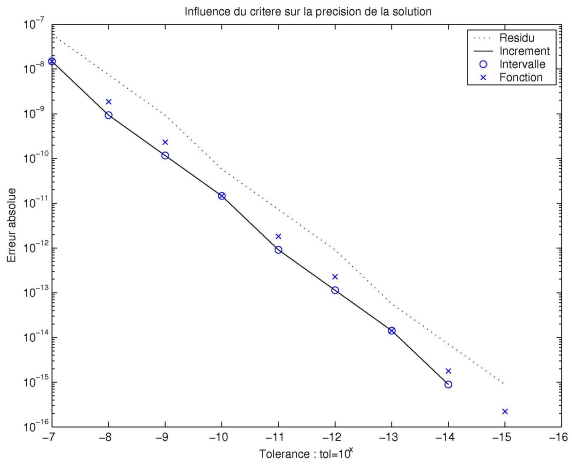
Un problème inhérent aux études de convergence est le choix du critère d'arrêt. Sur ce problème particulier où la suite récurrente (x_n) approche la solution de $f(x) = 0$, on peut tester différents critères :

1. un critère **absolu** (*cf.* celui de l'algorithme 1) : on sort de la boucle lorsque $x_n = a_n = b_n$, c'est-à-dire lorsque l'on atteint la précision machine (`eps` sous MATLAB) ;
2. un critère sur l'**incrément**, c'est-à-dire sur la quantité $|x_{n+1} - x_n|$ ou, ce qui est équivalent, sur $\max\big(|a_{n+1} - a_n|, |b_{n+1} - b_n|\big)$;[2]
3. si l'on connaît la limite ℓ (par exemple $\ell = 0$ pour la fonction arctan)[3], on peut mettre en place un critère sur le **résidu** en testant la quantité $|x_n - \ell|$;
4. on peut également utiliser la **taille de l'intervalle** : $|b_n - a_n|$;
5. on peut enfin penser au test sur la quantité $|f(x_n)|$, mais ce critère s'avère inadéquat pour les fonctions qui par exemple admettent une tangente horizontale au point d'annulation de la fonction.

2. Mais pas sur $|a_{n+1} - a_n|$ ou $|b_{n+1} - b_n|$ individuellement car l'un des deux est nul à chaque itération.
3. Ce qui n'est jamais le cas dans la pratique.



(a) Evolution du nombre d'itérations pour atteindre le critère d'arrêt



(b) Précision de la solution au critère d'arrêt

FIGURE 3 – Influence de la tolérance pour la fonction f_1

Pour déterminer les effets de ces critères et éventuellement observer que l'un est meilleur que les autres, il faut tous les implémenter et les lancer sur le même cas-test. Le paramètre pertinent est alors le nombre d'itérations nécessaires pour arriver à « convergence ». Deux remarques toutefois :

1. la valeur donnée par le programme n'est (le plus souvent) qu'une **approximation** de la solution : il faut garder à l'esprit qu'un calcul numérique reste approché par définition et ne pas parler de **la** solution.
2. une tolérance est à fixer pour les quatre derniers critères mais pas pour le premier, dont la précision est implicitement celle de la machine. Il faut donc comparer ce qui est comparable.

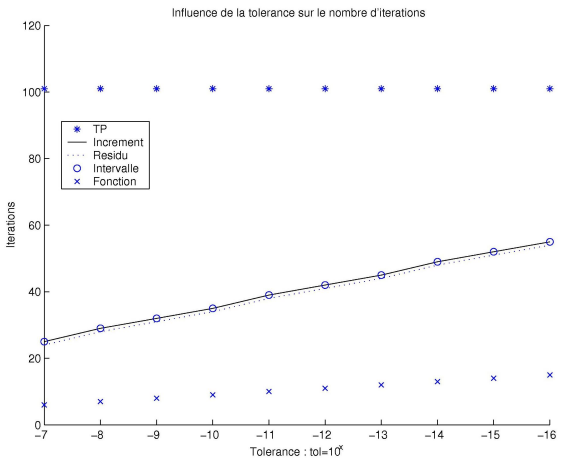
Prenons par exemple la fonction $f_1(x) = 3x^2 - 2x - 1$ dont les zéros sont 1 et $\frac{-1}{3}$. L'intervalle donné est $[0, 3]$. On observe sur la figure 3 l'influence de la tolérance sur le nombre d'itérations nécessaire pour atteindre le critère d'arrêt et sur la précision réelle de la solution numérique lorsque le critère d'arrêt est atteint (*i.e.* $|x_n - \ell|$). On constate que le critère sur le résidu dans ce cas présent est celui qui demande le moins d'itérations pour être atteint. La précision est cependant moindre que pour les autres critères. Ce résultat s'explique simplement : à tolérance égale, le critère sur le résidu est plus faible que le critère sur l'intervalle : si $b_n - a_n < \varepsilon$, sachant que $\ell \in]a_n, b_n[$, le milieu x_n est au moins proche à ε près de ℓ.

Effectuons un autre test avec la fonction $f_2(x) = \frac{1}{1000}x^3$. Le point d'annulation est 0 (racine triple). On trace alors les mêmes graphiques que précédemment (*cf.* Fig. 4). Ce cas-test met en valeur les difficultés liées au critère sur la fonction qui s'avère ici beaucoup moins précis que les autres, dont le comportement est similaire.

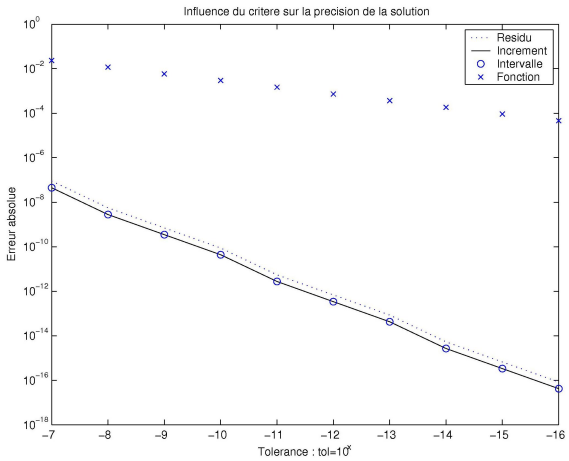
Une dernière remarque sur l'algorithme : il est bon de tester dans le critère que $f(x_n) \neq 0$ car même si l'on tombe exactement sur la solution, l'algorithme de base ne s'arrête pas ...

Cette méthode a l'avantage de toujours converger, et le nombre d'itérations pour atteindre une précision de ε sur la taille de l'intervalle est donné par :

$$|b_n - a_n| = \frac{|b-a|}{2^n} < \varepsilon \iff n \geqslant \log_2 \frac{|b-a|}{\varepsilon} + 1.$$



(a) Evolution du nombre d'itérations pour atteindre le critère d'arrêt



(b) Précision de la solution au critère d'arrêt

FIGURE 4 – Influence de la tolérance pour la fonction f_2

2.2 Méthode de Newton

Comme la méthode précédente, la méthode de Newton permet d'approcher le zéro d'une fonction continue. Il faut toutefois que la fonction soit de plus dérivable et que la dérivée au point d'annulation soit non nulle. L'idée est basée cette fois-ci sur le développement limité de la fonction au voisinage du zéro, noté $\bar{x}$:

$$\underbrace{f(\bar{x})}_{=0} = f(x) + (\bar{x} - x)f'(x) + o(|x - \bar{x}|).$$

En négligeant le $o(\cdot)$, on obtient la récurrence suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

A l'itération $n+1$, le point x_{n+1} correspond au point d'intersection de la tangente à la courbe de f en x_n avec l'axe des abscisses. Cette méthode converge à condition d'être initialement dans un voisinage de la solution.

On implémente l'algorithme en prenant soin de **tester la valeur courante de la dérivée** avant de diviser par $f'(x_n)$ et en appliquant un **test sur le nombre d'itérations** (cas de divergence) ! On peut de la même manière que dans la partie précédente **s'interroger sur le critère d'arrêt** (ce qui n'a pas été fait très souvent) : le critère sur l'incrément, le critère sur le résidu (si l'on connaît la solution exacte) ou le critère sur la fonction. On doit donc disposer d'une **fonction** qui prend comme argument la fonction, la donnée initiale x_0, le nombre maximal d'itérations et la tolérance associée au critère choisi, et d'un **script** qui demande à l'utilisateur de fixer les paramètres (voire de choisir le critère d'arrêt).

On présente ici des résultats pour un critère sur l'incrément pour la recherche du point d'annulation de la fonction arctan. On remarque que la méthode diverge sauf pour des petites valeurs de x_0. Déterminons l'intervalle de convergence en étudiant la suite récurrente associée :

$$x_{n+1} = x_n - (1 + x_n^2) \arctan x_n = g(x_n).$$

Si la suite converge, la limite est un point fixe de la fonction g (0 est l'unique point fixe de g). Cette fonction est décroissante sur $\mathbb{R}$, ce qui montre que les suites extraites (x_{2n}) et (x_{2n+1}) sont monotones.

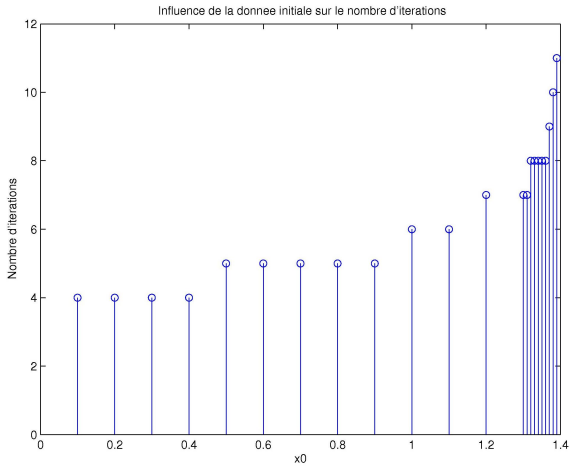


FIGURE 5 – Influence de x_0 pour la fonction arctan ($\varepsilon = 10^{-12}$)

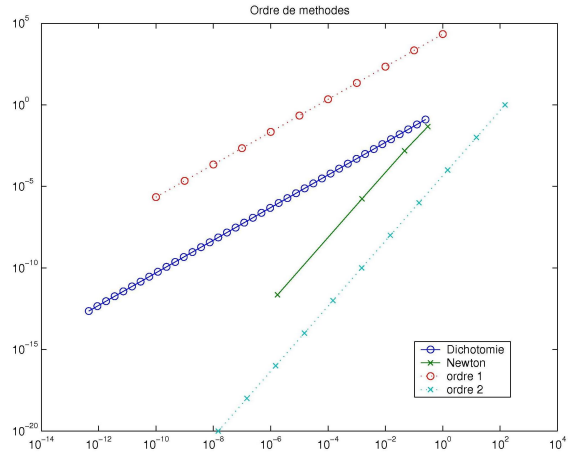


FIGURE 6 – Ordre des méthodes pour la fonction f_1 ($\varepsilon = 10^{-12}$)

Pour que la suite diverge, il faut que $|g(x_n)| > x_n$. Or l'équation $|g(x)| = |x|$ est équivalente à $x = 0$ ou $g(x) + x = 0$. Pour déterminer la solution de cette équation, on utilise l'algorithme de Newton. On trouve $\hat{x} \approx 1,3917$. Ainsi, pour $x_0 \in]-1.39, 1.39[$, la suite converge vers 0 en très peu d'itérations (*cf.* Fig. 5).

Pour assurer la convergence de la méthode de Newton, on effectue quelques (2 ou 3) itérations de dichotomie pour se ramener à l'intervalle de convergence avant de conclure avec la méthode classique de Newton.

Terminons par une comparaison sur l'ordre de ces deux méthodes. Reprenons la fonction f_1 définie précédemment. On constate sur la figure 6 que la méthode de dichotomie est d'ordre 1 tandis que la méthode de Newton est d'ordre 2.

3 Conclusion (numérique)

Ce TP permettait de mettre en place les concepts d'algorithmes, de fonctions de scripts, de critères d'arrêt et d'ordres. Il doit servir base pour toute étude de convergence, quel qu'en soit l'objet. Le numéricien doit prendre en compte les différents paramètres de l'algorithme, choisir judicieusement les cas-tests et les illustrations, s'assurer du bon fonctionnement du programme (tests, affichages) et le rendre le plus lisible et facile à utiliser.

4 Conception d'un rapport

Un rapport d'ingénieur doit avant tout mettre en l'accent sur les problèmes rencontrés et les solutions proposées. Sans être aussi ambitieux, le rapport doit présenter une problématique (en l'occurence la construction de suites récurrentes), les spécificités des algorithmes de construction, les résultats numériques obtenus (illustrés par des cas-tests) et la conclusion du travail mené.

En aucun cas, un rapport ne doit être :

- ✗ un renvoi systématique au code MATLAB ;
- ✗ une reprise de la numérotation des questions de l'énoncé du TP ;
- ✗ un catalogue de figures sans légende ni analyse ;
- ✗ une recopie intégrale du code MATLAB.

Un rapport doit donc se composer d'une **introduction**, d'une **conclusion**, et entre les deux, de parties constituées de problèmes mathématiques pour lesquels on propose des solutions que l'on **analyse** soigneusement sur la base des cours suivis (analyse, analyse numérique) et des observations faites à partir des simulations numériques. *Le rapport doit être rédigé dans l'optique de sa lecture par une personne qui n'aurait pas lu l'énoncé du TP.* Il convient donc ici de rappeler la construction des suites et éventuellement donner un contexte historique ou mathématique (origine de la méthode de dichotomie, de Newton).

Quelques remarques annexes :

- ✓ L'orhtographe et la grammaire sont des composantes fondamentales pour la rédaction d'un rapport. Une à plusieurs relectures s'imposent avant envoi.
- ✗ Eviter les marques de jugements du type « c'est simple », « c'est évident », « dans tous les cas » lorsque l'on a effectué que deux cas-tests, ...
- ✓ Une tolérance « raisonnable » est 10^{-10}, pas 10^{-4}. Imaginez si on se donnait une marge d'erreur de $10^{-4}m$ sur un avion ou une fusée. J'ai peur que certains n'arrivent pas à destination.
- ✓ Une figure doit avoir une légende, un titre et sauf en cas d'évidence, les axes doivent avoir des titres également.
- ✓ Tout résultat, figure, tableau, doit être analysé. En particulier, faire attention aux échelles sur les figures.
- ✗ Les figures ne doivent pas dépasser du cadre de la feuille. Idem pour les symboles mathématiques. Le compilateur le signale dans la fenêtre de sorties par un message du type `Overfull hbox`
- ✓ Pour écrire une formule mathématique, il suffit de placer des `$` autour de l'expression et non autour de chacun de ses termes.
- ✓ Pour qu'un caractère apparaisse avec une fonte mathématique, il faut qu'il soit entre `$`. Comparer $f(x_n) - n$ et f(x_n)-n (`$f(x_n)-n$` et `f($x_n$)-n`).
- ✗ **Faire tourner son code avant envoi** !!!

Références

[1] Malcolm E. Lines, *Dites un chiffre*, Flammarion, 2002.