# Deployment of Application

Within the assignment folder there has been included two versions of the application in the folders 'Original Digital Music Analysis' and 'Parallelized Digital Music Analysis'. With in both of these folders there is 2 sub directories with one being the compiled execution file for a final working release version for both projects, and a second folder called 'Visual Studio Project' that contains the Visual Studio project files for the application.

## Original Application Location

Within the 'Original Digital Music Analysis' file is the base unedited code for the original application without parallelization. With the executable for the original file being in the sub directory 'Original Digital Music Analysis exe' with the .exe file 'DigitalMusicAnalysis.exe' being the complied project for the n parallelized application. The file 'Visual Studio project Original' contains the visual studio project for the original un parallelized application. With this project the user can edit line 45 in 'MainWindow.xaml.cs' via uncommenting the command there to set a hard end point for the application for testing. Along with this line 34 and 35 can be edited so that the two variables on those lines are pointing to the commented-out location if testing need to be done with automatic input. To run the new build the user can run it or build it like any Visual Studio application with the green arrow saying DigitalMusicAnalysis. This will run the program and create a new exe file at the location '' Original Digital Music Analysis\Visual Studio project Original\DigitalMusicAnalysis\bin" either in the Debug or Release folder depending on the chosen build type.

## Parallelized Application Location

The parallelised application can be found within the second folder within the main directory with the file being called "Parallelized Digital Music Analysis". Within this fie again there is 2 folders one called "example 4 core build" which contains a build version for the edited parallelized release version of the application utilizing 4 threads, with the execution file being called "Digital_Music_Analysis_C4.exe". The Visual Studio file can be found within the folder called "Visual Studio project Parallelized", with this file the visual studio project used to create the application and execution file can be found. Within the project the user can edit line 53 in 'MainWindow.xaml.cs' via uncommenting the command there to set a hard end point for the application for testing. Along with this line 35 and 36 can be edited so that the two variables on those lines are pointing to the commented-out location if testing need to be done with automatic input. Along with this the user can edit the core_count variable at line 367 to any number of threads they want the parallelized loop to run across. The other file the user can edit within this project is 'timefreq.cs' with the user being able to edit the variable core_count at line 60, to enable the parallelized for loop to run across the desired number of cores. To run the new build the user can run it or build it like any Visual Studio application with the green arrow saying DigitalMusicAnalysis. This will run the program and create a new exe file at the location ''Parallelized Digital Music Analysis\Visual Studio project Parallelized\DigitalMusicAnalysis\bin" either in the Debug or Release folder depending on the chosen build type.

## Running the application and GUI layout

After opening the application, the user will be prompted to open a .wav file, two different .wav files have been included in the music folder within the main folder, called Jupiter.wav and B.B.scale.wav, the user can either select one of these or their own .wav file.

After opening a .wav file the user will then be prompted to open a .xml file that contains the sheet music to read, one .xml file that is compatible with the software has also been include in the same file as the .exe called Jupiter.xml, which the user can load into the application.

From there the program will run and open the GUI screen and play back the .wav file to the user. The user can then use the tabs at the top to look at the different screens, with the fist screen the user sees being the histogram showing frequencies of the .wav file in an image format. The second screen shows the different pitches played at each interval within the .wav file. The third screen than shows the staff screen showing the sheet music, with the notes played. With red notes showing notes played incorrectly while black notes have been played correctly, the user can than mouse over each note to see the note pitch, note and the relative error between the expected frequency and played frequency.