UNIVERSITY OF PELOPONNESE
NATIONAL OBSERVATORY OF ATHENS

MASTER THESIS

Advancements in monitoring the Mediterranean Sea with the use of Copernicus data

Delineation of Possible Fishing Zone

Author:
VYRON VASILEIADIS

Supervisor:
IPHIGENIA KERAMITSOGLOU

A thesis submitted in fulfillment of the requirements for the degree of

Master of Science in Space Science Technologies and Applications

October 24, 2017

University of Peloponnese
National Observatory of Athens

Abstract

Master of Science in Space Science Technologies and Applications

Advancements in monitoring the Mediterranean Sea with the use of Copernicus data

Delineation of Possible Fishing Zone

by VYRON VASILEIADIS

Potential Fishing Zone (PFZ) is a forecast of fish aggregation zones for a duration which is short (some hours to a couple of days). A fuzzy logic-based procedure, named Fish Alert, for generating daily PFZ maps of European anchovies and European sardines in the Mediterranean Sea is proposed and its implementation in Python is provided. Fish Alert uses bathymetric data provided by the General Bathymetric Chart of the Oceans (GEBCO) and daily data of Sea Surface Temperature (SST), Sea Level Anomaly (SLA) and Chlorophyll concentration (CHL) provided by Copernicus Marine Environment Monitoring Service (CMEMS). Far from perfect, Fish Alert provides a proof of concept that daily generation of PFZ maps for the two most important fishery of the Mediterranean Sea using free data is possible. An extended review of ocean monitoring techniques and important ocean parameters as well as general and hydrographic features of the Mediterranean Sea is also included.

Acknowledgements

First and foremost, i would like to congratulate all the researchers of the National Observatory of Athens and the professors of the University of Peloponnese who participated in the MSC Space Science Technologies and Applications for their innovative idea and effort to introduce such an informative and edifying subject.

Specifically, i would like to give special recognition to my supervisor and senior researcher of the National Observatory of Athens, Iphigenia Keramitsoglou, for her guidance and help during the researching and writing of this thesis.

A special thanks is reserved for my family, friends, partners, mentors and co-workers for supporting me during the last two years and making my life stimulating and meaningful.

Contents

Abstract iii

Acknowledgements v

List of Figures ix

List of Tables xi

Introduction 1

1 Ocean monitoring 3
1.1 Sensors for observing the ocean . . . 3
1.1.1 Visible wavelength radiometers . . . 3
1.1.2 Thermal-infrared radiometers . . . 4
1.1.3 Microwave radiometers . . . 5
1.1.4 Oblique-viewing radars . . . 5
1.1.5 Nadir-viewing radars . . . 6
1.2 Ocean parameters . . . 7
1.2.1 Chlorophyll . . . 7
1.2.2 Total Suspended Solids . . . 8
1.2.3 Colored Dissolved Organic Matters . . . 10
1.2.4 Sea Surface Temperature . . . 11
1.2.5 Sea Surface Height . . . 12
1.2.6 Salinity . . . 13
1.3 Producing ocean data products . . . 14
1.4 Sentinel-3 mission . . . 16
1.5 Copernicus Marine Environment Monitoring Service . . . 16

2 Area of Interest & Data 19
2.1 The Mediterranean Sea . . . 19
2.1.1 Area, depth and volume . . . 20
2.1.2 Meteorology . . . 20
2.1.3 Circulation pattern . . . 21
2.1.4 Ephemeral hydroghraphic features . . . 22
2.1.5 Persistent hydrographic features . . . 23
2.2 Data . . . 25

3 Delineation of Possible Fishing Zone 29
3.1 Overview . . . 29
3.2 Fishery Habitat . . . 29
3.2.1 Anchovy . . . 30
3.2.2 Sardine . . . 30
3.3 Data Processing . . . 30

3.4 Fuzzy Logic System 31

4 Results **37**

5 Conclusions **45**

A Fuzzy Logic **47**
A.1 Fuzzification 48
A.2 Fuzzy Inference Process 51
A.3 Defuzzification 52

B Source Code **53**
B.1 `facore` module 53
B.1.1 `__init.py__` 53
B.1.2 `downloader.py` 53
B.1.3 `collocator.py` 55
B.1.4 `fuzzifier.py` 57
B.1.5 `utilities.py` 76
B.2 `fishalert.py` 77

Bibliography **83**

List of Figures

1.1 Classification of satellite sensors used for viewing the ocean. 4
1.2 Schematic illustration of the measurement system of satellite altimetry. 6
1.3 The absorbtion spectrum of chlorophyll-a, chlorophyll-b and chlorophyll-c pigments. 8
1.4 Some sediment will settle to the bottom of a body of water, while others remain suspended. 9
1.5 Effect of temperature in photosynethesis. 11
1.6 The most common ions in sea water. 13
1.7 An outline of the procedures required to convert raw satellite data into ocean data products. 14

2.1 Simplified scheme of the Mediterranean's general circulation for surface water. 21
2.2 Simplified scheme of Levantine Intermediate Water (LIW) formation site and dispersal pathways. 22
2.3 Schematic representation of eddy dynamics. 23
2.4 Gyres generated during winter by wind stress of Mistral along the longitudinal axis of the Mediterranean Sea. 24
2.5 Schematic representation of a front structure. 24
2.6 Seasonal variation of wind-driven upwelling and downwelling zones in the Mediterranean Sea. 25
2.7 SST of Mediterranean Sea on 14-09-2017 as provided by CMEMS. . . . 26
2.8 SLA of Mediterranean Sea on 14-09-2017 as provided by CMEMS. . . . 27
2.9 SLA of Mediterranean Sea on 14-09-2017 as provided by CMEMS. . . . 27
2.10 GEBCO_2014 Grid continuous terrain model for the Mediterranean Sea. 28
2.11 Bathymetry of the Mediterranean Sea. 28

3.1 Overall workflow of PFZ generation procedure. 29
3.2 Original and collocated SLA data of the region south of Sardinia. . . . 32
3.3 Collocated SST, CHL, SLA data of the Mediterranean Sea on 15-09-2017. 33
3.4 Membership function of PFZ output variable. 34
3.5 Membership function of SST input variable. 34

4.1 3D visualization of the PFZ values for Anchovies on Winter in relation to CHL, SST values when depth is ideal. 38
4.2 PFZ map of anchovy for 14-09-2017 generated by Fish Alert procedure. 39
4.3 PFZ map of sardine for 14-09-2017 generated by Fish Alert procedure. 39
4.4 PFZ map of anchovies for 14-09-2017 with values greater than $60\,\%$. . . 40
4.5 PFZ map of sardines for 14-09-2017 with values greater than $60\,\%$. . . . 40
4.6 PFZ maps of anchovies for the week 14 - 20 September 2017. 42
4.7 PFZ maps of sardines for the week 14 - 20 September 2017. 43

A.1 Common membership functions. 49

A.2 Special cases of the trapezoidal membership function. 49
A.3 Membership functions of humidity level input variable. 50
A.4 Membership functions of motor speed output variable. 50
A.5 Different defuzzication methods of a trapezoidal membership function. 52

List of Tables

A.1 Fuzzy control rules for air dehumidifier system using two inputs. . . . 51

"You are not a drop in the ocean. You are the entire ocean in a drop."

– Jalaluddin Rumi

Introduction

Studies show that at least $2.7\,\text{billion}$ people are living in coastal zones globally [1] [2], and about $74\,\text{million}$ are living in EU coastal regions of the Mediterranean Sea [3]. Part of these populations are depending heavily on fishing for their livelihood. Locating and catching fish is always a challenging task. Often, the search for fish ends up in spending considerable time and resources, thus increasing the cost leading to low profitability.

Small pelagic fish are known to play a key ecological role in coastal ecosystems, transferring energy from plankton to upper trophic levels [4]. European anchovy (Engraulis encrasicolus) along with European sardine (Sardina pilchardus) make up the bulk of small pelagic fish catches in the Mediterranean Sea [5].

Their relatively low position in the marine food web, together with their short lifespan and their reproductive strategy of producing large quantities of pelagic eggs over an extended spawning season, makes their populations strongly dependent on the environmental conditions [6].

Potential Fishing Zone (PFZ) is a forecast of fish aggregation zones for a duration which is short (some hours to a couple of days). The forecast is done mainly by considering certain ocean parameters, such as sea level height, sea surface temperature, chlorophyll-a concentration and water depth.

The most mature operational use of PFZ maps originates in India. Remote sensing of small pelagic fishery resources was initiated in India in early nineties with collaborative efforts of the Indian Space Research Organization, Fishery Survey of India and Central Fisheries Research Institute. Presently, Ministry of Earth Sciences is providing advisories on Potential Fishing Zone for entire Indian coast. These multi-lingual PFZ advisories are disseminated via various modes like telephone, fax, e-mail, website, radio, news media, mobile services and mobile applications, and under funded projects to universities and partner NGOs. The feedback obtained from researchers working on various extensive validation projects as well as the feedback collected from fishermen indicated that the catch in the PFZ area is substantially higher when compared to the other areas [7]. PFZ advisories were found more beneficial to artisanal, motorised and small mechanised sector fishermen engaged in pelagic fishing activities such as ring seining, gill netting etc., thereby reducing the searching time which in turn result in the saving of valuable fuel oil and also human effort.

To the best of author's knowledge, no similar service has been developed in the area of the Mediterranean Sea. This research tries to fill this gap by proposing a fuzzy logic based procedure, named Fish Alert, for generating Potential Fishing Zone maps of anchovies and sardines in the Mediterranean Sea. The procedure is based on the existence of previous studies on the potential habitat of the specific fishery in relation to environmental conditions, specifically the depth of waters, Sea

Surface Temperature (SST), Sea Level Anomaly (SLA) and Chlorophyll concentration (CHL). These environmental parameters are obtained daily using remote sensing techniques.

Chapter 1

Ocean monitoring

1.1 Sensors for observing the ocean

Ocean remote sensing methods use three different sources of electromagnetic radiation (EMR) to communicate information about the sea from its surface to a sensor in Earth orbit above the atmosphere [8]. These are the following:

1. solar radiation reflected at or below the sea surface in the visible and near-infrared wavelengths of the EMR spectrum,
2. thermal radiation emitted by the sea surface in both the thermal infrared and the microwave
3. pulses of microwave radiation generated on the satellite, emitted toward the Earth, and then reflected by the sea surface.

Passive sensors use 1 and 2, whereas active sensors are needed to use 3.

Passive sensors operate in the three principal spectral windows through which EMR passes through the atmosphere, that is, the visible and near-infrared waveband, the thermal infrared, and the microwave.

Active sensors on satellites all operate in the microwave and so are types of radars, devices which supply their own energy, in the form of radar pulses emitted from the spacecraft, reflected from the sea surface, and received back at the sensor again. A major advantage is that the echo pulse can be compared to the emitted pulse in order to detect changes in its amplitude, shape, phase, or frequency, each of which may be used to derive particular information about the sea surface from which it was reflected. The effect of the intervening atmosphere can also be largely, though not entirely, ignored. There are two class of radar sensors, oblique-view and nadir-view radars. So far no lidars, active sensors operating in the visible waveband, have been used for ocean remote sensing from satellites.

Figure 1.1 shows the classification of the main sensors and methods of ocean remote sensing by the part of the spectrum they used and whether they are active or passive.

1.1.1 Visible wavelength radiometers

Visible and near-infrared waveband radiometers measure radiance at the top of the atmosphere (TOA), sampled in discrete spectral bands across the visible and near-infrared in order to describe its spectral composition or *color*; hence, these instruments are

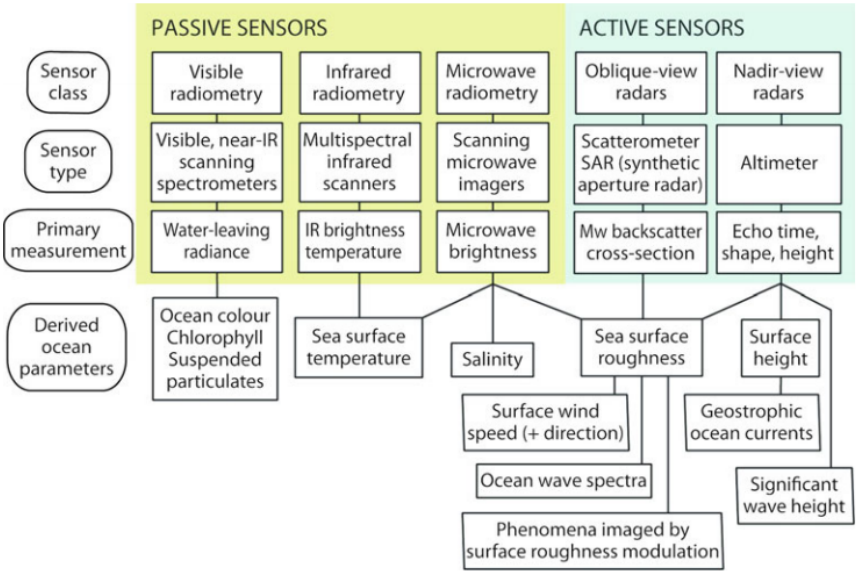


Figure 1.1 Schematic showing the different classes and types of satellite sensors used for viewing the ocean, the primary measurement that they make, and some of the ocean properties that are derived.

typically referred to as *ocean color* sensors. The primary variable delivered by an ocean color radiometer system is the water-leaving spectral radiance. Atmospheric correction [9] is a major component of the primary processing required to retrieve the ocean color (i.e., the water-leaving radiance). The near-infrared channels are essential for this atmospheric correction procedure. As there is almost no component of near-IR in the water-leaving radiance, any detected in the TOA signal provides a measure of the amount of atmospheric path radiance that is present.

Ocean color in itself is not of particular interest, but it is influenced by a number of different components which determine the water's inherent optical properties. Most important of these is the concentration of the pigment chlorophyll-a associated with the presence of phytoplankton and with primary production. Chlorophyll-a concentration is the main ocean variable derived from ocean color sensors, although other factors also influence the color such as dissolved organic matter and suspended solids.

1.1.2 Thermal-infrared radiometers

Thermal-infrared radiometers, measures the thermal emission from the sea surface skin (approximately the top $10\,\mathrm{mm}$), after it has passed through the atmosphere and been slightly absorbed by water vapor and other greenhouse gases. The primary measurement is the TOA brightness temperature (the temperature of a black body emitting the same radiance as that measured by the radiometer), and the main derived ocean variable is the sea surface temperature (SST) discussed in Section 1.2.4. To retrieve this requires atmospheric correction, which relies on sampling multiple spectral bands across the TIR, since the water vapor absorption is wavelength dependent [10]. Stratospheric aerosols also scatter and absorb the radiation, but these are less reliably detected and one of the current challenges in the field is to safeguard the

absolute accuracy and long-term stability of the SST climate record even during adverse conditions of stratospheric aerosols following major volcanic eruptions.

1.1.3 Microwave radiometers

Microwave radiometers detect the top of atmosphere brightness temperature in a number of different microwave bands and obtain additional independent information by separately measuring the horizontal and vertical polarized radiation. Microwave emission from the sea is more complex than for infrared [11] [12]. While the blackbody emission is linear with absolute temperature, the emissivity is less than 0.5 and varies with the viewing angle and with the dielectric properties of seawater which depend on temperature and salinity. Consequently the actual dependence of the brightness temperature on the SST is very complex and requires information about the surface roughness and dielectric properties. The positive outcome of this complex behavior is that the additional dependencies provide the opportunity to retrieve wind speed and direction and salinity as well as temperature from microwave radiometry.

1.1.4 Oblique-viewing radars

When the emitted pulse of an oblique viewing radar encounters a smooth sea surface, most of its energy is reflected by specular reflection away from the sensor. If the surface is roughened, the incident energy is scattered more diffusely and with increasing roughness more energy is backscattered to the radar. While it is difficult to describe this behavior precisely [12], the main mechanism for backscatter at incidence angles between 20° and 70° is based on Bragg scattering theory. The primary measured quantity is the radar backscatter cross section. This is the ratio between the returned and emitted signal, normalized for geometrical spreading; so, it changes only with the change in surface roughness and is usually denoted with symbol s_0. Depending on the radar frequency and its viewing angle, the Bragg wavelength is typically in the range between $1\,\mathrm{cm}$ and $30\,\mathrm{cm}$. These are the small ripples that are produced by wind stress on the sea surface, and so oblique viewing radars are very sensitive to wind conditions over the sea.

There are two special types of oblique-viewing radars used for ocean remote sensing, scatterometers, and synthetic aperture radars, normally referred to as SARs.

A scatterometer exploits the above procedure in order to serve as a wind-monitoring instrument. It measures the average backscatter from a fairly wide region. The backscatter is compared with an empirical database relating s_0 to wind speed and direction [13]. A scatterometer is designed so that the same patch of sea is viewed from more than one direction within a few minutes, which enables both wind speed and direction to be retrieved, with a typical spatial resolution of $50\,\mathrm{km}$.

A SAR samples the return pulse in very fine detail and applies signal processing techniques to map at a spatial resolution of about $20\,\mathrm{m}$, using the echoes from several hundred pulses to contribute to defining the magnitude of s_0 at each pixel on the reconstructed image. The magnitude of s_0 across a SAR image is set mainly by the spatial distribution of wind speed and direction, so a SAR image can map small-scale variability of the wind field. However, local convergence, divergence, or shear in the sea surface currents causes modulation of the wind-driven Bragg ripples. This

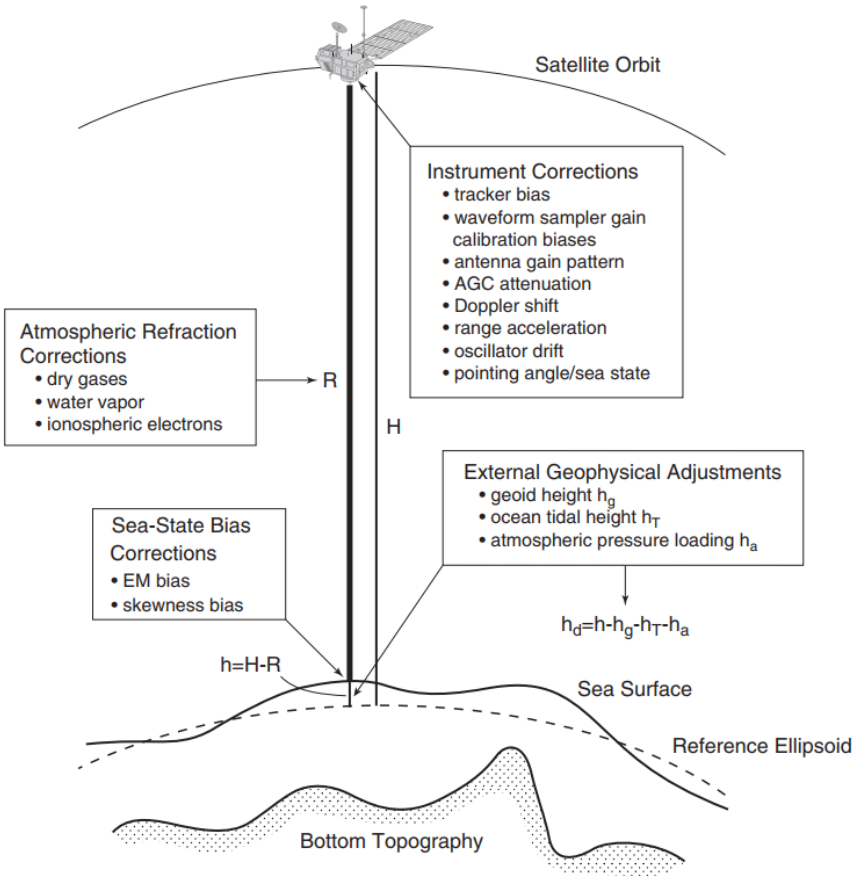


Figure 1.2 Schematic illustration of the measurement system of satellite altimetry.

allows signatures of the underlying small-scale ocean dynamics to be created as patterns on the image. A variety of features have been revealed by this means, most notably internal waves, surface swell waves, bathymetric features in shallow tidal seas, ocean fronts, and small-scale eddies [14] [15] [8].

1.1.5 Nadir-viewing radars

A nadir-viewing radar, also called *altimeter* views the surface at normal incidence using a nadir-pointing antenna. Its primary purpose is to measure the travel time of a radar pulse reflected from the sea surface and hence the distance between the altimeter and the sea surface. Precise knowledge of the satellite's position relative to the center of the Earth then allows the absolute height of the sea surface to be determined. As it travels along its orbit above the Earth, an altimeter can therefore measure the gradual along-track slope of the sea surface, which is another of the primary observable quantities of satellite oceanography. The slope of the sea surface over a distance of tens to hundreds of kilometers is an extremely important measurement since the surface geostrophic flow of the ocean can be derived from it.

The challenge of determining sea surface height from satellite altimetry is the measurement accuracy. For example, an error of $1\,\mathrm{cm}$ in altimetry is capable of producing an error in the water mass transport of ocean currents of the magnitude of several million tons per second, which is a significant fraction of the transport of

major ocean currents. The sources for these measurement errors are numerous as depicted in Figure 1.2 and the techniques for correcting them are described by Chelton et al. [16]. Nowadays, the most successful altimeters can measure height with a total range error of about 3 cm.

1.2 Ocean parameters

1.2.1 Chlorophyll

Phytoplankton are microorganisms that drift about in water. They are single-celled, but at times they can grow in colonies large enough to be seen by the human eye [17]. Phytoplankton are photosynthetic, meaning they have the ability to use sunlight to convert carbon dioxide and water into energy [18]. While they are plant-like in this ability, phytoplankton are not plants. The term “single-celled plants” is a misnomer, and should not be used. Instead, phytoplankton can be divided into two classes, algae and cyanobacteria [19]. These two classes have the common ability of photosynthesis, but have different physical structures. Regardless of their taxonomy, all phytoplankton contain at least one form of chlorophyll (chlorophyll-a) and thus can conduct photosynthesis for energy.

Phytoplankton, both algae and cyanobacteria, can be found in fresh or saltwater [20]. As they need light to photosynthesize, phytoplankton in any environment will float near the top of the water, where sunlight reaches [19]. Most freshwater phytoplankton are made up of green algae and cyanobacteria, also known as blue-green algae [20]. Marine phytoplankton are mainly comprised of microalgae known as dinoflagellates and diatoms, though other algae and cyanobacteria can be present. Dinoflagellates have some autonomous movement due to their "tail" (flagella), but diatoms are at the mercy of the ocean currents [21].

Chlorophyll is a color pigment found in plants, algae and phytoplankton. This molecule is used in photosynthesis, as a photoreceptor. Photoreceptors absorb light energy, and chlorophyll specifically absorbs energy from sunlight. Chlorophyll makes plants and algae appear green because it reflects the green wavelengths found in sunlight, while absorbing all other colors.

However, chlorophyll is not actually a single molecule. There are 6 different chlorophylls that have been identified [22]. The different forms (a, b, c, d, e and f) each reflect slightly different ranges of green wavelengths. Chlorophyll-a is the primary molecule responsible for photosynthesis. That means that chlorophyll-a is found in every single photosynthesizing organism, from land plants to algae and cyanobacteria. The additional chlorophyll forms are accessory pigments, and are associated with different groups of plants and algae and play a role in their taxonomic confusion. These other chlorophylls still absorb sunlight, and thus assist in photosynthesis. As accessory pigments, they transfer any energy that they absorb to the primary chlorophyll-a instead of directly participating in the process.

Chlorophyll-b is mainly found in land plants, aquatic plants and green algae. In most of these organisms, the ratio of chlorophyll-a to chlorophyll-b is 3:1 [23]. Chlorophyll-c is found in red algae, brown algae, and dinoflagellates [24]. Chlorophyll-d is a minor pigment found in some red algae, while the rare Chlorophyll-e has been found

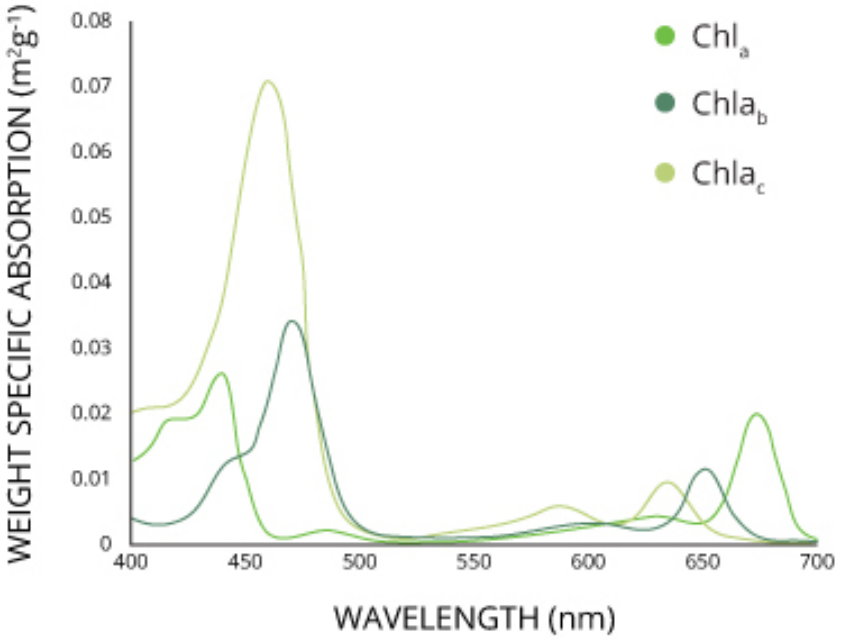


Figure 1.3 The absorbtion spectrum of chlorophyll-a, chlorophyll-b and chlorophyll-c pigments. The different forms of chlorophyll absorb slightly different wavelengths for more efficient photosynthesis.

in yellow-green algae [25]. Chlorophyll-f was recently discovered in some cyanobacteria near Australia. Each of these accessory pigments will strongly absorb different wavelengths, so their presence makes photosynthesis more efficient. This is depicted in Figure 1.3.

Various visual spectral bands and their ratios are widely used to quantify chlorophyll-a (chl-a). Spectral band ratios can reduce irradiance, atmospheric and air-water surface inuences in the remotely sensed signal [26]. The prominent scattering-absorption features of chl-a include strong absorption between 450 - $475\,\mathrm{nm}$ (blue) and at $670\,\mathrm{nm}$ (red), and reectance reaches to peak at $550\,\mathrm{nm}$ (green) and near $700\,\mathrm{nm}$ (NIR). The reectance peak near $700\,\mathrm{nm}$ and its ratio to the reectance at $670\,\mathrm{nm}$ have been used to develop a variety of algorithms to retrieve chl-a in turbid waters [27]. Gitelson [28] studied the behavior of the reectance peak near $700\,\mathrm{nm}$ and concluded that the $700\,\mathrm{nm}$ reectance peak was important for the remote sensing of inland and coastal waters, especially for measuring chlorophyll concentration. Han [29] pointed out that the spectral regions at 630 - $645\,\mathrm{nm}$, 660 - $670\,\mathrm{nm}$, 680 - $687\,\mathrm{nm}$ and 700 - $735\,\mathrm{nm}$ were found to be potential regions where the first derivatives can be used to estimate chlorophyll concentration. Dekker et al. [30] mentioned that the scattering and absorption characteristics of chl-a can be studied when more than one band is used.

1.2.2 Total Suspended Solids

Turbidity is an optical determination of water clarity. Turbidity scatters and absorbs the light rather than transmit it in straight lines, making turbid water appear cloudy, murky, or otherwise colored. Suspended solids are responsible for most of the scattering, whereas the absorption is controlled by chl-a and colored dissolved matter [31].

Suspended solids (or particulate matter) are particles that are larger than $2\,\mu m$ found in the water column, that is they are not settled or bed load [1] as seen in Figure 1.4. Anything smaller than $2\,\mu m$ (average filter size) is considered a dissolved solid. Most suspended solids are made up of inorganic materials, though bacteria and algae can also contribute to the total solids concentration. These solids include anything drifting or floating in the water, from sediment, silt,and sand to plankton and algae. Organic particles from decomposing materials can also contribute to the total suspended solids (TSS) concentration. As algae, plants and animals decay, the decomposition process allows small organic particles to break away and enter the water column as suspended solids. Even chemical precipitates are considered a form of suspended solids.

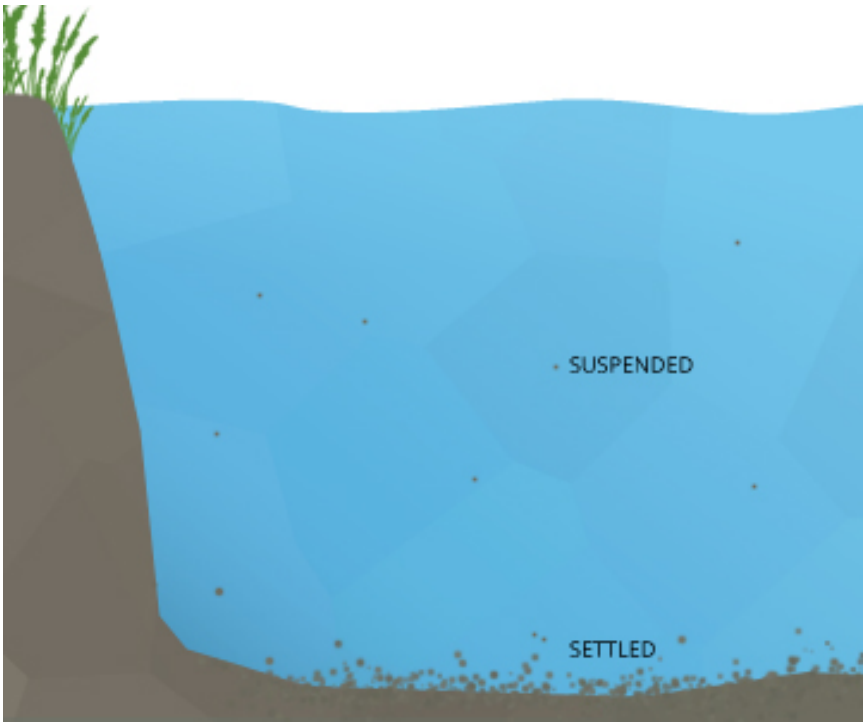


Figure 1.4 Some sediment will settle to the bottom of a body of water, while others remain suspended.

As water turbidity is mainly the result of the presence of suspended matter, turbidity measurement has often been used to calculate uvial suspended solids concentrations [32] and is commonly regarded as the opposite of clarity. The level of turbidity or murkiness is entirely dependent on the amount of suspended particles in a sample of water. The more suspended particles, the more difficult for light to travel through the water and therefore, the higher the water's turbidity. The complex nature of suspended substances in water changes the reectance of the water body and there fore causes variation in color. To this end, interpretation of remotely sensed data just based on the color of water is not adequate and accurate. Turbidity and total suspended matters are considered as important variables in many studies due to their linkage with incoming sunlight that in turn affects photosynthesis for growth of algae and plankton.

Remote sensing techniques are widely used to estimate and map the turbidity and concentrations of suspended particles, and to provide their spatial and temporal

[1] Sediments that *roll* along the sea floor.

variations. Theory shows that use of a single band provides a robust algorithm provided the band is chosen appropriately [33]. Curran et al. [34] and Novo et al. [35] showed that single band algorithms maybe adopted where TSS increases with increasing reectance. However, the complex substances in water change the reectance of the water body and therefore cause variation in colors, and thus, different spectral bands can be used for TSS retrievals [36]. Therefore, the advantage of using signal band or band ratios can be employed to obtain more accurate results in different concentrations in water bodies. In the Near-IR and Mid-IR regions, water increasingly absorbs the light and makes it look darker, which varies based on water depth and wavelength. An increase of dissolved inorganic materials in waterbodies causes the peak of visible reectance to shift from the green region (clearer water) toward the red region of the spectrum. The literature has shown that turbidity and TSS can be measured using various band ratios such as:

1. Ratio between green (500 - 600 nm) and red (600 - 700 nm) [37] [38]
2. Ratio between blue (400 - 500 nm) and red (600 - 700 nm) [39] bands [40]
3. Ratio between near infrared (NIR) and red (600 - 700 nm) bands [41] [42]

1.2.3 Colored Dissolved Organic Matters

Colored Dissolved Organic Matters (CDOM) consists of naturally occurring, water-soluble, biogenic, heterogeneous organic substances that are yellow to brown in color [43], which exist in both fresh and saline waters. These compounds are brown and can color the water yellowish brown in high concentrations. Therefore, they are also referred to as yellow matter. CDMOM together with chl-a and TSS dominate the water color.

CDOM absorbance spectrum can be several times and overlaps the chlorophyll absorption and can account for over 50 % of the total absorption at 443 nm, which is the wavelength that chlorophyll concentrations are usually measured [44]. The increase in the CDOM concentration mainly affects the reflectance values in the blue and green region of the spectrum (especially below 500 nm) and its absorbance increases exponentially with decreasing wavelength. This effect can complicate the use of chl-a retrieval algorithms and phytoplankton production models. Nonetheless, it is reported by Strömbeck and Pierson [45] that at high CDOM concentrations, absorbance of red light spectrum can be significant.

The literature showed that CDOM could be quantified using visual spectral bands and their ratios such as:

1. Single blue band (400 - 500 nm) [46] [47] [48]
2. Ratio between blue (400 - 500 nm and green (500 - 600 nm) bands [49] [50]
3. Ratio between green (500 - 600 nm) and red (600 - 700 nm) bands [51]

Furthermore, Taheri Shahraiyni et al. [52] mapped the spatial distribution of CDOM over the southern parts of the Caspian Sea by using reflectance values at 490, 510, 560, 620, and 885 nm of MERIS data and applying a fuzzy modeling technique called Active Learning Method (ALM).

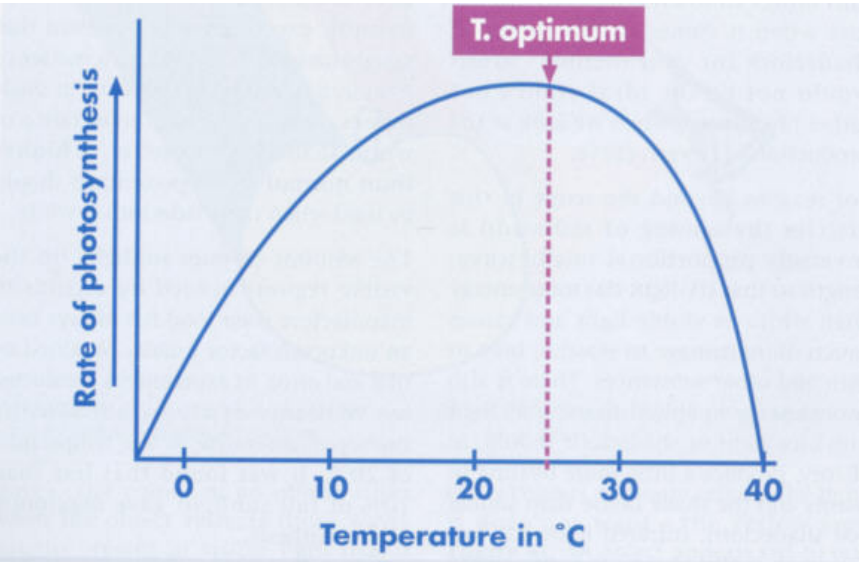


Figure 1.5 Effect of temperature in photosynethesis. T_o is the optimal temperature for photosynthesis, and depends on latitude and the specific plant's characteristics.

1.2.4 Sea Surface Temperature

Water temperature is a physical property describing how hot or cold water is. As hot and cold are both arbitrary terms, temperature can also be characterized as a measurement of the average thermal energy of a substance [53]. Thermal energy is the kinetic energy of molecules and atoms, so temperature in turn measures the average kinetic energy of the molecules and atoms [53]. This energy is transferrable between substances as the flow of heat. Heat transfer, whether from another water source, air, sunlight, or thermal pollution can change the temperature of water.

Water temperature has long been defined as the *abiotic master factor* [54] due to its impact on aquatic organisms.

Temperature fluctuations can affect the behavior choices of aquatic organisms, such as moving to warmer or cooler water after feeding, predator-prey responses and resting or migrating routines [55]. Some species of sharks and stingrays will even seek out warmer waters when pregnant [55]. Plants are also affected by water temperature. Although some aquatic plants bear with cooler waters, most prefer warmer temperatures [56]. Tropical plants in particular will show restricted growth in water temperatures below $21\,^{\circ}\mathrm{C}$ [56]. Temperature can also restrain plant respiration and photosynthesis [57]. In general, algal photosynthesis will increase with temperature, though different species will have different peak temperatures for optimum photosynthetic activity [57]. As Figure 1.5 shows, above and below the optimal temperature T_o, photosynthesis will be reduced.

Stratification is the division of a water column into layers of water with different properties. These divisions are usually defined by temperature and density, though other parameters such as salinity and chemical distinctions can also be used [58]. Stratification occurs because force (usually currents) is required to mix liquids of different densities [57]. Thermal stratification is usually seasonal, with clear delineations between layers during the summer, narrower layers in winter, and a

"turnover" in the spring and fall when temperature is fairly uniform throughout the water column [59].

Sea surface temperature (SST) is the water temperature near the ocean's surface. The exact meaning of surface varies according to the measurement method used, but it is usually between $1\,\mathrm{mm}$ and $20\,\mathrm{m}$ below the sea surface. Air masses in the Earth's atmosphere are highly altered by sea surface temperatures within a short distance of the seashore. Warm sea surface temperatures are known to be a cause of tropical cyclogenesis over the Earth's oceans. Tropical cyclones can also cause a cool wake, due to turbulent mixing of the upper $30\,\mathrm{m}$ of the ocean.

SST changes diurnally, like the air above it, but to a lesser degree. There is less SST variation on breezy days than on calm days. In addition, ocean currents such as the Atlantic Multidecadal Oscillation (AMO), can effect SST's on multi-decadal time scales [60], a major impact results from the global thermohaline circulation, which affects average SST significantly throughout most of the world's oceans.

SST may be the best-known ocean parameter on global scales [61].

The emitted thermal-infrared radiation (TIR) at $3 - 14\,\mu m$ is a well-established practice of determining ocean SST [62] [63]. In the terrestrial environment, TIR remote sensing of surface water temperature initially focused on lakes [64], and coastal applications such as thermal pollution from cooling water discharge from a power plant [65]. Currently, many TIR imaging sensors are available that have multiple spectral bands located at different wavelengths, which make them suitable for water temperature measurements. For the selection of appropriate band or bands, careful consideration on the least amount of instrument noise and atmospheric effects is necessary for accurate calculation of the water temperature. However, multiple bands can be averaged to reduce noise due to atmospheric or sensors differences and provide a better estimate of the actual temperature [66].

Thermal-infrared radiometers yield SST to around 0.5^oC precision, though its use is limited in shady zones due to the presence of clouds or fog. Therefore, standard remote sensing practices should be applied to identify and mask these issues out of the used images before one proceeds with the measurement of the water temperature by TIR radiation. Passive microwave techniques are used in cloudy areas with an accuracy limit of about $1.5 - 2^o$C by the relatively large variation of microwave emissivity with surface conditions, such as wind speed [67]. Addition of active microwave (radar) observations can enhance the precision of passive microwave estimates of SST.

1.2.5 Sea Surface Height

The sea surface is anything but flat. There are bumps and troughs , all due to different physical characteristics such as gravity, currents, temperature and salinity.

The geoid is defined as the ocean surface on which the Earth's gravity field is uniform. In the absence of any motions, the ocean surface would be conformed to the geoid.

The Sea Surface Height Anomaly (SLA) is the deviation of the vertical position of the sea surface relative to a known reference surface. Ideally, this reference surface is the oceanic geoid, but in practice it is commonly the mean of the sea surface height

variations relative to an arbitrary reference ellipsoid computed during a specified time period. In this case the reference surface includes the geoidal height and the mean topography associated primarily with the geostrophically balanced portion of the steady ocean currents.

1.2.6 Salinity

Salinity is an ambiguous term. As a basic definition, salinity is the total concentration of all dissolved salts in water [22]. These electrolytes form ionic particles as they dissolve, each with a positive and negative charge. As such, salinity is a strong contributor to conductivity. While salinity can be measured by a complete chemical analysis, this method is difficult and time consuming [68]. Seawater cannot simply be evaporated to a dry salt mass measurement as chlorides are lost during the process.

More often, salinity is not measured directly, but is instead derived from the conductivity measurement. This is known as *practical salinity*. These derivations compare the specific conductance of the sample to a salinity standard such as seawater. Salinity measurements based on conductivity values are unitless, but are often followed by the notation of practical salinity units (psu) [69].

There are many different dissolved salts that contribute to the salinity of water. The major ions in seawater (with a practical salinity of 35) are: chloride, sodium, magnesium, sulfate, calcium, potassium, bicarbonate and bromine. Many of these ions are also present in freshwater sources, but in much smaller amounts [69]. The ionic compositions of inland water sources are dependent on the surrounding environment. Most lakes and rivers have alkali and alkaline earth metal salts, with calcium, magnesium, sodium, carbonates and chlorides making up a high percentage of the ionic composition. Freshwater usually has a higher bicarbonate ratio while seawater has greater sodium and chloride concentrations [70].

Figure 1.6 The most common ions in sea water.

Even from 1970, spaceborne and airborne experiments proved the potential of passive L-band microwave radiometers for the measurement of sea surface salinity. Howerver, as the measurement of surface salinity by passive microwave radiometers requires long wavelengths (20 - 30 cm), accurate estimation of surface salinity from satellite altitudes would require an enormous antenna, which most satellites could not accommodate [71]. Active interferometric technology has made it possible to overcome such problems with antenna size [72]. For instance, the Moisture and Ocean Salinity satellite (SMOS) carrying the Microwave Imaging Radiometer using Aperture Synthesis (MIRAS) has been in use to measure surface salinity and provide synthesized maps with a high accuracy.

Many researchers have also used indirect methods by extracting salinity from other ocean parameters measurements and developed relationships, such as:

1. Relationship between salinity and temperature [73] [74] [75]; and

2. Relationship between salinity and CDOM [76] [77].

1.3 Producing ocean data products

Given the differences between the diverse methodologies identified in Section 1.1, the details of the processing chain to convert raw data into a useful data product differ considerably from sensor to sensor. Nonetheless, there is a generic set of tasks that need to be performed on data as part of the so-called ground segment of each Earth observation satellite mission, as illustrated in Figure 1.7. The *level* assigned to each data product relates to the stage of processing that has been reached.

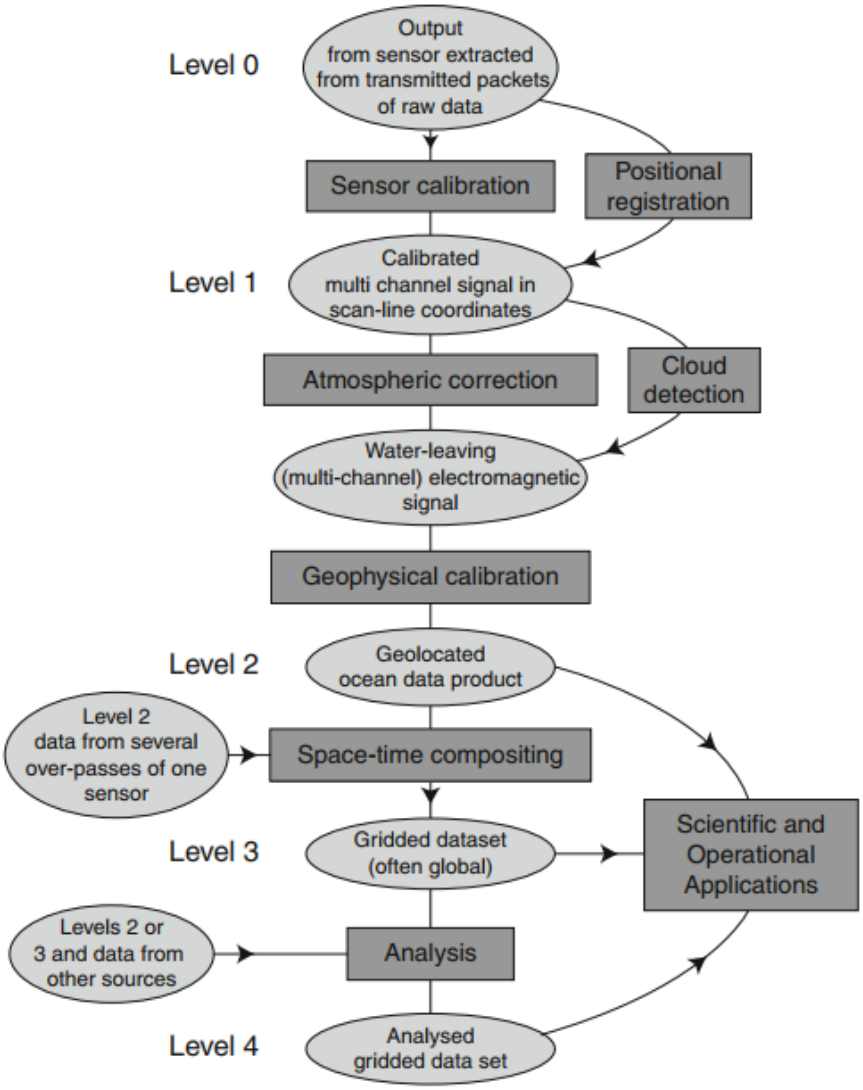


Figure 1.7 An outline of the procedures required to convert raw satellite data into ocean data products ready for use in ocean science and marine related applications.

Level 0 products are the uprocessed instrument and payload raw data at full resolution with any communications artifacts (e.g. synchronization frames, communications headers) removed. Since there are some fundamental corrections and calibrations that should be applied to the data before they are usable most agencies do not distribute Level 0 products.

Level 1 is usually divided into two distinct levels; Level 1A and Level 1B. Level 1A products are reconstructed instrument data at full resolution that are time-referenced and annotated with ancillary information, such as radiometric and geometric calibration coefficients and georeferencing parameters, that are computed and appended but not applied to Level 0 data. Level 1B are Level 1A data that have been processed to sensor units.

Users requiring data containing oceanographically useful products are advised to not use anything below Level 2. At this level the data will be geographically defined, but in the native sampling grid of the sensor (aligned along and across the satellite track rather than north–south), and be at the finest resolution possible. It will have had atmospheric corrections applied and will consist of useful ocean parameters.

The extraction of ocean parameters is based on a variety of algorithms depending on the canditate parameter and the sensor used. Known categories of algorithms include reflectance-based classification, spectral band-ratios, spectral band-difference and bio-optical models. Specifically for the extraction ocean color parameters it is worth noted that an additional step is required; the classification of water type of the area of interest, so the appropriate algorithm will be applied.

The terminology for Case 1 and Case 2 water classifications were established by Morel and Prieur [78] and Gordon and Morel [79] in their seminal work on the bio-optical basis for ocean color variations. The definition for Case 1 and Case 2 waters was updated by Mobley et al. [80] as follows:

1. Case 1 waters are those waters whose optical properties are determined primarily by phytoplankton and related colored dissolved organic matter (CDOM) and detritus degradation products.
2. Case 2 waters are everything else, namely waters whose optical properties are significantly inuenced by other constituents such as mineral particles, CDOM, or microbubbles.

Level 3 products will be resampled onto a regular geographical grid and may well be composed from several overpasses of the same sensor.

Level 4 products result from the analysis of data from sources beyond a single satellite and may include in situ data and the products of models.

As operational applications of satellite data products have matured, it is desirable for derived ocean variables to be accompanied by error estimates or values that allow the quality of the information in each pixel to be compared with other pixels in the same image and with other image products. For example, the cloud detection, atmospheric correction, geophysical calibration or ocean variable extraction processes may reach a more confident result for one pixel than another, as a result of which the user may wish to attach more weight to the influence of one pixel than another in driving a model or making an operational decision. Modern missions such as the Sentinel satellites have fully established this approach.

1.4 Sentinel-3 mission

The Sentinels are a fleet of satellites designed to deliver the wealth of data and imagery that are central to the European Commission's Copernicus programme.

With a focus towards ocean monitoring, Sentinel-3 mission measures the temperature, colour and height of the sea surface as well as the thickness of sea ice. It is based on a constellation of two identical satellites, Sentinel-3A and Sentinel-3B, both orbiting Earth at an altitude of $814.5\,\mathrm{km}$. Sentinel-3A was launched on 16 February 2016 and Sentinel-3B launch is planned for Q4 of 2017.

Sentinel-3 carry four instruments that work in synergy and is arguably the most complex of the Sentinel missions:

1. The Ocean and Land Colour Instrument (OLCI) delivering ocean color parameters such as chlorophyll concentration.
2. The Sea and Land Surface Temperature Radiometer (SLSTR) delivering accurate surface ocean, land and ice surface temperature.
3. The SAR Radar Altimeter (SRAL), a fully reduntant dual-frequency (Ku and C bands) nadir-looking radar altimeter used to measure the topography of ocean surfaces.
4. The Microwave Radiometer (MWR), a nadir-looking sounder operating at Ka band, assisting the SRAL by deriving atmospheric correction and water vapour measurements.

The four instruments, used in synergy, deliver a plethora of ocean monitoring products most notably being water leaving radiance (R), corrected from atmosphere reflection and sun glint, photosynthetically active radiation (PAR), chlorophyll-a concentration (Chl) in coastal and open ocean waters and associated error estimates, total suspended matter concentration (TSM) and associated error estimates, coloured dissolved organic matter concentration (CDOM) and associated error estimates, sea surface height (SSH) and sea surface height anomaly (SSHA), surface wind speed over ocean, sea surface temperature (SST), aerosol optical depth (AOD) and aerosol angstrom exponent (Å) over water at $550\,\mathrm{nm}$ and $865\,\mathrm{nm}$, thickness of sea ice with accuracy of 20 - $50\,\mathrm{cm}$, height of the sea ice with respect to the reference ellipsoid and variations of the sea ice height with respect to the mean sea surface.

1.5 Copernicus Marine Environment Monitoring Service

Copernicus, previously known as GMES (Global Monitoring for Environment and Security), is the European Programme for the establishment of a European capacity for Earth Observation and Monitoring.

It encompasses 3 components: *space* , *insitu* and *services*.

The space component includes the ESA's Sentinels which are developed for the specific needs of the Copernicus programme and the Contributing Missions, which are operated by national, European or international organisations and already provide a wealth of data for Copernicus and other services.

The insitu monitoring networks (e.g. maps, ground based weather stations, ocean buoys and air quality monitoring networks) is used to provide robust integrated information and to calibrate and validate the data from satellites.

Copernicus services address six main thematic areas:

1. Land Monitoring
2. Emergency Management
3. Marine Monitoring
4. Atmosphere Monitoring
5. Security
6. Climate Change

Copernicus Marine Environment Monitoring Service (CMEMS) offers a unique access to oceanographic products through an online catalogue. The service became fully operational on May 2015. Users can download products according to their needs in a unique format (NetCDF) and benefit from quality and validation information for each of them. All data are provided for free.

The online interactive catalogue allows users to select products according to:

- 7 geographical areas : Global Ocean, Arctic Ocean, Baltic Sea, Atlantic-European North West Shelf-Ocean, Atlantic-European South West Shelf-Ocean, Mediterranean Sea, Black Sea.
- Parameters: temperature, salinity, currents, sea ice, sea level, wind, ocean optics, ocean chemistry, ocean biology, ocean chlorophyll.
- Time Coverage: forecast, near real time, multi-year, time invariant.
- Models or Observations (satellite or insitu)
- Grid type
- Time span
- Vertical coverage
- Processing Level
- Temporal resolution

Chapter 2

Area of Interest & Data

2.1 The Mediterranean Sea

The Mediterranean is a semi-enclosed sea, connected to the Atlantic Ocean through the Strait of Gibraltar, to the Red Sea by the man-made Suez Canal and to the smaller enclosed Black Sea via the narrow Bosphorus Strait. The offshore pelagic zone comprises approximately $75\,\%$ of the total Mediterranean Sea surface area.

Two basins of almost equal size can be identified: the western basin and the eastern basin, connected by the Strait of Sicily. The Adriatic Sea extends northward between Italy and the Balkans, communicating with the eastern Mediterranean basin through the Strait of Otranto. The Aegean Sea lies between Greece and Turkey, connected to the eastern basin through several straits within the Grecian Island arc. Mediterranean circulation is driven by water exchange through the various straits, by wind stress, by freshwater and heat fluxes causing buoyancy flux on the surface.

The limited exchange of Atlantic and Mediterranean waters plays an important role in the circulation and productivity of the Mediterranean Sea. Through the Strait of Gibraltar warm surface water, already stripped of much of its nutrients by phytoplankton growth in the Atlantic, flows into the Mediterranean, returning approximately 80 - 100 years later, after circulating the Mediterranean basin in a counterclockwise (cyclonic) direction.

As the water passes eastward, the nutrient level in phytoplankton production decreases [81]. At the same time salinity increases from $36\,\mathrm{PSU}$ in Gibraltar to up to $39.1\,\mathrm{PSU}$ in the eastern basin. This is due to climatic factors such as evaporation (evaporation of salt water in Mediterranean Sea exceeds the precipitation and river runoff) [82]. A detailed description of the mechanics driving the Mediterranean Sea circulation is provided in Section **??**.

Since higher salinity increases water density, the water leaving the Mediterranean, Mediterranean Deep Water (MDW), flows below the lighter incoming Atlantic Water (AW). The decreasing west-to-east gradient of nutrients [83] results in a gradual west-to-east reduction in productivity.

A notable difference is also identified between the northern and southern Mediterranean regions in climate and geomorphology, and subsequently in vegetation and landuse [84]. The Sahara desert meets the southern coast of the Mediterranean, while the northern coasts host more productive lands. Inputs from the southern coasts through the atmosphere (dust deposition) may also play an important role in the functioning of the Mediterranean ecosystem. Most of the dust deposition occurs

during episodic events, carrying many trace elements like iron, manganese, beryllium and aluminium. Eolian input of iron, which is an important element for phytoplankton growth, is one of the highest recorded globally, possibly even exceeding the river input [85].

The importance of dust deposition in controlling ocean productivity has only recently received attention. Global climate change (global warming) may increase the areas of desert and consequently more dust could be entering the biogeochemical and sedimentary marine cycles. The effects of this process are largely unknown [86].

2.1.1 Area, depth and volume

The Mediterranean Sea has a total area of about $2\,536\,000\,\mathrm{km}^2$. It extends horizontally for $3860\,\mathrm{km}$ and its maximum width is $1600\,\mathrm{km}$. The principal rivers entering the Mediterranean are the Nile from Africa, and the Po, Rhone and Ebro from Europe. The western basin is separated from the eastern by a bank crossing the strait between Sicily and Cape Bon, where water is no deeper than $366\,\mathrm{m}$. Mean depth of the western basin is estimated to be $1612\,\mathrm{m}$, and the deepest sounding ever recorded is $3733\,\mathrm{m}$. In the eastern Mediterranean the maximum depth is $5150\,\mathrm{m}$, reached off the southern coast of Greece. The steepest slope observed is situated close to the island of Sapienza, near Navarino, where a depth of $4978\,\mathrm{m}$ is reached only 10 miles from land. The mean Sicilian-lonian sub-basin depth is $1620\,\mathrm{m}$., while the Levant sub-basin has a mean depth of $1451\,\mathrm{m}$. The total volume of the Mediterranean basins is approximately $3\,750\,000\,\mathrm{km}^3$.

2.1.2 Meteorology

The Mediterranean region forms a distinct climatic unit. The prevailing winds in this region are westerly, but during the winter the sea itself causes the formation of bands of low barometric pressure, frequently developing cyclonic disturbances. During the summer, the region comes under the influence of the northern margin of the tropical high-pressure belt and the weather is very stable.

The Mediterranean region is, hence, characteristically affected by winter rains. This characteristic becomes more clearly defined from south to north and the total annual rainfall increases in the same direction. From west to east, the climate becomes more continental (as opposed to maritime), even though with great local irregularities. The temperature in the Mediterranean region varies greatly, with annual means ranging from $14\,^\circ\mathrm{C}$ to $25\,^\circ\mathrm{C}$. The Atlantic influence limits the average annual temperature to about 10 - $12\,^\circ\mathrm{C}$ in the west, while it can reach 36 - $40\,^\circ\mathrm{C}$ in the east.

Autumn is warmer than spring, particularly in the coastal regions. This difference is enhanced in the eastern part of the region by local land winds replacing the cool sea breezes of summer. Local winds develop in nearly all the coastal areas of the Mediterranean, especially during the winter, as a result of the rapid change of temperature from the sea to the snow-clad hinterland. Cold, dry and very strong winds occur in Liguria (Tramontana), in Istria and Dalmatia (Bora), the Rhone valley (Mistral) and in the western Caucasus. In summer, a north-west trade wind called Maestro occurs in the Adriatic. The Scirocco is a cyclonic wind characteristic of the rainy winter season. In the Tyrrhenian and Adriatic regions it is usually accompanied by clouds, moisture and rain. In Sicily and southern Italy, however, the same wind

blows year-round, a dry and dusty wind from the south-east or south-west. It brings dust mainly from the Sahara. Similar winds occur also in Spain (Leveche), in Egypt (Khamsin) and in Algeria and Syria (Simooms), where their greatest development can be seen.

The mean surface temperature of Mediterranean waters ranges from over $21\,^{\circ}\mathrm{C}$ (eastern basin) to an average of $15.5\,^{\circ}\mathrm{C}$ (coast of the Gulf of Lion in the western basin). The $18\,^{\circ}\mathrm{C}$ isothermal line runs from Gibraltar to the north of Sardinia, passing through the Strait of Messina up to the Gulf of Corinth. Similarly, at a depth of $200\,\mathrm{m}$, temperature falls from $15.5\,^{\circ}\mathrm{C}$ in the eastern basin to $12.7\,^{\circ}\mathrm{C}$ in the Alboran Sea. Below $200\,\mathrm{m}$, temperature remains practically uniform down to the bottom, $13.6\,^{\circ}\mathrm{C}$ in the eastern basin and 12.7 - $13.2\,^{\circ}\mathrm{C}$ in the western basin.

2.1.3 Circulation pattern

The Mediterranean Sea is known to have a complex and multi-scale circulation, driven by wind, water flux, thermohaline and topographic features of its basins. The basic functioning of the sea, transforming Atlantic Water (AW) into Mediterranean Waters (MWs), is understood, as well as the process of dense water formation, making Atlantic Water sink offshore in specific northern zones of the western and eastern basins. However, the more detailed patterns of the circulation, mainly on basin scale, are very complex and still debated [87] [88].

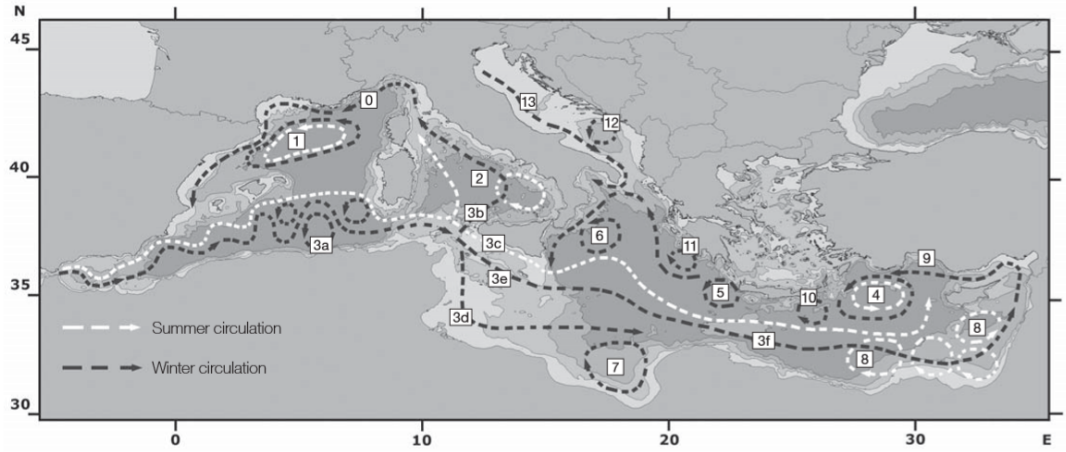


Figure 2.1 Simplified scheme of the Mediterranean's general circulation for surface water (from [89]).

The overall circulation structure is summarized in Figure 2.1 for the surface water, and in Figure 2.2 for the intermediate depths. In both the western and eastern basins, the AW current (100 - $200\,\mathrm{m}$ deep) flows along slopes, creating counter-clockwise (cyclonic) gyres, which can meander, bifurcate into veins or form branches because of seabed and coastal topography. To the south of each basin, parts of these gyres are markedly unstable and generate mesoscale anticyclonic eddies.

All along its course, AW is continuously modified, seasonally warmed (up to 20 - $28\,^{\circ}\mathrm{C}$ in the mixed layer) or cooled (down to $13\,^{\circ}\mathrm{C}$, or locally less) but, overall, its salt content increases (up to 38 - $39\,\mathrm{PSU}$) and it is thus made denser. This happens in winter, when cold and dry air masses, caused by relatively brief episodes of strong northerly winds, induce marked evaporation (increased salinity) and direct cooling

Figure 2.2 Simplified scheme of Levantine Intermediate Water (LIW) formation site and dispersal pathways (from [89]).

of AW. This dramatic increase of its density makes the AW sink. Sinking occurs in a series of specific zones: the Rhodes Gyre, the Southern-Aegean, the Southern-Adriatic, the Gulf of Lion and the Ligurian Sea.

In the western basin, intermediate and deep waters are formed in the Gulf of Lion and, during winter, in the Ligurian Sea, by the action of wind blowing from the north (Mistral and Tramontane respectively). After having circulated and accumulated at depths greater than $2000\,\mathrm{m}$, these waters flow towards the deep Tyrrhenian Sea of $-3900\,\mathrm{m}$ [90] [88].

Howerver, main circulation at intermediate depths emanates from the Rhodes gyre in the eastern basin, which is the major formation area of Levantine Intermediate Water (LIW) [91], spreading westward and northward. These intermediate waters can still be identified after leaving their zone of origin, but as they circulate they continuously mix. Finally, after around 80 - 100 years, they flow through the Strait of Gibraltar as rather homogeneous water; *the* Mediterranean Water.

Thus, Mediterranean Sea circulation transforms surface AW into denser MW, which can be recognised at a depth of 1000 - $1200\,\mathrm{m}$ in most of the northern Atlantic Ocean [92].

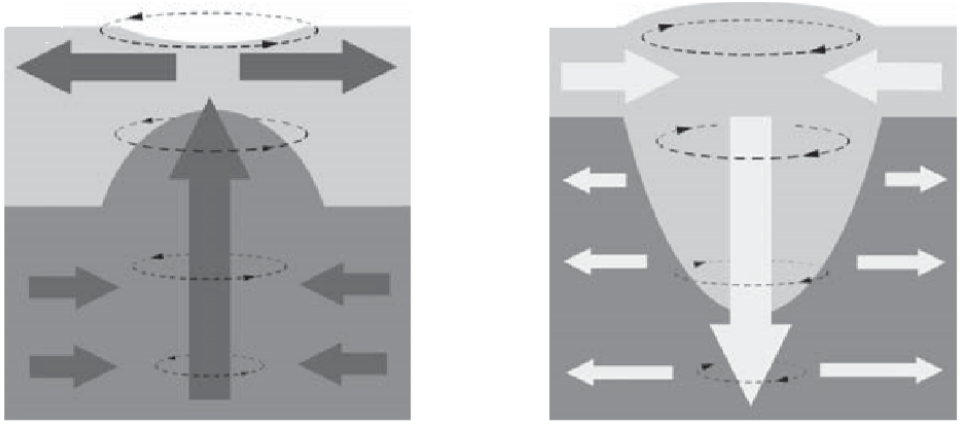
2.1.4 Ephemeral hydroghraphic features

Ephemeral hydrographic features are defined by short-living gradients in water properties, features that vanish once they mix with the surrounding waters. Highly mobile pelagic species can find and exploit ephemeral hydrographic features insofar as far as they persist [93].

Upwelling forcing is one of the best-understood ephemeral processes responsible for the creation of transient gradients in water properties. It provides nutrients to the surface layer and enhances primary production by vertically transporting cool, nutrient-rich water from the depths. Upwelling is temporally predictable and occurs seasonally in response to favourable wind conditions and counter-clockwise rotation of a current flow (cyclonic eddies). The centre of a cyclonic eddy is characterized by upwelling of nutrient-rich waters while along the periphery, downwelling can

generate retention fields. The interaction of upwelling water with the dominant flow gives rise to ephemeral fronts, which are characterized by strong gradients in physical and biological properties (Figure 2.3a).

In anticyclonic eddies, where the rotation pattern is clockwise, the aggregation of marine predators and their prey is mediated by convergence and downwelling at the core (Figure 2.3b). On a larger scale, evidence shows that when cyclonic and anticyclonic eddies act as contiguous structures, the region is characterized by elevated primary production. This results in a high zooplankton biomass and aggregation of pelagic fish, seabirds and marine mammals.



(a) Cyclonic eddy generating upwelling at the core.

(b) Anticyclonic eddy generating downwelling at the core.

Figure 2.3 Schematic representation of eddy dynamics (from [89]).

Eddies are dynamic hydrographic features, with diameters in the order of tens to hundreds of kilometres and lifespans of up to several months. As long as they are trapped by topography, they are defined as *forced eddies* and show the properties described above. Nevertheless, they can drift away from their original location, then becoming *free eddies*. In this case their effect on water masses is the inverse. Thus, a cyclonic eddy has a downwelling effect at its core, while an anticyclonic eddy generates an upwelling at its centre.

2.1.5 Persistent hydrographic features

Persistent hydrographic features, such as gyres and frontal systems, comprise some of the best-known oceanographic patterns.

Gyres are spiral currents driven primarily by large-scale wind systems and constrained by the topography of land surrounding the basins. Large, permanent mesoscale eddies can be defined as gyres even if they may have seasonal topography in location and intensity. The direction of a gyre's rotation is determined by the effect of prevailing winds on the mainstream flow. Figure 2.4 schematically shows wind-driven gyres during winter in the Mediterranean Sea [87]. Their overall effect on water mass dynamics can be the same as explained above for eddies. It must be taken into account that a gyre can encompass two or more eddies with different effects.

Surface fronts delineate boundaries between two different surface water types. Mixing processes near the interface between these water masses of different densities

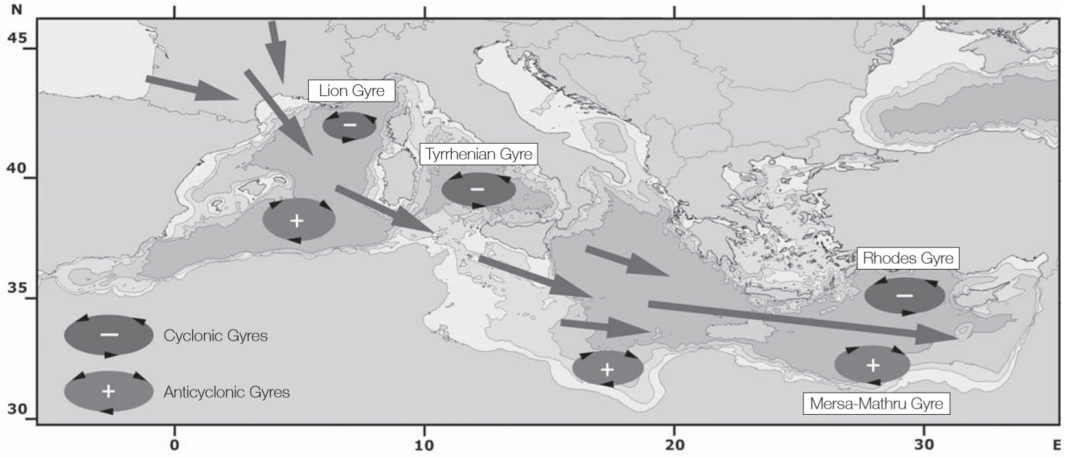


Figure 2.4 Gyres generated during winter by wind stress of Mistral along the longitudinal axis of the Mediterranean Sea (from [89]).

produces a new water type of intermediate density. This water tends to flow (downwards) under the less dense surface water type. Thus, both surface water types contribute to the formation of a mass of mixed water, which slowly sinks at the interface between them. Even in the case of two water types with equal densities, the mixture of the two will tend to have higher density due to a process known as *cabbeling*. The buoyancy-driven flows that supply this mixed water formation are directed from each of the water types toward the interface, resulting in a zone of convergence that acts to maintain the distinct frontal character of the boundary as seen in Figure 2.5.

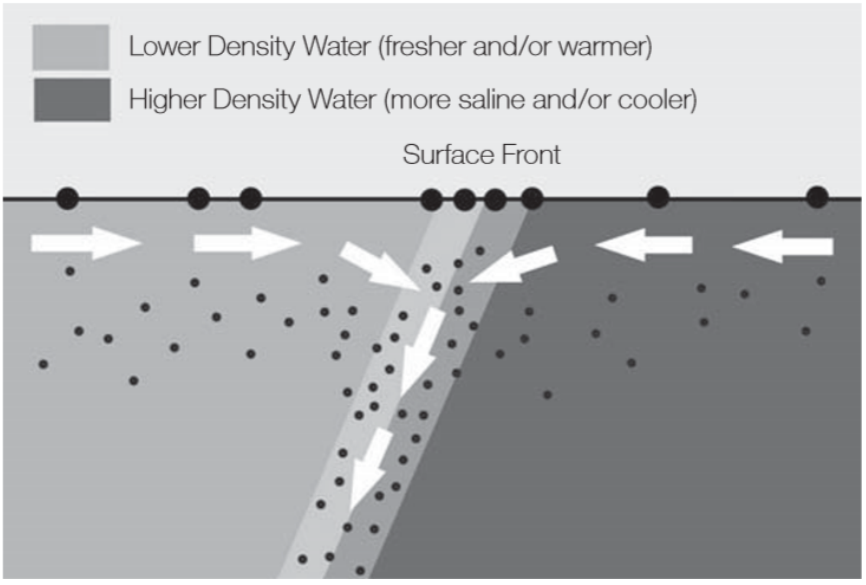
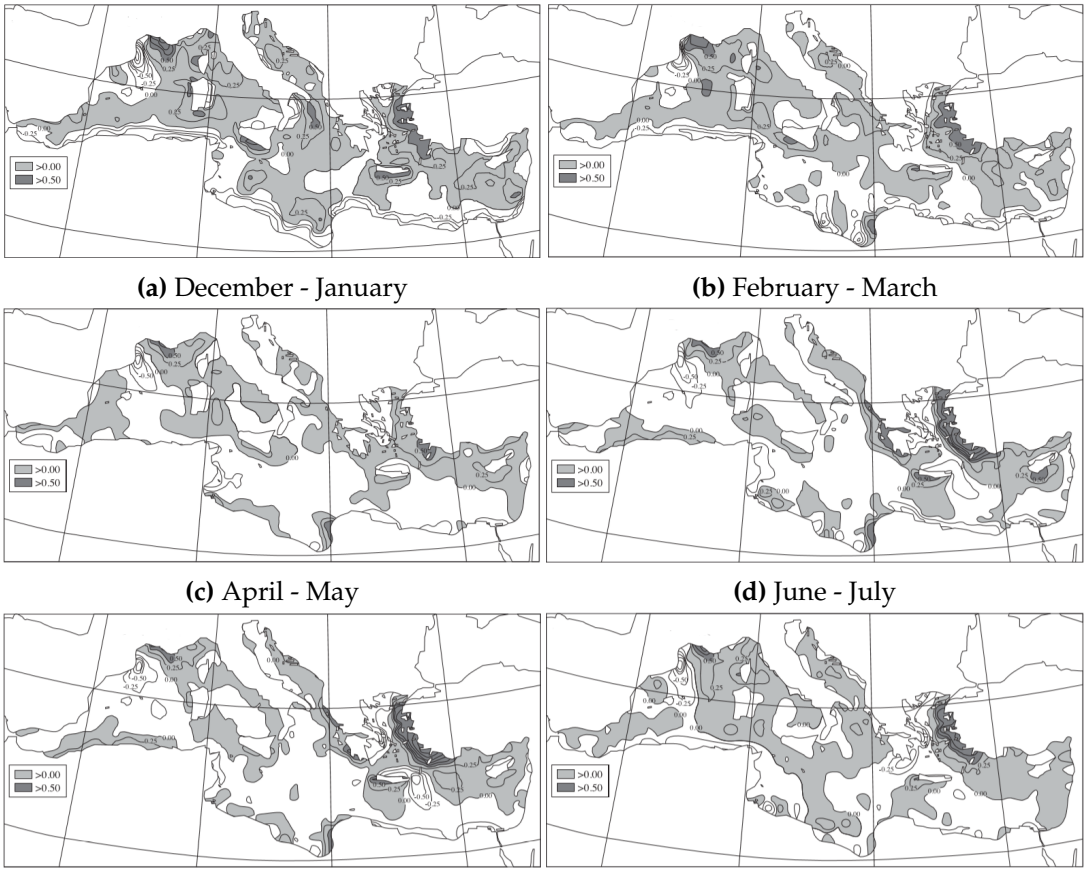


Figure 2.5 Schematic representation of a front structure and its effect on floating objects (larger black dots) and on suspended solids (smaller black dots).

Using satellite imagery (SST, SSH, Chlorophyll-a concentration, current direction and intensity, salinity, etc.), it is possible to identify offshore areas where divergence and convergence actions generate favourable conditions for the development of pelagic life. Reduced SST, by about 1 - 2 °C at the core of cyclonically rotating eddies, as well as negative SSH, with respect to the surrounding ambient, together

with increases in pigment concentrations, allow us to follow the dynamics of the enrichment processes.

Indeed, mean bi-monthly upwelling and downwelling maps for the whole Mediterranean have been already provided by Bakun and Agostini [94] (Figure 2.6).



(a) December - January **(b)** February - March

(c) April - May **(d)** June - July

(e) August - September **(f)** October - November

Figure 2.6 Seasonal variation of wind-driven upwelling (shaded areas) and downwelling (unshaded areas) zones in the Mediterranean Sea. Darker shading indicates greater upward velocities (from [94]).

2.2 Data

In order to generate the PFZ maps, daily data of Sea Surface Temperature (SST), Sea Surface Height Anomaly (SLA) and Cholorophyll concentration (CHL) as well as information about sea bottom depth are needed.

CMEMS provides daily near-real time gap-free maps (L4) of SST, SLA and CHL over the Mediterranean Sea.

SST is obtained from SST_MED_SST_L4_NRT_OBSERVATIONS_010_004 product, which covers the Mediterranean Sea and East Atlantic Ocean. This product is operationally produced and distributed in near-real time by the Istituto di Scienze dell'Atmosfera e del Clima - Gruppo di Oceanografia da Satellite (ISAC-GOS) and is based on the night-time images collected by the infrared sensors mounted on different satellite platforms. The ISAC-GOS processing chain includes several modules,

from the data extraction and preliminary quality control, to cloudy pixel removal and satellite images collating/merging. A two-step algorithm finally allows to interpolate SST data at high ($0.0625°$) and ultra-high ($0.01°$) spatial resolution, applying statistical techniques. The final product is shown in Figure 2.7.

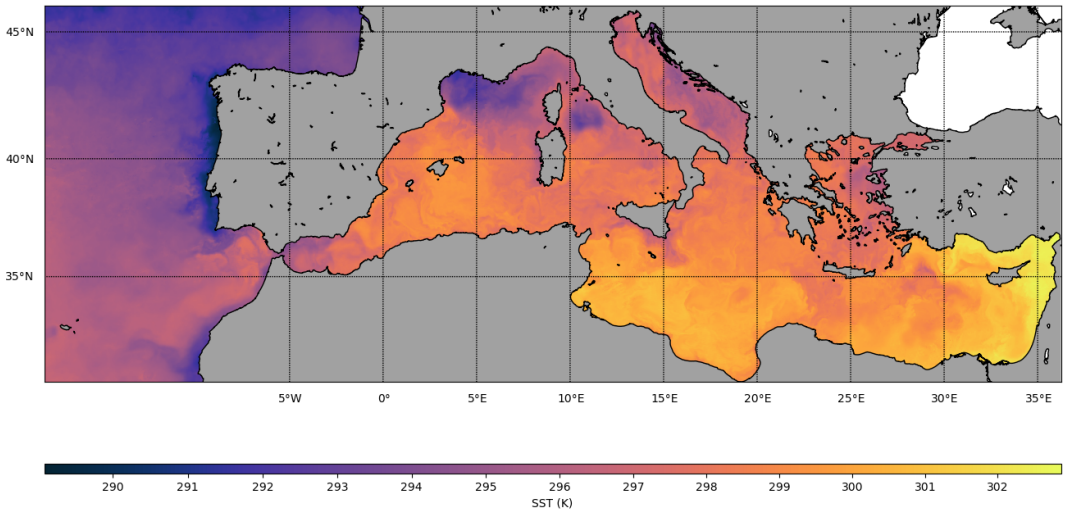


Figure 2.7 Sea Surface Temperature (SST) of Mediterranean Sea on 14-09-2017 as provided by CMEMS. SST is measured in Kelvin degrees (K).

SLA is obtained from SEALEVEL_MED_PHY_L4_NRT_OBSERVATIONS_008_050 product. This product is processed by the SL-TAC multimission altimeter data processing system. It serves in near-real time the main operational oceanography and climate forecasting centers in Europe and worldwide. It processes data from all altimeter missions: Jason-3, Sentinel-3A, HY-2A, Saral/AltiKa, Cryosat-2, Jason-2, Jason-1, T/P, ENVISAT, GFO, ERS1/2. All missions are homogenized with respect to a reference mission which is currently Jason-3 (future mission will be Sentinel-3). The acquisition of various altimeter data is a few days at most. The SLA is computed with a non-centered computation time window (6 weeks before the date). The spatial resolution is $0.125° \times 0.125°$. SLA of Mediterranean Sea on 14-09-2017 is shown in Figure **??**

CHL is obtained from OCEANCOLOUR_MED_CHL_L4_NRT_OBSERVATIONS_009_041 product. Starting from the multi-sensor (MODIS-Aqua, NPP-VIIRS and Sentinel-3A) L3 chlorophyll concentration, this L4 product includes both time averaged (8-days and monthly) datasets and the daily interpolated chlorophyll field with no data voids, all at $1\,\mathrm{km}$ spatial resolution. The datasets belonging to this product are obtained by means of the Mediterranean Ocean Colour algorithms. These are empirical ocean colour algorithms for chlorophyll retrieval for the Mediterranean Case 1 or Case 2 waters. CHL of Mediterranean Sea on 14-09-2017 is shown in Figuyre 2.9.

Detailed description of these products and the procedure for their generation can be found in the corresponding product data sheets [95] [96] [97].

Bathymetry information of the Mediterranean Sea is provided by the General Bathymetric Chart of the Oceans (GEBCO). GEBCO's aim is to provide the most authoritative publicly-available bathymetry of the world's oceans. It operates under the joint auspices of the International Hydrographic Organization (IHO) and the Intergovernmental Oceanographic Commission (IOC) of UNESCO.

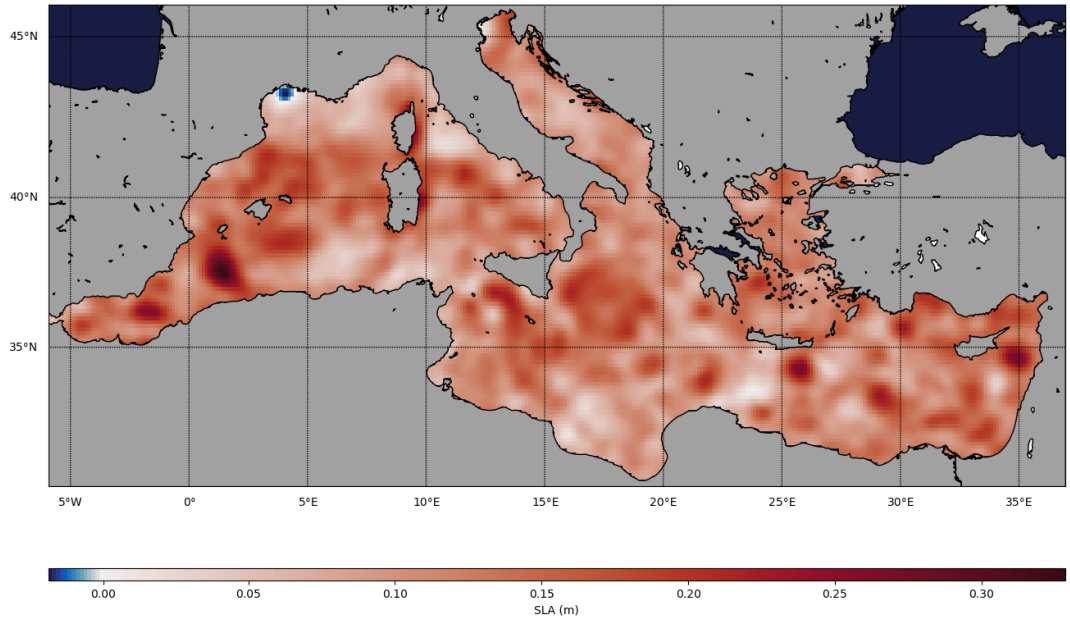


Figure 2.8 Sea Level Anomaly (SLA) of Mediterranean Sea on 14-09-2017 as provided by CMEMS. SLA is measured in meters (m).

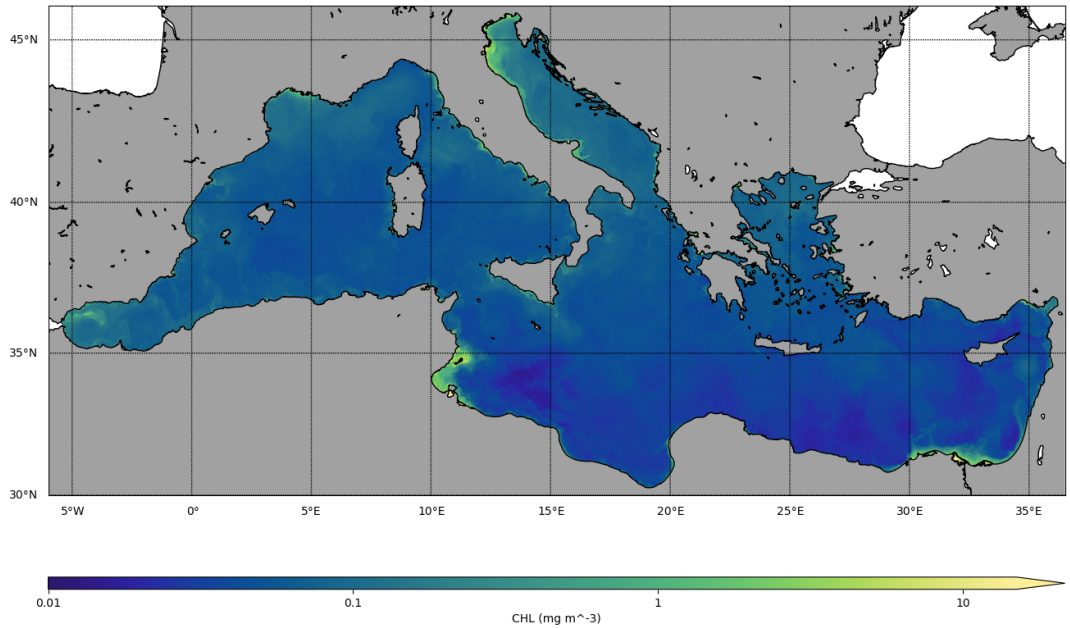


Figure 2.9 Chlorophyll concentration (CHL) of Mediterranean Sea on 14-09-2017 as provided by CMEMS. CHL is measured in milligrams per square meter ($\mathrm{mg\,m^{-3}}$).

GEBCO's gridded bathymetric data sets are global terrain models for ocean and land and include the `GEBCO\_2014 Grid`, a continuous terrain model for ocean and land with a spatial resolution of 30 arc-seconds. It was generated by combining quality-controlled ship depth soundings with interpolation between sounding points guided by satellite-derived gravity data. Where they improve on the existing grid, data sets developed by other methods are included. For more information about the development of the grids and their data sources the related paper [98] can

be referred. GEBCO_2014 Grid cropped for the Mediterranean Sea is shown in Figure 2.10.

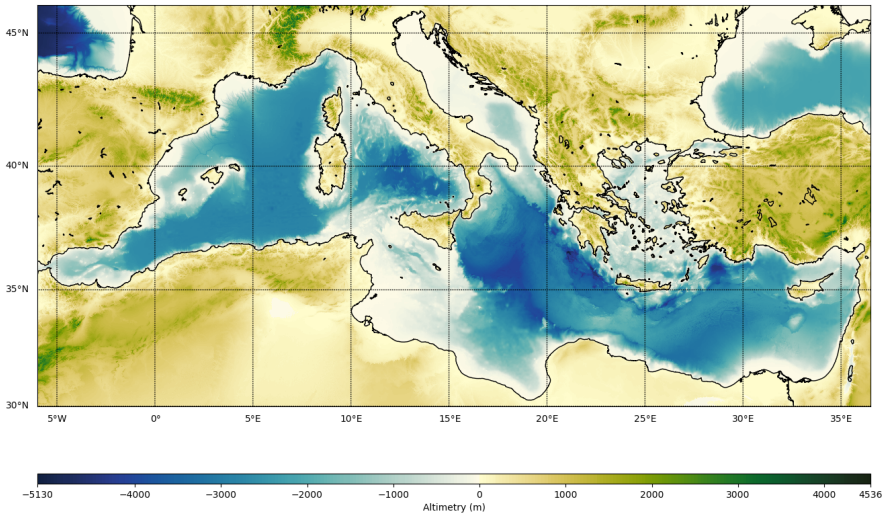


Figure 2.10 GEBCO_2014 Grid continuous terrain model for the Mediterranean Sea. Elevation is measured in meters (m).

GEBCO_2014 Grid is a terrain model for both sea and land but we need only the data referring to sea depth. This is easily achieved by masking out every pixel with elevation ≥ 0. The result is a data set with sea bottom depth infomration as shown in Figure 2.11.

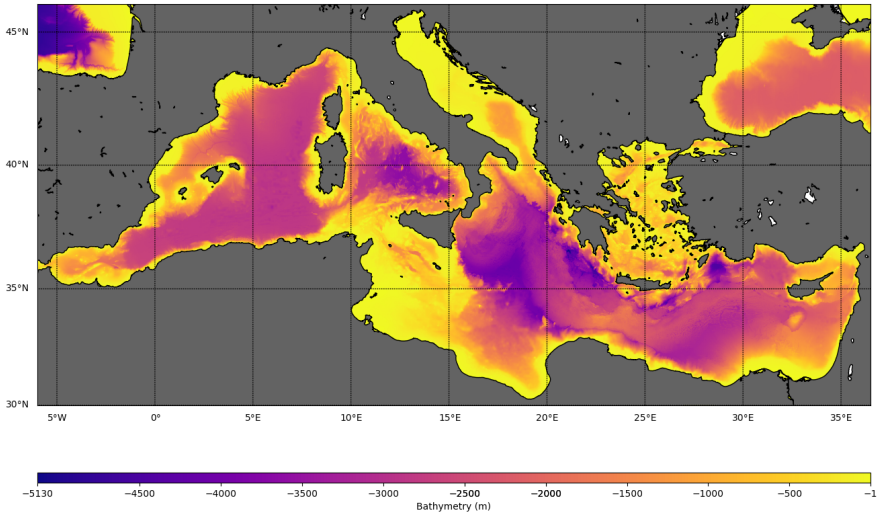


Figure 2.11 Bathymetry of the Mediterranean Sea. Bathymetry is measured in meters (m).

Chapter 3

Delineation of Possible Fishing Zone

3.1 Overview

In this chapter a new technique for generating Potential Fishing Zone (PFZ) based on fuzzy logic is proposed. This technique is based on previous studies on the potential habitat of anchovies and sardines in relation to environmental conditions. The results of these studies, discussed in Section 3.2, are expressed in fuzzy logic terms as variables and rules. Daily environmental data obtained through remote sensing are downloaded and fed into the fuzzy logic system. As a result, daily PFZ maps are generated for each fishery. The overall procedure, named Fish Alert, is depicted in Figure 3.1. Fish Alert is implemented in Python and executed from the terminal. It comprises of the main file `fishalert.py` and a python module, `facore`, that contains four files `downloader.py`, `collocator.py`, `fuzzifier.py`, `utilities.py`, corresponding to four classes `Downloader`, `Collocator`, `Fuzzifier`, `Utilities`. The full source code can be found in Appendix B.

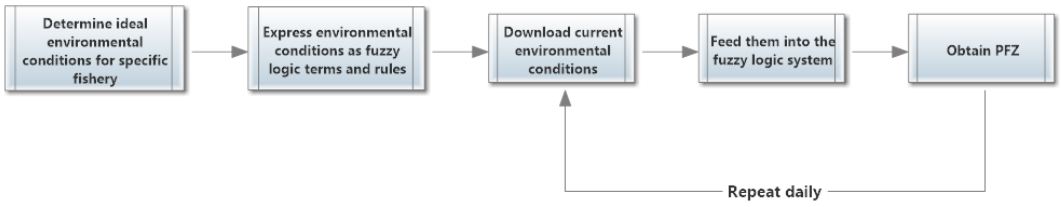


Figure 3.1 Overall workflow of PFZ generation procedure.

3.2 Fishery Habitat

Identification of the potential habitat of fishery at different life stages in relation to environmental conditions is an interesting subject from both ecological and management points of view and a necessary step in the proposed PFZ delineation process. Unfortunately, related studies are very limited in the area of the Mediterranean Sea.

Two studies were carried concerning European anchovy and European sardine habitat [99] [100]. Both used acoustic data from different seasons and different parts of the Mediterranean Sea along with satellite environmental and bathymetry data and

studied their relation using generalized additive models. Bottom depth, sea surface temperature, sea level anomaly and chlorophyll concentration were the variables found important in all models.

3.2.1 Anchovy

The spawning period of anchovy in the Mediterranean Sea lasts from spring to autumn with a peak in June to July [101] [102]. Therefore, during summer and early autumn, anchovy stocks are dominated by spawning adults, whereas in late autumn and winter they encompass a large number of juvenile fish.

Results showed that in summer (June - August) there is a higher probability of finding anchovy present in waters with depth <100 m combined with SST less than 22 °C and SLA between −12 cm and −6 cm, or SST between 24 °C and 26 °C and SLA between −6 cm and −2 cm.

In early autumn (September - mid-October), there is a higher probability of finding anchovy present in waters with depth <180 m combined with SLA of 5 - 12 cm and CHL of 0.5 - 7.4 mg m^{-3}.

In late autumn (mid-October - November), ideal waters have a depth of <150 m combined with SLA of −5 - 5 cm, SST of 17 - 19 °C and CHL values of 0.36 - 2 mg m^{-3}.

In winter (December - February), ideal conditions for finding anchovy are found in waters with depth of 15 - 60 m, SST of 8 - 10 °C or 12 - 14 °C and CHL of 0.9 - 5.4 mg m^{-3}.

3.2.2 Sardine

Results for European sardine showed that during May, June and July there is a higher probability of finding sardine present in areas with SST values be low 22 °C and depths less than 65 m and a sharply reduced probability in deeper waters. The model also indicates a higher probability of finding sardine present in areas with SLA values of −10 cm to −4 cm that also have CHL values of 0.08 - 0.39 mg m^{-3}. A high probability for sardine presence was also indicated for areas with SLA values of −3 cm to 0 cm that also have high CHL values (>1 mg m^{-3}).

During August, September and October the model indicates a higher probability of sardine presence in waters less than 110 m deep with SST values of 20 - 26 °C, SLA values of 2 - 10 cm and CHL values of 0.13 - 1.49 mg m^{-3}.

For the months of November, December and January there is a higher probability of finding sardine in waters with depth <90 m combined with SLA values of −5 cm to 0 cm, SST values of 14 - 17 °C and CHL values of 0.45 - 4.5 mg m^{-3}.

3.3 Data Processing

The first step is to download the required SST, SLA, and CHL data, discussed in Chapter 2.2, for the specific date. CMEMS data can be programmatically downloaded through the Motu client. Motu is a high efficient and robust Web Server

which fills the gap between heterogeneous data providers to end users. Motu handles, extracts and transforms oceanographic huge volumes of data without performance collapse. It enables to extract and download data through a python command line. Fish Alert encapsulates the call to Motu client into the `Downloader` class.

`Downloader` sets some essential variables upon initializiation, such as the filepath of Motu client script, user's credentials for CMEMS website and settings for the specific CMEMS products. These are provided by CMEMS websiste and are specific to each product . The `download` method calls the Motu client script which downloads the corresponding dataset according to the input date and parameter and saves it in the directory and filename given. If the `force_copy` flag is True, then the method overwrites any previous downloaded file. If the `verbose` flag is True, then the downloading details are shown in terminal.

Fish Alert's main script calls the `Downloader`'s `download` method for all three parameters, SST, SLA, and CHL.

Our workspace directory now contains four datasets; `SST.nc`, `SLA.nc`, `CHL.nc` and `bathymetry.nc`. However, as we discussed in Chapter 2.2, all datasets have different spatial resolution and coverage. Thus, the next step is to collocate these datasets into a uniformal grid. We do this utilizing the `Collocator` class.

`Collocator` imports the `snappy` module, which allows access to the ESA SNAP Java API from Python. `snappy` requires either a SNAP installation or a SNAP build. Utilizing `snappy` we get access to the Collocation Tool of ESA SNAP which allows to collocate two spatially overlapping products. Collocating two products implies that the pixel values of one product (the slave) are resampled into the geographical raster of the other (the master). The chosen sampling method is the bilinear interpolation sampling as it has shown to produce optimal results.

Fish Alert's main script calls the `Collocator` three times until all four datasets are collocated. The master dataset is bathymetry as it has the highest spatial resolution and we want to keep it.

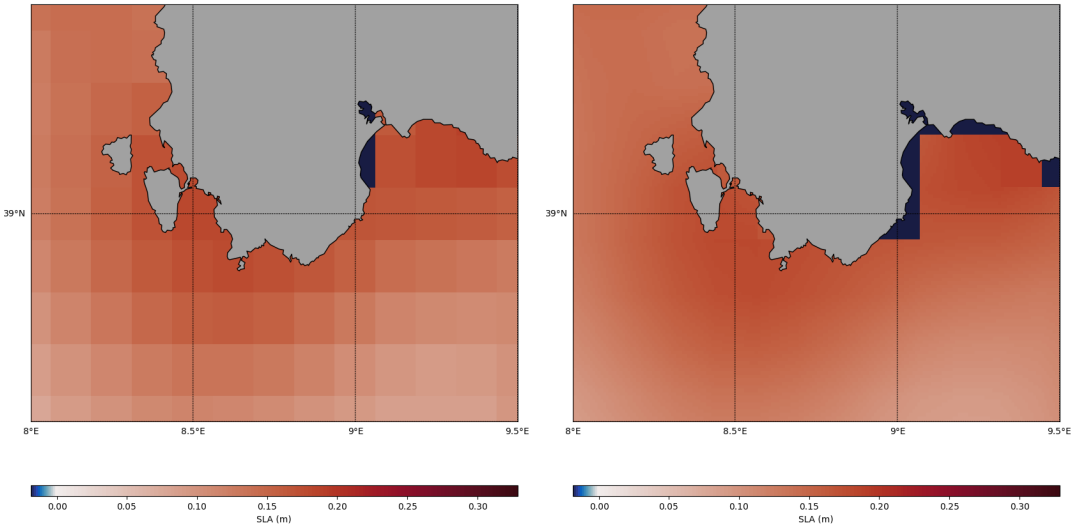
The result of this step is a single NETCDF file, `final.nc`, which contains all datasets as seperate variables. The collocated SST, CHL and SLA data for 15-09-2017 are shown in Figure 3.3. For better understanding of the resampling method, the original and collocated SLA data of the region south of Sardinia are compared in Figure 3.2.

3.4 Fuzzy Logic System

This file is the input to `Fuzzifier` constructor, which reads it and retrieves the data needed for the fuzzification process. `Fuzzifier`'s `run` method takes as input a fish F and a season S and together with the previously provided environmental data D generates the PFZ(F, S, D).

All variables of the procedure, both input and output, are expressed in fuzzy logic terms using trapezoidal membership functions. If you are unfamiliar with fuzzy logic concepts, it is advised to read Appendix A before continuing.

The output variable which denotes the possibility of PFZ existence is expressed using four terms:



(a) SLA data prior collocation. **(b)** SLA data after collocation.

Figure 3.2 Original and collocated SLA data of the region south of Sardinia.

1. Low: 0 - 25 %
2. Medium: 25 - 55 %
3. High: 55 - 85 %
4. Extreme: 85 - 100 %

The membership functions of PFZ variable is shown in Figure 3.4.

Input variables values change according to the fish and season, but follow some general guidelines. SST, SLA and CHL are usually expressed using three terms:

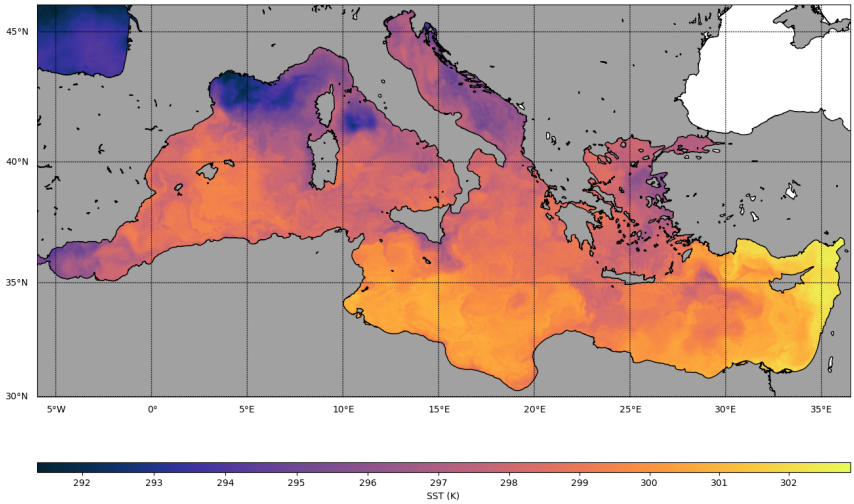
1. Low
2. Ideal
3. High

Ideal term has a value range corresponding to the values discussed in Section **??**. Low and High terms have a value range lower than the Ideal or higher respectively. For example, SST variable for anchovy during late Autumn has values as shown in Figure 3.5.

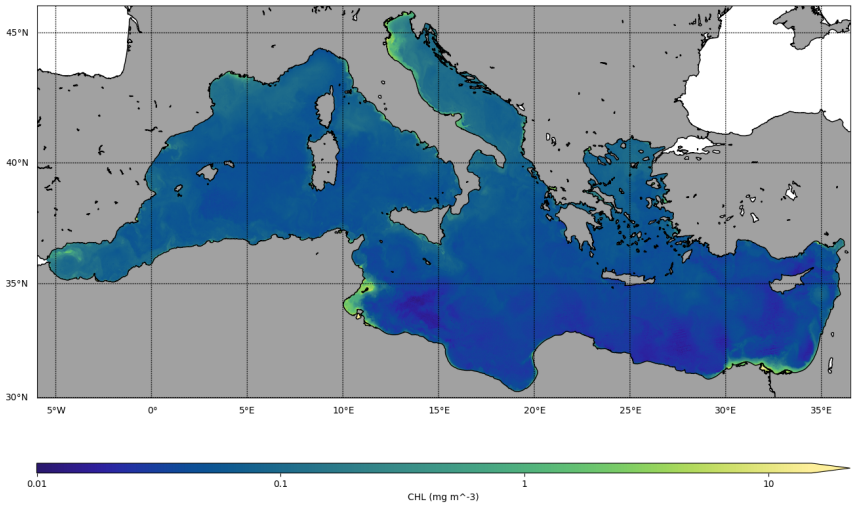
Similar terms are defined for every input variable (SST, SLA, CHLA, Depth) for every fish and for every season. These variables are then linked to the output PFZ variable through rules. Each rule is of the form IF-THEN. For example, the following rule states that if all variables have values in the ideal range then PFZ existance possibility is the highest (extreme).

```
self.rules.append(fuzz.control.Rule(
    (self.antecedents['chl']['ideal'] & self.antecedents['sst']['ideal'] &
        self.antecedents['sla']['ideal'] & self.antecedents['depth']['
    ideal']), self.consequent['extreme']))
```

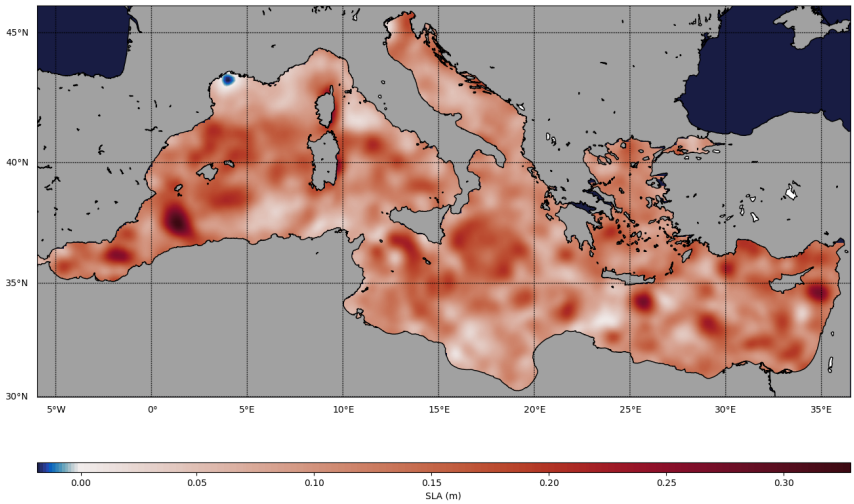
Similarly, if three variables have ideal value and one have a value too low or too high, then the PFZ existence possibility is high. If two variables have ideal value and two have a value too low or too high, then the PFZ existence possibility is medium.



(a) SST



(b) CHL



(c) SLA

Figure 3.3 Collocated SST, CHL, SLA data of the Mediterranean Sea on 15-09-2017.

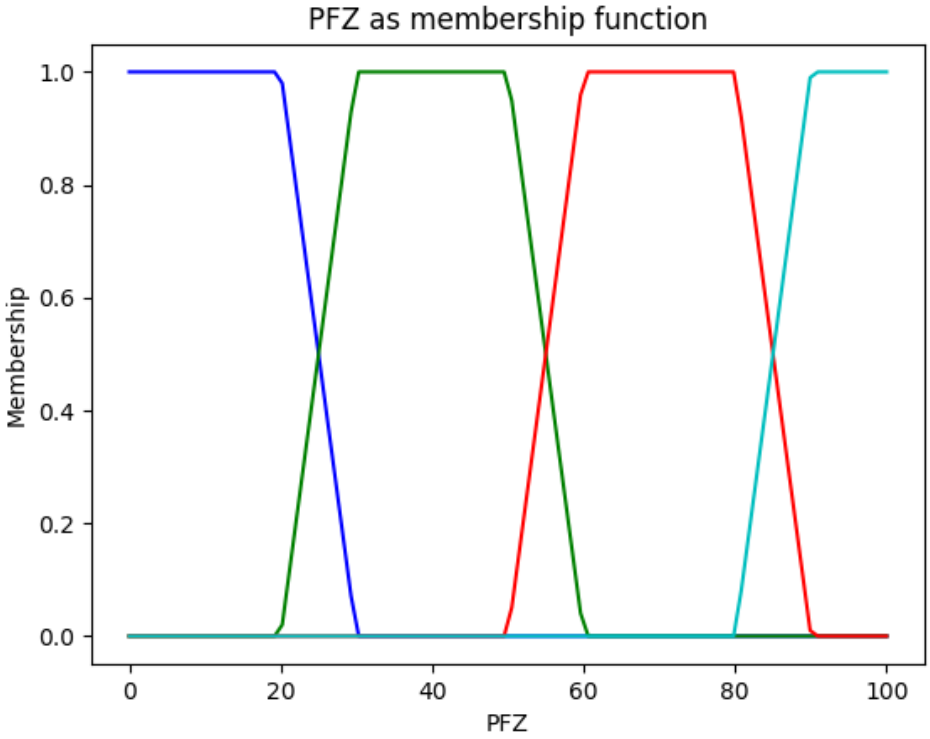


Figure 3.4 Membership function of PFZ output variable.

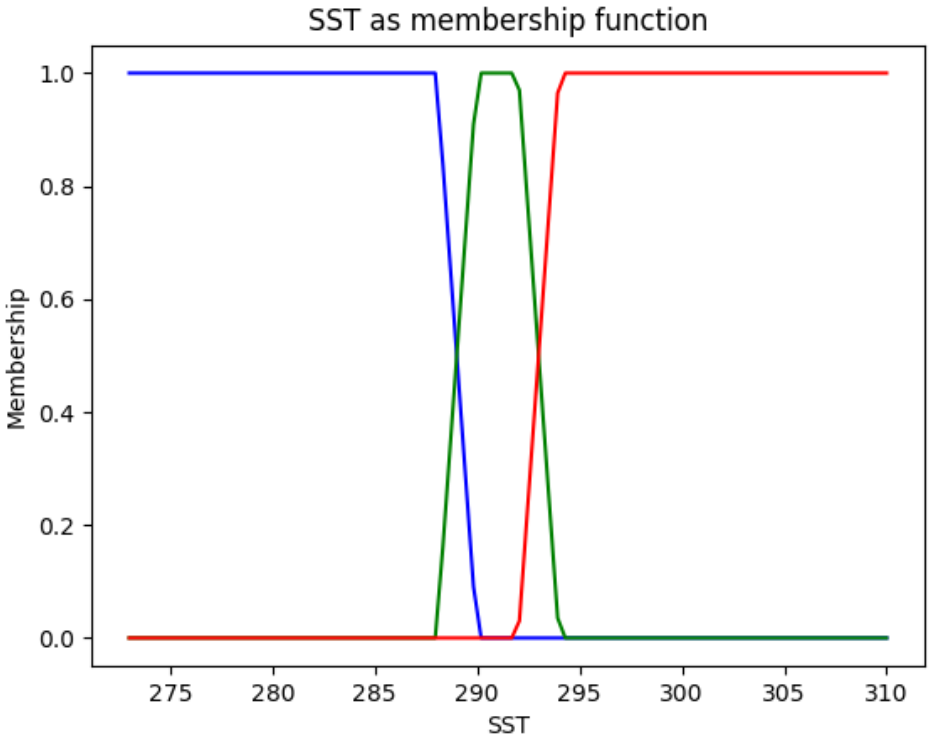


Figure 3.5 Membership function of SST input variable.

All combinations are expressed as rules, gradually lowering the possibility of PFZ existence as more input variables deviate from the ideal value range. When all the rules are defined, the algorithm runs the fuzzy inference process for every valid pixel of our ROI and revives the output value. The output is a value in the range 0 - 100.

Finally, the algorithm uses the `writeData` method to create a new NetCDF file and write the results.

Chapter 4

Results

Fish Alert produces a PFZ value for every pixel based on its environmental parameters' values. These parameters are related with each other through the fuzzy logic rules discussed in the previous section. A graphical visualization helps understanding those relations.

For instance, PFZ of anchovies during winter depends on CHL, SST and depth. In Figure 4.1 we can see how the output value PFZ changes depending on CHL and SST values assuming that depth is constant at its ideal value ($< 60\,m$). If CHL is in the range of 0.9 - $5.4\,\mathrm{mg\,m^{-3}}$ and SST is in the range of 8 - $10\,^{\circ}\mathrm{C}$, that is both are have ideal values, then PFZ is the highest (above $80\,\%$. As CHL or SST values leave their ideal range PFZ values also gradually drop down to $60\,\%$. If none of CHL and SST are in their ideal range, then PFZ drops to $40\,\%$.

After Fish Alert has calculated the PFZ for both anchovies and sardines (a procedure that can take up to 4 hours in my configuration[1], it writes the results in two seperate NetCDF files, `Anchovy.nc` and `Sardine.nc` respectively. The files generated for 19-09-2017 for anchovies and sardines are presented in Figure 4.2 and Figure 4.3 respectively.

Both PFZ maps have values over a large range. Anchovy PFZ has values from $13.5\,\%$ up to a maximum of $91.5\,\%$, while Sardine PFZ ranges from $13.25\,\%$ to $89.5\,\%$. We can examine each map separately.

Looking at the Anchovy PFZ map (Figure 4.2 and starting from the Gibraltar Strait we can see that in Alboran Sea, Anchovy PFZ is relatively low except from southern-western coast of Spain. High values are observed in the Balearic Sea near the coasts of the Balearic Islands and Spain. Gulf of lion also yields high values. Ligurian Sea and Central Tyrrhenian Sea have relatively low PFZ values, but this changes as we move near the western coast of Italy. However, in Lower Tyrrhenian Sea, PFZ values remain low even near the coasts. High PFZ values are then observed south of Sicily and specifically in the Carthagian stretch, the Sicilian Channel and the Maltese slope. The south-eastern coasts of Tunisia yield the highest PFZ values ($>90\,\%$). The high values of Anchovy PFZ continue along the African coast, especially in the Gulf of Sirte and the Nile fan. Deep waters of the eastern basin produce very low PFZ values, but small areas of high values occur in Phoenician coast and northern-eastern Cypriot Channel. Various areas of the Aegean produce high PFZ values, as well as the Ioanian coast of Greece. These values increase as we move north towards the Adriatic Sea and reach a maximum in the northeast coast of Italy.

[1] Intel Core i7 4700MQ, 16GB RAM, 256GB SSD

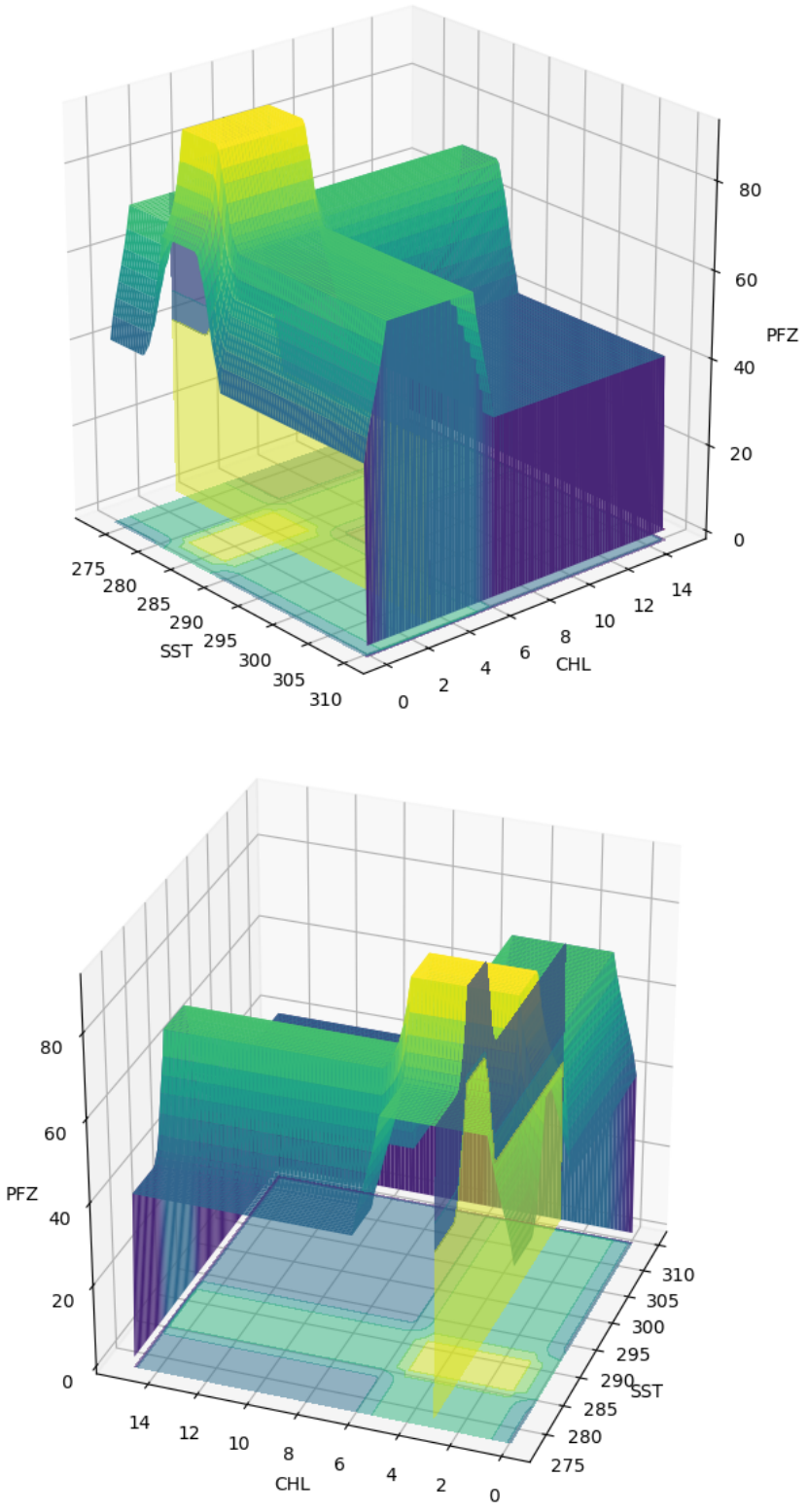


Figure 4.1 3D visualization of the PFZ values for Anchovies on Winter in relation to CHL, SST values when depth is ideal. The two figures show the same visualization from different angles

.

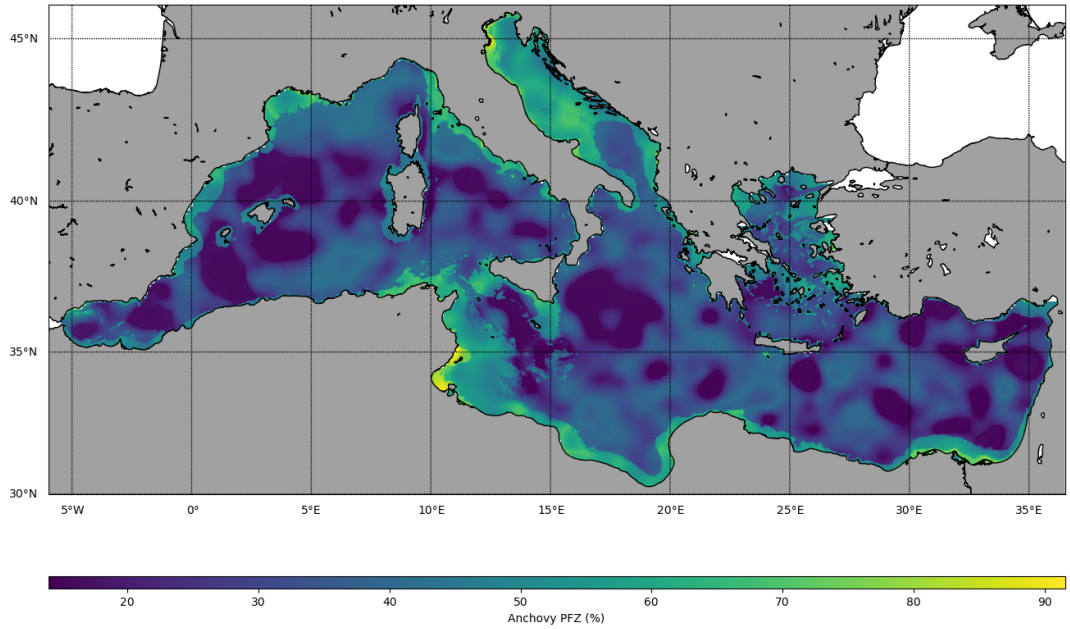


Figure 4.2 PFZ map of anchovies for 14-09-2017 generated by Fish Alert procedure.

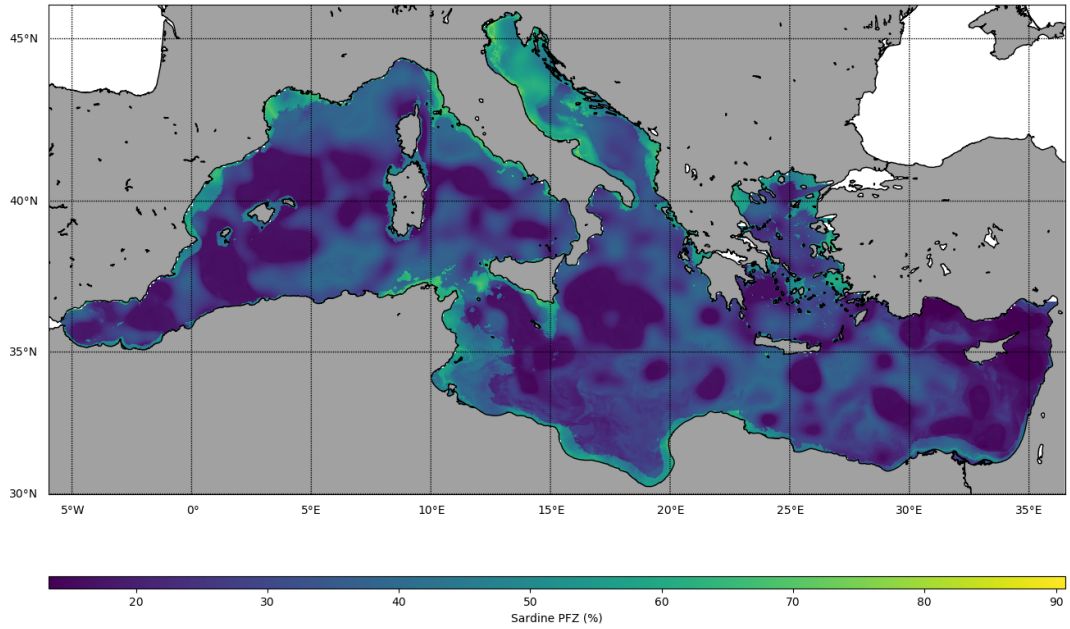


Figure 4.3 PFZ map of sardines for 14-09-2017 generated by Fish Alert procedure.

Looking at the Sardine PFZ map (Figure 4.3 we can observe high values of PFZ in the east coast of Spain in Baelaric Sea as well as in the Gulf of Lions. PFZ values are relatively low in the rest of the western basin except from the northwest coasts of Italy and the coasts of Tunisia and south Sicily. High values are also observed along the Libyan and Egyptian coasts especially around the Gulf of Sirte. Aegean Sea produces high values in various areas, like the Cyclades, the north Dodecanese and Thrakiko Pelagos. The highest PFZ values are observed in the Adriatic Sea.

In both cases, the distribution of PFZ is logical and follows the natural habitat of the fishery. Higher values of PFZ are observed in shallow areas with high productivity such as the Alboran Sea, the Gulf of Lion, the Adriatic Sea and certain areas of the Aegean Sea. On the other hand, the lowest PFZ values occur in very deep waters south of Italy and in the eastern basin. The algorithm does not estimate PFZ for the Atlantic Ocean and the Black Sea as the environmental conditions for the natural habitat of the fishery vary dramatically.

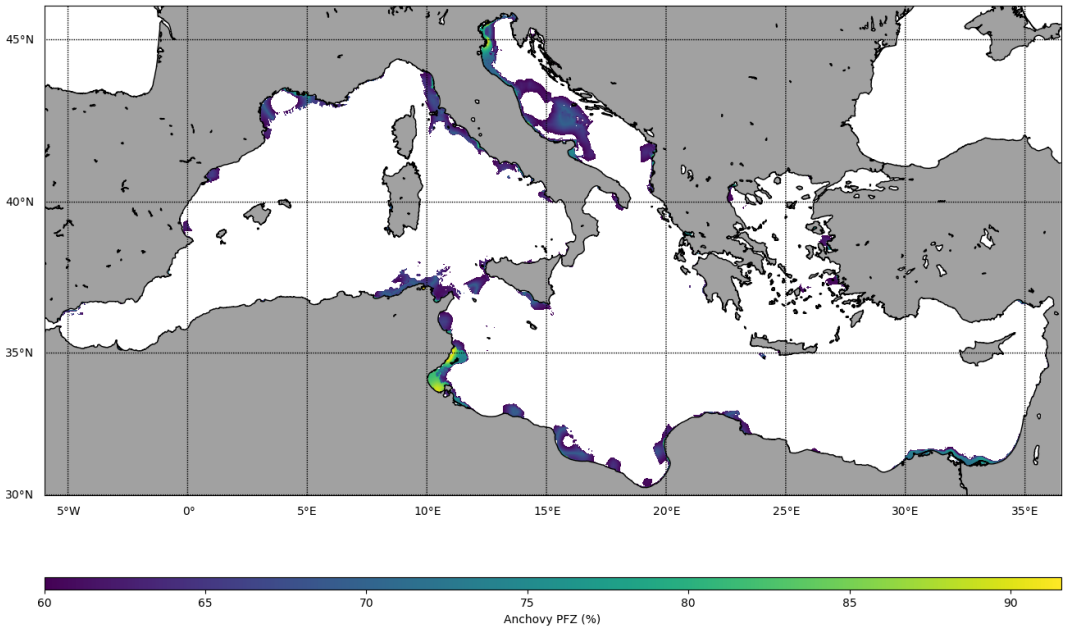


Figure 4.4 PFZ map of anchovies for 14-09-2017 with values greater than 60 %.

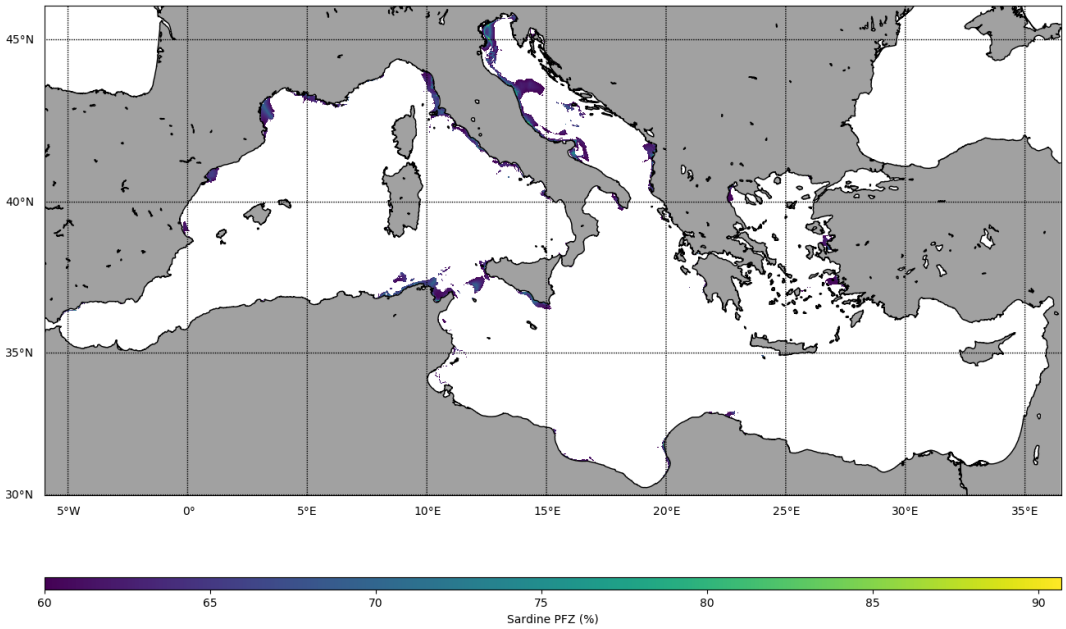
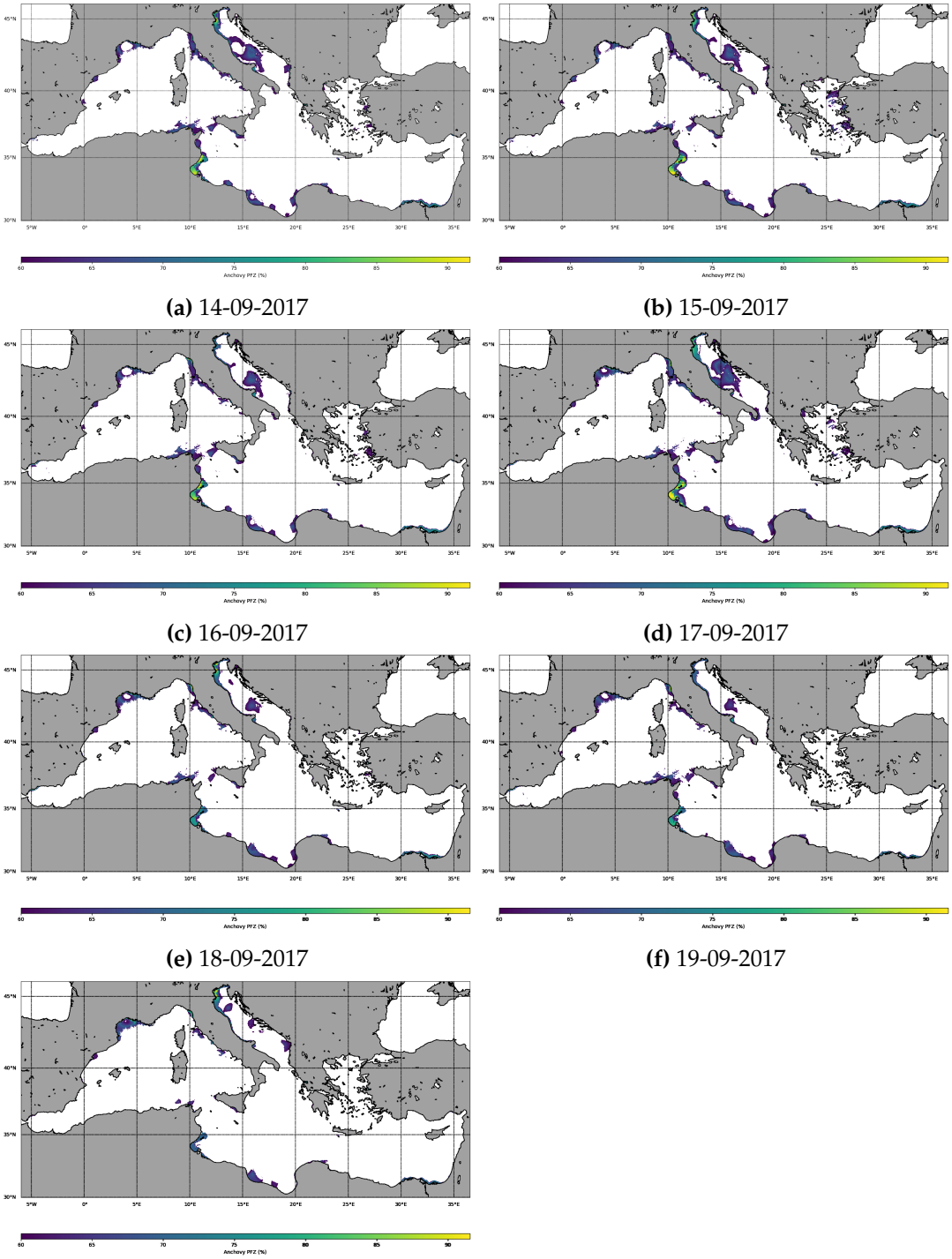


Figure 4.5 PFZ map of sardines for 14-09-2017 with values greater than 60 %.

In fact we are only interested in areas that there is a high probability to find fish populations. Thus, a better way to visualize PFZ is to isolate the areas where values

are greater than a certain threshold, with an empirical value being $60\,\%$. This is depicted in Figures 4.4 and 4.5 for anchovies and sardines respectively. Using this visualization the PFZ areas are more distinct and the differences between the two species more obvious.

It is also useful to study the results of the Fish Alert algorithm over a period of time. PFZ maps of anchovies and sardines for the week of 14 - 20 September are shown in Figure 4.6 and Figure 4.7 respectively. We can observe subtle differences in local areas with high productivity. Such areas are the Adriatic Sea, the Gulf of Lion and Sicilian Channel for anchovies and the Adriatic Sea, the Carthagian stretch and the Dodecanese for sardines. In both cases, PFZ areas are shifted smoothly over time and within logical spatial constraints.



(a) 14-09-2017

(b) 15-09-2017

(c) 16-09-2017

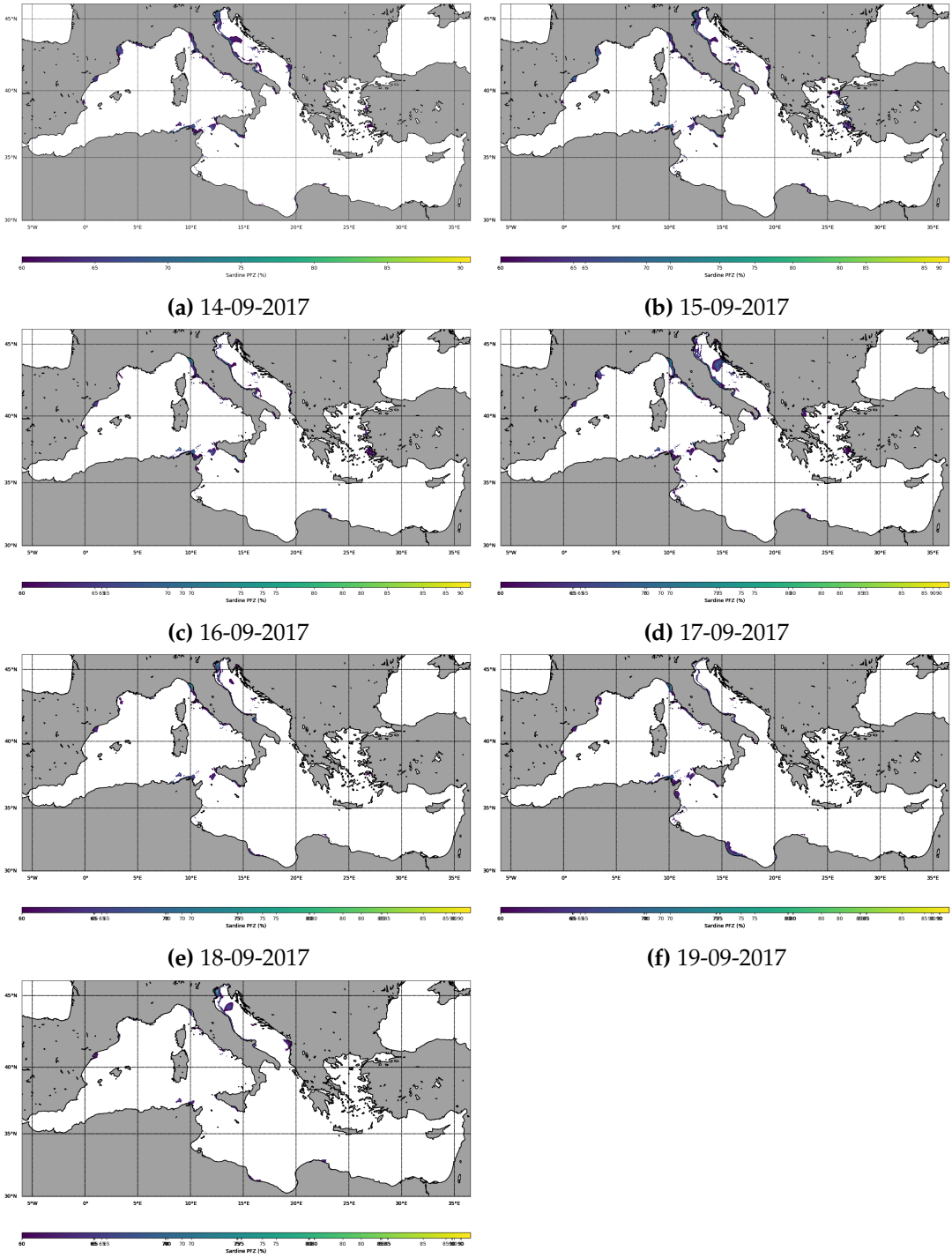
(d) 17-09-2017

(e) 18-09-2017

(f) 19-09-2017

(g) 20-09-2017

Figure 4.6 PFZ maps of anchovies for the week 14 - 20 September 2017.



(a) 14-09-2017

(b) 15-09-2017

(c) 16-09-2017

(d) 17-09-2017

(e) 18-09-2017

(f) 19-09-2017

(g) 20-09-2017

Figure 4.7 PFZ maps of sardines for the week 14 - 20 September 2017.

Chapter 5

Conclusions

Fishing is one of the major sources of food harvest for humanity and a provider of employment for economic benefits to those engaged in this activity. Remote sensing has gained increasing importance in studies of marine systems, for extracting oceanographic information and monitoring the dynamics of oceanic environment. One such important operational use of ocean remote sensing is the identification and delineation of Possible Fishing Zone (PFZ), areas with high possibility of fish aggregation.

Operational production of PFZ maps, such as the case of India, has proved to provide many benefits to society and maritime community including:

- Reduced search time, fuel cost and efforts
- Increase in profit
- $67\,\%$ success rate in PFZ
- Increase in benefit-cost ratio (Non-PFZ to PFZ):
  - 1.27 to 2.12 for trawling
  - 1.3 to 2.14 for gill netting

In this research study Fish Alert was presented. Far from perfect, it is the first initiative for an operational PFZ forecast mechanism in the Mediterranean Sea. As proof of concept, we have ensured that daily PFZ maps for the two most important fishery of the Mediterranean Sea using free satellite data is possible. Its major advantages are:

1. It provides PFZ forecasts for the whole Mediterranean Sea.
2. The forecast is species-specific.
3. The forecasts can be updated daily.
4. The mechanism is easily extensible to other species or even area of interest.

Copernicus Marine Environment Monitoring Service (CMEMS) plays a crucial role in Fish Alert implementation. Since CMEMS provides access to free, validated L4 data which are updated once daily (SST, CHL) or more (SLA), Fish Alert can forecast PFZ at least once per day. This interval is quite satisfactory. In the future, even higher frequency could be achieved as new satellite data are integrated into CMEMS (ie. when Sentinel-3 becomes fully operational).

Many improvements can be proposed including the following:

1. While Fish Alert produces logical results, the need for verification is present. This can be achieved by measuring the yield of the fishing vessels in the PFZ area and compare it to the yield of vessels in other areas. If the first case produces consistent greater yields then the procedure is adequate. Otherwise, the fuzzy logic rules should be adjusted accordingly.
2. While European sardine and European anchovy are the most important small pelagic fish in the Mediterranean Sea in terms of biomass and commercial interest, many more fish such as round sardinella (Sardinella aurita) and Sprat (Sprattus sprattus) are present. Researches focused on the relation of these species' potential habitat in relation to environmental conditions (such as those presented in Section **??**), will allow the integration of generation of the related PFZ into Fish Alert.
3. In order for Fish Alert to become useful and popular to the potential market (amateur and professional fishermen and fishing companies), a frontend application with a friendly user interface (UI) should be developed. Ideally, the application should be able to run from all modern web browsers as well as a mobile application. The application would present nearby PFZ in real time and provide other modern features such as notifications etc.

Appendix A

Fuzzy Logic

Fuzzy logic idea is similar to the human being's feeling and inference process. Unlike classical control strategy, which is a point-to-point control, fuzzy logic control is a range-to-point or range-to-range control. The output of a fuzzy controller is derived from fuzzifications of both inputs and outputs using the associated membership functions. A crisp input will be converted to the different members of the associated membership functions based on its value. From this point of view, the output of a fuzzy logic controller is based on itsmemberships of the different membership functions, which canbe considered as a range of inputs.

Fuzzy ideas and fuzzy logic are so often utilized in our routine life that nobody even pays attention to them. For instance, to answer some questions in certain surveys, most time one could answer with 'Not Very Satisfied' or 'Quite Satisfied', which are also fuzzy or ambiguous answers. Exactly to what degree is one satisfied or dissatisfied with some service or product for those surveys? These vague answers can only be created and implemented by human beings, but not machines. Is it possible for a computer to answer those survey questions directly as a human beings did? It is absolutely impossible. Computers can only understand either '0' or '1', and 'HIGH' or 'LOW'. Those data are called crisp or classic data and can be processed by all machines.

Is it possible to allow computers to handle those ambiguous data with the help of a human being? If so, how can computers and machines handle those vague data? The answer to the first question is yes. But to answer the second question,we need some fuzzy logic techniques and knowledge of fuzzy inference system.

The idea of fuzzy logic was invented by Professor L. A. Zadeh of the University of California at Berkeley in 1965 [103]. This invention was not well recognized until Dr. E. H. Mamdani, who is a professor at London University, applied the fuzzy logic in a practical application to control an automatic steam engine in 1974 [104], which is almost ten years after the fuzzy theory was invented. Then, in 1976, Blue Circle Cement and SIRA in Denmark developed an industrial application to control cement kilns [105]. That system began to operation in 1982. More and more fuzzy implementations have been reported since the 1980s, including those applications in industrial manufacturing, automatic control, automobile production, banks, hospitals, libraries and academic education. Fuzzy logic techniques have been widely applied in all aspects in today's society.

To implement fuzzy logic technique to a real application requires the following three steps:

1. Fuzzification - Convert classical data or crisp data into fuzzy data or membership functions
2. Fuzzy Inference Process - Combine membership functions with the control rules to derive the fuzzy output
3. Defuzzification - Use different methods to calculate each associated output and put them into a table: the lookup table. Pick up the output from the lookup table based on the current input during an application

A.1 Fuzzification

Fuzzification is the first step to apply a fuzzy inference system. Most variables existing in the real world are crisp or classical variables. One needs to convert those crisp variables (both input and output) to fuzzy variables, and then apply fuzzy inference to process those data to obtain the desired output. Finally, in most cases, those fuzzy outputs need to be converted back to crisp variables to complete the desired control objectives.

Generally, fuzzification involves two processes: derive the membership functions for input and output variables and represent them with linguistic variables. This process is equivalent to converting or mapping classical set to fuzzy set to varying degrees.

In practice, membership functions can have multiple different types, such as the triangular waveform, trapezoidal waveform, Gaussian waveform, bell-shaped waveform, sigmoidal waveform and S-curve waveform. The exact type depends on the actual applications. For those systems that need significant dynamic variation in a short period of time, a triangular or trapezoidal waveform should be utilized. For those system that need very high control accuracy, a Gaussian or S-curve waveform should be selected.

To illustrate the process of fuzzification, we will use a practical example - the dehumidifier, an appliance which reduces the level of humidity in the air. Assume that we have an dehumidifier control system that is under the control of only a humidity sensor. If the humidity level is low, the control motor should be turned off, and if the humidity level is high, that motor should be turned on, which is common sense. The total relative humidity range is from $0\,\%$ to $100\,\%$. In fuzzy logic terms, this range is called *universe*). For humans relative humidity below $30\,\%$ feels uncomfortable dry. Relative humidity above $60\,\%$ feels uncomfortable wet. Human comfort requires the relative humidity to be around 40 - $50\,\%$. These ranges can be translated into three fuzzy subsets (or *terms*), which are

1. Low humidity: 0 - $40\,\%$
2. Medium humidity: 20 - $60\,\%$
3. High humidity: 50 - $100\,\%$

Next those three terms need to be converted to linguistic variables: **LOW**, **MEDIUM** and **HIGH**, which correspond to the three temperature ranges listed above.

It is clear that a fuzzy set contains elements which have varying degrees of membership in the set, and this is contrasted with the classical or crisp sets because members

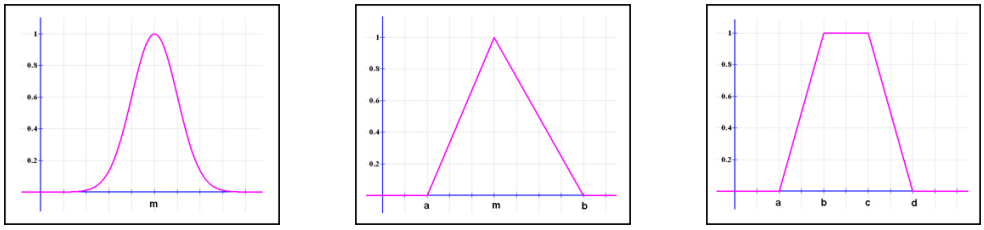
of a classical set cannot be members unless their membership is full or complete in that set. A fuzzy set allows a member to have a partial degree of membership and this partial degree membership can be mapped into a function called *membership function*. Assume that we have a fuzzy set A, and if an element x is a member of this fuzzy set A, this mapping can be denoted as

$$\mu_A(x) \in [0, 1], \quad (A = (x, \mu_A(x) \mid x \in X) \tag{A.1}$$

In essence, the membership function defines how a crisp value maps to the term on a scale of 0 to 1.

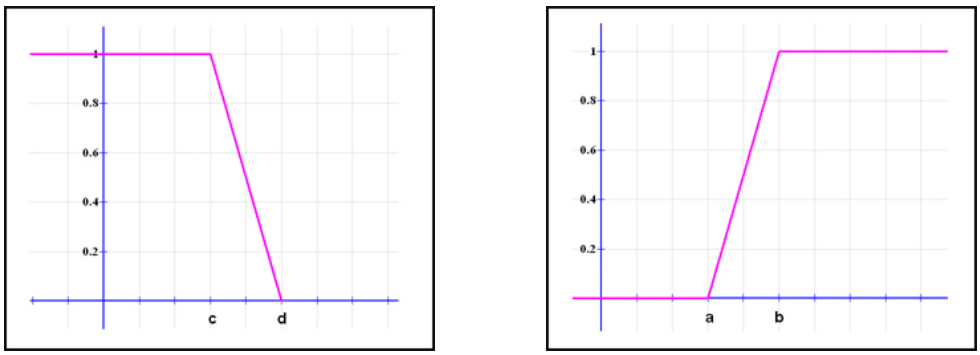
There are many membership functions with the three most common being the gaussian, triangular and trapezoidal as seen in Figure A.1. The triangular function is defined by a lower limit a, an upper limit b, and a value m, where $a < m < b$. The gaussian function is defined by a central value m and a standard deviation $k < 0$. The smaller k is, the narrower the "bell" is. The trapezoidal function is defined by a lower limit a, an upper limit d, a lower support limit b, and an upper support limit c, where $a < b < c < d$.

There are also two special cases of a trapezoidal function, which are called R-functions and L-functions. R-functions have $a = b = -\infty$, while L-functions have $c = d = \infty$. These functions are shown in Figure A.2.



(a) Gaussian membership function **(b)** Triangular membership function **(c)** Trapezoidal membership function

Figure A.1 Common membership functions.



(a) R-function **(b)** L-function

Figure A.2 Special cases of the trapezoidal membership function.

Back to our example, to make thing simple, a trapezoidal waveform is utilized as the membership function. This is depicted in Figure A.3. A crisp low humidity may be considered as a medium humidity to some degree in this fuzzy membership function representation. For instance, 30 % humidity will belong to both LOW and MEDIUM.

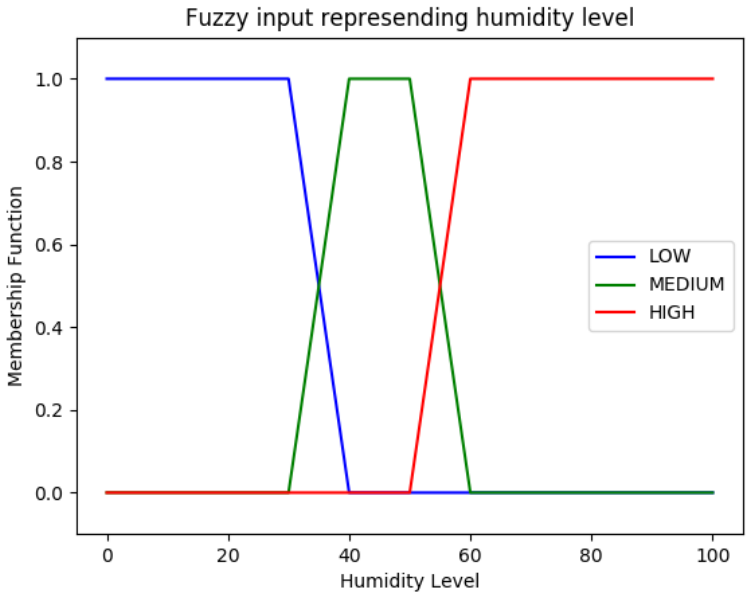


Figure A.3 Membership functions of humidity level input variable.

The desired output variable, the dehumidifier's motor speed, must also be expressed in fuzzy logic terms. As Figure A.4 shows, the selected terms are

1. Off: 0 - 40 rpm
2. Slow speed: 30 - 70 rpm
3. High speed: 60 - 100 rpm

with the corresponding linguistic variables of **OFF**, **SLOW** and **HIGH**.

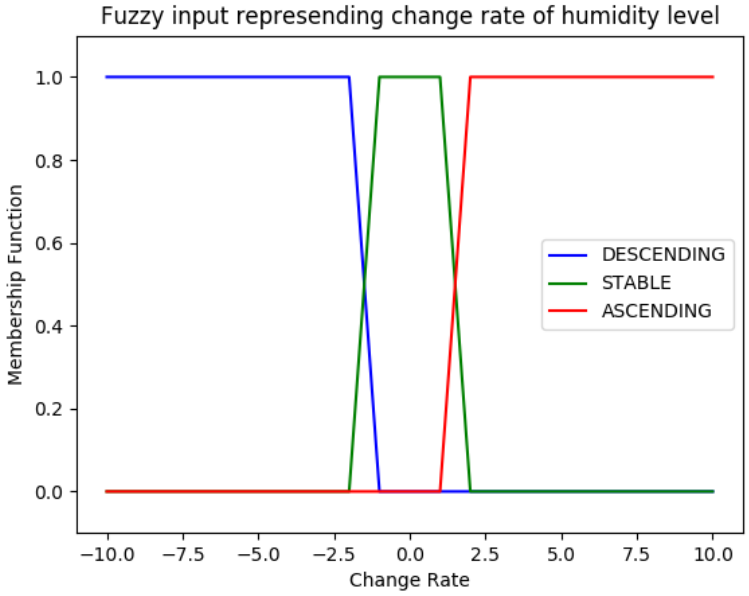


Figure A.4 Membership functions of motor speed output variable.

After the membership functions are defined for both input and output, the next step is to define the fuzzy control rule.

A.2 Fuzzy Inference Process

Fuzzy inference is the process of combining the membership functions with the control rules to derive the fuzzy output. A *fuzzy control rule* can be considered as the knowledge of an expert in any related field of application. The fuzzy rule is represented by a sequence of the form IF-THEN, leading to algorithms describing what action or output should be taken in terms of the currently observed information, which includes both input and feedback if a closed-loop control system is applied. The law to design or build a set of fuzzy rules is based on a human being's knowledge or experience, which is dependent on each different actual application.

The IF part, called the *antecedent*, is mainly used to capture knowledge by using the elastic conditions, and the THEN part, called the *consequent* can be utilized to give the conclusion or output in linguistic variable form. Still using our air dehumidifier system as an example, a fuzzy control rule can be derived as

> IF the humidity level is **HIGH**, THEN the humidifier motor speed should be **HIGH**.

For other input humidity levels, different rules should be developed. However, for most actual applications, the input variables are commonly more than one dimension. For example, in our air dehumidifier system, the inputs may include both current humidity level and the change rate of the humidity. The fuzzy control rules should also be extended to allow multiple inputs to be considered to derive the output. Table A.1 is an example of fuzzy control rules applied in our air dehumidifier system.

Rate \ Level	LOW	MEDIUM	HIGH
DESCENDING	OFF	OFF	SLOW
STABLE	OFF	SLOW	HIGH
ASCENDING	OFF	HIGH	HIGH

Table A.1 Fuzzy control rules for air dehumidifier system using two inputs.

The rows and columns represent the two inputs, the humidity level and the change rate of the humidity, and those inputs are related to the andecedent. The conclusion or control output can be considered as a third dimensional variable that is located at the cross point of each row and each column, and that conclusion is associated with the consequent. For example, when the current humidity level is **LOW**, and the current change rate of the humidity is **ASCENDING**, the dehumidifier motor speed should be set to **HIGH** speed to maintain the humidity level as soon as possible. This can be represented by the IF-THEN rule as

> IF the humidity level is **MEDIUM**, and the change rate of the humidity is **ASCENDING**, THEN the output (motor speed) should be **HIGH**.

All other rules follow a similar strategy, which is very similar to a human being's intuition.

A.3 Defuzzification

The output derived from the combination of input, output membership functions and fuzzy rules is still a fuzzy element, and this process is called fuzzy inference. To make that fuzzy output useful to real applications, a defuzzification process is needed. The defuzzification process is meant to convert the fuzzy output back to the crisp or classical output. Four defuzzification methods are commonly used, which are

1. Center of Gravity (or centroid)
   The Center of Gravity method is the most popular defuzzification technique and is widely utilized in actual applications. This method is similar to the formula for calculating the center of gravity in physics. The weighted average of the membership function or the center of the gravity of the area bounded by the membership function curve is computed to be the most crisp value of the fuzzy quantity.
2. Mean of Maximum
   The Mean of Maximum defuzzification method computes the average of those fuzzy outputs that have the highest degrees.
3. Minimum of Maximum
   The Minimum of Maximum defuzzification method returnes the minimum of those fuzzy outputs that have the highest degrees.
4. Maximum of Maximum
   The Maximum of Maximum defuzzification method returnes the minimum of those fuzzy outputs that have the highest degrees.

A graphic representation of these defuzzication methods is shown in Figure A.5.

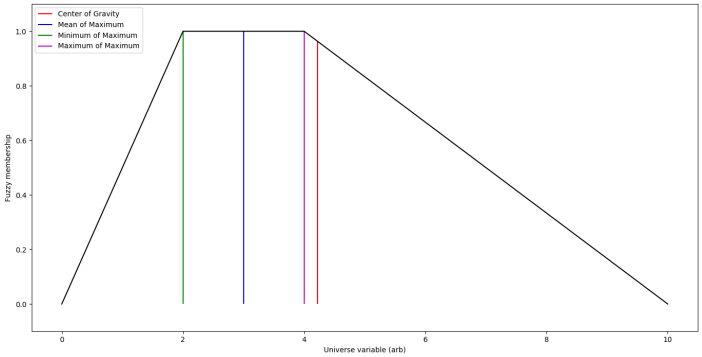


Figure A.5 Different defuzzication methods of a trapezoidal membership function.

Appendix B

Source Code

The source code presented below is as of October 24, 2017. For a more updated version please visit https://github.com/fedonman/ssta-thesis-fishalert.

B.1 `facore` module

B.1.1 `__init.py__`

```
__all__ = ['Collocator', 'Downloader', 'Fuzzifier', 'Utilities']
from .collocator import Collocator
from .downloader import Downloader
from .fuzzifier import Fuzzifier
from .utilities import Utilities
```

B.1.2 `downloader.py`

```
import argparse
import subprocess
import signal
import os
import sys
import datetime as dt

class Downloader:
    def __init__(self, motupath, username, password):
        self.motuPath = motupath
        self.username = username
        self.password = password
        self.urls = {'CHL': 'http://cmems-oc.isac.cnr.it/motu-web/Motu', '
            SST': 'http://cmems.isac.cnr.it/mis-gateway-servlet/Motu', 'SLA
            ': 'http://motu.sltac.cls.fr/motu-web/Motu'}
        self.services = {'CHL': '
            OCEANCOLOUR_MED_CHL_L4_NRT_OBSERVATIONS_009_041-TDS', 'SST': '
            SST_MED_SST_L4_NRT_OBSERVATIONS_010_004-TDS', 'SLA': '
            SEALEVEL_MED_PHY_L4_NRT_OBSERVATIONS_008_050-TDS'}
        self.products = {'CHL': 'dataset-oc-med-chl-multi-l4-
            interp_1km_daily-rt-v02', 'SST': '
            SST_MED_SST_L4_NRT_OBSERVATIONS_010_004_c_V2', 'SLA': 'dataset-
            duacs-nrt-medsea-merged-allsat-phy-l4-v3'}
        self.variables = {'CHL': ['CHL'], 'SST': ['analysed_sst', '
            analysis_error'], 'SLA': ['sla']}
```

```
        self.geo = {'CHL': '-x -6 -X 36.500480651855 -y 30 -Y
            45.998546600342', 'SST': '-x -18.120832443237 -X
            36.245834350586 -y 30.254167556763 -Y 45.995834350586', 'SLA':
            '-x -5.9375 -X 36.9375 -y 30.0625 -Y 45.9375'}

    def _prepareCmd(self, directory, filename, parameter, date):
        variables = ''
        for v in self.variables[parameter]:
            variables += '-v {0} '.format(v)
        cmd = 'python {0} -u {1} -p {2} -m {3} -s {4} -d {5} {6} -t "{7}"
            -T "{8}" {9} -o {10} -f {11}'.format(self.motuPath, self.
            username, self.password, self.urls[parameter], self.services[
            parameter], self.products[parameter], self.geo[parameter], date
            , date, variables, directory, filename)
        return cmd

    def download(self, directory, filename, parameter, date, verbose=False
        , force_copy=False):
        if not os.path.exists(directory):
            os.makedirs(directory)

        if os.path.isfile('{0}/{1}'.format(directory, filename)) and not
            force_copy:
            if verbose:
                print '{0} has already been downloaded. Skipping...'.
                    format(parameter)
            return True;

        cmd = self._prepareCmd(directory, filename, parameter, date)

        if verbose is True:
            print 'Downloading {0}...'.format(parameter)

        tries = 0
        while True:
            p = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE,
                stderr=subprocess.PIPE)
            returncode = p.wait()
            if returncode == 0:
                if verbose is True:
                    print '{0} downloaded successfully'.format(parameter)
                return True
            if returncode == 1:
                tries += 1
                if tries == 3:
                    if verbose is True:
                        print '{0} is not available'.format(parameter)
                    return False


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Download SST, SLA and
        CHL L4 daily data for Mediterranean Sea.')
    parser.add_argument('-d', '--directory', default='.', help='directory
        to download parameters')
    parser.add_argument('-m', '--motupath', help='the path to motu client'
        )
    parser.add_argument('-u', '--username', help='CMEMS username')
    parser.add_argument('-x', '--password', help='CMEMS password')
    parser.add_argument('-p', '--parameters', choices=['ALL', 'CHL', 'SST'
        , 'SLA'], default='ALL', help='the param to download')
    parser.add_argument('-d', '--date', default='today', help='a date or
        range of dates')
```

```
    parser.add_argument("-v", "--verbose", help="enable verbose mode",
        action="store_true")
    parser.add_argument("-f", "--force-copy", help="force copy if file
        exists", action="store_true")
    args = parser.parse_args()

    if not os.path.exists(args.directory):
        os.makedirs(args.directory)

    if (args.date == 'today'):
        date = dt.date.today().isoformat()
    else:
        try:
            date = dt.datetime.strptime(args.date, '%Y-%m-%d').date()
        except ValueError:
            sys.exit("Incorrect date, should be YYYY-MM-DD or today (
                default)")

    verbose = False
    if args.verbose:
        verbose = True

    force_copy = False
    if args.force_copy:
        force_copy = True

    downloader = Downloader(args.motupath, args.username, args.password)

    if args.parameters == 'CHL':
        downloader.download(args.directory, 'CHL.nc', 'CHL', date, verbose
            , force_copy)
    elif args.parameters == 'SST':
        downloader.download(args.directory, 'SST.nc', 'SST', date, verbose
            , force_copy)
    elif args.parameters == 'SLA':
        downloader.download(args.directory, 'SLA.nc', 'SLA', date, verbose
            , force_copy)
    elif args.parameters == 'ALL':
        downloader.download(args.directory, 'CHL.nc', 'CHL', date, verbose
            , force_copy)
        downloader.download(args.directory, 'SST.nc', 'SST', date, verbose
            , force_copy)
        downloader.download(args.directory, 'SLA.nc', 'SLA', date, verbose
            , force_copy)
```

B.1.3 collocator.py

```
from contextlib import contextmanager
import os
import sys

# Define a context manager to suppress stdout and stderr.
# From: https://stackoverflow.com/questions/11130156/suppress-stdout-
    stderr-print-from-python-functions
class suppress_stdout_stderr(object):
    '''
    A context manager for doing a "deep suppression" of stdout and stderr
        in
```

```
    Python, i.e. will suppress all print, even if the print originates in
        a
    compiled C/Fortran sub-function.
       This will not suppress raised exceptions, since exceptions are
           printed
    to stderr just before a script exits, and after the context manager
        has
    exited (at least, I think that is why it lets exceptions through).

    '''
    def __init__(self):
        # Open a pair of null files
        self.null_fds = [os.open(os.devnull,os.O_RDWR) for x in range(2)]
        # Save the actual stdout (1) and stderr (2) file descriptors.
        self.save_fds = (os.dup(1), os.dup(2))

    def __enter__(self):
        # Assign the null pointers to stdout and stderr.
        os.dup2(self.null_fds[0],1)
        os.dup2(self.null_fds[1],2)

    def __exit__(self, *_):
        # Re-assign the real stdout/stderr back to (1) and (2)
        os.dup2(self.save_fds[0],1)
        os.dup2(self.save_fds[1],2)
        # Close the null files
        os.close(self.null_fds[0])
        os.close(self.null_fds[1])

class Collocator:
    def __init__(self, snappypath):
        self.name = 'Collocator'
        sys.path.append(snappypath) #/home/fedonman/.snap/snap-python

    def isSNAPproduct(self, prod):
        return 'snap.core.datamodel.Product' in str(type(prod))

    def readProduct(self, file):
        import snappy
        # input parameter is already a SNAP product
        if self.isSNAPproduct(file):
            return file
        # input parameter is file, then convert it to SNAP product
        if os.path.isfile(file):
            prod = snappy.ProductIO.readProduct(file)
        elif os.path.exists(file):
            prod = None
        else:
            prod = None
        return prod

    def Collocate(self, masterFile, slaveFile, targetFile, verbose=True):
        import snappy
        if verbose is True:
            print 'Collocating {0} and {1} into {2}'.format(slaveFile,
                masterFile, targetFile)

        # Supress stdout because snappy raises warnings
        with suppress_stdout_stderr():
            # read master and slave products
            masterProduct = self.readProduct(masterFile)
            slaveProduct = self.readProduct(slaveFile)
```

```
            # import necessary Java types
            CollocateOp = snappy.jpy.get_type('org.esa.snap.collocation.
                CollocateOp')
            ResamplingType = snappy.jpy.get_type('org.esa.snap.collocation
                .ResamplingType')

            # create operator and set parameters
            ColOp = CollocateOp()
            ColOp.setParameterDefaultValues()
            ColOp.setMasterProduct(masterProduct)
            ColOp.setSlaveProduct(slaveProduct)
            ColOp.setResamplingType(ResamplingType.BILINEAR_INTERPOLATION)
            ColOp.setMasterComponentPattern('${ORIGINAL_NAME}')
            ColOp.setSlaveComponentPattern('${ORIGINAL_NAME}')

            # Apply operator
            targetProduct = ColOp.getTargetProduct()

            # Write target product as netcdf file
            snappy.ProductIO.writeProduct(targetProduct, targetFile, '
                NetCDF-BEAM')

            # dispose resources
            masterProduct.dispose()
            slaveProduct.dispose()
            del ColOp
            del CollocateOp
            del ResamplingType

            if verbose is True:
                print 'Collocation successful.'

            # return target filename
            return targetFile
```

B.1.4 fuzzifier.py

```
from __future__ import division
import os
import sys
import argparse
import netCDF4 as cdf
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
from skfuzzy import control as control
from enum import Enum

Fishery = {
    'Anchovy': {
        'Seasons': ['Summer', 'Early Autumn', 'Late Autumn', 'Winter'],
        'Output File': 'Anchovy.nc'
    },
    'Sardine': {
        'Seasons': ['June', 'September', 'December'],
        'Output File': 'Sardine.nc'
    }
}
```

```
class Fuzzifier:
    def __init__(self, file):
        self.file = file
        self.setData(file)

    def setData(self, file):
        self.data = {}
        data = cdf.Dataset(file, 'r+')
        self.data['lon'] = data['lon'][:]
        self.data['lat'] = data['lat'][:]
        self.data['chl'] = data['CHL'][:, :]
        self.data['sst'] = data['analysed_sst'][:, :]
        self.data['sla'] = data['sla'][:, :]
        self.data['depth'] = data['bathymetry'][:, :]
        self.X = len(self.data['lat'])
        self.Y = len(self.data['lon'])
        self.PixelCount = self.X * self.Y
        data.close()

    def setFuzzyRules(self, season, fishery):
        self.season = season
        self.fishery = fishery
        self.usedParameters = []
        self.antecedents = {}
        self.consequent = None
        self.rules = list()
        if fishery == 'Anchovy':
            self.outputParameter = 'anchovy'
            self.consequent = fuzz.control.Consequent(np.linspace(0, 100,
                10), self.outputParameter)
            self.consequent['low'] = fuzz.trapmf(self.consequent.universe,
                 [0, 0, 20, 30])
            self.consequent['medium'] = fuzz.trapmf(self.consequent.
                universe, [20, 30, 50, 60])
            self.consequent['high'] = fuzz.trapmf(self.consequent.universe
                , [50, 60, 80, 90])
            self.consequent['extreme'] = fuzz.trapmf(self.consequent.
                universe, [80, 90, 100, 100])
            if season == 'Summer':
                self.usedParameters = ['depth', 'sst', 'sla']

                self.antecedents['depth'] = fuzz.control.Antecedent(np.
                    linspace(-5000, 0, 100), 'depth')
                self.antecedents['sst'] = fuzz.control.Antecedent(np.
                    linspace(273, 310, 37), 'sst')
                self.antecedents['sla'] = fuzz.control.Antecedent(np.
                    linspace(-1, 1, 20), 'sla')

                self.antecedents['depth']['deep'] = fuzz.trapmf(self.
                    antecedents['depth'].universe, [-5000, -5000, -200,
                    -100])
                self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
                    antecedents['depth'].universe, [-200, -100, 0, 0])

                self.antecedents['sst']['low'] = fuzz.trapmf(self.
                    antecedents['sst'].universe, [273, 273, 285, 290])
                self.antecedents['sst']['ideal_1'] = fuzz.trapmf(self.
                    antecedents['sst'].universe, [285, 290, 295, 297])
                self.antecedents['sst']['ideal_2'] = fuzz.trapmf(self.
                    antecedents['sst'].universe, [295, 297, 298, 300])
                self.antecedents['sst']['high'] = fuzz.trapmf(self.
                    antecedents['sst'].universe, [298, 300, 310, 310])
```

```
        self.antecedents['sla']['low'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, -0.14, -0.12])
        self.antecedents['sla']['ideal_1'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.14, -0.12, -0.06,
            -0.04])
        self.antecedents['sla']['ideal_2'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.08, -0.06, -0.02, 0])
        self.antecedents['sla']['high'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.02, 0, 1, 1])

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['sla']['ideal_1'] & self.antecedents
                ['sst']['ideal_1'] & self.antecedents['depth']['
                ideal']) |
            (self.antecedents['sla']['ideal_2'] & self.antecedents
                ['sst']['ideal_2'] & self.antecedents['depth']['
                ideal'])
        , self.consequent['extreme']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['sla']['low'] & self.antecedents['
                sst']['ideal_1'] & self.antecedents['depth']['ideal
                ']) |
            (self.antecedents['sla']['high'] & self.antecedents['
                sst']['ideal_1'] & self.antecedents['depth']['ideal
                ']) |
            (self.antecedents['sla']['ideal_2'] & self.antecedents
                ['sst']['ideal_1'] & self.antecedents['depth']['
                ideal']) |
            (self.antecedents['sla']['low'] & self.antecedents['
                sst']['ideal_2'] & self.antecedents['depth']['ideal
                ']) |
            (self.antecedents['sla']['high'] & self.antecedents['
                sst']['ideal_2'] & self.antecedents['depth']['ideal
                ']) |
            (self.antecedents['sla']['ideal_1'] & self.antecedents
                ['sst']['ideal_2'] & self.antecedents['depth']['
                ideal']) |
            (self.antecedents['sla']['ideal_1'] & self.antecedents
                ['sst']['low'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['sla']['ideal_1'] & self.antecedents
                ['sst']['high'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['sla']['ideal_2'] & self.antecedents
                ['sst']['low'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['sla']['ideal_2'] & self.antecedents
                ['sst']['high'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['sla']['ideal_1'] & self.antecedents
                ['sst']['ideal_1'] & self.antecedents['depth']['
                deep']) |
            (self.antecedents['sla']['ideal_2'] & self.antecedents
                ['sst']['ideal_2'] & self.antecedents['depth']['
                deep'])
        , self.consequent['high']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['sla']['low'] & self.antecedents['
                sst']['low'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['sla']['low'] & self.antecedents['
                sst']['high'] & self.antecedents['depth']['ideal'])
                 |
```

```
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['ideal'])
                    |
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['ideal'])
                     |
                (self.antecedents['sla']['low'] & self.antecedents['
                    sst']['ideal_1'] & self.antecedents['depth']['deep'
                    ]) |
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['ideal_1'] & self.antecedents['depth']['deep'
                    ]) |
                (self.antecedents['sla']['ideal_2'] & self.antecedents
                    ['sst']['ideal_1'] & self.antecedents['depth']['
                    deep']) |
                (self.antecedents['sla']['low'] & self.antecedents['
                    sst']['ideal_2'] & self.antecedents['depth']['deep'
                    ]) |
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['ideal_2'] & self.antecedents['depth']['deep'
                    ]) |
                (self.antecedents['sla']['ideal_1'] & self.antecedents
                    ['sst']['ideal_2'] & self.antecedents['depth']['
                    deep']) |
                (self.antecedents['sla']['ideal_1'] & self.antecedents
                    ['sst']['low'] & self.antecedents['depth']['deep'])
                     |
                (self.antecedents['sla']['ideal_1'] & self.antecedents
                    ['sst']['high'] & self.antecedents['depth']['deep'
                    ]) |
                (self.antecedents['sla']['ideal_2'] & self.antecedents
                    ['sst']['low'] & self.antecedents['depth']['deep'])
                     |
                (self.antecedents['sla']['ideal_2'] & self.antecedents
                    ['sst']['high'] & self.antecedents['depth']['deep'
                    ])
            , self.consequent['medium']))
            self.rules.append(fuzz.control.Rule(
                (self.antecedents['sla']['low'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['deep']) |
                (self.antecedents['sla']['low'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['deep'])
                    |
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['deep']) |
                (self.antecedents['sla']['high'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['deep'])
            , self.consequent['low']))
        elif season == 'Early Autumn':
            self.usedParameters = ['depth', 'sla', 'chl']

            self.antecedents['depth'] = fuzz.control.Antecedent(np.
                linspace(-5000, 0, 100), 'depth')
            self.antecedents['sla'] = fuzz.control.Antecedent(np.
                linspace(-1, 1, 20), 'sla')
            self.antecedents['chl'] = fuzz.control.Antecedent(np.
                linspace(0, 30, 30), 'chl')

            self.antecedents['depth']['deep'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-5000, -5000, -360,
                -180])
            self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-360, -180, 0, 0])
```

```
        self.antecedents['sla']['low'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, 0.03, 0.05])
        self.antecedents['sla']['ideal'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [0.03, 0.05, 0.12, 0.14])
        self.antecedents['sla']['high'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [0.12, 0.14, 1, 1])

        self.antecedents['chl']['low'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0, 0, 0.4, 0.5])
        self.antecedents['chl']['ideal'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.4, 0.5, 7.4, 7.5])
        self.antecedents['chl']['high'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [7.4, 7.5, 30, 30])

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['ideal'
                ])
        , self.consequent['extreme']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['ideal'
                ]) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['deep'])
        , self.consequent['high']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['ideal'])
                |
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['deep'])
                |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['depth']['deep'])
                |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['deep'])
                |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['deep'])
        , self.consequent['medium']))
        self.rules.append(fuzz.control.Rule(
```

```
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['deep'])
                |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['low'] & self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sla']['high'] & self.antecedents['depth']['deep'])
        , self.consequent['low']))
    elif season == 'Late Autumn':
        self.usedParameters = ['depth', 'sst', 'sla', 'chl']

        self.antecedents['depth'] = fuzz.control.Antecedent(np.
            linspace(-5000, 0, 100), 'depth')
        self.antecedents['sst'] = fuzz.control.Antecedent(np.
            linspace(273, 310, 37), 'sst')
        self.antecedents['sla'] = fuzz.control.Antecedent(np.
            linspace(-1, 1, 20), 'sla')
        self.antecedents['chl'] = fuzz.control.Antecedent(np.
            linspace(0, 30, 30), 'chl')

        self.antecedents['depth']['deep'] = fuzz.trapmf(self.
            antecedents['depth'].universe, [-5000, -5000, -300,
            -150])
        self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
            antecedents['depth'].universe, [-300, -150, 0, 0])

        self.antecedents['sst']['low'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [273, 273, 288, 290])
        self.antecedents['sst']['ideal'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [288, 290, 292, 294])
        self.antecedents['sst']['high'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [292, 294, 310, 310])

        self.antecedents['sla']['low'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, -0.3, -0.05])
        self.antecedents['sla']['ideal'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.03, -0.05, 0.05,
            0.07])
        self.antecedents['sla']['high'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [0.05, 0.07, 1, 1])

        self.antecedents['chl']['low'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0, 0, 0.26, 0.36])
        self.antecedents['chl']['ideal'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.26, 0.36, 2, 2.1])
        self.antecedents['chl']['high'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [2, 2.1, 30, 30])

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['ideal'])
            , self.consequent['extreme']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['ideal']) |
```

```
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['deep'])
            , self.consequent['high']))
        self.rules.append(fuzz.control.Rule(
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
```

```
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['deep'])
            , self.consequent['medium']))
        self.ruls.append(fuzz.control.Rule(
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['ideal']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['low'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['high'] & self.antecedents['sla']['ideal'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['high'] & self.antecedents['
                sst']['ideal'] & self.antecedents['sla']['high'] &
                self.antecedents['depth']['deep']) |
            (self.antecedents['chl']['ideal'] & self.antecedents['
                sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['depth']['deep']) |
```

```
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['low'] & self.antecedents['sla']['high'] &
                    self.antecedents['depth']['deep']) |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['high'] & self.antecedents['sla']['low'] &
                    self.antecedents['depth']['deep']) |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['high'] & self.antecedents['sla']['high'] &
                    self.antecedents['depth']['deep'])
                , self.consequent['low']))
        elif season == 'Winter':
            self.usedParameters = ['depth', 'sst', 'sla', 'chl']

            self.antecedents['depth'] = fuzz.control.Antecedent(np.
                linspace(-5000, 0, 100), 'depth')
            self.antecedents['sst'] = fuzz.control.Antecedent(np.
                linspace(273, 310, 37), 'sst')
            self.antecedents['sla'] = fuzz.control.Antecedent(np.
                linspace(-1, 1, 20), 'sla')
            self.antecedents['chl'] = fuzz.control.Antecedent(np.
                linspace(0, 30, 30), 'chl')

            self.antecedents['depth']['deep'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-5000, -5000, -120,
                -60])
            self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-120, -60, 0, 0])

            self.antecedents['sst']['low'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [273, 273, 279, 281])
            self.antecedents['sst']['ideal'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [279, 281, 287, 289])
            self.antecedents['sst']['high'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [287, 289, 310, 310])

            self.antecedents['chl']['low'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [0, 0, 0.8, 0.9])
            self.antecedents['chl']['ideal'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [0.8, 0.9, 5.4, 5.5])
            self.antecedents['chl']['high'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [5.4, 5.5, 30, 30])

            self.rules.append(fuzz.control.Rule(
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['ideal'
                    ])
            , self.consequent['extreme']))
            self.rules.append(fuzz.control.Rule(
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['ideal'
                    ]) |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['ideal'
                    ]) |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['ideal'])
                    |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['ideal'])
                    |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['deep'])
            , self.consequent['high']))
```

```
            self.rules.append(fuzz.control.Rule(
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['ideal'])
                    |
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['ideal'])
                     |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['ideal'])
                    |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['ideal'])
                     |
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['deep'])
                     |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['ideal'] & self.antecedents['depth']['deep'])
                     |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['deep'])
                    |
                (self.antecedents['chl']['ideal'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['deep'])
            , self.consequent['medium']))
            self.rules.append(fuzz.control.Rule(
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['deep']) |
                (self.antecedents['chl']['low'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['deep'])
                    |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['low'] & self.antecedents['depth']['deep']) |
                (self.antecedents['chl']['high'] & self.antecedents['
                    sst']['high'] & self.antecedents['depth']['deep'])
            , self.consequent['low']))
        elif fishery == 'Sardine':
            self.outputParameter = 'sardine'
            self.consequent = fuzz.control.Consequent(np.linspace(0, 100,
                10), self.outputParameter)
            self.consequent['low'] = fuzz.trapmf(self.consequent.universe,
                 [0, 0, 20, 30])
            self.consequent['medium'] = fuzz.trapmf(self.consequent.
                universe, [20, 30, 50, 60])
            self.consequent['high'] = fuzz.trapmf(self.consequent.universe
                , [50, 60, 80, 90])
            self.consequent['extreme'] = fuzz.trapmf(self.consequent.
                universe, [80, 90, 100, 100])
            if season == 'June':
                self.usedParameters = ['depth', 'sst', 'sla', 'chl']

                self.antecedents['depth'] = fuzz.control.Antecedent(np.
                    linspace(-5000, 0, 100), 'depth')
                self.antecedents['sst'] = fuzz.control.Antecedent(np.
                    linspace(273, 310, 37), 'sst')
                self.antecedents['sla'] = fuzz.control.Antecedent(np.
                    linspace(-1, 1, 20), 'sla')

                self.antecedents['depth']['deep'] = fuzz.trapmf(self.
                    antecedents['depth'].universe, [-5000, -5000, -80,
                    -65])
                self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
                    antecedents['depth'].universe, [-80, -65, 0, 0])
```

```
        self.antecedents['sst']['ideal'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [273, 273, 290, 295])
        self.antecedents['sst']['high'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [290, 295, 310, 310])

        self.antecedents['sla']['low_1'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, -0.12, -0.1])
        self.antecedents['sla']['ideal_1'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.12, -0.1, -0.04,
            -0.02])
        self.antecedents['sla']['high_1'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.04, -0.02, 1, 1])
        self.antecedents['sla']['low_2'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, -0.05, -0.03])
        self.antecedents['sla']['ideal_2'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.05, -0.03, 0, 0.02])
        self.antecedents['sla']['high_2'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [0.02, 0.04, 1, 1])

        self.antecedents['chl']['low_1'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0, 0, 0.06, 0.08])
        self.antecedents['chl']['ideal_1'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.06, 0.08, 0.37,0.39])
        self.antecedents['chl']['high_1'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.37, 0.39, 30, 30])
        self.antecedents['chl']['low_2'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0, 0, 0.98, 1])
        self.antecedents['chl']['ideal_2'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.98, 1, 15, 15.2])
        self.antecedents['chl']['high_2'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [15, 15.2, 30, 30])

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_1
                '] & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_2
                '] & self.antecedents['chl']['ideal_2'])
            , self.consequent['extreme']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_1
                '] & self.antecedents['chl']['low_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_1
                '] & self.antecedents['chl']['high_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_2
                '] & self.antecedents['chl']['low_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal_2
                '] & self.antecedents['chl']['high_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['low_1']
                 & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['high_1'
                ] & self.antecedents['chl']['ideal_1']) |
```

```
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['low_2']
                 & self.antecedents['chl']['ideal_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['high_2'
                ] & self.antecedents['chl']['ideal_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_1'
                ] & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_2'
                ] & self.antecedents['chl']['ideal_2'])
            , self.consequent['high']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_1'
                ] & self.antecedents['chl']['low_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_1'
                ] & self.antecedents['chl']['high_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_2'
                ] & self.antecedents['chl']['low_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal_2'
                ] & self.antecedents['chl']['high_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['low_1']
                & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['high_1']
                 & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['low_2']
                & self.antecedents['chl']['ideal_2']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['high_2']
                 & self.antecedents['chl']['ideal_2']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_1']
                 & self.antecedents['chl']['ideal_1']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_2']
                 & self.antecedents['chl']['ideal_2'])
            , self.consequent['medium']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_1']
                 & self.antecedents['chl']['low_1']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_1']
                 & self.antecedents['chl']['high_1']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_2']
                 & self.antecedents['chl']['low_2']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal_2']
                 & self.antecedents['chl']['high_2']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['low_1'] &
                 self.antecedents['chl']['ideal_1']) |
```

```
                (self.antecedents['depth']['deep'] & self.antecedents[
                    'sst']['high'] & self.antecedents['sla']['high_1']
                   & self.antecedents['chl']['ideal_1']) |
                (self.antecedents['depth']['deep'] & self.antecedents[
                    'sst']['high'] & self.antecedents['sla']['low_2'] &
                     self.antecedents['chl']['ideal_2']) |
                (self.antecedents['depth']['deep'] & self.antecedents[
                    'sst']['high'] & self.antecedents['sla']['high_2']
                   & self.antecedents['chl']['ideal_2'])
                , self.consequent['low']))
        elif season == 'September':
            self.usedParameters = ['depth', 'sst', 'sla', 'chl']

            self.antecedents['depth'] = fuzz.control.Antecedent(np.
                linspace(-5000, 0, 100), 'depth')
            self.antecedents['sst'] = fuzz.control.Antecedent(np.
                linspace(273, 310, 37), 'sst')
            self.antecedents['sla'] = fuzz.control.Antecedent(np.
                linspace(-1, 1, 20), 'sla')
            self.antecedents['chl'] = fuzz.control.Antecedent(np.
                linspace(0, 30, 30), 'chl')

            self.antecedents['depth']['deep'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-5000, -5000, -130,
                -110])
            self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
                antecedents['depth'].universe, [-130, -110, 0, 0])

            self.antecedents['sst']['low'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [273, 273, 291, 293])
            self.antecedents['sst']['ideal'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [291, 293, 299, 301])
            self.antecedents['sst']['high'] = fuzz.trapmf(self.
                antecedents['sst'].universe, [299, 301, 310, 310])

            self.antecedents['sla']['low'] = fuzz.trapmf(self.
                antecedents['sla'].universe, [-1, -1, 0, 0.02])
            self.antecedents['sla']['ideal'] = fuzz.trapmf(self.
                antecedents['sla'].universe, [0, 0.02, 0.1, 0.12])
            self.antecedents['sla']['high'] = fuzz.trapmf(self.
                antecedents['sla'].universe, [0.1, 0.12, 1, 1])

            self.antecedents['chl']['low'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [0, 0, 0.11, 0.13])
            self.antecedents['chl']['ideal'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [0.11, 0.13, 1.49, 1.51])
            self.antecedents['chl']['high'] = fuzz.trapmf(self.
                antecedents['chl'].universe, [1.49, 1.51, 30, 30])

            self.rules.append(fuzz.control.Rule(
                (self.antecedents['depth']['ideal'] & self.antecedents
                    ['sst']['ideal'] & self.antecedents['sla']['ideal']
                     & self.antecedents['chl']['ideal'])
                , self.consequent['extreme']))

            self.rules.append(fuzz.control.Rule(
                self.antecedents['depth']['ideal'] &
                ((self.antecedents['sst']['ideal'] & self.antecedents[
                    'sla']['ideal'] & self.antecedents['chl']['low']) |
                (self.antecedents['sst']['ideal'] & self.antecedents['
                    sla']['ideal'] & self.antecedents['chl']['high']) |
                (self.antecedents['sst']['ideal'] & self.antecedents['
                    sla']['low'] & self.antecedents['chl']['ideal']) |
```

```
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['ideal'])
                |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['ideal']))
            , self.consequent['high']))

        self.rules.append(fuzz.control.Rule(
            self.antecedents['depth']['ideal'] &
            ((self.antecedents['sst']['ideal'] & self.antecedents[
                'sla']['low'] & self.antecedents['chl']['low']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['low']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['low'] & self.antecedents['chl']['high']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['high']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['low'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['low'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['low']) |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['low']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['high']) |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['high']))
            , self.consequent['medium']))
        self.rules.append(fuzz.control.Rule(
            self.antecedents['depth']['deep'] &
            ((self.antecedents['sst']['ideal'] & self.antecedents[
                'sla']['ideal'] & self.antecedents['chl']['low']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['high']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['low'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['ideal'] & self.antecedents['
                sla']['high'] & self.antecedents['chl']['ideal']) |
            (self.antecedents['sst']['low'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['ideal'])
                |
            (self.antecedents['sst']['high'] & self.antecedents['
                sla']['ideal'] & self.antecedents['chl']['ideal']))
            , self.consequent['medium']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
```

```
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['high']) |
```

```
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['high'])
            , self.consequent['low']))

    elif season == 'December':
        self.usedParameters = ['depth', 'sst', 'sla', 'chl']

        self.antecedents['depth'] = fuzz.control.Antecedent(np.
            linspace(-5000, 0, 100), 'depth')
        self.antecedents['sst'] = fuzz.control.Antecedent(np.
            linspace(273, 310, 37), 'sst')
        self.antecedents['sla'] = fuzz.control.Antecedent(np.
            linspace(-1, 1, 20), 'sla')
        self.antecedents['chl'] = fuzz.control.Antecedent(np.
            linspace(0, 30, 30), 'chl')

        self.antecedents['depth']['deep'] = fuzz.trapmf(self.
            antecedents['depth'].universe, [-5000, -5000, -110,
            -90])
        self.antecedents['depth']['ideal'] = fuzz.trapmf(self.
            antecedents['depth'].universe, [-110, -90, 0, 0])

        self.antecedents['sst']['low'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [273, 273, 285, 287])
        self.antecedents['sst']['ideal'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [285, 287, 290, 292])
        self.antecedents['sst']['high'] = fuzz.trapmf(self.
            antecedents['sst'].universe, [290, 292, 310, 310])

        self.antecedents['sla']['low'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-1, -1, -0.07, -0.05])
        self.antecedents['sla']['ideal'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [-0.07, -0.05, 0, 0.02])
        self.antecedents['sla']['high'] = fuzz.trapmf(self.
            antecedents['sla'].universe, [0, 0.02, 1, 1])

        self.antecedents['chl']['low'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0, 0, 0.43, 0.45])
        self.antecedents['chl']['ideal'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [0.43, 0.45, 4.5, 4.7])
        self.antecedents['chl']['high'] = fuzz.trapmf(self.
            antecedents['chl'].universe, [4.5, 4.7, 30, 30])

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal']
                 & self.antecedents['chl']['ideal'])
            , self.consequent['extreme']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal']
                 & self.antecedents['chl']['low']) |
```

```
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['ideal']
                 & self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['low'] &
                 self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['high']
                & self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['ideal'])
            , self.consequent['high']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['low'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['high']
                & self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['low'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['ideal'] & self.antecedents['sla']['high']
                & self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['high'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['ideal']
                & self.antecedents['chl']['high']) |
```

```
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['ideal'])
            , self.consequent['medium']))

        self.rules.append(fuzz.control.Rule(
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['ideal'] & self.antecedents['sla']['high'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['ideal']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['low']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['low'] & self.antecedents['sla']['ideal'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['deep'] & self.antecedents[
                'sst']['high'] & self.antecedents['sla']['ideal'] &
                 self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['low']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['low'] &
                self.antecedents['chl']['high']) |
            (self.antecedents['depth']['ideal'] & self.antecedents
                ['sst']['low'] & self.antecedents['sla']['high'] &
                self.antecedents['chl']['high']) |
```

```
                (self.antecedents['depth']['ideal'] & self.antecedents
                    ['sst']['high'] & self.antecedents['sla']['low'] &
                    self.antecedents['chl']['low']) |
                (self.antecedents['depth']['ideal'] & self.antecedents
                    ['sst']['high'] & self.antecedents['sla']['low'] &
                    self.antecedents['chl']['high']) |
                (self.antecedents['depth']['ideal'] & self.antecedents
                    ['sst']['high'] & self.antecedents['sla']['high'] &
                     self.antecedents['chl']['low']) |
                (self.antecedents['depth']['ideal'] & self.antecedents
                    ['sst']['high'] & self.antecedents['sla']['high'] &
                     self.antecedents['chl']['high'])
                , self.consequent['low']))

    def run(self, season, fishery, verbose=True):
        self.setFuzzyRules(season, fishery)
        system = fuzz.control.ControlSystem(self.rules)
        simulation = fuzz.control.ControlSystemSimulation(system)
        self.results = np.zeros((self.X, self.Y))
        if verbose is True:
            print 'Generating PFZ...'
        for x in range(self.X):
            if verbose is True:
                sys.stdout.write('Progress: {:2.1%}\r'.format((x * self.Y)
                     / self.PixelCount))
                sys.stdout.flush()
            for y in range(self.Y):
                data = { param:self.data[param].item(x,y) for param in
                    self.usedParameters }
                if any(np.isnan(value) for value in data.values()):
                    self.results.itemset((x, y), -999)
                else:
                    for param, value in data.items():
                        simulation.input[param] = value
                    simulation.compute()
                    self.results.itemset((x, y), simulation.output[self.
                        outputParameter])
        if verbose is True:
            print 'PFZ generated successfully.'

    def writeData(self, filename, verbose=True):
        if verbose is True:
            print 'Writing data to {0}'.format(filename)
        dsin = cdf.Dataset(self.file)
        dsout = cdf.Dataset(filename, 'w')
        for dname, the_dim in dsin.dimensions.items():
            dsout.createDimension(dname, len(the_dim))
        for v_name, varin in dsin.variables.items():
            if v_name == 'lat' or v_name == 'lon':
                outVar = dsout.createVariable(v_name, varin.datatype,
                    varin.dimensions)
                outVar.setncatts({k: varin.getncattr(k) for k in varin.
                    ncattrs()})
                outVar[:] = varin[:]

        var = dsout.createVariable(self.outputParameter, np.dtype('float32
            '), ('lat', 'lon'), fill_value=-999, zlib=True,
            least_significant_digit=1)
        var.units = '%'
        var.long_name = 'Possibility of {0} Fishing Zone'.format(self.
            fishery)
        var.valid_range = np.array((0.0, 100.0))
        var[:] = self.results[:]
```

```
        dsin.close()
        dsout.close()
        if verbose is True:
            print 'Data written successfully'

    def ViewMembershipRelationships(self):
        system = fuzz.control.ControlSystem(self.rules)
        simulation = fuzz.control.ControlSystemSimulation(system)
        sst = np.linspace(273, 310, 38)
        #sla = np.linspace(-1, 1, 21)
        chl = np.linspace(0, 20, 21)
        x, y = np.meshgrid(sst, chl)
        z = np.zeros_like(x)

        # Loop through the system 21*21 times to collect the control
            surface
        for i in range(len(chl) - 1):
            for j in range(len(sst) - 1):
                simulation.input['sst'] = x[i, j]
                simulation.input['chl'] = y[i, j]
                simulation.input['depth'] = -1000
                #print('{0}, {1}'.format(x[i,j], y[i,j]))
                simulation.compute()
                z[i, j] = simulation.output['anchovy']
                self.consequent.view(sim=simulation)

        # Plot the result in pretty 3D with alpha blending
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D  # Required for 3D
            plotting

        fig = plt.figure(figsize=(8, 8))
        ax = fig.add_subplot(111, projection='3d')

        surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='
            viridis',
                               linewidth=0.4, antialiased=True)

        cset = ax.contourf(x, y, z, zdir='z', offset=-2.5, cmap='viridis',
             alpha=0.5)
        cset = ax.contourf(x, y, z, zdir='x', offset=3, cmap='viridis',
            alpha=0.5)
        cset = ax.contourf(x, y, z, zdir='y', offset=3, cmap='viridis',
            alpha=0.5)

        ax.view_init(30, 200)
        plt.show()

    def printRules(self):
        for rule in self.rules:
            rule.view()
        fig = plt.figure(figsize=(8, 8))
        plt.show()
```

B.1.5 utilities.py

```
import netCDF4 as cdf
import argparse
import random
```

```
import string

class Utilities:
    @staticmethod
    def deleteCollocationFlags(file):
        dsin = cdf.Dataset(file, 'r+')
        dsin.renameVariable('collocation_flags', 'collocation_flags_'.join
            (random.choice(string.ascii_uppercase + string.digits) for _ in
             range(5)))
        dsin.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Utility functions needed
         in the generation of PFZs')
    group = parser.add_mutually_exclusive_group(required=True)
    group.add_argument('-d', '--deleteCollocationFlags', type=str, help='
        Remove collocation flags from file (actually just rename the
        variable).')
    args = parser.parse_args()

    if args.deleteCollocationFlags is not None:
        Utilities.deleteCollocationFlags(args.deleteCollocationFlags)
```

B.2 fishalert.py

```
from facore import *
import config
import os
import sys
import datetime
import argparse
import time
from shutil import copy

def date_to_season(date, fishery):
    year, month, day = date.split('-')
    month = int(month)
    day = int(day)
    if fishery == 'Anchovy':
        if month == 12 or month == 1 or month == 2:
            return 'Winter'
        elif month == 3 or month == 4 or month == 5:
            return None
        elif month == 6 or month == 7 or month == 8:
            return 'Summer'
        elif month == 9 or (month == 10 and day < 15):
            return 'Early Autumn'
        elif (month == 10 and day >= 15) or month == 11:
            return 'Late Autumn'
    elif fishery == 'Sardine':
        if month == 5 or month == 6 or month == 7:
            return 'June'
        elif month == 8 or month == 9 or month == 10:
            return 'September'
        elif month == 11 or month == 12 or month == 1:
            return 'December'
        else:
            return None
```

```
    else:
        return None

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Calculate Possible
        Fishing Zones in the Mediterranean Sea.')
    parser.add_argument('-f', '--fishery', choices=['ALL', 'Anchovy', '
        Sardine'], default='Anchovy', help='fishery')
    parser.add_argument('-d', '--date', default='today', help='a date')
    parser.add_argument('-v', '--verbose', help='enable verbose mode',
        action='store_true')
    parser.add_argument('-e', '--erase-files', help='erase temporary files
        ', action='store_true')
    parser.add_argument('-p', '--previous-day', help='calculate PFZ for
        previous date if not all data are available', action='store_true')
    args = parser.parse_args()

    if args.date == 'today':
        date = '{0}'.format(datetime.date.today().isoformat())
    else:
        try:
            date = '{0}'.format(datetime.datetime.strptime(args.date, '%Y
                -%m-%d').date())
        except ValueError:
            sys.exit('Incorrect date, should be YYYY-MM-DD or today (
                default)')

    fishery = list()
    if args.fishery == 'Anchovy':
        fishery.append('Anchovy')
    elif args.fishery == 'Sardine':
        fishery.append('Sardine')
    elif args.fishery == 'ALL':
        fishery.append('Anchovy')
        fishery.append('Sardine')

    verbose = False
    if args.verbose:
        verbose = True

    download_previous_day = False
    if args.previous_day:
        download_previous_day = True

    erase_files = False
    if args.erase_files:
        erase_files = True

    settings = config.settings
    fish_alert_directory = os.path.dirname(os.path.abspath(__file__))
    workspace_directory = config.settings['workspace']
    motu_path = config.settings['motu_path']
    username = config.settings['cmems_username']
    password = config.settings['cmems_password']
    snappy_path = config.settings['snappypath']

    if not os.path.exists(workspace_directory):
        os.makedirs(workspace_directory)

    os.chdir(workspace_directory)

    current_date_directory = os.path.join(workspace_directory, date)
    if not os.path.exists(current_date_directory):
```

```
        os.makedirs(current_date_directory)

    os.chdir(current_date_directory)

    if not os.path.isfile('bathymetry.nc'):
        if os.path.isfile('{0}/assets/bathymetry.nc'.format(
            fish_alert_directory)):
            copy('{0}/assets/bathymetry.nc'.format(fish_alert_directory),
                '.')
        else:
            sys.exit('Bathymetry file not available. Should be in assets/
                bathymetry.nc')

    depth_file = 'bathymetry.nc'
    chl_file = 'CHL.nc'
    sst_file = 'SST.nc'
    sla_file = 'SLA.nc'
    temp1_file = '_temp1.nc'
    temp2_file = '_temp2.nc'
    final_file = 'final.nc'

    downloader = Downloader(motu_path, username, password)
    if not os.path.isfile(chl_file):
        downloader.download(current_date_directory, chl_file, 'CHL', date,
            True)
    if not os.path.isfile(sst_file):
        downloader.download(current_date_directory, sst_file, 'SST', date,
            True)
    if not os.path.isfile(sla_file):
        downloader.download(current_date_directory, sla_file, 'SLA', date,
            True)

    # All environmental data are available
    if os.path.isfile(chl_file) and os.path.isfile(sst_file) and os.path.
        isfile(sla_file):
        # Collocate files into one
        collocator = Collocator(snappy_path)
        if not os.path.isfile(temp1_file):
            collocator.Collocate(depth_file, sst_file, temp1_file)
            Utilities.deleteCollocationFlags(temp1_file)
            time.sleep(1)
        if not os.path.isfile(temp2_file):
            collocator.Collocate(temp1_file, chl_file, temp2_file)
            Utilities.deleteCollocationFlags(temp2_file)
            time.sleep(1)
        if not os.path.isfile(final_file):
            collocator.Collocate(temp2_file, sla_file, final_file)
            Utilities.deleteCollocationFlags(final_file)
            time.sleep(1)

        # Create Fuzzifier using the collocated environmental data
        fuzzifier = Fuzzifier(final_file)
        # For each fishery
        for fish in fishery:
            # find the corresponding season
            season = date_to_season(date, fish)
            if season is not None:
                if os.path.isfile('{0}.nc'.format(fish)):
                    if verbose is True:
                        print 'PFZ for {0} on {1} already exists. Skipping
                            ...'.format(fish, date)
                    continue
                # run the fuzzification process
```

```
                fuzzifier.run(season, fish)
                # write the results to file
                fuzzifier.writeData('{0}.nc'.format(fish))
            else:
                if verbose is True:
                    print 'PFZ rules for {0} not available on {1}'.format(
                        fish, date)

    # Not all environmental data are available
    else:
        if verbose is True:
            print 'Not all environmental data are available for {0}. PFZ
                generation is not possible.'.format(date)

    # Delete temporary files if flag is set
    if erase_files is True:
        if verbose is True:
            print 'Deleting temporary files...'
        if os.path.isfile(chl_file):
            os.remove(chl_file)
        if os.path.isfile(sst_file):
            os.remove(sst_file)
        if os.path.isfile(sla_file):
            os.remove(sla_file)
        if os.path.isfile(depth_file):
            os.remove(depth_file)
        if os.path.isfile(temp1_file):
            os.remove(temp1_file)
        if os.path.isfile(temp2_file):
            os.remove(temp2_file)
        if os.path.isfile(final_file):
            os.remove(final_file)
        if verbose is True:
            print 'Temporary files deleted.'

'''
TODO
else:
        if download_previous_day:
            previous_date = (datetime.date.today() - datetime.timedelta(
                days=1)).isoformat()
            if verbose:
                print 'Calculating PFZ for {0}'.format(previous_date)
            if os.path.isfile('{0}.nc'.format(previous_date)):
                print 'PFZ is already calculated for {0}.'.format(
                    previous_date)
                sys.exit()

            chl_file = '{0}/{1}'.format(previous_date, 'CHL.nc')
            sst_file = '{0}/{1}'.format(previous_date, 'SST.nc')
            sla_file = '{0}/{1}'.format(previous_date, 'SLA.nc')

            if not os.path.isfile(chl_file):
                downloader.download(previous_date, 'CHL', 'CHL',
                    previous_date, True)
            if not os.path.isfile(sst_file):
                downloader.download(previous_date, 'SST', 'SST',
                    previous_date, True)
            if not os.path.isfile(sla_file):
                downloader.download(previous_date, 'SLA', 'SLA',
                    previous_date, True)
```

```
            if os.path.isfile(chl_file) and os.path.isfile(sst_file) and
                os.path.isfile(sla_file):
                temp1_file = '{0}/{1}'.format(previous_date, '_temp1.nc')
                temp2_file = '{0}/{1}'.format(previous_date, '_temp2.nc')
                final_file = '{0}/{1}'.format(previous_date, 'final.nc')
                depth_file = 'bathymetry.nc'

                collocator = Collocator(snappy_path)
                if not os.path.isfile(temp1_file):
                    collocator.Collocate(depth_file, sst_file, temp1_file)
                    Utilities.deleteCollocationFlags(temp1_file)
                    time.sleep(1)
                if not os.path.isfile(temp2_file):
                    collocator.Collocate(temp1_file, chl_file, temp2_file)
                    Utilities.deleteCollocationFlags(temp2_file)
                    time.sleep(1)
                if not os.path.isfile(final_file):
                    collocator.Collocate(temp2_file, sla_file, final_file)
                    Utilities.deleteCollocationFlags(final_file)
                    time.sleep(1)

                fuzzifier = Fuzzifier(final_file, Season.Summer, Fishery.
                    Anchovy)
                fuzzifier.run()
                fuzzifier.writeData('{0}.nc'.format(previous_date))
'''
```

Bibliography

[1] Barbara Neumann et al. "Future coastal population growth and exposure to sea-level rise and coastal flooding-a global assessment". In: *PloS one* 10.3 (2015), e0118571.

[2] Christopher J Crossland et al. *Coastal fluxes in the Anthropocene: the land-ocean interactions in the coastal zone project of the International Geosphere-Biosphere Programme*. Springer Science & Business Media, 2005.

[3] Eurostat regional Yearbook. *European Union, 2011*. 2011.

[4] Philippe Cury et al. "Small pelagics in upwelling systems: patterns of interaction and structural changes in "wasp-waist" ecosystems". In: *ICES Journal of Marine Science* 57.3 (2000), pp. 603–618.

[5] Jordi Lleonart and Francesc Maynou. "Fish stock assessments in the Mediterranean: state of the art". In: *Scientia Marina* 67.S1 (2003), pp. 37–49.

[6] Andrew Bakun. *Patterns in the ocean: ocean processes and marine population dynamics*. California Sea Grant in cooperation with Centro de Investigaciones Biologicas del Noroeste La Paz, Mexico, 1996.

[7] S Subramanian et al. "A Manual on The use of Potential Fishing Zone (PFZ) Forecast". In: (2014).

[8] Ian S Robinson. *Measuring the oceans from space: the principles and methods of satellite oceanography*. Springer Science & Business Media, 2004.

[9] Howard R Gordon. "Atmospheric correction of ocean color imagery in the Earth Observing System era". In: *Journal of Geophysical Research: Atmospheres* 102.D14 (1997), pp. 17081–17106.

[10] LM McMillin and DS Crosby. "Theory and validation of the multiple window sea surface temperature technique". In: *Journal of Geophysical Research: Oceans* 89.C3 (1984), pp. 3655–3661.

[11] CT Swift. "Passive microwave remote sensing of the ocean—A review". In: *Boundary-Layer Meteorology* 18.1 (1980), pp. 25–54.

[12] Fawwaz T Ulaby, Richard K Moore, and Adrian K Fung. "Microwave Remote Sensing Active and Passive-Volume II: Radar Remote Sensing and Surface Scattering and Enission Theory". In: (1982).

[13] Ad Stoffelen and David Anderson. "Scatterometer data interpretation: Estimation and validation of the transfer function CMOD4". In: *Journal of Geophysical Research: Oceans* 102.C3 (1997), pp. 5767–5780.

[14] Werner Alpers and Ingo Hennings. "A theory of the imaging mechanism of underwater bottom topography by real and synthetic aperture radar". In: *Journal of Geophysical Research: Oceans* 89.C6 (1984), pp. 10529–10546.

[15] Werner Alpers. "Theory of radar imaging of internal waves". In: *Nature* 314.6008 (1985), pp. 245–247.

[16] Dudley B Chelton et al. "Satellite altimetry". In: *International geophysics* 69 (2001), pp. 1–ii.

[17] J. C Walsh. *The Colonial Algae In Micrographia*. n.d. URL: http://www.micrographia.com/specbiol/alg/colo/colo0100.htm.

[18] Lindsey R. *What are Phytoplankton?* 2010. URL: http://earthobservatory.nasa.gov/Features/Phytoplankton/.

[19] WHOI. *Phytoplankton In Ocean Life*. 2014. URL: http://www.whoi.edu/main/topic/phytoplankton.

[20] L. Zimmerman. *Phytoplankton In Biological Resources*. n.d. URL: http://nerrs.noaa.gov/doc/siteprofile/acebasin/html/biores/phyto/pytext.htm.

[21] NOAA. *What are Phytoplankton?* 2014. URL: http://oceanservice.noaa.gov/facts/phyto.html.

[22] Robert G Wetzel. *Limnology: lake and river ecosystems*. Gulf Professional Publishing, 2001.

[23] Nicholas A Welschmeyer. "Fluorometric analysis of chlorophyll a in the presence of chlorophyll b and pheopigments". In: *Limnology and Oceanography* 39.8 (1994), pp. 1985–1992.

[24] Osmund Holm-Hansen et al. "Fluorometric determination of chlorophyll". In: *ICES Journal of Marine Science* 30.1 (1965), pp. 3–15.

[25] Min Chen et al. "A red-shifted chlorophyll". In: *Science* 329.5997 (2010), pp. 1318–1319.

[26] Thomas Lillesand, Ralph W Kiefer, and Jonathan Chipman. *Remote sensing and image interpretation*. John Wiley & Sons, 2014.

[27] Anatoly A Gitelson et al. "A simple semi-analytical model for remote estimation of chlorophyll-a in turbid waters: Validation". In: *Remote Sensing of Environment* 112.9 (2008), pp. 3582–3593.

[28] A Gitelson. "The peak near 700 nm on radiance spectra of algae and water: relationships of its magnitude and position with chlorophyll concentration". In: *International Journal of Remote Sensing* 13.17 (1992), pp. 3367–3373.

[29] Luoheng Han and Karen J Jordan. "Estimating and mapping chlorophyll-a concentration in Pensacola Bay, Florida using Landsat ETM+ data". In: *International Journal of Remote Sensing* 26.23 (2005), pp. 5245–5254.

[30] Arnold G Dekker, Tim J Malthus, and Ersin Seyhan. "Quantitative modeling of inland water quality for high-resolution MSS systems". In: *IEEE Transactions on Geoscience and Remote Sensing* 29.1 (1991), pp. 89–95.

[31] SW Myint and ND Walker. "Quantification of surface suspended sediments along a river dominated coast with NOAA AVHRR and SeaWiFS measurements: Louisiana, USA". In: *International Journal of Remote Sensing* 23.16 (2002), pp. 3229–3249.

[32] PD Wass et al. "Monitoring and preliminary interpretation of in-river turbidity and remote sensed imagery for suspended sediment transport studies in the Humber catchment". In: *Science of the Total Environment* 194 (1997), pp. 263–283.

[33] B Nechad, KG Ruddick, and Y Park. "Calibration and validation of a generic multisensor algorithm for mapping of total suspended matter in turbid waters". In: *Remote Sensing of Environment* 114.4 (2010), pp. 854–866.

[34] PJ Curran et al. "Multispectral remote sensing of nearshore suspended sediments: a pilot study". In: *International Journal of Remote Sensing* 8.1 (1987), pp. 103–112.

[35] EMM Novo, JD Hansom, and PJ Curran. "The effect of viewing geometry and wavelength on the relationship between reflectance and suspended sediment concentration". In: *International Journal of Remote Sensing* 10.8 (1989), pp. 1357–1372.

[36] Lian Feng et al. "Influence of the Three Gorges Dam on total suspended matters in the Yangtze Estuary and its adjacent coastal waters: Observations from MODIS". In: *Remote Sensing of Environment* 140 (2014), pp. 779–788.

[37] F Wang et al. "Applications of Landsat-5 TM imagery in assessing and mapping water quality in Reelfoot Lake, Tennessee". In: *International Journal of Remote Sensing* 27.23 (2006), pp. 5269–5283.

[38] K Osinska-Skotak, M Kruk, and M Mróz. "The spatial diversification of lake water quality parameters in Mazurian lakes in summertime". In: *Millpress: Rotterdam, The Netherlands* (2007).

[39] Robert M Cox Jr et al. "Assessing water quality in Catawba River reservoirs using Landsat thematic mapper satellite data". In: *Lake and Reservoir Management* 14.4 (1998), pp. 405–416.

[40] Jonathan W Chipman, Leif G Olmanson, and Anatoly A Gitelson. *Remote sensing methods for lake management: A guide for resource managers and decision-makers*. North American Lake Management Society, 2009.

[41] Guangyu Wu. "Seasonal Change Detection of Water Quality in Texas Gulf Coast Using MODIS Remote Sensing Data". In: *UC GIS Summer Assembly* (2003).

[42] Asif Mumtaz Bhatti et al. "Assessing the potential of remotely sensed data for water quality monitoring of coastal and inland waters". In: (2008).

[43] George R Aiken. *Humic substances in soil, sediment, and water: geochemistry, isolation, and characterization*. Vol. 1. Wiley-Interscience, 1985.

[44] Catherine A Corbett. "Colored Dissolved Organic matter (CDOM) workshop summary". In: (2007).

[45] Niklas Strömbeck and Donald C Pierson. "The effects of variability in the inherent optical properties on estimations of chlorophyll a by remote sensing in Swedish freshwaters". In: *Science of the total environment* 268.1 (2001), pp. 123–137.

[46] E Ortega-Retuerta et al. "Observations of chromophoric dissolved and detrital organic matter distribution using remote sensing in the Southern Ocean: Validation, dynamics and regulation". In: *Journal of Marine Systems* 82.4 (2010), pp. 295–303.

[47] Birgot Paavel et al. "Optical investigations of CDOM-rich coastal waters in Pärnu Bay". In: *Estonian Journal of Earth Sciences* 60.2 (2011), p. 102.

[48] Nickitas Georgas, Wei Li, and Alan F Blumberg. *Investigation of Coastal CDOM Distributions Using In-Situ and Remote Sensing Observations and a Predictive CDOM Fate and Transport Model*. Tech. rep. STEVENS INST OF TECHNOLOGY HOBOKEN NJ, 2009.

[49] Th Schroeder et al. "Remote sensing of apparent and inherent optical properties of Tasmanian coastal waters: application to MODIS data". In: *Proceedings of the XIX Ocean Optics Conference, Barga, Italy*. 2008, pp. 6–10.

[50] Nazanin Chaichi Tehrani et al. "Chromophoric dissolved organic matter and dissolved organic carbon from sea-viewing wide field-of-view sensor (SeaWiFS), moderate resolution imaging spectroradiometer (MODIS) and MERIS sensors: Case study for the northern gulf of mexico". In: *Remote Sensing* 5.3 (2013), pp. 1439–1464.

[51] SP Tiwari and P Shanmugam. "An optical model for the remote sensing of coloured dissolved organic matter in coastal/ocean waters". In: *Estuarine, Coastal and Shelf Science* 93.4 (2011), pp. 396–402.

[52] Taheri H Shahraiyni et al. "Application of the Active Learning Method for the estimation of geophysical variables in the Caspian Sea from satellite ocean

colour observations". In: *International Journal of Remote Sensing* 28.20 (2007), pp. 4677–4683.

[53] W. Brown. *Kinetic vs Thermal Energy*. 1999. URL: http://www.newton.dep.anl.gov/askasci/chem99/chem99045.htm.

[54] John R. Brett. "Energetic responses of salmon to temperature. A study of some thermal relations in the physiology and freshwater ecology of sockeye salmon (Oncorhynchus nerka)". In: *American Zoologist* (1971), pp. 99–113.

[55] WA Bennett and V Di Santo. "Effect of rapid temperature change on resting routine metabolic rates of two benthic elasmobranchs". In: *Fish Physiol Biochem. Springer Science* (2011).

[56] University of Illinois Extension. *Planting Aquatic Plants*. 2014. URL: http://urbanext.illinois.edu/watergarden/planting.cfm.

[57] Robert G. Wetzel. *Limnology: lake and river ecosystems*. Gulf Professional Publishing, 2001.

[58] K. R. Roussy. *Heat Transfer in Earth's Atmosphere*. 2006. URL: http://www.atmos.illinois.edu/earths_atmosphere/heat_transfer.html.

[59] Hou-Sung Jung et al. "Subset of heat-shock transcription factors required for the early response of Arabidopsis to excess light". In: *Proceedings of the National Academy of Sciences* 110.35 (2013), pp. 14474–14479.

[60] Gerard D McCarthy et al. "Ocean impact on decadal Atlantic climate variability revealed by sea-level observations". In: *Nature* 521.7553 (2015), pp. 508–510.

[61] Richard W Reynolds and Thomas M Smith. "Improved global sea surface temperature analyses using optimum interpolation". In: *Journal of climate* 7.6 (1994), pp. 929–948.

[62] David Anding and Richard Kauth. "Estimation of sea surface temperature from space". In: *Remote Sensing of Environment* 1.4 (1970), pp. 217–220.

[63] WJ Emery and Y Yu. "Satellite sea surface temperature patterns". In: *International Journal of Remote Sensing* 18.2 (1997), pp. 323–334.

[64] EF LeDrew and SE Franklin. "The use of thermal infrared imagery in surface current analysis of a small lake". In: *Photogrammetric Engineering and Remote Sensing* 51.5 (1985), pp. 565–573.

[65] Chuqun Chen, Ping Shi, and Qingwen Mao. "Application of remote sensing techniques for monitoring the thermal pollution of cooling-water discharge from nuclear power plant". In: *Journal of Environmental Science and Health, Part A* 38.8 (2003), pp. 1659–1668.

[66] RN Handcock et al. "Accuracy and uncertainty of thermal-infrared remote sensing of stream temperatures at multiple spatial scales". In: *Remote Sensing of Environment* 100.4 (2006), pp. 427–440.

[67] JF Vesecky et al. "Water surface temperature estimates using active and passive microwave remote sensing: Preliminary results from an outdoor wind-wave tank". In: *Geoscience and Remote Sensing Symposium, 1994. IGARSS'94. Surface and Atmospheric Remote Sensing: Technologies, Data Analysis and Interpretation., International*. Vol. 2. IEEE. 1994, pp. 1021–1023.

[68] FW Gilcreas. "Standard methods for the examination of water and waste water." In: *American Journal of Public Health and the Nations Health* 56.3 (1966), pp. 387–388.

[69] K Sommer and Spitzer. *Session H: Ocean Salinity in plenary lecture WMO-BIPM*. -. URL: http://www.bipm.org/ws/BIPM/WMO-BIPM/Allowed/Plenary_lectures/04.04-Session_H-Intro-Sommer.pdf.

[70] A Paytan. *Major Ions, Conservative Elements and Dissolved Gases in Seawater (Lecture in Marine Chemistry*. 2006. URL: http://ocean.stanford.edu/courses/bomc/chem/lecture_04.pdf.

[71] Victor Klemas. "Remote sensing of sea surface salinity: an overview with case studies". In: *Journal of Coastal Research* 27.5 (2011), pp. 830–838.

[72] Yann H Kerr et al. "The SMOS mission: New tool for monitoring key elements ofthe global water cycle". In: *Proceedings of the IEEE* 98.5 (2010), pp. 666–687.

[73] L Klein and C Swift. "An improved model for the dielectric constant of sea water at microwave frequencies". In: *IEEE Journal of Oceanic Engineering* 2.1 (1977), pp. 104–111.

[74] Gary SE Lagerloef, Calvin T Swift, and David M Le Vine. "Sea surface salinity: The next remote sensing challenge". In: *Oceanography* 8.2 (1995), pp. 44–50.

[75] Christophe Maes and David Behringer. "Using satellite-derived sea level and temperature profiles for determining the salinity variability: A new approach". In: *Journal of Geophysical Research: Oceans* 105.C4 (2000), pp. 8537–8547.

[76] Yan Bai et al. "Remote sensing of salinity from satellite-derived CDOM in the Changjiang River dominated East China Sea". In: *Journal of Geophysical Research: Oceans* 118.1 (2013), pp. 227–243.

[77] Th Schroeder et al. "Remote sensing of apparent and inherent optical properties of Tasmanian coastal waters: application to MODIS data". In: *Proceedings of the XIX Ocean Optics Conference, Barga, Italy*. 2008, pp. 6–10.

[78] Anclré Morel and Louis Prieur. "Analysis of variations in ocean color". In: *Limnology and oceanography* 22.4 (1977), pp. 709–722.

[79] Howard R Gordon and André Y Morel. *Remote assessment of ocean color for interpretation of satellite visible imagery: A review*. Vol. 4. Springer Science & Business Media, 2012.

[80] CURTIS D MOBLEY et al. "Optical modeling of ocean waters: Is the Case 1-Case 2 classification still useful?" In: *Oceanography* 17.2 (2004), pp. 60–67.

[81] F Dehairs et al. "The biological production of marine suspended barite and the barium cycle in the Western Mediterranean Sea". In: *Biogeochemistry* 4.2 (1987), pp. 119–140.

[82] John D Milliman and James PM Syvitski. "Geomorphic/tectonic control of sediment discharge to the ocean: the importance of small mountainous rivers". In: *The Journal of Geology* 100.5 (1992), pp. 525–544.

[83] MD Krom et al. "Phosphorus limitation of primary productivity in the eastern Mediterranean Sea". In: *Limnology and Oceanography* 36.3 (1991), pp. 424–432.

[84] Ljubomir Jeftic, John D Milliman, Giuliano Sestini, et al. "Climatic Change and the Mediterranean: environmental and societal impacts of climatic change and sea-level rise in the Mediterranean Region: Volume I". In: (1992).

[85] S Guerzoni, E Molinaroli, and R Chester. "Saharan dust inputs to the western Mediterranean Sea: depositional patterns, geochemistry and sedimentological implications". In: *Deep Sea Research Part II: Topical Studies in Oceanography* 44.3 (1997), pp. 631–654.

[86] Carol M Turley. "The changing Mediterranean Sea—a sensitive ecosystem?" In: *Progress in Oceanography* 44.1 (1999), pp. 387–400.

[87] N Pinardi and E Masetti. "Variability of the large scale general circulation of the Mediterranean Sea from observations and modelling: a review". In: *Palaeogeography, Palaeoclimatology, Palaeoecology* 158.3 (2000), pp. 153–173.

[88] Allan R Robinson et al. "Mediterranean sea circulation". In: *Ocean currents: a derivative of the Encyclopedia of Ocean Sciences* (2001), pp. 1689–1705.

[89] Maurizio Würtz. *Mediterranean pelagic habitat: oceanographic and biological processes, an overview*. IUCN, 2010.

[90] Claude Millot. "Circulation in the western Mediterranean Sea". In: *Journal of marine systems* 20.1 (1999), pp. 423–442.

[91] Alex Lascaratos et al. "Recent changes in deep water formation and spreading in the eastern Mediterranean Sea: a review". In: *Progress in oceanography* 44.1 (1999), pp. 5–36.

[92] Claude Millot and Isabelle Taupier-Letage. "Circulation in the Mediterranean sea". In: *The Mediterranean Sea* (2005), pp. 323–334.

[93] K David Hyrenbach, Karin A Forney, and Paul K Dayton. "Marine protected areas and ocean basin management". In: *Aquatic conservation: marine and freshwater ecosystems* 10.6 (2000), pp. 437–458.

[94] Vers N Agostini and Andrew Bakun. "Ocean triads' in the Mediterranean Sea: physical mechanisms potentially structuring reproductive habitat suitability (with example application to European anchovy, Engraulis encrasicolus)". In: *Fisheries Oceanography* 11.3 (2002), pp. 129–142.

[95] CMEMS. *MEDITERRANEAN SEA HIGH RESOLUTION AND ULTRA HIGH RESOLUTION SEA SURFACE TEMPERATURE ANALYSIS*. 2017. URL: http://cmems-resources.cls.fr/?option=com_csw&view=details&tab=info&product_id=SST_MED_SST_L4_NRT_OBSERVATIONS_010_004&format=docpdf.

[96] CMEMS. *MEDITERRANEAN SEA GRIDDED L4 SEA SURFACE HEIGHTS AND DERIVED VARIABLES NRT*. 2017. URL: http://cmems-resources.cls.fr/?option=com_csw&view=details&tab=info&product_id=SEALEVEL_MED_PHY_L4_NRT_OBSERVATIONS_008_050&format=docpdf.

[97] CMEMS. *MEDITERRANEAN SEA MONTHLY, 8-DAYS AND DAILY INTERPOLATED SURFACE CHLOROPHYLL CONCENTRATION FROM MULTI SATELLITE OBSERVATIONS*. 2017. URL: http://cmems-resources.cls.fr/?option=com_csw&view=details&tab=info&product_id=OCEANCOLOUR_MED_CHL_L4_NRT_OBSERVATIONS_009_041&format=docpdf.

[98] Pauline Weatherall et al. "A new digital bathymetric model of the world's oceans". In: *Earth and Space Science* 2.8 (2015), pp. 331–345.

[99] Marianna Giannoulaki et al. "Characterizing the potential habitat of European anchovy Engraulis encrasicolus in the Mediterranean Sea, at different life stages". In: *Fisheries Oceanography* 22.2 (2013), pp. 69–89.

[100] Maria Pilar Tugores et al. "Habitat suitability modelling for sardine Sardina pilchardus in a highly diverse ecosystem: the Mediterranean Sea". In: *Marine Ecology Progress Series* 443 (2011), pp. 181–205.

[101] Stylianos Somarakis et al. "Daily egg production of anchovy in European waters". In: *ICES Journal of Marine Science* 61.6 (2004), pp. 944–958.

[102] I Palomera et al. "Small pelagic fish in the NW Mediterranean Sea: an ecological review". In: *Progress in Oceanography* 74.2 (2007), pp. 377–396.

[103] Lotfi A Zadeh. "Fuzzy sets". In: *Information and control* 8.3 (1965), pp. 338–353.

[104] Ebrahim H Mamdani and Sedrak Assilian. "An experiment in linguistic synthesis with a fuzzy logic controller". In: *International journal of man-machine studies* 7.1 (1975), pp. 1–13.

[105] Lauritz P Holmblad. "Control of a cement kiln by fuzzy logic". In: *Fuzzy information and decision processes* (1982), pp. 389–399.