UNIVERSITY OF WATERLOO

Faculty of Engineering

Department of Mechatronics Engineering

Spatial Filtering Code Organization

Sick Kids Hospital – Department of Ophthalmology
555 University Ave., Toronto, ON, M5G 1X8

Prepared by
Jacqueline Fromme
ID: 20301153
Userid: jkfromme
2B
April 25th, 2011

Jacqueline Fromme
4 Seamist Cres.
Toronto, ON
M1V 3K4

April 25, 2011

Sanjeev Bedi, *Director of Mechatronics Engineering*
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Prof. Bedi,

This report, entitled "Spatial Filtering Code Organization", was prepared as my 2B work term report evaluation. This is my third work term report. The purpose of this report is to analyze the benefits of the new features added to the new design of the spatial filtering code. This code is used with the MEG machine readings to measure the brain activity of patients during experiments.

I was employed in the MEG branch of Dr. Agnes Wong's lab as a research assistant. This lab is part of the department of Ophthalmology at Sick Kid's hospital. The MEG branch is responsible for developing the technology of the MEG machine used to read the brain activity of patients with and without a condition called amblyopia. Amplyopia is a visual impairment condition and this lab is dedicated to doing research to find treatments for it.

I was supervised by Dr. Maher Quraan a research associate of the lab and the MEG team leader. He told me what features he wanted to be added to the code and I implemented, analyzed, and tested them. I would like to thank him for his extensive contribution to this report because he taught most of what I know about the software and MEG technologies. He was also responsible for developing this software over the last 5 years.

I hereby confirm that I have received no further help other than what is mentioned above in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Jacqueline Fromme
20301153

Table of Contents

List of Tables and Figures

Summary

The objective of this report is to describe and analyze the features added to the original design of the Spatial Filtering (SPF) code. This code essentially takes the raw magnetic field data from the magnetoencephalography (MEG) machine and translates it into information about the location of the brain current dipole activations, their magnitude, and their orientation.

This report covers an introduction on how the brain current dipole activations are calculated through the MEG hardware in combination with the SPF code software, but it does not describe in detail the complex calculations that need to be made in this process and the physics behind them. This report is geared towards a software development audience as opposed to an MEG imaging audience.

The features added to the code were central filtering and reading of dataset parameters as well as global access to the filtered data and the dataset parameters. Also, the main part of the program was reorganized by moving some data file calculations into separate functions in dependant files. The code was also made to terminate early if only averaging of the data was requested. The data files of the old code and new code were compared to ensure the code was implemented correctly.

The results indicated that the code processed faster with the centralized filtering and reading of the dataset parameters features added. Also the main part of the program was shortened with the calculation functions moved to other parts of the code. This also makes the code look much cleaner. Accuracy of the code was affected because the order of the code filtering and averaging was optimized in the new code. This had insignificant effects on the results of the code.

It is recommended that the new version of this code be used in all future endeavors because it benefits users in terms of readability, accuracy, flexibility, and most of all efficiency. It is also

recommended to continue reorganizing the main part of the program to make it cleaner and to implement a software version control system in the future.

1.0 Background Information

The magnetoencephalography (MEG) machine is used to measure the magnetic field created by small dipole currents in the brain and pinpoint their location. These brain currents are caused by electrical action potentials that occur naturally in neurons. When a neuron fires it creates a negative pole and when it depolarizes it creates a positive pole due to ion flow. These dipoles create small currents over short distances between synapses (Hansen et al). By mapping these current sources we can find the regions of the brain with the most activity at different time points during an experiment. This data is useful in many psychophysical studies.

Superimposing the data from the MEG sensors on a structural magnetic resonance image (MRI) of the brain, we can pinpoint exactly where these current sources are (within 0.5mm). We are also able to tell their orientation, and their magnitude in (N*A*m).

To collect the data from the MEG machine for an experiment, typically the subject is exposed to a stimulus that activates some part of their brain and measurements of their brain currents are taken throughout the experiment using a sensor helmet on the patient's head. The MEG magnetic field sensor measurements induced by the brain currents are taken at a rate of 625 samples per second and are taken over a window of time, for example -250ms to 2,000ms. At 0 ms the stimulus would be presented. The data is usually collected for a large number of runs (100 or more trials) for each experiment dataset.

The raw magnetic field data from the MEG sensors is run through a program written in C language so that it can be calculated, filtered, and averaged. This program is referred to as the Spatial Filtering (SPF) code. This report is about the restructuring of this SPF program. The SPF code originally came from (***) (with hardware or made up?).

To compute the current dipole localizations in the brain, the SPF code first reads in the magnetic field data file of each sensor for each time point during each trial. Some sensors are brain sensors and some are reference sensors. Reference sensors sit a bit higher than brain sensors in the helmet and they are used to measure the outside noise. The noise each brain sensor is exposed to can be removed from the signal knowing the reference measurements. Then, the data is band pass filtered so that any readings above or below the frequency of the target brain waves are discarded. The program also averages the magnetic field readings over these trials so that any outside noise or other non-stimulus brain waves are averaged out. Finally, the SPF code divides the brain into a spherical 3D grid of small 0.5mm long cubes called "voxels". If a dipole current exists in that brain voxel it should create a certain magnetic field in each surrounding sensor, assuming uniform conductivity of the brain, and given the dimensions of the brain from the MRI or using a standard head shaped template. Based on this principle, the dipole current of each voxel is calculated using a complex mathematical system of equations called the forward and inverse solution. It is out of the scope of this report to describe how the different methods of forward and inverse solutions are calculated and why some inverse solution methods are better at localizing brain signals than others, but they all use the aforementioned principle. The code can perform a variety of inverse solution methods if specified to.

The final results of the SPF process are printed to files. Some are text files including the data average (DA) file and the global field power data average (GFPDA) file. The DA file shows the magnetic field data from each sensor averaged over all trials at each time point. Again, this file will contain brain activity only due to the stimulus because of the averaging. This file is useful to look for bad sensors with strange data at different time points. The GFPDA file shows the average of all sensor DAs at each time point, this is useful to mark the point in time of the most

activity in the brain caused by the stimulus. By running the SPF code again inputting a time parameter of interest (usually time of highest activation) we can generate a file of all the source activation readings in each voxel of the brain at that time. This file is called the image data (ID) file. The ID file values can be used to make a picture of the source activations in the brain due to the stimulus at a certain time, this picture is called a functional image and it is what is laid over the MRI. In this way we can find correlations between different stimuli and the activation of different parts of the brain. There is also another feature that can be chosen to create a virtual sensor (VS) file. It shows the vector of each voxel's current source over a chosen range of time from which we can find the source's magnitude and orientation. It is called a virtual sensor because sensors measure source activations over time and this file contains source activations in each little voxel over time so it's as if a sensor were placed directly in a voxel. This file is useful to see source activation over time at a specific location when it is plotted.

It is useful to know the purpose of these data files because they will be compared between the old code and the new code to ensure that the new code design is working.

2.0 Spatial Filter Code Organization

2.1 Original Design

The SPF code is very large and uses a number of dependent files. Since a code of this size is too difficult to explain in its entirety, only those parts involving the new features will be mentioned. The main goal of this project is to organize the code, make it easier to read, and have similar calculations happen in only one place in the code.

A flow chart of the old and new code with its added features is included in Appendix A.

The first feature added is a function called ProcessData that creates a global array called SensorData, containing the filtered sensor data of the dataset. Making something global means that it can be accessed anywhere in the code, including in dependent files (files not containing the Main function of the program), which is very convenient. Originally, the sensor data was read in from a file and re-filtered every time the program needed filtered sensor data in a different part of the code. These parts of the code are shown on the flow chart for the old code in Appendix A. The new design does the filtering and reading once in ProcessData and stores the information in this SensorData array to be accessed everywhere.

The second feature of the new design is that the dataset parameters structure, dsParams, is made global. The dataset parameters are read in from a file, they include a lot of information about the dataset. Some of the dataset parameters dsParams stores include the number of trials and samples, the high pass and low pass cut off frequencies, and they a record for each sensor giving the position and gain of each sensor, and which sensors are references for which sensors. Each sensor record is used in many calculations to return a magnetic field value for that sensor. There are over 150 sensors. In the original code design, whenever the dataset parameters were needed,

they would be passed to or reread inside of the function that needed it. Therefore, by making it global we can bypass doing this multiple times and clean up the code a lot.

Finally the third feature has to do with increasing the SPF code organization and functionality. The parts of the code that compute the data average, image data, virtual sensor data, and write this information to result files, were removed from the Main of the code. Virtual sensor, image data, and averaging functions were made in separate dependent files so that the code could be more organized. As well, by making these processes separate from the main they could easily be controlled by adding options that only access these parts of the code to create and compute these files when they are requested. In the old design, as seen in the flow chart, the code would only terminate early if the VS option was chosen, in which case it would terminate after VS files were made. This happens near the end of the code. Otherwise, the code would continue on and make the image data, which is the very last part of the code. The option to make average data files at the beginning of the code and terminate early without computing image data or virtual sensor data was not available. So, this became part of the new design since the average files are commonly used.

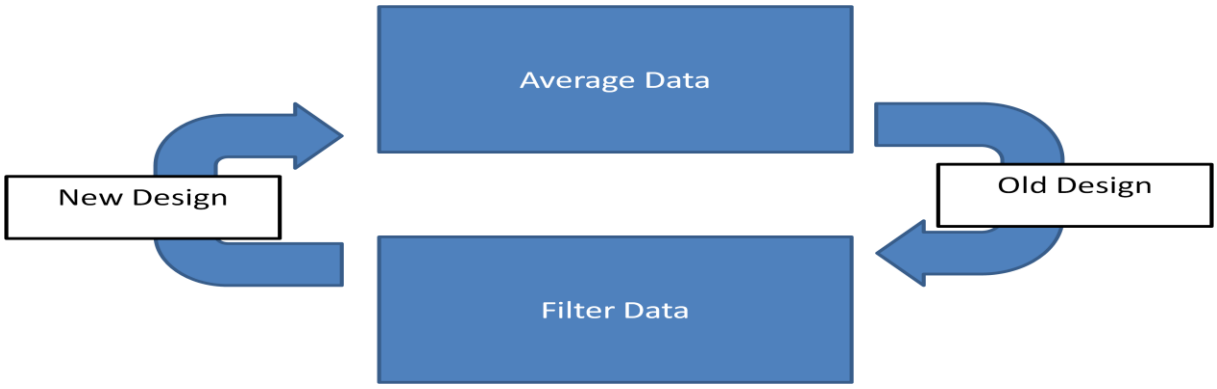
3.0 Implementing the New Design

3.1 Central Filtering

The first central filtering feature was implemented by creating the new ProcessData function. This function does three things: build the filter, apply the filter, and subtract the DC offset from the sensor data. Building the filter sets up the filter parameters. Applying the filter removes the data outside the requested bandwidth. Subtracting the DC offset averages the data and subtracts this average assuming there is a current offset of this magnitude. These three steps are all necessary for the filtering process and appear throughout the old code in the places labeled as filter data on the diagram in Appendix A. But, now they will be done in one place in this new function. The ProcessData function will return a 3D array of the filtered data from all trials, sensors, and samples. This array will be generated near the beginning of the code so that any functions that need to access it later on in the code have it at their disposal.

One of the main ProcessData implementation problems is the order of averaging and filtering. There are different averaging techniques in different areas of the code. In order to keep the filtering consistent, the data must be filtered first in ProcessData, and then averaged. In the old design, with multiple filtering, the data is first averaged, and then filtered again, which is the opposite order of the new design. The diagram below illustrates this.

Figure 1 - Difference in Averaging-Filtering Process Between Versions of SPF



Since this change is made in the process of averaging and filtering, the final results varied a bit from the original results.

3.2 Global Dataset Parameters

The dataset parameters, dsParams is one of the most widely used data structures. It is used by practically every file in the code. A special header file for dsParams had to be made so that the structure of it could be accessed anywhere in the code.

There is a function that reads dsParams that is called very often. It reads the dataset parameter information from a file. This function is called very often because it can read the file for sensors only information or it can read the file for sensor and reference sensor information depending on why it was called. This new design ensures that dsParams is only read once at the beginning of the code and stored so that it can be used globally everywhere. Hence, the dsParamsRead function needed to be adapted so that the information provided by it is enough for both when sensors only information is needed and when sensors and reference information was needed. Therefore, dsParams now has to have a sensors only count and a total count including sensors and references. If users want to use data for only sensors they must check the dataset parameter

isSensor to see if it's not a reference first. This is a consequence of reading the dataset parameters only once that user's must be aware of.

3.3 Adding Options to the Main Program

Moving the average, virtual sensor, and image data processing sections out of the main section of the code was very straight forward to implement. It involved copying and pasting sections of code, then moving them to other files, and making them into functions. These functions needed to have many parameters passed to them from the Main with the arguments required to perform the calculations. The benefit of this compared to the amount of effort required was quite large. Also the code was changed so that it could compute and write average files only and terminate early if requested.

4.0 Stress Tests

The SPF filtering process, ProcessData, was tested by running both the old and the new versions of SPF with a non-previously filtered dataset. The resulting DA, GFDA, ID and VS files were compared for the old and the new code. The results for these differed slightly. Therefore, to test that the results only differ because of the order of the filtering and not the implementation of one central filtering function, the filtering in the old code was switched to match the order or filtering in the new code. In this case, the results were the same. Therefore, this feature was implemented successfully.

The global dsParams was tested in the same fashion. Since this feature was added after ProcessData function was added it needed to be tested with the slightly modified version of the old code as well. The results of the data files between the old code and the new code matched, so this part was implemented successfully.

Finally, the ID, VS, and averaging options were tested by running different combinations of options and comparing the aforementioned data files with those of the slightly modified old code. These data files were the same. Therefore, this part of the code was also implemented successfully. Also, the data files that weren't requested in the command line weren't generated, so this new option system was working well.

5.0 Analyzing the New Design

5.1 Efficiency

The new ProcessData function, where the code is filtered only once, made the code a lot more efficient. Each time the data is filtered each trial has to be read in and filtered one by one. Since one trial contains roughly 100,000 samples and there are usually about 70 to 80 trials in a dataset, it is a lot of work. Extra, repeated filtering results in a longer processing time and puts a larger load on the server.

The global dsParams structure in which the parameters for the dataset would only be read once, increased the efficiency of the code but not greatly. There are only 125 sensors and 25 reference sensors, therefore it is not a lot of work to reread a list of parameters for each sensor. But, because the read dsParams function call appeared so often in the old code design, the most out of any function call in the program (10 times), it could save a little bit of processing time by only doing it once.

Lastly, by adding the option which allows the program to terminate early if only data averaging is requested, one can save a lot of processing time. Computing the image data and virtual sensor data requires many computations to find the forward and inverse solutions and take the bulk of the processing time. The averaging only requires a small amount of computation. Therefore, with the new feature added, the averaging could be done very quickly without the image data and virtual sensor data generated. If one of the ID or VS options are chosen majority of the longer calculations must be performed and not much time is saved if the option to write to the average files is turned off. This effect can be seen in Figure 2 on the next page.

In summation, all features added should benefit the efficiency of the new version to some degree.

In order to test this hypothesis the old code and the new code are run with different options and their user times are recorded to see which one completed the requested files in a faster time. The results of 5 trials of each set of options were averaged to get the most optimum results.

Figure 2 - User Time of SPF

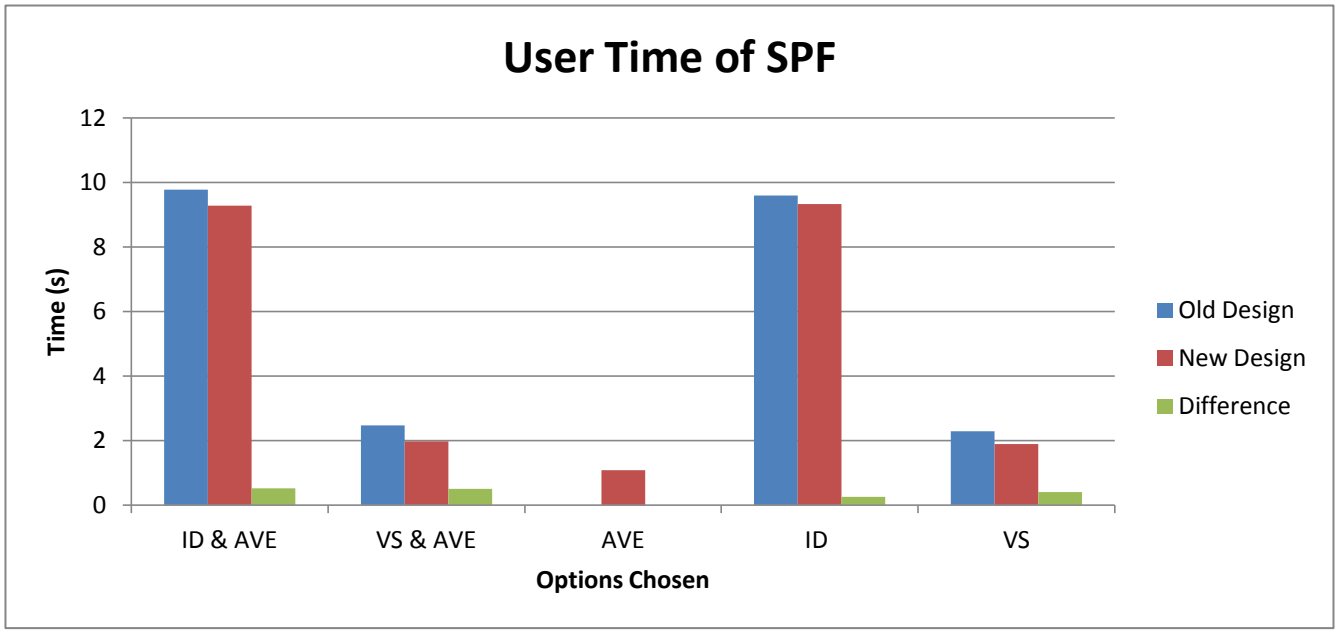


Table 1 - User Time of SPF Data

	Old	New	Diff Old & New
ID & AVE	9.78	9.27	0.51
VS & AVE	2.461	1.971	0.49
AVE	ERROR	1.083	n/a
ID	9.586	9.33	0.256
VS	2.291	1.885	0.406
		Average Diff	0.4115

As seen above, the new code is faster than the old code in all cases. On average the user time of the new design is 0.4115 seconds faster than the user time of the old design. The reason the new code is faster by a similar amount each trial is probably because of the central filtering and the

central reading of the dsParams, which would affect all differences between the old code and new code to the same degree.

Since the options are the same for both designs in every case except for the AVE only case, in which the old code generates an error, the code is terminating at the same point between the old code and the new code. Therefore, the third feature didn't benefit the code in terms of processing time except in the AVE only case. It appears that if the user only wanted to generate average files they could do this in 1.083 seconds as opposed to 2.5 seconds or 9.78 seconds because the new design allows the code to stop after the average files are generated and not process the ID and VS information.

5.2 Readability

Centralizing the filtering process and the reading of the dsParams has made the code much easier to follow compared to the way they were sporadically placed multiple times in different parts of the code in the old design. Also, because dsParams and SensorData arrays are now global it's easy to recognize them in all files because the data in them will always have the same name. The size of the main of the program was also cut down a fair bit when image data and virtual sensor data processes were removed from the main and put into other dependent files. Now there are just function calls to these which makes the main much easier to follow. There is also a function call to create average files which was taken out of the main as well. The main of the old version of code was 2282 lines long and the new version of code has a main that is 1755 lines long. This means the main code was cut down by almost a quarter because of this change.

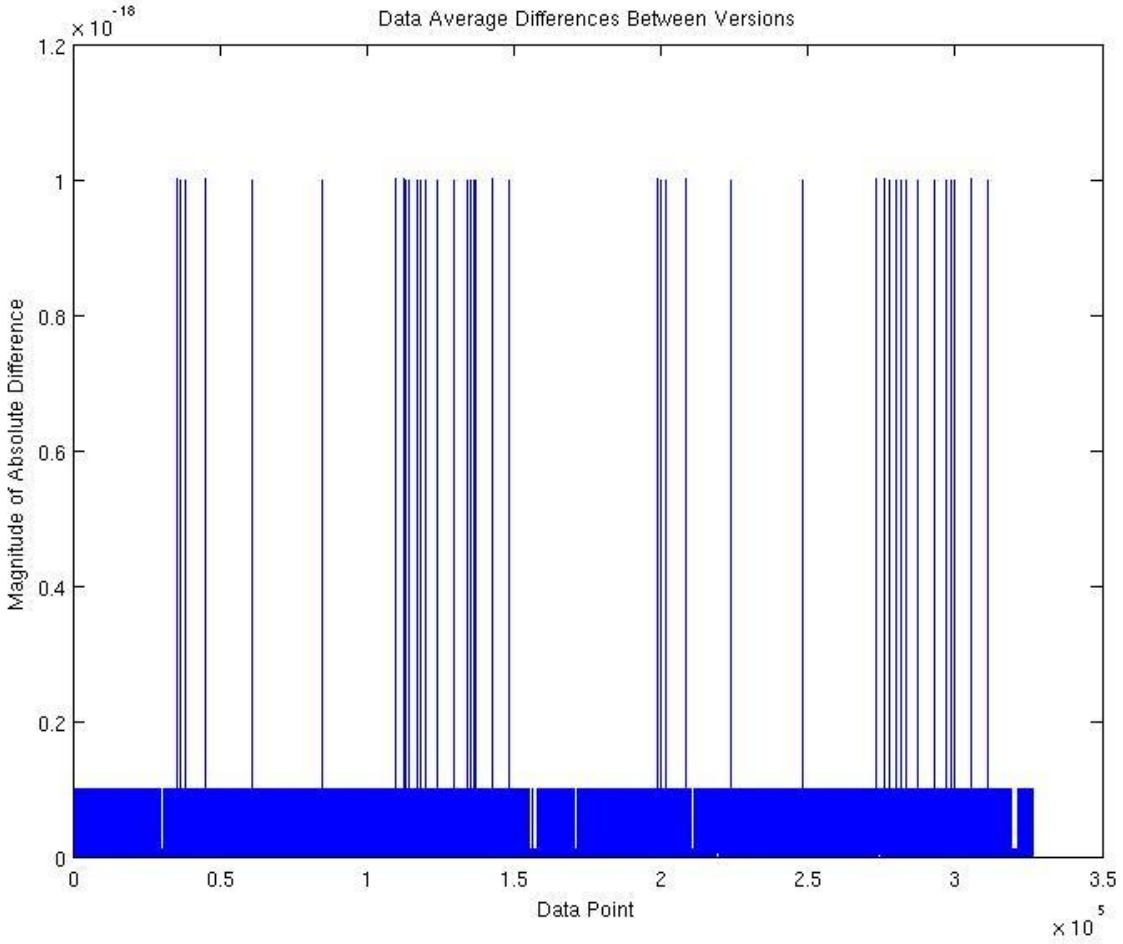
5.3 Code Flexibility

The code has become more flexible in the sense that the most used structures are now available globally. Future users won't have to deal with functions required to retrieve the data for dsParams and SensorData array because the information will already be there. Although, now the sensor records will always contain data for both sensors and references so the user must check which it is before using it, where as in the old design they might have reread the data for the type of data they needed. A future recommendation would be to make this more explicit, by organizing dsParams information better.

5.4 Accuracy

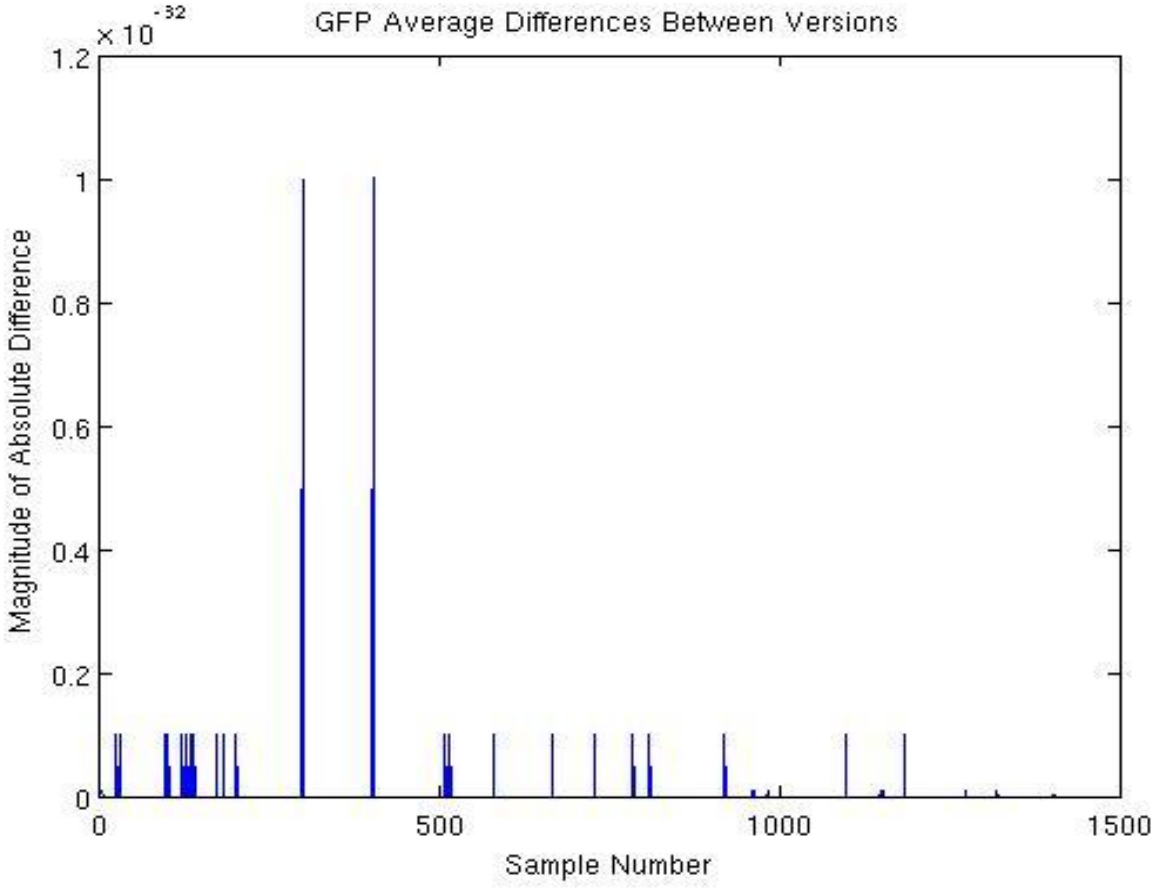
As mentioned in the implementation section, the results of the new code are slightly off from the results of the old code. This is because the filtering is done before the averaging in the new code, whereas in the old code, the averaging was done before the filtering. Filtering means that the signals read that are above or below a certain frequency (typically 1 to 58 Hz) are discounted. It would appear that filtering first would make more sense so that no bad data is included and no good data is excluded in the averages. The reason the code was originally written the opposite way is not known. There is a possibility that there would be less edge effect with less samples to filter if the samples are first averaged, in which the old way would be better (Quraan). The final results indicate that only a small difference is found in the results when running both versions with the same dataset, the graphs are shown below.

Figure 3 - Data Average Differences Between Versions



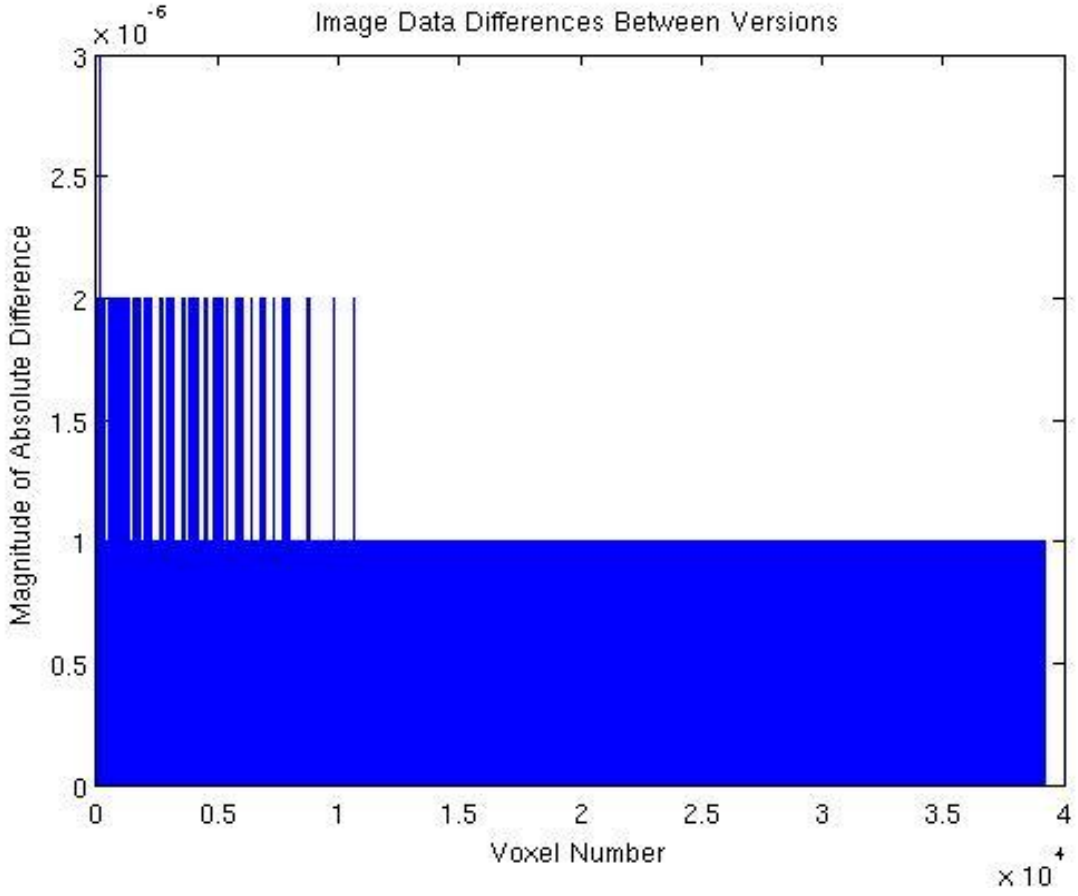
The maximum value of the difference in the data average is $1x10^{-18}$ and the average is $6.8x10^{-21}$. Since typically these values are in the $1x10^{-14}$ range of magnitude, these differences are insignificant.

Figure 4 - GFP Average Differences Between Versions



The maximum difference in the GFP data average is $1x10^{-32}$ and the mean is $6x10^{-35}$. As you can see most differences are 0. Since this data is typically in the 10^{-27} range of magnitude, these differences are insignificant.

Figure 5 - Image Data Differences Between Versions



The maximum difference in the image data is $4x10^{-6}$ and the mean is $2.7x10^{-7}$. Since this data is typically on the order of magnitude of $1x10^{1}$, these differences are insignificant.

Lastly, the localization of the brain activity laid over the MRI was compared and the sources were found to be in the same place with the same peak magnitude.

Therefore, it can be concluded that the new code has the same accuracy as the old code with insignificant changes in results due to the rearrangement of the order of the filtering averaging process.

6.0 Conclusions

It is concluded that the new design for the SPF code is stronger than the old design. The features of the new design were implemented successfully and the results are comparable with those of the old design. The new design of the code was analyzed in terms of flexibility, accuracy, readability, and efficiency. Improvements from the original design were found in each of these areas.

Efficiency

The code was found to have the biggest improvement in this area. The time that it takes to run the new code with the same options as the old code is now almost half a second less. This benefit is due to the central filtering and central reading of the dsParams which go through large sets of data and used to be done repeatedly in the old code. Also, now the average files on their own can be created very quickly due to the new option added to create the average files without creating VS or ID files.

Readability

The source code itself is more readable and easier to maintain because data filtering and reading of the dsParams are only done once throughout the entire code. Also, data file calculating and writing are all done in separate functions in dependant files and are called from the Main. The size of the Main was cut down by a quarter because this.

Flexibility

The often used filtered SensorData array and the dsParams structures can easily be accessed anywhere in the code because they were made global.

Accuracy

The results of the new code have insignificant differences from the results of the old code. It is also suggested that the filtering before averaging in the new design is the more logical and correct approach. Both codes give the same localization pictures in the end with the same peak activation magnitude and location.

In conclusion future code users should use this design as opposed to the old design because the code's performance has increased in terms of flexibility, readability, accuracy, and most prominently efficiency.

6.0 Recommendations

It is recommended that the new code is better than the old code in terms in efficiency, accuracy, readability, and flexibility. Therefore, the new code should be used in all future endeavors.

The SPF code should also be organized further so that majority of the calculations performed in the Main would instead be performed by functions in dependent files and accessed from the Main using function calls.

Also, dsParams should be split into dsParamsSensors and dsParamsReferences so future users aren't confused by the content of dsParams.

This code is also very complex and should have a software user's manual that is consistently updated while the code is updated. There is a lot to learn about this code, but there isn't any material to learn it from other than by asking the supervisor. Future users will benefit greatly from a manual.

Lastly, this lab will benefit greatly by implementing a software repository system such as subversion control (SVN). A couple days were spent integrating this version of the software with the work of another co-op student's whose changes could have been better documented and more easily merged with the new version by using SVN.

References

Quraan, Maher. Personal Interview. April 25, 2011.

Hansen, P.C., Kringlebach, M.L., & Salmelin R.
MEG: An Introduction to Methods. Oxford University Press, 2010. Print.

Glossary

Data Average (DA) – data file generated from SPF that shows the magnetic field data from each sensor averaged over all trials at each time point.

Dependant files – files that are not part of the Main function, generally they contain utilities used by the Main.

dsParams – dataset parameters structure containing information about the dataset.

dsParamsRead – function that reads the dataset parameters in from a file and stores them in the dataset parameters structure.

Functional image – image of brain localization generated from the data from the SPF code.

Global Field Power Average (GFPA) – data file generated from SPF that shows the average of all sensor DAs at each time point.

Image Data (ID) – data file generated from SPF that shows all the current source activation readings in each voxel of the brain integrated over a given time region.

Magnetoencephalography – a machine that reads and locates small currents in the brain using magnetic field sensors.

Main – the main function of the code where the code begins and ends.

ProcessData – a function that reads in sensor data, filters it, and returns a global array of filtered sensor data.

Reference sensor – used to measure the magnetic field surrounding the MEG sensors to remove outside noise.

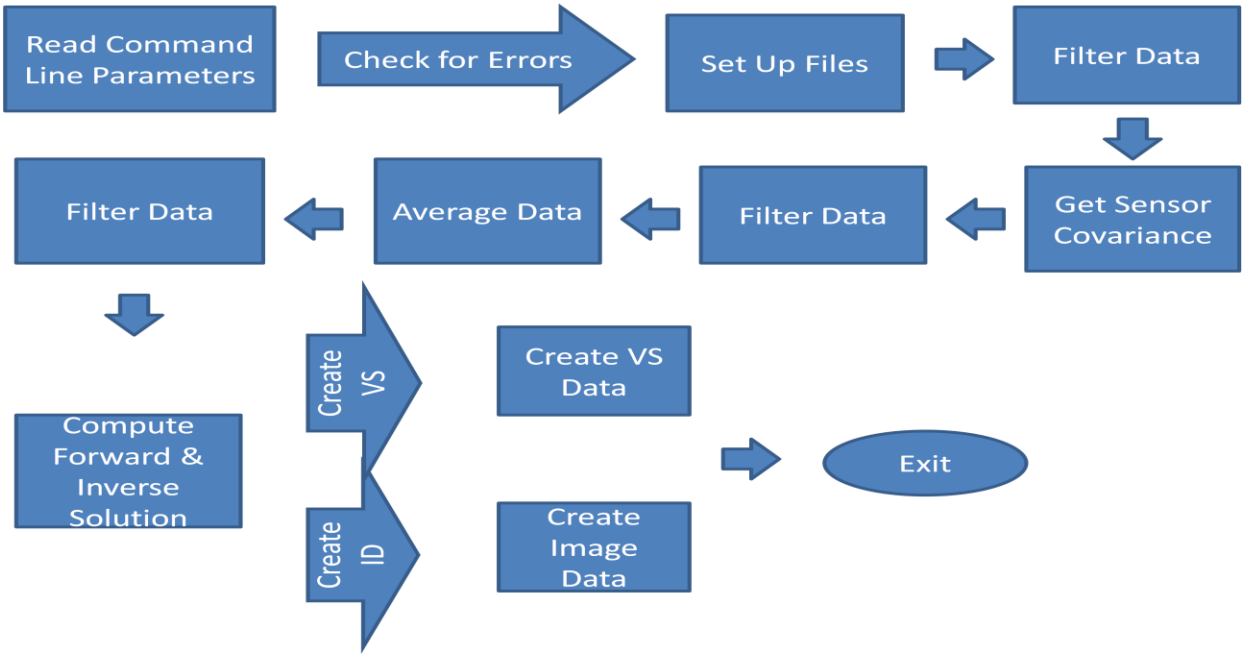
Sensor – the part of the MEG machine used to measure the magnetic field of the brain.

User time – amount of time it takes to process a code excluding the time it takes to process system calls to the kernel. This time should not include the extra time it takes to use the operating system due to a high server load.

Virtual Sensor Data (VS) – data file generated from SPF showing current source activation over time of each voxel.

Appendix A – Flow Chart of the Old & New Code

Old Code Design



New Code Design