VOLVOX COLONIES MOTION VISUALIZATION USING PYTHON

Sevi Nurafni[1*], Sparisoma Viridi[1,2]

[1]Master Program in Computational Science,Faculty of Mathematics and Natural Sciences,
Institut Teknologi Bandung, Bandung 40132, Indonesia
[2]Master Program in Physics,Faculty of Mathematics and Natural Sciences,
Institut Teknologi Bandung, Bandung 40132, Indonesia

ABSTRACT. Two colonies of Volvox globator can attached together and move together. this system can be modeled with a two-body system so as to produce translational, rotational, and combinational motion modes. The motion mode will be varied based on 3 axes, namely the x, y, and z axes so as to get 64 variations of motion modes. To make it easier to understand each motion mode, dynamic animation was created using the Python programming language with three important libraries, namely Granules, Numpy, and Matplotlib.

Keywords: Volvox globator, motion mode, Python

1 Introduction

Volvox is a genus of spherical green algae composed of thousands of cells [1] moving with flagella [2]. These multicellular organisms consist of 500-50,000 cells [3] of which 98% are somatic cells and the rest are gametes [4]. Volvocine green algae, ranging from unicellular Chlamydomonas to multicellular *Volvox* can be seen in Figure 1, have emerged as model organisms for a number of problems in biological fluid dynamics [5], such as the paired hydrodynamic interactions between them [6].

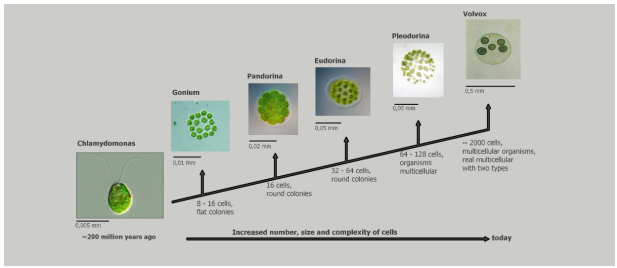


Figure 1: Evolution from unicellular to multicellular [7].

Drescher (2009) report that when two adjacent *Volvox* carteri colonies, they can "dance" and move together in two formations, namely the waltz and the minuet. Those two *Volvox* colonies cannot attached each other because the motion of the flagella of each colony produces a vortex [2]. Observation video of *Volvox* carteri colonies can be seen in Figure 2. This behavior is different from *Volvox* 3 shows two colonies of *Volvox* globator can attached each other and move together in harmony. Video observation obtained from Youtube [8] with the search keyword "Volvox Globator". Based on the discovery of this video, an idea emerged to visualize the approximate results of motion and position varying with time of the motion mode of two attached *Volvox* colonies.

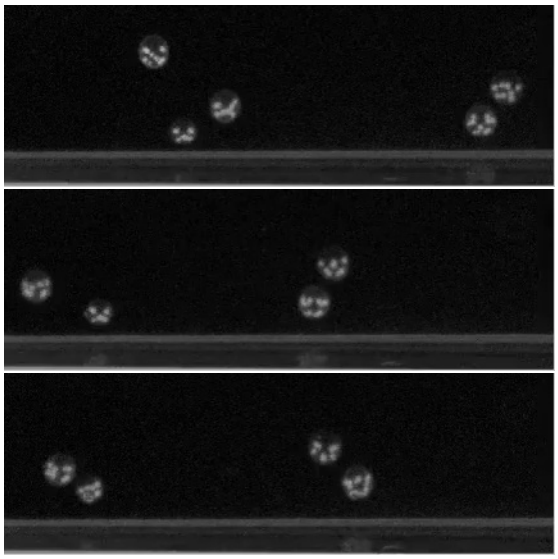


Figure 2: Volvox Carteri

The motion of the two *Volvox* colonies was modeled using classical mechanics[9], the two-body system. Consider the relative motion of a pair of three-dimensional particles that share the same point of contact. This contact interaction produces pure translational motion, rotational motion (rolling and twisting), and pure combination motion.

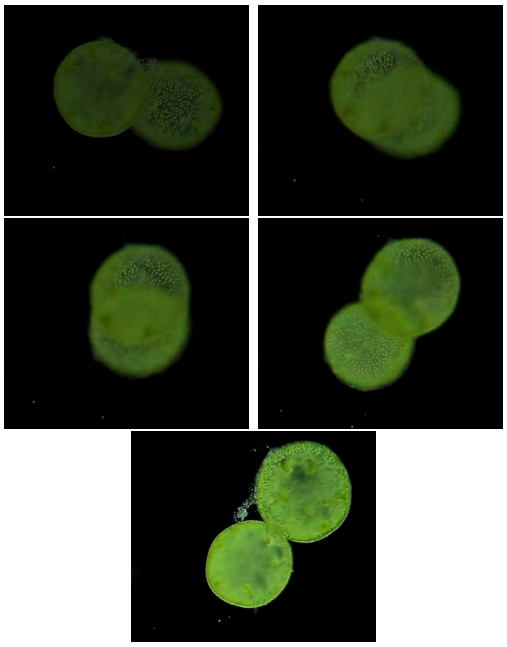


Figure 3: Observation video Volvox Globator

A more general visualization is to use a computer to implement a model to test it under different conditions, with the goal of learning how the model behaves. Therefore, the result of the visualization determines how well the system designed to represent the model can be represented. In other words, the process must be effective in order to provide the behavior of the actual model.

In this work, a visualization program was developed using the Python programming language. Python is a simple, predominant, well-organized interpreter language. Python is used to develop very powerful scientific applications, as well as to develop numerical computing applications, and is widely used [10]. Python often uses external libraries to perform scientific calculations [11]. Three important libraries for visualization with Python are Butiran, Numpy, and Matplotlib. The Butiran is library for is a library that has the wonderful ability to mathematically manipulate 3D vectors with explicit visualization. The Numpy is library for used for working with arrays. The Matplotlib is the Python library for plotting, data analysis and data visualization.

2 Two-body Systems

Figure 4 shows the two-body systems, the bodies will be referred as *Volvox* colonies. *Volvox* colony of mass m_1 whose position vector is $\vec{r_1}$ as well as a *Volvox* colony of mass m_2 with position vector $\vec{r_2}$. In terms of $\vec{r_1}$ and $\vec{r_2}$, the center of mass is located with position vector

$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2}}{m_1 + m_2} \tag{1}$$

Since already know the general motion of the center of mass, this information can be used to simplify th problem. To see how, it is necessary to define the relative distance vector

$$\vec{r} = \vec{r_2} - \vec{r_1} \tag{2}$$

distance-vector length $\vec{r}$,

$$\vec{r} = \vec{r_{b1}} + \vec{r_{b2}} \tag{3}$$

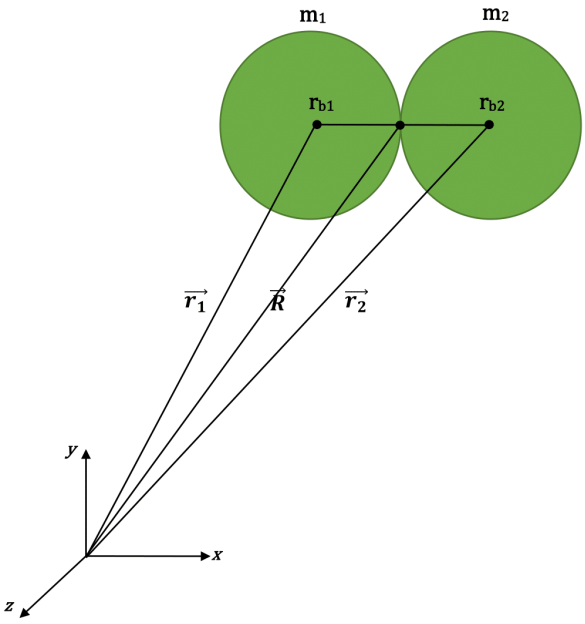


Figure 4: Two-body systems.

Here $\vec{r_1}$ and $\vec{r_2}$ are two vectors of the two colonies to the center of mass and are related to $\vec{r}$ by

$$\vec{r_1} = \vec{R} - \frac{m_2}{m_1 + m_2}\vec{r} \tag{4}$$

$$\vec{r_2} = \vec{R} - \frac{m_1}{m_1 + m_2}\vec{r} \tag{5}$$

The distance between the two centers of mass of each colony

$$\hat{r} = \frac{\vec{r}}{|\vec{r}|} \tag{6}$$

A system of two circles that can move in three dimensions, then spherical coordinates are used so that the position of the center of mass is

$$\hat{r} = \hat{x}\sin\theta\sin\phi + \hat{y}\cos\theta + \hat{z}\sin\theta\cos\phi \tag{7}$$

with

$$\theta = \theta_0 + \omega_\theta t \tag{8}$$

$$\phi = \phi_0 + \omega_\phi t \tag{9}$$

Use Newton's laws for uniform motion of an object in a straight line

$$\vec{R} = \vec{R_0} + \vec{V}t \tag{10}$$

with velocity vector in (x, y, z) direction, $\vec{V}$

$$\vec{R} = \hat{x}V_x + \hat{y}V_y + \hat{z}V_z \tag{11}$$

using Cartesian coordinates, the two circles are modeled from the $(0, 0)$ axis hence that we get,

$$\begin{aligned}\vec{R} &= \vec{R_0} + \vec{V}t \\ &= (x_0\hat{x} + y_0\hat{y} + z_0\hat{z}) + (\hat{x}V_x + \hat{y}V_y + \hat{z}V_z)t\end{aligned} \tag{12}$$

then obtained

$$\vec{R} = (x_0 + v_x t)\hat{x} + (y_0 + v_y t)\hat{y} + (z_0 + v_z t)\hat{z} \tag{13}$$

Substitute equation (4) with equation (3), (7) and (13) so that it is obtained,

$$\begin{aligned}\vec{r_1} = (x_0 + v_x t)\hat{x} + (y_0 + v_y t)\hat{y} + (z_0 + v_z t)\hat{z} \\ -\frac{m_2}{m_1 + m_2}(r_{b1} + r_{b2}) * [\hat{x}\sin(\theta_0 + \omega_\theta t) \\ \sin(\phi_0 + \omega_\phi t) + \hat{y}\cos(\theta_0 + \omega_\theta t) \\ +\hat{z}\sin(\theta_0 + \omega_\theta t)\cos(\phi_0 + \omega_\phi t)]\end{aligned} \tag{14}$$

$$\begin{aligned}\vec{r_2} = (x_0 + v_x t)\hat{x} + (y_0 + v_y t)\hat{y} + (z_0 + v_z t)\hat{z} \\ -\frac{m_1}{m_1 + m_2}(r_{b1} + r_{b2}) * [\hat{x}\sin(\theta_0 + \omega_\theta t) \\ \sin(\phi_0 + \omega_\phi t) + \hat{y}\cos(\theta_0 + \omega_\theta t) \\ +\hat{z}\sin(\theta_0 + \omega_\theta t)\cos(\phi_0 + \omega_\phi t)]\end{aligned} \tag{15}$$

with $\vec{r_1}$ in the direction (x, y, z), respectively

$$\begin{aligned}x = (x_0 + v_x t) - \frac{m_2}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\sin(\theta_0 + \omega_\theta t)\sin(\phi_0 + \omega_\phi t)]\end{aligned} \tag{16}$$

$$\begin{aligned}y = (y_0 + v_y t) - \frac{m_2}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\cos(\theta_0 + \omega_\theta t))]\end{aligned} \tag{17}$$

$$\begin{aligned}z = (z_0 + v_z t) - \frac{m_2}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\sin(\theta_0 + \omega_\theta t)\cos(\phi_0 + \omega_\phi t)]\end{aligned} \tag{18}$$

with $\vec{r_2}$ in the direction (x, y, z), respectively

$$\begin{aligned}x = (x_0 + v_x t) - \frac{m_1}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\sin(\theta_0 + \omega_\theta t)\sin(\phi_0 + \omega_\phi t)]\end{aligned} \tag{19}$$

$$\begin{aligned}y = (y_0 + v_y t) - \frac{m_1}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\cos(\theta_0 + \omega_\theta t))]\end{aligned} \tag{20}$$

$$\begin{aligned}z = (z_0 + v_z t) - \frac{m_1}{m_1 + m_2}(r_{b1} + r_{b2}) \\ *[\sin(\theta_0 + \omega_\theta t)\cos(\phi_0 + \omega_\phi t)]\end{aligned} \tag{21}$$

3 Illustration of Motion Mode

The motion of the two *Volvox* colonies is modeled with a two body system, where the center of mass (COM) is performing simple translational motion, while the two colonies center is rotating around the COM with roll or twist motions. These motions are varied based on the x, y, and z axes so as to produce 64 variations of motion.The default coordinate system in this work is left-handed with the x, y, and z axes point right, up and forward, respectively. Illustrations of 64 variations of motion can be seen in figure 5, figure 6, figure 7, and figure 8.

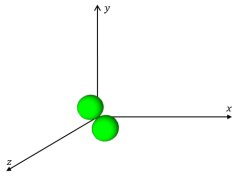


Figure 5: Motionless

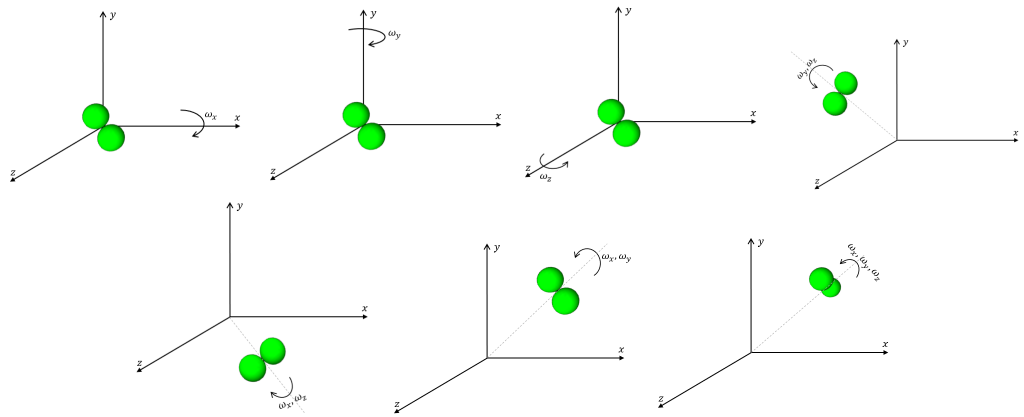


Figure 6: Rotational motion.

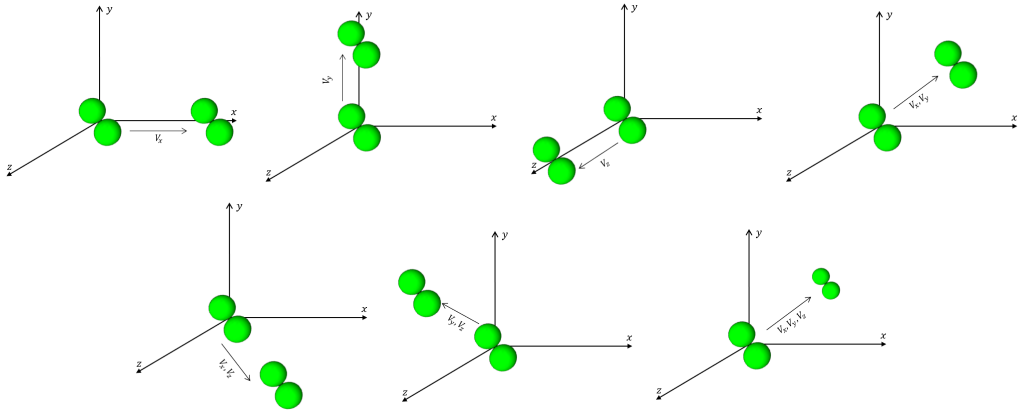
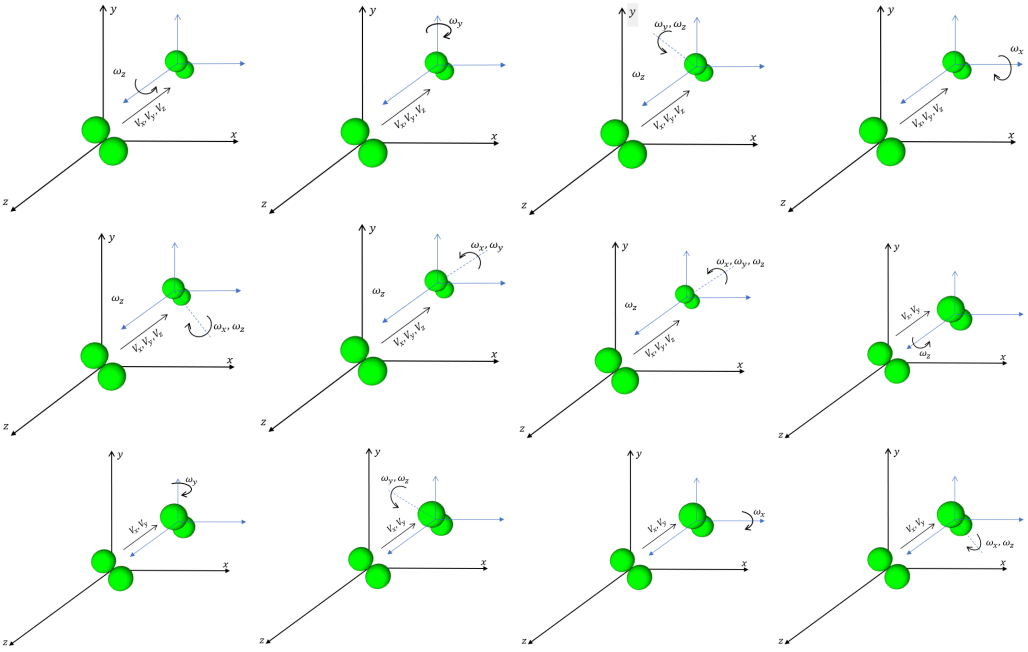
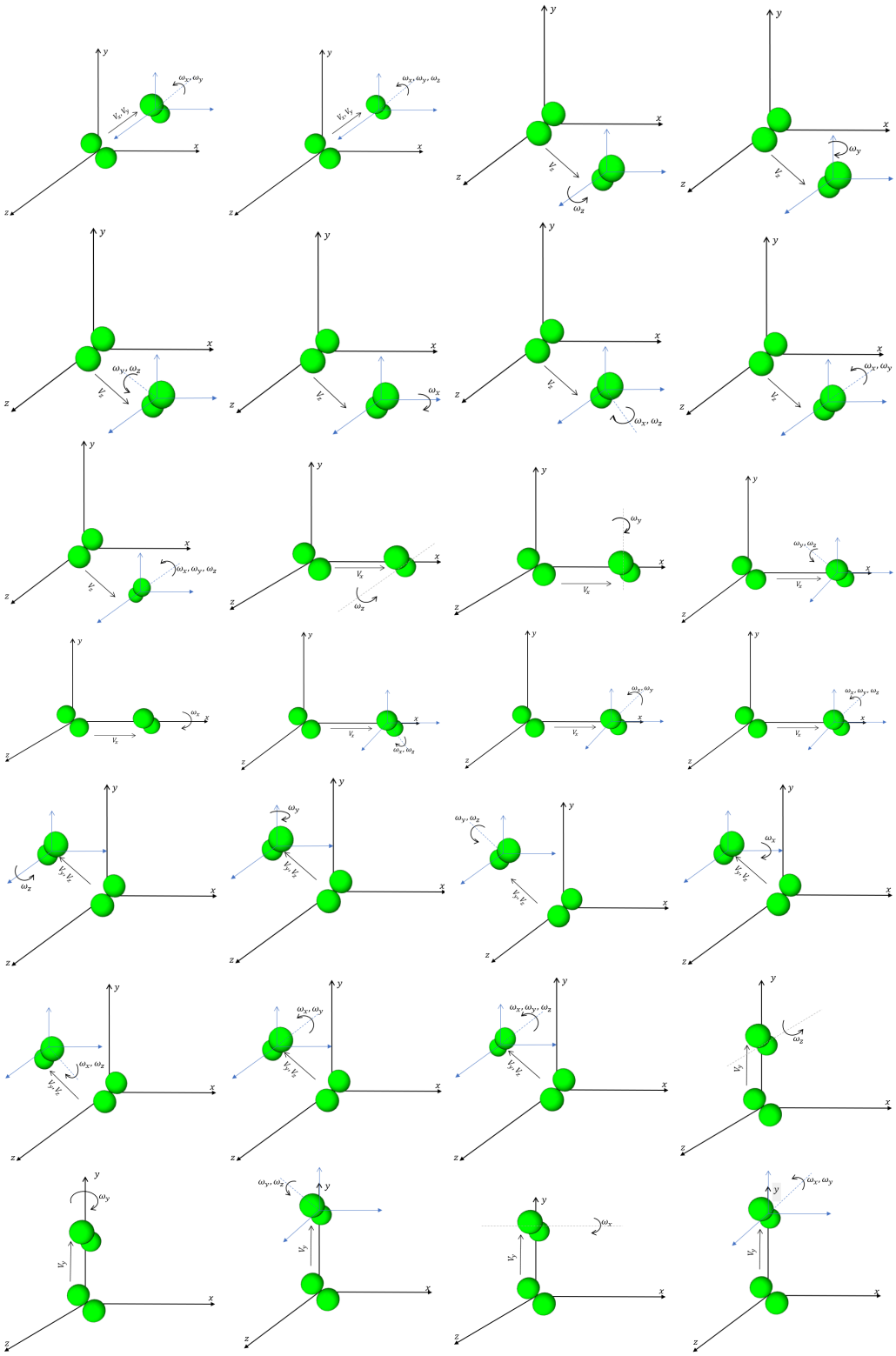


Figure 7: Translational motion.





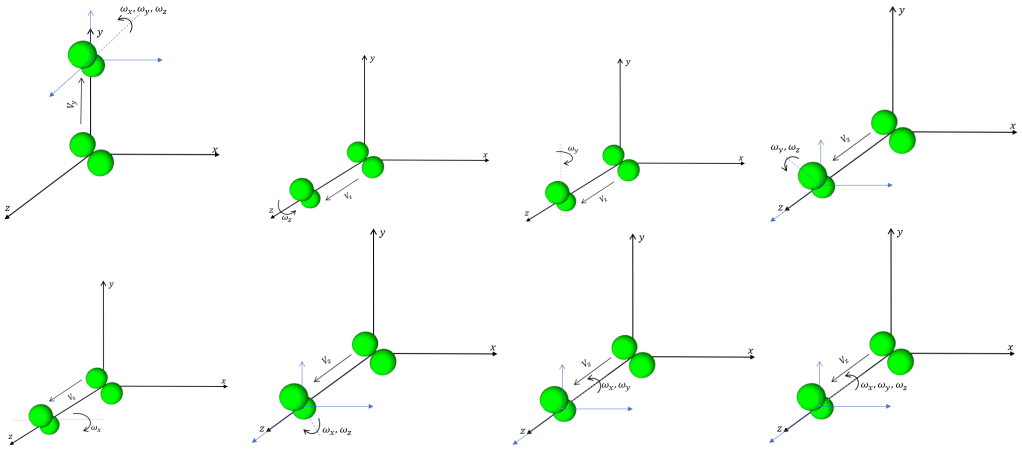


Figure 8: Combinational motion.

4 Flowchart

Target dari pekerjaan ini adalah membuat animasi gerak Volvox berdasarkan persamaan pada chapter sebelumnya.Pada gambar 9 adalah Flowchart utama yang digunakan dalam pembuatan program sedangkan gambar 10 adalah flowchart dari function-function yang ada di flowchart utama.

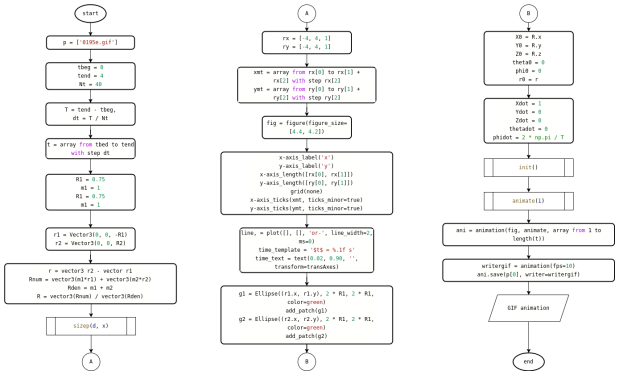


Figure 9: Main Flowchart.

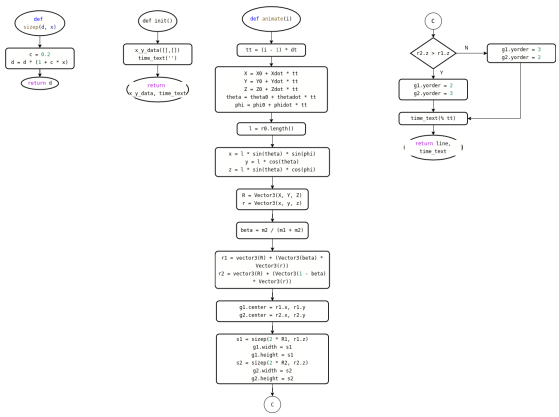


Figure 10: Function Flowchart.

Broadly speaking, this flowchart explains several things, namely setting parameters for plotting, creating time arrays, defining two colonies, calculating vectors r and R, defining colony size based on z perspective, setting ranges x and y for plotting, configure axes, configure curve, set initial position, sets kinematic velocities, displays the animation, creates the animation in GIF format, and saves it.

5 Motion Control

5.1 Time Setting

First, it should start by importing the appropriate Python libraries.

```
from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.patches import Ellipse
from butiran.vect3 import Vect3
```

Then, have to make a time array that will be used for the length of time animating the motion. The t time array will be created using the $arange()$ module of the numpy library. To use $range()$ specified arguments are required, namely $start, stop, andstep$. $Start$ will use the value of the $tbeg$ variable which is the start time, $stop$ will use the value of the $tend + dt$ variable which is the end time plus the amount of time each timestep covers, and $step$ will use the value of the dt variable .

```
tbeg = 0
tend = 4
T = tend - tbeg
Nt = 40
dt = T / Nt
t = np.arange(tbeg, tend + dt, dt)
```

The time (T) used is the end time($tend$) minus the start time($tbeg$). Meanwhile, to get the value of dt is the division between time(T) with the number of timesteps want to calculate (Nt).

5.2 Representation of Three Dimensions

The motions of the two volvox colonies are represented in three-dimensional space for the position and direction vectors. With the x axis to the right, the y axis to the top, and the z axis to the screen. To represent 3D vectors in the program, the Grain library with submodule Vect3 is used. Vect3 is an ordered triplet of numbers (labeled x, y,and z), which can be used to represent a number of things. The length will always be the straight-line distance from $(0, 0, 0)$ to (x, y, z) and the direction is also measured from $(0, 0, 0)$ towards (x, y, z). Vect3 has several methods for vector calculations, namely sub() for vector subtraction, $mul()$ for vector multiplication, $add()$ for vector multiplication, and the $div()$ method for vector division.

```
R1 = 0.5
r1 = Vect3(-R1, 0, 0)
m1 = 1
R2 = 0.5
r2 = Vect3(R2, 0, 0)
m2 = 1

r = Vect3.sub(r2, r1)
Rnum = Vect3.add(Vect3.mul(m1, r1),
    Vect3.mul(m2, r2))
Rden = m1 + m2
R = Vect3.div(Rnum, Rden)

r1 = Vect3.sub(R, Vect3.mul(beta, r))
r2 = Vect3.add(R, Vect3.mul(1 - beta, r))
```

Use the Vec3 submodule from the Details library in this work to define the position vectors $\vec{r_1}$ and $\vec{r_2}$ of two Volvox colonies, calculate the distance vector, and the position vector of the center of mass.

5.3 Configure Spheres and Curve

In this section we will use several sub-modules from the matplotlib library. First create an array of x and y to create a figure window, create a single axis in the figure, and then create our line object which will be modified in the animation.

To plot two circles which are Volvox colonies use the $Ellipse()$ Sub module. Certain parameters needed are center coordinates, width and height. The center coordinates are based on the x and y values of the position vectors $\vec{r_1}$ and $\vec{r_2}$. each colony is defined on the variables $g1$ and $g2$. Ellipse is an Ellipse patch. It accepts tuples from the center point, and then weigh and height. The fc argument gives a green, without a border.

```
rx = [-2, 2, 0.5]
ry = [-2, 2, 0.5]
xmt = np.arange(rx[0], rx[1] + rx[2],
    rx[2])
ymt = np.arange(ry[0], ry[1] + ry[2],
    ry[2])

fig = plt.figure(figsize=[2.4, 2.2])

ax = fig.add_subplot()
ax.set_xlabel('$y$')
ax.set_ylabel('$z$')
ax.set_xlim([rx[0], rx[1]])
ax.set_ylim([ry[0], ry[1]])
ax.grid(which='both')
ax.set_xticks(xmt, minor=True)
ax.set_yticks(ymt, minor=True)

axis to give the effect
fig.tight_layout(rect=[-0.04, -0.04, 1.03,
    1.04])

line, = ax.plot([], [], 'or-', lw=2, ms=0)
time_template = '$t$ = %.1f s'
time_text = ax.text(0.02, 0.90, '',
    transform=ax.transAxes)

g1 = Ellipse(
   (r1.x, r1.y), 2 * R1, 2 * R1,
   edgecolor='g', fc='#cfc', lw=1
)
ax.add_patch(g1)
g2 = Ellipse(
   (r2.x, r2.y), 2 * R2, 2 * R2,
   edgecolor='g', fc='#cfc', lw=1
```

```
)
ax.add_patch(g2)
```

5.4 Create Animation

Do animate Volvox motion need to define global variables for initial position and kinematic velocities. Variabel untuk initial position terdiri dari $X0, Y0, Z0$ yang nilainya berdasarkan vector posisi dari masing-masing sumbu, $\theta_0, \phi_0, dan r_0$. Variabel untuk kinematic velocities terdiri dari $Xdot, Ydot, Zdot$ untuk kecepatan pada sumbu x, y, z, $thetadot$ merupakan $\omega_\theta dan phidot merupakan \omega_\phi$.

forthcoming, create the functions which make the animation happen. $init()$ is the function which will be called to create the base frame upon which the animation takes place. Here use just a simple function which sets the line data to nothing. It is important that this function return the line object, because this tells the animator which objects on the plot to update after each frame.

The next piece is the animation function. It takes a single parameter, the frame number i and and draws motion that depends on i. Finally, create the animation object. This object needs to persist, so it must be assigned to a variable. Then choose an animation of an array of t frames with a delay of 70ms between frames. The blit keyword is an important one, this tells the animation to only re-draw the pieces of the plot which have changed. The time saved with blit=True means that the animations display much more quickly.

```
tt = (i - 1) * dt

    X = X0 + Xdot * tt
    Y = Y0 + Ydot * tt
    Z = Z0 - Zdot * tt
    theta = theta0 + thetadot * tt
    phi = phi0 + phidot * tt

    l = r0.len()
    x = l * sin(theta) * cos(phi)
    y = l * sin(theta) * sin(phi)
    z = l * cos(theta)

    R = Vect3(X, Y, Z)
    r = Vect3(x, y, z)
    beta = m2 / (m1 + m2)

    r1 = Vect3.sub(R, Vect3.mul(beta, r))
    r2 = Vect3.add(R, Vect3.mul(1 - beta, r))

    g1.center = r1.y, r1.z
    g2.center = r2.y, r2.z

    s1 = sizep(2 * R1, r1.x)
    g1.width = s1
    g1.height = s1

    s2 = sizep(2 * R2, r2.x)
    g2.width = s2
    g2.height = s2

    if r2.z > r1.z:
        g1.yorder = 2
        g2.yorder = 3
    else:
        g1.yorder = 3
        g2.yorder = 2

    time_text.set_text(time_template % tt)

    return line, time_text

ani = animation.FuncAnimation(
    fig, animate, np.arange(1, len(t)),
    interval=70, blit=True, init_func=init
)

writergif = animation.PillowWriter(fps=10)
ani.save(p[0], writer=writergif)
```

6 Result

Figure 11, Figure 12, and Figure 13 are sequence images of translational motion modes on the x-axis, rotational x-axis, and combinational on the x-axis. Combination motion mode is experimented with coding in order to get motion as shown in the illustration. This happens because of the lack of understanding in modeling the combination motion.

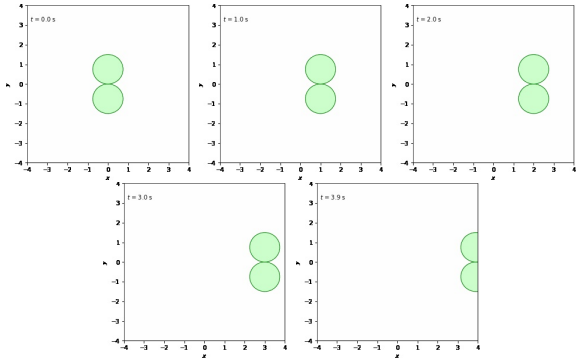


Figure 11: Translational Motion

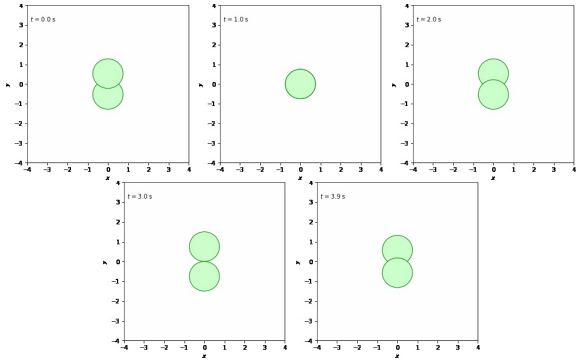


Figure 12: Rotational Motion

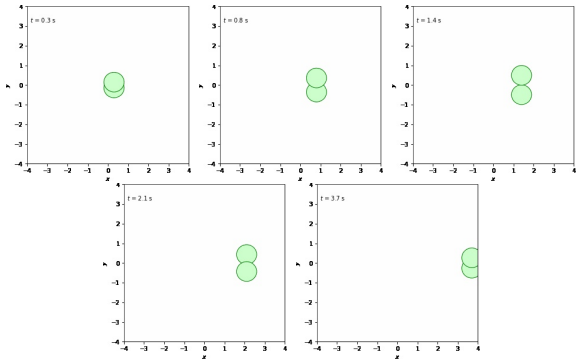


Figure 13: Rotational Motion

7 Conclusions

This work has described how to implement dynamic animation in Python. Animations are an interesting way of demonstrating how Volvox Globator effectively use the principles of classical mechanics in their swimming. Obtained 64 variations of the motion of the Volvox globator Colony based on the two-body system model.

References

[1] Kirk, D. L. (1998). *Volvox: A search for the molecular and genetic origins of multicellularity and cellular differentiation*. Cambridge University Press.

[2] Ishikawa, T., Pedley, T., Drescher, K., & Goldstein, R. E. (2020). Stability of dancing volvox. *Journal of Fluid Mechanics*, *903*.

[3] Kirk, D. L. (2005). A twelve-step program for evolving multicellularity and a division of labor. *BioEssays*, *27*(3), 299–310.

[4] Herron, M. D. (2016). Origins of multicellular complexity: Volvox and the volvocine algae.

[5] Goldstein, R. E. (2015). Green algae as model organisms for biological fluid dynamics. *Annual review of fluid mechanics*, *47*, 343–375.

[6] Ishikawa, T., Simmonds, M., & Pedley, T. J. (2006). Hydrodynamic interaction of two swimming model micro-organisms. *Journal of Fluid Mechanics*, *568*, 119–160.

[7] Hallmann, A., & Grotjohann, N. (2011). Die einäugigen unter den blinden-phototaxis als wettbewerbsvorteil. *Unterricht Biologie. Zeitschrift für die Sekundarstufe*, *365*.

[8] Smith, C. (2012). *Volvox globator*. Youtube. https://www.youtube.com/watch?v=VdDBoMujZsw&feature=youtu.be

[9] Chernous'ko, F. (2008). The optimal periodic motions of a two-mass system in a resistant medium. *Journal of Applied Mathematics and Mechanics*, *72*(2), 116–125.

[10] Ranjani, J., Sheela, A., & Meena, K. P. (2019). Combination of numpy, scipy and matplotlib/pylab -a good alternative methodology to matlab - a comparative analysis. *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, 1–5. https://doi.org/10.1109/ICIICT1.2019.8741475

[11] Borcherds, P. (2007). Python: A language for computational physics. *Com-*

puter Physics Communications, *177*(1-2), 199–201.

[12] Duran, J., & Mazozi, T. (1999). Granular boycott effect: How to mix granulates. *Physical Review E*, *60*(5), 6199.

[13] *Pyrthon package index - butiran.* (n.d.). Retrieved February 7, 2021, from https://pypi.org/project/butiran/

[14] Drescher, K., Leptos, K. C., Tuval, I., Ishikawa, T., Pedley, T. J., & Goldstein, R. E. (2009). Dancing volvox: Hydrodynamic bound states of swimming algae. *Physical review letters*, *102*(16), 168101.