REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Technical Report	-

4. TITLE AND SUBTITLE

Development and Implementation of GPS Correlator Structures in MATLAB and Simulink with Focus on SDR Applications: Implementation of a Standard GPS Correlator Architecture (Baseline) Implementation of the MIT Quicksynch Sparse Algorithm Development and Implementation of Parallel Circular

5a. CONTRACT NUMBER
W911NF-11-1-0180

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
206022

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHORS

Damian Miralles, Manuel Ortiz, Jennifer Sandoval, Marvi Teixeira

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

Polytechnic University of Puerto Rico
377 Ponce de Leon Ave.

Hato Rey, PR 00918 -0000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
58923-CS-REP.22

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

We developed a basic correlator design (baseline) using MATLAB code and a Simulink model. This standard method is used to validate the results and performance yielded by new techniques when actual GPS satellite signal records.

We also developed our own MATLAB code for the MIT Quicksync algorithm [2] and implemented this algorithm in a Simulink model that, at the moment, works only with simulated data. The MATLAB implementation of the Quicksync algorithm [2] takes into consideration the Doppler Effect in order to be able to validate our

15. SUBJECT TERMS

GPS Correlators

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Marvi Teixeira
UU	UU	UU	UU		19b. TELEPHONE NUMBER 787-622-8000

Report Title

Development and Implementation of GPS Correlator Structures in MATLAB and Simulink with Focus on SDR Applications: Implementation of a Standard GPS Correlator Architecture (Baseline) Implementation of the MIT Quicksynch Sparse Algorithm Development and Implementation of Parallel Circular Correlator Constructs.

ABSTRACT

We developed a basic correlator design (baseline) using MATLAB code and a Simulink model. This standard method is used to validate the results and performance yielded by new techniques when actual GPS satellite signal records. We also developed our own MATLAB code for the MIT Quicksync algorithm [2] and implemented this algorithm in a Simulink model that, at the moment, works only with simulated data. The MATLAB implementation of the Quicksync algorithm [2] takes into consideration the Doppler Effect in order to be able to validate our implementation with real GPS signal records.

Another development is the formulation and implementation of alternative parallel architectures to perform a circular correlation by decomposing the initial circular correlation into several smaller circular correlations. Such sub-correlations are independent of each other and can be processed in parallel [3]. Even though these results may be applied to any system that performs circular convolution or circular correlation, we will apply it to the acquisition of Global Navigation Satellite System (GNSS) signals using a, FFT-based, Parallel Code-phase Search (PCS) on the GPS L1 C/A signal. The parallel approach may have advantages for hardware-based implementations using Field Programmable Gate Array.

We also developed a preliminary customized Simulink library [3]. A block library is a collection of blocks that serve as prototypes for instances of blocks in a Simulink® model. Since Simulink libraries do not get simulated the library will give us advantages in the models performance. The blocks in such libraries are blocked, which gives code protection so that it cannot be unknowingly altered. Another safety measure of a library is that all blocks are linked and have the same implementation. If further developments are required, there is no need to go to the models and change the block structure, we only have to change the blocks in the library model.

Finally we are in the process of developing a web page to increase the availability of the GPS signal records with our results featuring the signal delay and Doppler shift regarding the satellites present in the signal record. This type of data is scarce in the internet and, if found, the majority does not include any validation information in order to verify which satellites are present. In this site we also plan to include algorithm source code to assist in the discussion, study, experimentation and research on the topic.

Technical Report

W911NF-11-1-0180
58923-CS-REP

Development and Implementation of GPS Correlator Structures in MATLAB and Simulink with Focus on SDR Applications

Implementation of a Standard GPS Correlator Architecture (Baseline)
Implementation of the MIT Quicksynch Sparse Algorithm
Development and Implementation of Parallel Circular Correlator Constructs.

Damian Miralles
(Undergraduate Research Assistant)

Manuel J. Ortiz
(Undergraduate Research Assistant)

Jennifer Sandoval
(Undergraduate Research Assistant)

Marvi Teixeira
(Principal Investigator)

Polytechnic University of Puerto Rico

May 2014

Contents

List of Figures

List of Tables

Introduction

Global Navigation Satellite Systems (GNSS) is the standard generic term for satellite navigation systems [1]. The GNSS allows electronic receivers to determine their location (longitude, latitude and height) with a precision of a few meters using time signals transmitted along a line-of-sight by radio on satellites.

Most receivers are built using hardware to perform most of the tasks. Recent approaches are focused on developing a software approach that adds flexibility to the hardware implementation. The goal is to use a software solution as close to the antenna as possible. The objective of this work is to develop a set of tools that can be used to test and implement more affordable and robust GPS receivers.

One of the main advantages of the software defined receiver (SDR) is the flexibility of such design that allows software reconfiguration of the SDR, which would incur in high cost if performed in hardware.

In this project the objective is to design and implement circular correlator architectures in MATLAB and Simulink that are suitable to be used in GPS receivers. These correlators should be amenable to a software defined radio (SDR) implementation and will be compared to a standard, baseline, circular correlator design. To this end we considered an implementation of the, MIT developed, Quicksynch algorithm [2] and, since our cyclic parallel convolution algorithms map with minimal modification to parallel correlator structures, we also explored the development of new constructs for the implementation of parallel correlators [3]. The designs should be able to work with GPS modeled data and with real data recorded from satellites in order to be able to test and validate its performance.

Summary

Due to the massive use of GPS technology in our personal life, government and in the private sector, there has been a steady desire for improving the existing GPS algorithms. The traditional detection method will be called the “baseline” method. We also tackled the implementation of another technique of synchronization developed at MIT called the “Quicksync” algorithm [2], which is based on the sparsity of the correlation results. Parallel circular correlator constructs were also explored [3].

The satellites being identified by its 1023 chips CDMA code (C/A code) send a signal that it is received shifted in time, meaning the time the signal took to arrive to the receiver [1]. The transmitting satellite is identified based on its code detection by circular correlation with all possible PRN codes. The circular correlation is usually carried out via frequency domain. Thus, the signal is acquired by the receiver that calculates its FFT and multiplies it by the conjugated FFT of the locally generated code following by the computation of the IFFT of the previous result. This effectively implements a circular correlation that yields a large peak, at the correct delay and Doppler shift, for the appropriate PRN code. A comparison of the highest peak power with the noise floor power is made, if it passes an established threshold, then the position of the spike indicates the synchronization delay, if that condition is not met, the correlation process is repeated for another data segment where the correlator outputs are compounded in an additive fashion. Eventually a different PRN code needs to be tested to found all satellites in view.

Implementations and Developments

We developed a basic correlator design (baseline) using MATLAB code and a Simulink model. This standard method is used to validate the results and performance yielded by new techniques when actual GPS satellite signal records.

We also developed our own MATLAB code for the MIT Quicksync algorithm [2] and implemented this algorithm in a Simulink model that, at the moment, works only with simulated data. The MATLAB implementation of the Quicksync algorithm [2] takes into consideration the Doppler Effect in order to be able to validate our implementation with real GPS signal records.

Another development is the formulation and implementation of alternative parallel architectures to perform a circular correlation by decomposing the initial circular correlation into several smaller circular correlations. Such sub-correlations are independent of each other and can be processed in parallel [3]. Even though these results may be applied to any system that performs circular convolution or circular correlation, we will apply it to the acquisition of Global Navigation Satellite System (GNSS) signals using a, FFT-based, Parallel Code-phase Search (PCS) on the GPS L1 C/A signal. The parallel approach may have advantages for hardware-based implementations using Field Programmable Gate Array.

We also developed a preliminary customized Simulink library [3]. A block library is a collection of blocks that serve as prototypes for instances of blocks in a Simulink® model. Since Simulink libraries do not get simulated the library will give us advantages in the models performance. The blocks in such libraries are blocked, which gives code protection so that it cannot be unknowingly altered. Another safety measure of a library is that all blocks are linked and have the same implementation. If further developments are required, there is no need to go to the models and change the block structure, we only have to change the blocks in the library model.

Finally we are in the process to develop a web page to increase the availability of the GPS signal records with our results featuring the signal delay and Doppler shift regarding the satellites present in the signal record. This type of data is scarce in the internet and, if found, the majority does not include any validation information in order to verify which satellites are present. In this site we also plan to include algorithm source code to assist in the discussion, study, experimentation and research on the topic.

Preliminary Development Platforms

Web Site

The web page of the project will be used as a dissemination tool. The web page offers a simple design that will allow others to see the project being developed, get information on the tools used and even download some of the code developed by the team members.

HTML Website Template

The team decided to obtain a free template from the Internet. The team came up with and excellent website offering free source code of html pages for download with a high quality and modern approach. The website template was downloaded from *All-free-download.com,* whose URL may be found at http://all-free-download.com/free-website-templates/. The team is thankful with the source code offered since it provided the project with very convenient management tools.

Website Layout

The website source code layout is as follows:

Directory	Description
archive	Html files placed here to handle the data request for the archive, this files would perform a finer documents organization listed by date, type of documents or newly added. The file obtains the source documents from the documents directory.
codeOutput	The directory is reserved to place the code's API developed during the project, this files are auto-generated by the Doxygen software so it was intuitively to add the html output for the general use of it.
css	The directory stores all the cascade sheet styles developed to make the web page look elegant and appealing. This directory was already part of the download package offered by All-free-download.com
documents	The directory stores the source documents to be displayed in the web site, pdf extension is highly recommended.
images	The directory stores all images and icons to be used on the website.
js	The directory stores the java scripts file to give the website a modern look and behavior. This directory was already part of the download package offered by All-free-download.com
php	
File	Description
contacts.html	The contacts page displays a contact form to allow direct communication between the users or visitors of the page and the web page administrator. A response php file is placed into the php directory to respond to the page processing. Notice that this specific implementation would depend on the web server being used and the policies established in by the server host.
documents.html	List all available documents currently being work by the team and that are currently in development. The page loads documents placed inside the documents directory. In addition the page offers direct local link to the archive directory where most of this documents are loaded but in a time stamp organization.
index.html	Load the home page of the web site, this is the page that is loaded when the site is accessed. The web server of preference would load
login.html	An optional login page script written in html but with easy connection to a php file with the same name inside the php directory. This file would perform a basic login control structure, however it is recommended to use the Apache .htaccess feature.
services.html	A description of the services offered by the Capstone group, ranging from a newly software library for GNSS applications to a group of data validation
software.html	Direct link for inspection or download of some of the code developed by the team during the time of work.

Table 1: Web site layout.

Web Server

In order to have a web page running on the Internet a web server is required, the design of the website have been created in such a way that any web server should be able to run the web site code. At the moment a prototype web site is being installed in one of the research assistant's computer and it is run on the Apache HTTP Server. Later on the web site will be available from an institutional server.

Apache is a web server application notable for playing a key role in the initial growth of the World Wide Web. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache quickly overtook NCSA HTTPd as the dominant HTTP server, and has remained the most popular HTTP server in use since April 1996. In 2009, it became the first web server software to serve more than 100 million websites. As of June 2013, Apache was estimated to serve 54.2% of all active websites and 53.3% of the top servers across all domain.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a Unix-like system,the software is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows, OS/2, TPF, OpenVMS and eComStation.

The Apache HTTP Server is running in a Linux machine, specifically in an Ubuntu 12.04 Server edition distro. The setup have been a transparent process and all modifications and editions on the web page are performed through an SSH connection.

Password Protection

Apache authentication can be configured to require web site visitors to login with a user id and password. This is different than adding a login form on a web page and creating your own authentication. This tutorial describes the various methods available for authentication with Apache and its' configuration. Login protection is applied to the web pages stored in a directory. The login dialog box which requests the user id and password is provided by the web browser at the request of Apache**.**

To get the system working the user should:

1. Place a file name .htaccess inside the directory of the website you want to protect and put on it the following code.

```
AuthName "Add your login message here."
AuthType Basic
AuthUserFile /home/domain/public_html/membersonly/.htpasswd
AuthGroupFile /dev/null
require user name-of-user
```

2. Create (or clobber if it already exists) the password file /home/domain/public_html/membersonly/.htpasswd using the program htpasswd:

```
htpasswd -c .htpasswd name-of-user.
```

3. Add a new user to the existing password file:

```
htpasswd .htpasswd name-of-user
```

4. Check that file look likes the following listing

```
user1:KgvCSeExtS4kM
USER1:KgvCSeExtS4kM
User1:KgvCSeExtS4kM
```

Git

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Many people's version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they're clever). This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.

In order to avoid such situation programmers long ago created a Version Control System(VCS), which in a way helped to avoid further problems, however VCS is also prone to errors because of the locality of the data, since it is placed in a local computer, and if the system fails your whole repository is in great danger.

The newest approach of VCS is called Distributed Version Control Systems (DVCS). In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every checkout is really a full backup of all the data

Git is a free and open source DVCS designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Keep track of code generation is a very difficult task when creating large amounts of code, for this reason we use Git, because it allows to keep track of all code recently changed and to return to it when desired without any problem. Git allowed us to avoid generation of version's code by implementing a separate file. Versions of the code are separated securely from one another in a single file by means of a Git snapshot, which offers a more organized version of coding.

Git with GUI

Even though it is extremely powerful the Git default software comes as a terminal interaction tool, many people just dislike this form of interaction and several projects have been created to provide Git with a GUI front end for easy interaction.

Even though there is not a standard in the process, since there are plenty of versions on which software should be used, a very common tool is gitk. The gitk package comes as a default in the Git software and it is used as a visualizer of the code generated in the platform.

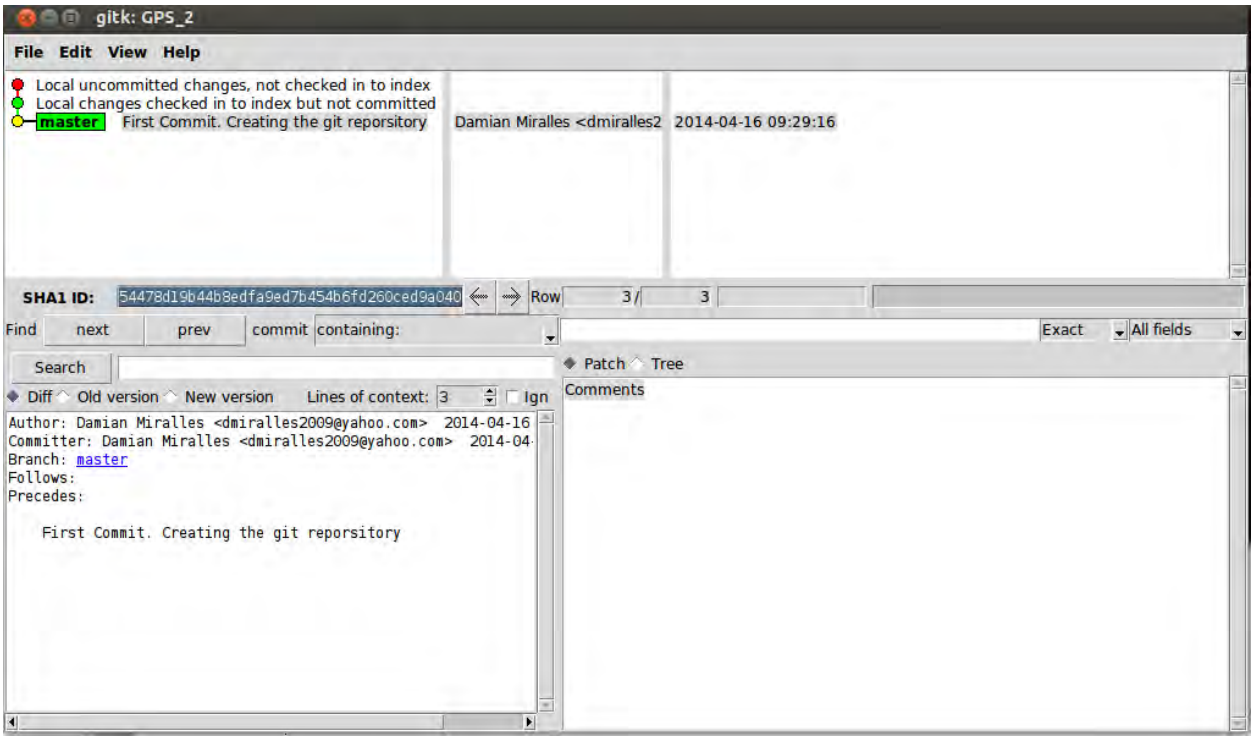

Figure 1: View of the gitk program on a Git repository

Project Management

Planner is a general purpose project management tool that provides a variety of features, which are available via 4 separate screen layouts called views. As you will see, the views are accessed by clicking the icons in the left hand toolbar in the Planner window.

Task View

Tasks are also shown on the Gantt view, but the Task view shows more detail for each task.

Task View is used for the following:

1. Task Definition - break down project deliverables into smaller, more manageable tasks
2. Task Sequencing - identify dependencies between tasks and other constraints via the task properties dialog, predecessors tab.
3. Task Duration Estimating - estimate the time it will take to complete the tasks
4. Task Cost Calculation - estimate what it will cost to complete the tasks

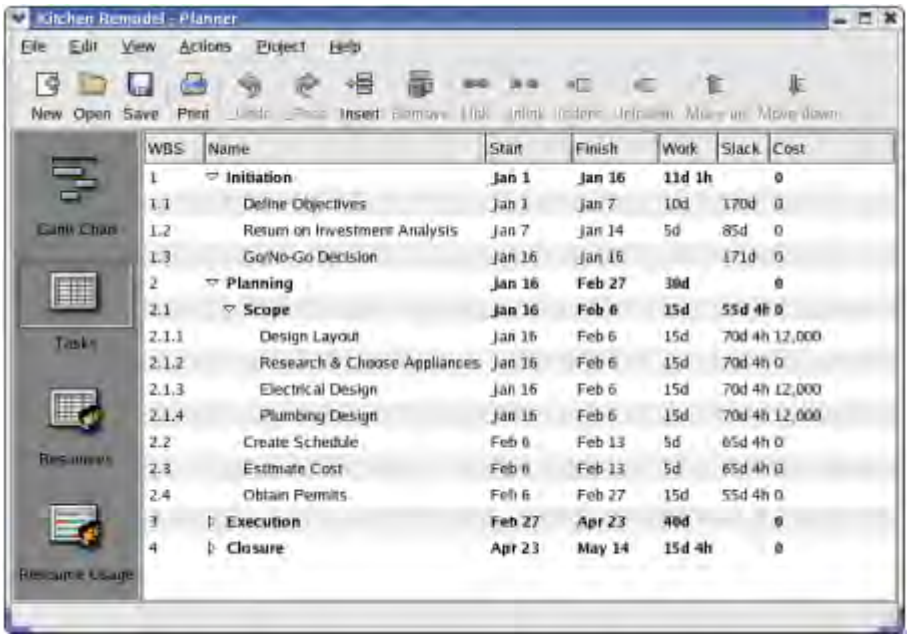

Figure 2: Task View window. Notice how task may be indented in a hierarchy model

Gantt View

The Gantt View combines the Gantt chart with a shortened version of the task view.

A Gantt chart is a graphical display of all the tasks that a project is composed of. Each bar on the chart is a graphical representation of the length of time the task is planned to take. The Gantt chart doesn't offer any features that are not available in the other views, but it is a valuable Project Management tool for the way it allows the user to see the project data.

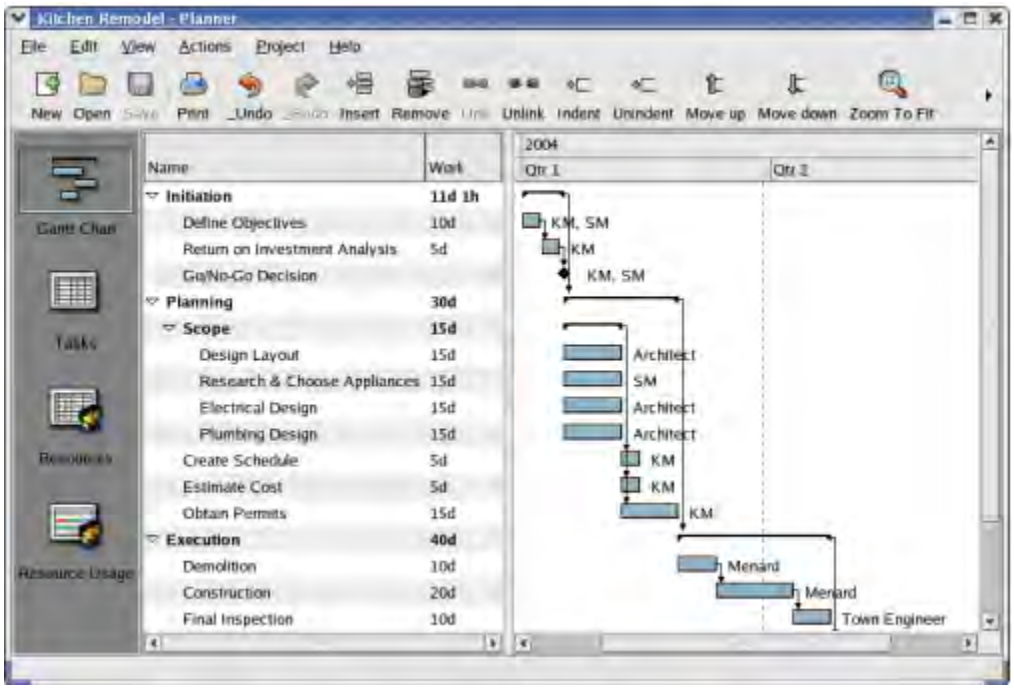

Figure 3: Gantt View window

The Gantt View allows you to:

1. View a graphical representation of the project schedule.
2. Manage task relationships using drag and drop.
3. Use Zoom in and Zoom out features to view different levels of detail.
4. View resources assigned to each task

You can change what columns are visible by choosing *View ▸ Edit Visible Columns*. This dialog will allow you to add, remove, and reorder the columns in the Gantt view. The same feature is available in the other views as well.

Resource View

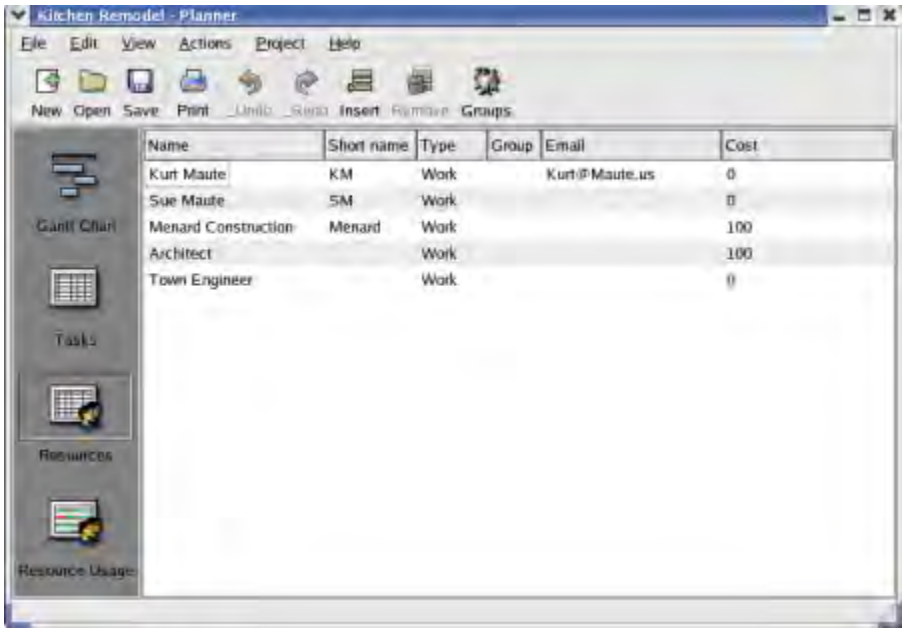

Figure 4: Resource View window

The Resource View gives you access to the following features:

1. Resource list management, including both human resources and materials.
2. Resource group management.
3. Resource cost management.

Resource Usage View

The Resource Usage View shows the availability of resources based upon the tasks they've been assigned to. They layout is similar to the Gantt chart, but this one is organized by resource.

The summary line shows the availability of the resource. The availability for each resource can be rolled up so that only the summary line is displayed by clicking on the triangle next to the resource name.

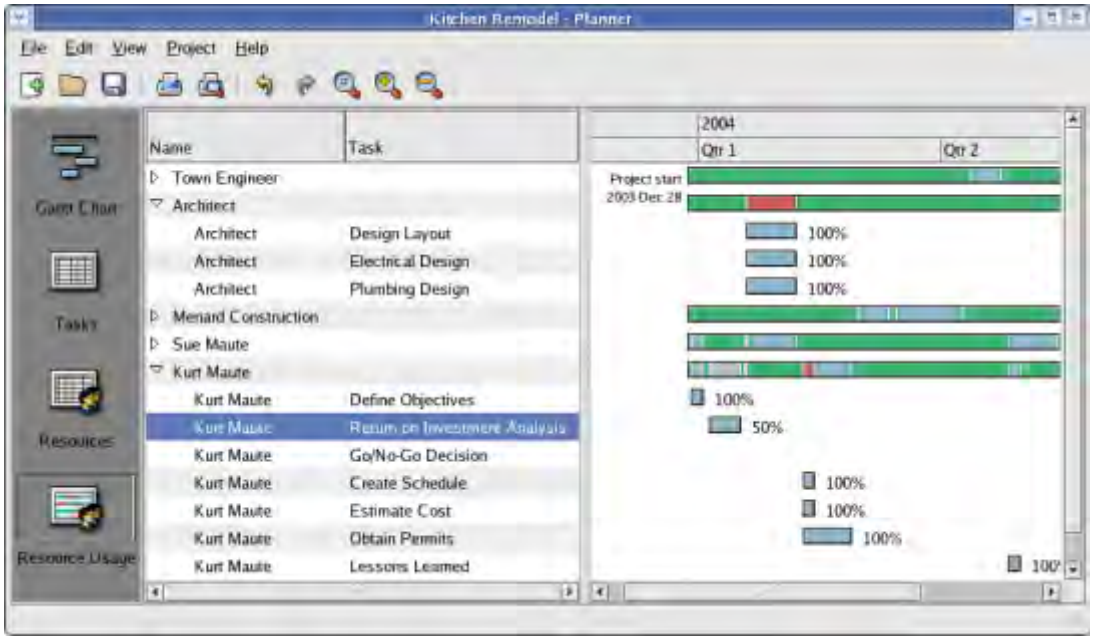

Figure 5: Resource Usage View window

The detail shows each task that the resource is assigned to and the time that the task is scheduled for is represented by a bar in the chart. Over-allocations are shown up as red in the summary bar, and it's easy to see where the task allocations line up to create the over-allocation.

Color coding is as follows:

1. Green shows that the resource is not allocated to any task at that time.
2. Blue has a slightly different meaning depending on its context. On the task line, it shows that the resource is either partially or fully allocated to the task (with the allocation percentage displayed next to it), but on the resource summary line, it shows that the resource is fully allocated at that time.
3. Grey shows that the resource is partially allocated at that time.
4. Red shows that the resource is over-allocated.

Doxygen

As an essential part of this capstone was the generation of code in the creation of the software libraries. Source code documentation has been always one of the topics more discussed in the coding community regarding its importance. Even though it has become a required part in recent years its development is still complicated. For that reason the team decided to use Doxygen as an essential tool to generate the desired documentation.

Doxygen is a documentation generator, a tool for writing software reference documentation. The documentation is written within code, and is thus relatively easy to keep up to date. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.

Doxygen has become the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP,

Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D. In addition plugins are available to add support in platform as MATLAB or Octave.

Doxygen, in addition can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in $\mbox{\LaTeX}$) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and UNIX man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.

You can configure Doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. Doxygen can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.

You can also use Doxygen for creating normal documentation as the team did for creating the documentation guidelines page and the Data Sets file. All output from this documentation format will be found in the project web page.

Doxygen is a very well documented platform, so additional information on installation, usage and configuration may be founded on their website. For this particular project a filter written in Perl is used to convert MATLAB code into C++ output to be processed by Doxygen. Documentation on how to reproduce this work will be found on the web page of the project under the Documentation guidelines section.

Doxywizard

Doxygen is a terminal alike program, so most of the interaction with the user happens in the terminal. This may be unappealing to several users; however Doxygen comes with a GUI that accomplishes the same result as those when interacting with the Doxygen shell. The GUI is called Doxywizard and is a GUI front-end used for configuring and running Doxygen.

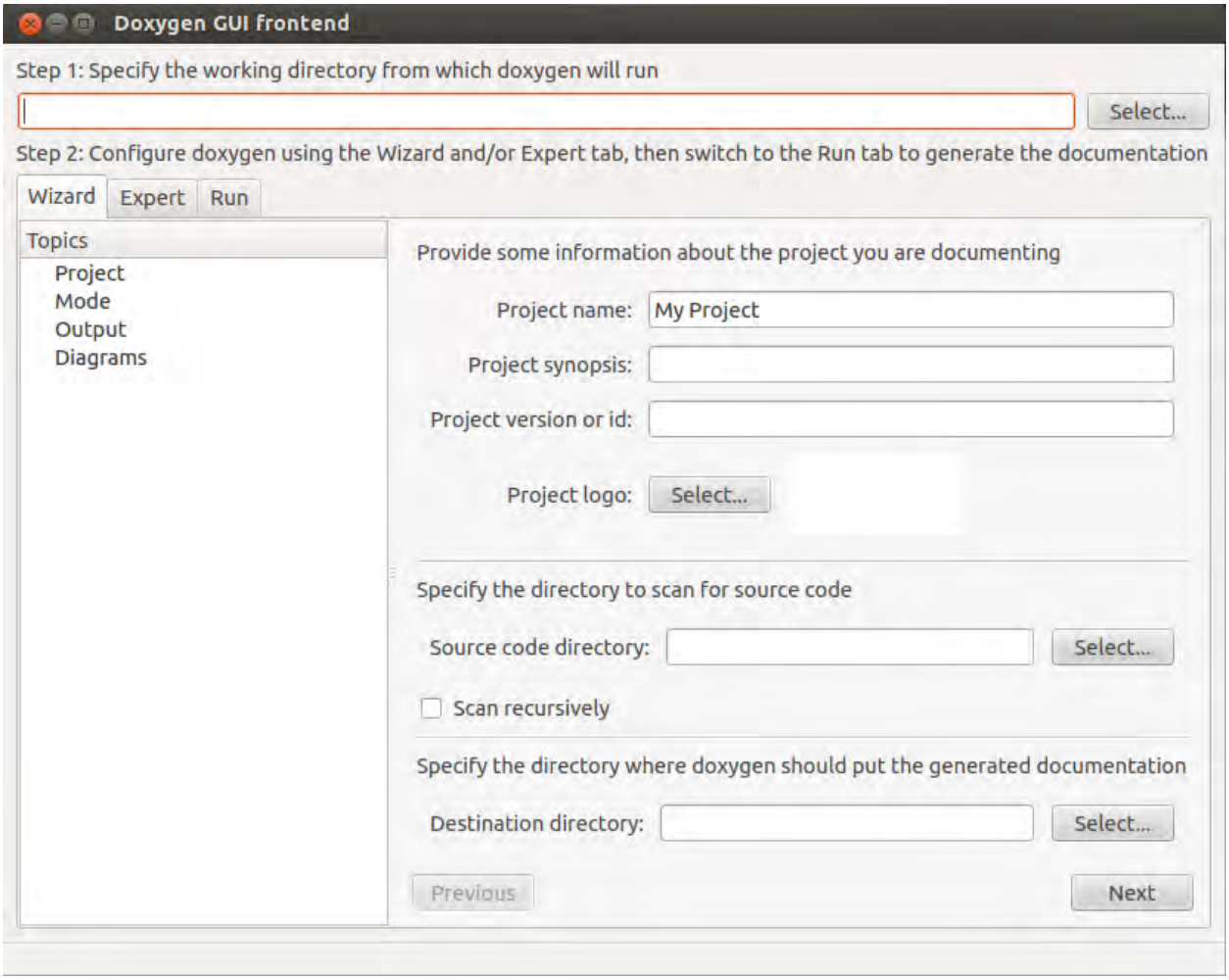

Figure 6: Doxywizard main window

When using the Doxywizard it is divided into 3 main tabs:

1. **Wizard:** Click this button to quickly configure the most important settings and leave the rest of the options to their defaults.
2. **Expert:** Click this button to gain access to the full range of configuration options.
3. **Run:** Click this tab to run the configuration options and obtain the Doxygen output.

After Doxygen is configured you need to save the configuration as a file to disk. This second step allows Doxygen to use the configuration and has the additional advantage that the configuration can be reused to run Doxygen with the same settings at a later point in time. Since some configuration options may use relative paths, the next step is to select a directory from which to run Doxygen. This is typically the root of the source tree and will most of the time already be filled in correctly. Once the configuration file is saved and the working directory is set, you can run Doxygen based on the selected settings. Do this by pressing the "Start" button. Once Doxygen runs you can cancel it by clicking the same button again. The output produced by Doxygen is captured and shown in a log window. Once Doxygen finishes, the log can be saved as a text file.

Microsoft Office Word

Microsoft Office Word is a word processor developed by Microsoft. It was first released in 1983 under the name Multi-Tool Word for Xenix systems. Subsequent versions were later written for several other platforms including IBM PCs running DOS (1983), Apple Macintosh running Mac OS (1985), AT&T Unix PC (1985), Atari ST (1988), SCO Unix (1994), OS/2 (1989), and Microsoft Windows (1989). Commercial versions of Word are licensed as a standalone product or as a component of Microsoft Office, Windows RT or the discontinued Microsoft Works suite. Freeware editions of Word are Microsoft Word Viewer and Office Online, both of which have limited features.

A full-featured word processing program for Windows and Mac OS X from Microsoft. Available stand-alone or as part of the Microsoft Office suite, Word contains rudimentary desktop publishing capabilities and is the most widely used word processing program on the market. Word files are commonly used as the format for sending text documents via e-mail because almost every user with a computer can read a Word document by using the Word application, a Word viewer or a word processor that imports the Word format (see Microsoft Word Viewer). Word 95 for Windows was the first 32-bit version of the product, released with Office 95 around the same time as Windows 95. It was a straightforward port of Word 6.0 and it introduced few new features, one of them being red-squiggle underlined spell-checking. Starting with Word 95, releases of Word were named after the year of its release, instead of its version number.

Microsoft Office Visio

Visio is a diagramming and vector graphics application and is part of the Microsoft Office suite. The product was first introduced in 1992, made by the Shapeware Corporation. It was acquired by Microsoft in 2000. The software is of great utility when creating flowcharts or diagrams for the software being developed in the project.

Microsoft Visio 2010 for Windows is available in three editions: Standard, Professional and Premium. The Standard and Professional editions share the same interface, but the latter has additional templates for more advanced diagrams and layouts, as well as unique capabilities intended to make it easy for users to connect their diagrams to data sources and display their data graphically. The Premium edition features three additional diagram types as well as intelligent rules, validation, and subprocess (diagram breakdown). At the moment of this document the team was using the student edition of the software available free of charge with a university Dream Spark account.

Project Time Table

This GPS project has a deadline that requires its completion in less than six months. Below is a Gantt chart that will show how the tasks where divided by the team members and the time expected to be invested in each and one of those tasks.

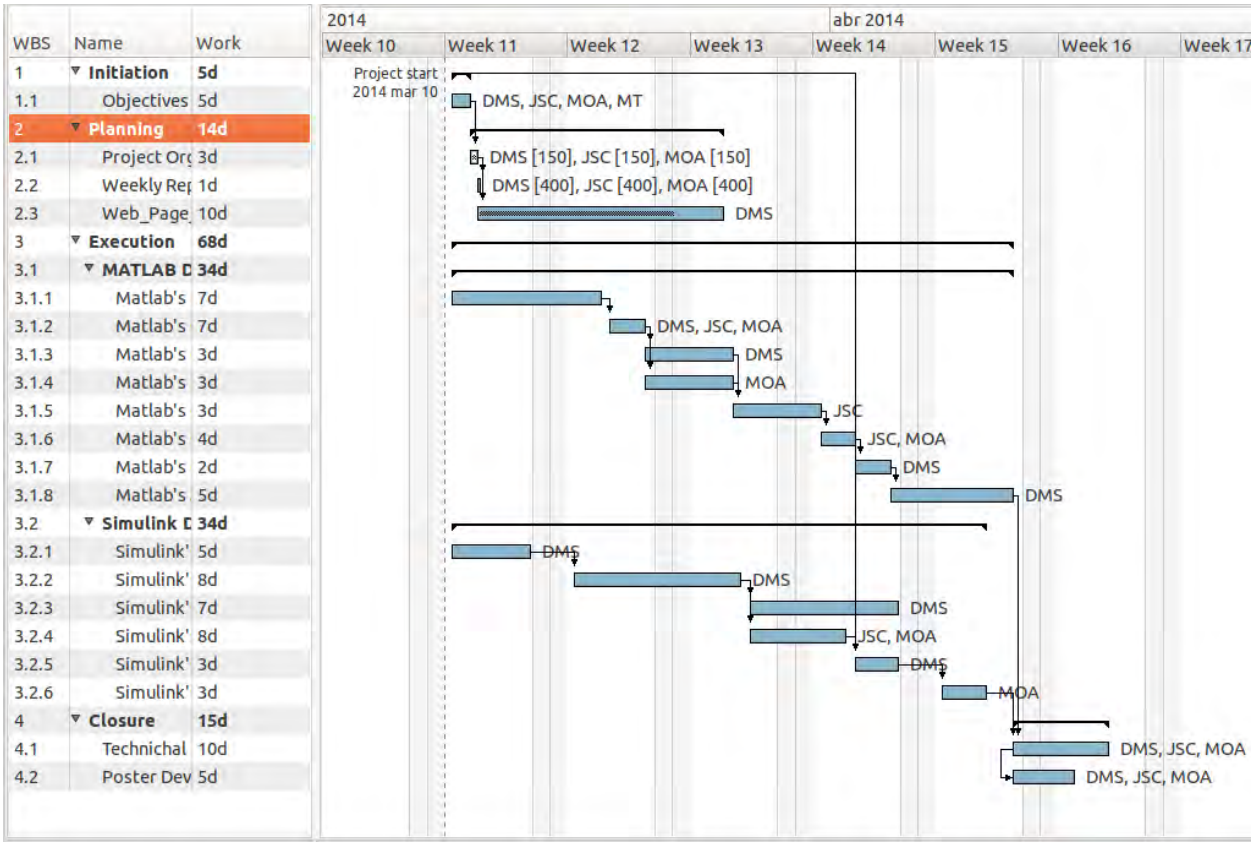

Figure 7: Gantt chart of the design's development progress.

Preliminary Results

MATLAB

For this project segment we developed five different MATLAB scripts and a set of functions. In these scripts we developed the standard baseline method for the code phase detection with modeled data and real data with different frequency resolutions regarding the Doppler Effect. We did the same for the MIT Quicksync algorithm. We developed a script for modeled data and another script for field recorded satellite signal data. The functions and scripts that were developed are for data modeling, correlation of the PRN codes with the modeled data and correlation of the PRN codes with the real recorded satellite data.

Simulink

In Simulink we managed to develop two models for the baseline method of code phase detection, one for the modeled data and the other one for recorded data. Another model was implemented for the MIT Quicksync algorithm that works with modeled data. We also implemented four models for the parallel architectures that will be discussed in the next section and are more extensively described in [3].

Parallel Architectures

We managed to implement a total of four parallel architectures. In general terms the proposed architectures were compared for the number of adders, multipliers and FFTs required by each construct in order to parallelize the correlation process. Details are provided in a separate technical report [3].

	First Architecture	First Modified Architecture	Second Architecture	Second Modified Architecture
Radix	2	2	2	2
FFTs	6	7	6	7
Multipliers	5	4	4	3
Adders	2	2	5	5
NCOs	1	0	1	0

Table 2: Comparison of the parallel design's architectures developed.

Radix-2 architecture should be almost twice as fast as the direct serial architecture if implemented in hardware. The tradeoff is an extended use of hardware resources. For details see [3].

Data Sets

The data sets where created using two signal records from the book *A Software-Defined GPS and Galileo Receiver* the files: *GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin* (collected in Turin, Italy) and *GPSdata-DiscreteComponents-fs38_192-if9_55.bin* (collected at the University of Colorado, Boulder, CO, USA). From the last file we only know that the PRN of the satellite 21 is present, from the other, the book doesn't indicate the presence of any specific PRN of a satellite.

On the other hand the other two signals record where acquired from the GPS Danish Center old web site: *http://kom.aau.dk/project/softgps/data.php*. There they had three files: *compactdata_20050407_142600.bin*, *Multipath.bin*, and *Multipath_short.bin*. The last two, even if they have different names, because of the results, they appear to be the same signal record.

How to Read the Files

To read the signal records in MATLAB the following instructions are required:

```
[fid, message] = fopen('compactdata_20050407_142600.bin'), 'r', 'b');
data = fread(fid,maxallowedbymemory, 'ubit1')';
```

For the file *compactdata_20050407_142600.bin* the mode to read the data is the type *ubit1*, as instructed in the web site, for the rest of the signal records, both from the book and the web site, the data type used is *bit8* and could be implemented the following way:

```
%file='Multipath';fs=16367600;fi= 4130400;hr='bit8';
%file='Multipath_short';fs=16367600;fi= 4130400;hr='bit8';
%file='compactdata_20050407_142600';fs=12000000;fi= 3563000; hr='ubit1';
%file='GPS_and_GIOVE_A-NN-fs16_3676-if4_1304';fs=16367600;fi= 4130400;hr='bit8';
file='GPSdata-DiscreteComponents-fs38_192-if9_55';fs=38192000;fi= 9550000;hr='bit8';


[fid, message] = fopen(strcat(file,'.bin'), 'r', 'b');
data = fread(fid,maxallowedbymemory, hr)';
```

Results

From the file *GPSdata-DiscreteComponents-fs38_192-if9_55.bin* from the book: *A Software-Defined GPS and Galileo Receiver* states that satellite with PRN 21 is present (as shown in Figure 7 and Figure 8). Since our implementation detects this satellite it validates the code we have developed. Also the book indicates that the file already mentioned does not include the PRN 19, and our code does not detect it either. Other satellites detected using further signal records are provided in the tables ahead. These results validate well with the information provided in the original data source.

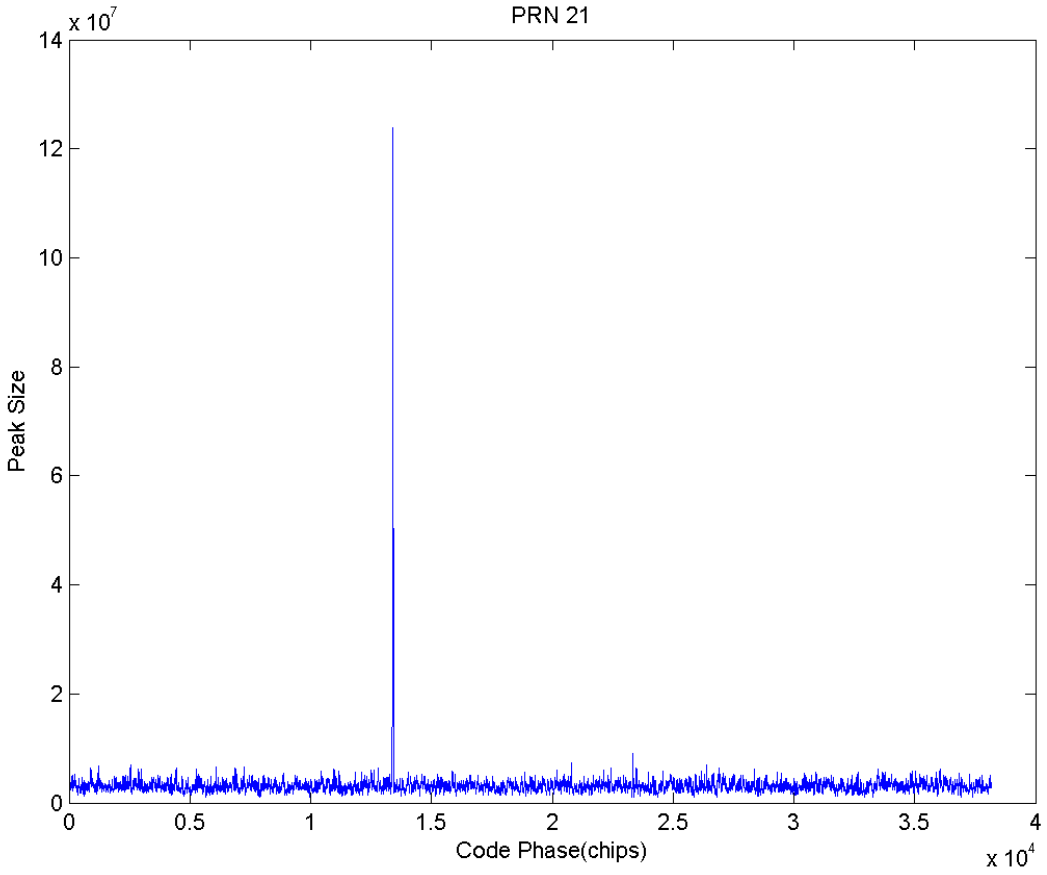

Figure 8: Peak detection of the PRN 21 in file *GPSdata-DiscreteComponents-fs38_192-if9_55.bin*.

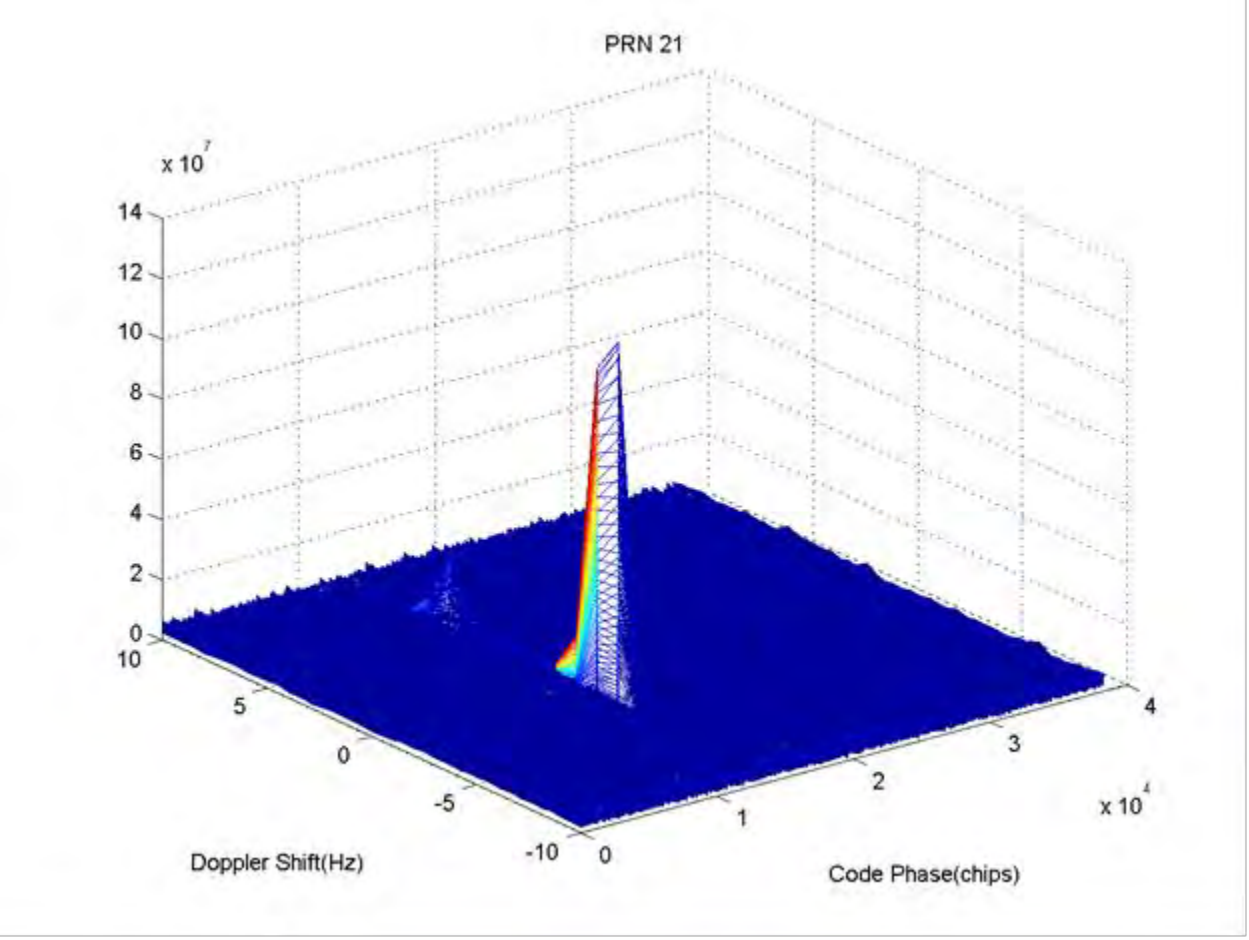

Figure 9: Acquisition plot for PRN 21 in file *GPSdata-DiscreteComponents-fs38_192-if9_55.bin*.

compactdata_20050407_142600.bin

PRN	Doppler Shift	Code Phase
1	3565000	1906
3	3559000	7982
7	3566000	8082
14	3562000	10555
19	3560000	10403
20	3566000	10216
22	3560000	3859
24	3567000	4558
28	3562000	10168
31	3565000	808

Table 3: Results of the compactdata_20050407_142600.bin file.

Multipath.bin and Multipath_short.bin

PRN	Doppler Shift	Code Phase
2	4126400	1798
6	4129400	9601
7	4129400	2445
10	4131400	5337
24	4128400	12978
30	4126400	9858

Table 4: Results of the Multipath.bin and Multipath_short.bin files.

GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin

PRN	Doppler Shift	Code Phase
3	4134400	7367
15	4132400	1496
16	4131400	2072
18	4132400	1532
19	4136400	6343
22	4134400	14077

Table 5: Results of the GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin file.

GPSdata-DiscreteComponents-fs38_192-if9_55.bin

PRN	Doppler Shift	Code Phase
3	9550000	34213
6	9544000	28203
9	9551000	4695
15	9550000	36321
18	9548000	20725
21	9547000	13404
22	9550000	6289
26	9545000	26828

Table 6: Results of the GPSdata-DiscreteComponents-fs38_192-if9_55.bin file.

References

[1] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. Holdt Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach. 2012, p. 175.

[2] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk, "Faster GPS via the sparse fourier transform," Proc. 18th Annu. Int. Conf. Mob. Comput. Netw. - Mobicom '12, 2012.

[3] D. Miralles, M. Ortiz, J. Sandoval and M. Teixeira. Development of a Simulink Library for the Design, Testing and Simulation of Software Defined GPS Radios. With Applications to the Development of Parallel Correlator Structures. Internal Technical Report, Polytechnic University of Puerto Rico. (Uploaded to DoD Web Site)

Bibliography

F. J. Coelho, "Software Defined GPS / Galileo Receiver Master of Science in Electrotechnical Engineering," no. April, 2011.

C. Jeffrey, An Introduction to GNSS. 2010, p. 96.

F. Principe, G. Bacci, F. Giannetti, and M. Luise, "Software-Defined Radio Technologies for GNSS Receivers: A Tutorial Approach to a Simple Design and Implementation," Int. J. Navig. Obs., vol. 2011, no. iii, pp. 1–27, 2011.

New Wave Instruments, "Linear Feedback Shift Registers." [Online]. Available:

http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm. [Accessed: 09-Jan-2014].

K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. Holdt Jensen, A Software-Defined GPS and Galileo Receiver. A Single-Frequency Approach. 2012, p. 175.

H. Hassanieh, F. Adib, D. Katabi, and P. Indyk, "Faster GPS via the sparse fourier transform," Proc. 18th Annu. Int. Conf. Mob. Comput. Netw. - Mobicom '12, p. 353, 2012.

Mathworks Documentation Center(2013),

J. Leclère, C. Botteron, and P. Farine, "Improving the Performance of the FFT-based Parallel Code-phase Search Acquisition of GNSS Signals by Decomposition of the Circular Correlation," in ION GNSS 2012, 2012, pp. 1–11.

M. Teixeira and D. Rodriguez, "A new method mathematically links fast Fourier transform algorithms with fast cyclic convolution algorithms," in *Proc. of the IEEE 37th Midwest Symp. on Circuit and Syst.*, Lafayette, LA, Aug. 1994.

M. Teixeira and D. Rodriguez, "A class of fast cyclic convolution algorithms based on block pseudocirculants*," IEEE Signal Processing Letters,* vol. 2, No. 5, pp. 92-94, May 1995.

[Word] Wikipedia, the free encyclopedia (2014, April 15). Microsoft Word [Online]. Available: http://en.wikipedia.org/wiki/Microsoft_Word

[Visio] Wikipedia, the free encyclopedia (2014, April 18). Microsoft Visio [Online]. Available: http://en.wikipedia.org/wiki/Microsoft_Visio

[apache] Wikipedia, the free encyclopedia (2014, April 11).Apache HTTP Server [Online]. Available: http://en.wikipedia.org/wiki/Apache_HTTP_Server

[Doxygen] Wikipedia, the free encyclopedia (2014, March 6). Doxygen [Online]. Available: http://en.wikipedia.org/wiki/Doxygen

[Doxygen 2] D. van Heesch (2014, April 21). Doxygen [Online]. Available: http://www.stack.nl/~dimitri/doxygen/

[git] S. Chacon, "Getting Started," in Pro Git, New York,NY: Apress, 2009, pp. 1-12

Appendix

Preliminary Source Code (Under Development)

Implementation of the MIT Quicksynch Algorithm [2] (MATLAB Function)

```
% ========================================================================
%> @brief MIT QuickSync algorithm implementation with Doppler correction and 1000 Hz of accuracy.
%> @details
%> Doppler correction performed in time by means of a complex
%> exponential multiplication.
%> Loop structure for detection is doppler->sat->buckets.
%> Support:  mteixeir@ieee.org
%> Support:  dmiralles2009@yahoo.com
%>    -# GPS PRN Correlator Detector (parallel code phase search)
%>    -# Doppler search step: 1000 Hz.
%>    -# Uses data 1ms in length (corresponds to 1 PRN) sampled at Fs
%> @date      May 15, 2014
%> @pre       Former File Name: sparse_mit_sat_out.m
%> @bug       Code Phase delay may be differ by 1 sampling period.
%> @note      This file represents the function version of the file name sparse_mit_sat_out.m
%> @author  Damian Miralles, Marvi Teixeira, Jennifer Sandoval, Manuel Ortiz
%> @param dataset Selects the data set to be read for data. For more information on the data sets
available refer to \ref rawFiles.
%> @param fs Sampling frequency
%> @param fi Intermediate frequency
%> @param satellites List of prn satellites numbers to be detected. Values should range from 1 to
32.
%> @param thresholdsparse Value to determine when a detection is valid
%>
%> @retval whole_set List of synchronization values to be returned.
% ========================================================================
function whole_set = sparse_mit_dpl_sat_time(dataset,fs,fi,satellites,thresholdsparse)


%Search for the number of arguments the user puts when calling the function
switch nargin
    case 1
        error('Please enter sampling frequency and intermediate frequency')
    case 2
        error('Intermediate frequency value is compulsory')
    case 3
        satellites = [1:32];
    case 4
        if (max(satellites)>37) || (min(satellites)<1) || (min(satsize(satellites))~=1)
            error('sv must be a row or column vector with integers between 1 and 37\n')
        end
        thresholdsparse = 9;
    case 5
        %do nothing. User entered all values into the function
    otherwise
        error('Wrong number of input values')
end

ts=1/fs;


%=========================================================================
% RAW DATA 1 Settings (I y Q) (Present 01 03 07 19 20 22 24 28 31)
% satellites=[01 03 07 19 20 22 24 28 31];
% fs=12000000;
% ts=1/fs;
% fi=3563000;
%=========================================================================
```

```
%==========================================================================
% %RAW DATA 2 Settings  (only I component, requires code modification)
% satellites=[01 21 29 30 31];
% fs=5456000;
% ts=1/fs;
% fi=4092000;
%==========================================================================

%==========================================================================
%RAW DATA 3 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
%(detected: ?3, 6,  21, 22, 26, 29,)
% satellites=[3 4 6 9 15 18 21 22 26];
% fs=38192000;
% ts=1/fs;
% fi=9548000;
%==========================================================================

%==========================================================================
%RAW DATA 4 Settings (I and Q) Akos-Book (2nd data set)
% satellites=[01 02 03 04 05 06 07 17 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: 19,...)
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%==========================================================================

%==========================================================================
%RAW DATA 5
% satellites=[22 3 19 14 18 11 32 6]; %(sorted from strongest to weakest
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%==========================================================================


g=cacode(satellites, fs/1023000); %Creates the matrix of prn codes
ntemp=length(g(1,:));    %Obtain pulse size based on the output of cacode
%function. Note that the size is stored in a temporary variable. This is
%made in order to allow the script to work with any sampling frequency.

p= round(sqrt(log(ntemp)/log(2)));   %define p according to the formula but with rounding its
value instead of flooring it.
B = round(ntemp/p);                   %define the bucket downsampling unit as n/p according to the
papers annotations
ceil_floor = 0;                      %determine operation 0 = floor, 1 =  ceil
if(B*p == ntemp)
    n=ntemp;                    %if B*p is equal to the size of the pulse, then n = ntemp
else
    if(B*p > ntemp)                     %check if the new value os different than the old value of
n, if yes, then n changes
        n= B*p;                          %define the signal new size, ensure that it always keep a
length of B*p
        ceil_floor = 1;
    else                                 %else (B*p < ntemp) , this is a floor
        n= B*p;                          %define the signal new size, ensure that it always keep a
length of B*p
        ceil_floor = 0;
    end
end
k = 5;                           %number of buckets to be used
maxallowedbymemory=n*p*k;        %data read from file

% SELECT DATA FILE TO READ
switch dataset
    case 1
        % Data Set 1: compact.bin should be compactdataq_etc_etc...
        % [fid, message] = fopen('compact.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'ubit1')';
        % fclose(fid);
        [fid, message] = fopen('compact.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'ubit1')';
        fclose(fid);
```

```
   case 2
       % Data 2, with I component only
        %[fid, message] = fopen('gps.samples.1bit.I.fs5456.if4092.bin', 'r', 'b');
        %
        %
      statements
    case 3
        % Data 3, from Borre-Akos book
        % [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 4
        % % Data 4, from Borre-Akos book (2nd data set)
        % [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 5
        % % Data 5,
        % [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    otherwise
       error('Wrong data set number');
end

nn=0:length(data)-1;
ficarrierQ=((cos(2*pi*fi*nn*ts)));
ficarrierI=((sin(2*pi*fi*nn*ts)));

% ******* I and Q using XOR of 1s and 0s *******
ficarrierI2=ficarrierI>0; % positives go to 1 negatives go to 0
ficarrierQ2=ficarrierQ>0;  % positives go to 1 negatives go to 0
data=data>0;
datafiI=xor(data,ficarrierI2); % xor data  with sin to obtain I
datafiQ=xor(data,ficarrierQ2); % xor data  with cos to obtain Q
wholesignal=(2*datafiI-1)+1*1i*(2*datafiQ-1);


% %******** I and Q using . (multiplication of 1s and -1s)********
% data = data * 2 - 1; %To convert values to +/- 1
% ficarrierI2=2*ceil(ficarrierI)-1; %
% ficarrierQ2=2*ceil(ficarrierQ)-1; %
% datafiI=(data.*ficarrierI2);
% datafiQ=(data.*ficarrierQ2);
% wholesignal=(datafiI)+1*j*(datafiQ);

%Variable preallocation. As recommende by MATLAB, variables that change
%size in a loop should be first declared. This increase execution speed.

cornp2 = (1:p);     %store result of final correlation between the doppler
% corrected signal and the pulse.
whole_set(length(satellites),6) = 0; %variable holding important values
%after threshold is detected. The matrix stores variables in the following
%order per PRN: PRN-SNR-Doppler Shift- Deviation from IF- Code Phase-
%number of buckets.


bucketnp(n*p) = 0;      %var holding the data in chunks of size p*n.
bucketnpexp(k,n*p) = 0;     % var holding the data with doppler correction by means of complex
exponential multiplication.
fftbucketfolded(k,B) = 0;    %var holding the aliased bucket (B samples apart are sum up
together)
```

```
xs(B) = 0;      %acumulation bucket results
xss(B) = 0;     %acumulation bucket results without peak signal
possibledelay(p) = 0;   %vector holding the p possible delays.

delays(length(satellites),p) = 0; %variable holding the p possible delay per
% satellites detected.
outputfoldedbuckets(B)=0;           %variable storing the output of the ifft


pulses = g;  %obtain prn pulses code given the PRN number
if(ntemp ~= n)
    if(ceil_floor == 0)
        pulses = pulses(:,1:n);
    else
        pulses(:,n) = 0;
    end
end
pulses = pulses*2-1;          %change 0 by -1.
conjfftpulsesfolded(length(satellites),B) = 0;

%Perform pulse manipulation outside main loop. Here memory usage is
%increased but execution time is reduced.
for prncounter=1:1:length(satellites)
    pulsesfolded = sum(reshape(pulses(prncounter,:),B,p).');
    fftpulsesfolded = fft(pulsesfolded);
    conjfftpulsesfolded(prncounter,:) = conj(fftpulsesfolded);
end

tic     %start counting execution time
range = 0:n*p-1; %range for exponential multiplication

for frequencydopplershift=-10*p:1:10*p

    for ii=1:k
        exponential = exp(1i*(2*pi/(n*p))*frequencydopplershift.*range);
        bucketnp=wholesignal((ii-1)*p*n+1:ii*n*p);
        if(ntemp ~= n)
            if(ceil_floor == 0)
                bucketnp = bucketnp(1:n*p);
                length(bucketnp);
            else
                bucketnp(n*p) = 0;
                length(bucketnp);
            end
        end
        bucketnpexp(ii,:) =  bucketnp.*exponential;
        bucketfolded= sum(reshape(bucketnpexp(ii,:),B,p^2).');
        fftbucketfolded(ii,:)=fft(bucketfolded);     % FFT of folded length-p*n bucket
    end
    stop = 0;

    for prncounter=1:1:length(satellites)

        xs = zeros(1,B);    %clean accumulation buckets variable before going to other satellite
        xss= zeros(1,B);    %clean accumulation buckets variable before going to other satellite
        SNRmethod4=0; % SNR method 4 init
        stop = 0;           %stop detection when for a given doppler threshold value is exceeded

         for ii=1:k
            %IFFT of the pointwise product of fft of folded pulse and fft of folded
            %bucket
            outputfoldedbuckets=ifft(conjfftpulsesfolded(prncounter,:).*fftbucketfolded(ii,:));

            %Find the magnitude of the reult added to the accumulated
            %buckets
            xs=(abs(outputfoldedbuckets).^2)+xs;% accumulated output buckets to enhance detection
peak


            % Looking for the sample with the largest spike in the accumulated output bucket
```

```
            % The delay in the bucket is the largest sample positon.
            % For example, if it is a delay of n/p+x or 2*n/p+x this method gives x as the delay
in the folded bucket.
            % Therefore we have to find a way to settle the ambiguity (see below)

            [sizepeak,delayinfoldedbucket]=max(xs); %delay estimation based on the accumulation
of several output buckets
            xss = xs; % auxiliary variable to mantain the accumulator untouched
            %SIGNAL TO NOISE RATIO  (signal: peak power    noise: noise floor power)
            xss(delayinfoldedbucket)=0; %take peak out of accumulated output folded bucket
            noisefloorpwr=mean(xss);%power of noise bed in output folded bucket
            SNRmethod2=(sizepeak/noisefloorpwr);


            if stop == 0 && SNRmethod2>thresholdsparse  % threshold comparison
               buckets_used_method2=ii;                  %amount of data of size n*p used for
detection
               detected_doppler_shift = -(frequencydopplershift) ;%Doppler shift calculated based
on doppler from break
               deviation_from_IF=fi+detected_doppler_shift*(1000)/p;
               detected_code_phase_folded=delayinfoldedbucket;
               stop = 1;

               for jj=1:p
                   possibledelay(jj)= detected_code_phase_folded+(jj-1)*n/p;
               end

               for jj=1:p

cornp2(jj)=sum(pulses(prncounter,:).*bucketnpexp(ii,(possibledelay(jj):possibledelay(jj)+n-1)));

               end
               [size,delaychoice]=max(abs(cornp2));
               detected_code_phase=detected_code_phase_folded+(delaychoice-1)*n/p;
               if SNRmethod2 > whole_set(prncounter,2)
                    delays(prncounter,:) = [possibledelay];
                    whole_set(prncounter,:) =
[satellites(prncounter),round(SNRmethod2),deviation_from_IF,detected_doppler_shift,detected_code_
phase,ii];
               end

            end
        end
    end
end

executiontime = toc
```

Implementation of the MIT Quicksynch Algorithm [2] (MATLAB Script)

```
% ======================================================================
%> @brief QuickSync algorithm implementation with Doppler correction and 1000 Hz of accuracy.
%> @details
%> Doppler correction performed in time by means of a complex
%> exponential multiplication. Loop structure for detection is doppler->sat->buckets.
%> Support: mteixeir@ieee.org
%> Support: dmiralles2009@yahoo.com
%>    -# GPS PRN Correlator Detector (parallel code phase search)
%>    -# Doppler search step: 1000 Hz.
%>    -# Uses data 1ms in length (corresponds to 1 PRN) sampled at Fs
%> @date      May 15, 2014
%> @pre       Former File Name: sparse_mit_sat_out.m
%> @bug       Code Phase delay may be differ by 1 sampling period.
%> @note      This file represents the function version of the file name sparse_mit_sat_out.m
%> @author    Damian Miralles, Marvi Teixeira, Manuel Ortiz, Jennifer Sandoval
%> @param dataset Selects the data set to be read for data. For more information on the data sets
available refer to \ref rawFiles.
%> @param fs Sampling frequency
%> @param fi Intermediate frequency
%> @param satellites List of prn satellites numbers to be detected. Values should range from 1 to
32.
%> @param thresholdsparse Value to determine when a detection is valid
%>
%> @retval whole_set List of syncronization values to be returned.
% ======================================================================
function whole_set = sparse_mit_dpl_sat_time(dataset,fs,fi,satellites,thresholdsparse)


%Search for the number of arguments the user puts when calling the function
switch nargin
    case 1
        error('Please enter sampling frequency and intermediate frequency')
    case 2
        error('Intermediate frequency value is compulsory')
    case 3
        satellites = [1:32];
    case 4
        if (max(satellites)>37) || (min(satellites)<1) || (min(satsize(satellites))~=1)
            error('sv must be a row or column vector with integers between 1 and 37\n')
        end
        thresholdsparse = 9;
    case 5
        %do nothing. User entered all values into the function
    otherwise
        error('Wrong number of input values')
end

ts=1/fs;


%======================================================================
% RAW DATA 1 Settings (I and Q) (Present 01 03 07 19 20 22 24 28 31)
% satellites=[01 03 07 19 20 22 24 28 31];
% fs=12000000;
% ts=1/fs;
% fi=3563000;
%======================================================================

%======================================================================
% %RAW DATA 2 Settings  (only I component, requires code modification)
% satellites=[01 21 29 30 31];
% fs=5456000;
% ts=1/fs;
% fi=4092000;
%======================================================================

%======================================================================
```

```
%RAW DATA 3 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
%(detected: ?3, 6,  21, 22, 26, 29,)
% satellites=[3 4 6 9 15 18 21 22 26];
% fs=38192000;
% ts=1/fs;
% fi=9548000;
%=========================================================================

%=========================================================================
%RAW DATA 4 Settings (I and Q) Akos-Book ( 2nd data set)
% satellites=[01 02 03 04 05 06 07 17 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: 19,...)
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%=========================================================================

%=========================================================================
%RAW DATA 5
% satellites=[22 3 19 14 18 11 32 6]; %(sorted from strongest to weakest
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%=========================================================================


g=cacode(satellites, fs/1023000); %Creates the matrix of prn codes
ntemp=length(g(1,:));    %Obtain pulse size based on the output of cacode
%function. Note that the size is stored in a temporary variable. This is
%made in order to allow the script to work with any sampling frequency.

p= round(sqrt(log(ntemp)/log(2)));   %define p according to the formula but with rounding its
value instead of flooring it.
B = round(ntemp/p);                   %define the bucket downsampling unit as n/p according to the
papers annotations
ceil_floor = 0;                      %determine operation 0 = floor, 1 =  ceil
if(B*p == ntemp)
    n=ntemp;                    %if B*p is equal to the size of the pulse, then n = ntemp
else
    if(B*p > ntemp)                     %check if the new value os different than the old value of
n, if yes, then n changes
        n= B*p;                          %define the signal new size, ensure that it always keep a
length of B*p
        ceil_floor = 1;
    else                                 %else (B*p < ntemp) , this is a floor
        n= B*p;                          %define the signal new size, ensure that it always keep a
length of B*p
        ceil_floor = 0;
    end
end
k = 5;                           %number of buckets to be used
maxallowedbymemory=n*p*k;        %data read from file

% SELECT DATA FILE TO READ
switch dataset
    case 1
        % Data Set 1: compact.bin should be compactdataq_etc_etc...
        % [fid, message] = fopen('compact.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'ubit1')';
        % fclose(fid);
        [fid, message] = fopen('compact.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'ubit1')';
        fclose(fid);
   case 2
       % Data 2, with I component only
        %[fid, message] = fopen('gps.samples.1bit.I.fs5456.if4092.bin', 'r', 'b');
        %
        %
      statements
    case 3
        % Data 3, from Borre-Akos book
        % [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
```

```
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 4
        % % Data 4, from Borre-Akos book (2nd data set)
        % [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 5
        % % Data 5,
        % [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    otherwise
       error('Wrong data set number');
end

nn=0:length(data)-1;
ficarrierQ=((cos(2*pi*fi*nn*ts)));
ficarrierI=((sin(2*pi*fi*nn*ts)));

% ******* I and Q using XOR of 1s and 0s *******
ficarrierI2=ficarrierI>0; % positives go to 1 negatives go to 0
ficarrierQ2=ficarrierQ>0;  % positives go to 1 negatives go to 0
data=data>0;
datafiI=xor(data,ficarrierI2); % xor data  with sin to obtain I
datafiQ=xor(data,ficarrierQ2); % xor data  with cos to obtain Q
wholesignal=(2*datafiI-1)+1*1i*(2*datafiQ-1);


% %******** I and Q using . (multiplication of 1s and -1s)********
% data = data * 2 - 1; %To convert values to +/- 1
% ficarrierI2=2*ceil(ficarrierI)-1; %
% ficarrierQ2=2*ceil(ficarrierQ)-1; %
% datafiI=(data.*ficarrierI2);
% datafiQ=(data.*ficarrierQ2);
% wholesignal=(datafiI)+1*j*(datafiQ);

%Variable preallocation. As recommende by MATLAB, variables that change
%size in a loop should be first declared. This increase execution speed.

cornp2 = (1:p);     %store result of final correlation between the doppler
% corrected signal and the pulse.
whole_set(length(satellites),6) = 0; %variable holding important values
%after threshold is detected. The matrix stores variables in the following
%order per PRN: PRN-SNR-Doppler Shift- Deviation from IF- Code Phase-
%number of buckets.


bucketnp(n*p) = 0;      %var holding the data in chunks of size p*n.
bucketnpexp(k,n*p) = 0;     % var holding the data with doppler correction by means of complex
exponential multiplication.
fftbucketfolded(k,B) = 0;    %var holding the aliased bucket (B samples apart are sum up
together)

xs(B) = 0;      %acumulation bucket results
xss(B) = 0;     %acumulation bucket results without peak signal
possibledelay(p) = 0;   %vector holding the p possible delays.

delays(length(satellites),p) = 0; %variable holding the p possible delay per
% satellites detected.
outputfoldedbuckets(B)=0;           %variable storing the output of the ifft
```

```
pulses = g;  %obtain prn pulses code given the PRN number
if(ntemp ~= n)
    if(ceil_floor == 0)
        pulses = pulses(:,1:n);
    else
        pulses(:,n) = 0;
    end
end
pulses = pulses*2-1;          %change 0 by -1.
conjfftpulsesfolded(length(satellites),B) = 0;

%Perform pulse manipulation outside main loop. Here memory usage is
%increased but execution time is reduced.
for prncounter=1:1:length(satellites)
    pulsesfolded = sum(reshape(pulses(prncounter,:),B,p).');
    fftpulsesfolded = fft(pulsesfolded);
    conjfftpulsesfolded(prncounter,:) = conj(fftpulsesfolded);
end

tic     %start counting execution time
range = 0:n*p-1; %range for exponential multiplication

for frequencydopplershift=-10*p:1:10*p

    for ii=1:k
        exponential = exp(1i*(2*pi/(n*p))*frequencydopplershift.*range);
        bucketnp=wholesignal((ii-1)*p*n+1:ii*n*p);
        if(ntemp ~= n)
            if(ceil_floor == 0)
                bucketnp = bucketnp(1:n*p);
                length(bucketnp);
            else
                bucketnp(n*p) = 0;
                length(bucketnp);
            end
        end
        bucketnpexp(ii,:) =  bucketnp.*exponential;
        bucketfolded= sum(reshape(bucketnpexp(ii,:),B,p^2).');
        fftbucketfolded(ii,:)=fft(bucketfolded);     % FFT of folded length-p*n bucket
    end
    stop = 0;

    for prncounter=1:1:length(satellites)

        xs = zeros(1,B);    %clean accumulation buckets variable before going to other satellite
        xss= zeros(1,B);    %clean accumulation buckets variable before going to other satellite
        SNRmethod4=0; % SNR method 4 init
        stop = 0;           %stop detection when for a given doppler threshold value is exceeded

         for ii=1:k
            %IFFT of the pointwise product of fft of folded pulse and fft of folded
            %bucket
            outputfoldedbuckets=ifft(conjfftpulsesfolded(prncounter,:).*fftbucketfolded(ii,:));

            %Find the magnitude of the reult added to the accumulated
            %buckets
            xs=(abs(outputfoldedbuckets).^2)+xs;% accumulated output buckets to enhance detection
peak


            % Looking for the sample with the largest spike in the accumulated output bucket
            % The delay in the bucket is the largest sample positon.
            % For example, if it is a delay of n/p+x or 2*n/p+x this method gives x as the delay
in the folded bucket.
            % Therefore we have to find a way to settle the ambiguity (see below)

            [sizepeak,delayinfoldedbucket]=max(xs); %delay estimation based on the accumulation
of several output buckets
            xss = xs; % auxiliary variable to mantain the accumulator untouched
            %SIGNAL TO NOISE RATIO  (signal: peak power    noise: noise floor power)
```

```
            xss(delayinfoldedbucket)=0; %take peak out of accumulated output folded bucket
            noisefloorpwr=mean(xss);%power of noise bed in output folded bucket
            SNRmethod2=(sizepeak/noisefloorpwr);


            if stop == 0 && SNRmethod2>thresholdsparse  % threshold comparison
               buckets_used_method2=ii;                  %amount of data of size n*p used for
detection
               detected_doppler_shift = -(frequencydopplershift) ;%Doppler shift calculated based
on doppler from break
               deviation_from_IF=fi+detected_doppler_shift*(1000)/p;
               detected_code_phase_folded=delayinfoldedbucket;
               stop = 1;

               for jj=1:p
                   possibledelay(jj)= detected_code_phase_folded+(jj-1)*n/p;
               end

               for jj=1:p

cornp2(jj)=sum(pulses(prncounter,:).*bucketnpexp(ii,(possibledelay(jj):possibledelay(jj)+n-1)));

               end
               [size,delaychoice]=max(abs(cornp2));
               detected_code_phase=detected_code_phase_folded+(delaychoice-1)*n/p;
               if SNRmethod2 > whole_set(prncounter,2)
                    delays(prncounter,:) = [possibledelay];
                    whole_set(prncounter,:) =
[satellites(prncounter),round(SNRmethod2),deviation_from_IF,detected_doppler_shift,detected_code_
phase,ii];
               end

            end
        end
    end
end

executiontime = toc
```

```
% ========================================================================
%> @brief Baseline implementation with Doppler correction and 1000 Hz of accuracy.
%> @details
%> Doppler correction performed in time by means of a complex
%> exponential multiplication. Loop structure for detection is doppler->sat->buckets.
%> Support: mteixeir@ieee.org
%> Support: dmiralles2009@yahoo.com
%>    -# GPS PRN Correlator Detector (parallel code phase search)
%>    -# Doppler search step: 1000 Hz.
%>    -# Uses data 1ms in length (corresponds to 1 PRN) sampled at Fs
%> @date      May 15, 2014
%> @pre       Former File Name: BL_DPL_SAT_FH_TIME.m
%> @bug       Code Phase delay may be differ by 1 sampling period.
%> @note      This file represents the function version of the file name BL_DPL_SAT_TIME.m
%> @author    Damian Miralles, Marvi Teixeira, Manuel Ortiz, Jennifer Sandoval
%> @param dataset Selects the data set to be read for data. For more information on the data sets
available refer to \ref rawFiles.
%> @param fs Sampling frequency
%> @param fi Intermediate frequency
%> @param satellites List of prn satellites numbers to be detected. Values should range from 1 to
32.
%> @param thresholdbaseline Value to determine when a detection is valid
%>
%> @retval whole_set List of syncronization values to be returned.
% ========================================================================
function whole_set = bl_dpl_sat_fh_time(dataset,fs,fi,satellites,thresholdbaseline)

%Search for the number of arguments the user put when calling the function
switch nargin
    case 1
        error('Please enter sampling frequency and intermediate frequency')
    case 2
        error('Intermediate frequency value is compulsory')
    case 3
        satellites = [1:32];
    case 4
        if (max(satellites)>37) || (min(satellites)<1) || (min(satsize(satellites))~=1)
            error('sv must be a row or column vector with integers between 1 and 37\n')
        end
        thresholdbaseline = 10;
    case 5
        %do nothing. User entered all values into the function
    otherwise
        error('Wrong number of input values')
end

ts=1/fs;

%=========================================================================
% %RAW DATA 1 Settings (I and Q) (Present 01 03 07 19 20 22 24 28 31)
% satellites=[01 03 07 19 20 22 24 28 31];
% fs=12000000;
% ts=1/fs;
% fi=3563000;
%=========================================================================

%=========================================================================
% %RAW DATA 2 Settings  (only I component, requires code modification)
% satellites=[01 21 29 30 31];
% fs=5456000;
% ts=1/fs;
% fi=4092000;
%=========================================================================

%=========================================================================
% %RAW DATA 3 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
%(detected: ?3, 6,  21, 22, 26, 29,)
```

```
% satellites=[3 4 6 9 15 18 21 22 26];
% fs=38192000;
% ts=1/fs;
% fi=9548000;
%=========================================================================

%=========================================================================
% %RAW DATA 4 Settings (I and Q) Akos-Book ( 2nd data set)
% satellites=[01 02 03 04 05 06 07 17 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: 19,...)
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%=========================================================================

%=========================================================================
% % %RAW DATA 5
% satellites=[22 3 19 14 18 11 32 6]; %(sorted from strongest to weakest
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%=========================================================================

%CREATE PRN VECTOR TABLE
g=cacode(satellites, fs/1023000);
n=length(g(1,:));
p= round(sqrt(log(n)/log(2)));
k = 5;                              % number of buckets
maxallowedbymemory= n*p*k;          %use this amount of samples, based on
%MIT algorithm. This is used as a method for comparison with other
%implementations

% SELECT DATA FILE TO READ
switch dataset
    case 1
        % Data Set 1: compact.bin should be compactdataq_etc_etc...
        % [fid, message] = fopen('compact.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'ubit1')';
        % fclose(fid);
        [fid, message] = fopen('compact.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'ubit1')';
        fclose(fid);
   case 2
       % Data 2, with I component only
        %[fid, message] = fopen('gps.samples.1bit.I.fs5456.if4092.bin', 'r', 'b');
        %
        %
      statements
    case 3
        % Data 3, from Borre-Akos book
        % [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 4
        % % Data 4, from Borre-Akos book (2nd data set)
        % [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
    case 5
        % % Data 5,
        % [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        % data = fread(fid,maxallowedbymemory, 'bit8')';
        % fclose(fid);
        [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
        data = fread(fid,maxallowedbymemory, 'bit8')';
        fclose(fid);
```

```
    otherwise
       error('Wrong data set number');
end

nn=0:length(data)-1;
ficarrierQ=((cos(2*pi*fi*nn*ts)));
ficarrierI=((sin(2*pi*fi*nn*ts)));

% ******* I and Q using XOR of 1s and 0s *******
ficarrierI2=ficarrierI>0; % positives go to 1 negatives go to 0
ficarrierQ2=ficarrierQ>0;  % positives go to 1 negatives go to 0
data=data>0;
datafiI=xor(data,ficarrierI2); % xor data  with sin to obtain I
datafiQ=xor(data,ficarrierQ2); % xor data  with cos to obtain Q
wholesignal=(2*datafiI-1)+1*1i*(2*datafiQ-1);


%DATA INSTANTIATION
whole_set(length(satellites),6) = 0;
corr4(n) = 0;
fftsignal(k,n)=0;
x(n) = 0;

pulses = g;                   %obtain prn pulse code given the PRN number
pulses = pulses*2-1;           %change 0 by -1.

conjfftpulses(length(satellites),n) = 0;

%Perform pulse manipulation outside main loop. Here memory usage is
%increased but execution time is reduced.
for prncounter=1:1:length(satellites)
    fftpulses = fft(pulses(prncounter,:));
    conjfftpulses(prncounter,:) = conj(fftpulses);
end

tic
%DOPPLER SHIFT SEARCH
range = 0:n-1;
for frequencydopplershift=-10:1:10

    stop = 0;
    %Perform Doppler correction outside the main loop
    for ii=1:k
        exponential = exp(1i*(2*pi/n)*frequencydopplershift.*range);
        signalexp = wholesignal((ii-1)*n+1:ii*n).*exponential;
        fftsignal(ii,:) = fft(signalexp);
    end

    for prncounter=1:1:length(satellites)

        corr4=zeros(1,n); % correlation result preallocation
        SNRmethod4=0; % SNR method 4 init
        x=zeros(1,n); %acumulation bucket
        stop = 0;

        for ii=1:k %floor(length(wholesignal)/n) % divides signal into buckets of length n

            corr4 = ifft(conjfftpulses(prncounter,:).*fftsignal(ii,:));
            x=abs(corr4).^2+(x);            %accumulates with previous correlation

            [size,delaysamples4]=max((x));  % find the maximum size and position in accumulated
correlations

            %SIGNAL TO NOISE RATIO Calculation (signal: peak power    noise: noise floor power)
            xx=x; % auxiliary variable in order to keep the accumulator as it is
            xx(delaysamples4)=0; %takes peak out of correlation result
            %noisefloorpwr=var(corr); %calculate noise variance %power of noise in output bucket
            %noisefloorpwr=var(detrend(corr)); %power of noise in output bucket
            noisefloorpwr=mean((xx));  %power of noise bed in output bucket (peak not included)
            SNRmethod4=size/noisefloorpwr;  % SNR calculation for this method
```

```
            if stop == 0 && SNRmethod4>thresholdbaseline  % threshold comparison
                stop = 1;
                detected_doppler_shift_a= -(frequencydopplershift) ;%Doppler shift calculated
based on doppler from break
                deviation_from_IF_a=fi+detected_doppler_shift_a*1000;
                detected_code_phase_a=delaysamples4;
                if SNRmethod4 > whole_set(prncounter,2)
                    whole_set(prncounter,:) =
[satellites(prncounter),round(SNRmethod4),deviation_from_IF_a,detected_doppler_shift_a,detected_c
ode_phase_a,ii];
                end

            end

        end
    end

end
operation_time = toc
% results(programloops) = operation_time;
% %
%  end
```

```
%> @file      BL_DPL_SAT_FH_TIME.m
%> @brief     Standard baseline implementation with doppler correction and 1000 Hz
%of accuracy. Doppler correction performed in time by means of a complex
%exponential multiplication. Loop structure for detection is
%doppler->sat->buckets.
% Support: mteixeir@ieee.org
% Support: dmiralles2009@yahoo.com
%> @details
%>              1. GPS PRN Correlator Detector (parallel code phase search)
%>              2. Doppler search step: 1000 Hz.
%>              3. Uses data 2ms in length (corresponds to 1 PRN) sampled at Fs
%> @date       Apr 27, 2014
%> @pre        Former File Name: baseline_stand_alone_5b.m
%> @bug        Code Phase delay may differ by 1 sampling period.
%> @note       Data doppler correction performed ahead of the loop process.

%for programloops = 1:1:50
clear all
clc

%==========================================================================
% >RAW DATA 1 Settings (I and Q) (Present 01 03 07 19 20 22 24 28 31)
% ><a href="linkURL">link text</a>
% >satellites=[01 03 07 19 20 22 24 28 31];
% >fs=12000000;
% >ts=1/fs;
% > fi=3563000;

% %RAW DATA 1 Settings (I and Q) (Present 01 03 07 19 20 22 24 28 31)
% satellites=[01 03 07 19 20 22 24 28 31];
% fs=12000000;
% ts=1/fs;
% fi=3563000;
%==========================================================================

%==========================================================================
% > @attention RAW DATA 2 Settings  (only I component, requires code modification)
% > @<a href="linkURL">link Link to acces file</a>
% > @var satellites=[01 21 29 30 31];
% > @brief Set of PRN satellites are on the data sample
% > @var fs=5456000;
% > @brief Sampling frequency if the file
% > @var ts=1/fs;
% > @var fi=4092000;

% %RAW DATA 2 Settings  (only I component, requires code modification)
% satellites=[01 21 29 30 31];
% fs=5456000;
% ts=1/fs;
% fi=4092000;
%==========================================================================

%==========================================================================
% > RAW DATA 3 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
% > <a href="linkURL">link text</a>
% > satellites=[01 02 03 04 05 06 07 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: ?3, 6,
21, 22, 26, 29,)
% > fs=38192000;
% > ts=1/fs;
% > fi=9548000;

% %RAW DATA 3 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
 %(detected: ?3, 6,  21, 22, 26, 29,)
satellites=[3 4 6 9 15 18 21 22 26];
fs=38192000;
ts=1/fs;
```

```
fi=9548000;
%================================================================================

%================================================================================
% > RAW DATA 4 Settings (I and Q) Akos-Book ( 2nd data set)
% > <a href="linkURL">link text</a>
% > satellites=[01 02 03 04 05 06 07 17 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: 19,...)
% > fs=16367600;
% > ts=1/fs;
% > fi=4130400;

% %RAW DATA 4 Settings (I and Q) Akos-Book ( 2nd data set)
% satellites=[01 02 03 04 05 06 07 17 19 21 22 23 24 25 26 27 28 29 30 31]; %(detected: 19,...)
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%================================================================================

%================================================================================
% > RAW DATA 5 Settings (I and Q) Akos-Book (21 is present according to book, 19 is not)
% > <a href="linkURL">link text</a>
% > satellites=[22 3 19 14 18 11 32 6]; %(sorted from strongest to weakest
% > fs=16367600;
% > ts=1/fs;
% > fi=4130400;

% % %RAW DATA 5
% satellites=[22 3 19 14 18 11 32 6]; %(sorted from strongest to weakest
% fs=16367600;
% ts=1/fs;
% fi=4130400;
%================================================================================

%> @fn G=CACODE(SV,FS)
%> @author    Dan Boschen
%> @date      15 Apr 2007
%> @brief     Generates C/A Codes for selected PRNs, up to 37 codes.
%> G is a matrix with 1023*FS columns with a row for each PRN desired.
%> SV is a vector c

g=cacode(satellites, fs/1023000);
n=length(g(1,:));
p= round(sqrt(log(n)/log(2)));
k = 5; % number of buckets
maxallowedbymemory= n*p*k;          %use this amount of samples, based on
%MIT algorithm. This is used as a method for comparison with other
%implementations

% SELECT DATA FILE TO READ

% % the data 1: compact.bin should be compactdataq_etc_etc...
% [fid, message] = fopen('compact.bin', 'r', 'b');
% data = fread(fid,maxallowedbymemory, 'ubit1')';
% fclose(fid);

% Data 2, with I component only
%[fid, message] = fopen('gps.samples.1bit.I.fs5456.if4092.bin', 'r', 'b');
%
%
% Data 3, from Borre-Akos book
[fid, message] = fopen('GPSdata-DiscreteComponents-fs38_192-if9_55.bin', 'r', 'b');
data = fread(fid,maxallowedbymemory, 'bit8')';
fclose(fid);

% % Data 4, from Borre-Akos book (2nd data set)
% [fid, message] = fopen('GPS_and_GIOVE_A-NN-fs16_3676-if4_1304.bin', 'r', 'b');
% data = fread(fid,maxallowedbymemory, 'bit8')';
% fclose(fid);

% % Data 5,
% [fid, message] = fopen('gioveAandB_short.bin', 'r', 'b');
```

```
% data = fread(fid,maxallowedbymemory, 'bit8')';
% fclose(fid);

% figure
% hist(data)
% figure
% pwelch(data)


thresholdbaseline=12;


nn=0:length(data)-1;
ficarrierQ=((cos(2*pi*fi*nn*ts)));
ficarrierI=((sin(2*pi*fi*nn*ts)));

% ******* I and Q using XOR of 1s and 0s *******
ficarrierI2=ficarrierI>0; % positives go to 1 negatives go to 0
ficarrierQ2=ficarrierQ>0;  % positives go to 1 negatives go to 0
data=data>0;
datafiI=xor(data,ficarrierI2); % xor data  with sin to obtain I
datafiQ=xor(data,ficarrierQ2); % xor data  with cos to obtain Q
wholesignal=(2*datafiI-1)+1*j*(2*datafiQ-1);


% %******** I and Q using . (multiplication of 1s and -1s)********
% data = data * 2 - 1; %To convert values to +/- 1
% ficarrierI2=2*ceil(ficarrierI)-1; %
% ficarrierQ2=2*ceil(ficarrierQ)-1; %
% datafiI=(data.*ficarrierI2);
% datafiQ=(data.*ficarrierQ2);
% wholesignal=(datafiI)+1*j*(datafiQ);

whole_set(length(satellites),6) = 0;
deviation_from_IF_a = 0;
detected_code_phase_a = 0;
counter = 1;
corr4(n) = 0;
fftsignal(k,n)=0;
x(n) = 0;

pulses = g;                   %obtain prn pulse code given the PRN number
pulses = pulses*2-1;           %change 0 by -1.

conjfftpulses(length(satellites),n) = 0;

%Perform pulse manipulation outside main loop. Here memory usage is
%increased but execution time is reduced.
for prncounter=1:1:length(satellites)
    fftpulses = fft(pulses(prncounter,:));
    conjfftpulses(prncounter,:) = conj(fftpulses);
end

tic
%DOPPLER SHIFT SEARCH
stop=0;

range = 0:n-1;
for frequencydopplershift=-10:1:10

    stop = 0;
    %Perform Doppler correction outside the main loop
    for ii=1:k
        exponential = exp(1i*(2*pi/n)*frequencydopplershift.*range);
        signalexp = wholesignal((ii-1)*n+1:ii*n).*exponential;
        fftsignal(ii,:) = fft(signalexp);
    end

    for prncounter=1:1:length(satellites)

        corr4=zeros(1,n); % correlation result preallocation
```

```
        SNRmethod4=0; % SNR method 4 init
        x=zeros(1,n); %acumulation bucket
        stop = 0;

        for ii=1:k %floor(length(wholesignal)/n) % divides signal into buckets of length n

            corr4 = ifft(conjfftpulses(prncounter,:).*fftsignal(ii,:));
            x=abs(corr4).^2+(x);            %accumulates with previous correlation

            [size,delaysamples4]=max((x));  % find the maximum size and position in accumulated
correlations



            %SIGNAL TO NOISE RATIO Calculation (signal: peak power    noise: noise floor power)
            xx=x; % auxiliary variable in order to keep the accumulator as it is
            xx(delaysamples4)=0; %takes peak out of correlation result
            %noisefloorpwr=var(corr); %calculate noise variance %power of noise in output bucket
            %noisefloorpwr=var(detrend(corr)); %power of noise in output bucket
            noisefloorpwr=mean((xx));  %power of noise bed in output bucket (peak not included)
            SNRmethod4=size/noisefloorpwr;  % SNR calculation for this method


            if stop == 0 && SNRmethod4>thresholdbaseline  % threshold comparison
                stop = 1;
                detected_doppler_shift_a= -(frequencydopplershift) ;%Doppler shift calculated
based on doppler from break
                deviation_from_IF_a=fi+detected_doppler_shift_a*1000;
                detected_code_phase_a=delaysamples4;
                if SNRmethod4 > whole_set(prncounter,2)
                    whole_set(prncounter,:) =
[satellites(prncounter),round(SNRmethod4),deviation_from_IF_a,detected_doppler_shift_a,detected_c
ode_phase_a,ii];
                end

            end

        end
    end

end
operation_time = toc
% results(programloops) = operation_time;
% %
%  end
```