COVENTRY UNIVERSITY

5011CEM BIG DATA PROGRAMMING PROJECT
PORTFOLIO

SPEED COMPARISON OF DATA DENSITY BASED CLUSTERING (DDC) ALGORITHM USING MATLAB EXTERNAL LANGUAGE INTERFACES

April 2020

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LOGBOOK

Week 1 (20-01-2020 to 26-01-2020)
Work Planned: None, First Class, Introduction to project. Work Achieved: Understanding project brief. Next Steps: Learn Matlab (courses and basics to write code). Challenges & Debugging: Took some time and a lot of questions to understand brief. Not sure what sub-project would be a better choice. Professional Development: None.

Week 2 (27-01-2020 to 02-02-2020)
Work Planned: Complete Matlab Onramp course from official website. (15 Lessons) Work Achieved: 6 Lessons completed[1], Lecture on specifications, wrote down a basic outlook on project [2]. Next Steps: Finish Matlab Onramp Course. Challenges & Debugging: Understanding SMART targets for a technical project (went through various articles and examples). Professional Development: Half-way to achieving certificate from matlab course. Matlab commands, Vector and array manipulations.

Week 3 (03-02-2020 to 09-02-2020)
Work Planned: Finish Matlab Onramp Course Work Achieved: Lesson 7-15 Completed [3][4], Lecture on Code Planning. Next Steps: Download Data, open files and manipulate arrays, start spec. Challenges & Debugging: Completing the projects in matlab onramp course (recognizing attributes and findings from given data). Professional Development: Achieved certificate for Matlab Onramp course.

Week 4 (10-02-2020 to 16-02-2020)
Work Planned: Dowaload data files. Open and run manipulations. Start spec. Work Achieved: Lecture on report writing, Downloaded all data and extracted to array. Basic outline of spec version one. Next Steps: Run DDC algorithm on matlab with the scaled data. Challenges & Debugging: Understanding dimensions of 4D arrays and manipulating to get certain rows. Professional Development: Learnt about ncinfo and 4D array manipulation.

Week 5 (17-02-2020 to 23-02-2020)
Work Planned: Run DDC algorithm on matlab with the scaled data. Work Achieved: Imported O3Scaled Data, Ran and timed DDC algorithm function with different flags for merging and visualization. Next Steps: Run DDC user script in python using matlab engine api. Challenges & Debugging: Issues with calling function from a separate script due to the fact that I saved the function in a script file with different name instead of saving it in a function file with same name. Used uiimport and load to import data but function would not run as the 1 by 1 resulting structure dimensions are not accepted by the function input. Then imported the mat file and sliced the data to fit required dimensions. Function ran with merge flag true but in verbose mode, out of index errors arose due to some lines. Changed array indexing limits in the visualization function to get the visualized output. Professional Development: Understanding function calls in matlab, difference between uiimport, load and importing using toolbar, understanding results of DDC algorithm. Cluster has the centre coordinates and radii values. Results has two columns, one with ozone values and another with the corresponding cluster number.

Week 6 (24-02-2020 to 01-03-2020)
Sick

Week 7 (02-03-2020 to 08-03-2020)
Sick

Week 8 (09-03-2020 to 15-03-2020)
Work Planned: Run DDC user script in python using matlab engine api. Work Achieved: Connected matlab to python and called the test script. Next Steps: Run DDC user script in C++ using matlab engine api. Challenges & Debugging: Went through documentation and downloaded python engine in matlab. Called and ran a test program which worked but when running the DDC script first error that popped up was that the file could not be found. So, I added statement to navigate to the directory in which the file exists and it worked. The next major issues was that running the program resulted in error of “O3Scaled not found. Un-identified variable”. Went through documentation, tried calling the function instead of user script, then downloaded and imported scipy and numpy modules. Used the modules to load the mat file into python and pass it to matlab workspace but slicing the array to pass it didn’t work. Had to transfer rest of the work to next week. Professional Development: Usage of matlab engine api for python, usage of scipy and numpy, updating system environment variables and dependencies.

Week 9 (16-03-2020 to 22-03-2020)
Work Planned: Run DDC user script in C++ using matlab engine api. Work Achieved: Connected matlab to C++ using the matlab engine api for c++ and called the test script, Used matlab compiler runtime (mcr) to create executables and generating dll files of the script. Next Steps: Call the DDC user script and run and time in both python and C++. Challenges & Debugging: Engine would not connect due to issues with dependencies, not really clear in documentation for running with Visual studio. Found a matlab Q&A

which listed out the steps and additional libraries for building project in Visual Studio. The test script connected and ran. Errors of "O3Scaled not found. Un-identified variable.", occurred here too. Decided to try mex files and generate a dll. Matlab compiler runtime (mcr) was not installed so ran installation process but ran out of space in systems SSD so had to cleansweep through files and program. After installing mcr, generated executables and dll files to run the script. Tried to run the executable but system hardware said mcr was not installed, so added the path to system environment variables. Then the error of mcr version popped up, system asked for a previous version. Even though dll and executables were created they were not run successfully.

Professional Development: Usage of matlab engine api for C++, C++ mex applications, usage of matlab compiler runtime to create dll files and executables.

Week 10 (23-03-2020 to 29-03-2020)

Work Planned: Call the DDC user script and run and time in both python and C++.

Work Achieved: Used matlab coder to convert function into c++ code, Ran and timed the DDC user script in python and C++ by using a shared matlab session.

Next Steps: Run DDC user script in Java using matlab engine api.

Challenges & Debugging: Taking on from last week, this week I tried to convert matlab code to C++ code using the Matlab Coder. Downloaded matlab coder and ran the review on the function. Had to remove visualization and merge from to successfully convert code to C++. After converting function, used matdgns.c and mattest.mt to read mat data into C++ and pass through function. It worked but since python and Java will be using the matlab engine api's to make sure there are no discrepancies tried to call the user script using api again. O3Scaled was still unrecognized. Then tried to use system call() to run user script but same error occurred. Realized that when connect to matlab in python or C++, it connects to a new session and this session does not have the data imported to the workspace. To overcome the above, connected to a shared matlab session which had data already imported to the workspace and then called the corresponding C++ and python files.

Professional Development: Using matlab coder to convert matlab code to C++, connecting to shared session of matlab in Python and C++.

Week 11 (30-03-2020 to 05-04-2020)
Work Planned: Run DDC user script in Java using matlab engine api. Work Achieved: Connected matlab to Java and ran test file, Ran and times DDC user script in Java using matlab engine api. Next Steps: Update Spec and work on report. Run numerous simulations of each code to get best average time. Challenges & Debugging: Added dependencies and libraries to connect but test file would not compile in the command prompt. Had to add java path to system environment variables, update JDK version as per requirements. Then the imported module needed to create matlab engine threw an not found error. Realised that project dependencies were not uploaded probably with both engine.jar and javabuilder.jar, so rectified that. After the test file ran successfully, taking on feedback learnt from previous weeks, connected to a shared matlab session with data imported and timed the time elapsed. Professional Development: Usage of matlab engine api for Java, usage of netbeans ide and updating compiler version in its configuration files.

Week 12 (06-04-2020 to 12-04-2020)
Work Planned: : Update Spec and work on report. Run numerous simulations of each code to get best average time. Work Achieved: Completed Spec. Started work on report. Ran simulations and took data of runtimes from each application. Next Steps: Finish report and portfolio. Fix C++ executable runtime. Challenges & Debugging: While running the code in various languages matlab and python were quite similar. Java took between 45 minutes to an hour in three different sessions while on netbeans. C++ compilation and executable was also running for a long time, didn't reach result of cluster 2 till after the two hour mark. Figured something must be wrong with code so scheduled to fix issues next week. Professional Development: Professional documentation, Writing SMART targets.

Week 13 (13-04-2020 to 19-04-2020)

Work Planned: Finish report and portfolio. Fix C++ executable runtime.

Work Achieved: Completed report and portfolio.

Next Steps: Work on ppt and recording, Try running C++ executable again.

Challenges & Debugging: Checked C++ code, there were no issues with code, tried nargout=0 to manage output. It still took way too long to run on my system. Tried to run the using dll file instead, same issue occurred. Thinking that system hardware and visual studio executables are the reason the speed is so low.

Professional Development: Professional documentation, Kanban board, burn-down chart, agile processes.

Week 14 (20-04-2020 to 26-04-2020)

Work Planned: Work on ppt and recording, Try running C++ executable again.

Work Achieved: Finished ppt and recording.

Next Steps: Submit project on 27-04-2020.

Challenges & Debugging: Tried running C++ code again, took almost two hours again to reach cluster 2 in results. Compared processor speed used by each application in each language using task manager and listing current processes running. Java was slower than matlab and python but C++was going at almost halt the processor speed. Ran all four files again, making sure there are no background processes (system interrupts were unavoidable).

Professional Development: Presentation of technical projects.

VERSION CONTROL

Version control was employed by keeping track of files in each language by numbering them. Updates to personal GitHub were also done using feature branches for each language. Please note that test files have not been included in version control. These will be address in the section automated testing.

Matlab

1. Version one had function and function call both in one script.
2. Version two had divided function and user script for function calls.
3. Version three- user script stayed the same but array indexing in visualization function in lines 139, 140 ,147 ,153 were changed.
4. Final version has comments and section breaks added in user script.

Python

1. Version one had basic code, finding directory path and script call.
2. Version two had added scipy module, loading mat data file and script call changed to function call.
3. Version three had added numpy module, loading mat data file and script call change to function call.
4. Version four was built by reverting to version one and changing connect engine function to shared connect engine function.
5. Version six has added code to compute elapsed time using timeit.
6. Final version has comments and alternate script call added.

C++

1. Version one had basic code to connect matlab engine and call user script.
2. Version two was usage of matlab compiler runtime to create dll and executables.
3. Version three was code created using matlab coder, converted ddc function code from matlab to C++.
4. Version four was based on version one with system() to call user script.
5. Version five was also based on version one but matlab connect was changed to shared session.
6. Version six had comments, usage of chrono to compute time elapsed, and alternative find and connect to shared session function added.

Java

1. Version one has basic code, class to connect to shared engine and run user script.
2. Version two has changed imports from just importing MatlabEngine to import all modules of mathworks.engine.

3. Version three has computation of time elapsed using nanoTime().
4. Version four reverts to version two and computes time elapsed using currentTimeMillis().
5. Version five has alternative find and connect to shared session class added.
6. Final version has docstrings and comments added.

AUTOMATED TESTING

Testing was implemented by using sample data and arrays, and test code. The same basic test code was run across all applications to measure speed taken to execute to make sure findings were similar to resulting findings.

Matlab

1. Running the DDC algorithm with test data and test arrays.

Python

2. Ran a test program of square of two numbers to check connection on both shared and not shared sessions.

C++

3. Ran same test program of square of two numbers to check connection on both shared and not shared sessions.

Java

4. Ran same test program of square of two numbers to check connection on both shared and not shared sessions.

Overall project

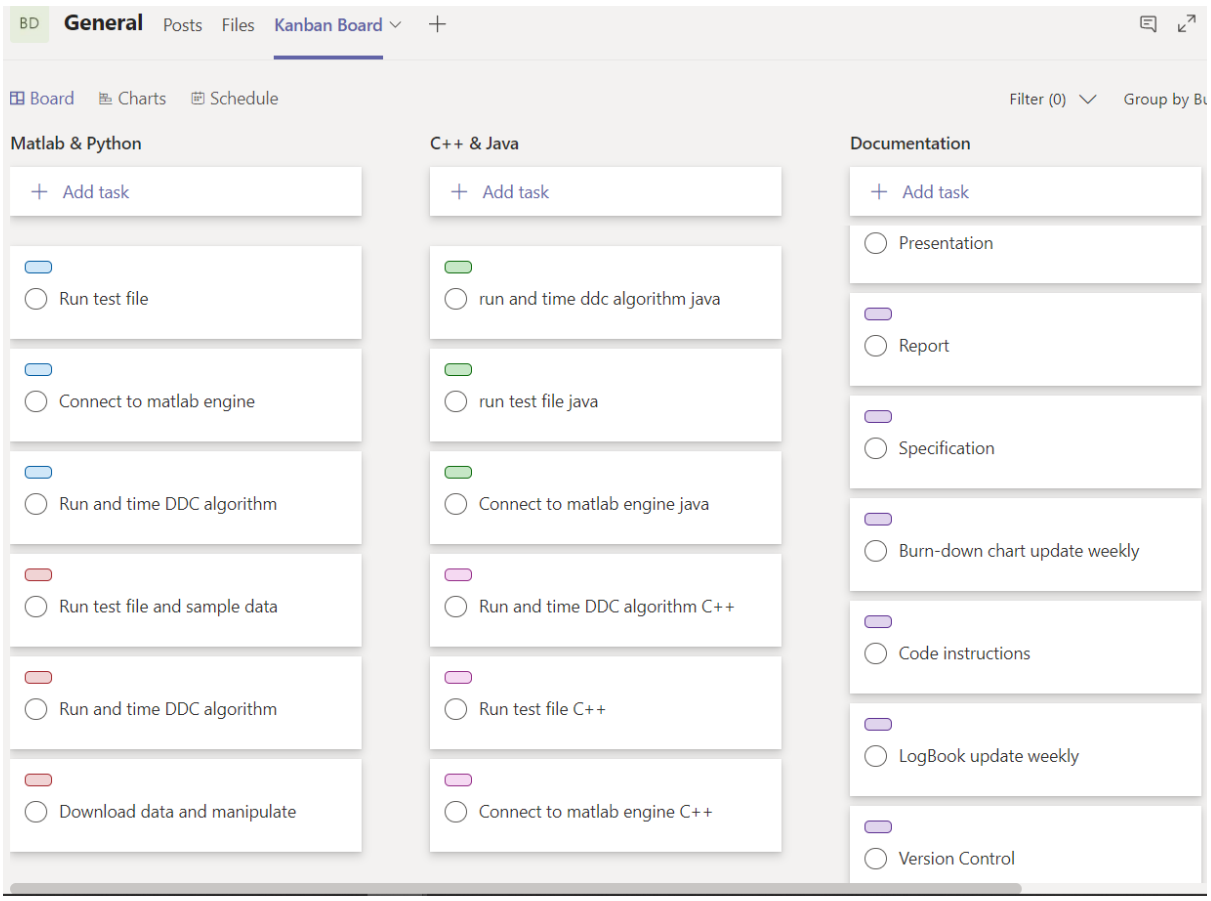
5. Comparison of resulting clusters and data points.
6. Speed comparison of the test program across all applications.

AGILE PROCESSES

Agile processes implemented in project management are weekly sprint planning as documented in logbook, continuous professional developed as measured in logbook, usage of Kanban board and burn-down chart to keep track.

Kanban

As this as an individual project, instead of the typical Kanban board, I made use of a more personalised one with buckets for different aspects of the project. As, the burn-down chart and logbook had to be updated each week, these tasks had sub checklists. All tasks were colour coded to make keeping track easy.



Matlab & Python	C++ & Java	Documentation
Add task	Add task	Add task
Run test file	run and time ddc algorithm java	Presentation
Connect to matlab engine	run test file java	Report
Run and time DDC algorithm	Connect to matlab engine java	Specification
Run test file and sample data	Run and time DDC algorithm C++	Burn-down chart update weekly
Run and time DDC algorithm	Run test file C++	Code instructions
Download data and manipulate	Connect to matlab engine C++	LogBook update weekly
		Version Control

Figure 1: Kanban Board Tasks

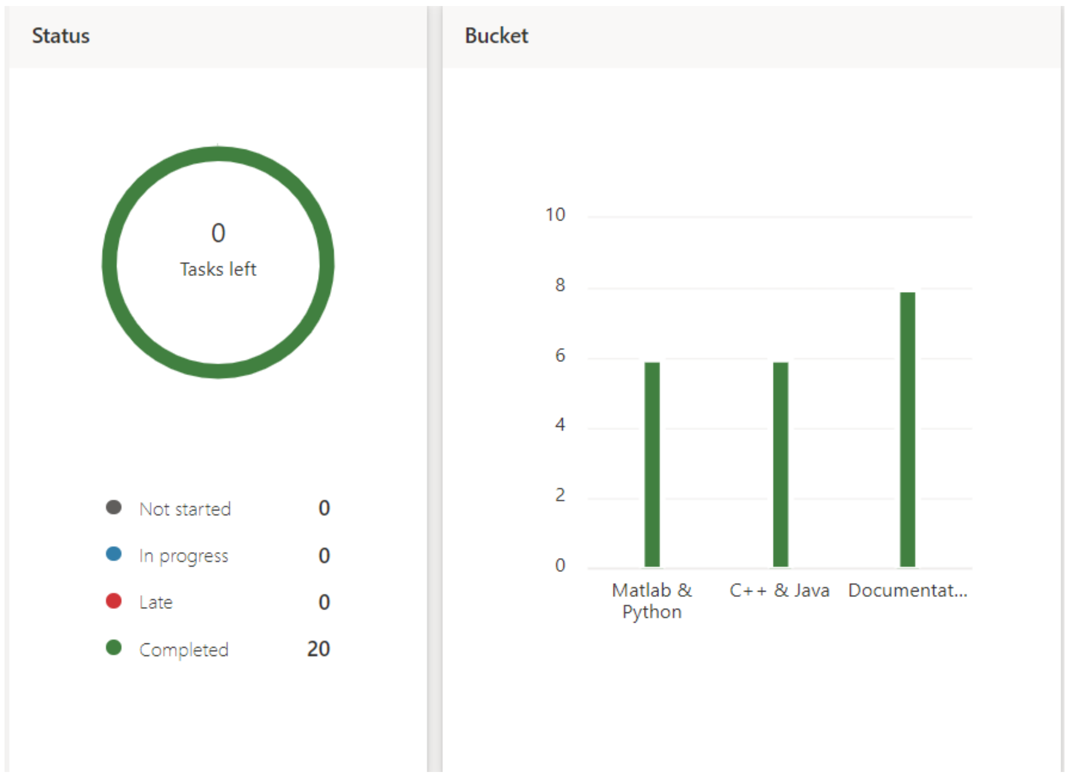
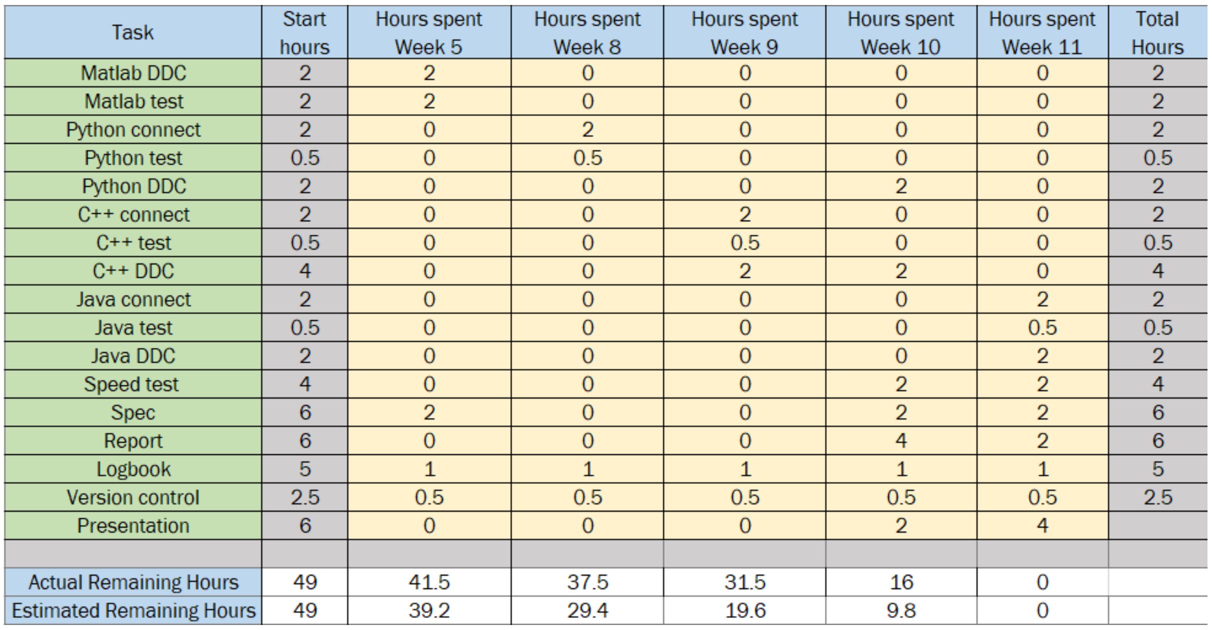


Figure 2: Kanban Board Completed Tasks

Burn-down Chart

The burn-down chart was used to keep track from when coding started, i.e., week 5 in the log. It was updated weekly after each weekly sprint.



Task	Start hours	Hours spent Week 5	Hours spent Week 8	Hours spent Week 9	Hours spent Week 10	Hours spent Week 11	Total Hours
Matlab DDC	2	2	0	0	0	0	2
Matlab test	2	2	0	0	0	0	2
Python connect	2	0	2	0	0	0	2
Python test	0.5	0	0.5	0	0	0	0.5
Python DDC	2	0	0	0	2	0	2
C++ connect	2	0	0	2	0	0	2
C++ test	0.5	0	0	0.5	0	0	0.5
C++ DDC	4	0	0	2	2	0	4
Java connect	2	0	0	0	0	2	2
Java test	0.5	0	0	0	0	0.5	0.5
Java DDC	2	0	0	0	0	2	2
Speed test	4	0	0	0	2	2	4
Spec	6	2	0	0	2	2	6
Report	6	0	0	0	4	2	6
Logbook	5	1	1	1	1	1	5
Version control	2.5	0.5	0.5	0.5	0.5	0.5	2.5
Presentation	6	0	0	0	2	4	
Actual Remaining Hours	49	41.5	37.5	31.5	16	0	
Estimated Remaining Hours	49	39.2	29.4	19.6	9.8	0	

Figure 3: Burn-down calculation

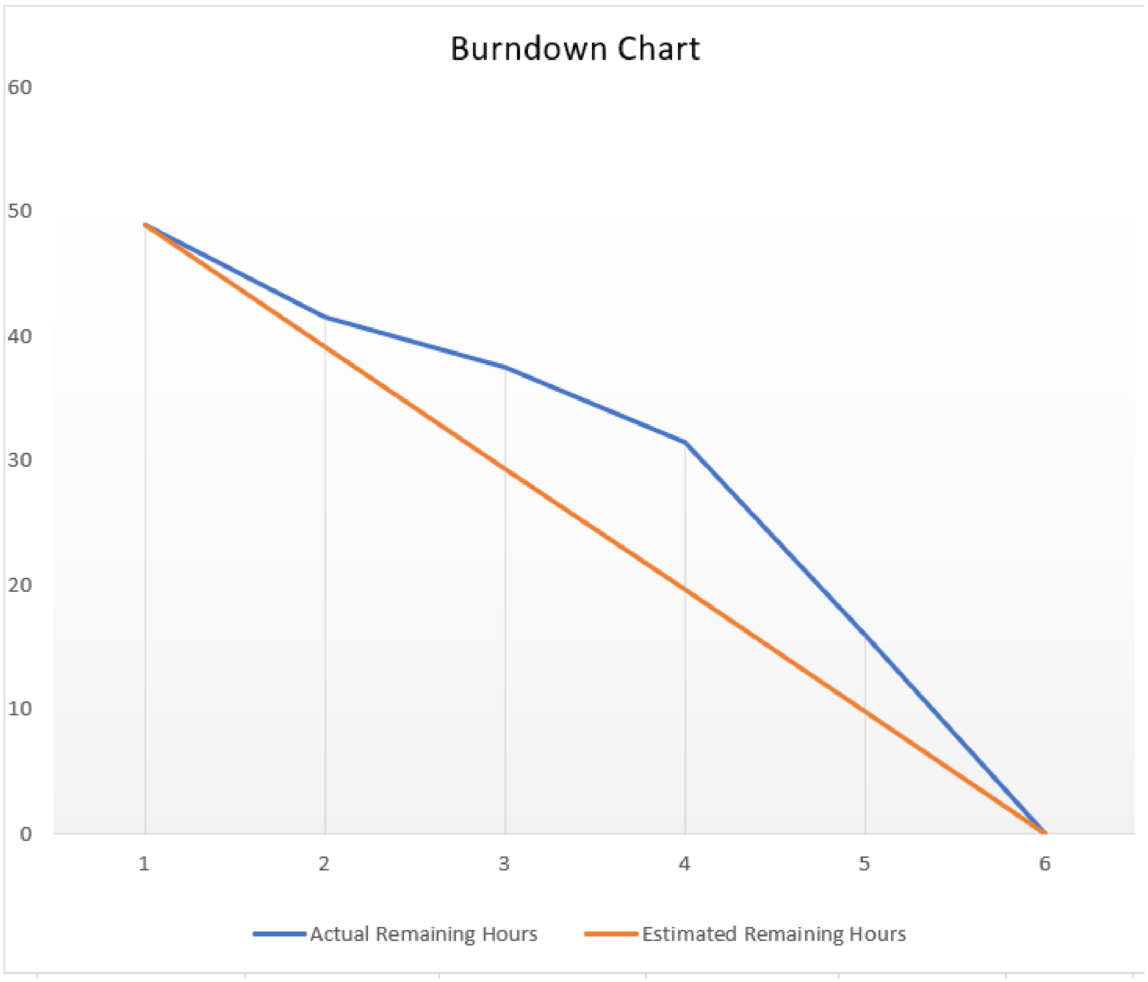
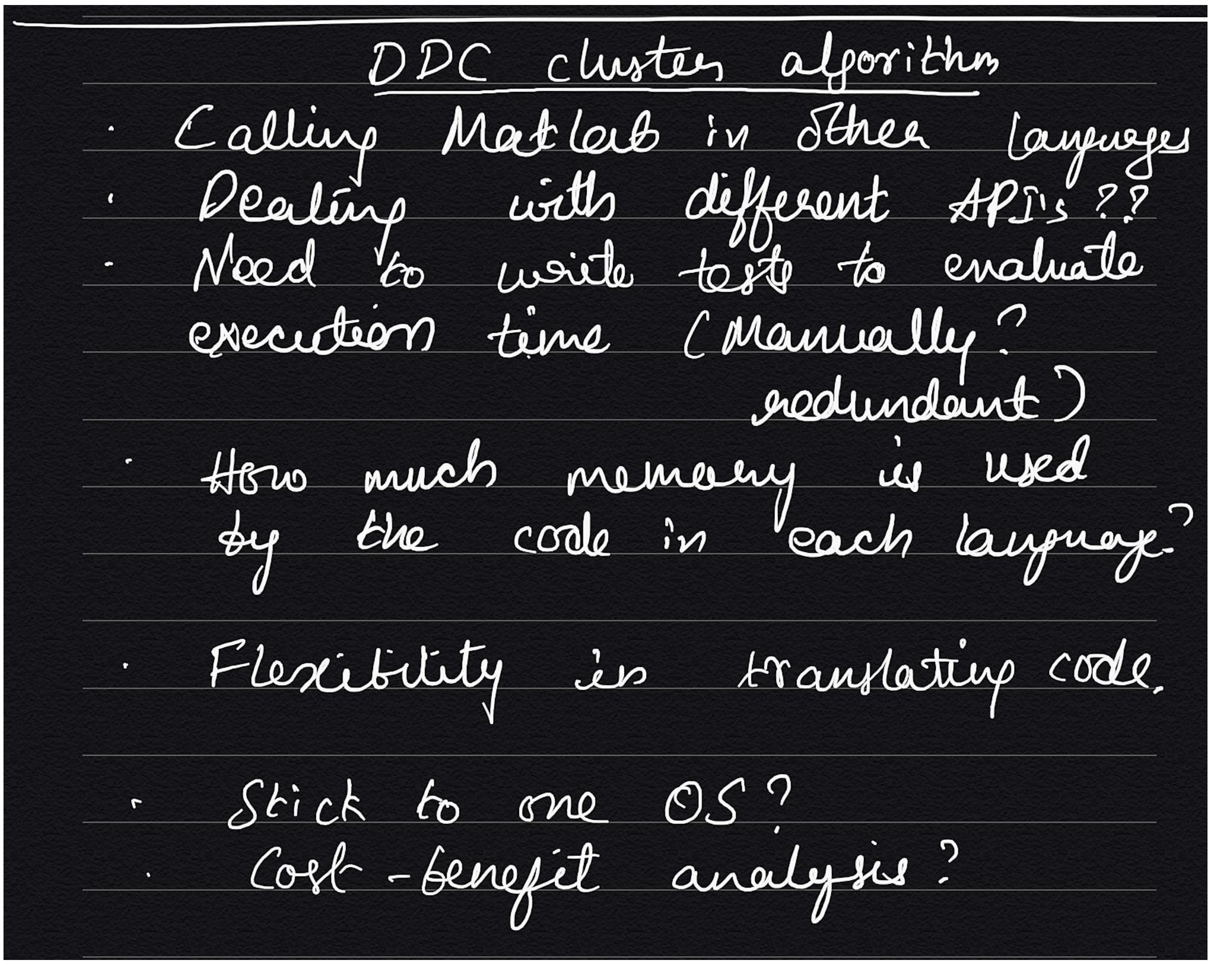


Figure 4: Burn-down Chart

APPENDIX

[1]

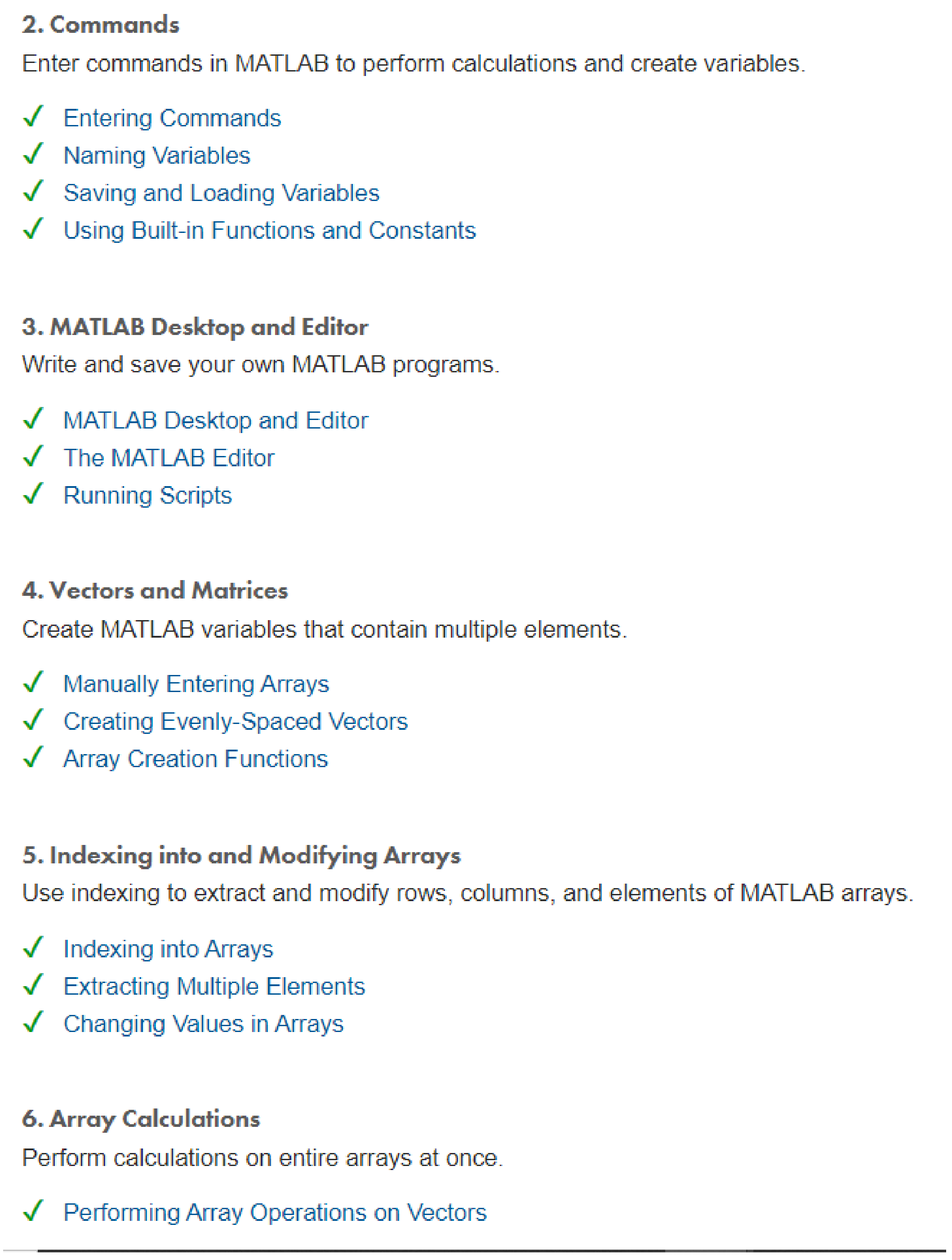


DDC cluster algorithm

- Calling Matlab in other languages
- Dealing with different API's ??
- Need to write tests to evaluate execution time (Manually? redundant)
- How much memory is used by the code in each language?
- Flexibility in translating code.
- Stick to one OS?
- Cost-benefit analysis?

Figure 5: Basic Outline

[2]



2. Commands

Enter commands in MATLAB to perform calculations and create variables.

- ✓ Entering Commands
- ✓ Naming Variables
- ✓ Saving and Loading Variables
- ✓ Using Built-in Functions and Constants

3. MATLAB Desktop and Editor

Write and save your own MATLAB programs.

- ✓ MATLAB Desktop and Editor
- ✓ The MATLAB Editor
- ✓ Running Scripts

4. Vectors and Matrices

Create MATLAB variables that contain multiple elements.

- ✓ Manually Entering Arrays
- ✓ Creating Evenly-Spaced Vectors
- ✓ Array Creation Functions

5. Indexing into and Modifying Arrays

Use indexing to extract and modify rows, columns, and elements of MATLAB arrays.

- ✓ Indexing into Arrays
- ✓ Extracting Multiple Elements
- ✓ Changing Values in Arrays

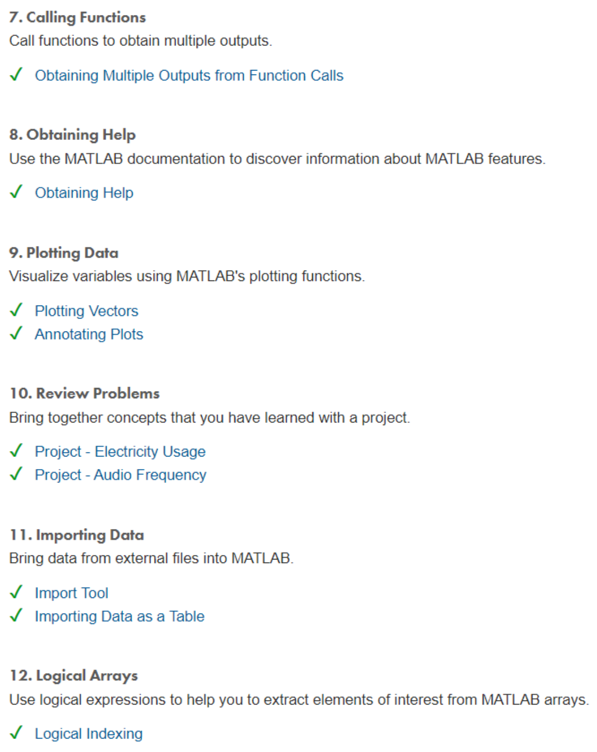
6. Array Calculations

Perform calculations on entire arrays at once.

- ✓ Performing Array Operations on Vectors

Figure 6: Lesson 1-6 Matlab Course

[3]



7. Calling Functions
Call functions to obtain multiple outputs.

✓ Obtaining Multiple Outputs from Function Calls

8. Obtaining Help
Use the MATLAB documentation to discover information about MATLAB features.

✓ Obtaining Help

9. Plotting Data
Visualize variables using MATLAB's plotting functions.

✓ Plotting Vectors
✓ Annotating Plots

10. Review Problems
Bring together concepts that you have learned with a project.

✓ Project - Electricity Usage
✓ Project - Audio Frequency

11. Importing Data
Bring data from external files into MATLAB.

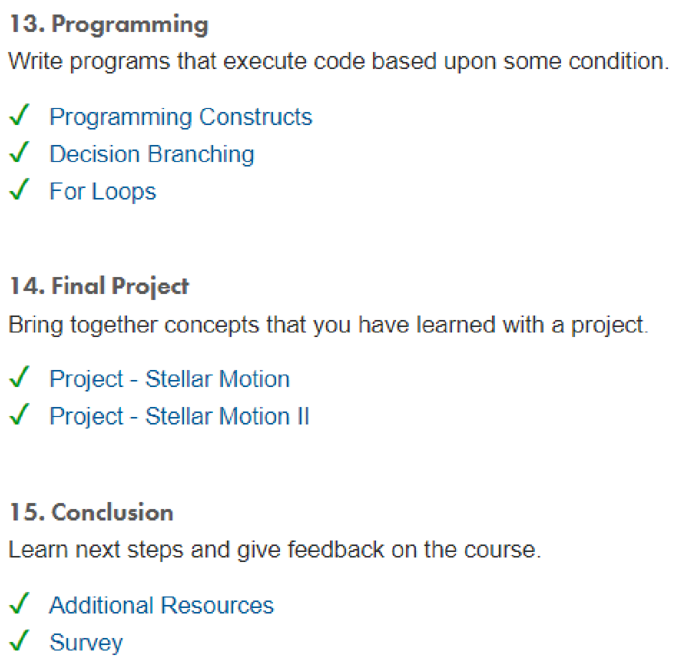
✓ Import Tool
✓ Importing Data as a Table

12. Logical Arrays
Use logical expressions to help you to extract elements of interest from MATLAB arrays.

✓ Logical Indexing

Figure 7: Lesson 7-12 Matlab Course

[4]



13. Programming
Write programs that execute code based upon some condition.

✓ Programming Constructs
✓ Decision Branching
✓ For Loops

14. Final Project
Bring together concepts that you have learned with a project.

✓ Project - Stellar Motion
✓ Project - Stellar Motion II

15. Conclusion
Learn next steps and give feedback on the course.

✓ Additional Resources
✓ Survey

Figure 8:Lesson 13-15 Matlab Course