TP N°1 - Observations sur les rapports de TP rendus et ce qui était attendu.

Laurence LAMOULIE

26 mars 2008

Table des matières

1 Préliminaires

Même si vous ne trouvez pas cette note drôle, lisez là jusqu'au bout : elle vous aidera à comprendre ce qu'on attend de vous, aujourd'hui comme dans un stage ou un emploi.

2 Globalement

Dans l'ensemble les travaux rendus négligent le rapport et en particulier l'analyse du problème.

Il n'a pas semblé clair pour vous que l'analyse numérique consiste à **analyser** des **résultats** obtenus dans un **programme** qui consiste à résoudre un problème que l'on présente dans un **rapport**. Tous ces éléments doivent donc être présents dans votre travail : un programme sans rapport n'a aucun intérêt, pas plus qu'un rapport sans résultats ni discussion. C'est pour cela qu'un modèle de rapport a été mis à votre disposition et que vous devez respecter sa forme sans omettre des paragraphes.

Il est maladroit de présenter un résumé de ce que vous allez faire et de ne pas traiter ensuite ce que vous avez annoncé. Le lecteur n'aime pas les promesses non tenues.

3 Ce que vous devriez déjà savoir

3.1 Equation

On appelle **équation** une expression mathématique qui a un premier et un second membre. En conséquence, un polynôme de degré 3 n'est pas une équation.

3.2 Fonction

Une fonction associe à un élément d'un ensemble de départ, une image dans un ensemble d'arrivée. Par conséquent $x + \sin(bx)$ n'est pas une fonction mais l'image du réel x par la fonction f définie par :

$$f : x \mapsto x + \sin(bx)$$

Si vous voulez introduire une fonction vous devez écrire : "Soit la fonction définie par $x \mapsto x + \sin(bx)$" au minimum.

3.3 Arrondi

L'arrondi et l'arrondissement n'ont rien en commun. L'orthographe du mot n'est pas incertaine, son genre est parfaitement défini.

On peut raisonnablement penser qu'un élève ingénieur en informatique a compris que plus on a de bits, moins on fait d'arrondi, et plus le résultat est précis. En conséquence, le but d'un TP ne peut pas être de répéter ces constatations toutes les deux lignes. Il doit certainement y avoir quelquechose de plus subtil à trouver.

3.4 Orthographe et style

Un ingénieur est censé écrire des rapports qui seront lisibles par des interlocuteurs (supérieurs hiérarchiques, collaborateurs, clients). Il peut arriver que certains d'entre eux sachent encore écrire et parler français. Si vous avez la malchance de tomber sur l'un d'entre eux, il pourrait penser que vos fautes sont le signe d'un crétinisme profond. Attention aux ennemis qui rodent !

Dans un rapport technique, on ne parle pas à la première personne : le "je" est donc à proscrire, au profit de la tournure impersonnelle. Utilisez donc soit le "on", soit des "il est possible de ...". En fait, en tant que rédacteur, d'une part vous représentez votre société ou votre service (et donc votre personne n'est qu'un rouage de la machine, inutile de vous mettre en avant), et d'autre part vous ne faites que retranscrire une vérité scientifique (donc c'est une chose universelle que tout le monde doit accepter, qu'elle vienne de vous ou pas).

3.5 Comparaison de nombres

Vous avez appris un jour, ou du moins vos profs vous l'ont dit un jour, que le corps des complexes n'est pas totalement ordonné. En clair cela veut dire que l'on ne peut pas comparer deux nombres complexes pour dire lequel est le plus grand des deux. On ne peut donc pas non plus calculer une erreur entre deux complexes comme on le fait avec deux réels :

$$e_x = |x_2 - x_1|$$

a un sens si $x_1, x_2 \in \mathbb{R}$. Mais si on a $z_1, z_2 \in \mathbb{C}$ que devient

$$e_z = |z_2 - z_1|$$

? Et bien, c'est pour cela qu'a été inventé le **module** ! Il permet de donner une indication numérique de la distance entre 0 et un nombre complexe, et prolonge la notion de distance dans $\mathbb{R}$.

Par conséquent quand vous voulez comparer des racines réelles d'une équation de degré 2 (dans le cas ou le discriminant Δ est positif) avec des racines complexes (dans le cas ou le discriminant Δ est négatif) il faut passer par le module. J'ai de gros doutes sur la méthode que vous appliquez quand je vois les graphiques que vous faites à ce sujet, et les remarques qui les accompagnent.

Heureusement Scilab n'est pas bête, il a même prévu votre négligeance car "abs(x) est la valeur absolue des éléments de x . Quand x est complexe, abs(x) est le module des éléments de x . " mais ce n'est pas une raison pour ignorer le problème.

3.6 Discret ou continu ?

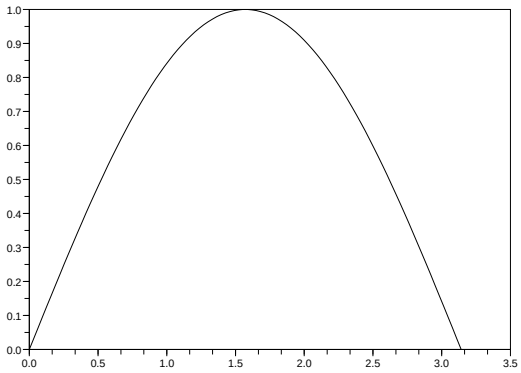
Lorsque vous représentez des valeurs obtenues par un calcul, au moyen d'un graphique, vous devez vous poser des questions.

Scilab ne permet pas de représenter une fonction à partir de sa définition, donc on contourne le problème en générant des valeurs d'abscisses assez proches pour obtenir une courbe continue. Ce n'est qu'une astuce car on a bien en fait un vecteur de valeurs de x et un vecteur de valeurs de y. Il est dans ce cas utile de joindre les points par une courbe, ce que Scilab fait par défaut.

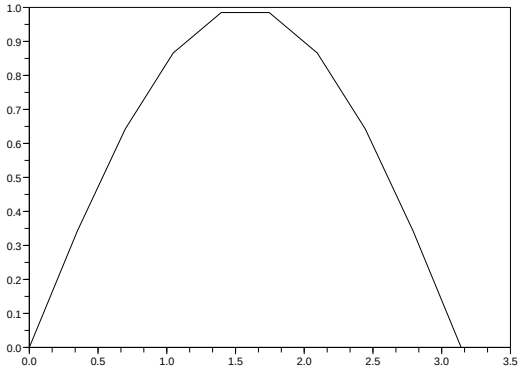
Exemple : On veut représenter la fonction sin sur l'intervalle $[0, \pi]$. On génère n points dans $[0, \pi]$, On calcule leurs images et on représente la courbe réunissant les points. Le code correspondant est donc :

```
function DessineSinus(nb_points)
 xbasc();
 vecteur_abscisses = [0:%pi/(nb_points-1):%pi]; // je génère nb_points valeurs de 0 à Pi
 vecteur_ordonnees = sin(vecteur_abscisses); // je calcule les images des points
 plot2d(vecteur_abscisses,vecteur_ordonnees)
endfunction
```

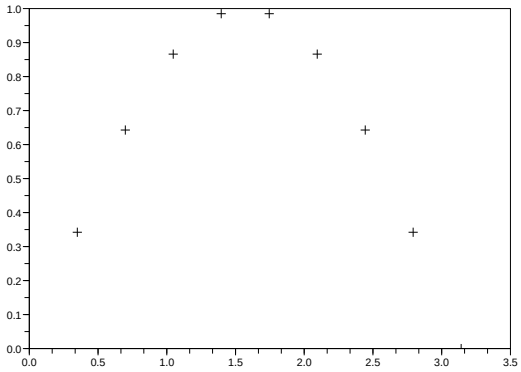
Si on prend assez de points (par exemple 100), la courbe obtenue est :



Si on ne prend pas assez de points (par exemple 10), la courbe obtenue devient :

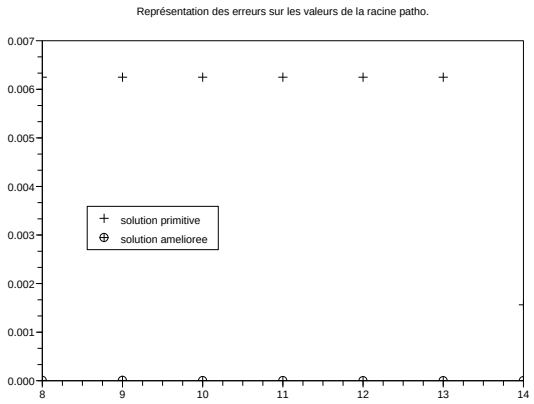


On voit bien que scilab n'a en fait que relié des points, ceux que l'on aurait pu représenter sans les relier avec la commande `plot2d(vecteur_abscisses,vecteur_ordonnees, -1)` pour obtenir :

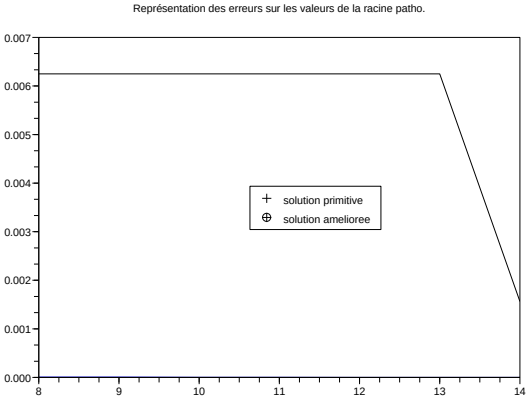


Mais dans certains cas, si l'on veut représenter un ensemble de **points** et non pas une fonction, il faut afficher un nuage de points et non pas une courbe continue. Chaque point représente une mesure, et **les relier n'a**

aucun sens. Par exemple, pour des nombres de bits entiers compris entre 8 et 14, représenter l'erreur obtenue avec des méthodes différentes doit donner un dessin du genre :



et pas :



.

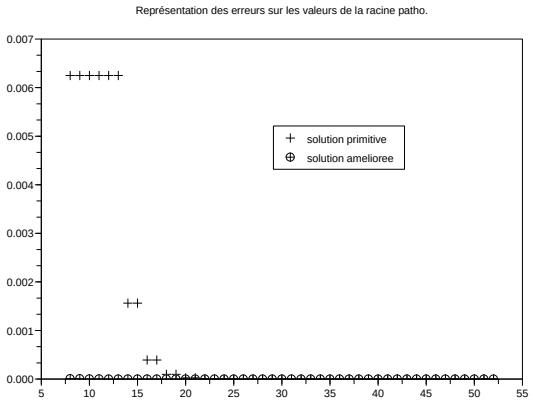
Le second graphique est illisible et on ne sait même plus pour quelles valeurs ont été réalisées les mesures. Quant au sens à donner à 8,2 bits...

Le choix du mode de représentation d'une courbe n'est donc pas à laisser au hasard, ni à scilab d'ailleurs. **Vous devez choisir une courbe continue ou un nuage de points, et insérer les instructions adéquates dans le code.**

Bien sûr, dans ces deux cas, le problème du choix de la fenêtre de représentation demeure. C'est un facteur important, que vous devez régler par exemple par l'option `rect[xmin, ymin, xmax, ymax]` de `plot2d` qui permet de définir les coordonnées de cette fenêtre.

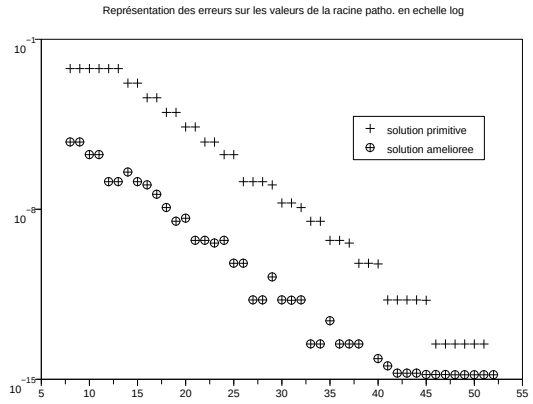
3.7 Une échelle adéquate

Lorsque vous représentez sur un graphique des valeurs d'ordres de grandeurs différents, le résultat peut se révéler illisible pour tout ou partie des valeurs (même en tenant compte du caractère discret du graphique). Par exemple :



. Le fait que certaines valeurs semblent presque nulles n'est pas suffisamment précis, il faut savoir si elles sont de l'ordre de 10^{-4} ou de l'ordre de 10^{-14}. Donc, **ne vous appuyez pas sur un graphique mal fait pour déduire des commentaires idiots** du genre "c'est plus petit qu'avant et c'est presque nul".

Si on ne voit rien sur votre graphique ce n'est pas forcément qu'il n'y a rien à voir. **Changez d'échelle** : cela peut souvent améliorer les choses. Par exemple le graphique précédent en échelle logarithmique pour les ordonnées donne :



.

4 Sur l'analyse

C'est dans cette partie que vous exposez le problème et les différents points que vous allez programmer. Vous devez également interpréter les indications qui vous sont données dans le texte de TP, et exposer à quoi elles correspondent. Par exemple si on vous demande une autre méthode pour trouver les racines d'une équation, en quoi est-elle utile, et à quoi voit-on qu'elle est efficace..

Si vous ne faites pas d'analyse cela signifie que vous programmez une méthode que vous ne comprenez peut-être pas. Dans ce cas vous ne serez pas à même de discuter ses résultats, puisque vous ne savez pas à quoi vous devez vous attendre, ni si ce que vous obtenez est logique ou pas.

5 Sur les programmes

Je vous rappelle que vous avez suivi un cours d'Algorithmique et Programmation, que j'ai consulté. Dans ce cours sont données un certain nombre de consignes pour la programmation, dont :

- indentation,
- choix des noms de variables et format du nom,
- en-têtes et commentaires dans le code,

- modularité,...

Ces consignes ne sont pas réservées aux cours d'informatique, il faut les appliquer quel que soit le contexte de programmation. Certains d'entre vous les ont complètement oubliées, rendant un code peu ou pas lisible, peu ou pas réutilisable, peu ou pas indenté, des variables aux noms mal choisis ou de signification douteuse...

Croyez-vous que vos responsables de stage dans l'industrie vont passer leur temps à vous rappeler des choses aussi simples ? Ce sont des normes de programmation que tout le monde se doit d'appliquer, donc je pénalise ceux qui ne le font pas.

Les programmes doivent être facilement appellables si le lecteur veut les utiliser. Cela suppose que vous donniez (dans le rapport) au moins le prototype de la fonction de plus haut niveau (celle qui appelle toutes les autres) et la signification de ses arguments, en entrée comme en sortie. Normalement vous devez aussi commenter votre code de façon à donner la signification de tous les arguments, surtout si les noms des variables ne sont pas suffisamment explicites. Cela n'est pas seulement utile pour le lecteur, c'est aussi essentiel pour la réutilisabilité des fonctions.

Concernant la réutilisabilité, elle implique que vous devez exclure toutes les lignes de code qui ne sont pas incluses dans des fonctions. Tout ce qui est codé doit faire partie d'une fonction (Initialisation, résolution, calcul d'erreur...) ou du programme principal (qui est lui-même une fonction).

Attention à toutes les instructions de type `while` (notamment pour le TP2) : vous devez toujours prévoir une condition qui assurera une sortie de la boucle. Vous omettez parfois ce point important : une erreur sur un procédé itératif peut ne pas décroitre (cela s'appelle la divergence !) et dans ce cas vous allez rester combien de temps à attendre le résultat ? Il est simple d'arrêter les itérations lorsque leur nombre devient trop élevé, le seuil est à choisir proportionnel à la taille de la matrice dans le cas de la méthode de Jacobi.

6 Sur les résultats

On attend de vous des résultats, ce qui suppose :

1. des instructions qui impriment quelquechose dans votre code, soit à l'écran soit dans un fichier (c'est mieux puisqu'on peut le conserver et le traiter sous excel par exemple).
2. une présentation dans le rapport de données **chiffrées** concernant par exemple le détail des valeurs d'erreur lorsqu'on atteint des grands nombres de bits, ou la différence entre les valeurs des racines d'une équation selon la méthode utilisée.
3. des graphiques lisibles (cf paragraphe ci-dessus).

On ne vous demande pas si vous trouvez que le résultat est *joli*, *rapide*, *décevant*, *surprenant* et j'en passe. On vous demande *combien* on a trouvé, pour que cette valeur soit comparable à d'autres dans des contextes différents.

Un rapport sans résultats n'apprend rien au lecteur, et son rôle n'est pas de faire tourner votre code pour savoir ce qu'on trouve, c'est à vous de le présenter.

7 Sur la discussion

On attend de vous une attitude critique par rapport aux résultats, c'est à dire que vous devez :

- présenter les déductions que vous faites des résultats,
- les analyser pour savoir si elles sont cohérentes entre elles et conformes à la théorie (que vous avez présentée dans l'analyse).

Votre discussion ne doit pas mentionner les difficultés que vous avez rencontrées pour en arriver là : cela signifie que vous avez perdu du temps donc que vous n'êtes pas un ingénieur rentable (et qui en voudra de celui-là ?).

Vous pouvez éventuellement critiquer votre propre travail, mais toujours en proposant une alternative pour l'améliorer. Vous pouvez mentionner les points qui seraient intéressants à creuser au vu de ce que vous avez fait, mais qui n'étaient pas demandés au départ, ou que vous n'avez pas eu le temps de traiter.

8 Sur les conclusions

Les conclusions doivent pouvoir être lues sans avoir pris connaissance de ce qui précède. Elles sont un résumé de tout votre rapport et doivent éclairer le lecteur sur l'intérêt de la méthode programmée. Vous devez dire si c'est une méthode efficace, stable, facile à implémenter, dont les paramètres peuvent être optimisées, etc. Vous pouvez conseiller des méthodes plus efficaces le cas échéant, en citant vos sources.

Attention si vous suggérez ce qu'on aurait pu faire de plus, car la remarque est immédiate : pourquoi ne l'avez vous pas fait ?

On ne vous demande pas si vous avez eu du mal ou si le TP vous a intéressé : vos états d'âme n'intéressent personne, et si vous avez peiné c'est que vous n'êtes pas bon, donc inutile de donner le bâton pour vous faire battre.

Même si ce n'est pas facile, essayez de vous imaginer dans la situation d'un ingénieur qui produit un rapport destiné à présenter le fruit de son travail. Il doit être argumenté, objectif, efficace. Le lecteur doit comprendre ce qu'il écrit sans lire son code, mais il doit également pouvoir le faire tourner s'il veut vérifier les résultats.

J'espère que ces remarques vous aideront à comprendre ce que l'on attend de vous et dans quel esprit j'ai examiné votre travail.