Development of case studies for using the Julia language in mathematics

James Byrne + Valentin Imbach

Supervised by Prof. Stephen Eglen

Funding provided by Cambridge University Press

Purpose of the project

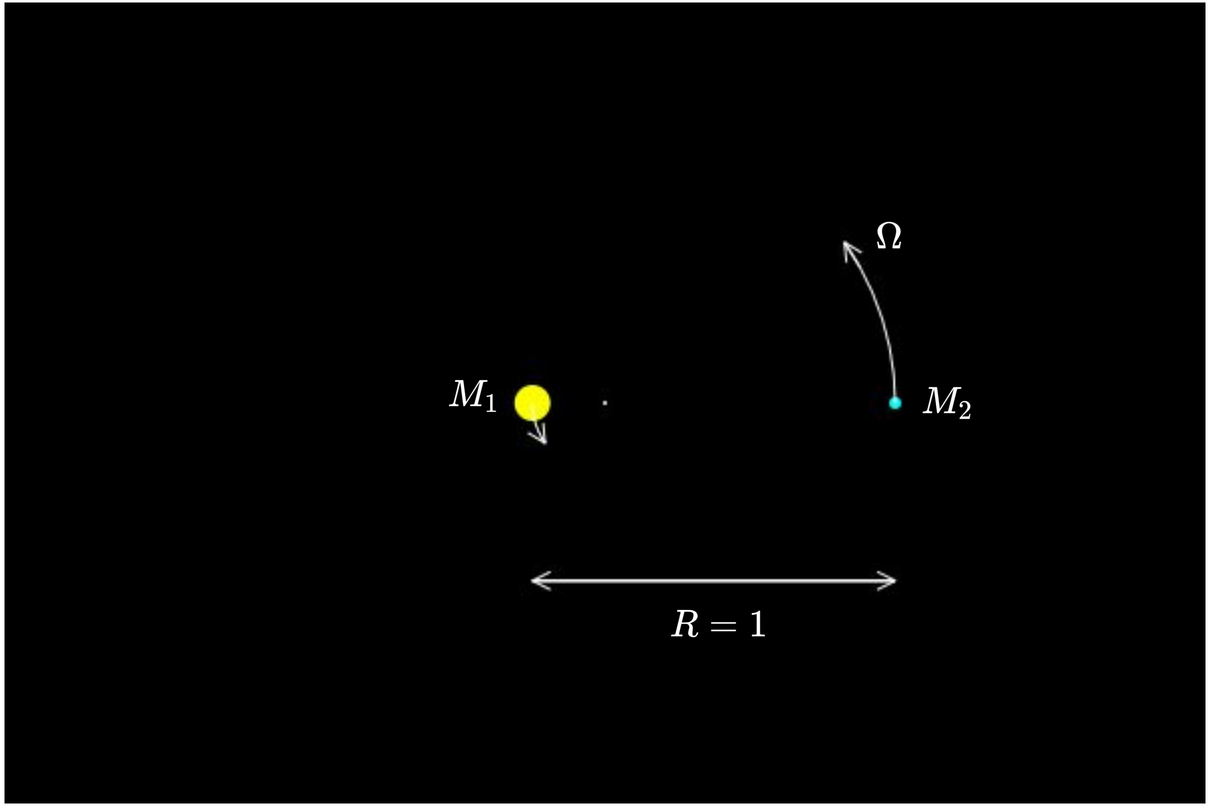
For many second and third year undergraduates, little thought is put into choice of programming language to complete their computational projects (CATAM).

MATLAB is almost universally chosen, which we suggest is more due to the Faculty's support of it than any of its inherent features.

We look to explore an alternative language, namely **Julia**, and:

- Demonstrate its usage through case studies
- Discover the advantages and disadvantages it has over other languages, particularly MATLAB

Lagrange points in Julia - an example



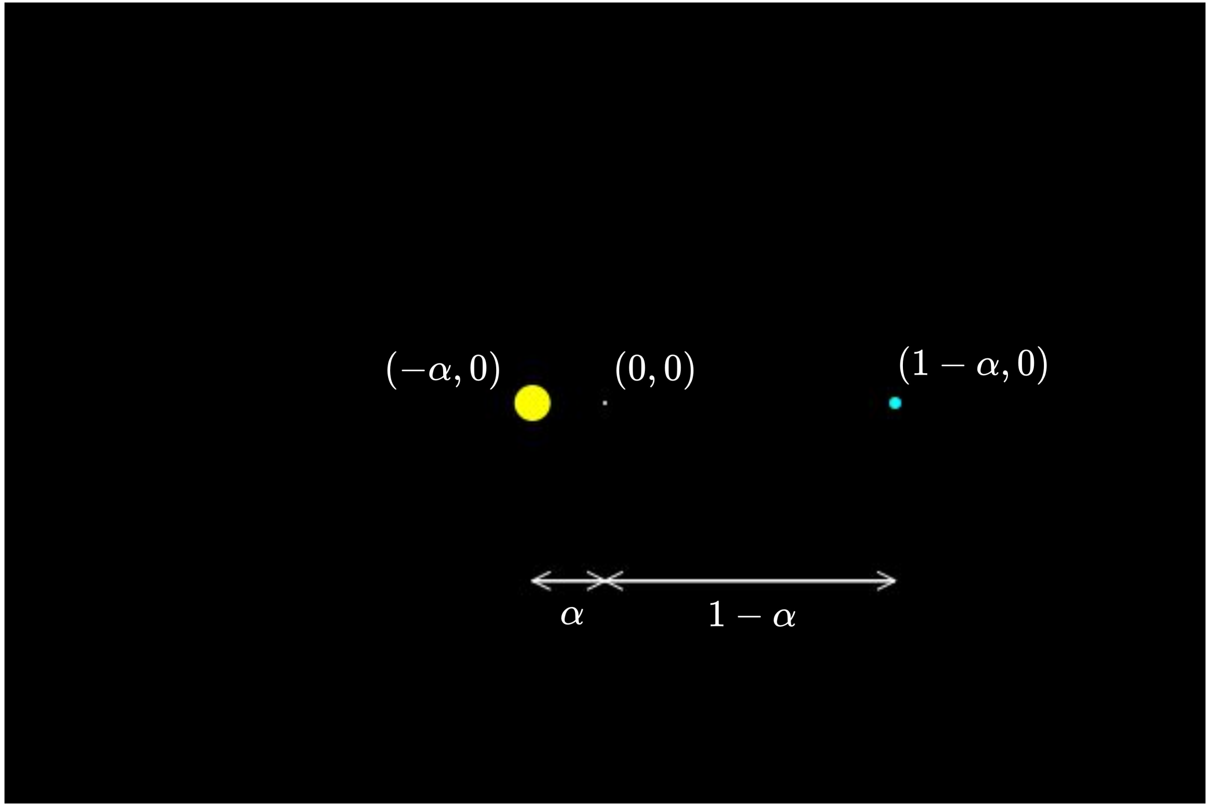
Consider a system with two large bodies:

- A sun of mass M_1
- A planet of mass $M_2 \leqslant M_1$

They orbit at fixed distance R $(= 1$ wlog) from each other

They orbit their centre of mass at rate Ω

Lagrange points in Julia - an example



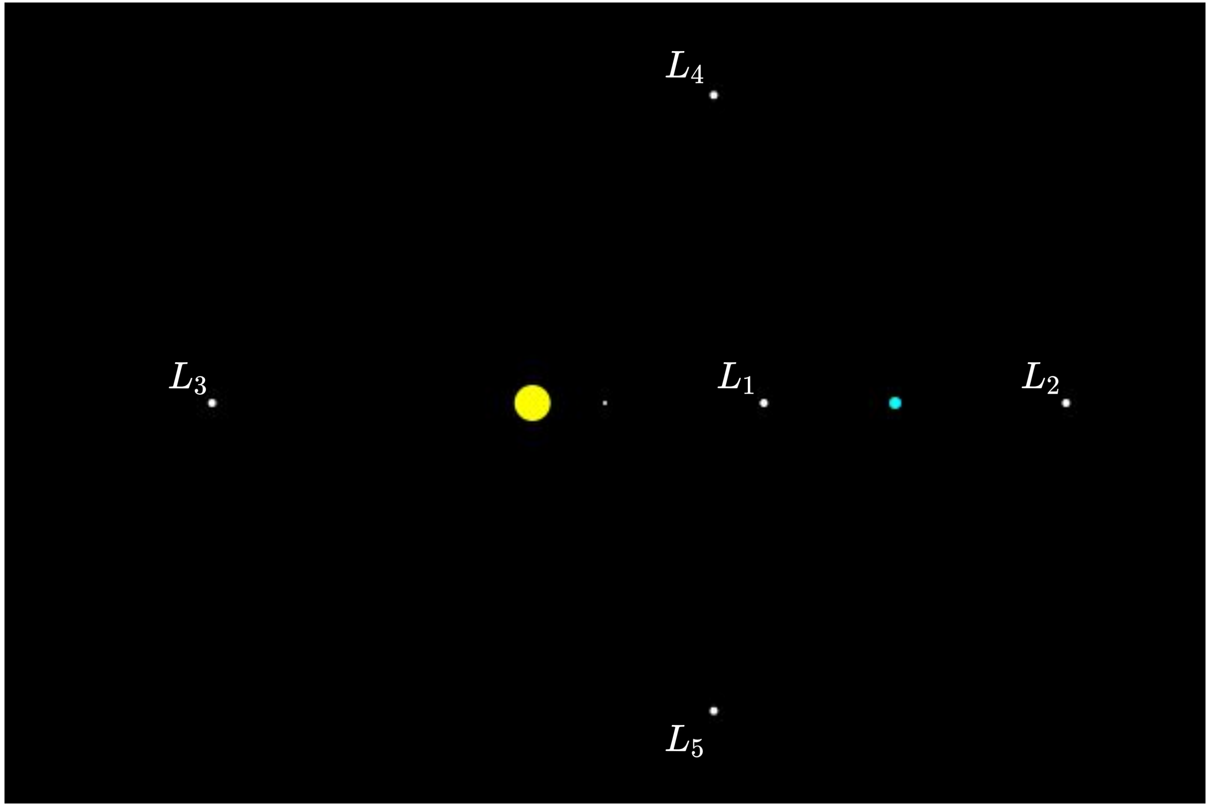
Now consider a rotating frame of reference such that:

- M_1 is fixed at $(-\alpha, 0)$
- M_2 is fixed at $(1-\alpha, 0)$

where $\alpha = \frac{M_2}{M_1+M_2}$

Note that the centre of mass is hence fixed at origin, with the frame rotating around it with angular velocity Ω

Lagrange points in Julia - an example



Let a third (much) smaller body have mass m

The Lagrange points are the five points in the rotating frame of reference where m can stay fixed

Lagrange points in Julia - an example

The acceleration of the mass m at (x, y) are (including fictitious forces due to rotating frame of reference, with the gravitational constant $G = 1$ in an appropriate system of units):

$$\left[\Omega^2 x - \frac{\Omega^2(1-\alpha)(x+\alpha)}{\sqrt{(x+\alpha)^2+y^2}^3} - \frac{\Omega^2\alpha(x+\alpha-1)}{\sqrt{(x+\alpha-1)^2+y^2}^3} + 2\Omega\dot{y}\right]\hat{\mathbf{x}}$$
$$+\left[\Omega^2 y - \frac{\Omega^2(1-\alpha)y}{\sqrt{(x+\alpha)^2+y^2}^3} - \frac{\Omega^2\alpha y}{\sqrt{(x+\alpha-1)^2+y^2}^3} - 2\Omega\dot{x}\right]\hat{\mathbf{y}}$$

Lagrange points occur where there is no net force on a body at rest, i.e. at (x, y) satisfying:

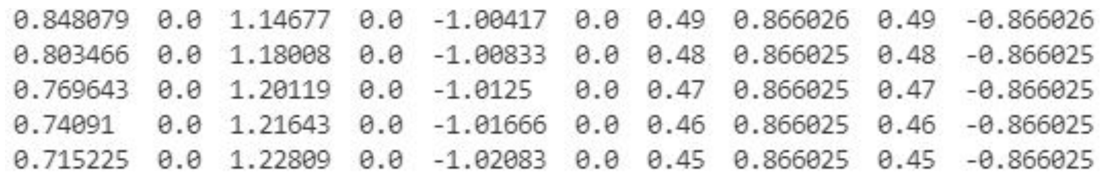
$$x - \frac{(1-\alpha)(x+\alpha)}{\sqrt{(x+\alpha)^2+y^2}^3} - \frac{\alpha(x+\alpha-1)}{\sqrt{(x+\alpha-1)^2+y^2}^3} = 0, \qquad y - \frac{(1-\alpha)y}{\sqrt{(x+\alpha)^2+y^2}^3} - \frac{\alpha y}{\sqrt{(x+\alpha-1)^2+y^2}^3} = 0$$

Lagrange points in Julia - an example

To solve these equations in Julia, I use the package **NLSolve**.

I look for solutions matching each Lagrange point while varying α, the only parameter left in the equations

For example, for $\alpha = 0.01,\ 0.02,\ 0.03,\ 0.04,\ 0.05$, the results are



```
0.848079  0.0  1.14677  0.0  -1.00417  0.0  0.49  0.866026  0.49  -0.866026
0.803466  0.0  1.18008  0.0  -1.00833  0.0  0.48  0.866025  0.48  -0.866025
0.769643  0.0  1.20119  0.0  -1.0125   0.0  0.47  0.866025  0.47  -0.866025
0.74091   0.0  1.21643  0.0  -1.01666  0.0  0.46  0.866025  0.46  -0.866025
0.715225  0.0  1.22809  0.0  -1.02083  0.0  0.45  0.866025  0.45  -0.866025
```

These match up well with theoretical values

Lagrange points in Julia - an example

I now investigate the stability of Lagrange points

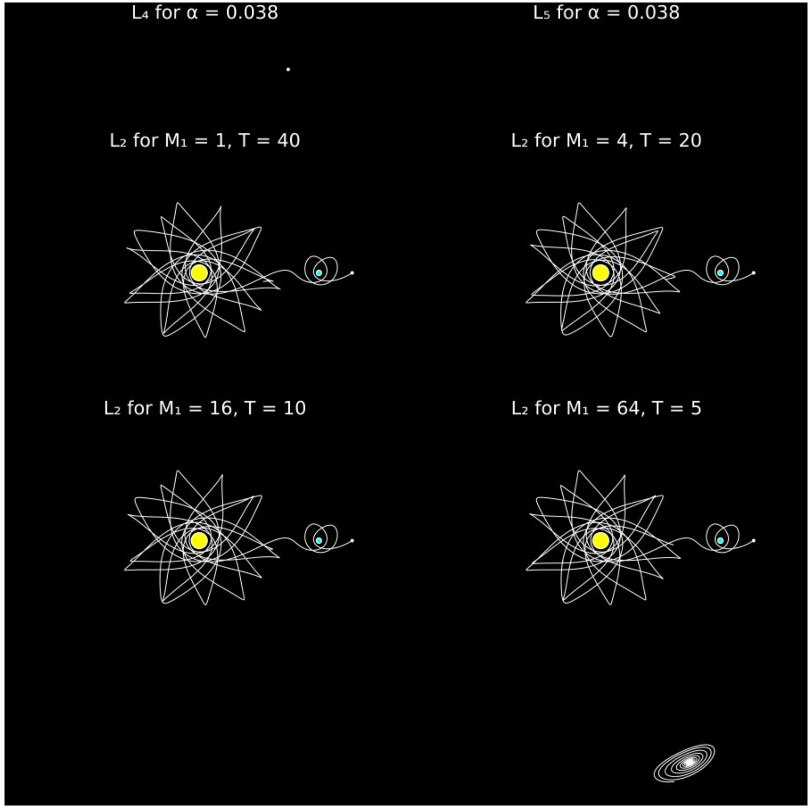
Instead of doing this analytically, I can use Julia to numerically simulate the motion of the mass m starting at rest in the rotating frame of reference near each Lagrange point

The equation for acceleration is as before, giving a differential equation for (x, y), where

- Equating gravitational force with centripetal force gives $\Omega^2 R^3 = G(M_1 + M_2)$
- Since $G = 1$ and $R = 1$ in the system of units, $\Omega = \sqrt{M_1 + M_2}$

This allows the system to be expressed entirely in terms of two parameters, M_1 and α

Lagrange points in Julia - an example



The equations can be numerically solved in Julia by the **DifferentialEquations** package

Varying α first:

- $L_1,\ L_2,\ L_3$ are unstable for all values of α
- L_4, L_5 are stable for (approx) $\alpha \leqslant 0.038$

Now, varying M_1:

- Stability is unchanged for all five points
- The time taken for the orbit to be traced is roughly proportional to $\frac{1}{\sqrt{M_1}}$

Card shuffles in Julia - an example

Case studies have been developed and written up as **Pluto Notebooks**, an interactive, dynamic programming and typesetting interface for Julia.

Advantages:

- Quick prototyping without manual compilation (ideal for beginners)
- Beautiful aesthetic with LaTeX and markdown compatibility

Disadvantages:

- Rather slow to load and execute (large overhead)
- A few minor bugs due to ongoing development

Advantages/Disadvantages of Julia

- + Compact and easily readable syntax that is intuitive to mathematicians
- + Unicode support
- - MATLAB is still more approachable for complete beginners
- + Package system, with many powerful packages written entirely in Julia
- + Very active community
- - Active development of packages may make code unstable
- + Efficiency (especially in comparison to MATLAB)
- - Compilation can be slow (although only has to be done once)
- + Strong type system (as opposed to Python)
- - Strong type system might be hard to get used to when coming from Python
- + Powerful macro functionality

Development of case studies for using the Julia language in mathematics

Full project including code and Pluto notebooks at
https://sje30.github.io/catam-julia/

Also available from the list of Julia tutorials at
https://julialang.org/learning/tutorials/