University of Alaska Anchorage

CSCE A470

CAPSTONE PROJECT

Automated Analysis of Oceanic Current Flows using LCS Algorithm

Author:

Jon Rendulic

Supervisor:

Prof. Jifeng Peng, PhD

Anchorage AK, April 2016.

Computer Science & Engineering Department
UNIVERSITY of ALASKA ANCHORAGE

jon.rendulic@gmail.com

Abstract

The target of this capstone is to create a web-server for computing and analyzing atmospheric and oceanic flows. Atmospheric and oceanic flows play an important role in the transport of any pollutant in the atmosphere or in the ocean. The flows are usually chaotic with some large-scale dominant patterns and small-scale turbulence. Therefore, it is difficult to accurately forecast pollutant transport. Over the past 15 years (Venkataraman, 2009), scientists have made enormous strides in their ability to identify and make visually represent the underlying mechanics of flowing air and water, and to predict how objects move through these flows.

In 2006, a new analyzing tool was developed in which a dynamic systems approach is used to identify attracting structures in the flow field. This tool, a MATLAB software package was developed in the Biological Propulsion Laboratory at California Institute of Technology. It enables users to input a time-series of 2-D velocity field data and compute the corresponding finite-time Lyapunov exponent (FTLE) fields, from which Lagrangian Coherent Structures (LCS) such as vortices and fluid transport barriers can be identified.

For this project we are to establish a computer server that automatically computes and analyzes real-time oceanic flows using NOAA forecast data. The tasks for the server are to download ocean currents data. NOAA regularly publish ocean currents forecast data and the server would download these data for analysis. We would then compute the attracting structures using an existing algorithm.
Next, set up the algorithm on the server for it to automatically analyze the currents data and determine the attracting structures then visualize the attracting structures. This would be displayed and visualized on a high resolution map showing the area of interests.

This project and its tasks were divided up between two students. My task was the initial setup, setting up the server using Ubuntu Linux and creating remote access accounts for both VPN and SSH. Installing all the required software and other administrative functions. My partner was mostly responsible for the backend of the program while I worked on the front end. He worked on gathering the raw data, parsing it, and preparing it for the algorithm. My main responsibility was to automate the algorithm that computes the attracting structures then plot the data.

Acknowledgements

I would like to give thanks to Dr. Jifeng Peng, the supervisor of this project. He brought the project to my attention via Dana Collin. I worked on this with him as research during the summer of 2015 and fall of 2015. It is an honor to work on this project for Dr. Peng and the University of Anchorage Alaska's Computer Systems Engineering Department. I would like to also give thanks to my Partner Tuan Huynh whom without his extensive programming knowledge this would not have been possible. I would also like to thank my former coworker Steve Hoskins, a Senior Electrical Engineer at TDX Power who took me under his wing and taught me a lot of what I know in MATLAB. Lastly I would like to thank Dr. Moulic for his mentoring and academic advising through out my college career as well as sharing all his valuable knowledge.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

This project's main objective is to create a server for computing and analyzing oceanic flows. Oceanic flows play an important role in the transport of pollutants in the ocean. The flows are usually chaotic with some large-scale dominant patterns and small-scale turbulence. Therefore, it is difficult to accurately forecast pollutant transport.

Recently, a new analyzing tool was developed in which a dynamic systems approach is used to identify attracting structures in the flow field. The attracting structures dictate the pollutant transport and mixing, therefore providing information on potential hazard zones. The hazard zones are less prone to inaccuracy of the flow field and the pollutant source data than pollutant trajectories, especially in long-term prediction. For this project we used a BSD license.

1.2 Application

For this project we are to establish a computer server that automatically computes and analyzes real-time oceanic flows. The tasks for the server include:

- Download ocean currents data. NOAA regularly publish ocean currents forecast data and the

server is to download this data for analysis.

- Compute the attracting structures using an existing algorithm. Set up the algorithm on the server for it to automatically analyze the currents data and determine the attracting structures.
- Visualize the attracting structures. The attracting structures are displayed and visualized on the map showing the area of interests.

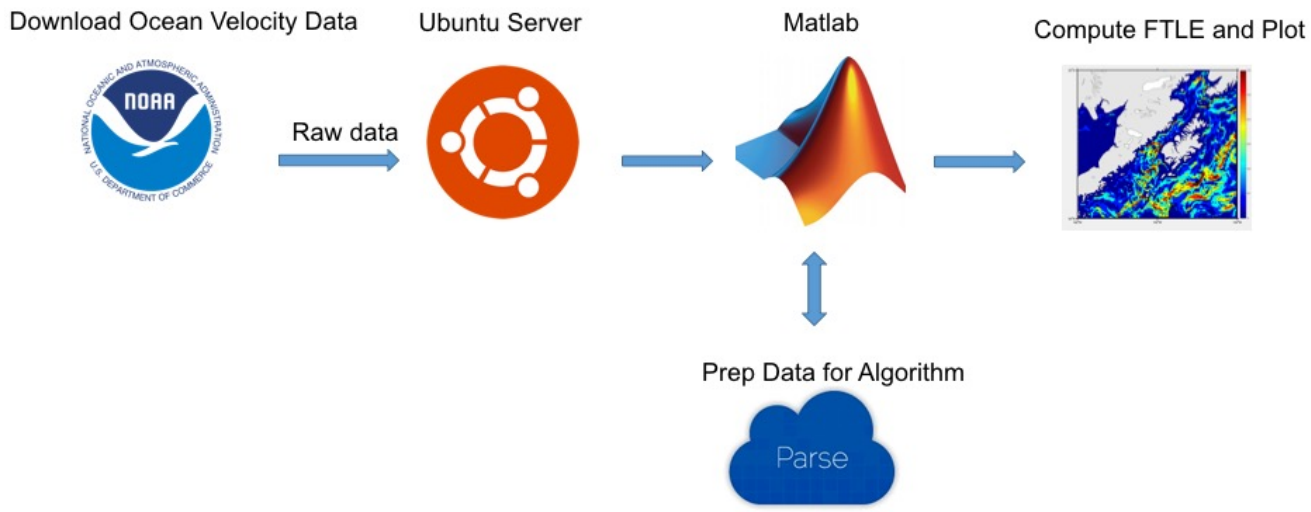


Figure 1 Simple System Diagram

1.3 Motivation

The motivation behind this capstone project is to automate the process of analyzing oceanic flows. The current MATLAB software package developed LCS Matlab Kit V2 (Shadden, Dabiri, & Marsden, 2006) by the Biological Propulsion Laboratory at California Institute of Technology that enables users to input a time-series of 2-D velocity field data and compute the corresponding exponent (FTLE) fields, from which Lagrangian Coherent Structures (LCS) can be identified requires a significant amount of user input. The goal of this project was to create a server which automatically downloads on a daily basis the regularly published ocean current data and then automatically compute the attracting structures to be visualized.

Chapter 2

Technology

2.1 Introduction

The project is composed multiple of subsections. The subsections of development include backend, frontend, and data input. Backend is a server located in UAA. The server has Linux Ubuntu installed and it will have a database that communicates with front end and the sensors. Front end is composed of the website and the Android application. Both website and Android application is chosen to support variety of platforms. The data input will be a simulator that will communicate with the database. It will send current sensor readings to the database.

2.2 Software

The software used in the project are chosen based on their application or because we were modifying some existing software that was already written in a specific language. With the exception of MATLAB our motivation behind software selection was also to use free and open source software that has plenty of documentation on the web.

2.2.1 Ubuntu Server 14.04 LTS

Figure 2 Ubuntu logo.

Ubuntu is an open-source Debian-based Linux operating system and distribution for personal computers, smartphones and network servers. Development of Ubuntu is led by UK-based Canonical Ltd. The Ubuntu project is publicly committed to the principles of open-source software development; people are encouraged to use free software, study how it works, improve upon it, and distribute it. We chose it because I have previous experience with it and it is free open source software that has long-term support (LTS) releases. It has a regular release cycle, and is an agile and secure, deploy-anywhere technology. (Canonical Ltd., 2016)

2.2.2 Xfce

Figure 3 Xfce logo.

Xfce is a lightweight desktop environment for UNIX-like operating systems. It aims to be fast and low on system resources, while still being visually appealing and user friendly. (About - Xfce, 2016)

2.2.3 xrdp

Figure 4 xrdp logo.

xrdp is an open source remote desktop protocol(rdp) server based on the work of FreeRDP and rdesktop, xrdp uses the remote desktop protocol to present a GUI to the user. It provides a fully functional Linux terminal server, capable of accepting connections from rdesktop, freerdp, and Microsoft's own terminal server / remote desktop clients. (xrdp, 2016)

2.2.4 Crontab

Cron is the system process which will automatically perform tasks for you according to a set schedule. The schedule is called the crontab, which is also the name of the program used to edit that schedule. The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. **C**rontab stands for "cron table," because it uses the job scheduler cron to execute tasks; cron itself is named after "chronos," the Greek word for time. We used Crontab to automatically run our MATLAB script daily. (CronHowto - Community Help Wiki, 2016)

2.2.5 MATLAB

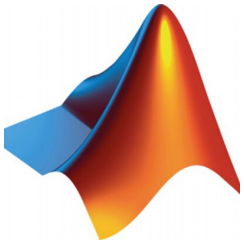


Figure 5 MATLAB logo.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include: Math and computation. Algorithm

development. Our code and the LCS algorithm are all written in MATLAB. (MATLAB - MathWorks, 2016)

2.2.6 NOAA's ERDDAP Server

Figure 6 NOAA logo.

ERDDAP is a data server that gives you a simple, consistent way to download subsets of gridded and tabular scientific datasets in common file formats and make graphs and maps. The ERDDAP server we use gives us 8 days' worth of RTOFS 2D forecast data for u and v_velocity data in meters at 3 hour intervals. We selected the .csv0 format which removes column names and units to allow for easier parsing. (RTOFS Nowcast, 2D, 3-Hourly Prognostic, Global, Latest Model Run, 2016)

2.2.7 LCS Matlab Kit V2

MATLAB software package was developed in the Biological Propulsion Laboratory at California Institute of Technology. It enables users to input a time-series of 2-D velocity field data (e.g., DPIV measurements or CFD calculations) and compute the corresponding finite-time Lyapunov exponent (FTLE) fields, from which Lagrangian Coherent Structures (LCS) such as vortices and fluid transport barriers can be identified. (Shadden, Dabiri, & Marsden, 2006)

2.2.8 M_Map

M_Map is a set of mapping tools written for Matlab v5 and later. (M_Map: A mapping package for Matlab, 2016)

These tools include:

- Routines to project data in 19 different spherical projections (and determine inverse mappings)
- A grid generation routine to make nice axes with limits either in lat/long terms or in planar X/Y terms.
- A coastline database (with 1/4 degree resolution)
- A global elevation database (1 degree resolution)
- Hooks into freely available high-resolution coastline and bathymetry databases
- GSHHS capability

2.2.8 GSHHS high-resolution coastline database

GSHHS is a high-resolution shoreline data set amalgamated from two data bases (the CIA world database WDBII, and the World Vector Shoreline database) in the public domain. The data have undergone extensive processing and are free of internal inconsistencies such as erratic points and crossing segments. The shorelines are constructed entirely from hierarchically arranged closed polygons. The four-level hierarchy is as follows: seashore, lakes, islands within lakes, ponds within islands within lakes. (Shoreline/Coastline Databases | NCEI, 2016)

2.3 Hardware

Main hardware used for the projects are the server and a flat screen monitor which are located in the EIB Heat and Transfer lab.

2.4 Agile Development

We are using the agile methodology to development this project. Agile development provides opportunities to assess the direction throughout the development lifecycle. This is achieved through regular cadences of work, known as Sprints or iterations, at the end of which our team must present a potentially finished product. (What Is The Agile Software Development Methodology, 2016) We have frequent meeting with our supervisor and the scope of the project is discussed and we adjust as we make progress. The google map and webserver implementation proved too complicated and time consuming. So we have removed the implementation of that functionality from the application at this time.

2.4.1 Gannt Chart

The project will be broken into 10 weeks with the final delivery of the project falling on the week of the April 4th. The Gantt chart shows the estimated completion time for each task of the project.

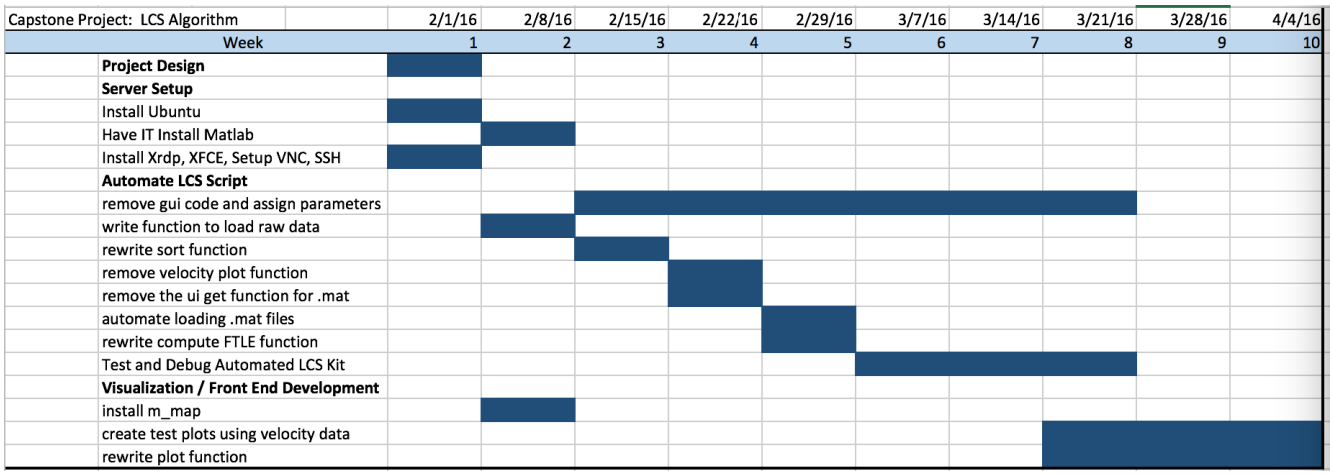


Figure 7 Gannt Chart.

Chapter 3

Setup, User Interface, and Design Testing

3.1 Server Setup

The server has Ubuntu installed with Xfce as the desktop environment. We wanted something lightweight that would not consume a significant amount of resources so we could save those for computing attracting structures. I then installed xrdp so that we could remotely access the server via VPN from off campus. See (Figure 8).

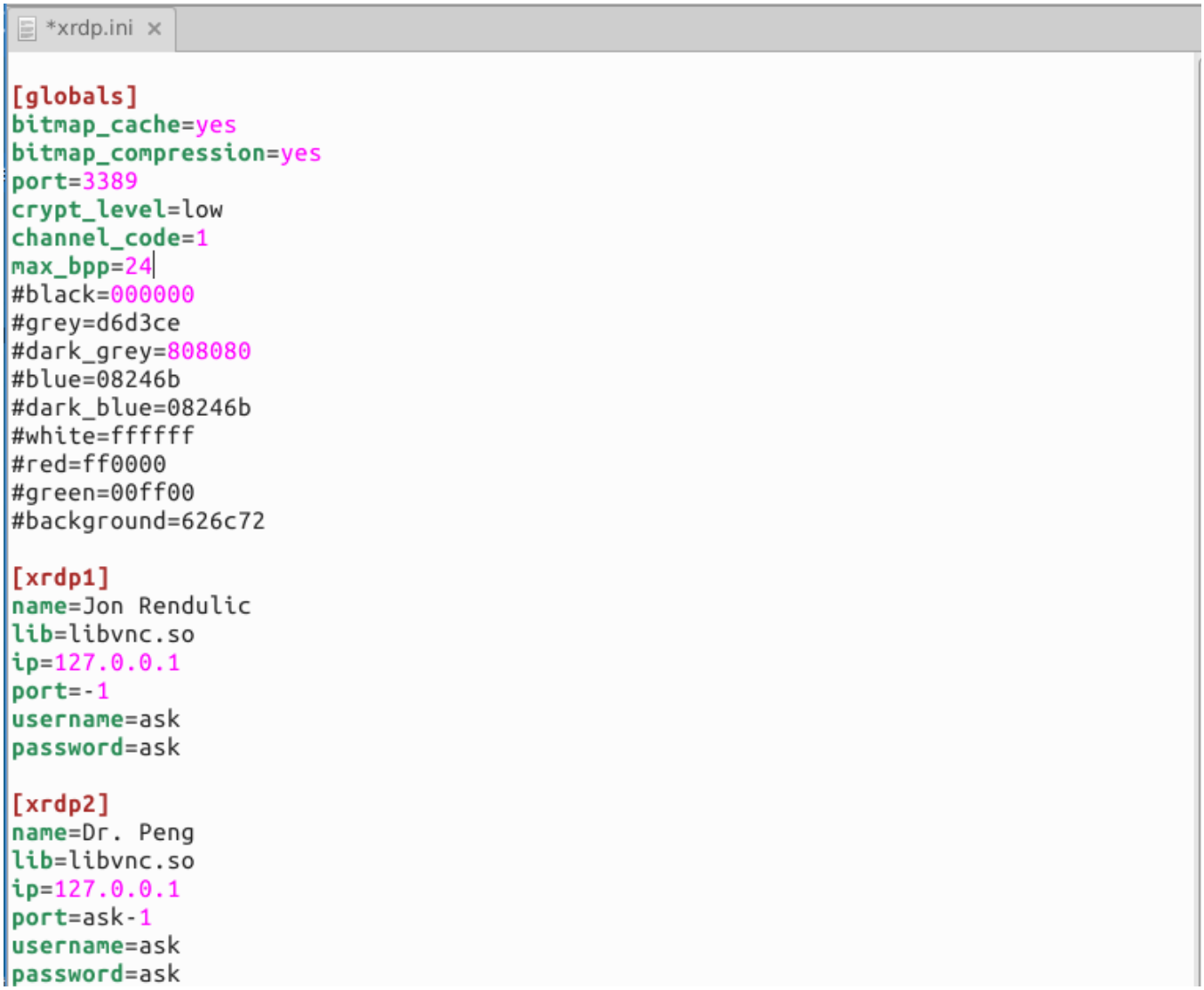


Figure 8 xrdp config file.

Until just recently we were able to connect to the server remotely without any problems, as of a few weeks ago IT implemented new security policies that require us to install Cisco AnyConnect (Cisco AnyConnect Secure Mobility Client - Products, 2016) on the remote client in order to VPN in to campus routing our traffic through a dedicated VPN server. This caused us some trouble and delayed some of our progress while IT was testing and implementing the new server and updated security policies. See (Figure 9).

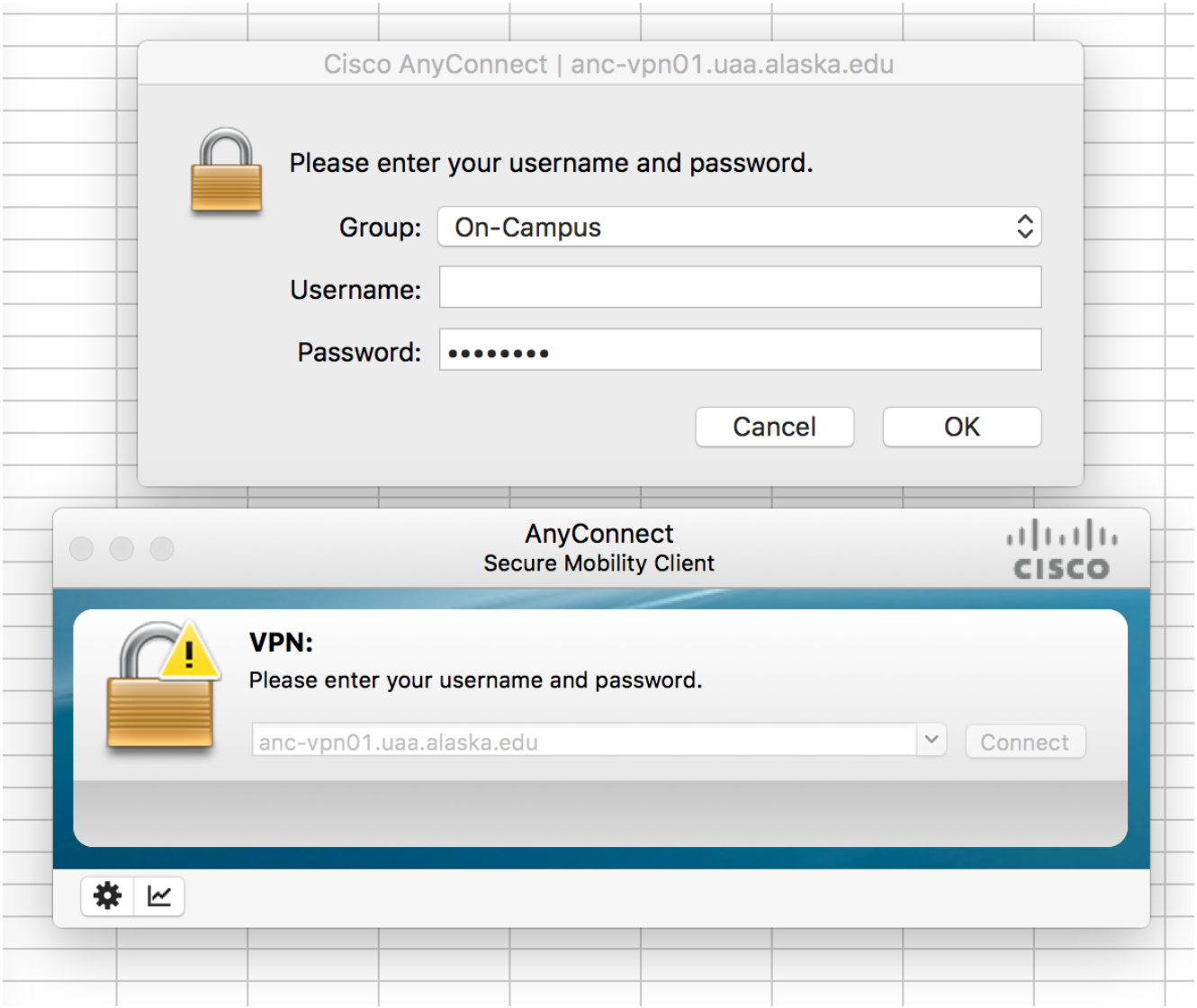


Figure 9 Cisco AnyConnect.

I installed Crontab in order to schedule scripts to run automatically at scheduled intervals. I then had to schedule time with IT in order to get a University Licensed copy of MATLAB installed on the server. Once installed I was then able to install all the required toolboxes and m_map toolbox as well as the LCS MATLAB kit.

3.2 User Interface

The eliminating the need for a user interface is one of the most important parts of our project. The current LCS MATLAB script requires a lot of user input and the main focus of my part of this project after the initial server setup is to remove the graphical user interface and automate the analysis and visualization of this application.

I think it is important to note that before modifying the LCS Kit, Dr. Peng the supervisor of this project had me as a proof of concept demonstrate I could automatically download the raw NOAA ocean current velocity data on a daily basis and then plot the velocity data over a high resolution map.

I was able to do all of this in BASH in Linux using Cron to schedule the downloads and a simple script to download and parse the data. See code below:

```
#!/bin/bash

dl_date="$(date)"
dstamp="$(date +%Y%m%d)"
from="$(date +%Y-%m-%d)"

to=$(date --date='8 day' "+%Y-%m-%d")
day1=$(date --date='1 day' "+%Y-%m-%d")
day2=$(date --date='2 day' "+%Y-%m-%d")
day3=$(date --date='3 day' "+%Y-%m-%d")
day4=$(date --date='4 day' "+%Y-%m-%d")
day5=$(date --date='5 day' "+%Y-%m-%d")
day6=$(date --date='6 day' "+%Y-%m-%d")
day7=$(date --date='7 day' "+%Y-%m-%d")
day8=$(date --date='8 day' "+%Y-%m-%d")


######################################### DAY 00 ################################################

#printf "$dl  date - Downloaded Ocean Velocity Data for Forecast date $from \n\n"

#wget http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DFore3hrlyProg.csv0?u_velocity"[($from):1:($to)]""[(1.0):1:(1.0)]""[
(40):1:(60)]""[(190):1:(210)]",v_velocity"[($from):1:($to)]""[(1.0):1:(1.0)]""[(40):1:(60)]""[(190):1:(210)]" -O rawdata.txt

printf "$dl_date - Downloading Ocean Velocity Data for Forecast date $from \n\n"

#wget http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DFore3hrlyProg.csv0?u_velocity"[($from):1:($to)]""[(1.0):1:(1.0)]""[
(50):1:(60)]""[(200):1:(210)]",v_velocity"[($from):1:($to)]""[(1.0):1:(1.0)]""[(50):1:(60)]""[(200):1:(210)]" -O rawdata.txt

#cut --complement -f 2-2 -d, rawdata.txt|sed 's/,/ /g' > velocity_data_$from.txt

----------------------

#wget http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DFore3hrlyProg.csv0?u_velocity"[($from):1:($day1)]""[(1.0):1:(1.0)]
""[(55):1:(60)]""[(185):1:(190)]",v_velocity"[($from):1:($day1)]""[(1.0):1:(1.0)]""[(55):1:(60)]""[(185):1:(190)]" -O rawdata1.txt

#cut --complement -f 2-2 -d, rawdata0.txt|sed 's/,/ /g' > velocity_data_"$from"_d1.txt

#wget http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DFore3hrlyProg.csv0?u_velocity"[($from):1:($day2)]""[(1.0):1:(1.0)]
""[(55):1:(60)]""[(185):1:(190)]",v_velocity"[($from):1:($day2)]""[(1.0):1:(1.0)]""[(55):1:(60)]""[(185):1:(190)]" -O rawdata2.txt

#cut --complement -f 2-2 -d, rawdata1.txt|sed 's/,/ /g' > velocity_data_"$from"_d2.txt

#wget http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DFore3hrlyProg.csv0?u_velocity"[($from):1:($day3)]""[(1.0):1:(1.0)]
""[(55):1:(60)]""[(185):1:(190)]",v_velocity"[($from):1:($day3)]""[(1.0):1:(1.0)]""[(55):1:(60)]""[(185):1:(190)]" -O rawdata3.txt

#cut --complement -f 2-2 -d, rawdata1.txt|sed 's/,/ /g' > velocity_data_"$from"_d3".txt
```

I then used MATLAB and m_maps to plot the data. See (Figure 10).

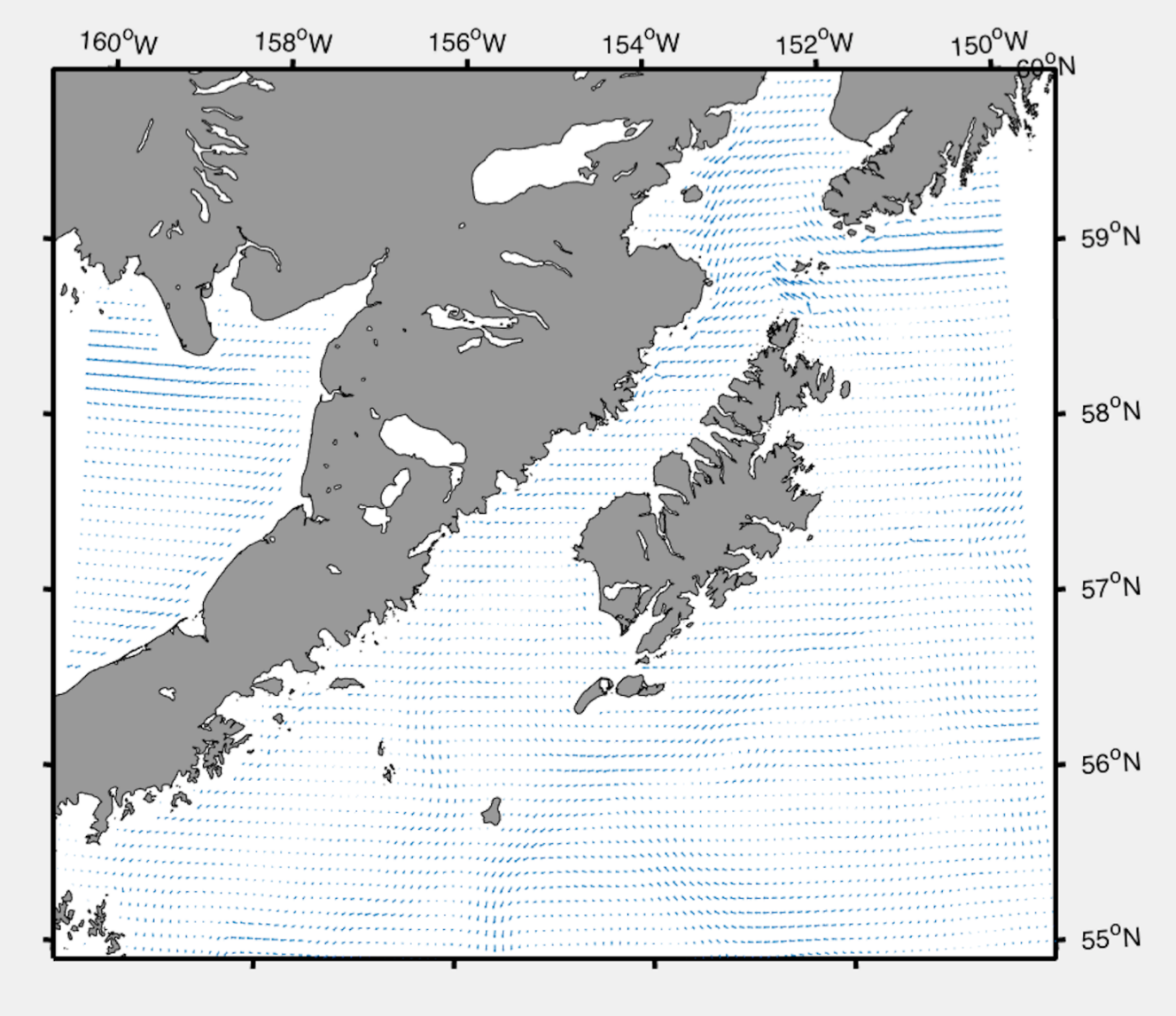


Figure 10 Velocity Plot.

3.2.1 LCS MATLAB kit V2

The original MATLAB software package is 1840 lines of code and requires several steps that needed to be followed in order to compute the attracting structures from calculated particle FTLE. Below are the steps required to run the software that I eliminated or automated:

1. Input the required parameters for instance FTLE Type, FTLE Frames, Integration Settings, Region Settings and Mesh Size. See the GUI in (Figure 11).

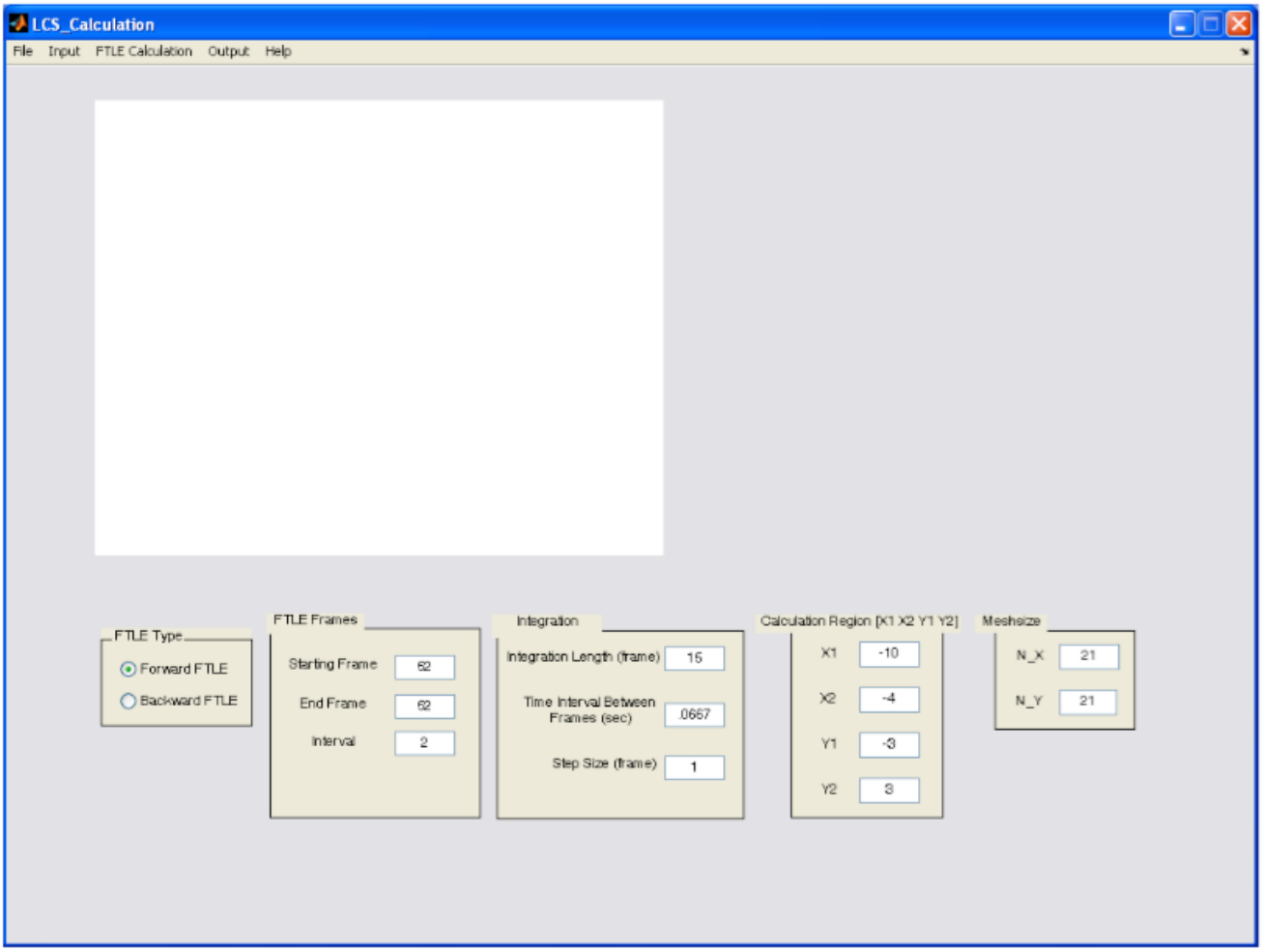


Figure 11 LCS_Calculation.m GUI.

2. Users need to load the raw data which must be parsed and in a specific MxN grid format with x and y coordinates and x and y velocity data or U and V data. This requires the user using the UI to point it do the directory and select the files to be used.
3. The user then must select from the input menu the option to convert those files to MAT files and point it to the directory in which the user wants to save that data.
4. Next is to plot the Velocity Field which is not required so that function is eliminated all together.
5. Assuming all the parameters mentioned above are correctly input into the GUI the user is then required to point the program to the directory and then load the velocity files.
6. Next the user then Calculates the FTLE file using the compute function.
7. The result can then be saved to an FTLE.txt file but is not necessary for our project and that section of code was eliminated.

In summary all of these steps were removed and hard coded into the script reducing the code from 1840 lines to 590, therefore reducing the code by 1250 lines or a 68% decrease.

3.2.2 The Theory

Calculation of Particle FTLE

The FTLE calculation described above consider dynamical systems of ideal fluid tracers, which has the same velocity as local flow velocity. A similar approach can be used to study dynamical systems of finite-size inertial particles. This code can also calculate particle FTLE in dynamical systems of small, rigid, spherical particles, whose dynamics are described by the linearized Maxey-Riley equation (Maxey & Riley, 1983):

$$\frac{d\mathbf{v}}{dt} - \frac{3R}{2}\frac{D\mathbf{u}}{Dt} = -A(\mathbf{v}-\mathbf{u})$$

$$R = \frac{2\rho_f}{\rho_f + 2\rho_p}, \quad A = \frac{R}{St}, \quad St = \frac{2}{9}\left(\frac{a}{L}\right)^2 Re.$$

In this equation, the variable v represents the velocity of the particle, u, that of the fluid, ρp, the density of the particle, ρf , the density of the fluid, υ, the kinematic

viscosity of the fluid, and a, the radius of the particle. The equation has several non-dimensional parameters: R is the mass ratio parameter; St is the particle Stokes

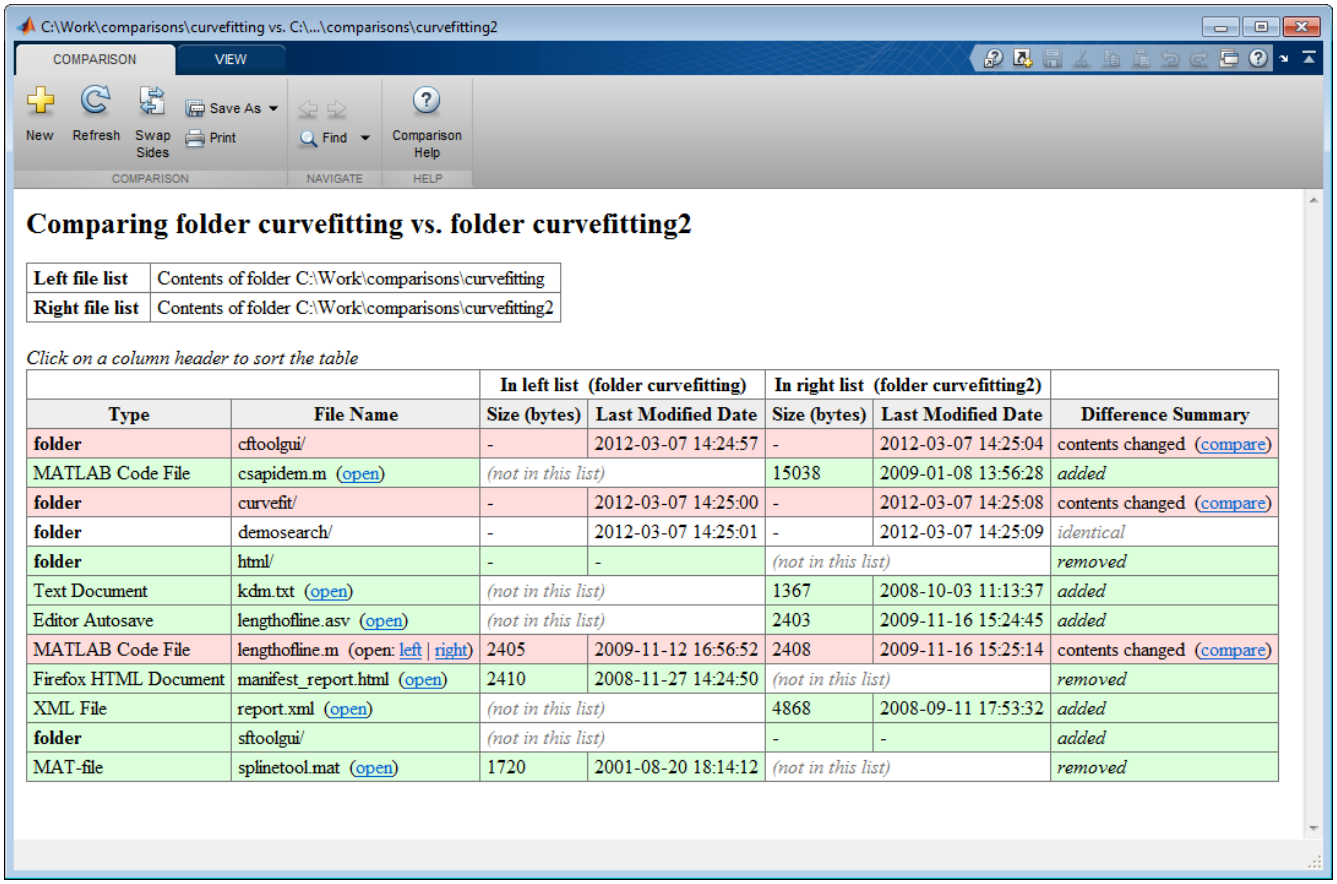
number, Re = UL/υ is the Reynolds number of the flow, and A is the size parameter describing the inertia effect of particles. The assumptions for the linearized

Maxey-Riley equation are: (1) small sphere (a<<1); [reference]. small particle Reynolds number (aV/υ<<1); and (3) small particle Stokes number ((a2/υ)(U/L)<<1).

3.3 Design Testing

Testing of the project is done at various stages along the development process to verify functionality. Final testing is also required to verify system stability. The project is made of multiple parts and testing needs to be done for each part and some parts individual functions needed to be verified.

Various testing was done to compare the results of the automated script versus the results from the GUI script. In order to confirm the accuracy of the modified algorithm I would compare the output of each function using the files and folder comparison tool that is built into MATLAB. This tool would then give me a summary of the results. See (Figure 12 & 13).



		In left list (folder curvefitting)		In right list (folder curvefitting2)		
Type	**File Name**	**Size (bytes)**	**Last Modified Date**	**Size (bytes)**	**Last Modified Date**	**Difference Summary**
folder	cftoolgui/	-	2012-03-07 14:24:57	-	2012-03-07 14:25:04	contents changed (compare)
MATLAB Code File	csapidem.m (open)	*(not in this list)*		15038	2009-01-08 13:56:28	*added*
folder	curvefit/	-	2012-03-07 14:25:00	-	2012-03-07 14:25:08	contents changed (compare)
folder	demosearch/	-	2012-03-07 14:25:01	-	2012-03-07 14:25:09	*identical*
folder	html/	-	-	*(not in this list)*		*removed*
Text Document	kdm.txt (open)	*(not in this list)*		1367	2008-10-03 11:13:37	*added*
Editor Autosave	lengthofline.asv (open)	*(not in this list)*		2403	2009-11-16 15:24:45	*added*
MATLAB Code File	lengthofline.m (open: left | right)	2405	2009-11-12 16:56:52	2408	2009-11-16 15:25:14	contents changed (compare)
Firefox HTML Document	manifest_report.html (open)	2410	2008-11-27 14:24:50	*(not in this list)*		*removed*
XML File	report.xml (open)	*(not in this list)*		4868	2008-09-11 17:53:32	*added*
folder	sftoolgui/	*(not in this list)*		-	-	*added*
MAT-file	splinetool.mat (open)	1720	2001-08-20 18:14:12	*(not in this list)*		*removed*

Figure 12 Folder Comparison Summary.

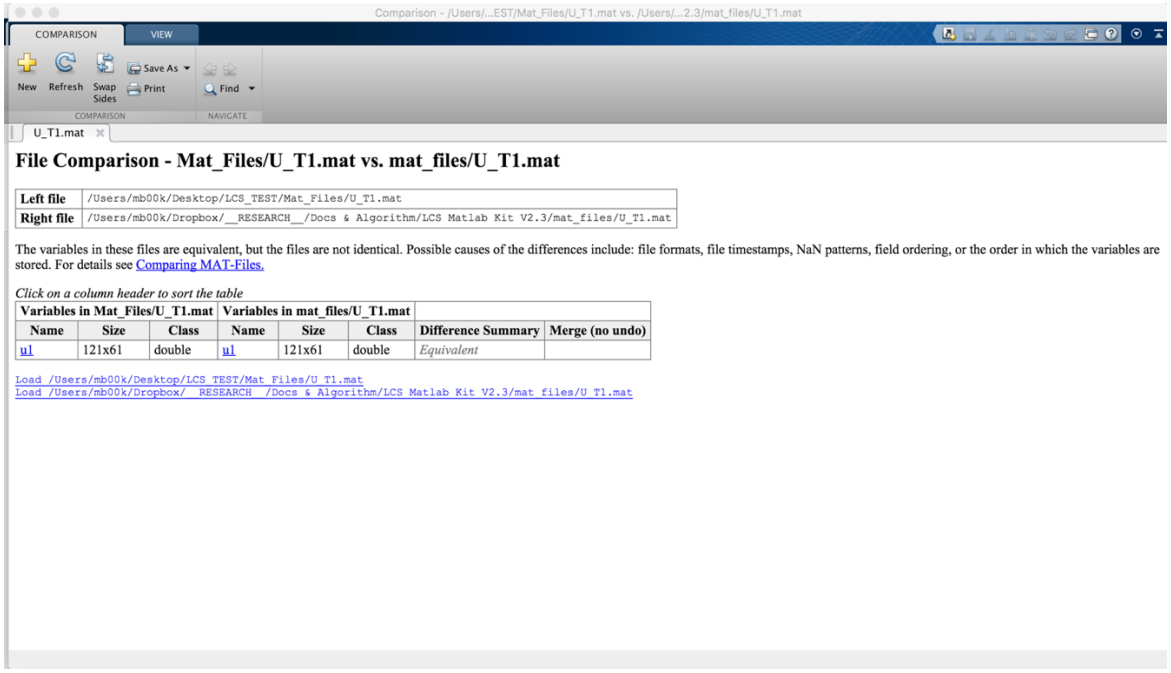


Figure 13 File Comparison Summary.

By plotting the FTLE plot using the function built into the GUI, the function unaltered in the automated script and the new plot function over the high resolution map we can compare results to make sure everything is plotting correctly.

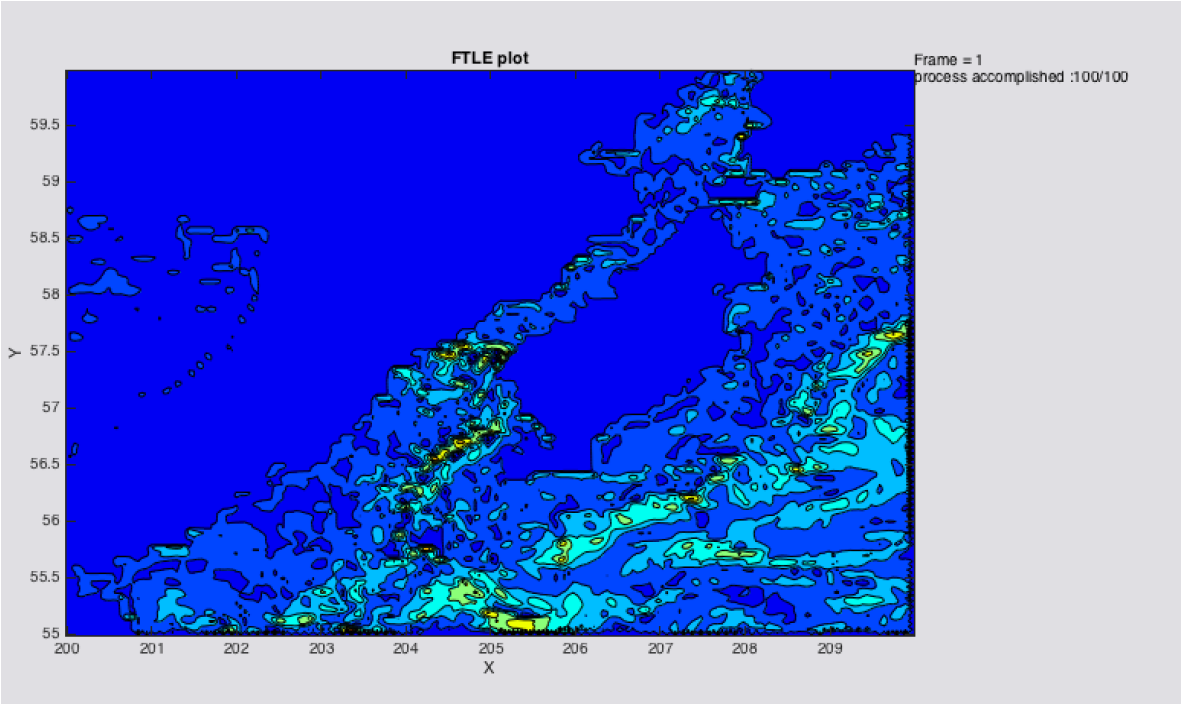


Figure 14 GUI FTLE plot.

We ran into some issues when plotting the data where the contour plot was flipped about its main diagonal and what we found is that sigma0 needed an apostrophe operator (for example, A') which performs a complex conjugate transposition. This flips a matrix about its main diagonal, and also changes the sign of the imaginary component of any complex elements of the matrix.

Below in (Figure 15) is the case mentioned above where we found a bug when rewriting the code to visualize the attracting structures.

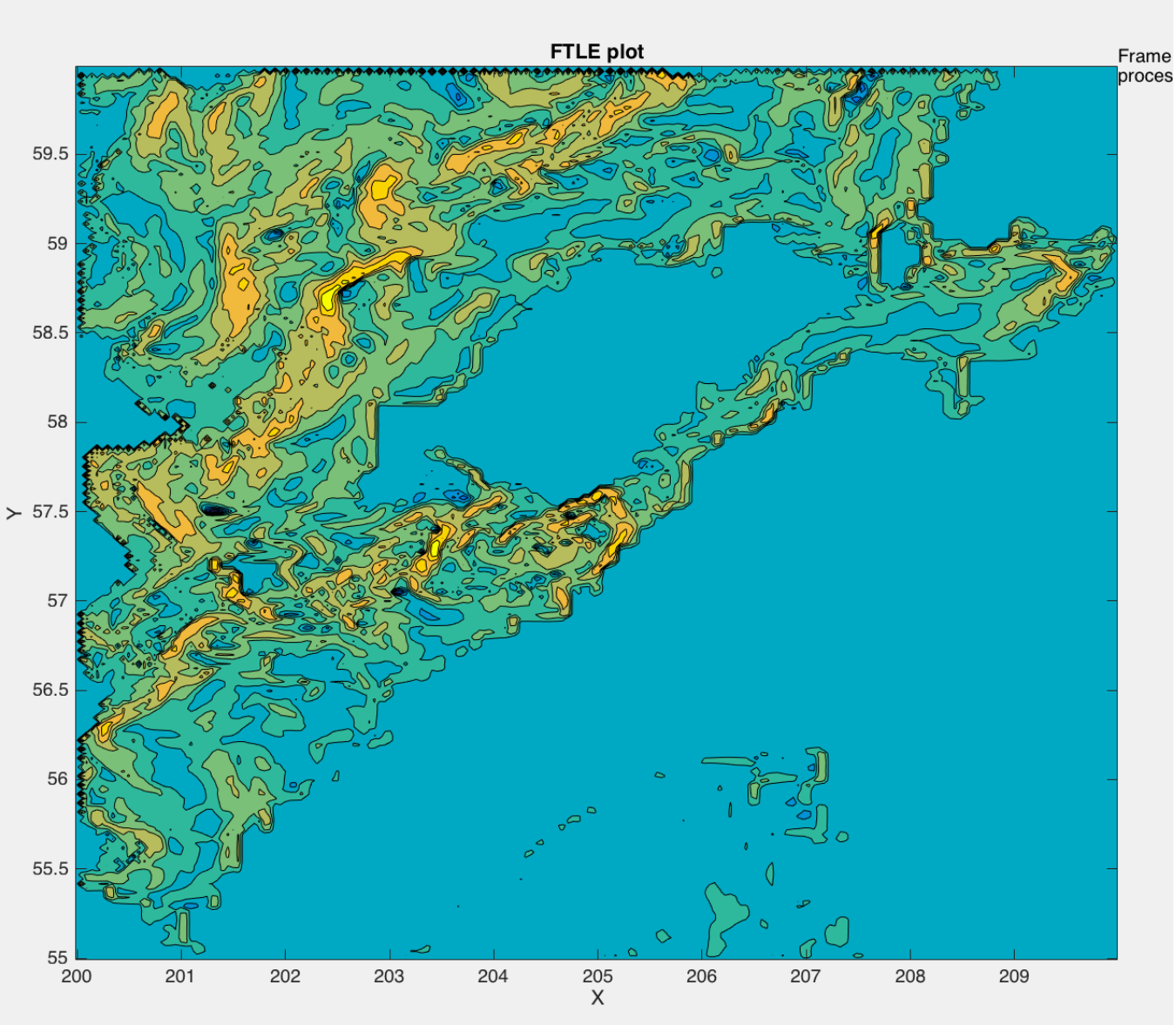


Figure 15 GUI Plot in Automated Script with incorrect Sigma0.

Below in (Figure 16) is an example plot with the correct sigma0 or Z when plotting the contour fill plot to visualize the attracting structures.

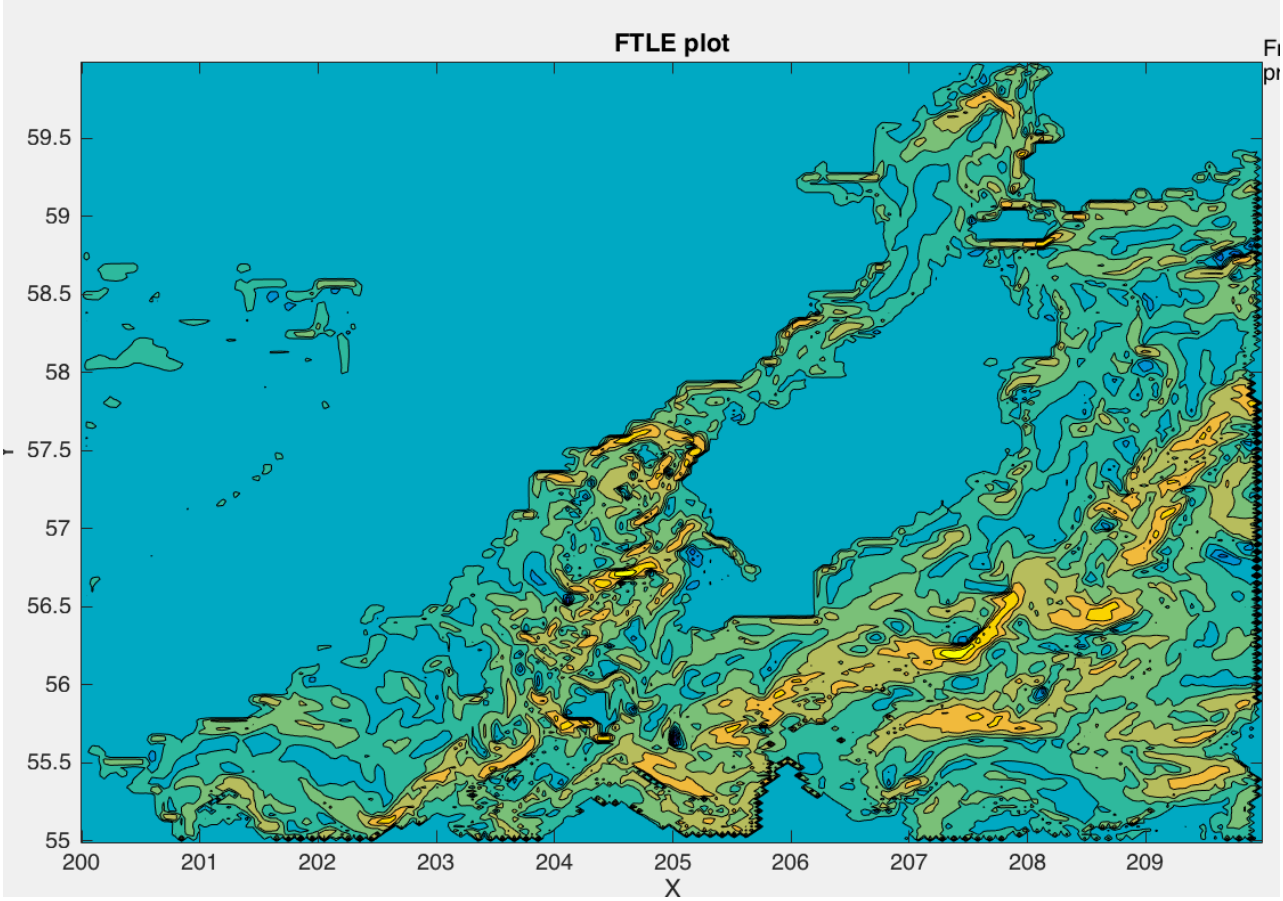


Figure 16 Automated FTLE Plot with corrected Sigma0.

Chapter 4

Results and Discussion

4.1 Results Summary

This project creating a server for computing and analyzing oceanic flows incorporates multiple applications of computer science and computer systems engineering. A through knowledge of Linux, server administration, MATLAB programing, and an understanding of the mathematics behind the algorithm all contributed to making this project a success.

A significant amount of work went into modifying the existing LCS Matlab Kit before the front end portion of this project could be written and tested. In addition to the modification of the LCS Kit on the back end, automating the process of gathering NOAA data, the parsing and preparation of that data for use by the algorithm were required.

4.2 Gather NOAA data

What is great about this website is that it there are so many different format file types for downloaded your gridded data and you have the option to generate the url so that you can code or script that into your program. For our project we required the data in the .csv0 format which assists us in parsing the data. This format is without column names or units. See (Figure 17 & 18).

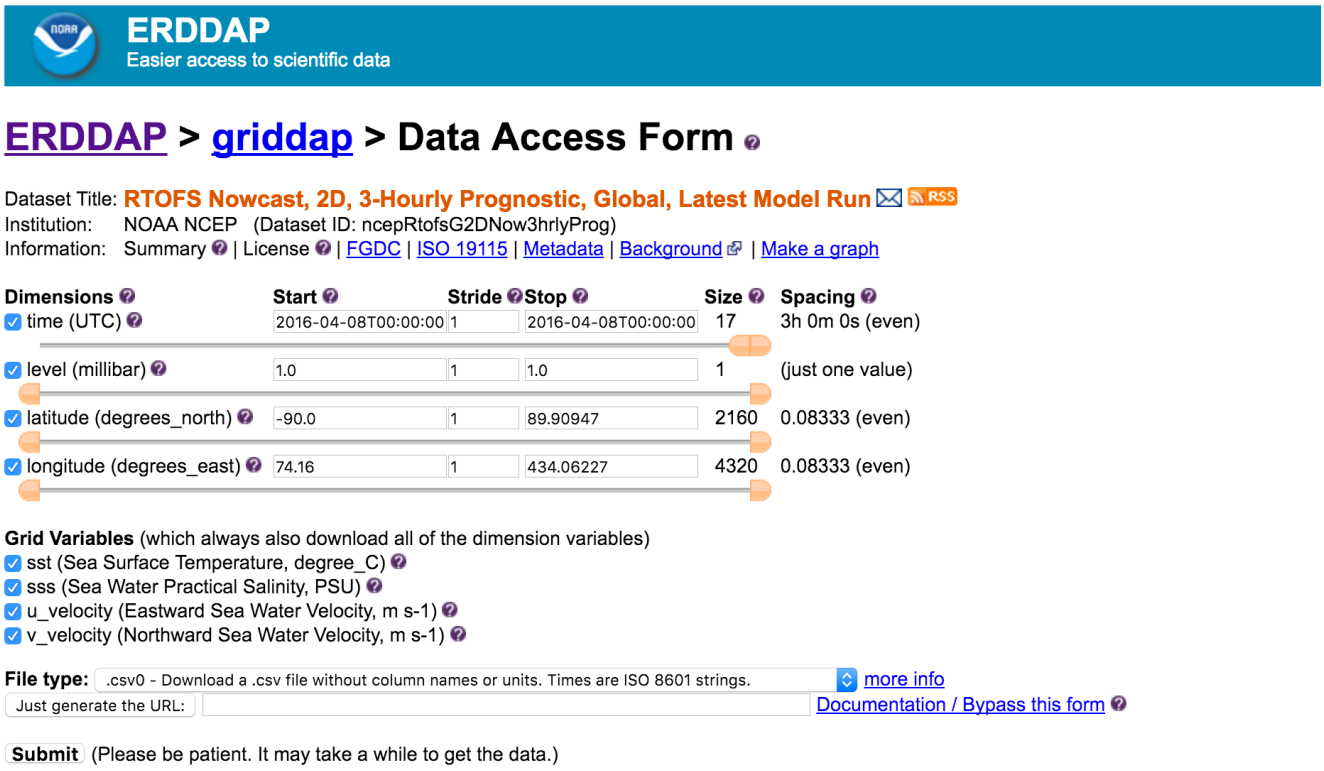


Figure 17 NOAA ERDDAP data access site.

Data fileTypes	Description	Info	Example
.asc	View OPeNDAP-style comma-separated ASCII text.	info	example
.csv	Download a comma-separated ASCII text table (line 1: names; line 2: units; ISO 8601 times).	info	example
.csvp	Download a .csv file with line 1: name (units). Times are ISO 8601 strings.	info	example
.csv0	Download a .csv file without column names or units. Times are ISO 8601 strings.	info	example
.das	View the dataset's metadata via an OPeNDAP Dataset Attribute Structure (DAS).	info	example
.dds	View the dataset's structure via an OPeNDAP Dataset Descriptor Structure (DDS).	info	example
.dods	OPeNDAP clients use this to download the data in the DODS binary format.	info	example
.esriAscii	Download an ESRI ASCII file (lat lon data only; lon must be all below or all above 180).	info	example
.fgdc	View the dataset's FGDC .xml metadata.	info	example
.graph	View a Make A Graph web page.	info	example
.help	View a web page with a description of griddap.	info	example
.html	View an OPeNDAP-style HTML Data Access Form.	info	example
.htmlTable	View a .html web page with the data in a table. Times are ISO 8601 strings.	info	example
.iso19115	View the dataset's ISO 19115-2/19139 .xml metadata.	info	example
.json	View a table-like JSON file (missing value = 'null'; times are ISO 8601 strings).	info	example
.mat	Download a MATLAB binary file.	info	example
.nc	Download a NetCDF-3 binary file with COARDS/CF/ACDD metadata.	info	example
.ncHeader	View the header (the metadata) for the .nc file.	info	example
.ncml	View the dataset's structure and metadata as an NCML .xml file.	info	example
.odvTxt	Download time,lat,lon,otherVariables as an ODV Generic Spreadsheet File (.txt).	info	example
.tsv	Download a tab-separated ASCII text table (line 1: names; line 2: units; ISO 8601 times).	info	example
.tsvp	Download a .tsv file with line 1: name (units). Times are ISO 8601 strings.	info	example
.tsv0	Download a .tsv file without column names or units. Times are ISO 8601 strings.	info	example
.xhtml	View an XHTML (XML) file with the data in a table. Times are ISO 8601 strings.	info	example

Figure 18 Example of ERDDAP file types.

Example data:

```
2016-04-09T03:00:00Z,1.0,57.74409,204.57145,NaN,NaN
2016-04-09T03:00:00Z,1.0,57.74409,204.65478,NaN,NaN
2016-04-09T03:00:00Z,1.0,57.74409,204.73811,NaN,NaN
2016-04-09T03:00:00Z,1.0,57.74409,204.82144,-0.09547188,-0.005023675
2016-04-09T03:00:00Z,1.0,57.74409,204.90477,-0.18158858,0.06311403
2016-04-09T03:00:00Z,1.0,57.74409,204.9881,-0.18363585,0.0430129
2016-04-09T03:00:00Z,1.0,57.82742,199.9883,0.21551046,0.28213015
2016-04-09T03:00:00Z,1.0,57.82742,200.07163,0.20809937,0.28527796
2016-04-09T03:00:00Z,1.0,57.82742,200.15496,0.20004965,0.286647
2016-04-09T03:00:00Z,1.0,57.82742,200.23829,0.19255856,0.28759393
```

So this raw data needs to be parsed because in order for the velocity data to be processed using the algorithm in the LCS Script a few requirements must be met.First the current version supports velocity data files in text (.txt) and worksheet (.wk1) format. So we chose to parse the data from csv and save it as a txt file. However, the data layouts in these files have to be of the following format:

1. Each file contains velocity data on the same M×N grid. It has four columns:

   Column 1: x coordinates (X);
   Column 2: y coordinates (Y);
   Column 3: x velocity (U);
   Column 4: y velocity (V);

2. X and Y coordinates are sorted in ascending order;

3. Each row indicates the velocity data (U, V) at that location (X, Y).

   The following is an example of the velocity file:

   X1 Y1 U11 V11
   X1 Y2 U12 V12
   X1 Y3 U13 V13
   .....
   X1 YN U1N V1N
   X2 Y1 U21 V21
   .....
   X2 YN U2N V2N
   .....
   XM Y1 UM1 VM1
   .....
   XM YN UMN VMN

4. These files then are to be named in such a way that their names have the same prefix and an integer from an ascending number series. For example, ‘velocity00.txt’, ‘velocity01.txt’, ‘velocity02.txt’

4.3 Compute Attracting Structures

Now that the data is parsed an in a format that can be processed by the LCS Script we now run the automated version of the script that I have reduced down from 1840 lines of code down to 590. Now keep in mind that this is being done automatically. The main program is run, the data is downloaded and parsed and then passed to the LCS script to be analyzed.

The LCS portion of the script then preforms the following functions:

1. It loads the raw data using the LoadOriginalDataMenuItem_Callback Function. This used to require the user using the UI to point it do the directory and select the files to be used but I have modified this to automatically point to the correct directory.
2. Next the Script runs a sort function called sort_char_array to prepare the data for the next function.
3. The next function is ConvertToMATFilesMenuItem_Callback which is to convert the data into MAT files for later processing. This function also sorts the data again and builds the meshgrid and saves it along with all the MAT files. Again I have hard coded in the directory removing the uigetdir function, and I hard coded the starting frame at 1.
4. It then runs the ComputeFTLEMenuItem_Callback function which starts the process of computing the FTLE. This is where the parameters that used to be input in the GUI are used and hard coded, for instance FTLE Type, FTLE Frames, Integration Settings, Region Settings and Mesh Size. I also add to add a few lines to save an additional MAT file for each frames Z or sigma0 for plotting.

4.4 Visualize the Attracting Structures

Using the MAT File which saves Z or Sigma0 I am able to plot the attracting structures. I do this using the mapping toolbox and m_maps toolbox. See the following code:

```
figure(4)

    load('/Users/mb00k/Desktop/LCS_TEST/Mat_Files/Z_T1.mat');


 slat = 55;
 nlat = 60;
 wlon = 200;
 elon = 210;
 Z = sigma0';

z1 = min(min(Z)); z2 = max(max(Z));
color_range = z1:((z2-z1)/11):z2;
Contours = 10.^color_range;

m_proj('mercator','lat',[slat nlat],'lon',[wlon elon]);
m_pcolor(xmesh,ymesh,Z),shading flat, hold on
m_contourf(xmesh,ymesh,Z, color_range),shading flat, hold on
caxis([0 0.6]), colorbar, colormap();
m_gshhs_i('patch',[.9 .9 .9]);
m_grid('linestyle','none','tickdir','out','linewidth',2,'xtick',[wlon:5:elon]
,'ytick',[slat:5:nlat],'fontsize',10);
    xlabel('X');
    ylabel('Y');
    title('FTLE plot');
```

And this code produces the following plot as seen in (Figure 19).

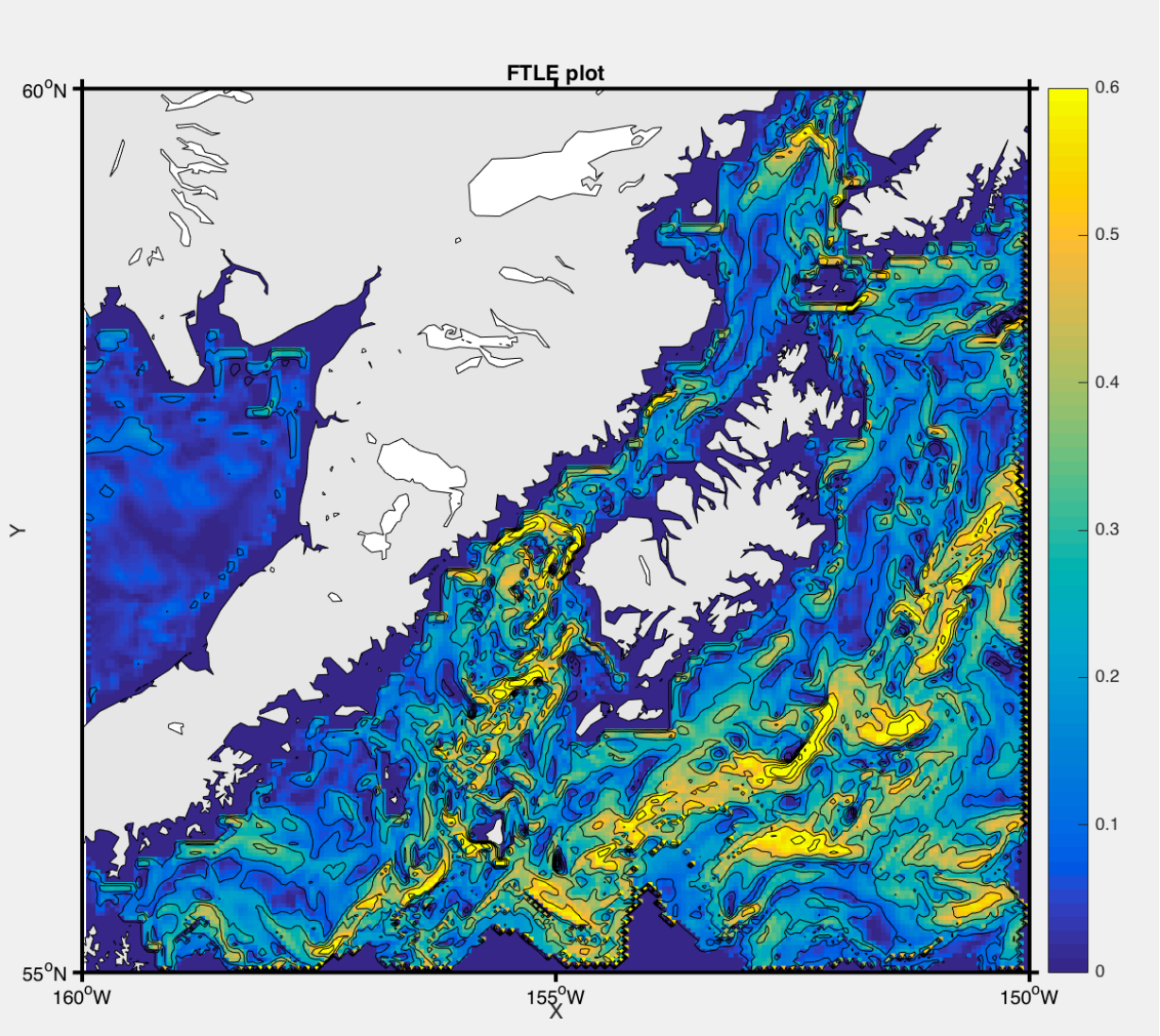


Figure 19 Updated Plot Script with m_map and correct sigma0.

4.4 Discussion

We are very happy with what we were able to accomplish with a small team of two and some limited knowledge NOAA weather datasets, Lagrangian Coherent Structures, fluid mechanics, or of the actual theory behind the algorithm. There are still a lot of room for the improvement. Ideally we would like to visualize this data over google maps with access to the data via a webserver. There is also some modification to the code required to allow an increase in the number of frames the data can be integrated over. In order to integrate over the full 64 frames a function needs to be written that converts the X and Y latitude and longitude into UTM format so that the time interval over which the integration is being performed can be increased with the frames. Each Frame is three hours apart as there are eight days' worth of forecast data in three hour increments.

Chapter 5

Summary and Conclusion

5.1 Summary

Overall this project has been a tremendous learning experience and opportunity to over come significant challenges, as well been a very rewarding experience. I am very pleased with the progress we made and the experience we gained while working on this project. We were able to meet all the major objectives of this project and have a solid foundation in place for future development.

5.2 Implications

When developing this application, one of the goals we had in mind was the ability to quickly select a target area and be able to accurately calculate the attracting structures using NOAA velocity forecast data. Another objective was to simplify and automate the process of gathering, computing, and visualizing the data. One possible application may be to more efficiently organize environmental disaster clean up efforts. Another application may be to help locate a missing person in the ocean. It is not uncommon to hear of kayakers to be swept out to see by currents and turn up missing. An application like this would be able to predict the most likely path an object or particle may have traveled at any given time.

5.2 Future Development

There is no limit to improving this project. months or years could be spent writing the front end of this application on a webserver and visualizing the attracting structures to ones liking. Additional features can be easily added to the original source codes of the project. Some of the future developments includes a webserver, plots over google maps See (Figure 20), incorporating wind data in addition to ocean current data, code improvement to increase the frames over which the data is integrated, and some performance improvements in the code.

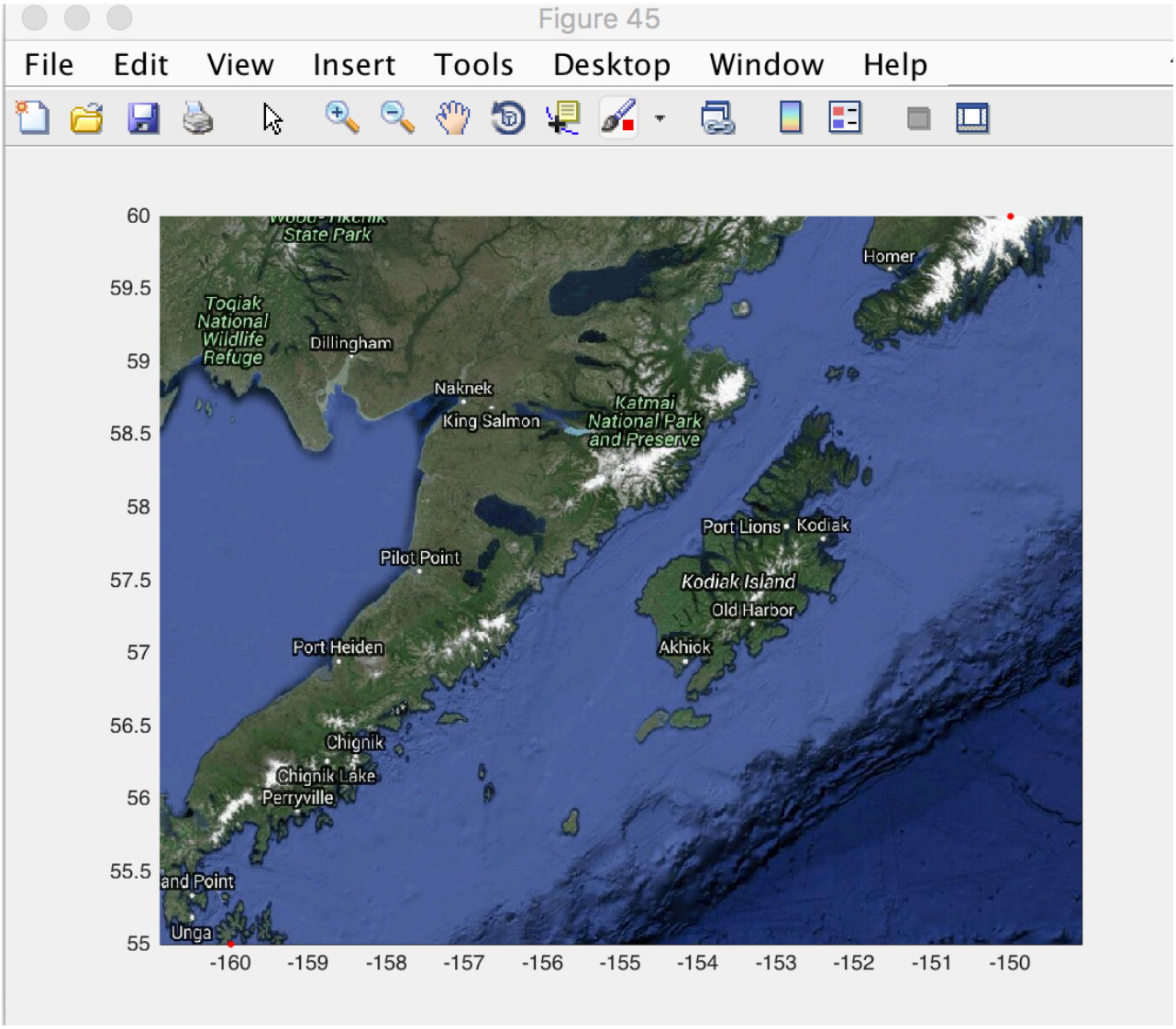


Figure 20 Google Map Plot.

When developing the script, the scope of the project was reduced due to the complexity and the time constraints. Hindsight is 20/20. Had we known what we know now we could have done things differently. Due to the new security policies regarding the UAA network, the remote server connection issues and time spend troubleshooting that also hindered our development time. Eventually it was discovered that IT required outside traffic to be routed through there VPN server. Like I mentioned this

resulted it a delays of development as we could not access the server remotely and required additional troubleshooting.

Having said that, I would like to see this project through and/or see future Computer Science or Computer Systems Engineering students pick up our work where we left off and put those future developments into place. I encourage and challenge my underclassman to pick this project up, as I have found it very challenging yet very rewarding and I am honored for the opportunity to be a part of this research and this project.

5.3 Conclusions

During the development of this project I had the chance to collaborate with my peers and faculty to apply what I have learned over the years at UAA, internships, and from previous job experience to see a project through the various stages of development. When starting this project, we quickly came to the realization that a project of this magnitude will provide many challenges and require a significant amount of research to learn and understand the existing algorithm and the theory behind Lagrangian Coherent Structures. That being said, the project definitely proved to be a lot harder than we had expected. Nevertheless, Tuan and I pushed forward, and were able to reduce the code by 68% and meet the projects main objects of automating the process of data retrieval, parsing and computing, as well as visualizing the attracting structures.

Appendix A – LCS Matlab Kit Source Code

LCS MATLAB Kit Version 2.3 *(w/ inertial particle modelling!)*

http://dabirilab.com/s/LMK23.zip

Appendix B – Source Code

https://github.com/darkstar939/CSCE_A470_Capstone_Project

Works Cited

About - Xfce. (2016, April 13). (X. D. Team, Producer) Retrieved from http://www.xfce.org/about

Canonical Ltd. (2016, April 13). *Download Ubuntu Server | Download | Ubuntu*. Retrieved from http://www.ubuntu.com/download/server

Cisco AnyConnect Secure Mobility Client - Products. (2016, April 13). (Cisco) Retrieved from http://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html

CronHowto - Community Help Wiki. (2016, April 13). Retrieved from https://help.ubuntu.com/community/CronHowto

M_Map: A mapping package for Matlab. (2016, April 13). Retrieved from https://www.eoas.ubc.ca/~rich/map.html

MATLAB - MathWorks. (2016, April 13). (I. The MathWorks, Producer) Retrieved from http://www.mathworks.com/products/matlab/?refresh=true

Maxey, M. R., & Riley, J. J. (1983). Equation of motion for a small rigid sphere in a nonuniform flow. *Phys. Fluids, 26*, 883-889.

RTOFS Nowcast, 2D, 3-Hourly Prognostic, Global, Latest Model Run. (2016, April 13). Retrieved from http://coastwatch.pfeg.noaa.gov/erddap/griddap/ncepRtofsG2DNow3hrlyProg.html

Shadden, S. C., Dabiri, J. O., & Marsden, J. E. (2006). Lagrangian analysis of entrained and detrained fluid in vortex rings. *Phys. Fluids, 18*.

Shoreline/Coastline Databases | NCEI. (2016, April 13). (N. M. Relief, Producer) Retrieved from https://www.ngdc.noaa.gov/mgg/shorelines/

Venkataraman, B. (2009, September 28). *Finding Order in the Apparent Chaos of Currents*. Retrieved April 13, 2016, from http://www.nytimes.com/2009/09/29/science/29chaos.html?_r=4&ref=science

What Is The Agile Software Development Methodology. (2016, April 13). Retrieved from http://agilemethodology.org/

xrdp. (2016, April 13). (J. Sorg, Producer) Retrieved from www.xrdp.org