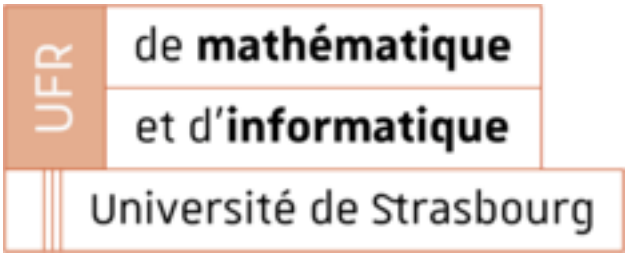
Systèmes Complexes et Optimisation Stochastique Massivement Parallèle

Rapport de Projet

Equipe:
ALLEMAND Fabien
LEBOT Samuel



Contents

List of Figures

1 Introduction

Evolutionary algorithms and genetic programming are used in a wide range of applications. From basic applications in industry and engineering to complex problem resolution in resarch department, these two methods, whose development started with Alan Turing in 1950, no longer need to prove their efficiency.

On the one hand, evolutionary algorithms, inspired by biological evolution, allows us to find quatities or caracteristics of an object, and on the other hand, genetic programming was designed to find interactions between objects. By combining these two methods, it is theoredicaly possible (assuming there is a good way to evaluate our results) to find not only caracteritics defining two entities, but also the way they interact with each other. These two entities could very well be daily life objects or even animals. We could actually be able to find how birds interact with each other in a flock. These two entites could be less concrete and be ridiculously small: let us find out how electrons interact in matter! But they could also be enormous like celestial bodies...

Let us go back in time and imagine we only have the knowledge and tools to measure the position of the Sun relatively to the Earth. Would we be able to find caracteritics about the Earth or the Sun like their mass or their speed? Is there a way to find out how they interact?

The goal of this project is to utilise the two afore mentionned programming methods to retrieve caracteristics and laws regarding the interactions between two celestial bodies.
We will focus on Earth and Sun as there is a strong interaction between them and yet are comletely different. First, we will conduct a physics study regarding the celestial bodies in order to evaluate results to come. In a second time, we will attempt to find caracteristics about the two planets using evolutionary algorithms. Finally, we will use genetic programming to re-discover Newton's law of universal gravitation.

GitHub repository: `https://github.com/FABallemand/ProjetSystemesComplexesOptimisationStochastiqueMassive`

2 Sampling

Later in ou study, we will use the trajectory of the Earth in the heliocentric renferential in order to evaluate our models. To be more precise, the score of a model will be the difference between the real Earth's trajectory and the trajectory we estimate with the values or functions we find.
In order to do that, we need to be able to sample the Earth path in that referential. Let us study the Earth in the heliocentric renferential.

System: Earth, assimilated to a material point of mass M_T
Referential: Galilean assumed heliocentric renferential
Coordinates System: Polar coordinates
Balance of forces: Attraction force of the Sun of mass M_S: $\vec{F} = -G\frac{M_T M_S}{r^2}\vec{e_r}$

According to Newton's second law:

$$M_T\vec{(a)} = \vec{F} \iff \begin{cases} M_T(\ddot{r} - r\dot{\theta}^2) = -G\frac{M_T M_S}{r^2} \\ M_T(2\dot{r}\dot{\theta} + r\ddot{\theta}) = 0 \end{cases} \quad (1) \quad \iff \begin{cases} \ddot{r} = r\dot{\theta}^2 - G\frac{M_T M_S}{r^2} \\ \ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r} \end{cases} \quad (2)$$

In order to sample the Earth's trajectory, we perform two Euler methods simultaneously: the first one allows us to obtain the speed and the second one gives us the actual position.

$$\begin{cases} r_{n+1} = r_n + \dot{r_n}\delta t \\ \theta_{n+1} = \theta_n + \dot{\theta_n}\delta t \end{cases} \quad (3)$$

$$\begin{cases} \dot{r_{n+1}} = \dot{r_n} + \ddot{r_n}\delta t \\ \dot{\theta_{n+1}} = \dot{\theta_n} + \ddot{\theta_n}\delta t \end{cases} \quad (4)$$

with $\delta t = \frac{365{,}25\times 24\times 3600}{1024}$ as we want to sample 1024 points.
We also have to set the initial conditions. Let us suppose we start to sample when the Earth is located at the perihelion, as the distance between the Earth and the Sun and the speed of the Earth at this point are known, we obtain the following conditions:

$$\begin{cases} r_0 = 147,1 \times 10^9 \\ \theta_0 = \pi \\ \dot{r_0} = 0 \\ \dot{\theta_0} = \frac{2\pi\times 30{,}2\times 10^3}{\pi\left(3(a+b)-\sqrt{(3a+b)\times(a+3b)}\right)} \end{cases} \quad (5)$$

with $a = 1521,0 \times 10^8$ and $b = 1471,0 \times 10^8$ the distances between the Earth and the Sun respectively at the aphelion and at the perihelion.
With this set of equations we can write a program that precisely compute 1024 positions of the Earth while orbiting around the Sun (Figure 1).

3 Evolutionary Algorithm

4 evol

The first part of this project consisted in finding three quantities that influence the orbit of Earth around Sun: the mass of Sun, the mass of Earth and the speed of Earth at the perihelion, using the evolutionary algorithms implemented in EASEA.

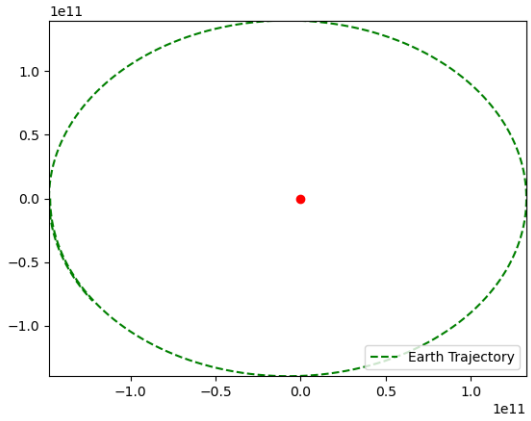


Figure 1: Earth trajectory around the Sun (1024 points sampling)

$$error = \sum_{i=0}^{n} dist(p_i, \hat{p_i})$$

where:

- n denotes the number of samples (typically 1024)
- p_i corresponds to the i-th sampled point on the real Earth's trajectory
- $\hat{p_i}$ correspond to the i-th sampled point on the estimated Earth's trajectory
- $dist$ is the Euclidian distance

Figure 2: Expression of the error for evolutionary algorithms

4.1 Method

In order to find the three physics quantities describing the system, we performed three distinct experiences because the sampling method does not allow us to find these three quantities simultaneously. It is impossible to find the mass of Earth and Sun at the same time because they are only bound in a product in Newton's law of universal gravitation. So there are a plethora of pairs of values leading to the same result.

The first step consist in sampling the Earth orbit around Sun with the real quantities in order to obtain a list of polar coordinates of Earth in the heliocentric referential.
The genom of each individual correspond to the quantity to find (the mass or the speed) and is initialised to a value of the same order of magnitude as the real value.
From one generation to another, children are created by computing the mean of the two parents. Each individual can mutate with a given probability. This mutation correspond to an increase or a decrease of a given percentage of its genom. This percentage is randomly selected in a fixed interval.
The score is defined as the difference between the real orbit and the orbit with the quantity found in the genoms. That is to say: the sum of the distances between corresponding points from the two trajectories (Figure 2). By doing so, the result of the algorithm does not directly depend on the value we want to find but rather on samples of coordinates that could have been made without knowing the quantity.
We settled for weak elitism, a form of elitism that ensures the best individual of the global population (children and parents) is conserved in the next generation.

```
================================================================================
SUN MASS: 19883999999999999986508702416896.000000 (real),
    19883999999999999986508702416896.000000 (estimated)
SCORE: 0.000000
================================================================================
SUN MASS: 19883999999999999986508702416896.000000 (real),
    20781194869844941721635934699 52.000000 (estimated)
SCORE: 41697618869063.750000
================================================================================
     59          1.430s            6000            6000 0.000000000e+00 3.0e
    +11 3.0e+12 3.0e+13
EASEA LOG [INFO]: Seed: 1675245974
EASEA LOG [INFO]: Best fitness: 0
EASEA LOG [INFO]: Elapsed time: 1.4326
```

Figure 3: Output of the EASEA program when seeking the value of Sun's mass

Number of generations	60
Population size	100
Offspring size	100%
Mutation probability	0.2
Mutation variation rate	0.05

Table 1: Parameters used to find the mass of Sun

4.2 Mass of Sun

Using the method previously described, we managed to find the mass of the Sun with 100% accuracy (Figure 3). As we can see on Figure 4, the trajectory of the Earth around the Sun slowly approach the real world trajectory over generations.

The results are quite impresive given it required less than a hundred generations with only a hundred individuals (Table 1).

4.3 Mass of Earth

We did not managed to find Earth's mass using evolutionary algorithm as this quantity does not appear in the equations defining Earth's trajectory around Sun. Nonetheless, we have a couple ideas regarding how we could achieve this.
The first being to simply change the referential: use the geocentric referential in order to study the movement of Sun. If we only consider those two planets (Earth and Sun) then their movements are relative to each other, that is to say: Earth is orbiting Sun and Sun is also orbiting Earth. In that case, we can estimate the mass of Earth exactly the same way as we did in the previous part for the mass of Sun.
Another way to get to that result would be to study those two celestial bodies in a referential centered on the center of Sun's trajectory. But in this case, we would have to study the movement of Earth is a non-galilean referential which is clearly out of the scope of this study.

4.4 Speed of Earth at Perihelion

In order to find the speed of planet Earth at the perihelion we apply the same steps as before: initialisation, crossover, mutation and evaluation. Unless this time we have to change a couples of values. As the speed of Earth in radian per second is so little, it is not well handled by our program (i.e. assimilated to zero). To combat that, we simply consider the speed in radian per day leading to values that can properly be handled by computers. This implies also converting the gravitationnal constant and our sampling step to fit the new

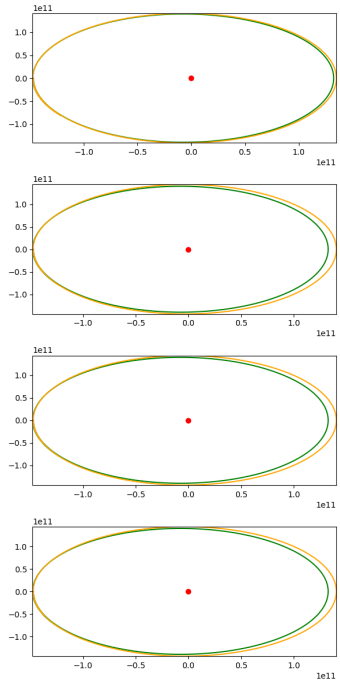


Figure 4: Earth's trajectory over generations during the evolution of Sun's mass

Number of generations	100
Population size	150
Offspring size	100%
Mutation probability	0.2
Mutation variation rate	0.05

Table 2: Parameters used to find the speed of Earth at the perihelion

unit scale.

Once again, we obtained very high accuracy results (Figure 5) and we clearly see how the trajectory evolves throughout the generations (Figure 6). The results are also really impressive given the parameters we used (Table 2). The amount of generations and the population size had to be increased compared to the first experiment, however the mutation probability and rate are still low.

Playing around with the paramters reveals interesting results. On Figure 7, corresponding to a evolution with very low mutation probability and mutation rate (typically 1%), we clearly see that the evolution converges at a slower pace: there is basically no change accross the evolution meaning the initial diversity of the population is not enough to quickly get to the result. Increasing the mutation's paramters show significantly better results. Figure 8 prooves that mutation generates diversity in the population leading to a faster evolution. The two trajectories are not yet stacked but the best individual dramatically improved during ten generations with a really small population (5 individuals). But Figure 9 shows the limits of the mutation operator. By setting a mutation probability of 90% and allowing a mutation to modify the individual up to 90%, the population struggles to converge in the right direction. The impact of the mutation on the genome is too important, hence, we observe oscillations: instead of approaching the real trajectory from one side, the estimated trajectory will reach it changing side when mutation has a strong effect.

This part of the project demonstrates that evoltuion algorithms are very power tools when handled properly. As long as the fitness value is meaningful and the parameters allows the population to converge in the given amount of generations —that is to say there is enough genetic variety (population size) and good

```
================================================================================
PERIHELION SPEED: 0.017440 (real), 0.017440 (estimated), [1.000000]
SCORE: 0.000000
================================================================================
        99           3.397s            15000            15000 0.000000000e+00 1.6
   e+11 1.3e+12 1.5e+13
EASEA LOG [INFO]: Seed: 1675247174
EASEA LOG [INFO]: Best fitness: 0
EASEA LOG [INFO]: Elapsed time: 3.39933
```

Figure 5: Output of the EASEA pogram when seeking the value the speed at the perihelion

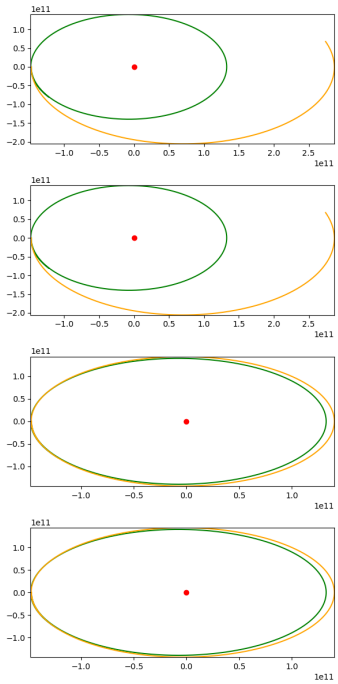


Figure 6: Earth's trajectory over generations during the evolution of Earth's speed at perihelion

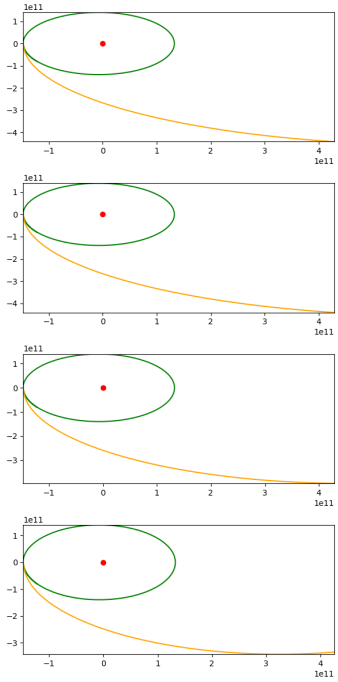


Figure 7: Earth's trajectory over generations during the evolution of Earth's speed at perihelion (low mutation probability and rate)

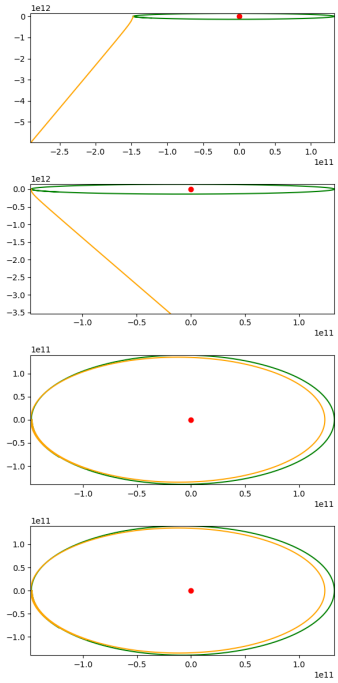


Figure 8: Earth's trajectory over generations during the evolution of Earth's speed at perihelion (balanced mutation probability and rate)

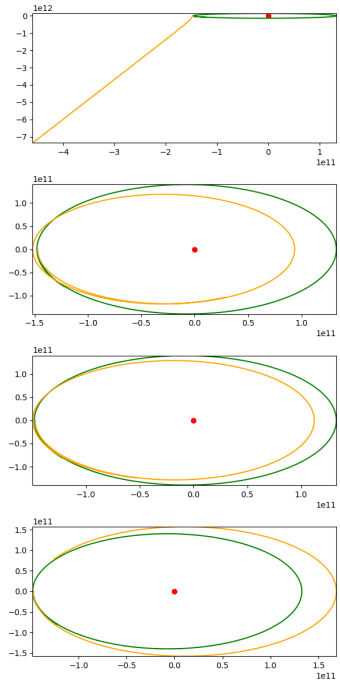


Figure 9: Earth's trajectory over generations during the evolution of Earth's speed at perihelion (high mutation probability and rate)

genetic operators (efficient mutation operator)—the output should be close to the expected result. In this project the result structure was simple (a single value) but in other more complex tasks the output might look a lot different and a bit more analysis, involving a good amount of critical thinking, must be performed in order to evaluate the correctness of the result.

5 Genetic Programming

In the second part of the project, we had to find Newton's law of universal gravitation using genetic programming.

5.1 Fitness Measurement

Our original plan was to sample the Earth's trajectory only once with the target law, that is to say Newton's law of universal gravitation. Then to compute the Earth's trajectory for each new individual (i.e. for each new law) produced by the genetic algorithm. The fitness score of each individual would have been the sum of the distances between two corresponding points in the samplings as explained in Figure 2. However, EASEA does not allow the user to call the generated function with "dynamic" arguments, so it was not possible to perform the sampling anew. So we had to abandon this idea.

After some thought, we decided to do only one sampling with the correct universal law of gravitation. Knowing the position of the Earth in the heliocentric referential, we were able to compute the norm of the attractive force existing between Earth and Sun. Instead of computing the sum of the errors between the coordinates like in Section **??**, the error was set to correspond to the average of the differences between the real force and the estimated force at each point of the path (10).

Although this approach worked (as shown in Section **??**), it has a major flaw with compared to our first idea. We need the force at each point to find the target law. In other words: we need to know something that

$$error = \frac{1}{n}\sum_{i=0}^{n} |f(p_i) - \hat{f}(p_i)|$$

where:

- n denotes the number of samples (typically 1024)
- p_i corresponds to the i-th sampled point on the real Earth's trajectory
- f correspond to Newton's law of universal gravitation
- $\hat{f}$ is the estimation of the Newton's law of universal gravitation made by the program

Figure 10: Expression of the error for evolutionary algorithms

is only computable with the real law in order to find the law...

5.2 Newton's law of universal gravitation

?? In order to find Newton's law of universal gravitation using genetic programming we needed to adjust some parameters on top of the fitness measure.

Initially, we gave our individual four binary operators (+,-,*,/), three constants (the mass of the sun, the mass of the earth and the gravitational constant) and one variable (the distance between the two heavenly bodies). The program also had the possibility to generate constants between zero and one.

Figure 11 shows it is possible to find the law albeit at the expense of a long computation time. In fact in order to find the exact law we needed to create a large population (typically 5000 individuals) and let the algorithm works during hundreds of generations (in this instance 500 generations). It is also worth mentionning that the algorithm did not always converged to the correct law but sometimes to laws that look really different (Figure 12) but give

However, given the gigantic scale of our constant, the genetic program created suboptimal individuals with outliers (e.g., adding 0.1 to the final result) without compromising their fitness score, so we decided to disable ERC (constant between 0 and 1).
We also end up disabling the + and - operators to speed up the process and keep simpler formulas.

Due to the rapid convergence of the GP population, we started with 5000 individuals and only 75 generations.

```
495          443.908s          2480000          2480000  0.000000000e+00  1.4e+04
    3.3e+05  8.2e+06
496          445.386s          2485000          2485000  0.000000000e+00  3.7e+03
    1.6e+05  8.2e+06
497          446.839s          2490000          2490000  0.000000000e+00  4.1e+03
    1.7e+05  8.2e+06
498          448.327s          2495000          2495000  0.000000000e+00  4.1e+03
    1.5e+05  8.2e+06
499          449.773s          2500000          2500000  0.000000000e+00  4.1e+03
    1.7e+05  8.2e+06
EASEA LOG [INFO]: Seed: 1676050646
EASEA LOG [INFO]: Best fitness: 0
EASEA LOG [INFO]: Elapsed time: 449.774
((((G)*(s))*(e))/((r)*(r)))
```

Figure 11: Output of the EASEA genetic program (500 generations of 5000 individuals)

```
495          440.582s          2480000          2480000  0.000000000e+00  2.1e+04
    4.1e+05  8.2e+06
496          441.881s          2485000          2485000  0.000000000e+00  5.0e+04
    2.0e+06  1.4e+08
497          443.036s          2490000          2490000  0.000000000e+00  2.9e+04
    4.9e+05  8.2e+06
498          444.144s          2495000          2495000  0.000000000e+00  3.2e+04
    5.1e+05  8.2e+06
499          445.236s          2500000          2500000  0.000000000e+00  3.2e+04
    5.1e+05  8.2e+06
EASEA LOG [INFO]: Seed: 1676051502
EASEA LOG [INFO]: Best fitness: 0
EASEA LOG [INFO]: Elapsed time: 445.236
(((((e)/(e))+((G)+(((G)*(G))+(e))))*((s)*(G)))/((r)*(r)))
```

Figure 12: Output of the EASEA genetic program (500 generations of 5000 individuals)

6 Conclusion

In conclusion, we first used evolutionary algorithms to find the mass of the Earth, the mass of the Sun and the speed at which the Earth revolves around the Sun. Evolutionnary Algotihms proved to be able to solve these problems without any issue and in a short time as long as there is enough genetic diversity in the dataset. This diversity comes from a large initial population and a set of parameters defining the genetic operators.

The second part of the project was dedicated to the recovery of Newton's universal law of gravitation. However, during this part, EASEA prevented us from implementing the solution we found the most relevant. We then had to backtrack and use a different approach to solve the problem. This approach was successful, but it was not as effective as we had originally planned. Nevertheless, we were able to complete the project and obtain valuable information about the process, such as the influence of the parameter on the convergence speed and the quality of the final results.

Now that we know that genetic programming is capable of finding a physical law from simple observations and constants, it would be interesting to see if it is also capable of finding more complex models such as the Navier-Stokes equations, which describe fluid flow, perhaps by taking advantage of the acceleration offered by island parallelism and GPGPUs.