Sur quelques obstacles inattendus apportés par Julia et par l'automatisation de certaines tâches de programmation

Daniel Porumbel[1]

Conservatoire National des Arts et Métiers

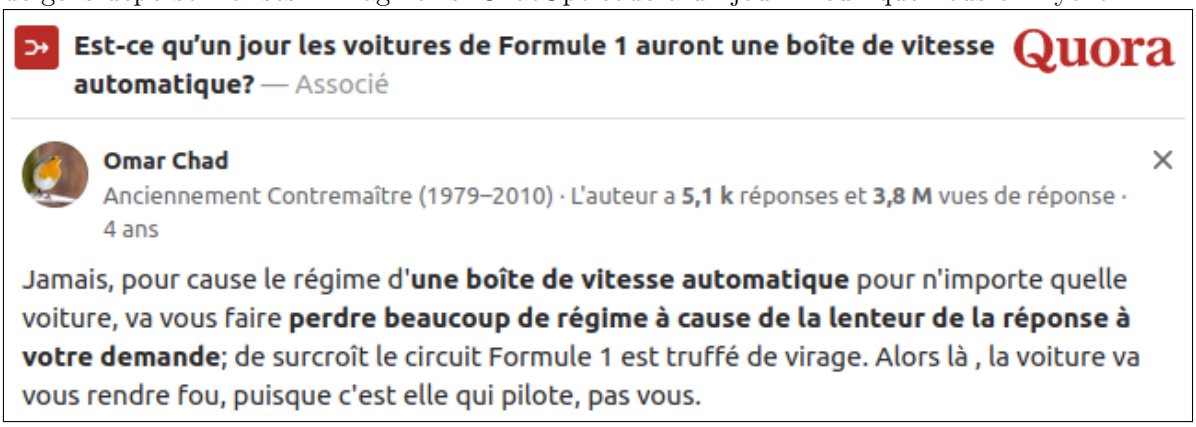
daniel.porumbel@cnam.fr

Mots-clés : *langages de programmation en RO, Julia, C++*

1 Introduction, synthèse et conclusion

Ma principale raison pour rejeter Julia, Matlab ou Python est la suivante : je veux savoir plus-ou-moins ce qui se passe lorsqu'un bloc de code est exécuté. Même le C++ peut me déplaire à cet égard, par exemple si on utilise trop d'héritage en cascade. Je me sens plus à l'aise avec un « C-like C++ ». Une telle préférence peut être défendue avec des arguments d'efficacité, mais c'est d'abord une question psychologique. S'il y a trop de couches intermédiaires dans la chaine de commande, je sens que je commence à perdre le contrôle de ma machine ; je ne la conduit plus tout seul. Je vous l'accorde : personne ne sait exactement ce qui se passe dans la tête d'une machine, car on ne peut pas la programmer en assembleur. Mais on doit lui limiter les caprices et les réactions inattendues, d'où un besoin d'ordre et de certitudes pour minimiser la confusion. Si vous avez peur de déclarer n comme variable globale pour ne pas plomber l'efficacité, vous avez peur des caprices de Julia. Les caprices d'efficacité ne seront jamais étonnants tant que travaillerez avec un mille-feuille logiciel. Ces milles couches logicielles seront toujours là, car ces langages très modernes feront le maximum pour rendre efficace même le code le plus superficiel, écrit à la légère sans trop connaître les structures utilisées derrière.

Il n'y a vraiment aucun problème si beaucoup sont conquis par ces nouveaux langages. C'est seulement la consommation massive et excessive qui risque de créer des désordres, en poussant le monde vers la légèreté. Par exemple, les profs de types ou de structures de données auront bientôt du mal à trouver des gens pour les écouter. Le principe « il faut savoir les utiliser mais c'est pas essentiel de savoir comment ça fonctionne à l'intérieur » prend de plus en plus de poids. Si on délègue trop de tâches à des machines, on risque de se retrouver un jour avec trop de gens dépersonnalisés – imaginez si ChatGpt codera un jour mieux que nous en Python.



Ce texte n'a pas été écrit par un médecin qui utilise ses mots au sens clinique. Il a juste peut-être vu un phénomène bizarre : la machine commence à effacer la personnalité humaine. Tu peut te faire abrutir par la machine ou le système que t'as fait pour te déplacer. Aujourd'hui on peut dire « un tel conduit sa voiture de telle ou telle manière, un autre la conduit d'une

autre manière, etc ». Au fur et à mesure qu'on permet à la voiture de prendre de plus en plus de décisions (aussi efficaces soit elles), les individus vont conduire de plus en plus de la même manière, *i.e.*, ils seront de plus en plus robotisés.

Voici un exemple de travail de RO : (1) prendre un problème de logistique portuaire ; (2) écrire un modèle linéaire ; (3) coder ce modèle en Julia (4) appliquer la décomposition de Benders et faire apparaitre un polytope ; (5) lancer Julia et produire de belles figures. Pour l'étape (1), ChatGpt pourra trouver un problème à résoudre sans aucune difficulté : il suffit de brasser le contenu des communications des éditions précédentes de la Roadef. Il est possible d'automatiser l'étape (2) si le problème choisi est bien posé avec des données bien quantifiés. L'étape (3) était à il y 10 ans un peu un casse-tête, mais (merci Julia et JuMP !) cela est désormais une banalité. L'étape (4) pourrait s'avérer difficile, mais aujourd'hui (merci Cplex 12.9 !) on peut faire la décomposition de Benders automatiquement. L'étape (5) demandait à une époque une certaine maitrise des formats des figures (vectoriels, raster, etc), mais je suis sûr que les nouveaux paquetages Julia peuvent sortir de très belles figures d'une manière « effortless ».

Je crains qu'on peut trouver sur cette Terre des pratiques scientifiques bien plus propres à fausser les problèmes qu'à les résoudre. Ce phénomène peut exister en science comme dans toute autre domaine de réflexion humaine. Tu peux bien croire en la science, mais tu n'es pas plus à l'abri que ceux qui croient d'autres choses. Ce n'est pas parce que quelqu'un vous parle en termes de théorèmes vérifiables, x ou y, « **for** i », « **for** j » ou `model.solve()`, qu'il ne peut pas vous mystifier.

Si on avait donné il y a 10 ans le projet de RO plus haut à 40 étudiants, on aurait pu se retrouver avec une douzaine de restitutions relativement différentes, car chaque étudiant aurait pu suivre les consignes à sa sauce, selon les tendances de son esprit. Mais aujourd'hui, grâce aux nouvelles technologies ci-dessus, on risque de se retrouver avec 40 étudiants qui donnent plus ou moins la même réponse un peu standard, tous conquis et étonnés par l'efficacité des outils d'automatisation. Au niveau plus global, j'imagine que beaucoup d'étudiants (un peu partout) peuvent commencer aujourd'hui les TPs avec la solution fournie par ChatGpt qu'ils suffit de modifier un peu pour la faire mieux coller aux consignes. Pareil pour traduire un texte. Ceux qui ne font que ça seront peut-être contents d'avoir un diplôme plus facilement. Mais au passage ils seront abrutis par la machine, incapables de développer leur esprit, incapable de trouver leur différence. Je ne m'en réjouis pas, je ne fait qu'avertir autant que j'ai pu comprendre.

Je change de sujet. Une autre raison d'antipathie vis à vis de Julia vient de l'expérience personnelle. Imaginez un touriste qui visite trois fois une ville très appréciée par tout le monde et qui trouve à chaque fois plus d'obstacles que prévus. Est-ce qu'il a le droit de rester superficiel aussi ? Doit il faire une étude vraiment très poussée de la ville avant de penser aller ailleurs ? L'industrie hôtelière devrait elle commencer à dresser une liste de vertus de la ville pour remettre le touriste perdu sur le bon chemin ? Si votre réponse à ces deux questions est « Oui », évitez cet exposé à tout prix. Je ne mets plus d'effort dans ce texte, car un lecteur pressé pourra penser que je m'acharne à faire un travail négatif. Je veux juste partager avec Monsieur Tout-le-Monde une autre vision sur la programmation en RO, présentée par un « lay man » – un homme plus ou moins capable de réaliser diverses tâches mais qui ne cherche pas à suivre les grandes politiques de programmation de telle ou telle école de pensée. Je n'ai pas écouté les discours des grandes sommités du domaine : j'ai dû apprendre sur le tas ; je n'ai pas eu le temps.