Early Earthquake and Tsunami Warning Viewer

Yicheng Shao (Eason)

October 7, 2024

Abstract

Give a brief summary outline of your project.

Contents

List of Tables

List of Figures

Listings

1 Analysis

Overview

This section introduces the background for the EEW system in place in Japan, consisting of EEW Forecasts and EEW Warnings, and together with the tsunami warnings. Concepts such as intensities, magnitude and epicentres are defined. Two key users and targets are identified, specifically passionate geologists and earthquake-sensitive industries, each having different requirements. A client in the former was interviewed, together with a thorough analysis and comparison of some existing solutions such as SREV, JQuake, KEVI and Quarog. A detailed requirement specification and the critical path is also outlined, consisting of DM-D.S.S. parsing, real-time monitoring, past-earthquake viewing and joint functionalities.

1.1 Background Information

1.1.1 The Early Earthquake Warning System

Earthquake is one of the most common natural disasters in the whole world, and direct consequences of earthquakes include tsunamis which could be catastrophic.

Japan, sitting on the intersection of the Eurasian, the Philippine and the North-American plates, is the countries with most earthquakes. Historically, the Great Kantō Earthquake in 1923, the Great East Japan Earthquake in 2011 (a.k.a. the Tōhoku Earthquake) and the recent 2024 Noto Peninsula Earthquake all caused hundreds of deaths, both due to the result of the earthquake(s) and the resulting tsunami.

To provide protection to its residents, the Japan Meteorological Agency (JMA), together with the National Research Institute for Earth Science and Disaster Resilience (NIED) placed thousands of **earthquake sensors** across Japan (the Hi-net, the K-NET, the KiK-net and the F-NET), with several of them lying deep in the sea bed, measuring displacement, velocity and acceleration, which are connected to multiple servers, including two located in Ōsaka and Tōkyo.

Using data obtained from the sensors, computers do some complicated algorithms (mentioned below) to send out **early earthquake warnings (EEWs)** automatically within milliseconds. There are two types of EEWs:

1. **EEW (Forecast).** Sent out to **highly-dependent industries** (e.g. rail industry, power plants) and **subscribed users**, when maximum intensity level of more than 3, or a magnitude of more than 3.5 is expected.

2. **EEW (Warning).** Sent out to **everyone** via TV, Radio, Mobile Phone, SMS, etc., when a maximum intensity level of more than 4 is expected.

After the earthquake, JMA staff will determine the location and severity of tsunami warnings to be issued, if necessary.

1.1.2 Earthquake Terminology

- **Intensity.** The intensity describes the intensity vibration of a point due to an earthquake. It is not unique to an earthquake - **different places can have different intensities** due to the distance to the epicentre, and intensity will also change over time. JMA measures intensity using **9 levels: 1, 2, 3, 4, 5–, 5+, 6–, 6+ and 7** in increasing order.

- **Magnitude/Scale.** The magnitude of an earthquake describes the energy released in the earthquake in a logarithmic scale. **It is unique to an earthquake.**

- **Epicentre/Hypocentre.** The epicentre is the surface point directly above the true centre of the earthquake.
- **Focal Depth.** The focal depth is the depth of the true centre of the earthquake.
- **P-Wave and S-Wave.** These are seismic waves, sourced from the true centre of the earthquake, travelling at different speeds, with Primary (P)-Wave travelling faster and Secondary (S)-Wave travelling slower.

1.2 Problem Area

The main goal of this application is to provide a visualisation of the earthquake/tsunami related data feed(s) provided by JMA's affiliated institution, Disaster Mitigation Data Send Service (DM-D.S.S). There are numerous apps providing a list of recent earthquakes, the real-time data measured by the sensors, and the real time earthquake warning displayed on a map, but rarely are there good apps that combine all those features together satisfyingly, with just the necessary features the author needs.

Some applications are no longer being updated due to change in the user's policy of the related data feed. Furthermore, most of the apps available are only in Japanese, not in English or my home language Chinese, which can create trouble for the author to understand.

1.3 Client and End User

The primary target of this application will be passionate geographers and geologists who are interested in the study of earthquake observations and predictions. The age group of this vary all the way from primary-school students to adults, including the author who has been amazed by the technology since the age of 12. They could take any employment, ranging from students to full-time jobs. Their proficiency usually varies, since there are people new to this field who probably does not have much knowledge, so the interface of the application should be relatively user-friendly and understandable, hiding unnecessary technical complexities.

Another target client could be industries which highly rely on earthquake predictions due to the risk imposed by earthquakes. High-speed railway and nuclear power plants are good examples of this. Therefore, the staff in charge monitoring will usually have higher proficiency and would like more detailed data of the earthquake. However, they will only need the necessary data from earthquakes happening close to them and only require intensity data of the point in interest (e.g. the power plant). To put this into content, an earthquake happening 1000 km away from them does not need to be fed into their system, while they would like to see the intensity of the shock and the arrival time of the seismic waves to decide the actions. In fact, the author really likes investigating on the rail industry, whose infrastructure could be greatly affected by earthquakes.

Table 1 compares relative features of these two target users/clients.

1.4 Research Methodology

1.4.1 Client Interview

The author interviewed my friend Wesley Ma, who is a passionate geologist on earthquake studies and also monitor earthquakes regularly.

1. *Which earthquake monitoring apps do you use?*

   Response: JQuake, SREV, Quarog, KEVI, Kyoshin-Monitor (Support discontinued), Kiwi Monitor (Support discontinued)

Feature	Primary Target	Secondary Target
Description	Passionate Geologists	Earthquake-Sensitive Industries
Age	Varies (Middle School – Adults)	Work Age
Reason	Monitor Live Latest Earthquakes	Monitor Risks to Infrastructure
Proficiency	Varies (Beginner – Amateur)	Trained Professional
General Requirements	Monitor Overall Movement	Alert about intensity and arrival time at specific points

Table 1: Comparison of target users and clients

2. *Do you subscribe/pay to services such as DM-D.S.S. to use earthquake monitoring apps, and do you think it is worth the price?*

   Response: Yes. However, DM-D.S.S. is a little expensive. However, the price becomes more affordable considering the information provided by the subscription.

3. *Do you watch YouTube livestreams on earthquake monitoring?*

   Response: I do not usually watch the live streams, as almost no one who has already has a monitoring app will use the stream. They provide mostly the same information as the applications, just real-time streaming the windows.

4. *Why do you use earthquake monitoring apps?*

   Response: To monitor the earthquake. This is derived from my interest in broadcasting culture in Japan. This eventually led me to be intrigued with the development of earthquake monitoring technologies and theories in Japan.

5. *How often do you use earthquake monitoring apps (e.g. all the time/after school/only after big earthquakes)?*

   Response: After big earthquakes. But I usually open one or two apps for all-day monitoring to catch potential major (or medium) earthquakes.

6. *Describe the advantages and disadvantages of each of them, mentioning the specific features.*

   Response:

   - SREV is a good one, but it is only available in a browser with no app.
   - Kyoshin-Monitor and Kiwi Monitor are relatively more stable, but the source is not the same as that from the former two apps, and its support is also discontinued.
   - Quarog has weak response time and interacting interface.
   - JQuake is the most developed app, which includes nearly all functions that can be thought of. However sometimes the connection of WebSocket is unstable.
   - KEVI does not have the sound files configured by default. Some information are not displayed clearly enough.

7. *What features do you use the most/least?*

   Response: Basic earthquake notifications, and should be including sufficient and prompt information.

8. *What features are redundant in the earthquake monitoring apps you use?*

   Response: KEVI has a weather monitoring function, which could be redundant. But that could be caused by the different purposes of the app. Hence, no further comments. It is not a bad thing.

9. *What are the critical features of an earthquake monitoring app?*

   Response: To provide accurate, prompt and detailed information according to the source released by the JMA, the information display interface should be easy to read and understand. This is not only helping the people who like to monitor, but more importantly, provides the easiest way to the people who really need to seek information to minimise the harm brought by earthquakes and successive disaster.

10. *What additional features would you like to have in those existing apps?*

    Response: Summarise all the useful features from different apps into one app. Stable connection. Nothing else.

1.4.2 Existing Applications and Solutions

Based on the applications the author uses and the feedback from interviewee, there are the following commonly-used applications:

- JQuake
- Scratch Real-time Earthquake Viewer (SREV), available at kotoho7/scratch-realtime-earthquake-viewer-page in compiled form
- Kyoshin EEW Viewer for Ingen (KEVI), available at ingen084/KyoshinEewViewerIngen
- Quarog

Supported platforms of those apps are listed in Table 2. In particular, note that SREV is a web-based and GitHub Pages-hosted application therefore supporting all platforms. KEVI is written in .NET Framework and supports the second most platforms, with JQuake not supporting Linux and Quarog only supporting Windows.

Platform	JQuake	SREV	KEVI	Quarog
Windows	✓	✓	✓	✓
macOS	✓	✓	✓	
Linux	✓	✓	✓	
Android/iOS		✓		

Table 2: Supported platforms of existing solutions

An overview feature table of monitoring is analysed in Table 3. In particular, note that due to the nature of SREV being Scratch-programmed and web-based, it reached a special agreement with DM-D.S.S. to use the API without the need of all users paying for this, since it is hard to integrate such function into a web application.

Quarog is a relatively new application and only supports basic functionalities of EEW Viewing and past earthquake listing, as shown in Figure 2. JQuake is solely dedicated to earthquake and tsunami monitoring as shown in Figure 1, while KEVI has features like rain clouds map and

natural disaster warning which is beyond the scope of this analysis (which is a burden to users only requiring earthquake monitoring and a waste of storage space).

It is worth noting that for SREV, DM-D.S.S. is integrated with special permission with no login required. For JQuake, although it has tsunami warnings/special warnings, the tsunami forecast is not available.

Feature	JQuake	SREV	KEVI	Quarog
NIED Real-time Shake Support	✓	✓	✓	
DM-D.S.S. WebSocket Support	✓	✓	✓	✓
Real-time Sensor Data	✓	✓	✓	
Vibration Alert	✓	✓	✓	
Past Earthquake List	✓	✓	✓	✓
Past Earthquake Details		✓	✓	✓
Tsunami Warning	✓	✓	✓	
Real-time EEW	✓	✓	✓	✓
Calculated Seismic Wavefronts	✓	✓	✓	✓
User-Defined Key Monitor Point	✓		✓	
Sub-Map for the Okinawa Area	✓		✓	
Replay	✓		✓	

Table 3: Feature comparison in monitoring of existing solutions

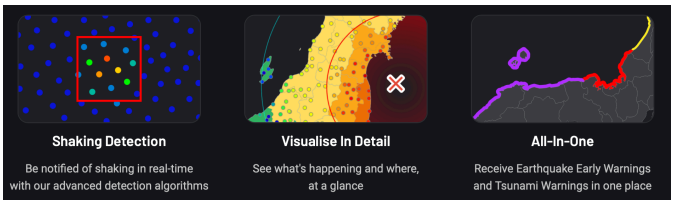

Figure 1: Feature introduction of JQuake, screenshot from website.

There are also a variety of configuration options available for all apps, as listed in Table 4. Both KEVI and Quarog supports the adjustment of the colour theme, and Quarog even supports changing the style of how blocks are displayed and coloured as shown in Figure 3 and 4. Playing a sound on the speaker is also common among the apps to remind the user of earthquakes.

Feature	JQuake	SREV	KEVI	Quarog
DM-D.S.S. Login	✓		✓	✓
Sound Alert	✓	✓	✓	✓
System Notification			✓	
Colour Theme			✓	✓
Map Colouring Style		✓		✓

Table 4: Feature comparison in configuration and customisability of existing solutions

Figure 2: Feature introduction of Quarog, screenshot from website. Top-left: Past earthquake information; Top-right: Real-time EEW; Bottom-Left: Past earthquake list; Bottom-Right: Details of EEW.

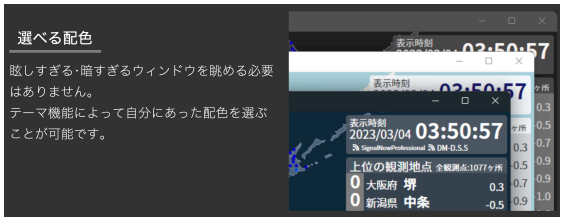

Figure 3: Customisable colour scheme of KEVI, screenshot from website.

1.5 Features of proposed solution

Based on the potential user/client interview results, and the research into the existing solutions, the following key features should appear in the solution, since they are essential to earthquake monitoring applications:

1. DM-D.S.S. Login functionality (for the data source)
2. Real-Time Sensor Shake Intensity Data (w/ vibration alert)
3. EEW Visualisation (w/ Calculated Seismic Wavefronts)

Figure 4: Customisability of Quarog, screenshot from website.

4. Past Earthquake List (w/ Option to review details)
5. Tsunami Warning Visualisation (w/ Related Sounds)

However, due to the limitation of time and the difficulty of implementation, the following functionalities are not the key to implementation, which include mostly the customisation parts, ranked in decreasing importance:

1. User-Defined Key Monitor Point
2. Customisable Sound Alert
3. Customisable Colour Theme
4. Customisable Map Colouring Style

The following features might not be available due to the time constraints and the complexity behind such system:

1. Replay (due to a server needing to store past data)
2. Sub-Map for the Okinawa Area (due to the difficulty in implementation)
3. Map Zooming Feature (due to the complexity in the map colouring functionality)

With those key features implemented, the application should mirror all essential functionalities of an earthquake monitoring application, as compared to the four existing solutions above.

1.6 Critical Path

The functionality of this application can essentially be divided into $1 + 2 + 1 + n$ steps, where the 1 is to receive information from the API/WebSocket provided by DM-D.S.S. and NIED, and 2 is to, briefly saying:

1. Real-time Monitoring: Produce a real-time map to monitor real time vibration and plot real-time EEW/tsunami warnings
2. Past-earthquake Viewing: Produce a menu to select an earthquake to display information on the map.

The next 1 is to merge these two functions, specifically the functionality to switch back to the real-time monitoring option immediately when a new EEW is released (to make sure the user does not miss any information on real-time EEWs while looking at past earthquake information).

The final n is to implement a setting page for customisation, and some extra additional features.

Part 1. NIED and DM-D.S.S. API/WebSocket Connection

1. Investigate into the list of APIs/WebSockets that should be used by the application.
2. Implement classes/DTOs to convert the information into C# objects.
3. Implement classes to transfer real-time WebSocket XML/JSON to the classes and DTOs.
4. Use simple API Key to test the functionality of this sub-system.
5. Implement OAuth 2 login to reduce complexity and increase security.

Part 2a. Real-time Monitoring Map

1. Implement a Japan map in the application.
2. Achieve and store the locations of the monitoring points in Japan, using necessary information from JMA, NIED and DM-D.S.S.
3. Colour the monitoring points using a colour scheme on the map.
4. Investigate into and implement an algorithm to detect shake in a certain area of the map.
5. Achieve and store the names of areas of earthquake epicentres from JMA.
6. Investigate into and implement an algorithm to calculate seismic wave fronts.
7. Plot and display real time EEWs, considering special cases such as:
   - Cancellation of EEW;
   - Upgrade from EEW Forecast to EEW Warning;
   - Multiple Earthquakes (hence displaying epicentres with labels).
8. Implement functionality of colouring areas on the map with the maximum expected intensity.
9. Achieve and store the names of shorelines from JMA.
10. Plot and display real time tsunami warnings.

Part 2b. Past-earthquake Viewing

1. Implement a side-list of a list of path earthquakes.
2. Provide a button to view the detailed information on the earthquake (e.g. magnitude, epicentre, time).
3. Implement functionality to plot the detected maximum intensities on the map, from where they are detected.
4. Provide external link to view earthquake in the JMA website.

Part 3. Joint functionality

1. Provide sidebar to switch between real-time monitoring and past-earthquake viewing.
2. Implement automatic functionality to switch back to real-time earthquake monitoring when a shake over a certain magnitude is detected, or an EEW/Tsunami Warning is being published.

Part 4. Setting page for customisation and some more

1. Implement customisable voice and sound playing (e.g. when EEW (over certain magnitude) is published, when an earthquake information detail is received, etc.)
2. Implement customisable colour scheme for the colouring of different intensity scales, with several built in default.
3. Implement key monitoring point with special warnings when an earthquake with more than a certain intensity is expected to hit that point.
4. Implement a map-zooming feature.
5. Implement a sub-map for the Okinawa Area.

1.7 Requirements Specification

A detailed specification is defined here that is to be aimed to be fulfilled at the end of the project, split into four sections:

1. **NIED and DM-D.S.S. Functionality** – Corresponding to **Part 1**, in Table 6
2. **Real-Time Earthquake Monitoring** – Corresponding to **Part 2a**, in Table 7
3. **Past-earthquake Viewing** – Corresponding to **Part 2b**, in Table 8
4. **GUI Design** – Corresponding to **Part 3, 4**, in Table 9

The objectives here are SMART, meaning they are specific, measurable, achievable, realistic and timely.

Note that those marked with an * means they are optional.

Table 5 for types of testing. [M] afterwards will stand for a necessary manual testing (i.e. specific user inputs), while (M) will stand for supplementary manual testing (i.e. will have stand-alone tests as well as manual tests).

Test Method	Abbr.
Unit Testing	UT
Integrated Testing	IT
Performance Testing	PT

Table 5: Measurement methods

Req. №	Description	Success Criteria	Testing
1(i)	Login to DM-D.S.S. using an API Key	Login successful	UT [M]
1(ii)	Call HTTP-Based APIs	Successful calls	UT (M)
1(iii)	Connect to WebSocket Data Feed	Successful calls	UT [M]
1(iv)	Obtain stable connection on WS	Connected for 30min	PT (M)
1(v)	Successfully parse JSON and XML into C# objects	Information successfully parsed	UT
1(vi)	Exception handling for incorrect JSON and XML	Exceptions thrown	UT
1(vii)	Exception handling for failed connections	Exceptions thrown	UT (M)
1(viii)	Integrated functionality from login to providing objects	Correct objects created	IT [M]
1(ix)*	Login to DM-D.S.S. using OAuth2	Login successful	UT (M)

Table 6: Requirements for Part 1

Req. №	Description	Success Criteria	Testing
2a(i)	Display real time shake-intensities on the map	Correct coloured points at correct locations	IT
2a(ii)	Implement algorithm to display shake detection on the map	Use squares to indicate shake detected in area	UT, IT, PT
2a(iii)	Display the time at the corner of the UI	Time displayed with less than 100ms error	UT
2a(iv)	Display real-time EEW when issued	Epicentre at correct position	IT
2a(v)	Update/Cancel EEW when appropriate	Correctly updated and plotted	IT
2a(vi)	Calculate and display seismic wavefronts by algorithm	Calculated and plotted without significant delay	UT, IT, PT
2a(vii)	Colour map with expected maximum intensity of EEW	Correctly coloured	UT, IT
2a(viii)	Display the exp. magnitude, location, depth and intensity when EEW issued	Correctly formatted and displayed	UT, IT
2a(ix)	Provide additional EEW information (e.g. detailed time, algorithm used) when prompted	Correctly formatted and displayed	IT [M]
2a(x)	Display tsunami warnings when issued	Coloured at correct location	UT
2a(xi)*	Display real-time shake data at a user-defined point	Display the name of the point with acceleration and intensity	IT

Table 7: Requirements for Part 2(a)

Req. №	Description	Success Criteria	Testing
2b(i)	Display past-earthquake side list using data from DM-D.S.S.	Displayed and updated	UT, IT
2b(ii)	Display the time, location, depth, intensity and magnitude of earthquake on the side list	Correctly formatted and displayed	UT, IT
2b(iii)	Colouring of the map by the maximum intensity of an earthquake when prompted	Correctly coloured	UT, IT
2b(iv)	Display details of shake and other additional information provided by JMA when prompted on a sub-window	Correctly formatted and displayed	UT, IT
2b(v)	Provide link to external weather services for details	Linked to correct website and earthquake	UT [M]
2b(vi)	Provide link to JMA Earthquake details	Linked to correct earthquake	UT [M]

Table 8: Requirements for Part 2(b)

Req. №	Description	Success Criteria	Testing
3(i)	Provide sidebar to switch between real-time, past earthquake and settings	Sidebar functioning	IT [M]
3(ii)	Switch to real-time monitoring map when EEWs are issued	Switch with delay less than 1s	IT, PT
3(iii)	Provide an easy-to-use GUI	Potential user gives rating of more than 7 out of 10	[M]
3(iv)*	Provide page to design custom colour scheme	Colour scheme designed and applied to the whole UI	UT
3(v)*	Play sound when events happen and provide option to customise sound files	Correct sound played when required	UT, IT
3(vi)*	Provide option to define a point on the map to monitor	Option provided and meet specs defined in Part 2a	UT, IT
3(vii)*	Provide zooming feature on map	Map functioning with specs in Part 2	UT, IT
3(viii)*	Provide sub-map to display Okinawa Area	Map functioning with specs in Part 2	UT, IT

Table 9: Requirements for Part 3

2 Design

Overview

This section includes a breakdown of the application into sections of data processing, GUI functionalities and joint functionalities. The use of external sources including DM-D.S.S. and NIED data sources is discussed, together with the relevant formats (XML, JSON) and the objects related. A UML class diagram is included to discuss OOP relations of classes, records (record classes) and enums including use of inheritance, composition, association and aggregation. They also implement different interfaces. An outline of the design of the user interface is included. The expected hardware requirements of the systems are also listed, but any laptop with an up-to-date operating system (running Windows or macOS) should be able to run the program.

2.1 Data Structures/Data modelling

2.1.1 External Data Sources

There are two data sources this program will use: the NIED and the DM-D.S.S. Specifically, the former one is used to achieve the real-time shake data of the sensor points which were set up by the government (whose data is free to use), and the latter one is used to achieve past earthquake information and EEW information sent out by the JMA (which is pay-to-use). Note that DM-D.S.S. does also provide the real-time intensity data of the observation points, however it is pay-to-use only for companies and institutions on request. Therefore, it will not be feasible to use this data source in the program since one of and the principle target users is people passionate in monitoring earthquakes.

NIED Data Source As mentioned before, NIED has numerous 'earthquake observation nets' across Japan. Specifically, there is the K-NET and the KiK-net, which is dedicated to the observation of strong seismic motion. The K-NET consists of approximately 1000 sensors located across Japan, while the KiK-net also includes some sensors which are located within the earth, which will often have different readings compared to those located on the surface. They are extremely capable of detecting strong motion of ground. Furthermore, the K-NET and the KiK-net provides real-time intensity data webpage of two types, the 'Kyoshin' monitor and the long-period ground motion (LPGM) monitor (not working at the time of investigation). The Hi-net stands for high-sensitivity seismograph network, and it is dedicated for observation of minor motions of the ground. They release the waveforms to those who are researching seismic movements. As for the F-net which stands for the Full Range Seismograph Network of Japan, which is used to analyse the mechanism of a certain earthquake by analysing movements. None of the three nets provide a real-time API data feed.

Having compared the functionalities described above of the K/KiK-net, Hi-net and F-net and how they feed the data sources, the most suitable data source to reflect real-time motion of ground movements will be the **K/KiK-net**'s data feed, since it detects strong ground movements and is available real-time for the purpose of the application. (This is also the data source that JQuake and KEVI use in fact.)

In fact, in addition to these three networks, there are also the S-net, the DONET and the N-net, which detects the ground seismic movements in the sea. These data were adapted by SREV, but this is beyond the scope of this NEA analysis.

A comparison from the official website of MOWLAS (Monitoring of Waves on Land and Seafloor) of the three nets are included in Figure 5 and a map of the distribution of the sensors are included in Figure 6.

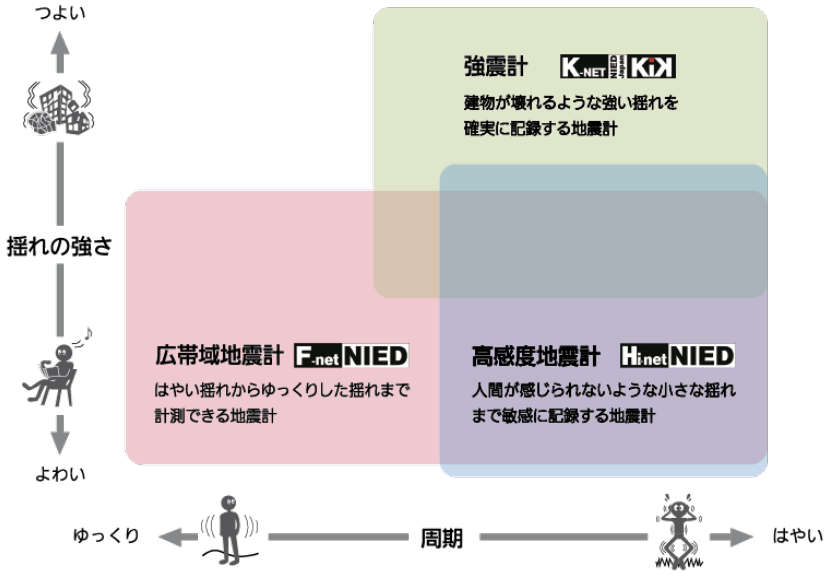

Figure 5: A comparison of the K-NET, F-net and Hi-net.

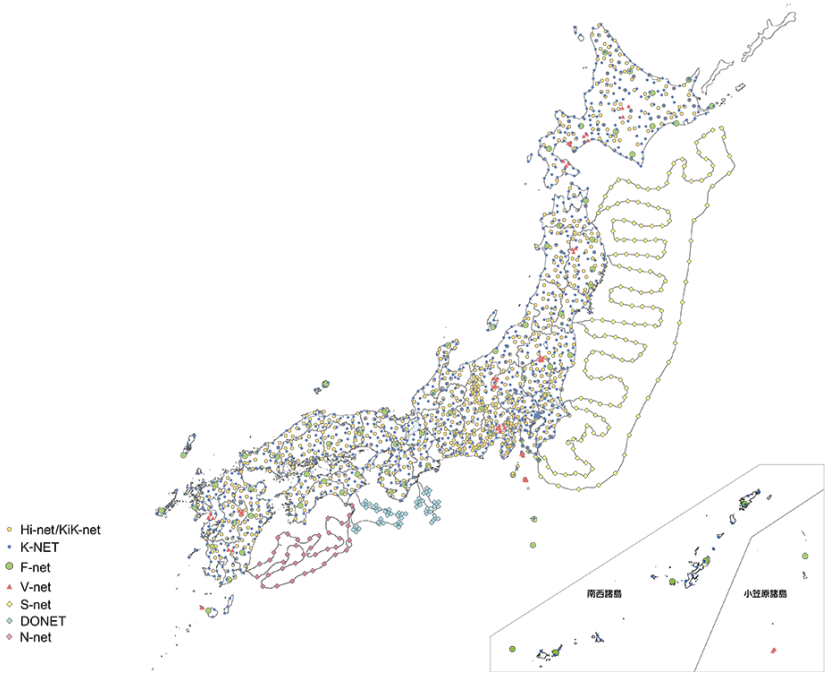

Figure 6: Distribution of the sensors of different nets.

Achieving image format data source The data source fed by the 'Kyoshin' Monitor is split into 8 types (detailed below in Table 10), and each type split into 2 types of data sources, surface sensors and borehole (earth) sensors, with codes in Table 11. The link to the GIF image

is in the following format:

`http://www.kmoni.bosai.go.jp/data/map_img/RealTimeImg/[#1]_[#2]/[yyyyMMdd]/[yyyyMMdd][hhmmss].[#1]_[#2].gif`

In the link, the `[yyyyMMdd]` and the `[hhmmss]` part should be replaced with the date and time respectively (in JST, UTC+8), and the `#1` replaced with the codes detailed below for the data types, and `#2` replaced with the codes detailed below for data sources. An example of the imaged achieved is in Figure 7.

Data Type	Description/Meaning	Code in `#1`
Real-time Shindo	Real-time Measured Intensity	`jma`
PGA	Peak (Maximal) Ground Acceleration	`acmap`
PGV	Peak (Maximal) Ground Velocity	`vcmap`
PGD	Peak (Maximal) Ground Displacement	`dcmap`
Response 0.125Hz	Response spectrum for 0.125Hz PGV	`rsp0125`
Response 0.250Hz	Response spectrum for 0.250Hz PGV	`rsp0250`
Response 0.500Hz	Response spectrum for 0.500Hz PGV	`rsp0500`
Response 1.000Hz	Response spectrum for 1.000Hz PGV	`rsp1000`
Response 2.000Hz	Response spectrum for 2.000Hz PGV	`rsp2000`
Response 4.000Hz	Response spectrum for 4.000Hz PGV	`rsp4000`

Table 10: Data available in 'Kyoshin' monitor

Sensor Type	Description/Meaning	Code in `#2`
Surface	K-NET and KiK-net sensors	`s`
Borehole	KiK-net sensors within earth	`b`

Table 11: Sensors available in 'Kyoshin' monitor

Extracting colour for each observation point from image Unfortunately, it seems to the author (and is widely accepted in the EEW monitoring app development society) that the position of the points (squares) on the image does not follow any significant pattern of position, i.e. there is no obvious conversion of coordinates to us from the official longitude/latitude locations to the positions on the image. Therefore, a manual conversion one-to-one mapping has to be developed.

NIED does have an official released list of observation points, which include their names and positions. This list has around 1700 of those observation points. However, in the actual image (like those in Figure 7), there are only 1000 of those in use in real time, consistent with K-NET's official introduction, and the rest 700 of those are invalid observation points. Therefore, it will be worth removing them from the list of earthquake monitoring points, before attempting to make the dictionary.

Unfortunately, 1000 is still quite a lot for us to deal with. Luckily, Ingen who used a similar approach to develop the KEVI application has already made such a mapping inside his open-source application in the file ShindoObsPoints.mpk.lz4 within ingen084/KyoshinEewViewerIngen, and even developed an editor for this at ingen084/KyoshinShindoPlaceEditor.

Due to the limited time for this NEA, the author will primarily use the pre-determined observation points for the K-net, and will use the existing application to map the points for KiK-net, which is still a considerable amount of work, but significantly less.

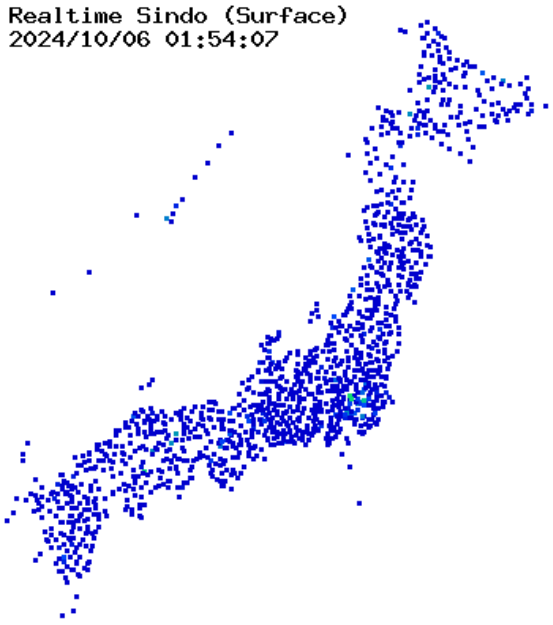

Figure 7: Sample GIF image achieved from 'Kyoshin' Monitor

This paragraph referred to this blog article written by Ingen.

Converting colour back to number format for further processing The true numerical data does not seem fully necessary at the first glance (since we might just as well just achieve the colour from the image and just plot them on the map, without the need to convert to a colour and back). However, for us to detect the shake in certain regions, it is necessary for us to achieve the numerical value to run the algorithm on it. Nevertheless, it is just good to have the number for us to have the numerical value for potential future developments.

As shown in Figure 8, the NIED 'Kyoshin' Monitor does indeed provide a scale of colours and reference to numerical values. However, there is a chance that a certain colour is not 'exactly' mapped on the scale, and further concerning that it is very slow and difficult to 'loop over' a colour legend, it is necessary to have an algorithmic-approach (numerical mapping-based approach) to map the colours in the colour space to numerical values (and back) is necessary.

Notice that the scale for PGA/PGV/PGD follow a logarithmic scale, while measured intensity follows a linear scale (though noting that the way intensity and magnitude is calculated is logarithmic as well). Therefore, if we normalise the vertical distance from the bottom of the axis h to $0 \leq h \leq 1$ (i.e. $h = 0$ at the bottom of the scale, $h = 1$ at the top of the scale), and if we denote intensity using I in JMA scale, PGA as a in gal, PGV as v in cm per second, and PGD

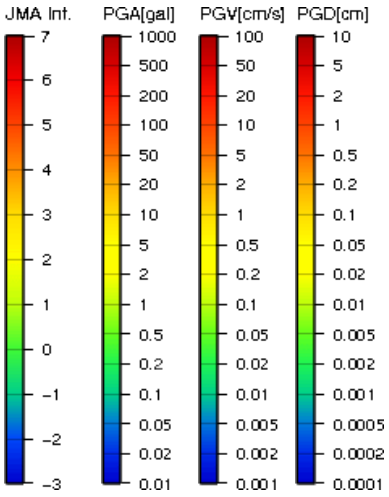

Figure 8: Scale colours of different measurements

as s in cm, from the scale, the following transforming formulae obviously hold:

$$\begin{aligned} I = 10h - 3 &\iff h = \frac{I+3}{10}, \\ a = 10^{5h-2} &\iff h = \frac{\lg a + 2}{5}, \\ v = 10^{5h-3} &\iff h = \frac{\lg v + 3}{5}, \\ x = 10^{5h-4} &\iff h = \frac{\lg x + 4}{5}. \end{aligned}$$

However, it is worth noting that NIED did use $1, 2, 5, 10$ on the logarithmic scale at equal intervals, so it is not a perfect logarithmic scale. The author is unsure why they designed the scale like this, nor if it's an intended approximation. Nevertheless, the logarithmic scale is a good enough approximation.

The next step is to develop a mapping from this colour space $\mathcal{C}$ to h, which of course should be invertible. Denote this as $f : [0, 1] \to \mathcal{C}$.

We consider using a suitable base to decompose $\mathcal{C}$. The colour of the given scale is an immediate suggestion to use a base containing **hue**, which in fact is designed to describe how human perceive colour, and unlike RGB and CMYK which uses principle colours to describe colour. A hue scale is shown in Figure 9.

Figure 9: The hue scale in HSL/HSV encoding

Therefore, a colour in the colour space $\mathcal{C}$ can be represented as a 3-D vector $\mathcal{C} \ni C = (H, S, V)$,

where $H \in [0, 360)$ in degrees is the hue value, $S \in [0, 1]$ stands for the saturation, and $V \in [0, 1]$ stands for the value (a brightness). And hence we will be able to decompose f into three components $f = (f_H, f_S, f_V)$.

Figure 10 plots the values of H, S and V against h (this is the graph of f and its components) of discrete values of h, and depending on the result we will attempt some fit/regression to a suitable function.

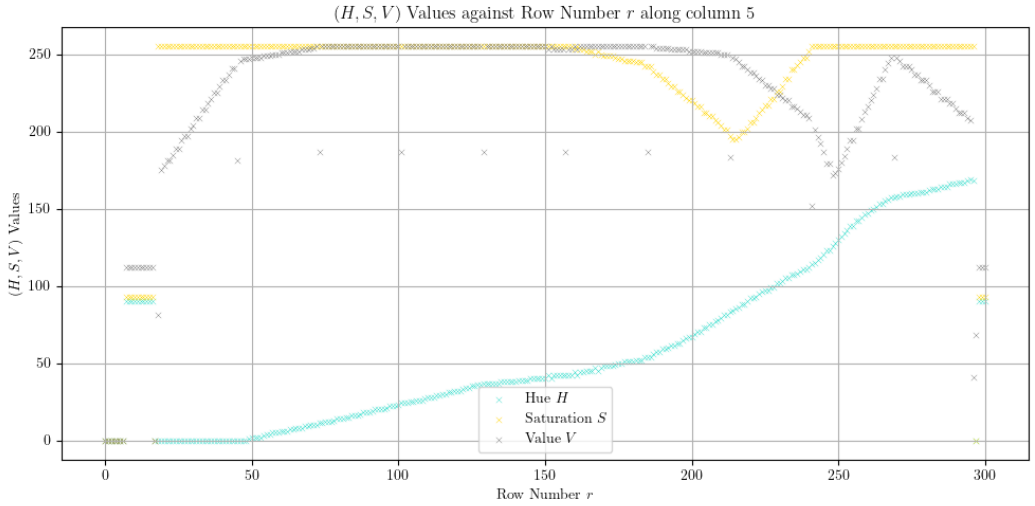

Figure 10: The values of (H, S, V) against pixel row r

Notice that in this plot, all values of (H, S, V) in fact range from 0 to 255.

It is worth noting that the scale has some space on the top (to show the type), and some space at the bottom. Notice that when the row $r = 17$ and $r = 297$ have values significantly different, so we extract the rows $r = 18$ and $r = 296$ to correspond (linearly) to $h = 1$ and $h = 0$, i.e.,

$$h = 1 - \frac{r - 18}{278}.$$

Figure 11 shows the result of this transformation being applied.

From here onwards, all values of (H, S, V) will be adjusted to be within the range which they should be in, i.e. $H \in [0, 360), S \in [0, 1], V \in [0, 1]$.

We consider finding f_H first, which is the cyan line. Notice that its trend can be split into 4 parts:

- $h \in [0, 0.1]$: linear;
- $h \in [0.1, 0.6]$: curving, ideally a cubic;
- $h \in [0.6, 0.9]$: linear;
- $h \in [0.9, 1]$: constant (0).

Furthermore, boundary conditions in Table 12 are applied to ensure that the function is continuous and nicely-behaving while matching the existing data. We use the following function

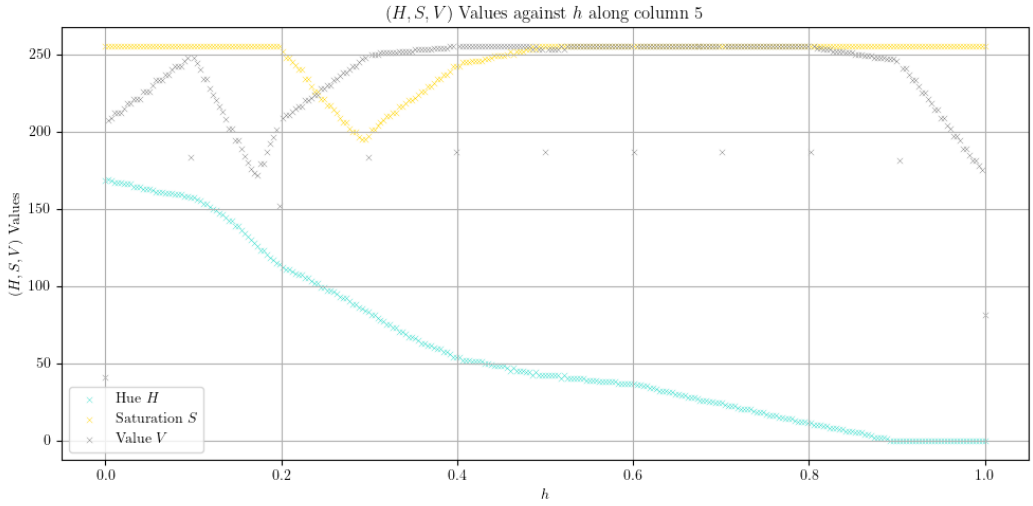

Figure 11: The values of (H, S, V) against normalised height h

h	$H = f_H(h)$
0	237
0.1	222
0.6	51
0.9	0
1	0

Table 12: Initial values for f_H

to apply the fit:

$$f_H(h) = \begin{cases} -150h + 237, & h \in [0, 0.1], \\ \odot, & h \in [0.1, 0.6], \\ -170h + 153, & h \in [0.6, 0.9], \\ 0, & h \in [0.9, 1]. \end{cases}$$

Here,

$$\begin{aligned} \odot = & \frac{222 \cdot (h - 0.3) \cdot (h - 0.4) \cdot (h - 0.6)}{(0.1 - 0.3) \cdot (0.1 - 0.4) \cdot (0.1 - 0.6)} \\ & + \frac{y_1 \cdot (h - 0.1) \cdot (h - 0.4) \cdot (h - 0.6)}{(0.3 - 0.1) \cdot (0.3 - 0.4) \cdot (0.3 - 0.6)} \\ & + \frac{y_2 \cdot (h - 0.1) \cdot (h - 0.3) \cdot (h - 0.6)}{(0.4 - 0.1) \cdot (0.4 - 0.3) \cdot (0.4 - 0.6)} \\ & + \frac{51 \cdot (h - 0.1) \cdot (h - 0.3) \cdot (h - 0.4)}{(0.6 - 0.1) \cdot (0.6 - 0.3) \cdot (0.6 - 0.4)}. \end{aligned}$$

Here, m_1 is the gradient of the line for $h \in [0, 0.1]$, $y_1 = f_H(0.3), y_2 = f_H(0.4)$ for $h \in [0.1, 0.6]$ (using Lagrange Polynomial), and the equation between $h \in [0.6, 0.9]$ is in fact fixed due to the initial conditions.

By applying a curve fit to the original data, the following results are obtained:

$$(y_1, y_2) = (115, 79.5).$$

Plotting H and $f_H(h)$ against h gives us Figure 12, which is decent.

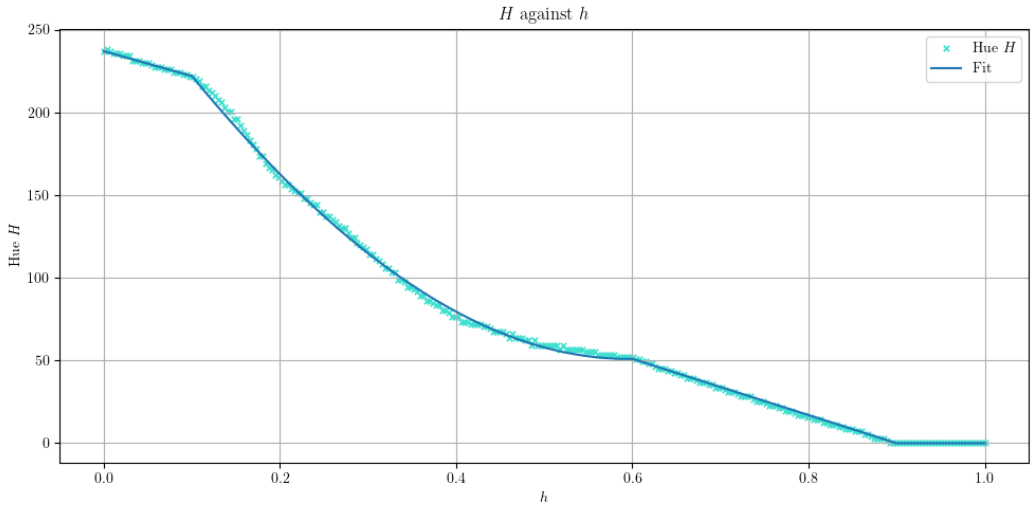

Figure 12: The fit result for $f_H : h \mapsto H$

As for $f_S(h)$, the obvious thing to do is to split it into 5 (4) piecewise functions, specifically $f_S = 1$ for $h \in [0, 0.2] \cup [0.5, 1]$, and three linear functions for $h \in [0.2, 0.29], h \in [0.29, 0.4]$ and $h \in [0.4, 0.5]$. Initial values are included in Table 13.

h	$S = f_S(h)$
0	1
0.2	1
0.29	0.765
0.4	0.95
0.5	1
1	1

Table 13: Initial values for f_S

This gives us that

$$f_S(h) = \begin{cases} 1, & h \in [0, 0.2], \\ -2.611h + 1.522 & h \in [0.2, 0.29], \\ 1.682h + 0.277, & h \in [0.29, 0.4], \\ 0.5h + 0.75 & h \in [0.4, 0.5], \\ 1, & h \in [0.5, 1]. \end{cases}$$

Plotting this out gives Figure 13.

As for $f_V(h)$, we shall divide it into even more piecewise linear functions. Specifically, I chose the intervals $[0, 0.1], [0.1, 0.175], [0.175, 0.2], [0.2, 0.3], [0.3, 0.4], [0.4, 0.8], [0.8, 0.9]$ and $[0.9, 1]$. Initial values are included in Table 14.

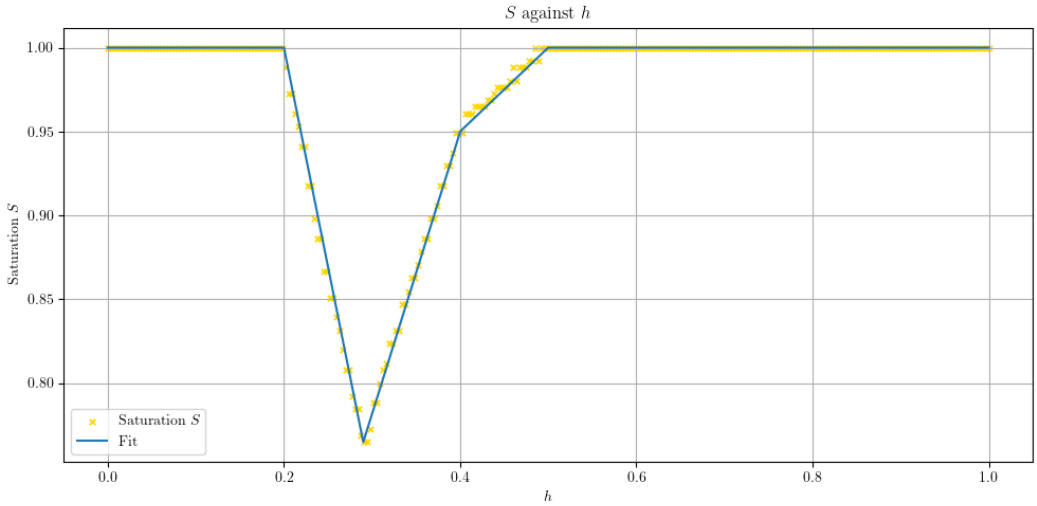

Figure 13: The fit result for $f_S : h \mapsto S$

h	$V = f_V(h)$
0	0.8
0.1	0.98
0.172	0.66
0.2	0.82
0.3	0.98
0.4	1
0.8	1
0.9	0.97
1	0.68

Table 14: Initial values for f_V

This gives us the piecewise function

$$f_V(h) = \begin{cases} 1.8h + 0.8, & h \in [0, 0.1], \\ -4.444h + 1.424, & h \in [0.1, 0.172], \\ 5.714h - 0.323, & h \in [0.172, 0.2], \\ 1.6h + 0.5, & h \in [0.2, 0.3], \\ 0.2h + 0.92, & h \in [0.3, 0.4], \\ 1, & h \in [0.4, 0.8], \\ -0.3h + 1.24, & h \in [0.8, 0.9], \\ -2.9h + 3.58, & h \in [0.9, 1]. \end{cases}$$

Plotting this out gives us Figure 14.

Note that in this plot, V when $h = 0$ or $h = 1$ is excluded, since just like every $h = 0.1k$ for some $k \in \mathbb{N}$, they are anomalies created by the horizontal black line in the scale.

To find $f^{-1} : \mathcal{C} \to [0, 1]$, we do not need necessarily to find an expression of h in terms of

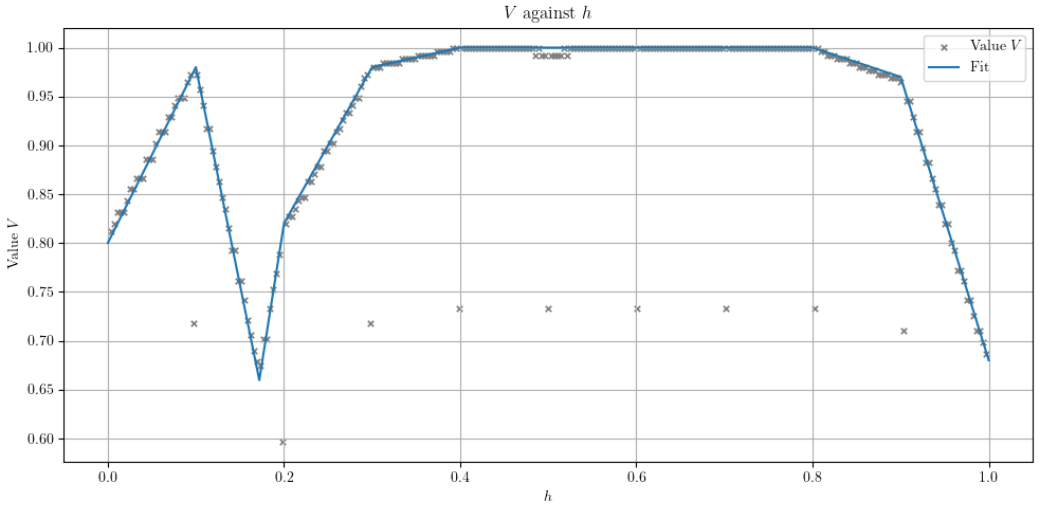

Figure 14: The fit result for $f_V : h \mapsto V$

(H, V, S). If we notice that f_H is one-to-one on $h \in [0, 0.9]$, and f_V is one-to-one on $h \in [0.9, 1]$, we can use f_H^{-1} to determine h from H only if H is non-zero, and use f_V^{-1} otherwise.

$$f^{-1}(H, S, V) = \begin{cases} f_H^{-1}(H), & H \neq 0, \\ f_V^{-1}(V), & H = 0. \end{cases}$$

Notice that for $h \in [0.1, 0.6]$, f_H is a cubic and is not easily invertible. However, it would be plausible to use a binary-search algorithm to find h based on H since it is monotonic, and it is within a reasonable amount of time, to relatively good accuracy. Otherwise, on the linear parts, it is fine to simply mathematically invert it.

Flowchart of data and sidenotes To summarise, we discussed the mapping from the colour space$\mathcal{C}$ to the normalised height h, and back, and we also discussed how h is related with the measured intensity I, the PGA a, the PGV v, and the PGD x. They can be transformed forwards and backwards using simple mathematical explicit relations, and specifically for $f_H^{-1}(H)$ will use a binary search algorithm.

Figure 15 shows the data flow, Figure 16 shows the relation between abstract variables, and 17 shows the result of colour generated compared with the original.

It is worth noting that an existing NuGet Library, ingen084/KyoshinMonitorLib which is designated to manage intensities, as well as extracting intensities from the 'Kyoshin' monitor. This NEA did refer to this for some guidance but is not dependent on this library, and its necessary functionalities within the scope of this NEA is realised again using the author's own code. The developer of this library did also mention that it is quite purpose-built so might not be suitable for general use.

It is also worth noting that, technically, scraping the data from the 'Kyoshin' monitor page of NIED is not explicitly allowed, but not explicitly banned either. However, extracting and displaying numerical data in the application is strictly banned by the NIED, and therefore the numerical values will only serve as internal values of the application and will not be displayed in any way.

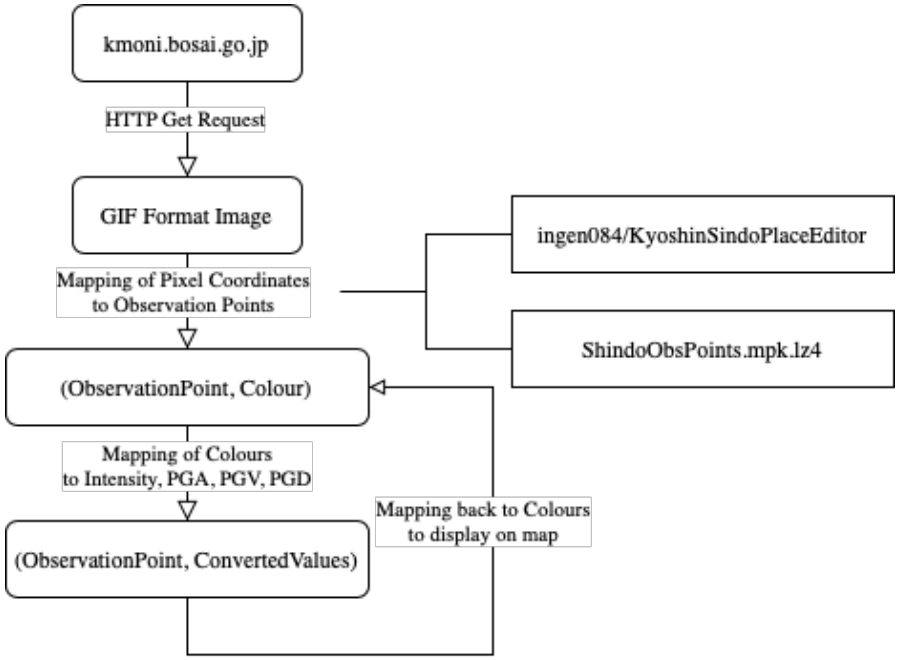

Figure 15: Flow of data in NIED data sources

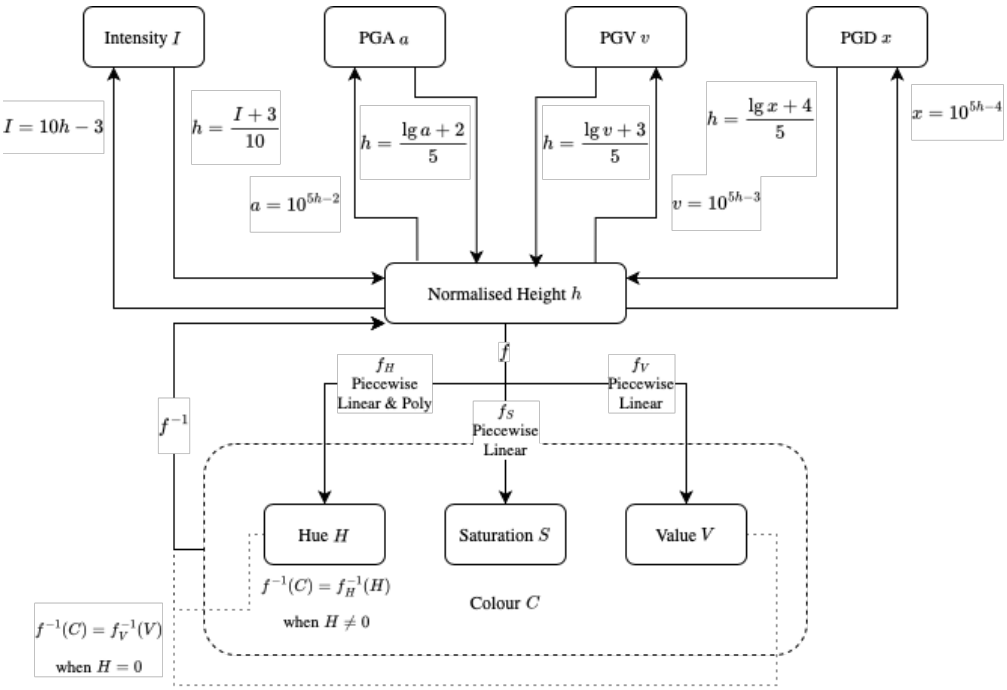

Figure 16: Relation between abstract variables

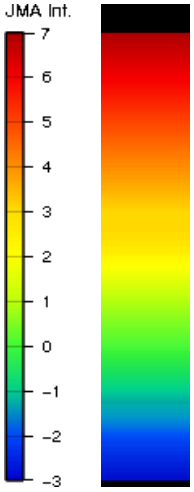

Figure 17: Colour generated using fitted functions

This paragraph referred to the blog article written by NoneType1, author of JQuake. The code used for this section is in Listing 1.

DM-D.S.S. Data Source DM-D.S.S. is a well-structured official data source with low latency and reliable information and services. This is going to be the primary data source for most part of the application.

Their APIs are split into two types: HTTP based requests and WebSocket based connections. HTTP based requests are typically for more static information, while WebSocket connections are for live time-essential data feeds, such as the EEW warnings and latest earthquake information.

Authorisation There are two types of authorisation that DM-D.S.S. supports, API Keys and OAuth2 Access Tokens.

API Keys access tokens are extremely easy to program, since it simply uses Basic BasicBath64 Authorisation in the header, and uses the key as the username (without a password). However, this introduces an extra layer of complexity for the users, since they would have to go to the settings of the DM-D.S.S. webpage and achieve an API Key to paste into the application, as shown in Figure 18.

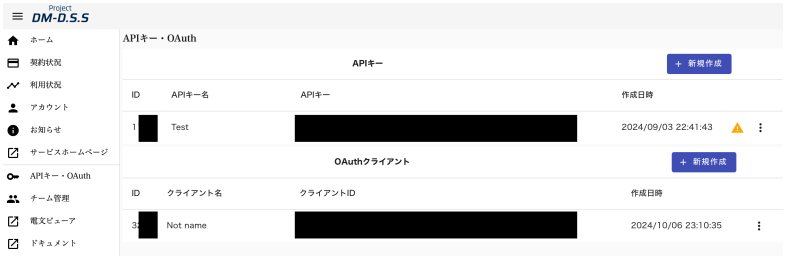

Figure 18: Control panel for API Keys

As for OAuth2, it will be much simpler for the users, since it will provide the user with a login interface on the website, and ask them to give the program certain permissions, which is

just a few simple clicks. Rather than the user sharing the credentials with the application, they are shared between the authorisation server (DM-D.S.S.) and the application directly, without the need for the user to deal with such human-unreadable codes.

How the OAuth2 works for DM-D.S.S. is outlined below:

For the purposes of this NEA, we will primarily use the API Key way of accessing DM-D.S.S. since it will be easier to code and debug, and OAuth2 will introduce quite a lot of complexity to the program. However, the program should be designed to be able to modify to OAuth2 authentication without much modification, and if time permits OAuth2 will be implemented in the application.

HTTP Based Requests Table 15 shows the necessary permissions would be necessary for the application to function. How each of them functions will be discussed below.

Permission Code	Permission Details
`contract.list`	
`eew.get.forecast`	
`gd.earthquake`	
`gd.eew`	
`parameter.earthquake`	
`parameter.realtime`	
`parameter.tsunami`	
`socket.start`	
`socket.list`	
`socket.close`	
`telegram.data`	
`telegram.get.earthquake`	
`telegram.list`	

Table 15: Necessary access permissions for DM-D.S.S.

WebSocket Connections

Flowchart of data and sidenotes It is worth noting that an existing NuGet Library, ingen084/DmdataSharp supports dealing with the DM-D.S.S. data flow and converting them to C# objects (and exceptions). However, for the purposes of this NEA, we will implement our own way to interact with the APIs and the correlated C# DTOs. The developer of this library did also mention that it is quite purpose-built so might not be suitable for general use.

This section referred to the official document for DM-D.S.S.

2.1.2 OOP Model

OOP modelling (classes, methods, attributes, inheritance etc.). Class diagrams would be useful (these are covered in Bond book 1 page 185 onwards). Diagrams should follow conventions for inheritance/composition and private/protected/public methods/attributes.

2.2 Hierarchy Chart

As discussed in the analysis section, the program consists of three parts: data-parsing from external data sources, GUI functionalities and joint functionalities, where the GUI part will

be divided into two parts focusing on real-time monitoring and past-earthquake information, respectively. A detailed discussion into how different modules can be further split up while at the same time being interacted is discussed, and Figure 19 is a hierarchy diagram for the whole application. This shows how **decomposition** technique is applied to reduce a sophisticated problem into more attackable problems.

A top-down approach to problem-solving will lead to the identification of tasks with sub-tasks. i.e. modules and functions required.

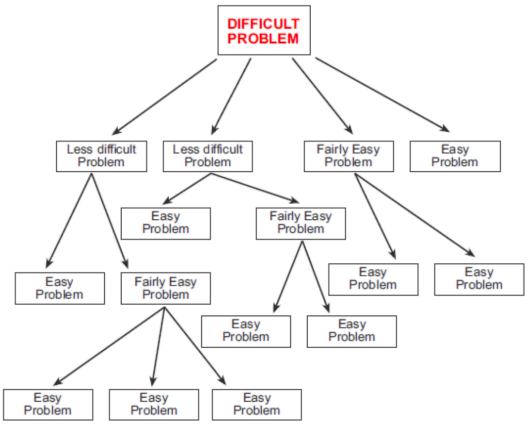

Figure 19: Hierarchy Chart.

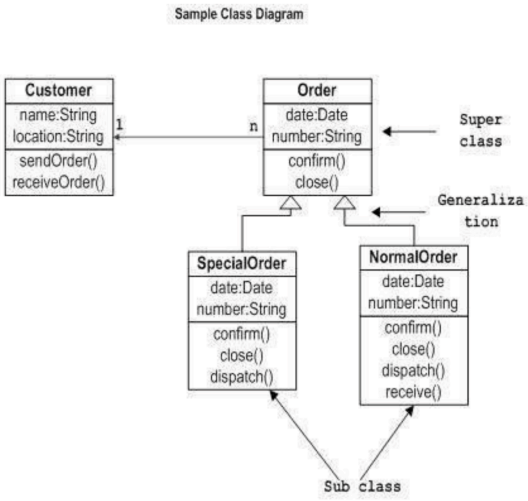

Figure 20: Class Diagram.

2.3 User Interface

You will need to draw up a prototype for the user interface. You may do this within the software package you implement your solution in.

- Screen designs
- Menu options/sequences
- Buttons/keys/commands (command line)

2.4 Hardware Software Requirements

Draw up a hardware and software specification for items that are required.

3 Technical Implementation

3.1 Key Code Segments

3.1.1 Data structures

Implementation of ADTs and OOP Classes to be demonstrated.

3.1.2 Modularity

Code should be created and tested in separate modules that are integrated later. Use subheadings for each module, define the purpose of the module, and show unit testing of the module.

3.1.3 Defensive Programming/Robustness

Exception handling

4 Testing

Consider how you will test your project. You should devise a test strategy that encompasses a range of methods.

4.1 Test Strategy

- Unit testing (of individual functions)
- Integration testing (e.g. different modules/class files)
- Robustness (demonstrating defensive programming skills/exception handling)
- Requirements testing (against your initial requirements - a table with test number, description, test data, expected result, evidence (screenshot/video time link) would be suitable)
- Independent end user beta testing (this will assist with your evaluation)

4.2 Testing Video

- You can include a video to assist (but you will need to reference the time point at which relevant evidence appears)
- If you include a video you will need to have it publicly available.
- It is suggested that you include a QR code in your testing to give a link to it the video (for the moderator) rather than just giving a long URL on its own.

4.3 System Tests (against original requirements' specification)

You need to give evidence in support of requirements that have been met e.g. reference to a relevant test/screenshot/relevant code.

Requirement №	Description	Success Criteria	Tests + Evidence

Table 16: Table of Tests.

5 Evaluation

5.1 Requirements Specification Evaluation

Personal evaluation

- Copy and paste your original requirements from your project analysis
- You need to review each requirement and comment objectively on whether it was *fully met/partially met/not met.*

Requirement №	Description	Success Criteria	Fully/Partial/Not met (Reflective Comment)

Table 17: Table of Evaluation.

5.2 Independent End-User Feedback

End user/client evaluation

- there **must** be meaningful end user feedback
- You should hold a review meeting with your end user
- Write down any key feedback that they give you. E.g. Agreement that a particular requirement has been meet/comments as to aspects that they find suboptimal/comments as to additions they would like to see

Requirement №	Description	Acceptance Y/N	Additional Comments

Table 18: Table of Feedback.

5.3 Improvements

You need to give consideration to a number of potential future improvements that could be made. They may arise from either your experience or from feedback given to you by your end user. Ideally at least one should be in response to end user feedback.

- Write a paragraph for each potential improvement/change
- The improvements/changes could result from additional functionality that has been identified as being beneficial or could be as a result of required efficiencies if some processes are clunky or require faster run-times

- You should then comment on how the proposed change could be implemented moving forward. i.e. what would need to be changed/developed and how? You are not expected to actually make any changes; just comment on the possibilities.

A Code Listing

Listing 1: Polymomial Fit Code

```
# %% [markdown]
# # Polynomial fitting for $f: [0, 1] \to \mathcal{C}$

# %%
from PIL import Image, PyAccess # for image reading
import matplotlib.pyplot as plt # to plot graphs
from scipy import optimize # to do fitting

plt.rcParams['text.usetex'] = True # LaTeX in plots
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = 'STIXGeneral'

# %% [markdown]
# ## Plot $C = (H, S, V)$

# %%
orig_img_path: str = 'jma-scale.png'
orig_img: Image.Image = Image.open(orig_img_path) # read original image

orig_img_hsv: Image.Image = orig_img.convert('HSV') # convert to HSV colour format

img_width: int
img_height: int
img_width, img_height = orig_img.size # get size of image

img_width, img_height

# %%
# read the (H, S, V) in a fixed column

column: int = 5
hsv_list: list[tuple[int, int, int]] = [(orig_img_hsv.getpixel((column, y))) for y in
    range(img_height)] # type: ignore

hsv_list

# %% [markdown]
# ### Plot $f_H, f_S, f_V$ against $r$

# %%
r_list: list[int] = list(range(img_height))

# split up list into separate lists

hsv_h_list: list[int] = [c[0] for c in hsv_list]
hsv_s_list: list[int] = [c[1] for c in hsv_list]
hsv_v_list: list[int] = [c[2] for c in hsv_list]

# %%
# setup options for plotting graphs

marker_size: int = 15
marker_char: str = 'x'
line_width: float = 0.3

fig_size: tuple[int, int] = (10, 5)

```

```
h_colour: str = 'turquoise'
s_colour: str = 'gold'
v_colour: str = 'gray'

h_label: str = 'Hue $H$'
s_label: str = 'Saturation $S$'
v_label: str = 'Value $V$'

# %%
def init_plt(xlabel: str, ylabel: str, title: str) -> None:
    plt.figure(figsize = fig_size)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.grid(True)
    plt.tight_layout()

# %%
# plot graph of (H, S, V) against r

init_plt('Row Number $r$', '$(H, S, V)$ Values', f'$(H, S, V)$ Values against Row Number $r$ along column {column}')

plt.scatter(r_list, hsv_h_list, label=h_label, color=h_colour, s=marker_size, marker=marker_char, linewidths=line_width)
plt.scatter(r_list, hsv_s_list, label=s_label, color=s_colour, s=marker_size, marker=marker_char, linewidths=line_width)
plt.scatter(r_list, hsv_v_list, label=v_label, color=v_colour, s=marker_size, marker=marker_char, linewidths=line_width)

plt.legend()

plt.savefig('hsv-against-row.png')

# %%
plt.close()

# %% [markdown]
# ### Plot $f_H, f_S, f_V$ against $h$

# %%
# setup the initial values for h and r

h_0 = 18
h_1 = 296 + 1
r_len = h_1 - h_0

def r_to_h(r: int) -> float:
    return 1 - ((r - h_0) / (r_len - 1))

h_list: list[float] = [r_to_h(r) for r in r_list] # convert list of r to list of h

h_r_list: list[float] = h_list[h_0:h_1] # extract the values of h within range

hsv_h_r_list: list[int] = hsv_h_list[h_0:h_1] # extract the values of (H, S, V) within range
hsv_s_r_list: list[int] = hsv_s_list[h_0:h_1]
hsv_v_r_list: list[int] = hsv_v_list[h_0:h_1]

h_r_list

# %%
```

```
# plot graph of (H, S, V) against h

init_plt('$h$', '$(H, S, V)$ Values', f'$(H, S, V)$ Values against $h$ along column {column}')

plt.scatter(h_r_list, hsv_h_r_list, label=h_label, color=h_colour, s=marker_size, marker=marker_char, linewidth=line_width)
plt.scatter(h_r_list, hsv_s_r_list, label=s_label, color=s_colour, s=marker_size, marker=marker_char, linewidth=line_width)
plt.scatter(h_r_list, hsv_v_r_list, label=v_label, color=v_colour, s=marker_size, marker=marker_char, linewidth=line_width)

plt.legend()

plt.savefig('hsv-against-h.png')

# %%
plt.close()

# %% [markdown]
# ## Regression

# %% [markdown]
# ### Prepare Lists to do Regression

# %%
# normalise values of h, s, v to our desired values

hsv_h_n_list: list[float] = [hsv_h_r_list[i] / 255 * 360 for i in range(279)]
hsv_s_n_list: list[float] = [hsv_s_r_list[i] / 255 * 1 for i in range(279)]
hsv_v_n_list: list[float] = [hsv_v_r_list[i] / 255 * 1 for i in range(279)]

# %% [markdown]
# ### Regression on $f_H$

# %%
# define f_H with unkwnown parameters y_1, y_2 which operates on a list

def f_h_params(xl: list[float], y_1: float, y_2: float) -> list[float]:
    yl: list[float] = [0 for _ in xl]
    for i, x in enumerate(xl):
        if 0 <= x <= 0.1:
            yl[i] =  -150 * x + 237
        elif 0.1 <= x <= 0.6:
            yl[i] = (
                ((222  * (x - 0.3) * (x - 0.4) * (x - 0.6)) / ((0.1 - 0.3) * (0.1 - 0.4) * (0.1 - 0.6))) +
                ((y_1 * (x - 0.1) * (x - 0.4) * (x - 0.6)) / ((0.3 - 0.1) * (0.3 - 0.4) * (0.3 - 0.6))) +
                ((y_2 * (x - 0.1) * (x - 0.3) * (x - 0.6)) / ((0.4 - 0.1) * (0.4 - 0.3) * (0.4 - 0.6))) +
                ((51   * (x - 0.1) * (x - 0.3) * (x - 0.4)) / ((0.6 - 0.1) * (0.6 - 0.3) * (0.6 - 0.4)))
            )
        elif 0.6 <= x <= 0.9:
            yl[i] = -170 * x + 153
        elif 0.9 <= x <= 1:
            yl[i] = 0
        else:
            yl[i] = 0
    return yl

```

```
168 # %%
169 h_init: list[int] = [0, 0] # initial guess to indicate number of paameters
170
171 h_params: list[int]
172
173 h_params, _ = optimize.curve_fit(f_h_params, h_r_list, hsv_h_n_list, p0=h_init) #
        optimised parameters
174
175 h_params
176
177 # %%
178 def f_h(x) -> list[float]: # this is a partial function which gives in the fitted
        parameters
179     return f_h_params(x, *h_params)
180
181 h_p_list: list[float] = [i / 1000 for i in range(0, 1001, 1)] # list to make sure the
        plotted graph looks quite smooth
182 hsv_h_fitted_list: list[float] = f_h(h_p_list) # fitted line
183
184 # %%
185 line_width = 1.25 # make the markers look a bit thicker
186
187 # %%
188 # plot result of fit
189
190 init_plt('$h$', h_label, '$H$ against $h$')
191
192 plt.scatter(h_r_list, hsv_h_n_list, label=h_label, color=h_colour, s=marker_size,
        marker=marker_char, linewidth=line_width)
193 plt.plot(h_p_list, hsv_h_fitted_list, label='Fit')
194
195 plt.legend()
196 plt.savefig('h-against-h.png')
197
198 # %%
199 plt.close()
200
201 # %% [markdown]
202 # ### Regression on $f_S$
203
204 # %%
205 # define f_S
206
207 def f_s(xl: list[float]) -> list[float]:
208     yl: list[float] = [0 for _ in xl]
209     for i, x in enumerate(xl):
210         if 0 <= x <= 0.2:
211             yl[i] = 1
212         elif 0.2 <= x <= 0.29:
213             yl[i] = -2.611 * x + 1.522
214         elif 0.29 <= x <= 0.4:
215             yl[i] = 1.682 * x + 0.277
216         elif 0.4 <= x <= 0.5:
217             yl[i] = 0.5 * x + 0.75
218         elif 0.5 <= x <= 1:
219             yl[i] = 1
220         else:
221             yl[i] = 0
222     return yl
223
224 hsv_s_fitted_list: list[float] = f_s(h_p_list) # fitted line
225
```

```
# %%
# plot result of fit

init_plt('$h$', s_label, '$S$ against $h$')

plt.scatter(h_r_list, hsv_s_n_list, label=s_label, color=s_colour, s=marker_size,
    marker=marker_char, linewidth=line_width)
plt.plot(h_p_list, hsv_s_fitted_list, label='Fit')

plt.legend()
plt.savefig('s-against-h.png')

# %% [markdown]
# ### Regression on $f_V$

# %%
# define f_V

def f_v(xl: list[float]) -> list[float]:
    yl: list[float] = [0 for _ in xl]
    for i, x in enumerate(xl):
        if 0 <= x <= 0.1:
            yl[i] = 1.8 * x + 0.8
        elif 0.1 <= x <= 0.172:
            yl[i] = -4.444 * x + 1.424
        elif 0.172 <= x <= 0.2:
            yl[i] = 5.714 * x - 0.323
        elif 0.2 <= x <= 0.3:
            yl[i] = 1.6 * x + 0.5
        elif 0.3 <= x <= 0.4:
            yl[i] = 0.2 * x + 0.92
        elif 0.4 <= x <= 0.8:
            yl[i] = 1
        elif 0.8 <= x <= 0.9:
            yl[i] = -0.3 * x + 1.24
        elif 0.9 <= x <= 1:
            yl[i] = -2.9 * x + 3.58
        else:
            yl[i] = 0
    return yl

hsv_v_fitted_list: list[float] = f_v(h_p_list) # fitted line

# %%
# plot result of fit

init_plt('$h$', v_label, '$V$ against $h$')

plt.scatter(h_r_list[1:r_len - 1], hsv_v_n_list[1:r_len - 1], label=v_label,
    color=v_colour, s=marker_size, marker=marker_char, linewidth=line_width) # the
    first and last points are excluded from the plot
plt.plot(h_p_list, hsv_v_fitted_list, label='Fit')

plt.legend()
plt.savefig('v-against-h.png')

# %% [markdown]
# ## Back-Generate the image

# %%
# setup an HSV formatted image and get its pixels

```

```
image_generated: Image.Image = Image.new('HSV', (img_width, img_height))
generated_pixels: PyAccess.PyAccess = image_generated.load() # type: ignore

# returns a list of tuples of (H, S, V) for a column

def f(yl: list[int]) -> list[tuple[int, int, int]]:
    hl: list[float] = [r_to_h(y) for y in yl]

    hsv_h: list[float] = f_h(hl)
    hsv_s: list[float] = f_s(hl)
    hsv_v: list[float] = f_v(hl)

    hsv_h_i: list[int] = [int(h / 360 * 255) for h in hsv_h]
    hsv_s_i: list[int] = [int(s * 255) for s in hsv_s]
    hsv_v_i: list[int] = [int(v * 255) for v in hsv_v]

    return [(hsv_h_i[i], hsv_s_i[i], hsv_v_i[i]) for i in yl]

my_pixels: list[list[tuple[int, int, int]]] = [f(list(range(img_height))) for _ in
    range(img_width)]

# copy this into the generated_pixels reference

for x in range(img_width):
    for y in range(img_height):
        generated_pixels[x, y] = my_pixels[x][y]

# convert to RGB and output

image_generated_rgb: Image.Image = image_generated.convert('RGB')
image_generated_rgb.save('generated-colour.png')
```

Listing 1: Polymomial Fit Code