Simulating Tidal Interactions between Galaxies: A Pre-University Student Project

M. Brea-Carreras, M. Thiel & M. Pössel*

Haus der Astronomie, MPIA-Campus, Königstuhl 17, 69117 Heidelberg, Germany

Abstract

We report on a project undertaken this Summer by pre-University student interns at Haus der Astronomie: point-particle simulations of galaxy collisions with the aim of reproducing observational data from such collisions. We succeeded in providing a visually similar representation of both NGC 5426/7 and the "Antennae" galaxies (NGC 4038/9), and were able to make deductions about the relative positions and orbits of these galaxies. The project is an example for how participants with little more than high-school level previous knowledge can successfully tackle, and understand, advanced topics from current astrophysical research. This report was written by the two participants (M. B.-C. and M. T.), on whose experiences it is based, in collaboration with their supervisor at Haus der Astronomie (M. P.).

1 Introduction

There is a broad consensus among science educators about the importance for science education of hands-on activities, elements of inquiry-based learning, and understanding science as a process [9, 2, 3]. For high-school students, or students in limbo between high school and university, internships at research institutions, are a suitable way of experiencing first hand the process of science, whether as part of a larger project or in the course of working on a dedicated student research project.

Haus der Astronomie has offered research internships since 2009, and is currently offering a yearly "International Astronomy Summer Internship Program" for advanced high school students, or to students transitioning from high school to college/university.[1] Among the authors of this e-print, two of us attended the 2017 internship program (M. B.-C. and M. T.), while one of us was the internship supervisor (M. P.).

*Corresponding author: poessel@hda-hd.de

[1] Up-to-date information can be found on http://www.haus-der-astronomie.de/en/what-we-do/internships/summer-internship

The project described here used computer simulations to reproduce classic results about galaxy evolution and the observational properties of interacting galaxies. Pedagogically, numerical treatments have several advantages [4, 1, 12]: They allow students to explore dynamical situations that are too advanced for the analytical tools at their disposal; in fact, mathematically, the approach is simpler than many analytical methods. Numerical approaches also allow for a descriptions that are beyond reach for even the most advanced analytical methods. The associated challenge is, of course, that in order to write such simulations themselves (as opposed to using existing software), students need appropriate programming skills.

During the first three weeks of the 2017 internship, the project was worked on by M. T. and another intern; during the second three weeks, M. T. and M. B.-C. continued the project in their spare time, on their own initiative. This unusual, and remarkably successful, extra effort prompted M. P. to have the two interns present their results at the WE Heraeus Summer School "Astronomy From Four Perspectives: The Dark Universe" in Heidelberg.

This e-print serves a double function: For one, it describes the student research project undertaken by M. B.-C. and M. T., to whom the collective "we" in the following sections 2 to 4.4 refers; this aspect, we hope, will make the text of interest to those involved in astronomy education, who might be interested in including this or similar student activities in their own programs. In addition, the e-print is part of the project itself. After all, writing up your methods and results is an integral part of scientific work — even if, in this case, we have included additional aspects related to astronomy education which are not found in ordinary astronomy papers. Each intern presents a personal view of their experiences in the appendix, sections **??** and **??**.

The scripts, simulations parameters and presentations written for this project are publicly available and can be downloaded from GitHub[2].

2 Project summary

Theories of galaxy evolution and formation need to be able to explain, among other things, the appearance of interacting galaxies. The most widely accepted explanation was first studied more intensively in the 1960s [11], and identifies tidal interactions between galaxies as the cause of the deformations of the galaxies involved. The main reference for our student research project is a classic 1972 paper by Alar and Juri Toomre [13], which describes basic numerical simulations of galaxy interactions. The authors argue that the bridges and tails visible in some spiral galaxies are remnants of tidal interactions between galaxies involved in a close encounter — and they support their argument with restricted three-body calculations of selected encounters, including reconstructions of well-known pairs the "antennae" NGC4039/39, the "mice" NGC 4676, and M51 with its small companion NGC 5195.

For our research project, we followed in the footsteps of Toomre and Toomre by creating our own galaxy simulation, written using the Python programming language.[3] We describe our simulation techniques in section 3, while section 5 presents our results. In section 5, we draw on our per-

[2]The repository can be cloned from https://github.com/manuelbrea99/simulating-tidal-interactions.git

[3]Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org

sonal experience to give recommendations for similar educational projects.

3 Methods

Our simulations were written in Python. We are aware that using a general programming language like Python as the basis of a student research project will raise the bar higher than reliance on more specialised and restricted software for students unfamiliar with programming; in our case, one of us (M. B.-C.) had previous experience in using Python, while the other (M. T.) learned Python during his internship, but had previous experience programming in Java. Going by our personal experience, we think that Python is a good choice for those who are new to programming.

The simulations were all three-dimensional, in the sense that a third dimension was supported by our code and taken into account for the calculations. However, in the first two examples both galaxies share the same orbit plane and their movement is always parallel to said plane; the z-coordinate is alway zero for every particle, making them in effect equivalent to a two-dimensional simulation.

Regarding the structure of the simulation itself, we focused our efforts on making our implementation simple, clean and understandable at its core. The basic elements of our physical model are point particles with mass on the one hand and point particles without mass, that is: test particles who are acted upon by gravitational forces but are not themselves treated as sources of gravity. Following Toomre and Toomre, the mass of each galaxy is represented by a single point particle with the total galaxy mass located in the center of the galaxy. Galaxy structure is simulated using test particles representing stars in each galaxy's (initially undisturbed) disk.

That means that in a two-galaxy simulation we have just two sources of gravity: the two central point particles carrying each galaxy's mass. For all other particles, we only need to calculate how they react to the gravitational forces exerted by the two central point particles.

These core elements require only their three-dimensional position and velocity information to be stored — and, in the case of the central particle of each galaxy, their scalar mass. This data, along with a predefined time-step, is passed to an update function, which returns the particles' new positions and velocities. Results are plotted using the Python 2D plotting library matplotlib [6], which we found simple and straightforward to use for our purposes.

3.1 Update Function

In order to define a physics update function for a simulation we need to perform a second order integration: to solve for the trajectory of moving bodies affected by a location-dependent acceleration. Whereas the two-body problem for two point particles orbiting each other under the influence of their mutual gravitational attraction can be solved analytically, the general situation of three or more bodies can only be described numerically.

In our particular set-up, we have N particles, each designated by an index j. We denote the jth particle's position at time t as $\vec{x}_j(t)$, its velocity at time t by $\dot{x}_j(t)$ and its acceleration by $\ddot{x}_j(t)$. The acceleration felt by the jth particle at time t results from the sum of all the other particles exerting a gravitational force; each separate gravitational influence is described by Newton's inverse-square law of gravity. The total ac-

celeration of the jth particle is

$$\ddot{\vec{x}}_j(t) = -G \sum_{i \neq j} m_i \cdot \frac{\vec{x}_j(t) - \vec{x}_i(t)}{|\vec{x}_j(t) - \vec{x}_i(t)|^3}, \quad (1)$$

where m_i is the mass of the ith particle and G denotes the gravitational constant. As our test particles are massless, they can be skipped in the summation. Only the "galaxy centers of mass" need to be taken into account when computing the acceleration of any particle j.

For the purpose of numerical integration, time is divided into discrete time steps. The update function is used to calculate the next time step, given the information available at that time step or earlier time steps. We define the elapsed time at time-step n as

$$t_n = t_0 + nh \quad (2)$$

where h denotes the (in our case: universal) time interval between one time step and the next. This discretisation is a simplification, and it stands to reason that the value defined for h will have direct consequences for the accuracy of any numerical calculation.

3.1.1 Euler method

The most straightforward way of numerical integration is the Euler method, in which all rates of change are approximated as linear, and the change in a function is taken to be the rate of change times the duration of the time step. Following this prescription, the Euler integration algorithm computes the updated velocity and position for each body j and for each iteration n as

$$\dot{\vec{x}}_j(t_{n+1}) = \dot{\vec{x}}_j(t_n) + h\ddot{\vec{x}}_j(t_n) \quad (3)$$

$$\vec{x}_j(t_{n+1}) = \vec{x}_j(t_n) + h\dot{\vec{x}}_j(t_n), \quad (4)$$

where n is the number of iterations, and thus nh is the elapsed time. At each step, we need to evaluate $\ddot{\vec{x}}_n^{(j)}$ following equation (1).

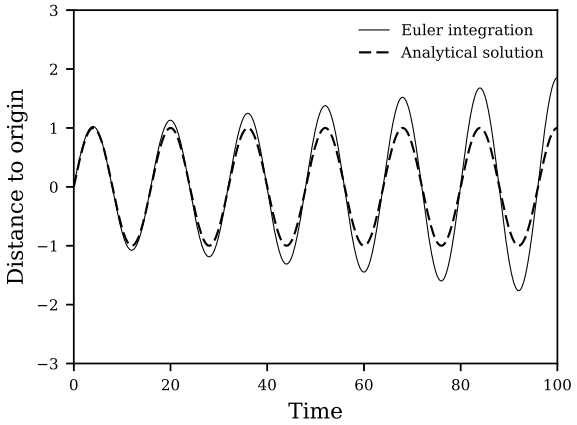

Figure 1: Displacement from origin over time using Euler integration method (step length $h = 0.08$), compared with the analytical solution for a simple harmonic oscillator with amplitude $A = 1$ and angular frequency $\omega = \frac{\pi}{8}$.

3.1.2 Numerical errors and the Velocity Verlet Algorithm

Euler integration incorporates some systematic errors, which lead to cumulative divergence between the numerical solution and the true solution. An instructive example is that of the one-dimensional harmonic oscillator, where the acceleration is given by

$$\ddot{x} = -\omega^2 x \quad (5)$$

for some constant angular frequency ω. In this simple case, we can directly write down an analytical solution, namely

$$x(t) = A \cdot \sin(\omega t), \quad (6)$$

with A the amplitude of the oscillation. This allows for a direct comparison between the analytical and numerical solution which exposes some of the problems inherent in numerical simulations. Imagine that the moving mass of that oscillator is moving towards positive x, so that we are in an phase where $\dot{x} > 0$. The Euler prescription (3)

assumes that the acceleration acting on our mass during the time step t_n is equal to the acceleration at the *beginning* of that time step. But in reality, we know that the effect of the acceleration will be greater than that. After all, during that time step, the mass is moving towards greater x, and the acceleration pulling it back will increase at greater x values. Evidently, the Euler method systematically underestimates the pull experienced by our mass. Conversely, when our mass is moving back towards the origin, the Euler method will systematically overestimate the pull. Both effects erroneously increase the total energy, and thus the amplitude, of the oscillation: the first since it allows the mass to move further outward than allowed, and the second because it imparts a larger momentum on the mass. The cumulative effect can be seen in figure 1, where the numerical solution progressively diverges from the analytical solution.

Such integration errors have led to the development of alternative numerical integration schemes. One such scheme is the Velocity Verlet algorithm [14]. This method introduces additional "half-steps" for time, which we will designate as $t_{n+1/2} = (n + 1/2)h$. First, the velocity is calculated at one such half-step as

$$\dot{\vec{x}}_j(t_{n+1/2}) = \dot{\vec{x}}_j(t_n) + \frac{h}{2}\ddot{\vec{x}}_j(t_n) \qquad (7)$$

In our case, the acceleration is given by eq. (1). Once computed, this half-step velocity is used to find the position at t_{n+1} as

$$\vec{x}_j(t_{n+1}) = \vec{x}_j(t_n) + h\dot{\vec{x}}_j(t_{n+\frac{1}{2}}) \qquad (8)$$

Next, using these newly-determined position values, the acceleration *evaluated at* t_{n+1} is computed, and used to compute $\dot{\vec{x}}^{(j)}_{n+1}$, as

$$\dot{\vec{x}}_j(t_{n+1}) = \dot{\vec{x}}_j(t_{n+\frac{1}{2}}) + \frac{h}{2}\ddot{\vec{x}}_j(t_{n+1}). \qquad (9)$$

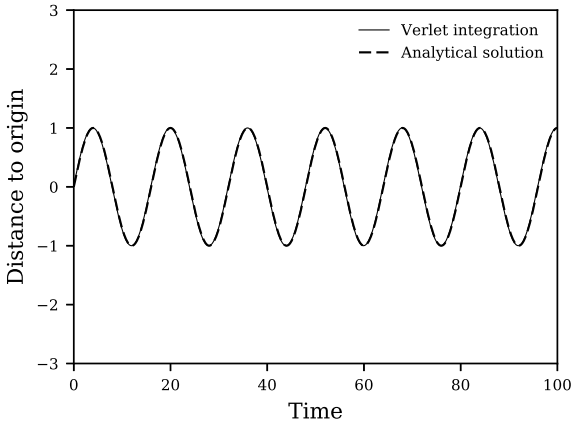

Figure 2: Displacement to origin over time using Verlet integration method ($h = 0.08$) against the analytical solution of a simple harmonic oscillator $x(t) = A \cdot \sin(\omega t)$, with $A = 1$ and $\omega = \frac{\pi}{8}$).

For the simple example of the harmonic oscillator, the comparison between figures 1 and 2 shows the remarkable increase in numerical stability of this improved algorithm — at least within the range of the simulation shown in those figures, the Verlet version does not visibly overshoot the mark! The increase in computational cost is a mere 50%, as the half-steps are used only in the velocity calculation, not in the position calculation.

3.2 Initial values

The algorithms described in the previous two sections allow us to simulate the *evolution* of physical systems. In order to pick out a particular solution, we need to specify appropriate initial values, as is generally the case when solving differential equations.

The aim of our project is to simulate interactions between galaxies; our initial conditions amount to modelling, in an appropriate way, the original two galaxies involved in the interaction. We make

the assumption that, before the interaction proper, we can regard the galaxies as completely separate systems. For each of the galaxies, the test particles are assumed to be in circular orbit around the central galaxy particle.

3.3 The two-body problem

Since we have only the two central particles acting as sources of gravity, it is possible to calculate their motion analytically — the two-body problem, after all, allows for direct analytical solutions. We have made use of this analytical description in choosing the initial conditions for the two central particles, computing the orbital eccentricity e and eventual pericenter distance R_{min} in each case. In particular, we have made use of the following relations [8, chapter 9].

First of all, due to the conservation of angular momentum, the two-body motion takes place only in the two-dimensional plane defined by the two initial velocity vectors of the central point particles.

For two centers of mass m_1 and m_2 with positions $\vec{x}_1$ and $\vec{x}_2$, their relative position $\vec{r}$ and reduced mass μ are defined as

$$\vec{r} = \vec{x}_2 - \vec{x}_1 \tag{10}$$

and

$$\mu = \frac{m_1 m_2}{m_1 + m_2}, \tag{11}$$

respectively. Solving the two-body problem then amounts to solving two one-body problems: free, linear motion for the center of mass of the system, plus an effective one-body problem for the relative motion vector that results in a Kepler ellipse. Using polar coordinates with one of the focus points at the origin, the shape of that ellipse is given by

$$r = \frac{r_0}{1 - e\cos\theta}. \tag{12}$$

Here, $\theta = 0$ corresponds to the direction from the focus to the center of the ellipse; e is the ellipse's eccentricity. The quantity r_0 is called the semilatus rectum and sets the length scale of the ellipse.

Each orbit is characterized by two conserved quantities: the mechanical energy

$$E = \frac{1}{2}\mu\dot{r}^2 - \frac{G m_1 m_2}{r} \tag{13}$$

and the angular momentum

$$L = \dot{r}_{tan} r \mu, \tag{14}$$

where $\dot{r}$ is the relative speed of the two centers of mass and $\dot{r}_{tan}$ the component of the relative velocity that is tangential to the orbit.

Using these two quantities, the two parameters in the ellipse equation (12) can be written as

$$r_0 = \frac{L^2}{\mu G m_1 m_2} \tag{15}$$

$$e = \sqrt{1 + \frac{2EL^2}{\mu (G m_1 m_2)^2}}. \tag{16}$$

Computing the minimal distance between the two particles R_{min} is then just a matter of solving an optimization problem, namely setting

$$\frac{\mathrm{d}r}{\mathrm{d}\theta} = \frac{-r_0}{1 - e\cos\theta}(e\sin\theta) \stackrel{!}{=} 0. \tag{17}$$

The result is

$$\sin\theta = 0, \tag{18}$$

so we must either have $\theta = 0$ or $\theta = \pi$ for $0 \leq \theta < 2\pi$. Inserting these values into (12), we find that

$$R_{min} = \frac{r_0}{1 + e}. \tag{19}$$

In our attempts to model certain specific interactive pairs of galaxies, we made use of this relation in choosing (albeit with a certain amount of trial and error) suitable initial values.

3.4 Optimization

Even with computers inordinately more powerful than in the early 1970s where Toomre and Toomre created their simulations, run time values and run time differences in our Python scripts were notable features of our project — and just like in professional simulation projects, we experimented with ways of optimizing the performance of our programs.

One area where different implementations turned out to make a considerable difference involved different ways of storing our simulation data. The simplest way of handling simulation data, which we would recommend for students with little to no programming experience, involves the use of Python list objects. The simulation itself involves iterative loops that wrap the update function.

This implementation turns out to be comparatively slow. In those of our simulations that involved a considerable number of test particles, run time did set the pace both for testing our programs and for finding the best initial values by trial and error.

As an alternative, we took advantage of the array structures and associated linear algebra functionality of the NumPy core package within the SciPy ecosystem for Python [7, 10]. User-defined functions such as those necessary to implement our update function can be "vectorized" to allow their direct application to array objects, eliminating the need for explicit loops. While the linear algebra aspects of this approach is likely to put it out of reach of implementations of these simulations as high school projects, we did find this implementation to be substantially faster, as shown in table 1. In accordance with good computing practices, we also experimented with an object-oriented approach to programming our simulation, as a way of keeping the code modular and easier to organise and debug. In the end, though, we found it difficult to reconcile a fully object-oriented approach with the vectorisation of our variables and the update algorithm.

m	Non-vectorized	Vectorized
28	11.291s	1.428s
134	45.404s	1.511s
202	67.753s	1.597s
268	85.136s	1.621s
406	126.231s	1.763s
540	166.262s	1.915s
676	206.818s	1.993s
810	251.798s	2.123s

Table 1: Execution times for increasing particle count m for a constant number of 5000 time steps

Last but not least, creating the diagrams used to display the results of our simulations proved a significant contribution to overall run time. We made use of the Matplotlib library [6] for this purpose, which uses the CPU for displaying graphics and thus has a substantial impact on performance. This impact can be reduced when rendering only each nth frame, e.g. only every 10^8 or 10^7 years in simulation time. An alternative possibility for optimisation, which we did not attempt but which might be of interest to others, would be to switch to VPython (http://vpython.org/) or PyOpenGL (http://pyopengl.sourceforge.net/), both of which are GPU-based.

4 Simulations

The simulations we created for our project were of two types: in our first step, we set out to replicate two of what Toomre & Toomre call their "elementary examples"

[13, section II], namely the retrograde passage and the passage of a small companion. In both cases, there is a main galaxy modelled as a central particle surrounded by a flat disk of orbiting test particles, while the second galaxy is modelled only as a single central particle. In these encounters, the orbital plane of the two central particles coincides with the disk plane of the main galaxy, so the situation is confined to a two-dimensional plane.

The main part of our project, however, was dedicated to simulating the interactions in the NGC 5426/7 system and for the "Antennae" (NGC 4038/9) pair. In these simulations, each of the interacting galaxies has a disk of orbiting stars, modelled as a disk of test particles, but the simulation is no longer contained in a two-dimensional plane.

4.1 Retrograde passage

Figure 3 shows the passage of an equal-mass companion. The orbit of the perturbing body is retrograde with respect to the revolution of the particles.

As seen at $t = 3 \cdot 10^8$ years and onwards, only the outermost rings show obvious perturbations in their orbit. This results in the formation of a galactic tail at $t = 5 \cdot 10^8$ years, which then dissipates soon after. The result is overall similar to that of Toomre & Toomre [13]. A small difference, namely the extent of the outermost regions of the disk at $t = 5 \cdot 10^9$ years, are readily explained by differences in the initial values.

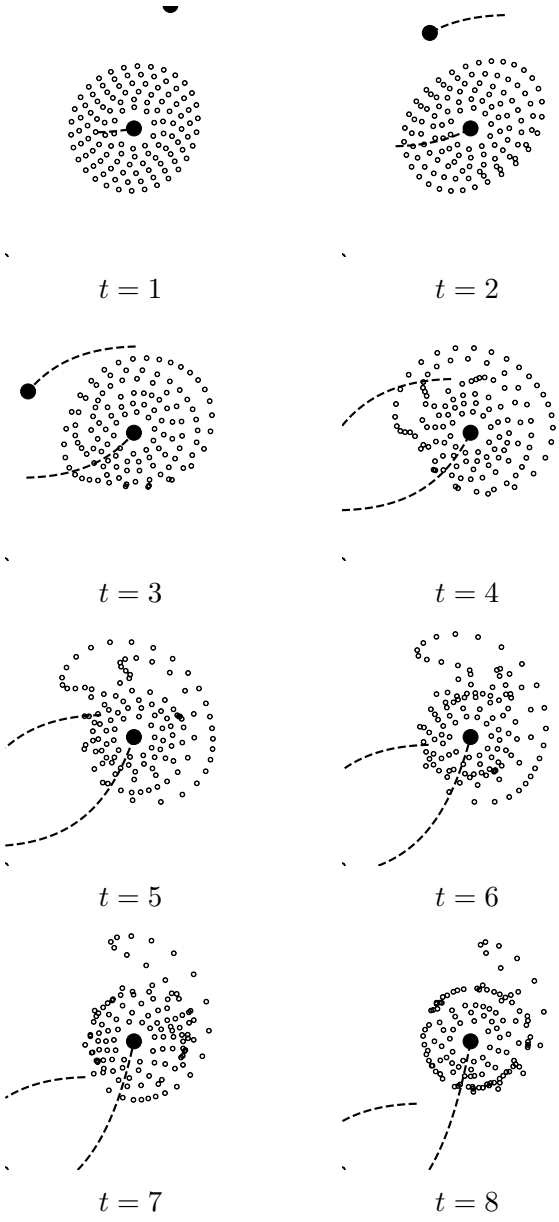

Figure 3: Flat retrograde passage of a companion with equal mass. Both the main galaxy and the companion have a mass of main galaxy mass $10^{11}\,M_\odot$*. All time values are given in multiples d of* 10^8 *years. In this example,* $e \approx 1.21$ *and* $R_{min} \approx 30.46\,\mathrm{kpc}$*.*

4.2 Passage of a smaller companion

In this example, a quarter-mass companion passes by our main galaxy. This is a *direct* passage, unlike the last example, which is retrograde. The resulting encounter is shown in figure 4. The formation of a bridge between the main galaxy and the companion is very visible in this example, but the bridge does not last very long; it dissipates

completely by about $t = 12 \cdot 10^8$ years, much sooner than its tidal counter-arm.

4.3 Three-dimensional model of NGC 5426

The galaxy pair consisting of NGC 5426 and NGC 5427, cf. figure 5, is a remarkable example of spiral arms and a bridge forming due to tidal interactions. As part of our project, we attempted to recreate these structures.

In the encounter we simulated, the revolution of the particles were retrograde to the orbit of the interacting galaxies with each other. The results of our simulation can be seen in figure 6. From the chosen perspective, NGC 5426 has initially started on the left side of the view. By the time $t = 5 \cdot 10^8$ years, however, the first close encounter has already occurred, and NGC 5426 can be seen to the right of its companion. In the snapshot at $t = 7\ 10^8$ years, one can see the bridge connecting the two galaxies.

In the snapshots at $t = 7 \cdot 10^8$ years and $t = 12 \cdot 10^8$ years, both galaxies are moving away from each other, and are approaching their apocenters by $t = 22 \cdot 10^8$ years.

At $t = 22$, the simulated galaxies share visual similarities with their observed counterparts: global spiral arms similar to those in the observed galaxies are present.

4.4 Antennae galaxies NGC 4038/9

The last simulation presented in this article was an attempt to replicate the encounter that gave rise to the antennae-like shape of NGC 4038/9. A view of the antennae galaxies, taken with the $20''$ telescope at Kitt Peak Observatory, can be seen in figure 7. For comparison, the results of the final version of our simulation are shown in figure 8.

Using an eccentricity value similar to Toomre & Toomre's ($e = 0.44$ instead of $e = 0.5$), the resulting shape closely resembles the structure of the observed pair. It is interesting to note that, similarly to the previous example, both centers of mass are relatively close to their respective apocenters at $t = 9$. This is better represented in the Figure 8's second depiction of $t = 9$, where the projection plane is no longer perpendicular to the orbit plane.

5 Lessons learned, and recommendations

Personalised accounts of the experiences of M. B.–C. and M. T. can be found in appendices **??** and **??**, respectively. This section is our attempt to extract more general lessons we learned from the project, which are relevant for those intending to introduce a project of this kind in a high-school setting.

In terms of prerequisites, we found that in our case, the most important thing that is required is a solid interest in the project. Students replicating our project as a whole — writing all their own code — will undoubtedly face obstacles while working and spend numerous hours troubleshooting their code. They will feel frustrated at more than one point, and it is that interest and commitment what will push them to reach their final results. We have found that with this level of commitment, plus the required concepts from high school physics and maths, reaching the proposed goals is feasible; some basic linear algebra was a big plus for the optimization strategy discussed in section 3.4.

In the form presented here, this project

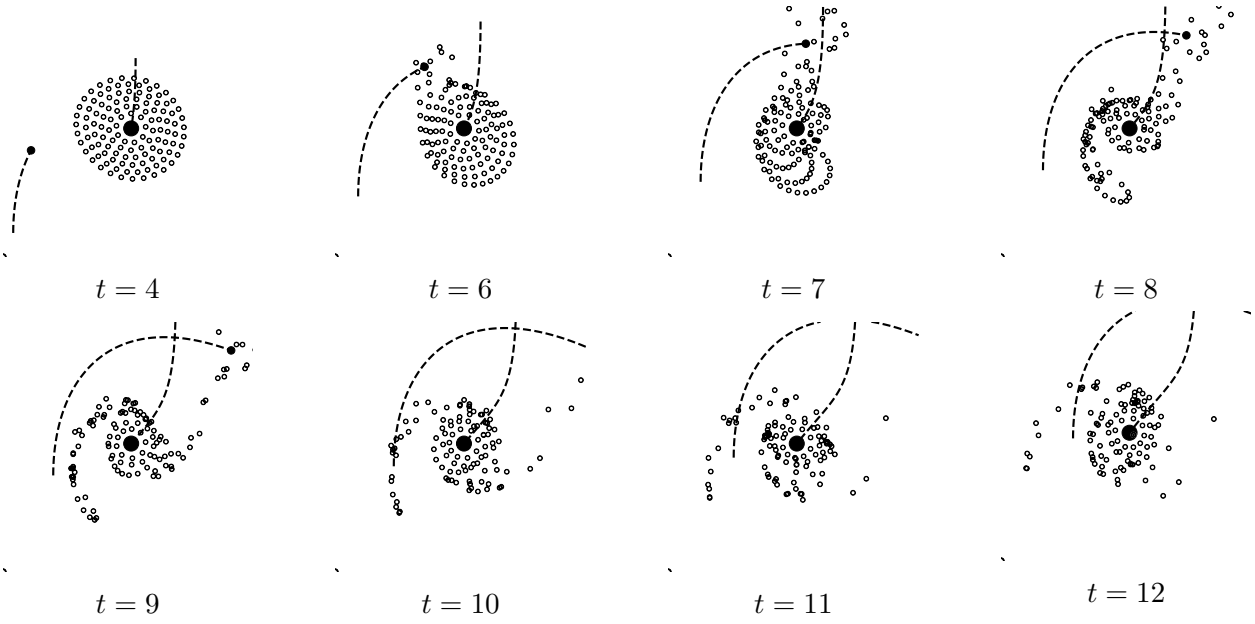

Figure 4: Flat direct passage of a quarter-mass companion: main galaxy mass $10^{11}\ M_\odot$*, companion galaxy mass* $2.5 \cdot 10^{10}\ M_\odot$*. All time values are in multiples of* 10^8 *years. The orbital parameters are* $e \approx 1.04$ *and* $R_{min} \approx 27.02\,\mathrm{kpc}$*.*

requires a considerable amount of time; as such, it could be suitable as a programming project linking the subjects of physics and computation (such as the newly introduced subject IMP linking computer science, mathematics, and physics, in the German state of Baden-Württemberg). Given that the project provides opportunities for introducing most of the key programming concepts, such as list or array structures, simple calculation and evaluation of expressions, conditions, and loop structures, it could be paired with a general introduction to these concepts in the framework of an introductory programming class.

The students could work their way up to the full numerical integration using simpler examples such as the harmonic oscillator, similar to our on exploration of numerical stability as described in section 3.1.2. Starting with two-dimensional situations provides a further simplification step.

Conceptually, the Euler method should be easiest, and provides a useful first numerical integration scheme. For advanced students, who have a notion of the concept of a derivative as an infinitesimal approximation of a function's graph, the connection between that approximation and Euler integration can be made explicit.

Once students have programmed a working two-dimensional orbit simulator on their own, they are prepared to use a paper such as the one of Toomre & Toomre [13] and should be able to tackle replications of those authors' "Four Elementary Examples", all of which are two-dimensional.

A reduced version of our project might well end here. By this point, students should have achieved a basic understanding of numerical integration and orbital mechanics in a two-dimensional space. An optional, but useful further step would be the introduction of the velocity Verlet algorithm, together with an elementary discussion of numerical stability similar to the

Figure 5: The galaxy pair NGC 5426 and 5427. Image credit: Gemini Observatory/Association of Universities for Research in Astronomy

Galaxy designation
NGC 4038/4039 ("Antennae")
NGC 4676 ("The mice")
NGC 2207 and IC 2163
AM 0500-620
Arp 271
NGC 6786
UGC 9618
UGC 8335
Arp 256
ESO 593-8
ESO 77-14

Table 2: List of notable interacting galaxies

one in our section 3.1.2.

The next major step is the transition from a two-dimensional to a three-dimensional simulation. Judging by our own experience, the most difficult aspect of this step is the geometry of the description in terms of the standard orbital elements. Personally, we found Rodrigues' rotation formula [5] to be a helpful short-cut.

Once this transition is achieved, students can tackle the task of using their simulations to recreate real astronomical images of interacting galaxies, similar to what we reported on in our section . Examples for interacting galaxies suitable for this part of the project, for which images are readily available online, can be found in table 2.

Our description so far refers to a version where the student work comparatively independently. For us (M. B.-C. and M. T.), one of the most positive parts of this project was that we were often left with little to no guidance, forcing us to do our own research and develop our problem-solving skills. We suggest that a similar approach could be beneficial when recreating this project with suitably advanced and motivated students.

In addition, there is of course the possibility of reducing the degree of difficulty of the project, but at the expense of also losing the corresponding benefits of independent work. Just as scientists frequently learn new software skills by studying and by creating variations of existing script and programs, students can be given existing scripts or parts of such scripts to experiment with or to add particular parts that have deliberately been left out. The scripts written in the course of this project, available at GitHub[4], can be used for this purpose.

Acknowledgements

The work described in this paper was only made possible through this internship, and thus M. B.–C. and M. T. sincerely thank

[4]https://github.com/manuelbrea99/simulating-tidal-interactions.git

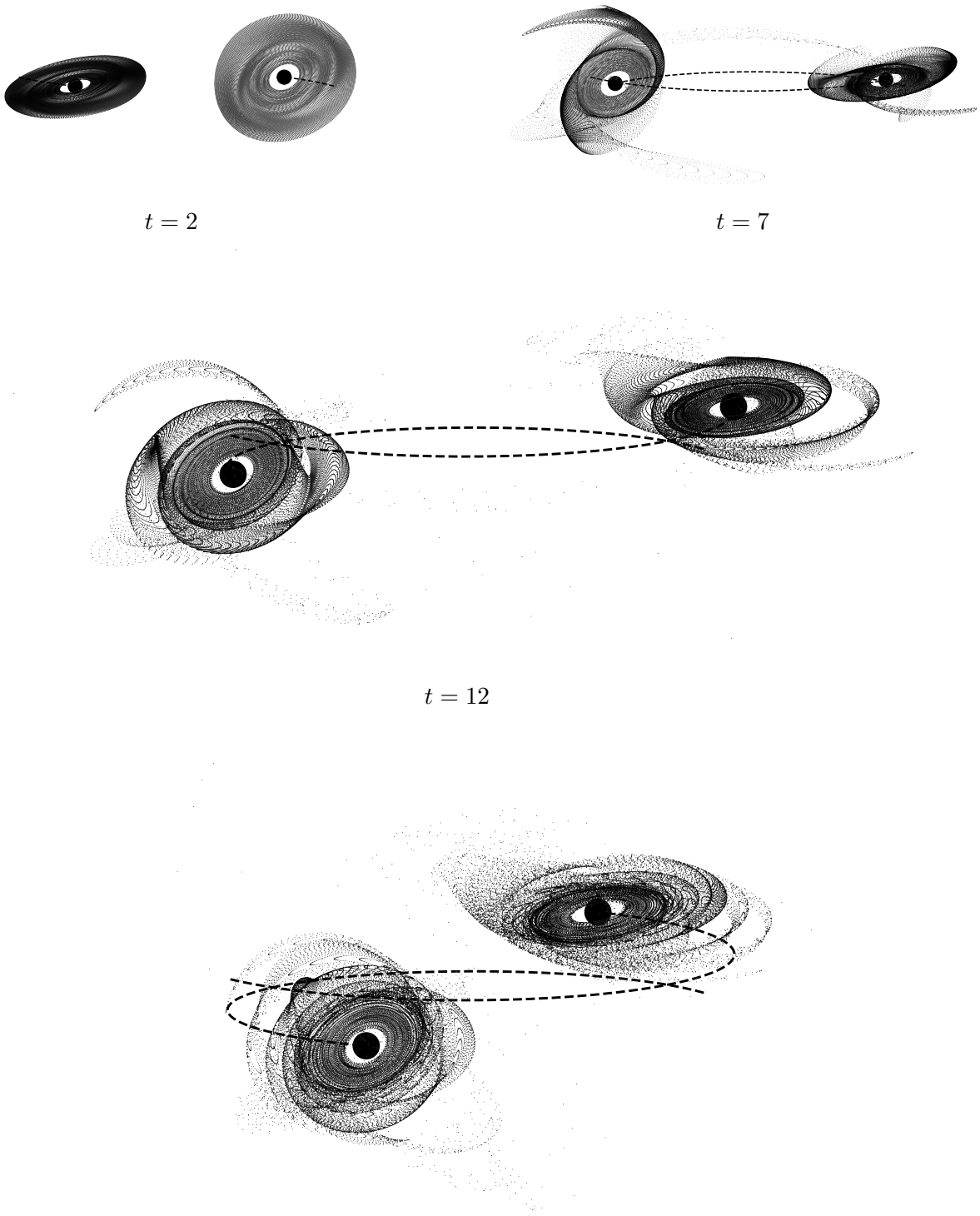

Figure 6: Simulated interaction between galaxies analogous to NGC 5426 and NGC 5427. The encounter was a retrograde passage where both galaxies had equal mass. The projection plane formed and angle of $83°$ *with the two central particles' orbital plane. The two particle disks have a relative inclination of* $30°$ *with respect to each other. In this simulation, the eccentricity of the orbit is* $e \approx 0.67$ *and the distance of closest approach is* $R_{min} \approx 28.16\,\mathrm{kpc}$.

Figure 7: Visible-light observation of the antennae galaxies NGC 4038 and NGC 4039. Image credit: Bob and Bill Twardy/Adam Block/NOAO/AURA/NSF

the Haus der Astronomie as well as the Max-Planck-Institute for Astronomy for offering this opportunity to high school students. Furthermore, they would like to express our gratitude towards both facilities for introducing them to the world of scientific research and providing insight into life as an astronomer. Lastly and most importantly, M. B.–C. and M. T. thank M. P. for supervising the internship and giving them the chance to attend the WE Heraeus Summer School and present their results.

References

[1] G. Christiansen. Elementary computer physics, a concentrated one-week course. *American Journal of Physics*, 46:748–751, July 1978.

[2] European Commission. *Science Education NOW: A renewed Pedagogy for the Future of Europe*, volume EUR22845. Office for Official Publications of the European Communities, 2007.

[3] National Research Council. *Successful K-12 STEM Education: Identifying Effective Approaches in Science, Technology, Engineering, and Mathematics*. National Academies Press, 2011.

[4] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics,*, volume 1. Addison-Wesley, 1970. Section 9.6–9.7.

[5] J. J. Gray. Olinde rodrigues' paper of 1840 on transformation groups. *Archive for History of Exact Sciences*, 21:375–385, 1980.

[6] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[7] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[8] C. Kittel. *Mechanics*. McGraw-Hill, 1965.

[9] R. Millar and J. Osborne. *Beyond 2000: Science Education for the Future*. King's College London, School of Education, 1998.

[10] Travis E. Oliphant. A guide to numpy, 2006.

[11] J. Pfleiderer and H. Siedentopf. Spiralstrukturen durch Gezeiteneffekte bei der Begegnung zweier Galaxien. *Zeitschrift für Astrophysik*, 51:201, 1961.

[12] R. W. Stanley. Numerical methods in mechanics. *American Journal of Physics*, 52:499–507, June 1984.

[13] A. Toomre and J. Toomre. Galactic bridges and tails. *Astrophysical Journal*, 178:623–666, 1972.

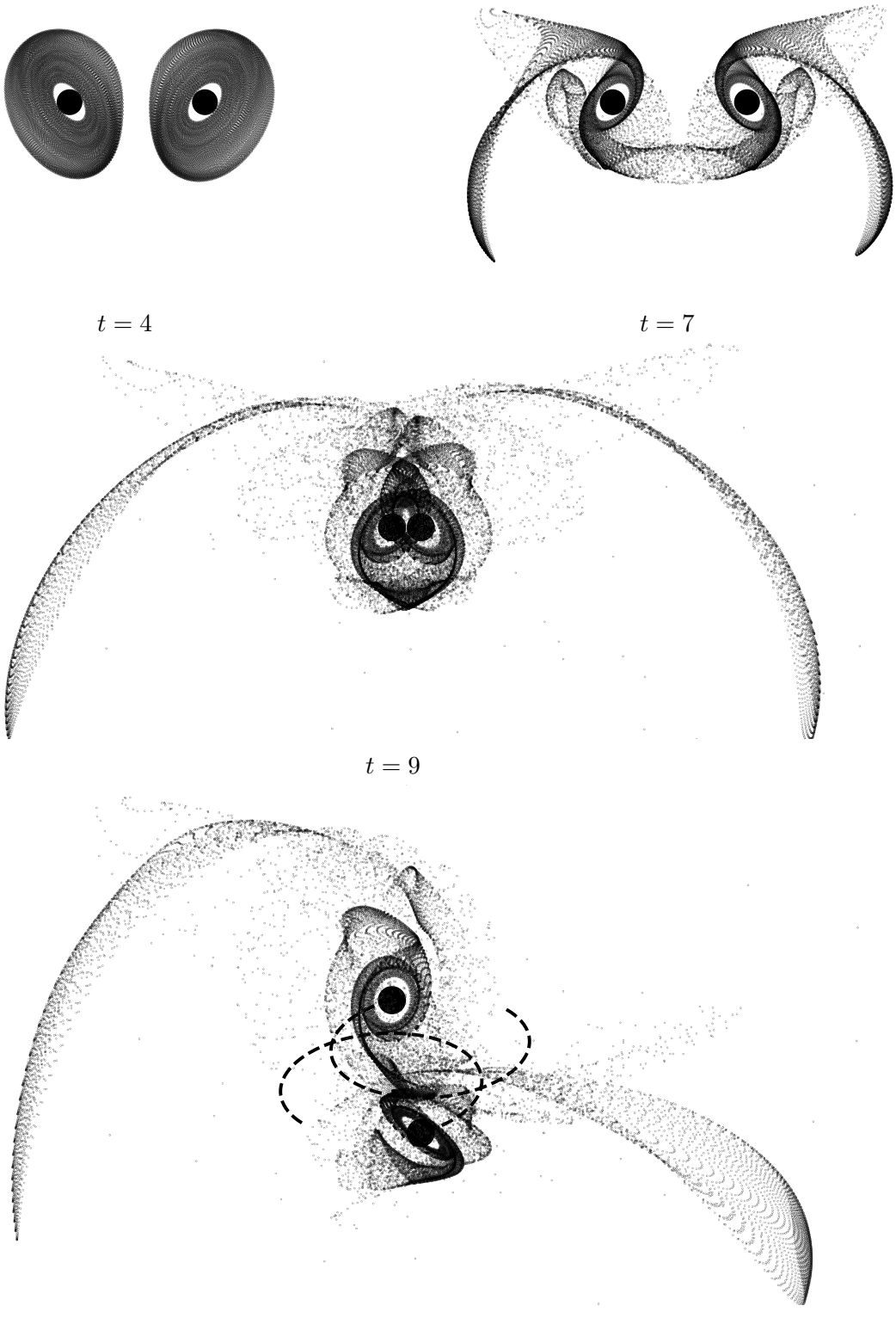

Figure 8: Simulated interaction between NGC 4038 and NGC 4039, commonly known as "The Antennae". The encounter was a direct passage where both galaxies had equal mass. In the first three subfigures, the projection plane is perpendicular to the orbit plane. However, in the second depiction of the galaxies at $t \overset{14}{=} 9$*, angle formed by the orbit plane and the projection plane is* $60°$*. In this passage,* $e \approx 0.44$ *and* $R_{min} \approx 18.57\,\mathrm{kpc}$*.*

[14] L. Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*, 159:98–103, July 1967.