Lab 12 - A thermocline in the Charles River Basin?

2.S01 Introduction to Autonomous Underwater Vehicles

Spring 2025

Supun Randeni, supun@mit.edu
Michael Benjamin, mikerb@mit.edu
Mike Sacarny, msacarny@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

1 Learning Objectives

- Learning how to plan an AUV operation to achieve specific goals
- Understanding the various factors that need to be considered when planning an AUV operation
- Learning the importance of virtual experiments
- Understanding virtual and real-life Concept-of-Operations (ConOps)

2 What is a Thermocline

A thermocline is a distinct layer in oceans where the water temperature changes rapidly with depth. At the surface, water temperature is influenced by solar heating during the day, cooling at night, and heat exchange with the atmosphere through convection. Wind, waves, and currents help mix this surface layer down to about 50 to 100 meters. Below the surface lies cold, deep water, which originates in the polar regions and moves slowly toward the equator, shaped by complex interactions with currents and landmasses. The thermocline lies between these two layers, warm surface water above and cold deep water below, and typically ranges from 300 to 1,000 meters thick.

In major ocean basins, the thermocline often reflects a "permanent" or long-term average temperature profile, with a seasonal variation mostly affecting the upper mixed layer. The specific structure of the thermocline can vary widely depending on location. In shallower bodies of water, seasonal changes are more dominant and can significantly alter the profile.

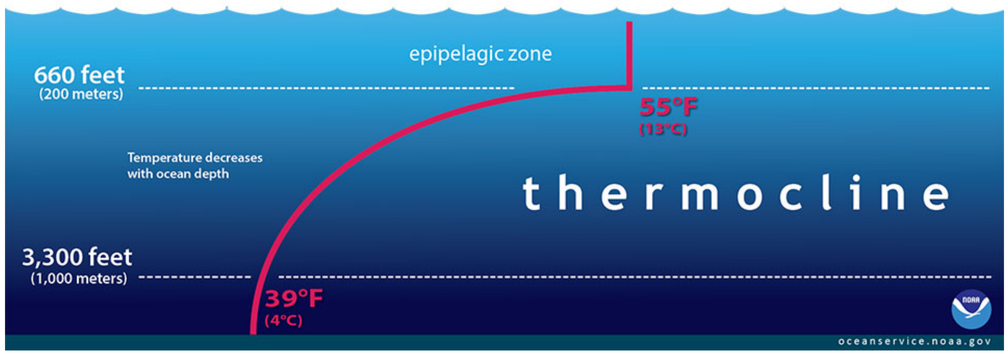

Figure 1: The red line in this illustration shows a typical seawater temperature profile. In the thermocline, temperature decreases rapidly from the mixed upper layer of the ocean to much colder deep water in the thermocline.

2.1 Effects on buoyancy

Temperature has a significant effect on the density of water. In fact, density profiles are often the mirror image of temperature profiles. Water is typically warmest at the surface; and therefore least dense. As depth increases, especially in the presence of a thermocline, the sharp drop in temperature leads to a corresponding increase in density. In some cases, underwater vehicles have been known to become trapped near the thermocline layer due to unexpected changes in buoyancy caused by this density shift.

2.2 Effects on sound propagation

In the ocean, the speed of sound is strongly influenced by temperature in the upper layers and by pressure in the deeper layers. As water temperature decreases, the speed of sound decreases, while increasing pressure with depth causes the speed of sound to increase. Sound waves naturally bend, or refract, toward regions where the sound speed is lower. As a result, when a sound wave passes through a thermocline, it bends downward due to the decrease in speed of sound with decreasing temperature, but then bends upward again as increasing depth causes the sound speed to rise due to pressure. This repeated up-down bending of low-frequency sound waves allows them to travel thousands of meters with minimal energy loss.

2.3 Thermocline in smaller bodies of water

Thermoclines are also present in lakes, particularly in colder climates, where they give rise to a phenomenon known as stratification. During the summer, warm water, being less dense, remains on top of colder, denser water, with a thermocline acting as the boundary between them. The warm water is exposed to the sun during the day, creating a stable system with minimal mixing between the warm and cold layers, especially during calm weather.

As a result of this stability, oxygen levels below the thermocline decrease over time. Since the water beneath the thermocline does not circulate to the surface, oxygen is gradually depleted by aquatic organisms. As winter approaches, surface water cools due to nighttime temperature drops, eventually reaching a point where its density surpasses that of the deeper water. This triggers overturning, as the denser surface water sinks under the force of gravity. Wind or other processes, such as currents, can further facilitate this mixing.

3 Objectives of Your Operation

Your goal for this operation is to use the AUV you have built over the past five weeks to find out if there is a thermocline in the Charles River Basin. If you do find one, your job is to measure it and its variation with the depth (i.e. temperature-depth profile). The Blue Robotics depth sensor onboard your SeaBeaver AUV also measures water temperature and posts it to `MOOSDB` under the variable name `EXTERNAL_TEMPERATURE`, which you can use to capture the thermocline.

This exercise is open-ended, and you are free to decide how you want to approach it. However, keep in mind that the maximum in-water runtime for each AUV mission is 8 minutes. To help you get started, we have included some useful tips and guidelines to consider while planning your experiment, along with a few mission planning tools that you can use.

4 Key Considerations When Planning Your AUV Operation

Planning an AUV operation involves achieving mission objectives while prioritizing safety. This includes ensuring the safety of yourself, others around you, and the AUV itself. Several key factors should be considered when planning an AUV operation.

4.1 Repeatability of data

In this operation, the goal is to determine whether there is a thermocline in the Charles River Basin. To achieve this, you'll need to get your AUV to perform at least one dive cycle; i.e. descending close to the riverbed and surfacing while capturing water temperature data.

But, how can you verify that a single temperature-depth profile is valid? For example, the Blue Robotics depth sensor is designed to measure external pressure, with water temperature as a byproduct. As a result, sensor readings may not always be precise or accurate. To gain more confidence in your data, you can perform multiple dive cycles (see Figure 2); however, this comes at the cost of reduced in-water runtime available for spatial coverage.

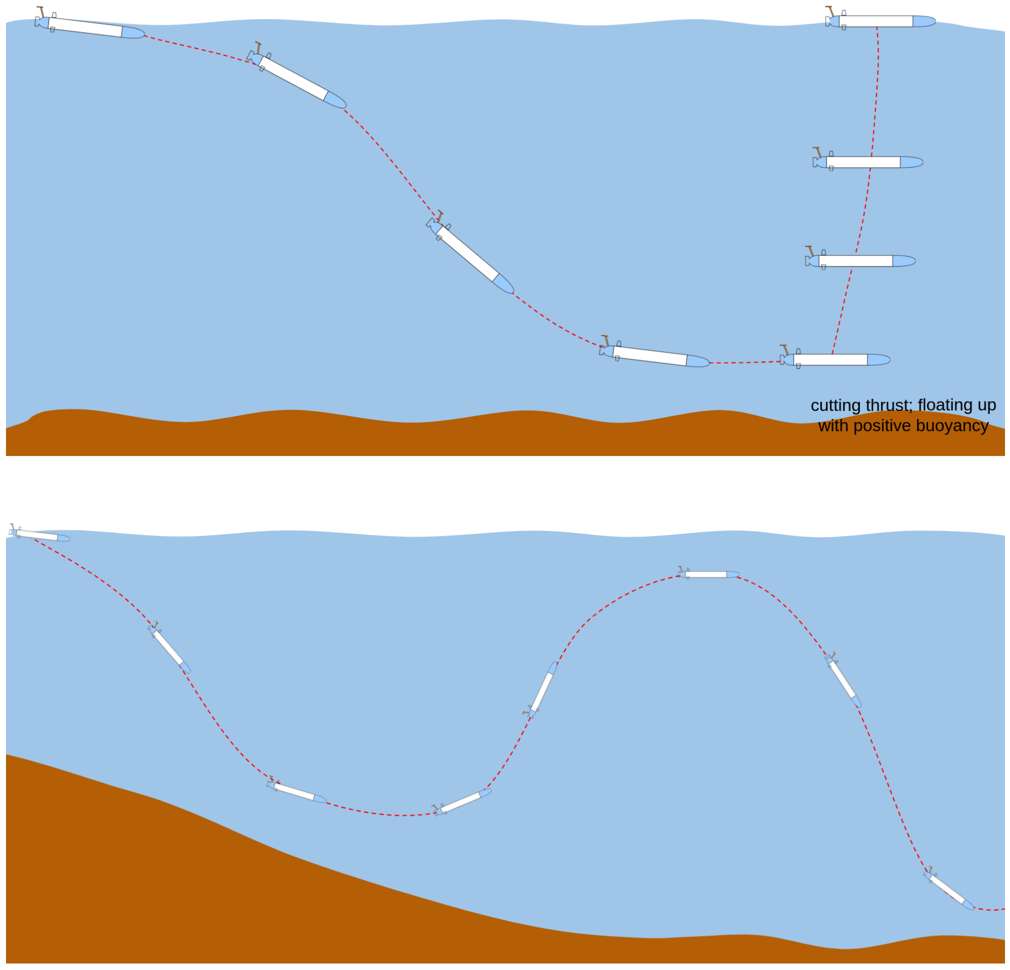

Figure 2: Top Panel: The AUV is performing a single temperature profile (against depth), descending close to the riverbed, cutting thrust, and floating back up using its positive buoyancy. Bottom Panel: The AUV is conducting multiple temperature profiles by yo-yoing through the depth plane, by using a sequence of depth commands.

4.2 Temporal and spacial variation of data

The temperature-depth profile may have both temporal and spatial variability. Temporal variability refers to how the profile changes over time, which can be short-term (e.g., day vs. night) or long-term (e.g., summer vs. winter). Our current operation is aimed towards understanding if there is a thermocline in the Charles River basin in mid-May. This operation is focused on determining whether a thermocline exists in the Charles River Basin during mid-May. However, it maybe important to log contextual information; such as the time of the experiment, weather conditions, and other environmental factors, as these may influence the results and aid in future comparisons.

Spatial variability refers to how the temperature-depth profile changes with location within the river basin. For example, the profile near the MIT Sailing Pavilion dock may differ from that observed in the middle of the river due to factors such as proximity to boundaries (e.g., walls and structures), human activity, wind effects, water depth and other localized influences. Therefore, with limited in-water runtime, it may be necessary to strategically collect measurements from different locations across the river.

4.3 Sensor calibration

Calibration and verification of this temperature reading may be important to determine the accuracy of the recorded data. A simple way to check the sensor's accuracy is by comparing its readings with an external temperature source. For example, you can compare the temperature measurements recorded before and after the dive (i.e. when the AUV is at the dock) with an independent temperature measurement at the same location.

4.4 Water depth

As you can imagine, your mission plan will likely need to bring the AUV close to the riverbed to capture a complete temperature-depth profile. However, this comes with a significant risk: the Charles River has a muddy bottom, and if your AUV touches down, it may get stuck - and potentially lost. This means you'll need to carefully balance data quality with the safety of your vehicle.

The SeaBeaver AUVs you built are not equipped with altimeters, so they cannot autonomously detect and avoid the riverbed. Therefore, when planning your mission, it is essential to plan the vehicle's depth based on the sensing area. The bathymetry (i.e., water depth) of the Charles River varies by location. You can find a bathymetric chart here: `https://gisviz.mit.edu/maps/charleschart/` (note that depth values are in feet).

4.5 Control performance

As you have observed by now, the depth control performance of your AUV depends heavily on how well the PID constants are tuned. If not tuned properly, the vehicle may oscillate in the depth plane instead of maintaining a stable depth. This could lead to the AUV unintentionally hitting the riverbed, especially if your mission plan does not include a sufficient safety margin for vehicle depth.

In previous in-water operations, you reviewed logs to evaluate your vehicle's control performance. From this, you should have a general idea of how your AUV behaves in the depth plane; e.g., the extent of its depth fluctuations. Use this information to set safe target depths for your mission to reduce the risk of bottom contact.

4.6 Navigation performance

As you have learned from SITL simulations and by reviewing past logs, the AUV's navigation solution tends to drift over time while underwater; i.e., the vehicle may not actually be where it thinks it is. As a result, it could unintentionally move into a shallower area while assuming it remains in deeper water, increasing the risk of hitting the bottom. This is an important factor to consider when planning your mission.

4.7 Mid-mission re-surfacing

Similar to the risk of hitting the bottom, the AUV may unintentionally surface if the mission plan brings it too close to the water's surface. When operating near/at the surface, the vehicle is at risk of encountering other vessel traffic and may also fail to dive again, potentially compromising the mission. Therefore, it is important to include a sufficient safety margin when planning how close the AUV will operate to the surface.

4.8 Collaborative operations with multiple AUVs

While your team has only one vehicle, there are other teams in your class with their own AUVs. Using multiple AUVs to measure the spatial variability of the temperature-depth profile is far more efficient than relying on a single vehicle, especially given the limited in-water runtime for each team. By coordinating with your classmates and collaboratively planning your missions, you can significantly enhance the overall sensing coverage and data quality.

5 Mission Planning Guidelines, Tools and Tips

5.1 Updating class software

The instructors frequently update the `VECTORS`, `StackUxV`, `StackUxV-drivers`, and `missions-StackUxV` repositories. Therefore, you should always begin each lab by updating these repositories and rebuilding the software on the topside laptop, Raspberry Pi, and PocketBeagle computers.

5.1.1 Updating software on the topside laptop

While we use the `build_stackuxv` script to automatically update and rebuild the software on the embedded computers, we perform these steps manually on the topside laptop.

Update `VECTORS` by pulling the latest code of `StackUxV` and re-runing the `dependencies.sh` script:

```
$ cd ~/StackUxV/scripts/
$ git pull
$ sudo ./dependencies.sh
[sudo] password for sb-topside-4:
```

Re-build `StackUxV` with latest software:

```
$ cd ~/StackUxV/
$ ./build.sh
```

Update and rebuild StackUxV-drivers:

```
$ cd ~/StackUxV-drivers/
$ git pull
$ ./build.sh
```

5.1.2 Updating your git branch of missions-StackUxV with the latest class software

Before we merge the 2025_2s01 branch into your own branch, it is good practice to check the status of your branch and make sure there are no uncommitted local changes on your topside laptop:

```
$ git status
On branch <your-branch-name>
Your branch is up to date with ‘origin/<your-branch-name>’.

nothing to commit, working tree clean
```

In my case, the bee branch does not have any modified or untracked files, so I am ready to merge. However, if your branch contains untracked files or code modifications that have not been committed, you may choose to either destroy these changes using `$ git reset --hard` command or commit these changes.

Switch to 2025_2s01 branch and pull the latest code onto your local topside computer:

```
$ git checkout 2025_2s01
$ git pull origin 2025_2s01
```

Let's switch back to your own branch merge 2025_2s01 branch into your own branch

```
$ git checkout <your-branch-name>
$ git merge 2025_2s01
```

A terminal window might pop up with nano text editor, asking to the enter a comment about this merge. You may enter a comment and exit by pressing ctrl+x key. Then you should see an output similar to the follows:

Push the updated version of your branch to the git remote server, so the latest version of your branch can also be pulled on to your Raspberry Pi and PocketBeagle:

```
$ git push origin <your-branch-name>
```

5.1.3 Updating software on the embedded computers

Log into your AUV's Raspberry Pi and run the build_stackuxv program. You will be prompted to enter the passwords for the Raspberry Pi and PocketBeagle.

```
$ build_stackuxv
```

5.2 Creating a new cruise

For this deployment, you may choose to either create a new cruise or modify an existing one in your own branch. A partially completed template mission called `c_depth_profiling` is also available for use as a starting point, if desired. When creating or modifying a cruise, make sure to do so on your topside computer. Once you're confident in your changes, commit them and then pull the updates to your AUV.

To create your own cruise, duplicate the `missions-StackUxV/cruise/c_depth_profiling` directory and rename it to match your cruise's name. You can then modify the files within that directory as needed:

```
$ cd ~/missions-StackUxV/cruise/
$ cp -r c_depth_profiling/ <your-cruise-name>
```

As we discussed before, every mission configured using the `configure_cruise.sh` script has two primary associated files, located in the `missions-StackUxV/cruise/<your-cruise-name>` directory. For example, the `c_depth_profiling` cruise has the following two associated files:

1. `missions-StackUxV/cruise/c_depth_profiling/cruise.def`

2. `missions-StackUxV/cruise/c_depth_profiling/pMITFS_MissionScript.plug`

You will need to modify the contents of these two files to plan your operation. The `cruise.def` file primarily defines the overall mission runtime; i.e. the mission's start and end times (in seconds).

```
// Mission related configurations ---------------------------------------------------------
#define MISSION_START_TIME                  60
#define MISSION_END_TIME                   540
```

The `pMITFS_MissionScript.plug` file defines the mission profile. In this lab, you will carefully plan your own mission by rewriting the mission profile to suit your objectives. As an example, the `c_depth_profiling/pMITFS_MissionScript.plug` file contains only a single task configured.

```
// Configuring the mission profile
add_mission: DEPLOY_MODE=CONSTANT_COURSE, mission_start_second=60, heading=150, speed=0.8,
depth_sequence=1.5:30|2.0:30|2.5:30|2.0:30, duration_minutes=3
```

This mission profile commands the vehicle to begin the mission 60 seconds after launch and maintain a constant course with a desired heading of 150 degrees and a speed of 0.8 m/s for three minutes. Instead of holding a single depth, it executes a depth sequence: maintaining 1.5 m for 30 seconds, then transitioning to 2 m for another 30 seconds, followed by 2.5 m for 30 seconds, then back to 2 m for another 30 seconds – repeating this cycle throughout the mission.

5.3 Recommended behaviors for transiting in the x-y plane

In this section, we discuss the behaviors you can use in this deployment to control transitions in the x–y plane.

5.3.1 Constant course mode

We have used this simple mode in the `a_confidence` and `b_zigzag` cruises. Under the hood, it relies on constant heading, constant speed, and either constant depth or depth sequencing behaviors. You can include multiple constant course mode missions, as illustrated below:

```
add_mission: deploy_mode=constant_course, mission_start_second=60, heading=150,
speed=0.8, depth=1.5, duration_minutes=7

add_mission: deploy_mode=constant_course, mission_start_minute=3, heading=240,
speed=0.8, depth=1.5, duration_minutes=7
```

According to the mission profile above, the first mission (i.e., a 150-degree heading) will begin 60 seconds after the launch. The second line will update the mission parameters (i.e., changing the heading to 240 degrees) three minutes after launch. The total duration of the mission is 7 minutes.

5.3.2 Loitering mode

We can use the loitering mode to configure the AUV to loiter around a center point with a specified radius. Under the hood, this mode utilizes the MOOS-IvP loiter behavior—a behavior designed for transit to and repeated traversal of a set of waypoints forming a convex polygon https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=BHV.Loiter.

An example mission configuration for this mode is as follows:

```
add_mission: deploy_mode=LOITER, mission_start_minute=1,  radius=20, segments=6, speed=0.8,
loiter_here=false, center_x=50, center_y=-100,  depth=1.5, duration_minutes=10
```

According to this mission profile, the AUV will loiter around the center point (50, -100) for 10 minutes, starting 1 minute after the mission launch (see Figure 3). The loiter radius is set to 20 m. During this time, the AUV will attempt to maintain a constant depth of 1.5 m and a speed of 0.8 m/s.

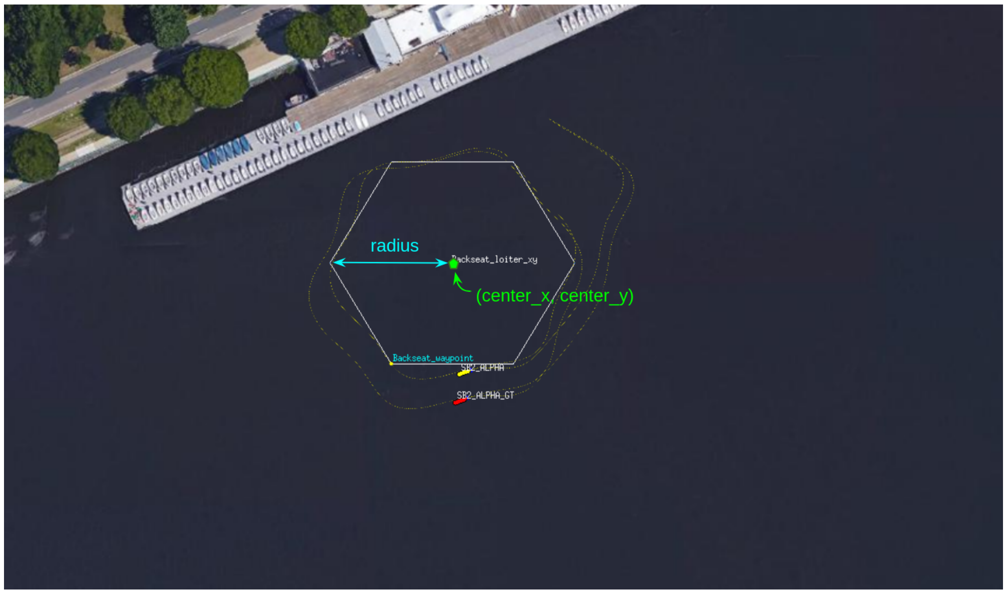

Figure 3: Parameters of the loitering mode

You can also mix and match these modes. For example, you might configure the vehicle to begin with a constant course mission at a heading of 150 degrees to move sufficiently away from the dock, and then transition into loitering mode. Note that instead of specifying a fixed center for the loiter, you can set `loiter_here=true`, which instructs the vehicle to use its current location as the center of the loiter pattern.

```
add_mission: deploy_mode=constant_course, mission_start_second=60, heading=150,
speed=0.8, depth=1.5, duration_minutes=8

add_mission: deploy_mode=LOITER, mission_start_minute=3,  radius=20, segments=6, speed=0.8,
loiter_here=true, depth=1.5, duration_minutes=8
```

5.3.3 Transit mode

In transit mode, the vehicle can be commanded to navigate to a specified X/Y position. Under the hood, this mode utilizes the MOOS-IvP waypoint behavior https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=BHV.Waypoint.

An example mission configuration for this mode is as follows:

```
add_mission: deploy_mode=TRANSIT, mission_start_minute=1,  transit_speed=0.8, local_x=150,
local_y=-100, depth=1.5, duration_minutes=8
```

In this mission profile, the vehicle will begin 1 minute after mission launch and transit to the point (150, -100) (see Figure 4).

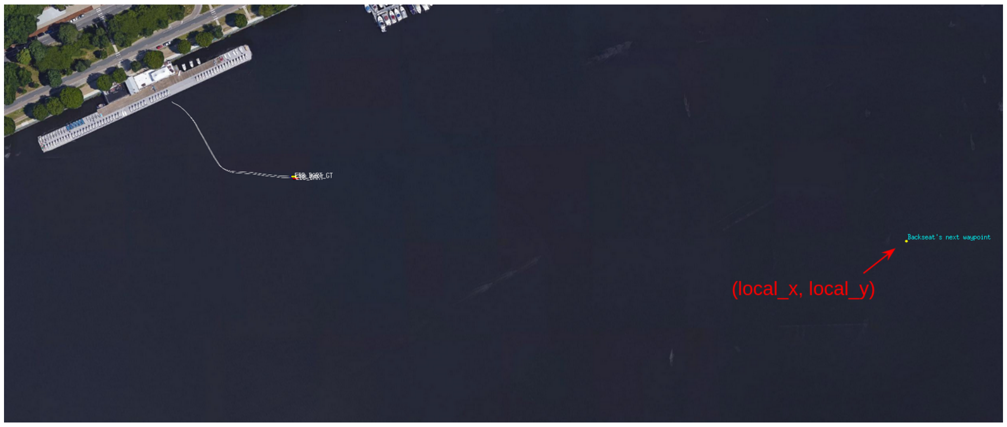

Figure 4: Parameters of the transit mode

You can configure multiple transit points, where the vehicle will go to the first waypoint and then proceed to the second, continuing in sequence as defined.

```
add_mission: deploy_mode=TRANSIT, mission_start_minute=1,  transit_speed=0.8, local_x=150,
local_y=-100, depth=1.5, duration_minutes=10

add_mission: deploy_mode=TRANSIT, mission_start_flag_key=MISSION_COMPLETE,
mission_start_flag_value=TRANSIT, transit_speed=0.8, local_x=205, local_y=-160, depth=1.5,
duration_minutes=10, mission_order=1

add_mission: deploy_mode=TRANSIT, mission_start_flag_key=MISSION_COMPLETE,
mission_start_flag_value=TRANSIT, transit_speed=0.8, local_x=100, local_y=-245, depth=1.5,
duration_minutes=10, mission_order=2
```

In the mission profile above, the vehicle will begin 1 minute after mission launch and transit to the first point at (150, -100). Once this waypoint is reached (i.e. under the hood, when the MOOS variable `MISSION_COMPLETE=TRANSIT` is posted) the vehicle will proceed to the second waypoint at (205, -160), and then continue to the third waypoint at (100, -245). Note that the `mission_order` parameter is used to distinguish between the second and third transit points, ensuring the correct sequence of operations.

5.4 Recommended behaviors for the depth plane

You can choose between two depth modes for your operations, and you are free to mix and match these modes across multiple legs in your mission profile.

5.4.1 Constant depth mode

This mode will drive the vehicle at a specified depth. Under the hood, this mode utilized MOOS-IvP constant depth behavior https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=BHV.ConstantDepth.

5.4.2 Depth sequence mode

In depth sequence mode, you can define a series of depth values for a single leg of a mission. The vehicle will maintain each specified depth for a given duration before transitioning to the next, as illustrated in Figure 5. Throughout this sequence, the vehicle continues to execute the same x–y behavior. Under the hood, this mode utilizes MOOS-IvP GoToDepth behavior https://oceanai.mit.edu/ivpman/pmwiki/pmwiki.php?n=BHV.GoToDepth

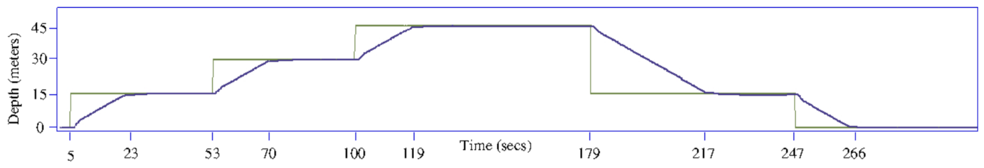

Figure 5: Parameters of the depth sequence mode

As an example, you may choose to conduct a loitering mission in which the vehicle oscillates between depths of 1.5 m and 2.5 m. The vehicle will remain at each depth for 30 seconds before transitioning to the next.

```
add_mission: deploy_mode=LOITER, mission_start_minute=1,  radius=20, segments=6, speed=0.8,
loiter_here=true, depth_sequence=1.5:30|2.5:30, duration_minutes=8
```

5.5 Virtual operations before actual operations!

When planning and configuring your own mission, the SITL simulation is your friend. You can use it to ensure that your mission behaves as expected. Once your mission is planned, you should review it with the instructors before in-water deployments. Be prepared to demonstrate a simulation of your planned mission to them.

5.6 Transferring code updates from the topside to embedded computers

Once you are satisfied with the cruise, commit it and push it to the Git remote server. Afterward, you can pull the updated cruise to your AUV's embedded computers.

5.7 Data post-processing

Once the deployment is complete, download the logs and analyze the temperature-depth variations. Is there a thermocline in the Charles River Basin in mid-May?