MARÇAL
THOMAS

Groupe : P

Projet IA : Star Wars

1. Quelle est la taille de l'espace de recherche (utiliser une notation scientifique) ?

La taille de l'espace de recherche est :

$$\boldsymbol{p_{i\in\{1,..,6\}}} \in [\mathbf{-100\,;\,100}]^{\mathbf{6}}$$

2. Quelle est votre fonction fitness ?

J'ai choisi la **norme euclidienne au carré**, en tant que fonction fitness. Cette norme est utilisée pour définir la distance entre deux points dans un plan.

Ainsi, j'applique cette norme à la différence entre les points théoriques et ceux calculés, afin d'établir la moyenne (à savoir, nous disposions de 30 points théoriques dans le document .CSV), ce qui donne :

$$\left[\sum_{\boldsymbol{i=0}}^{\mathbf{30}} ([\boldsymbol{X_i} - (\boldsymbol{p_{1\,Calc}} * \boldsymbol{sin}(\boldsymbol{p_{2\,Calc}} * \boldsymbol{T_i} + \boldsymbol{p_{3\,Calc}}))]^{\mathbf{2}} + [\boldsymbol{Y_i} - (\boldsymbol{p_{4\,Calc}} * \boldsymbol{np.\sin}(\boldsymbol{p_{5\,Calc}} * \boldsymbol{T_i} + \boldsymbol{p_{6\,Calc}}))]^{\mathbf{2}})\right] \mathbf{/30}$$

En conséquence, plus cette fonction fitness est minimale, plus la précision de nos solutions se rapprochera de la réalité et sera stable.

3. Décrivez les opérateurs mis en œuvre (mutation, croissement) ?

D'une part, s'agissant de la **mutation**, j'ai décidé de remplacer une valeur aléatoire de mon inconnu. Dès lors, pour un indice aléatoire *i* de mon individu (donc un nombre entier compris entre 1 et 6), je modifie la $i^{ème}$ valeur de mon inconnu, selon la précision de ma fonction fitness à l'instant *t*.

Ainsi, si le système ne converge pas vers une solution possible, c'est-à-dire que la fonction fitness n'est pas assez bien respectée. Je remplace le $i^{ème}$ élément par une valeur comprise dans l'intervalle [-100 ; 100], aux fins que l'algorithme génétique ne se bloque pas sur des valeurs.

Puis, lorsque la convergence des solutions est entamée, autrement dit la fonction fitness est assez bien respectée. J'affine ma recherche, en remplaçant cette fois-ci le $i^{ème}$ élément par une valeur dans l'intervalle [val[i] – 0.1 ; val[i] + 0.1]. Il résulte que j'obtiendrai ainsi une convergence plus rapide avec une précision meilleure dans les résultats voulus.

D'une autre part, s'agissant du **croisement**, j'ai pris la décision qu'à partir de deux individus, mis en argument, j'échangerais leurs données.

En somme, les 3 premières données d'un individu 1 soient suivies des 3 dernières de l'individu 2 ; et inversement, les 3 premières données de l'individu 2 soient suivies des 3 dernières données de l'individu 1.

4. Décrivez votre processus de sélection.

J'ai d'abord évalué ma population selon la fonction fitness, puis j'ai appliqué le processus de sélection.

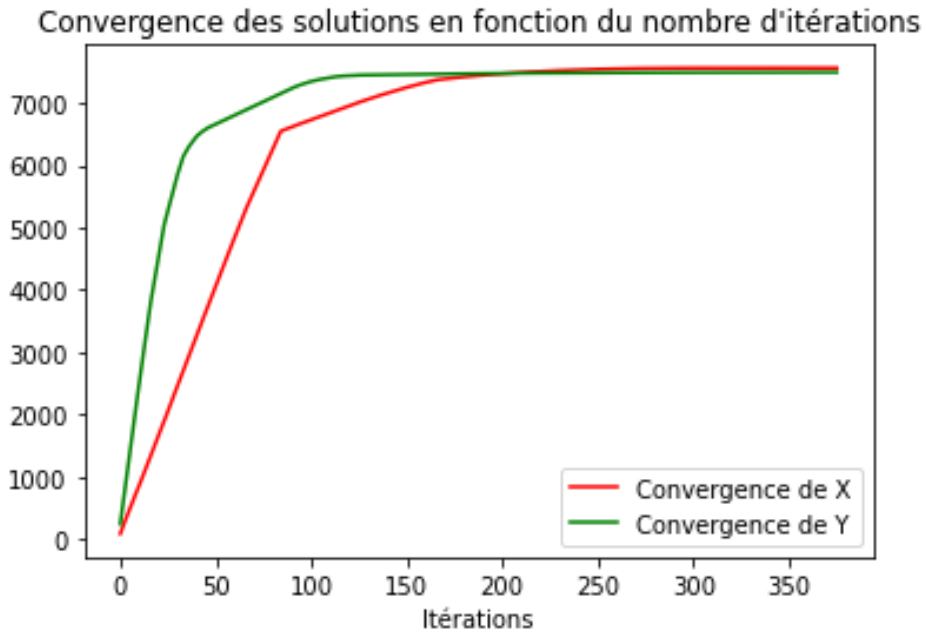
Ainsi, partant de la population évaluée, j'ai renvoyé les *n* premiers éléments de cette même population, compte tenu qu'ils correspondent aux individus respectant le mieux la fonction fitness. Mais également, les *m* derniers éléments ($n>m$), qui respectent quant à eux le moins bien la fonction fitness.

Ce mécanisme m'a permis d'obtenir un éventail large de valeurs, qui permet de répondre au mieux à l'application de cet algorithme.

5. Quel est la taille de votre population, combien de générations sont nécessaires avant de converger vers une solution stable ?

J'ai pris une population de taille 150.

Afin de montrer combien de générations sont nécessaires avant de converger vers une solution stable, j'ai élaboré une fonction Python *Convergence* qui permet d'illustrer cet objectif.



Et, d'après le graphique des convergences de X et Y, pour un epsilon valant 0.04, il est nécessaire d'avoir environ 125 itérations pour que l'algorithme converge vers une solution stable.

6. Combien de temps votre programme prend en moyenne (sur plusieurs runs) ?

Pour un epsilon valant 0.05, on a les temps suivants :

4.021	5.991	5.917	6.060	4.459	7.631	6.925	5.725	8.142	4.180
3.862	6.162	5.101	6.605	4.835	4.656	6.782	4.262	4.128	4.693

$$Moyenne = \frac{\sum_{i=1}^{20} t_i}{20} = \frac{110.137}{20} = 5.507\ secondes$$

⇨ Sur plusieurs runs, mon programme prend en moyenne **5.057 secondes**.

7. Si vous avez testé différentes solutions qui ont moins bien fonctionnées, d´écrivez-les et discutez-les.

Dans un premier temps, dans le cadre de la mutation, je n'effectuais pas de conditions *if/else*. Dès lors, je ne pouvais pas affiner le remplacement de l'élément aléatoirement sélectionné de mon inconnu, par rapport à

sa valeur initiale. Mon algorithme mettait alors plus de temps à converger vers une solution stable, dans un intervalle de 2 minutes, au lieu de 10 secondes rendues possibles après ce changement.

En effet, avant cette modification, lors de la mutation, le travail préalablement effectué devait se répéter ; d'où la mise en place d'une valeur limite, qui correspond à la fonction fitness de l'inconnu de ma population respectant au mieux ce critère, et qui permet que si la fitness est suffisamment respectée d'affiner, ou non, le critère de mutation.

Dans un second temps, dans le cadre de la fonction fitness, j'avais opté comme première approche pour une norme première :

$$\left[\sum_{i=0}^{30}(\boldsymbol{abs}\left(\left[X_i - \left(p_{1\,Calc} * \boldsymbol{sin}(p_{2\,Calc} * T_i + p_{3\,Calc})\right)\right]\right) + \boldsymbol{abs}([Y_i - (p_{4\,Calc} * \boldsymbol{np}.\sin(p_{5\,Calc} * T_i + p_{6\,Calc}))])\right] / 30$$

Or, l'application de cette norme ne permettait pas une convergence aussi précise et rapide que la norme euclidienne. En effet, pour a un réel positif, $\|x\|_2 \leq a^*\|x\|_1$ pour tout vecteur x de E, un espace euclidien. La norme euclidienne, ou seconde est donc plus précise.