Trajectory optimization with gradient descent for a variable-volume float

by

J.D. Snijders

Bachelor End Project
BSc Applied Mathematics
BSc Applied Physics
Delft University of Technology

Student number: 5346274
Supervisors: Prof. dr. ir. M. Verlaan
Prof. dr. ir. C.R. Kleijn

TU Delft

Abstract

In the realm of fluid dynamics and particle transport, the control of particle trajectories represents a formidable challenge. It would be useful to be able to optimally navigate an oceanographic float from one pre-set location to another by solely changing its buoyancy. In this thesis, a first step in discovering whether this is possible and what optimization strategy can be used is taken.

To do so, first, the physical situation is translated into a mathematical model. Then, an optimization strategy for changing the buoyancy to optimally travel to a set location is constructed. The strategy is based on gradient descent and implemented in Python. Four different definitions of an optimal trajectory to a target location are considered, those are 1) any trajectory that leads to the target location, 2) the most time-efficient trajectory, 3) the most energy-efficient trajectory, and 4) a trajectory that is both time and energy-efficient.

The optimization strategy is tested for five different starting and target locations for a small spherical float in an idealized two-dimensional linear flow field. It is concluded that it is possible to use the optimization strategy to navigate a float using buoyancy changes for all four optimization objectives, although the current implementation is not efficient enough for targets far away.

The first objective of future research should be to increase the coding efficiency. Thereafter, other steps toward a more realistic situation can be taken, such as testing for non-linear flow fields, three-dimensional fields, and bigger floats.

J.D. Snijders
Thursday 2nd November, 2023

Contents

1. Introduction

Although oceans cover more than 70% of the earth, we do not know everything that lies within. Fascination with the water, its flow, and its ecosystems is timeless, and for most of history, exploring the oceans was done by manned expeditions. However, in the past decades, technological developments have made it possible to send autonomous underwater drones to do the work without constant monitoring. [15]

There are different kinds of autonomous underwater vehicles (AUVs), classified by method of movement. Propelled AUVs have propellers and are more maneuverable than other kinds. Underwater gliders change their buoyancy to move up and down, while their hydrofoils (wings) change this into more or less controlled forward motion.[28] Oceanographic floats drift with the currents, but their density can be changed, causing the floats to rise up and down through the water column while collecting data, see figure 1.

How long an AUV can be used, depends mainly on the battery life. [1] A propelled AUV uses more energy than an AUV that drifts with the currents. Therefore, it is interesting to look at the possibilities of navigating buoyancy-changing devices, specifically floats, toward set destinations. At first glance, it might not seem possible to control the destination of a float, but since the velocity of the surrounding water is not equal at all depths and the vertical movements can be influenced by the buoyancy, the trajectory of the float can be influenced. This is similar to how a hot-air balloon can be steered by adding heat to decrease the density to enter an air layer where the currents are different. [8] Within the sea, another similar effect can be seen in some fish larvae: some larvae are known to be able to migrate from the open sea towards inshore nursery areas, probably by laying on the seabed during ebb tides and moving upwards during flood tides. [22] Although it is not certain if larvae move upwards by swimming, by changing their buoyancy, or by something different, this does raise confidence that various locations could be reached by only using the water currents.

While there are many studies available on path optimization in gliders and propelled AUVs [16][28][15], not so much research has been done on the trajectory planning of floats. Floats are generally used as profiling floats, for example in the Argo program, [1] and thus not to navigate to a specific point. Additionally, underwater gliders and propelled AUVs are not usually deployed in coastal areas, as they are expensive and could easily be damaged by human activity or be washed ashore. [10]

This thesis is a first step to discovering to what extent the variable buoyancy of a float can be used for navigation, with the intention to determine the viability of a small and cheap AUV that can be navigated toward set locations in coastal regions. Should significant steering be possible, then the results can also be used to make underwater gliders and propelled AUVs more energy efficient, as those are affected by the water flow too, especially when their velocity relative to the water is low. The aim of this thesis is to determine whether it is possible to optimally navigate a small ($D = 10^{-4}\mathrm{m}$) spherical float to a target location in an idealized flow field with an optimization strategy based on gradient descent. The idealized flow field is two-dimensional and linear, and the water has a constant density and viscosity, and the Reynolds number for the float in the flow field is lower than 1. Four different definitions of op-

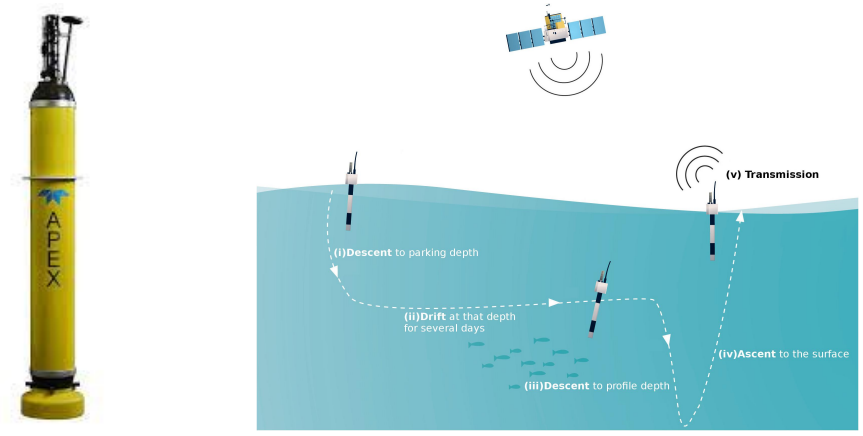


Figure 1: An apex float, specifically one built by Teledyne Webb Research [21]. The sketched example trajectory was made by X. André et al.[2]

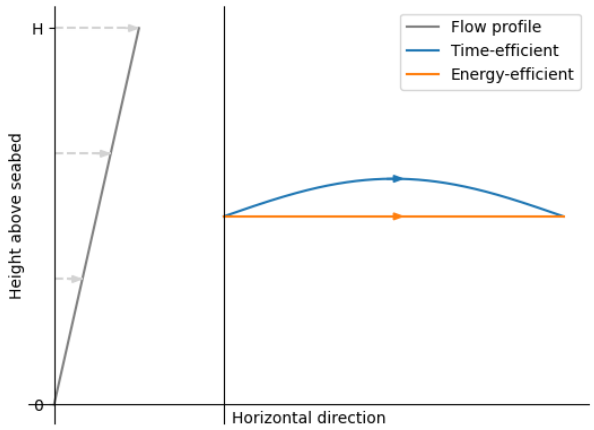


Figure 2: An example of a flow field (left) and two trajectories that a float can follow (right). Both trajectories have the same starting and target location, but the higher path will lead the float to its destination faster than the lower path, while the lower path requires less energy, as the buoyancy does not have to be changed.

timally navigating to a target location are considered, those are 1) following any trajectory that leads to the target location, 2) following the most time-efficient trajectory, 3) following the most energy-efficient trajectory, and 4) following a trajectory that is both time and energy-efficient. The difference between these definitions of optimal navigation is shown in the example in figure 2: if the flow is to the right and faster in higher water, then a float that moves through higher water layers will be faster than a float that does not, while a float that does not change buoyancy will require less energy. An optimization strategy is constructed, and with an implementation in Python, it is tested whether the strategy could be used to determine the required buoyancy changes to travel to different target locations in time or energy-efficient ways.

The structure of this thesis is as follows. In chapter 2, the forces on and the movement of a submerged spherical float are converted into a mathematical model and simplified for the idealized two-dimensional linear flow field. In chapter 3, the optimization strategy for optimizing the buoyancy changes is described, and it is determined how the position, time, and energy can be combined into one cost function, with weighting factors that depend on whether the float should follow a time or energy-efficient path. Results of optimization with the implemented strategy are presented in chapter 4 and discussed in chapter 5. Chapter 6 contains the conclusion.

2. The Model

In this chapter, the physical situation of a submerged spherical buoyancy-changing float is translated into a mathematical model.

2.1. Equations of motion

The velocity $\vec{v}$ of an object submerged in a fluid is not necessarily equal to that of the fluid. The movement depends on the forces applied to the object, as described by Newton's second law.

$$m\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \sum \vec{F} \tag{1}$$

The different components of this equation will be detailed in the next sections: first, the mass m, and then the forces $\vec{F}$ at work.

The rotation of the body can be described with the equation of conservation of angular momentum in the body-centered frame:

$$\mathbf{I}\frac{\mathrm{d}\vec{\omega}}{\mathrm{d}t} + \vec{\omega} \times (\mathbf{I}\vec{\omega}) = \sum \vec{M} \tag{2}$$

In this equation, $\mathbf{I}$ is the mass moment inertia tensor, $\vec{\omega}$ is the angular velocity, and $\sum M$ denotes the sum of the torque moments.

In the rest of this thesis, the rotation of the float will be neglected. This simplification is only possible when the angular velocity is low, as a high angular velocity would cause the pressure on one side of the sphere to rise and on the other side to decrease, which would cause a lift force that deflects the sphere and can not be neglected. [6]

2.1.1. Added mass

The added mass is not a real mass, but rather a clever way of adding the inertia. When something is submerged in a fluid, the fluid must flow around it. Therefore, some of the fluid must accelerate if the object does and vice versa. This can be incorporated in equation (1) by adding an imaginary mass component m_{add} to the conventional mass m_f of the float. Throughout the whole thesis, subscript w will refer to the water flow and subscript f to the float. For a sphere, the added mass is related to the density of the surrounding fluid ρ_w and the diameter of the sphere as [5]

$$m_{add} = \frac{\pi D^3}{12}\rho_w \tag{3}$$

This is half of the mass of the displaced water. The total mass that should be considered in eq. (1) can be expressed in terms of the volume V and the density of both the sphere and the fluid:

$$m_{tot} = (\rho_f + \frac{\rho_w}{2})V \tag{4}$$

2.1.2. Forces on the float

In addition to the previously mentioned neglected lift force, three other forces act on the float: the gravitational force F_g, the buoyancy force F_a, and the drag force F_d. When applied to a sphere, in a frame of reference with the direction of $\hat{z}$ opposite to the gravitational force. Those forces can be calculated with the following equations:

$$\vec{F}_g = -\rho_f\left(\frac{\pi}{6}D^3\right)g\ \hat{z} \tag{5}$$

$$\vec{F}_a = \rho_w\left(\frac{\pi}{6}D^3\right)g\ \hat{z} \tag{6}$$

$$\vec{F}_d = -\frac{1}{2}C_d\left(\frac{\pi}{4}D^2\right)\rho_w|\Delta\vec{v}|\Delta\vec{v} \tag{7}$$

Where g is the gravitational constant ($9.81\mathrm{m/s^2}$) and $\Delta\vec{v}$ is the velocity relative to the water, defined as $\Delta\vec{v} = \vec{v}_f - \vec{v}_w$. The drag coefficient C_d depends on the Reynolds number, which is defined as

$$\mathrm{Re} = \frac{\rho_w|\Delta\vec{v}|D}{\mu} \tag{8}$$

μ is the dynamic viscosity of the fluid. When $10^3 < \mathrm{Re} < 3\ 10^5$, $C_d \approx 0.47$ for a smooth sphere.[12] For $\mathrm{Re} < 0.1$, a situation that is also called Stokes flow, George Stokes found that C_d and Re of a smooth sphere can be linked through [25]

$$C_d = \frac{24}{\mathrm{Re}} \tag{9}$$

Although this relation is not valid for higher Reynolds numbers, it is still reasonable to use it for Reynolds numbers up to 1. [12] Thus, for $\mathrm{Re} < 1$, the drag force can be simplified to

$$\vec{F}_d = \vec{F}_s = -3\pi D\mu\Delta\vec{v} \tag{10}$$

2.1.3. Summary equations of motion

When equations (4), (5), (6), and (7) are substituted in eq. (1), the resulting equation of motion can be simplified to

$$\frac{\mathrm{d}\vec{v}_f}{\mathrm{d}t} = \frac{\rho_w - \rho_f}{\rho_f + \frac{\rho_w}{2}} g\,\hat{z} - \frac{3}{4}\frac{C_d}{D}\frac{\rho_w}{\rho_f + \frac{\rho_w}{2}}|\Delta\vec{v}|\Delta\vec{v} \tag{11}$$

or, when eq. (10) can be used,

$$\frac{\mathrm{d}\vec{v}_f}{\mathrm{d}t} = \frac{\rho_w - \rho_f}{\rho_f + \frac{\rho_w}{2}} g\,\hat{z} - \frac{18\mu}{(\rho_f + \frac{\rho_w}{2})D^2}\Delta\vec{v} \tag{12}$$

ρ_f and D change when the buoyancy of the float is altered, and ρ_w and μ are not constant in general. However, in this thesis, they are assumed to be constant, with values $\rho_w = 1.026\ 10^3 \mathrm{kg/m^3}$ and $\mu = 1.2\ 10^{-3}\mathrm{Pa\ s}$. Those values correspond to a temperature of $15°C$ and a salinity of $35\mathrm{g/kg}$. [24]

2.2. Change in buoyancy

There are different methods to change the buoyancy of an object. The buoyancy depends on the density of the object, thus can be altered by changing the mass while maintaining the volume, changing the volume while maintaining the mass, or a combination of those. While the buoyancy of underwater gliders and floats is commonly controlled with a hydraulic pump that moves oil in and out of an external bladder, [1] inspiration for this thesis is provided by another design provided by nature: swim bladders in fish and fish larvae. Both are examples of changing the volume while keeping the mass (almost) constant. In fish, oxygen and other gasses flow through veins to and from the swim bladder. There, clever use of biological processes allows the fish to secrete gasses into the bladder, despite the high pressure.[13] The shortage or excess amount of dissolved gasses in the veins can be restored with little mass change when the blood flows through the gills.

The volume V_{SB} of a swim bladder changes due to four processes: the changing water pressure, leakage of gasses from the bladder, absorption of gasses from the bladder, and secretion of gasses into the bladder. Those processes will be explained first. Then, a differential equation describing the total change in volume will be derived.

2.2.1. Water pressure

The volume of a gas depends on the pressure. In water, the hydrostatic pressure P can be derived from Pascal's law: [20]

$$P(d) = P_0 + \rho_w g d \tag{13}$$

P_0 is the atmospheric pressure ($1.01325\ 10^5\mathrm{Pa}$) and d is the depth in m.

A swim bladder is adapted for a certain pressure, but the wall of such a bladder can stretch easily, such that the internal and external pressures are equal and the enclosed gas obeys the ideal gas law for pressures between 0.5 and 1.4 times the pressure to which the bladder is adapted.[11] For example, this implies that a bladder that is adapted to $5\mathrm{m}$ depth could be used on depths between $0\mathrm{m}$ (as the float or fish will not rise out of the water) and $7\mathrm{m}$, while one that is adapted to $20\mathrm{m}$ could be used between $5\mathrm{m}$ and $28\mathrm{m}$. Thus, when the pressure is between the imposed boundaries,

$$V_{SB}(t) = \frac{nRT}{P(d(t))} \tag{14}$$

n, R and T are the number of moles of gas within the bladder, the universal gas constant, and the temperature.

2.2.2. Leakage

Because a swim bladder is not impermeable, gasses will diffuse into the surrounding tissues. [9] The volume change due to leakage $\frac{\mathrm{d}V_L}{\mathrm{d}t}$ in m^3/s is negative and proportional to the surface S of the bladder and the pressure difference ΔP between the gasses inside the bladder, $P(d)$, and the gasses dissolved in the surrounding water, P_0 [14][26]

$$\frac{\mathrm{d}V_L}{\mathrm{d}t} = -GS\Delta P = -GS(P(d) - P_0) \tag{15}$$

G is a permeability constant, a realistic value in fish can be ca. $1.5\ 10^{-13}\mathrm{m}^3\mathrm{O}_2\ \mathrm{m}^{-2}\mathrm{Pa}^{-1}\mathrm{s}^{-1}$.[26] The surface area of a spherical swim bladder can be related to the volume through

$$S = \pi\left(\frac{6V_{SB}}{\pi}\right)^{\frac{2}{3}} \tag{16}$$

2.2.3. Absorption (active deflation)

Fish can deflate their swim bladder by reabsorbing gasses from the bladder into the bloodstream.[9] The maximum rate depends on the properties of the fish blood: the amount of blood flowing to the bladder, the specific gasses, and the hemoglobin binding sites in the blood, and only to a lesser extent on the depth of the fish.[11] However, the optimization technique that will be tested is intended for broad purposes, and the exact rate is only important for very specific cases, thus an estimation for the maximum rate will suffice.

It is assumed that the reabsorption of gasses does not cost energy, as the gasses diffuse from a location with a high concentration of gasses to blood with a lower concentration.

2.2.4. Secretion (active inflation)

Like with absorption, the amount of gas that a fish can secrete into the bladder depends on the amount of blood flowing to the bladder and the amount of gasses in the blood. [11] Again, the specific rates are unimportant for this research.

The process of inserting gas into the swim bladder costs energy. Firstly when the gasses are filtered from the blood and compressed from the partial pressure in the blood P_{blood} to $P(d)$ and secondly when the compressed gasses are added to the swim bladder. When a short time is considered and the volume changes slowly, the second process can be seen as quasi-static. Additionally, the pressure adapts to the external pressure, thus is nearly constant on a short time scale, thus the process is also isobaric. The work associated with a quasi-static isobaric process is equal to

$$\Delta W = \int \mathrm{d}V = P(d)\Delta V_{SB} \tag{17}$$

However, as there are always energy losses, the associated energy will be higher by an efficiency factor E_{eff}:

$$\Delta E = P(d)\Delta V_{SB}E_{eff} \tag{18}$$

The energy associated with the first process, on the other hand, will strongly depend on the method used to filter and compress the gasses. For example, when the compression is done with a continuous process without energy loss and the sea acts as a heat bath, ΔW can theoretically be as low as $0\mathrm{J/s}$. But when the compression is done in batches (and slow enough to assume isothermal compression), the required work to compress the molecules that are added in one time step ($\dot{n}\Delta t$) would be $\Delta W = (\dot{n}\Delta t RT)\ln\left(\frac{P(d)}{P_{blood}}\right)$. [26] In the rest of this thesis, the energy for this step is simplified by assuming that the required energy is proportional to the number of added molecules, and thus can be taken into account by raising E_{eff}.

2.2.5. Differential equation

In this section, a differential equation describing the volume change is derived. The change is equal to the added volume minus the lost volume plus the produced volume. In the derivation, temperature differences are neglected, since the temperature differences within the sea are small and the sea acts as a heat bath.

The 'produced' volume is the volume change due to the change in pressure. This is equal to the derivative of eq. (14) with respect to time, which can be simplified using the derivative of eq. (13):

$$\frac{\mathrm{d}V_{SB}(t)}{\mathrm{d}t} = \frac{\mathrm{d}\frac{nRT}{P(d(t))}}{\mathrm{d}t} = -\frac{nRT}{P(d(t))^2}\frac{\mathrm{d}P(d(t))}{\mathrm{d}t} = -\frac{V_{SB}(t)}{P(d(t))}\rho_w g\frac{\mathrm{d}d(t)}{\mathrm{d}t} \tag{19}$$

The added and removed volumetric flows can be divided into a passive and an active flow. The passive flow consists of the change due to leakage, see eq. (15).

The active change in volume due to secretion and absorption will be called $\frac{\mathrm{d}V_a}{\mathrm{d}t}$ and depends on how many gas molecules are inserted into the bladder. Since it is difficult to instinctively relate the number of molecules to a volume, it was chosen to introduce a volume change rate q in m^3/s with respect to the atmospheric pressure. This choice can be justified by the fact that the number of gas molecules that can be transported to and from the bladder depends mainly on the properties of the fish blood, [11] while the volume that corresponds to this amount of molecules does vary significantly with depth due to the pressure. This implies that the boundary values for q are constant.

According to the ideal gas law, the molecules that occupy a volume of $\int q\mathrm{d}t$ at P_0 occupy at $P(d)$ a volume equal to

$$V_a = \frac{P_0}{P(d)}\int q\mathrm{d}t \tag{20}$$

The derivative of this equation with respect to the time is equal to

$$\frac{\mathrm{d}V_a}{\mathrm{d}t} = \frac{P_0}{P(d)}q - \frac{P_0}{P(d(t))^2}\rho_w g\frac{\mathrm{d}d(t)}{\mathrm{d}t}\int q\mathrm{d}t \tag{21}$$

The total rate of volume change consists of the sum of equations (19), (21), and (15):

$$\frac{\mathrm{d}V_{SB}(t)}{\mathrm{d}t} = -\frac{V_{SB}(t)}{P(d(t))}\rho_w g\frac{\mathrm{d}d(t)}{\mathrm{d}t} + \frac{P_0}{P(d)}q - \frac{P_0}{P(d(t))^2}\rho_w g\frac{\mathrm{d}d(t)}{\mathrm{d}t}\int q\mathrm{d}t - G\Big(\frac{6V_{SB}(t)}{\pi}\Big)^{\frac{2}{3}}(P(d) - P_0) \tag{22}$$

The only part of this equation that can be influenced during the operation of the float is q.

2.3. A small float in an idealized two-dimensional linear flow field

The optimizing strategy that will be derived in the next chapter will be tested for a small float in an idealized two-dimensional linear flow field. The initial diameter of the considered float will be $D_0 = 10^{-4}\mathrm{m}$. When the buoyancy changes through volume changes, the diameter will change too. To protect the structural integrity of the float, the maximal expansion and compression will be limited. The seawater density and viscosity will be assumed to be constant.

In this section, first, the chosen linear flow field will be described. Then, it will be derived that in this field, $\mathrm{Re} < 1$ for a float with diameter $D_0 = 10^{-4}\mathrm{m}$ and a realistic maximal water velocity. Lastly, the equations of motion for the float in the chosen field will be summarized.

2.3.1. Flow field

In this first research, a two-dimensional linear flow field is assumed. The two dimensions are the vertical direction ($\hat{z}$) and one horizontal direction ($\hat{x}$). The velocity of the water in such a flow field can be written as

$$\vec{v}_w(x,z) = \vec{v}_w(0,0) + (\alpha_{xx}x + \alpha_{zx}z)\hat{x} + (\alpha_{xz}x + \alpha_{zz}z)\hat{z} \tag{23}$$

with all α_i constant and $\vec{v}_w(0,0)$ the velocity in an arbitrary origin.

In a two-dimensional slice of the sea, the seabed does not have to be flat and waves and tides can influence the height of the water column. For a general case, the vertical coordinate of the seabed can be described with a formula $B(x)$ and the height of the water column with $H(x,t)$. However, in the flow field that will be considered in the implementation, it will be assumed that there are no sandbanks or other bumps present in the slice of the sea. Additionally, the time-dependent waves and tides are disregarded, as those would usually correspond to a non-linear flow field. Hence, when the origin is set on the seabed, $B(x) = 0\mathrm{m}$ and $H(x,t) = H$, with H constant.

In the sea, the horizontal scale is much larger than the vertical one. Therefore, it is reasonable to assume that the horizontal velocity of the water is much larger than the vertical one. Additionally, when

the seabed is flat, there is no reason why the flow velocity would depend on the horizontal location. Therefore, α_{zx} in eq. (23) can be expected to be much higher than α_{xx}, α_{xz}, and α_{zz}. In the implementation, α_{xx}, α_{xz}, and α_{zz} will be neglected. Additionally, the velocity on the seabed was set to $\vec{0}\mathrm{m/s}$, as interactions with the seabed slow the flow down.

Hence the velocity $\vec{v}_w$ of the water of the considered flow field can be expressed as

$$\vec{v}_w(x,z) = \alpha z\, \hat{x} \tag{24}$$

α is equal to both $\frac{|\vec{v}_w(x,H)|}{H}$ and $\frac{dv_{wx}(z)}{dz}$. In the sea, it can be assumed that the maximal velocity is roughly $1\mathrm{m/s}$. Figure 3 depicts the considered flow field.

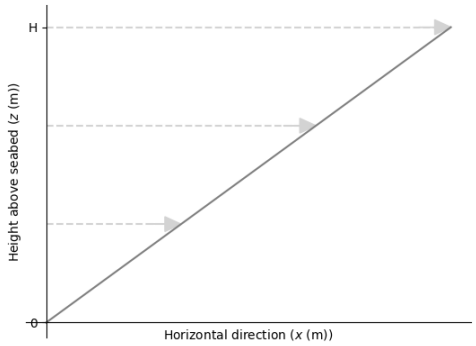


Figure 3: The considered flow profile. The flow velocity is $0\mathrm{m/s}$ in the vertical direction (z) and is linearly increasing with height in the horizontal direction (x). The velocity is $0\mathrm{m/s}$ at the seabed.

2.3.2. Stokes flow

The simplifications corresponding to Stokes flow can be used for a spherical float with diameter $D \approx 10^{-4}\mathrm{m}$ in the assumed flow field. This will be proven by assuming that eq. (12) can be used, and then deriving that the maximal Reynolds number is indeed smaller than 1, thus that eq. (12) could indeed be used. As this number depends on the maximal velocity difference $|\Delta\vec{v}|$, eq. (12) has to be solved. Throughout this whole derivation, it is assumed that the density of the float is constant.

A differential equation can only be solved when boundary conditions are known. It is assumed that the initial height of the float is z_0 and that the initial velocity matches the velocity of the water at this point: $\vec{v}_f(t=0) = \alpha z_0 \hat{x}$. For this initial velocity, the assumption of Stokes flow is correct, as $\Delta\vec{v} = \vec{0}\mathrm{m/s}$, thus the initial Reynolds number is 0. When $\rho_f \neq \rho_w$, the float will move upwards or downwards, and the velocity differences will increase.

Because the velocity vector $\Delta\vec{v}$ in eq. (12) is not squared, the vector and thus the whole equation can easily be decomposed in a horizontal and a vertical part. When $A = \frac{\rho_w - \rho_f}{\rho_f + \frac{\rho_w}{2}} g$ and $t_0 = \frac{(\rho_f + \frac{\rho_w}{2})D^2}{18\mu}$ are substituted, those parts are respectively

$$\frac{\mathrm{d}v_{fx}(t)}{\mathrm{d}t} = -\frac{v_{fx}(t) - v_{wx}(t)}{t_0} \tag{25}$$

$$\frac{\mathrm{d}v_{fz}(t)}{\mathrm{d}t} = A - \frac{v_{fz}(t) - v_{wz}(t)}{t_0} \tag{26}$$

The reason for naming the second variable t_0 will become clear soon. In the considered case, $t_0 \approx 10^{-3}\mathrm{s}$. This can be seen by inserting $D \approx 10^{-4}\mathrm{m}$, $\rho_w \approx 10^3\mathrm{kg/m^3}$, and $\mu \approx 10^{-3}\mathrm{Pa\,s}$, and assuming that the density of the float is similar to that of the surrounding water (i.e. $\rho_f \approx \rho_w$).

With the flow $v_{wz}(t) = 0\mathrm{m/s}$ and the boundary condition $v_{fz}(t=0) = 0$, the vertical part, eq. (26), can be solved as

$$v_{fz}(t) = At_0(1 - e^{-t/t_0}) \tag{27}$$

Note that $\Delta v_z(t) = v_{fz}(t)$ and that t_0 is a characteristic time, useful for estimating how long it takes before the float reaches its maximal velocity.

To solve the equation for the horizontal velocity, eq. (25), $v_{wx}(t)$ has to be known. This velocity depends on the height, which can be derived by integrating the vertical velocity, eq. (27):

$$\begin{aligned} z(t) &= z_0 + \int_0^t v_{fz}(t)\mathrm{d}t \\ &= z_0 + \int_0^t At_0(1 - e^{-t/t_0})\mathrm{d}t \\ &= z_0 + At_0 t + At_0^2(e^{-t/t_0} - 1) \end{aligned} \tag{28}$$

Hence the horizontal velocity of the water at the location of the float at time t is

$$v_{wx}(t) = \alpha(z_0 + At_0 t + At_0^2(e^{-t/t_0} - 1)) \tag{29}$$

The derivative of $\Delta v_x(t) = v_{fx}(t) - v_{wx}(t)$ can be simplified by substituting eq. (25) and the derivative of eq. (29):

$$\begin{aligned} \frac{\mathrm{d}\Delta v_x}{\mathrm{d}t} &= \frac{\mathrm{d}v_{fx}(t)}{\mathrm{d}t} - \frac{\mathrm{d}v_{wx}(t)}{\mathrm{d}t} \\ &= \left(- \frac{v_{fx}(t) - v_{wx}(t)}{t_0} \right) - \left(\alpha A t_0 - \alpha A t_0 e^{-t/t_0} \right) \\ &= -\frac{\Delta v_x}{t_0} - \alpha A t_0 (1 - e^{-t/t_0}) \end{aligned} \tag{30}$$

One way to solve this equation is to assume a solution of the form $\Delta v_x(t) = C(t)e^{-t/t_0}$. With boundary condition $\Delta v_x(t = 0) = 0\mathrm{m/s}$, the solution is

$$\Delta v_x(t) = \alpha A t_0 t e^{-t/t_0} - \alpha A t_0^2 (1 - e^{-t/t_0}) \tag{31}$$

For the next step, it will be convenient to know an upper bound of $|\Delta v_x(t)|$. Note that $\alpha A t_0 t e^{-t/t_0}$ and $\alpha A t_0^2 (1 - e^{-t/t_0})$ are both $0\mathrm{m/s}$ at $t = 0\mathrm{s}$ and have the same sign for higher t. Additionally, the absolute value of the derivative of the first term is always lower than that of the second one. Hence

$$0 \leq |\Delta v_x(t)| \leq |\alpha A t_0^2 (1 - e^{-t/t_0})| \tag{32}$$

Before determining the Reynolds number, the ratio $|\Delta v_x| : |\Delta v_z|$ will be determined.

$$\frac{|\Delta v_x(t)|}{|\Delta v_z(t)|} \leq \frac{|\alpha A t_0^2 (1 - e^{-t/t_0})|}{|A t_0 (1 - e^{-t/t_0})|} = \frac{\alpha t_0}{1} \ll 1 \tag{33}$$

The last inequality is only valid in this specific context, as $t_0 \approx 10^{-3}\mathrm{s}$ and $\alpha = \frac{|\vec{v}_w(x,H)|}{H}$, which will be lower than $1\mathrm{s}^{-1}$ in a flow field in a calm estuary. This implies that the x component does not contribute much to the total length of Δv, thus $\Delta v \approx \Delta v_z$. Hence, the Reynolds number is

$$\mathrm{Re} \approx \frac{\rho_w |\Delta v_z| D}{\mu} \leq |A t_0| \frac{\rho_w D}{\mu} = |\rho_w - \rho_f| \frac{\rho_w g D^3}{18\mu^2} \tag{34}$$

In the considered situation, this implies $\mathrm{Re} < 1$, as $D \approx 10^{-4}\mathrm{m}$, $\rho_w \approx 10^3 \mathrm{kg/m^3}$, $\mu \approx 10^{-3}\mathrm{Pa\,s}$, and ρ_f is within reasonable bounds, that is, not lighter than air, but also not more than twice the density of water. It is indeed reasonable to use the equations of motion in Stokes flow.

2.3.3. Simplification of the horizontal velocity

In eq. (33), it was derived that Δv_x is much smaller than Δv_z. Therefore, in this situation, it is reasonable to approximate $\Delta v_x = 0\mathrm{m/s}$, which corresponds to

$$v_{fx}(t) = v_{wx}(t) = \alpha z(t) \tag{35}$$

2.3.4. Summary idealized flow field

In the simplified setting of this subsection, the position and other changing properties of the float can be summarized in a four-dimensional state vector. The four states are all described in the previous section: horizontal and vertical position, vertical speed, and volume. The vector will be called $\vec{X}$.

$$\vec{X}(t) = \begin{pmatrix} x \\ z \\ v_{fz} \\ V \end{pmatrix} \tag{36}$$

The derivative of the different components of the state vector were derived previously in this chapter, in equations (35), (26), and (22). As the setting is described using the height above the seabed and not with the depth, $d = H - z$ and $\frac{\mathrm{d}d}{\mathrm{d}t} = -\frac{\mathrm{d}z}{\mathrm{d}t} = v_{fz}$ are substituted.

Additionally, it is assumed that the bladder takes up all of the volume, i.e. $V = V_{SB}$, and that the mass of the added and removed gas molecules can be neglected compared to the mass of the wall, such that the mass m can be considered constant. Such an allocation of the volume is not possible in real life, as both the mechanism controlling the volume change and measuring equipment would be outside of the bladder. However, it is reasonable to assume that when the optimization strategy can be used in the simplified case, it can also be used for a float that does not entirely consist of a swim bladder.

The derivative of $\vec{X}$ is

$$\vec{f}(\vec{X}, t) = \frac{\mathrm{d}\vec{X}}{\mathrm{d}t} = \begin{pmatrix} \alpha z \\ v_{fz} \\ \frac{\rho_w - \frac{m}{V}}{\frac{m}{V} + \frac{\rho_w}{2}} g - \frac{18\mu}{\left(\frac{m}{V} + \frac{\rho_w}{2}\right)\left(\frac{6V}{\pi}\right)^{2/3}} v_{fz} \\ \rho_w \frac{V}{P(H-z)} g v_{fz} + q(t)\frac{P_0}{P(H-z)} + \frac{P_0}{P(H-z)^2}\rho_w g v_{fz} \int q(t)\mathrm{d}t - G\left(\frac{6V(t)}{\pi}\right)^{\frac{2}{3}}(P(H-z) - P_0) \end{pmatrix} \tag{37}$$

3. Optimization strategy and implementation

In the first section of this chapter, an optimization strategy that is based on gradient descent and can be used to determine the required changes in the buoyancy of a float to navigate it from a location (x_0, z_0) to another location (x_g, z_g) is explained. Integration methods that can be used to determine the trajectory that a float follows are described in section 3.2, and different ways to quantify whether the buoyancy changes are optimal are derived in section 3.3.

The optimization strategy is implemented in Python for the small float in a two-dimensional linear flow field that was described in section 2.3. JAX, a package in Python that can automatically differentiate a wide range of functions, [4], is used to calculate the gradients. See appendix A for more information about automatic differentiation.

Finally, in section 3.4, a test that can be used to determine whether this strategy can indeed be used to find an optimal sequence of buoyancy changes is presented.

3.1. The modified gradient descent algorithm

3.1.1. Introduction to gradient descent

The buoyancy of the float is changed through volume changes, as described in eq. (22). When the optimization strategy is implemented, the situation will be discretized in time. Let Q be the array consisting of the volume rates of change q at every time step of the discretization and define a cost function that is a measure of how close the float approaches the intended target location when the volume is changed as described with Q. When Q is random, the probability that the float ends up in the desired location is very small, and the cost function is high. Thus, a random Q is not optimal. However, when it is known that increasing or decreasing one or more of the elements of this Q in a certain way decreases the value of the cost function, this change can be used to construct a slightly better Q.

The derivative of a function with respect to an input value describes whether the function increases or decreases when the input is increased. Therefore, derivatives can be used to determine in which way a single element of Q should be altered to decrease the cost function.

Gradient descent algorithms[23] are based on this idea: for some input Q, the gradient of the cost function $cost(Q)$ with respect to the inputs is derived, and Q is updated in the opposite direction of the gradient:

$$Q_{new} = Q_{old} - \eta \cdot \nabla_Q cost(Q_{old}) \tag{38}$$

η is called the learning rate and determines how much the input Q is changed per iteration.

3.1.2. Modification

The standard gradient descent method is modified to account for boundary conditions and to include changes in the learning rate. The flow chart in figure 4 shows the algorithm, which will also be described below.

Let $cost(Q)$ be the considered cost function, Q_0 the initial guess, q_{max} and q_{min} the boundary conditions for the elements of Q, J the number of iterations, and δ the first step size. Before the first iteration is started, the cost corresponding to the input is calculated. Then, the following steps are repeated for all J iterations.

- With reverse automatic differentiation, the gradient $\nabla_Q cost(Q)$ of the cost function with respect to the current Q is determined.
- In the next step, the gradient is going to be scaled such that the highest absolute value in the gradient corresponds to δ or $-\delta$. However, some values of Q cannot be changed much, as there is a physical limit to how much volume can be added or subtracted at one time. Therefore, the components of $\nabla_Q cost(Q)$ that potentially can alter the corresponding q to a value higher than q_{max} or lower than q_{min} are temporarily neglected. The remaining values of the gradient are called $(\nabla_Q cost(Q))^*$. When this set is empty, no alterations are possible. In that case, δ is halved and a new iteration is started.
- The learning rate η depends on δ and the highest absolute value of $(\nabla_Q cost(Q))^*$, in such a way that the q that is altered the most still satisfies the boundary conditions.

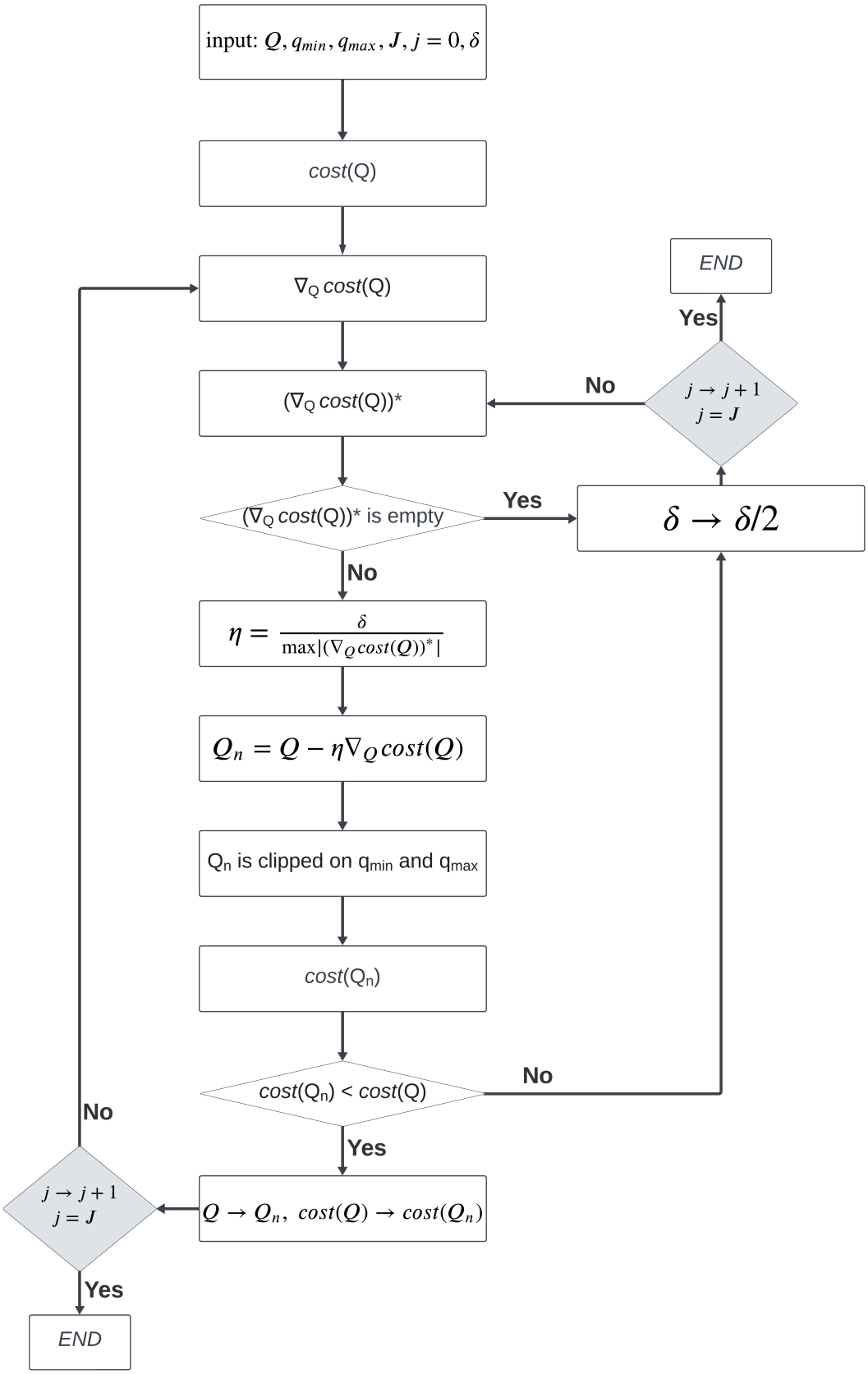


Figure 4: Flow chart describing the modified gradient descent algorithm.

- A new Q_n is derived with eq. (38) and clipped such that all q are between the maximal and minimal values.
- The cost of Q_n is determined, and when the $cost(Q_n)$ is lower than the original $cost(Q)$, Q_n will be accepted as Q. If the cost is higher, Q_n will be rejected and δ will be halved. In both cases, a new iteration starts.

The last value Q is called the optimized Q for the evaluated cost function and is denoted Q_f.

3.2. Trajectory calculation

To determine the trajectory for a certain input Q, the differential equations that describe the state of the float of interest have to be integrated. Two different finite difference integration methods, Forward Euler (FE) and Runge-Kutta (RK4), are used in the implementation and testing process. The state of the float will be denoted as $\vec{X}(t)$. Both methods approximate the state $\vec{X}(t+\Delta t)$ based on $\vec{X}(t)$ and the derivative of the state:

$$\vec{f}(\vec{X},t) = \frac{\mathrm{d}\vec{X}}{\mathrm{d}t} \tag{39}$$

When using FE, the state Δt seconds after time t is calculated as follows:

$$\vec{X}(t+\Delta t) = \vec{X}(t) + \Delta t \cdot \vec{f}(\vec{X}(t),t) \tag{40}$$

When using RK4, the state Δt seconds after time t is determined with four intermediate predictors: $\vec{k}_1$, $\vec{k}_2$, $\vec{k}_3$, and $\vec{k}_4$:

$$\begin{aligned}
\vec{k}_1 &= \Delta t \cdot \vec{f}(\vec{X}(t),t) \\
\vec{k}_2 &= \Delta t \cdot \vec{f}(\vec{X}(t) + \frac{\vec{k}_1}{2}, t + \frac{\Delta t}{2}) \\
\vec{k}_3 &= \Delta t \cdot \vec{f}(\vec{X}(t) + \frac{\vec{k}_2}{2}, t + \frac{\Delta t}{2}) \\
\vec{k}_4 &= \Delta t \cdot \vec{f}(\vec{X}(t) + \vec{k}_3, t + \Delta t)
\end{aligned} \tag{41}$$

The new state is

$$\vec{X}(t+\Delta t) = \vec{X}(t) + \frac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \tag{42}$$

3.2.1. Time step

When a finite integration method is used, a time step Δt has to be chosen for the discretization. The accuracy of a method depends on the time step and is usually better when Δt is smaller. [27] FE requires less computing power than RK4, but the accuracy is lower when using the same time steps. [18]If the time step is too big, the solution might be unstable, which means that the errors introduced due to the discretization increase with every step. Stability conditions for the time step depend on the eigenvalues of the Jacobian of eq. (39). [27]

The dynamics of the considered system are not stable, thus it is not possible to determine an upper bound for Δt using the eigenvalues of the Jacobian of eq. (37). This instability stems from the facts that both x and z are unlimited and that the volume of the float is unstable: when there is no active control over the volume and the float rises a bit, the pressure decreases, causing the volume to increase and the float to rise even further. When the float sinks a bit, the opposite process occurs. The volume will not increase or decrease limitless, due to the active addition and removal of volume together with boundaries for the volume and height (the float cannot sink below the seabed and cannot rise out of the sea).

Even though the dynamics are unstable, a time step has to be chosen. It was decided to choose the time step based on the model that corresponds to the situation when the volume is set to be constant. Then only the vertical velocity is of interest:

$$\frac{\mathrm{d}v_{fz}}{\mathrm{d}t} = \frac{\rho_w - \rho_f}{\rho_f + \frac{\rho_w}{2}} g - \frac{18\mu}{(\rho_f + \frac{\rho_w}{2})D^2} v_{fz} \tag{43}$$

The eigenvalue associated with this equation is

$$\lambda = \frac{\partial \frac{\mathrm{d}v_{fz}}{\mathrm{d}t}}{\partial v_{fz}} = -\frac{18\mu}{(\rho_f + \frac{\rho_w}{2})D^2} = -\frac{1}{t_0} \tag{44}$$

The minimal value for λ corresponds to the situation where D is maximal. In the considered setting, the order of magnitude of λ is then 3. The stability condition for the time step is for FE

$$\Delta t \leq -\frac{2}{\lambda} = 2t_0 \tag{45}$$

and for RK4 [27]

$$\Delta t \leq -\frac{2.8}{\lambda} = 2.8t_0 \tag{46}$$

To be on the safe side, $\Delta t = 0.5\ 10^{-3}\mathrm{s}$ is used in the implementation.

3.3. Cost functions

As a gradient can only be calculated for a known cost function, a cost function will have to be defined. The cost function depends on the purpose of the float. In this research, the navigation of a float to a desired location is the primary objective. However, it is usually preferred that this location is reached in a specific way: fast, energy-efficient, or a combination of those two. In order to determine whether the optimization strategy can be used for any float, the optimization strategy will be tested for all those different objectives. The objectives will be numbered as follows:

1. The trajectory corresponding to the optimized Q leads to the desired destination.
2. The trajectory corresponding to the optimized Q leads to the desired destination and requires the least amount of time to do so.
3. The trajectory corresponding to the optimized Q leads to the desired destination and requires the least amount of energy to do so.
4. The trajectory corresponding to the optimized Q leads to the desired destination and combines efficiency in time and energy in some way.

For each objective, a cost function that relates the input Q to how well the input and trajectory meet the objective has to be defined. This is done in two steps. First, the costs for the three individual parts of the objectives are defined. The individual parts are the position, time, and energy and the corresponding cost functions will be called partial cost functions. Then, weighting factors are introduced and calculated. Those are necessary when more than one partial cost function is examined at the same time, which is the case for objectives 2, 3, and 4. If one partial cost function is given too much weight compared to another cost function, the results might be unsatisfactory. See figure 5 for an example.

3.3.1. Partial cost functions

When given an input Q, the trajectory can be derived with one of the methods that are described in section 3.2. Let $x(t)$ and $z(t)$ be the coordinates of the trajectory. While the trajectory is only defined every Δt seconds, the intermediate points on the trajectory can be approximated using interpolation.

The positional cost has to be large when the float never comes close to destination (x_g, z_g). Therefore, it makes sense to let the cost depend on the minimal distance between the trajectory and the destination.

The cost function will be defined as

$$cost_p(Q) = \min_t \left((x(t) - x_g)^2 + (z(t) - z_g)^2\right) \tag{47}$$

which is measured in m^2.

When the horizontal velocity of the water is much larger than the vertical velocity, as is the case in the studied flow field as described with eq. (23), the horizontal velocity of the float is much larger than the vertical velocity and it can be assumed that the position closest to the destination has horizontal

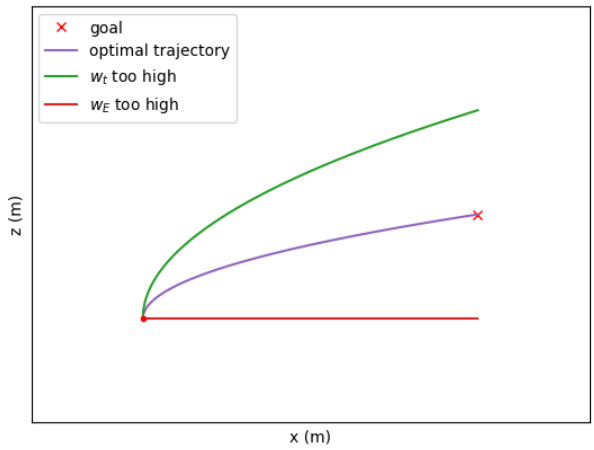


Figure 5: An example of unsuitable weighting factors. If the weighting factor corresponding to the time (w_t) is too high, the found solution might be to rise as far as possible and ignore the target position. If the weighting factor corresponding to the energy (w_E) is too high, the found solution might be to not change the volume at all, and thus stay at the same height.

coordinate x_g. Let z_{xg} be the vertical coordinate that corresponds to x_g. Then the positional cost can also be approximated with

$$cost_p(Q) = (z_{xg} - z_g)^2 \tag{48}$$

The time it takes to reach the location closest to the destination will be called t_{xg} and can be used as the cost for the time:

$$cost_t(Q) = t_{xg} = \underset{t}{\operatorname{argmin}} \left((x(t) - x_g)^2 + (z(t) - z_g)^2\right) \tag{49}$$

this is measured in s. Alternatively, the time corresponding to x_g can be used.

The energy consumption depends on all the volume rates of change q before the closest point is reached. As it was assumed that only the addition of volume costs energy, we will only consider positive values of q:

$$g(q) = \begin{cases} q & \text{if } q \geq 0\text{m}^3/\text{s} \\ 0 & \text{if } q < 0\text{m}^3/\text{s} \end{cases} \tag{50}$$

With the ideal gas law, the positive volume change on depth d can be calculated from the volume change relative to atmospheric pressure:

$$\Delta V_+ = (g(q) \cdot \Delta t)) \frac{P_0}{P(d)} \tag{51}$$

This volume can be converted into energy with eq. (18), and has to be added for all time steps. Let Q' denote the set of volume rates of change q for all time steps before t_{xg} is reached. Then

$$cost_E(Q) = \sum_{q \in Q'} (g(q(t)) \cdot \Delta t) P_0 E_{eff} \tag{52}$$

measured in J.

3.3.2. weighting factors

When the objective is to optimize Q for a combination of two or more of the three partial cost functions, the different cost functions will have to be added:

$$cost_i = w_p cost_p + w_t cost_t + w_E cost_E \tag{53}$$

In this equation, the w are weighting factors that make the different partial cost functions dimensionless and balance the relative importance of the objectives.

The cost functions corresponding to the four different objectives will be derived below, based on the setting as described in section 2.3 and assumptions for acceptable and unacceptable costs for the four objectives.

$cost_1$, Position

For the first objective, only the final position of the float is important. Thus w_t and w_E are set to zero. w_p has to be chosen in such a way that $cost_1$ is dimensionless. Although the actual scaling does not influence the results when only one partial cost function is taken into account, it is useful to choose a w_p that can also be used for the other objectives. This is done by dividing the distance to the target location by a length that will be a measure of what deviation is considered acceptable. This deviation is set to the initial float diameter D_0.

$$w_p = \frac{1}{D_0^2} \tag{54}$$

Hence, the first cost function is:

$$cost_1(Q) = \left(\frac{z_{xg} - z_g}{D_0}\right)^2 \tag{55}$$

$cost_2$, Position and Time

For the second objective, time and position are combined, thus only w_E is 0. Since a trajectory-dependent term is added to $w_p cost_p$, the minimum of $cost_2$ is not at $z_{xg} = z_g$. The idea for w_p and w_t is to use the previously derived w_p, and a w_t that is as big as possible, under the condition that $|z_{xg} - z_g| \leq D_0$ for the z_{xg} where $cost_2$ is minimal. The fastest time is obtained when the float gains height quickly, as v_{fx} is larger for a bigger z in the assumed flow field. Because the various paths that lead to the same destination are almost all too complicated to calculate exactly, only one easy trajectory is considered: a horizontal trajectory, thus $z(t) = z_{xg}$ at every time step. It was derived that the horizontal velocity can be approximated with $v_{fx} = \alpha z_{xg}$ (eq. (35)). Therefore, the corresponding time is

$$t_{xg} = \int_0^{x_g} \frac{1}{v_{fx}(z)} \mathrm{d}x = \frac{x_g}{\alpha z_{xg}} \tag{56}$$

When z_{xg} is $z_g + D_0$, the trajectory is considered acceptable, but a slightly larger z_{xg} is not acceptable, even though $cost_t$ will be lower. This implies that the derivative of $cost_2$ with respect to z_{xg} has to be positive in $z_{xg} = z_g + D_0$.

$$cost_2(Q) = \left(\frac{z_{xg} - z_g}{D_0}\right)^2 + w_t \frac{x_g}{\alpha z_{xg}} \tag{57}$$

$$\frac{\mathrm{d}cost_2(Q)}{\mathrm{d}z_{xg}} = \frac{2(z_{xg} - z_g)}{D_0^2} - w_t \frac{x_g}{\alpha z_{xg}^2} \tag{58}$$

When the derivative is set to 0 in $z_{xg} = z_g + D_0$,

$$w_t \leq \frac{2\alpha}{D_0} \frac{(z_g + D_0)^2}{x_g} \tag{59}$$

Thus the cost function for the second objective is

$$cost_2(Q) = \left(\frac{z_{xg} - z_g}{D_0}\right)^2 + \frac{2\alpha}{D_0} \frac{(z_g + D_0)^2}{x_g} t_{xg} \tag{60}$$

$cost_3$, Position and Energy

For the third objective, time and energy are combined, thus only w_t is 0. The idea for w_p and w_E is to use the previously derived w_p, and a w_E that is as big as possible, under the condition that $|z_{xg} - z_g| \leq D_0$ for the z_{xg} where $cost_2$ is minimal. To rise higher requires a lower buoyancy and thus more energy. Therefore, w_E should be chosen in such a way that using enough energy to rise to a height $z_g - D_0$ is preferable to using slightly less energy and rising to a higher z_{xg}. Because changes in the buoyancy keep

affecting all future time steps, density changes directly after the float leaves the initial position are more energy efficient than changes close to the target location. Hence, the optimal Q will be maximal for the first moments, and $0\mathrm{m}^3/\mathrm{s}$ the rest of the time.

The true optimal trajectory cannot be used to derive w_E, as the differential equations cannot be solved exactly and w_E is necessary to optimize numerically. Therefore, some simplifications were made to approximate the energy required for the optimal Q. Firstly, the volume changes due to pressure and leakage are neglected, and secondly, it is assumed the total density change is achieved instantly and does not change later.

In this case, the horizontal component of eq. (12) can be written as

$$\frac{\mathrm{d}v_{fz}}{\mathrm{d}t} = A - \frac{v_{fz}}{t_0} \tag{61}$$

with $A = \frac{\rho_w - \rho_f}{\rho_f + \frac{\rho_w}{2}} g$ and $t_0 = \frac{18\mu}{(\rho_f + \frac{\rho_w}{2})D^2}$ constant. The solution to this equation is

$$v_{fz}(t) = At_0 + (v_{fz}(0) - At_0)e^{-t/t_0} \tag{62}$$

As previously derived, $t_0 \approx 10^{-3}\mathrm{s}$ in the assumed setting. As any implementation will be used for situations on a much larger timescale, this velocity can be approximated with

$$v_{fz}(t) \approx At_0 \tag{63}$$

Hence the height can be written as

$$z(t) = \int_0^t v_{fz}(t)\mathrm{d}t \approx z_0 + At_0 t \tag{64}$$

z_0 is the initial height. Therefore, the time required to rise to a height z_{xg} is

$$t_{xg} = \frac{1}{At_0}(z_{xg} - z_0) \tag{65}$$

Since the horizontal speed can be expressed with eq. (35), the horizontal traversed distance in this time is

$$\Delta x = \int_0^{t_{xg}} \alpha z \mathrm{d}t = \int_0^{t_{xg}} \alpha(z_0 + At_0 t)\mathrm{d}t = \alpha(z_0 t_{xg} + At_0 \frac{t_{xg}^2}{2}) \tag{66}$$

By substituting $\Delta x = x_g$ and eq. (65), the value for At_0 can be determined:

$$At_0 = \frac{\alpha(z_{xg}^2 - z_0^2)}{2x_g} \tag{67}$$

A and t_0 both depend solely on the volume (by means of the diameter and the density), the corresponding required volume change ΔV_r can be determined. The energy required to induce this volume change can be derived with eq. (18) and will be called $cost_{E,min}(z_{xg})$. Just like when w_t was derived, w_E can be determined by considering the derivative of $cost_3$ with respect to z_{xg} at the accepted boundary.

$$\frac{\mathrm{d}cost_3(Q)}{\mathrm{d}z_{xg}} = \frac{2(z_{xg} - z_g)}{D_0^2} + w_E \frac{\mathrm{d}cost_{E,min}(z_{xg})}{\mathrm{d}z_{xg}} \tag{68}$$

In this case, the derivative has to be negative at $z_{xg} = z_g - D_0$, as a lower z_{xg} is undesirable. This implies

$$w_E \leq \frac{2}{D_0 \frac{\mathrm{d}cost_{E,min}(z_{xg}=z_g-D_0)}{\mathrm{d}z_{xg}}} \tag{69}$$

The value of $\frac{\mathrm{d}cost_{E,min}(z_{xg}=z_g-D_0)}{\mathrm{d}z_{xg}}$ can be numerically derived, for example in Python. The total cost function is

$$cost_3(Q) = \left(\frac{z_{xg} - z_g}{D_0}\right)^2 + \frac{2}{D_0 \frac{\mathrm{d}cost_{E,min}(z_{xg}=z_g-D_0)}{\mathrm{d}z_{xg}}} \left(\sum_{q \in Q'} (g(q(t)) \cdot \Delta t) P_0 E_{eff} \right) \tag{70}$$

$cost_4$, Position, Time and Energy

For the last objective, all properties are considered simultaneously. This can be interesting when the goal is to find Q such that the corresponding trajectory does not cost a lot of energy but also does not take forever. When there is a specific reason to consider all functions at the same time, the weighting factors will have to be adapted to the circumstances. For example, when the device has an additional constant fuel cost, for example due to measurement taking or leakage compensation, then the energy used for those processes will be proportional to the time, and it might be useful to scale the functions accordingly.

In this thesis, there is no specific additional reason to consider a combination of both time and energy, aside from determining whether the implemented program can handle such a cost function. Therefore, the idea is to make $cost_t$ and $cost_E$ roughly equally important, by using w_t as derived in eq. (54) and half of the previously derived w_t and w_E.

$$cost_4 = w_p cost_p + \frac{1}{2} w_t cost_t + \frac{1}{2} w_E cost_E \tag{71}$$

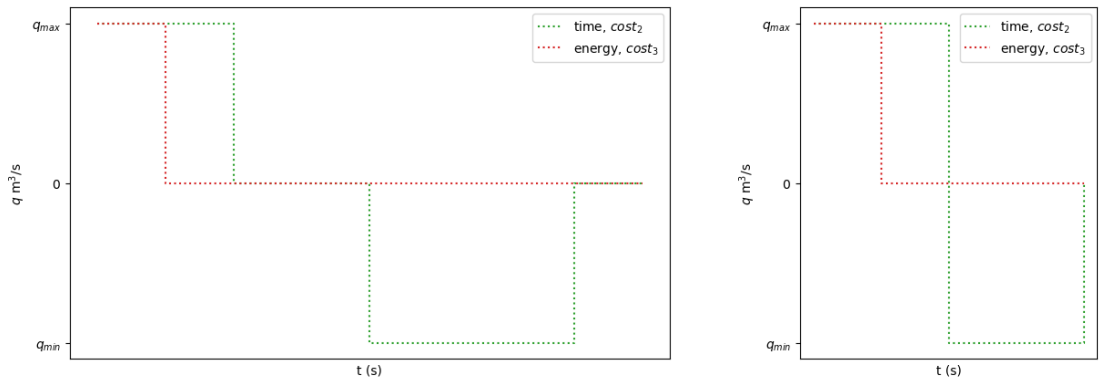
3.3.3. Expectations for the different cost functions

The expectation is that, given enough iterations, optimization for the position ($cost_1$) will always steer the float to the target location. However, the specific Q_f will depend on the initial guess.

Q_f for the time ($cost_2$) is expected to resemble a step function, first, the elements of Q_f are expected to be maximal for as long as possible, in order to maximally utilize the faster flow on higher z, and then, the volume changes are expected to be maximal negative to make the float sink back to the desired height.

The expectation for Q_f for the energy ($cost_3$) is that it will only be positive that it is strictly necessary to reach the target location. When some q are positive, it is expected that those are only the first values, as a change in volume directly after the float is let go will have the most impact on the final height. The expectations for $cost_2$ and $cost_3$ are illustrated in figure 6.

Q_f for the combination of all properties ($cost_4$) should be in between the values found for $cost_2$ and $cost_3$. If Q_f for $cost_4$ is very close to that of $cost_2$ or $cost_3$, the time or the energy might be favored disproportionally.



(a) (b)

Figure 6: The expected volume rates of change for a target location higher than the initial location. The left figure is for a target far away. When optimizing for the time, the bladder will first expand until the maximum value, and then be constant until it is decreased to the minimum value, and stay minimum until the target is reached. The figure on the right is for a target close by. In this case, the bladder is not completely filled in the time-optimization before the float has to sink again. In both cases, when optimizing for the energy, only as much volume as strictly necessary is added.

3.4. Testing the strategy

The described strategy for optimizing Q is implemented in Python and will be tested for a small spherical float of initial diameter $D_0 = 10^{-4}\mathrm{m}$ in the simplified two-dimensional linear water flow of eq. (24) for

a few different starting and target locations and for the four different cost functions. The determined optimal Q_f will be compared to the intuitive expected Q for the different cost functions, in order to determine whether the optimization strategy can be used for any float, regardless of the objective.

Only the starting and target locations will be varied between tests, thus exact values have to be chosen for the properties of the float and flow field and for other variables in the algorithm. Those are described in the next section. Additionally, values for the initial guess Q_0 and the number of iterations J will have to be chosen, those values will be determined with the procedures described in sections 3.4.2 and 3.4.3.

3.4.1. Specifications for implementation

Table 1 contains the additional variables that describe the situation, their assumed value, and a justification for this choice. The properties that were specified for the float and the flow field in section 2.3 and direct derivations of those, such as $V_0 = \frac{\pi}{6}D^3 \approx 5\ 10^{-13}\mathrm{m}^3$ and $\rho_w = 1.026\ 10^3\mathrm{kg/m}^3$, are not included.

The other variables that have to be specified in the implementation are the size of the time step Δt, which is chosen to be $0.5\ 10^{-3}\mathrm{s}$ (see section 3.2), and the size of the first step size in the gradient descent algorithm δ, which is chosen to be $\frac{q_{max}}{5}$ (this allows for relatively big changes in the first few iterations).

To decrease the number of computations per iteration, it was chosen to integrate with Forward Euler during the optimization. Runge-Kutta was used to determine the trajectory corresponding to Q_f, as this method is more accurate, and thus can be used to partially check whether the introduced error due to FE causes the trajectory to converge to an incorrect location.

Table 1: Assumptions for the variables in the implemented model.

Variable		Value	Justification
α	Proportionality constant between z and v_{wx}	$\frac{1\mathrm{m/s}}{H} = 0.1\mathrm{s}^{-1}$	The maximal velocity is estimated to be ca. $1\mathrm{m/s}$.
E_{eff}	Energy efficiency factor	arbitrary, 1	This value drops out of the equation when the derived w_E is used.
H	Height of the water column	$10\mathrm{m}$	The considered situation is close to the shore, thus a relatively low height is chosen.
q_{max},q_{min}	Maximal and minimal rate of change of the volume	$\pm V_0 \frac{10\%}{1\mathrm{s}} \approx \pm 5\ 10^{-14}\mathrm{m}^3/\mathrm{s}$	The bladder cannot be maximally inflated or deflated instantly and when the value is too low, the density changes take very long, which complicates the testing process. 10% change per second is chosen.
ρ_{f0}	Initial density of the float	$\rho_w = 1.026\ 10^3\mathrm{kg/m}^3$	The float can only switch between rising and sinking when the density passes ρ_w, thus only initial densities close to this value are interesting.
v_{fz0}	Initial vertical velocity	$\mathrm{m/s}$	Another initial value would only cause an offset.
$\frac{\mathrm{d}V_L}{\mathrm{d}t}$	Leakage	$0\mathrm{m}^3/\mathrm{s}$	Both G and S are very small, and the product is much smaller than other terms in eq. (22).
V_{max},V_{min}	Maximal and minimal volume	$2V_0 \approx 10^{-12}\mathrm{m}^3$, $\frac{V_0}{2} \approx 2.6\ 10^{-14}\mathrm{m}^3$	When the volume increases or decreases too far, the bladder can be damaged. Those values are approximately the volumes corresponding to 0.5 and 1.4 times the initial pressure.

3.4.2. The initial guess Q_0

Ideally, the optimized rates of change of the volume Q_f do not depend on Q_0. However, in reality, this is not always the case. This can be because the costs corresponding to two different Q can be equal. But the optimized Q_f corresponding to two different initial Q_0 are also different because of the finite number of iterations: one initial Q_0 might cause the corresponding Q_f to converge faster to the true optimal Q than another Q_0 does. Therefore, it is essential to choose a suitable Q_0.

There are infinitely many possibilities, but four different initial guesses will be compared to make a measured decision on which Q_0 is suitable to use in the testing process.

The four Q_0 that will be compared are:

1. $Q_0 = Q_{max}$, where Q_{max} is defined as the array that consists of only elements q_{max}, which is to say that the initial volume change rate is maximal at all times.

2. $Q_0 = \mathbf{0}$. $\mathbf{0}$ is an array with all elements $0\text{m}^3/\text{s}$. This means that the initial volume does not change.

3. Q_0 is a discontinuous sequence that is q_{max} for the first half of the estimated required time, and $0\text{m}^3/\text{s}$ for the other half of the time. This Q_0 is chosen to represent all discontinuous inputs.

4. Q_0 has a sinusoidal shape with two periods, the amplitude is q_{max}. This Q_0 is chosen to represent all continuous, though non-constant, inputs.

The comparison will be done as follows: the optimized Q_f after J iterations will be determined for all four Q_0 with $cost_2$ (finding the fastest trajectory), an initial location $(x_0, z_0) = (0.0\text{m}, 5.0\text{m})$, and a final location $(x_g, z_g) = (1.0\text{m}, 5.0004\text{m})$. Those locations were chosen, because the intuition in this case is relatively easy. The optimized Q_f will be plotted in the same figure, such that it can be determined which initial guesses converge faster to the true optimal Q than others. The number of iterations for this comparison will be chosen in such a way that all Q_f start to resemble the true optimal Q, but are still quite distinct.

3.4.3. The number of iterations J

Usually, more iterations result in a Q_f that is closer to the true optimal Q. However, the available time and computing power are limited, thus the number of iterations will have to be a trade-off between enough iterations such that Q approximates the optimal solution, but few enough iterations that the computational time is limited.

The number of iterations to use when the strategy is tested will be determined in the same context as the initial guess, that is: time optimization from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$. The initial Q_0 will be the guess that was deemed the most suitable in the previous section. The Q_f corresponding to different numbers of iterations ranging from 0 up to 100 will be determined and plotted together in the same figure. The J that will be used in the following tests will be chosen based on those results.

3.4.4. Testing method

The goal is to determine whether the optimization strategy could potentially be used for broad purposes, thus for different starting and target locations and for different cost functions. To do so, optimized Q_f are derived for the different cost functions from section 3.3 and different starting and target locations with the choices for Q_0 and J as determined in the previous procedures.

If the Q_f and corresponding trajectories for a cost function meet the expectations for the true optimal Q and indeed steer the float to the target location for all tested starting and target locations, it is reasonable to assume that the optimization strategy can be used in almost all similar situations. If the Q_f does not match the expectations, the optimization strategy most probably does not work. If a Q_f matches the expectations, but the float is not steered towards the target location, it has to be determined whether this is due to the optimization strategy or due to the choice of the cost function.

Five different starting and target locations are considered:

- $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$
- $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0\text{m})$
- $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 4.9996\text{m})$
- $(0.0\text{m}, 0.0\text{m})$ to $(1.0\text{m}, 0.0\text{m})$
- $(0.0\text{m}, 5.0\text{m})$ to $(100.0\text{m}, 5.5\text{m})$

In the first three cases, the float starts in the middle of the water column and is instructed to rise, return to the same height, and sink. Those cases are considered to determine if the strategy works for standard manoeuvres when different costs are considered.

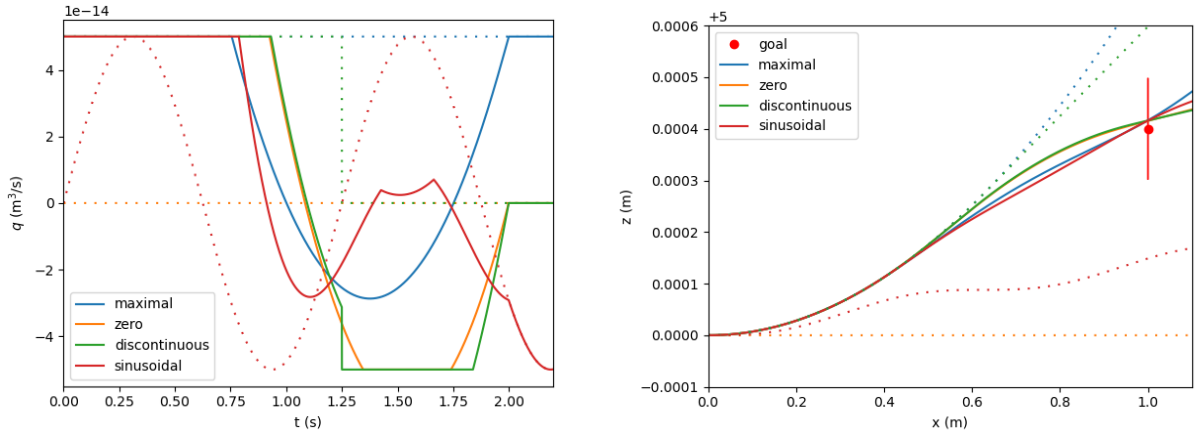
The other two situations are to determine whether the strategy can also be used to optimize in more extreme situations, that is, when the horizontal velocity is very small (since z_g is close to the seabed) or when x_g is very large. In the fourth case, $z_0 = 0$, thus the float would not rise from the seabed if $Q_0 = \mathbf{0}$, therefore, in this situation, $Q_0 = Q_{max}$ will be used, regardless of the Q_0 derived using the process in section 3.4.2.

4. Results

The optimization strategy as described in section 3.1 is tested in Python as described in section 3.4. In this chapter, the individual results are plotted and discussed. The implications and further discussion are presented in the next chapter.

4.1. Initial guess Q_0

Different initial series of volume rates of change Q_0 were tested for $cost_2$ with the method described in section 3.4.2. It was decided that 50 iterations are sufficient to determine for which initial guess Q_0 the corresponding Q_f is closest to the true optimal Q. The initial Q_0 and the resulting trajectories are depicted in figure 7. Although all trajectories approach the target location, all Q_f still have some faults compared to the expectation: the Q_f corresponding to the sinusoidal input is furthest from the expected step function; the Q_f for the maximal input returns to the maximal value after x_f is reached; the Q_f corresponding to the discontinuous input consists of two distinct parts; and the Q_f corresponding to the zero input is more like a parabola than a step function. The optimized Q for $Q_0 = \mathbf{0}$ was considered the most acceptable and was used in the testing process.



(a) (b)

Figure 7: Time optimization for different initial guesses Q_0 for a float that is to be steered from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$. 7a shows the optimized Q_f together with the initial controls. The initial guesses are plotted with dotted lines and the optimized Q_f with solid lines. 7b shows the corresponding trajectories. The accepted deviation from the target location in 7b is equal to the initial diameter of the float.
Note that all vertical axis scales are very small because the float is very small, and $q_{max} \approx 5\ 10^{-14}\text{m}^3/\text{s}$.

4.2. Iterations

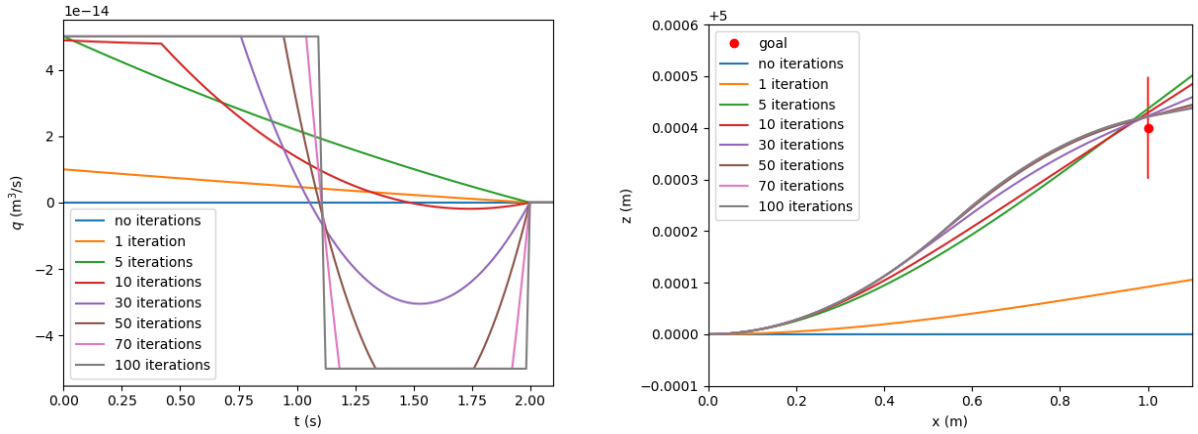
Different numbers of iterations J were tested for $cost_2$ with the method described in section 3.4.3. The results are depicted in figure 8. It is clear that using more iterations results in a Q_f that better resembles the expected step function. A number of 70 iterations was considered a reasonable trade-off between approaching the eventual optimal Q and the use of computing power, and was in the rest of the testing process.

4.3. Testing

The implemented optimization strategy is tested for five different initial and target locations as described in section 3.4.4.

4.3.1. Standard manoeuvres

The first three situations require the float to respectively rise, return to the same height, and sink, as the float is set to move from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, z_g)$, where the z_g are 5.0004m, 5.0m and 4.9996m. The optimized Q_f and corresponding trajectories for the four different cost functions are shown in figure 9.



(a) (b)

Figure 8: Time optimization for different numbers of iterations for a float which is to be steered from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$, with initial guess $Q_0 = \mathbf{0}$. 8a shows the optimized Q_f and 8b shows the corresponding trajectories. The accepted deviation from the target location in 8b is the initial diameter of the float.
Note that all vertical axis scales are very small because the float is very small, and $q_{max} \approx 5\ 10^{-14}\text{m}^3/\text{s}$.

Most Q_f and corresponding trajectories match the expectations that were expressed in section 3.3.3 for all cost functions. Only the trajectory corresponding to the optimized Q_f for the energy ($cost_3$) when the target is $z_g = 5.0004\text{m}$ does not pass within a radius of D_0 of the target position. As the Q_f matches the expectations for $cost_3$, it can be concluded that the strategy does function in this situation, but that the weighting factors should be altered in order to obtain a Q_f that does steer the float to the target location. Another option is to increase the number of iterations, but this will only result in a trajectory that comes closer to 5.0004m if the minimal possible value for $cost_3$ corresponds to a z_{xg} such that $|z_{xg} - 5.0004\text{m}| \leq D_0$.

Some of the optimal Q_f and trajectories overlap; those are the positional, energetic, and combined optimizations in figures 9c and 9d, and the positional and energetic optimal paths in figures 9e and 9f. The equality of the energetic and positional costs in both cases makes sense, as only the expansion of the bladder requires energy and no expansion is required to reach the target location. The equality of the optimized trajectory for the combined, energy and position optimizations when the target is at the same height as the initial position implies that the energetic cost is much more important than a shorter time in the considered case.

4.3.2. From the seabed to the seabed

The fourth situation is a float for which the initial and target locations are on the seabed: from $(0.0\text{m}, 0.0\text{m})$ to $(1.0\text{m}, 0.0\text{m})$. The optimized Q_f and corresponding trajectories for the four different cost functions are shown in figure 10.

The trajectories that correspond to the Q_f are nearly the same for all cost functions, and while they steer the float closer to the target location than the initial guess did, none of the trajectories come within the intended D_0 radius of the target, and none of the Q_f for $cost_2$, $cost_3$ and $cost_4$ match the expectations. Those observations are interlinked, as the positional cost ($cost_p$) is included in all four cost functions, and the weights w_t and w_E are chosen in such a way that $cost_t$ and $cost_E$ only become important when the trajectory comes within a radius D_0 around the target location, which is not yet the case after 70 iterations. Therefore, all Q_f resemble the optimized Q for $cost_p$, which explains the shape of Q: The bladder is full after about 20s, as the maximum volume of the bladder was set to $2 \cdot V_0$, the maximum increase at atmospheric pressure to $V_0 \cdot \frac{10\%}{1\text{s}}$, and the pressure at the seabed is almost twice the atmospheric pressure. After the bladder is filled, the volume change rate has to be non-positive. The float has risen due to the volume changes, but the target location is on the seabed. Therefore, the volume has to be decreased until the density of the float is higher than that of the water. Between 30s and 80s, the volume decreases to the minimum volume of $\frac{V_0}{2}$, and then the volume change rate plateaus again, as the volume cannot decrease any further. In the last part of the graph, the volume change rate returns to the initial guess, as changes in the volume in the last few moments before the target location

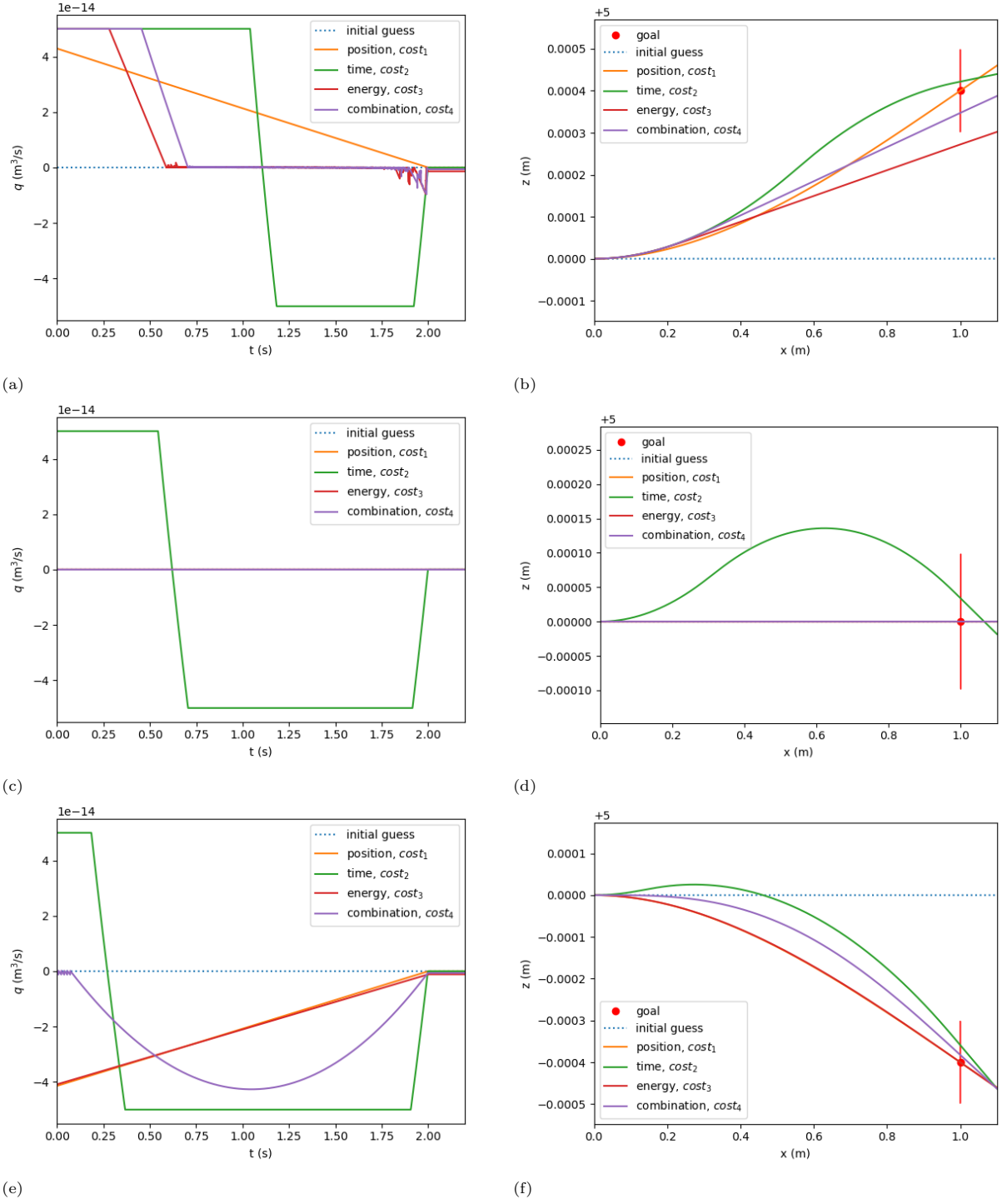
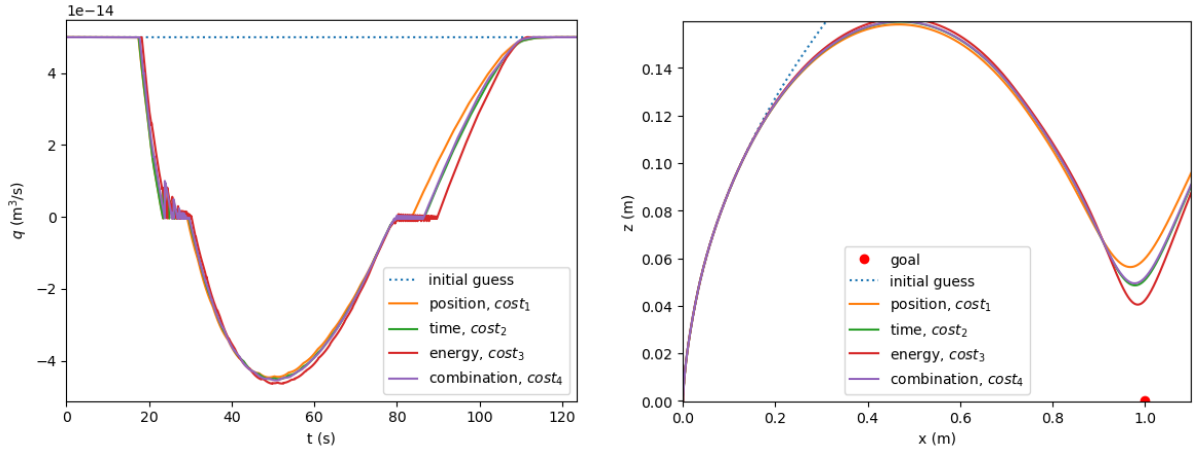


Figure 9: Optimization for a float which is to be steered from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, z_g)$ for different z_g and different cost functions, as defined in equations (55), (60), (70), and (71). From top to bottom, $z_g = 5.0004\text{m}$, $z_g = 5.0\text{m}$, and $z_g = 4.9996\text{m}$. $Q_0 = \mathbf{0}$ and 70 iterations are used. The left figures depict the optimized Q_f, while the figures on the right show the corresponding trajectories, where the plotted accepted deviation from the target location is the initial diameter of the float.
Some of the optimal trajectories overlap; in figures 9c and 9d, positional, energetic, and combined optimizations overlap, and in figures 9e and 9f, the positional and energetic optimizations overlap. Note that all vertical axis scales are very small because the float is very small, and $q_{max} \approx 5\ 10^{-14}\text{m}^3/\text{s}$.

is ‘reached’ do not affect the final location much, and therefore will only be changed slowly per iteration.



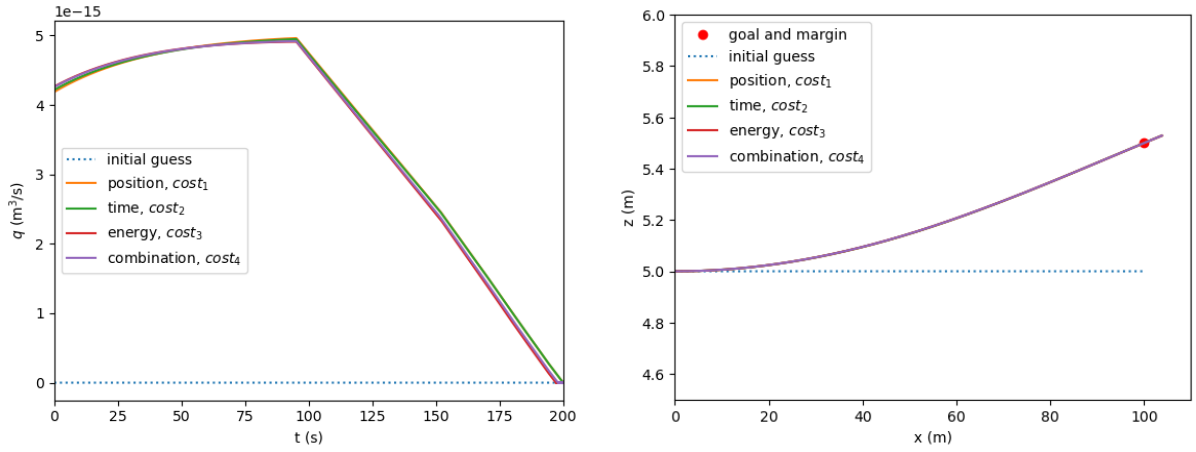
(a) (b)

Figure 10: Trajectory optimisation from $(0.0\text{m}, 0.0\text{m})$ to $(1.0\text{m}, 0.0\text{m})$ for different cost functions, as defined in equations (55), (60), (70), and (71), with $Q_0 = Q_{,max}$ and 70 iterations. 10a shows the optimized Q_f and 10b shows the corresponding trajectories. Note that all vertical axis scales are very small because the float is very small, and $q_{max} \approx 5\ 10^{-14}\text{m}^3/\text{s}$.

4.3.3. Target location far away

The last tested situation is a float that starts in the middle of the water column and has to traverse a longer horizontal distance than before: from $(0.0\text{m}, 0.0\text{m})$ to $(1.0\text{m}, 0.0\text{m})$. The optimized Q_f and corresponding trajectories for the four different cost functions are shown in figure 10.

The Q_f are almost equal, and all trajectories except for that corresponding to the energy come within D_0 of the target location; the $w_p cost_p$, equal to $cost_1$ or eq. (55), are respectively $9\ 10^{-7}$, 0.2, 7.2 and 0.9. Additionally, the Q_f corresponding to $cost_2$, $cost_3$, and $cost_4$ do not match the expected shapes. This will be discussed further in the next chapter.



(a) (b)

Figure 11: Trajectory optimisation from $(0.0\text{m}, 5.0\text{m})$ to $(100.0\text{m}, 5.5\text{m})$ for different cost functions, as defined in equations (55), (60), (70), and (71), with $Q_0 = \mathbf{0}$ and 70 iterations. 11a shows the optimized Q_f and 11b shows the corresponding trajectories. Note that the vertical axis scale in subfigure 11a is even smaller than the vertical scales in the other Q graphs in this chapter.

5. Discussion

The results of the individual tests were discussed in the previous chapter. In this chapter, the flaws of gradient descent, the flaws in the implementation, and the implications and conclusions for general flow fields and floats will be discussed.

5.1. Disadvantages of gradient descent

It is important to be aware of the disadvantages of gradient descent. The most prominent disadvantage is typical for the majority of numerical optimization methods: gradient descent will find a local minimum. When the cost function has more than one local minimum, the found solution will depend on the initial guess and an appropriate initial guess is essential.

A second challenge is the choice of learning rate η. A constant function is often not optimal, as a too low η requires a high number of iterations, and a too high η can cause fluctuations around the minimum of the cost function. In the strategy proposed in this thesis, η was chosen in such a way that the maximal change in one element of Q per step was constant until the cost function no longer decreased, and was then lowered. This is not the only possible approach. [7] For future applications, it is recommended to determine a more efficient learning rate, such that fewer iterations are required.

5.2. Code efficiency

The most prominent flaw in the implementation was the efficiency of the code. Every iteration of the implemented program requires a lot of computations. Since the cost depends on the trajectory, the chosen integration method is repeated for every time step, and then the gradient of the input at each of those steps is determined with reverse automatic differentiation. This has two inconvenient consequences. First, an iteration can take a lot of time when the float is to be steered during a long time interval. Second, the Python kernel might run out of memory when too many inputs or iterations are evaluated.

To reduce the number of computations per iteration, it was chosen to integrate with Forward Euler during the optimization. To check whether the errors introduced by this method cause convergence to another location than the intended target location, the more accurate Runge-Kutta was used to determine the trajectory corresponding to Q_f.

However, even with this reduction, the number of computations was a limiting factor. For example, in the results for navigation over a long horizontal distance, it was not evident whether the Q_f did not meet the expectations due to a too low number of iterations or for another reason. While the obvious next step was to increase the number of iterations and repeat the optimization, this proved to be impossible with the current implementation, as the Python Kernel kept being terminated due to out-of-memory errors, even when only 100 iterations were considered. This implies that the current implementation of the strategy is not robust and efficient enough to be applicable in larger studies. Some suggestions to increase the efficiency of the implemented strategy are:

- To use another coding environment than Python, for example C. This would improve the efficiency, as Python is an interpreted coding language and such languages are generally slower and require more memory than compiled coding languages. [17]
- To test other Python packages than JAX, as this might not be the most efficient package.
- To decrease the number of iterations that is required before an acceptable Q_f is obtained, for example by determining a better initial guess, or a better learning rate, as described in section 5.1.

5.3. Implications

5.3.1. Conclusions for the considered field

Even though only five situations are tested, it is possible to draw conclusions for the whole field. Based on the standard movements for positive, negative, and zero vertical displacements, it can be concluded that the optimization method can indeed be used to optimize the buoyancy changes of a float in order to navigate it to a certain location, even with different secondary objectives such as energy-efficiency, in at least a few simple cases. However, from the results for a target location on the seabed and a target location 100m away, it can be concluded that there are limitations on when the optimization strategy can be used.

It can only be concluded that the optimization method can be applied to the whole field if all reasons why no optimal Q can be found are due to the efficiency of the implementation and not due to the optimization strategy itself. There are four factors that can potentially restrict the performance of the implemented method for general initial and target locations, compared to the performance for the tested movements up, down, and to the same height. Those limiting factors are the reachability of the target location, the boundary conditions on the state of the float, whether Q can converge to the true optimal Q within the maximal number of iterations, and the choice of weighting factors.

Reachability

The target location can only be reached if a sequence of density changes exists such that the float reaches the destination. In a two-dimensional linear flow field, the reachable locations can be found by determining the paths for maximal positive and maximal negative buoyancy changes. The locations that theoretically can be reached are between those extremes. There is no point in trying to optimize the buoyancy changes when the target location cannot be reached.

Boundary conditions on the state of the float

In the tests for the standard movements, none of the boundary conditions on the state of the float had to be enforced: the volume of the float was never maximal or minimal and the float never came close to the seabed or surface. When the float is to be steered for a longer time or closer to the seabed or surface, those boundary conditions might have to be enforced. In that case, there is a possibility that the implemented program is unable to properly calculate the gradients, which would greatly limit the potential applications of the optimization strategy.

However, from the results for navigation to the seabed, it can be concluded that the enforcement of the boundary conditions is not a limiting factor for the performance of the optimization strategy: even though the target location is not reached, the optimized Q does satisfy the boundary conditions for the volume.

Efficiency in convergence

When Q cannot converge to the true optimal Q within the maximal number of iterations, the optimized Q_f will be inaccurate. This is in all likelihood the reason why the target location on the seabed is not reached, and this could be the reason why the optimized inputs for the energy in the results from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$ and for all secondary objectives in the results for the target location far (100m) from the initial position are inadequate.

When the convergence is the only limiting factor on the performance, it can be concluded that the strategy theoretically works, but that the possibilities for applications are limited by the efficiency of the implemented code, which can be improved in any of the methods mentioned in section 5.2.

Choice of weighting factors

Although the results for a positive, negative, and zero vertical displacement prove that there are situations where the optimization method can be used when the objective is to navigate the float in a time or energy-efficient way, it should be noted that the performance for those secondary objectives is highly dependent on the choice of the weighting factors. When the weighting factors are chosen incorrectly, the minimum of the cost function might not satisfy the condition that the float ends up close to the target location, or it might take a lot of iterations before optimizing for the secondary objective decreases the cost function more than decreasing the distance to the target location does. In section 3.3, it was attempted to derive general weighting factors, but the assumptions and simplifications on the trajectories might have been too crude to apply to long distances, which is a second potential explanation for the flawed optimized results for the energy in the results from $(0.0\text{m}, 5.0\text{m})$ to $(1.0\text{m}, 5.0004\text{m})$ and for all secondary objectives in the results for the target location far (100m) from the initial position.

In future research, especially when the considered float has a specific secondary objective, it is recommended to determine more suitable weighting factors.

All in all, the flaws in the performance of the implemented strategy are in all likelihood due to the number of iterations and the choice of weighting factors, and not due to the optimization strategy and implementation. It can be concluded that the optimization strategy can be used for initial and target locations in the whole field, as long as the target location is reachable, and when the float has to be

navigated for a longer time than a few seconds, the implementation has to be more efficient than the current one, and the weighting factors for time and energy have to be chosen more carefully.

5.3.2. Conclusions for all idealized two-dimensional linear fields

In this section, it will be made plausible that the previous conclusions for the field with variables chosen as in table 1 can be applied to small floats in all general idealized two-dimensional linear flow fields. That is, flow fields where the density and dynamic viscosity are constant, and the assumption of Stokes flow can be used.

Most choices of variables in the tested field only cause an offset. For example, if the leakage is not neglected, some or all of the values of Q have to be higher to compensate for the losses, and when the initial density is higher, more volume has to be added before the float can start rising. Furthermore, it is likely that additional components of the flow field in the vertical direction will also only cause an offset, as long as the assumptions leading to Stokes flow hold in this direction, but it was not possible to test this in the time available for this thesis.

Therefore, the limitations for applying the results in other fields are the assumptions leading to $\mathrm{Re} < 1$: α has to be small enough, and the density of the float cannot become too high or low.

5.3.3. Conclusions for general floats and flow fields

This thesis is a first step in testing the viability of a gradient descent-based optimization strategy for navigating a float to a target location using only density changes. Many additional steps have to be taken before this strategy can be used for a real float, as a lot of assumptions were made.

Firstly, the considered float was spherical and had an initial diameter $D_0 = 10^{-4}\mathrm{m}$. This simplified the situation, as the rotational inertia of the float could be neglected and the flow could be approximated as Stokes flow around a sphere. When the strategy is tested for a bigger float or a float with another shape, the implementation has to be altered to include rotational inertia and the non-simplified equations of motion.

Furthermore, the buoyancy-changing method was based on simplified processes in a swim bladder filled with an ideal gas, while the buoyancy of profiling floats and underwater gliders is commonly controlled with bladders filled with oil that are expanded and drained with a hydraulic pump. [1] The choice of a modified swim bladder did actually increase the complexity of the model, as the physical system is unstable because a rising float will expand, hence rise faster, and vice versa, while Oil is far less compressible than gasses are. It is recommended that future research works with the more common oil-filled bladders instead of swim bladders, as more research has been done for those bladders, and the use of such a bladder would greatly simplify the formula describing the volume change.

Lastly, the considered flow field was very idealized. It was two-dimensional and linear, the density and dynamic viscosity of the water were constant, and the Reynolds number had to be smaller than 1 at all times. All of those assumptions can only be applied to a real situation on a very small time scale. When longer time scales are considered, the model will have to be expanded to include the third dimension, non-linear and time-dependent flow fields, a changing water density and viscosity, and $\mathrm{Re} \geq 1$. For future research into the viability of a steerable buoyancy-changing float, it is recommended to include those generalizations one at a time.

6. Conclusion

The goal of this thesis was to take the first steps necessary to determine whether it is possible to navigate an oceanographic float that is only able to alter its buoyancy to specific targets, while also minimizing the required time or energy, with an optimization strategy based on gradient descent. Such a float could possibly be cheaper and more practical to use than underwater gliders and propelled AUVs in coastal regions. Those first steps consisted of the construction and testing of a gradient descent-based optimization strategy in an idealized linear two-dimensional flow field.

To do so, a mathematical model for a volume-changing spherical float in water was derived and then simplified for the situation where the water has a constant density and viscosity, $\mathrm{Re} < 1$ at all times, and the flow field is two-dimensional and linear. Then, an optimization strategy for optimizing the buoyancy changes was derived, and it was determined how the position, time, and energy can be combined into one cost function, with weighting factors that depend on whether the float should follow a time or energy-efficient path. In a nutshell, the optimization strategy is to start from an initial guess Q_0 for the required sequence of volume change rates to reach the target location, and then iteratively alter the guess by taking small steps in the opposite direction of the gradient of the cost function, until an optimal sequence Q is determined. The strategy was tested in five different initial and target locations for four different cost functions.

For target locations that could be reached within a few seconds, the strategy, implemented as described in chapter 4, could be used to properly optimize the volume change rates for all cost functions for positive, negative, and zero vertical displacement targets, in most cases within 70 iterations. Although the optimized volume change rates could not steer the float in $1\mathrm{m}$ from the seabed back to the seabed, this test proved that the implemented strategy can be used even when there are restrictions on the maximum and minimum volume of the float. The optimized volume change rates in the last test, a target location far ($100\mathrm{m}$) away, did not meet the expectations for a time or energy-efficient optimal Q within 70 iterations. It could not be determined whether this was due to inadequately chosen weighting factors in the cost functions, or due to the number of iterations, as the current coding efficiency prevented tests with more iterations.

All in all, it can be concluded that the optimization strategy can most probably be applied to small floats in all idealized two-dimensional linear flow fields, but that the current implementation is not efficient enough to optimize for targets far away, and that a reassessment of the weighting factors between position, time, and energy in the cost functions is necessary for longer distances or specific floats.

This is an acceptable first step in testing the viability of navigating a buoyancy-changing float to specific targets, but due to the current efficiency and the large number of assumptions, it is not yet possible to assess whether a workable program for optimizing buoyancy changes for navigation of a real-life float can be made. Many steps have to be taken before a float based on these principles can be released into our coastal waters.

The first step in future research should be to increase the coding efficiency. Thereafter, other steps toward a more realistic situation can be taken, such as testing for non-linear flow fields, three-dimensional fields, and bigger floats.

References

[1] About the Argo program. https://argo.ucsd.edu/about/. Accessed: 23-10-2023.

[2] Xavier André et al. “Preparing the New Phase of Argo: Technological Developments on Profiling Floats in the NAOS Project”. Frontiers in Marine Science 7 (2020).

[3] Atilim Gunes Baydin et al. “Automatic differentiation in machine learning: a survey”. Journal of Marchine Learning Research 18 (2018), pp. 1–43.

[4] James Bradbury et al. JAX: composable transformations of Python+NumPy programs. Version 0.3.13. 2018.

[5] Christopher E. Brennen. Introduction to Added Mass. http://brennen.caltech.edu/fluidbook/basicfluiddynamics/unsteadyflows/addedmass/introduction.pdf. Accessed: 12-09-2023. 2006.

[6] Lyman J Briggs. “Effect of spin and speed on the lateral deflection (curve) of a baseball; and the Magnus effect for smooth spheres”. American Journal of Physics 27.8 (1959), pp. 589–596.

[7] Christian Darken, Joseph Chang, John Moody, et al. “Learning rate schedules for faster stochastic gradient search”. Neural networks for signal processing. Vol. 2. Citeseer. 1992, pp. 3–12.

[8] Tuhin Das, Ranjan Mukherjee, and Jonathan Cameron. “Optimal trajectory planning for hot-air balloons in linear wind fields”. Journal of guidance, control, and dynamics 26.3 (2003), pp. 416–424.

[9] E.J. Denton. “6 - The Buoyancy of Fish and Cephalopods”. Progress in Biophysics and Biophysical Chemistry 11 (1961), pp. 177–234.

[10] Pablo Rodríguez Fornes, Núria Pujol Vilanova, and David Roque Atienza. “AUV Risk Management in Coastal Water surveys”. Instrumentation viewpoint (2013).

[11] FR Harden Jones and P Scholes. “Gas secretion and resorption in the swimbladder of the cod Gadus morhua”. Journal of Comparative Physiology B 155 (1985), pp. 319–331.

[12] L.P.B.M Janssen and M.M.C.G. Warmoeskerken. Transport Phenomena Data Companion. 3rd ed. Delft Academic Press / VSSD, 2006.

[13] W Kuhn et al. “The filling mechanism of the swimbladder: Generation of high gas pressures through hairpin countercurrent multiplication”. Experientia 19 (1963), pp. 497–511.

[14] George N Lapennas and Knut Schmidt-Nielsen. “Swimbladder permeability to oxygen”. Journal of Experimental Biology 67.1 (1977), pp. 175–196.

[15] Pierre FJ Lermusiaux et al. “A future for intelligent autonomous ocean observing systems”. Journal of Marine Research 75.6 (2017), pp. 765–813.

[16] Daoliang Li and Ling Du. “Auv trajectory tracking models and control strategies: A review”. Journal of Marine Science and Engineering 9.9 (2021), p. 1020.

[17] Abhinav Nagpal and Goldie Gabrani. “Python for Data Analytics, Scientific and Technical Applications”. 2019 Amity International Conference on Artificial Intelligence (AICAI). 2019, pp. 140–145.

[18] R Bosede Ogunrinde, S Emmanuel Fadugba, and J Temitayo Okunlola. “On some numerical methods for solving initial value problems in ordinary differential equations”. IOSR Journal of Mathematics (IOSRJM) 1 (2012), pp. 25–31.

[19] Seon Ki Park and Kelvin K Droegemeier. “Sensitivity analysis of a 3D convective storm: Implications for variational data assimilation and forecast error”. Monthly weather review 128.1 (2000), pp. 140–159.

[20] Blaise Pascal. Traités de l’equilibre des liqueurs. Paris, 1663.

[21] Teledyne Webb Research. APEX Profiling Float. http://www.argo.org.cn/data/apex_apf11_usermanual.pdf. Accessed: 25-10-2023. 2014-2017.

[22] AD Rijnsdorp, M Van Stralen, and Hendrik Willem Van Der Veer. “Selective tidal transport of North Sea plaice larvae Pleuronectes platessa in coastal nursery areas”. Transactions of the American Fisheries Society 114.4 (1985), pp. 461–470.

[23] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. arXiv preprint arXiv: 1609.04747 (2016).

[24] Mostafa H Sharqawy, John H Lienhard, and Syed M Zubair. “Thermophysical properties of seawater: a review of existing correlations and data”. Desalination and water treatment 16.1-3 (2010), pp. 354–380.

[25] GG Stokes. “On the effect of internal friction of fluids on the motion of pendulums”. Transactions of the Cambridge Philosophical Society 9.8 (1850), p. 106.

[26] Espen Strand, Christian Jørgensen, and Geir Huse. “Modelling buoyancy regulation in fishes with swimbladders: bioenergetics and behaviour”. Ecological Modelling 185.2 (2005), pp. 309–327.

[27] C. Vuik et al. Numerical Methods for Ordinary Differential Equations. Delft Academic Press / VSSD, 2018.

[28] Runfeng Zhang et al. “Surfacing Positioning Point Prediction of Underwater Glider with a New Combination Model”. Journal of Marine Science and Engineering 11.5 (2023).

A. Automatic Differentiation

A.1. Why would Automatic Differentiation be used?

There are four methods for computing derivatives in computer programs: coding manually derived derivatives, Numerical Differentiation, Symbolic Differentiation, and Automatic Differentiation. [3] They all have strengths and weaknesses:

Coding manually worked out derivatives leads to exact solutions, but it is prone to error and time-consuming. [19]

Numerical Differentiation or finite difference methods are easy to implement, but round-off and truncation errors can result in large deviations from the real values. Additionally, it scales poorly for gradients, thus is inefficient for cases with many inputs. [3]

Symbolic Differentiation is when the program applies differentiation rules to a closed-form expression. It provides exact expressions, but those expressions are often huge and cluttered. [3]

Like Symbolic Differentiation, Automatic Differentiation (AD) breaks down the function into elementary arithmetic operations and functions, and applies the chain rule. However, AD is certainly not the same as Symbolic Differentiation, as the second determines a formula for the derivative, while AD determines only the result for a specific input. AD counters most of the downsides of the other methods: it is possible to differentiate functions without a closed-form expression, like loops and recursion, and to evaluate derivatives at nearly machine precision, since the derivatives are determined through accumulation of values during the execution of the code, rather than explicit expressions. [3]

A.2. How does Automatic Differentiation work?

In a nutshell, AD breaks down the function on which it is applied into elementary arithmetic operations, like addition and multiplication, and elementary functions, like $\sin()$ and $\exp()$, and then applies the chain rule repeatedly on intermediate values. There are two types of AD: forward and reverse accumulation. Forward accumulation builds the derivative starting at the input, while reverse accumulation starts at the output. The different accumulation methods will be explained using the example function $f(x_1, x_2) = \sin(x_1) + x_1 x_2^2$.

A.2.1. Forward accumulation

To compute the derivative of f with respect to x_1 for some input, for example $(\pi, 2)$, with forward accumulation, first it is determined how f is composed of intermediate values that interact only with elementary operations. Subsequently, the values for every intermediate value and the corresponding derivatives with respect to x_1 can be determined. Figure 12 shows the intermediate values. Define

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1} \tag{72}$$

Knowing $\dot{v}_{-1} = 1$ and $\dot{v}_0 = 0$, the values and derivative of intermediate steps are calculated with the previous values until $\frac{\partial f(\pi,2)}{\partial x_1} = 3$ is determined, this is visible in table 2. The downside of this method is that the whole process has to be repeated to determine the derivative with respect to x_2.

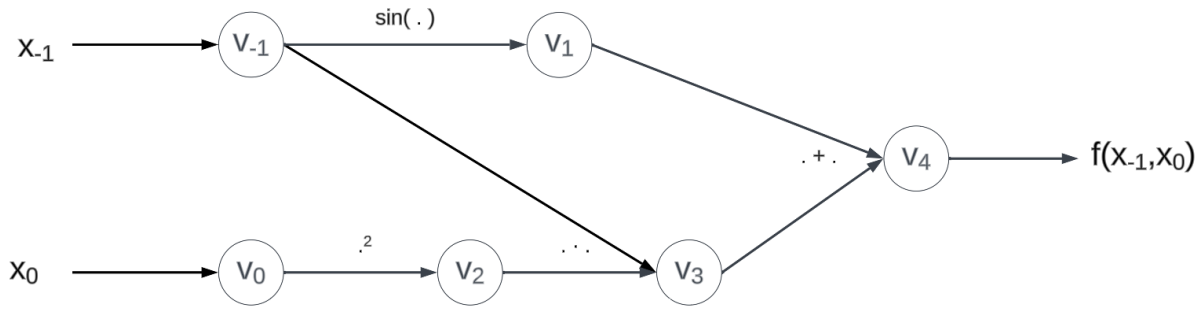


Figure 12: Flowchart for $f(x_1, x_2) = \sin(x_1) + x_1 x_2^2$ with intermediate values.

Table 2: Computation of the derivative of $f(x_1, x_2) = \sin(x_1) + x_1 x_2^2$ with respect to x_1 using forward accumulating AD. The first column has the intermediate values: v_{-1} for x_1 and v_0 for x_2, then $v_i, i \in \{1, 2, \ldots\}$ for all elementary calculations. The second column shows how the v_i depends on previous values, while the third column has the values of the example. The fourth and fifth column have the calculation of the derivatives and how they can be computed using the previous values. The derivative is determined by working from top to bottom.

	value of v_i		derivative of v_i w.r.t. x_1	
intermediate values	dependency	value	dependency	value
v_{-1}	x_1	π	$\dot{x}_1$	1
v_0	x_2	2	$\dot{x}_0$	0
v_1	$\sin(v_{-1})$	$\sin(\pi)$	$\cos(v_{-1}) \cdot \dot{v}_{-1}$	$-1 \cdot 1$
v_2	v_0^2	2^2	$2 \cdot v_0 \cdot \dot{v}_0$	$2 \cdot 2 \cdot 0$
v_3	$v_{-1} \cdot v_2$	$\pi \cdot 4$	$\dot{v}_{-1} \cdot v_2 + v_{-1} \cdot \dot{v}_2$	$1 \cdot 4 + \pi \cdot 0$
v_4	$v_1 + v_3$	$0 + 12.566$	$\dot{v}_1 + \dot{v}_3$	$-1 + 4$
$f(\pi, 2)$	v_4	12.566	$\dot{v}_4$	3

A.2.2. Reverse accumulation

When using reverse accumulation, the algorithm works from the output towards the input. The derivatives now are defined as

$$\overline{v}_i = \frac{\partial f(x_1, x_2)}{\partial v_i} \tag{73}$$

Let the 'successors of i' be defined as the j's such that v_j depends directly on v_i. Thus in the example, even though v_4 is influenced by v_0, 4 is not a successor of 0, only 2 is. Then

$$\overline{v}_i = \sum_{j \in \text{successors of } i} \frac{\partial f(x_1, x_2)}{\partial v_j} \frac{\partial v_j}{\partial v_i} = \sum_{j \in \text{successors of } i} \overline{v}_j \frac{\partial v_j}{\partial v_i} \tag{74}$$

In the example, only -1 has two successors, namely 1 and 3, and the other $i's$ each have only one successor. First, all the intermediate v_i are computed with a forward run, and then the derivatives are determined backward, starting with $\overline{v}_4 = 1$, as can be seen in table 3. Both $\frac{\partial f(x_1, x_2)}{\partial x_1}$ and $\frac{\partial f(x_1, x_2)}{\partial x_2}$ can be calculated in one sweep.

Table 3: Computation of the derivative of $f(x_1, x_2) = \sin(x_1) + x_1 x_2^2$ with respect to x_1 and x_2 using reverse accumulating AD. The first column has the intermediate values: v_{-1} for x_1 and v_0 for x_2, then $v_i, i \in \{1, 2, \ldots\}$ for all elementary calculations. The second column shows how the v_i depends on previous values, while the third column has the values of our example. The fourth and fifth columns have the calculation of the derivatives of f w.r.t. the intermediate values and how they can be computed using the succeeding values. The v_i are computed from top to bottom, and the $\overline{v_i}$ from bottom to top.

	values of v_i		derivative of f w.r.t. v_i	
intermediate values	dependency	value	dependency	value
v_{-1}	x_1	π	$\overline{v}_1 \frac{\partial v_1}{\partial v_{-1}} + \overline{v}_3 \frac{\partial v_3}{\partial v_{-1}} = \overline{v}_1 \cdot \cos(v_{-1}) + \overline{v}_3 \cdot v_2$	$1 \cdot -1 + 1 \cdot 4 = 3$
v_0	x_2	2	$\overline{v}_2 \frac{\partial v_2}{\partial v_0} = \overline{v}_2 \cdot 2 \cdot v_0$	$3.142 \cdot 2 \cdot 2 = 12.566$
v_1	$\sin(v_{-1})$	$\sin(\pi)$	$\overline{v}_4 \frac{\partial v_4}{\partial v_1} = \overline{v}_4 \cdot 1$	$1 \cdot 1$
v_2	v_0^2	2^2	$\overline{v}_3 \frac{\partial v_3}{\partial v_2} = \overline{v}_3 \cdot v_{-1}$	$1 \cdot \pi$
v_3	$v_{-1} \cdot v_2$	$\pi \cdot 4$	$\overline{v}_4 \frac{\partial v_4}{\partial v_3} = \overline{v}_4 \cdot 1$	$1 \cdot 1$
v_4	$v_1 + v_3$	$0 + 12.566$		1
$f(\pi, 2)$	v_4	12.566	$\overline{v}_4$	1

B. Code

The following is an excerpt from the code. This contains one of the cost functions ($cost_2$), how the gradient of this function is determined, and the gradient descent algorithm.

```
import jax.numpy as jnp
from jax import grad

def f_cost_tijd(Vc_control,x0,z0,vz0,V0,xf,zf):
    final_xzvzV, temp_cost = lax.scan(scan_fun, (x0,z0,vz0,V0), Vc_control)
        # scan_fun integrates the trajectory; temp_cost is an array with the energy, x, and z
                                                    at every dt

    z_iv, cost_t = minimum(temp_cost[1],temp_cost[2],xf)
        # minimum interpolates the postion where cost[1]=xf, and determines the corresponding
                                                    z and t

    cost_pos = (z_iv-zf)**2/Dn**2
    tijd_factor = 2*alpha*(Dn+zf)**2/xf/Dn
    return cost_pos + tijd_factor*cost_tijd

grad_cost_tijd = grad(f_cost_tijd,argnums=0)

def gradient_descent(control0,x0,z0,vz0,V0,control_min,control_max,cost,grad_cost,delta=0.001,
                                                    max_iter=70):
    control = control0
    cost_all = []

    cost0 = cost(control,x0,z0,vz0,V0,xf,zf)
    cost_all.append(cost0)

    for i in range(max_iter):
        grad_cost0 = grad_cost(control,x0,z0,vz0,V0,xf,zf)
        search_direction = -grad_cost0

        check_array = search_direction[((control_min+delta<control)|(search_direction>0))&((
                                                    control_max-delta>control)|(
                                                    search_direction<0))]
        if jnp.size(check_array) == 0:
            delta = delta/2
            continue
        max_step = jnp.max(jnp.abs(check_array))
        search_direction = search_direction/max_step

        control_new = control + delta*search_direction
        control_new = jnp.clip(control_new,control_min,control_max)

        cost_new = cost(control_new,x0,z0,vz0,V0,xf,zf)
        if cost_new <= cost0:
            cost0 = cost_new
            control = control_new

        else:
            delta = delta/2

        cost_all.append(cost0)
    return control, cost_all, delta
```