Faculty of Engineering Science
Department of Mechanical Engineering
Celestijnenlaan 300 – box 2420 – B-3001 Heverlee

Industrial Experience Report (H04O7A)

Company Name	Siemens Industry software
Student Name	Sonai Pandiyan Subramanian (R0648684)
Topic	Data Processing and Optimization of Errors in sensor position.

Contents

List of Figures

1 Abstract

This report discusses in detail about the internship work done by Mr. Sonai Pandiyan at Siemens Industrial Software, Leuven as a part of coursework in 'Masters of Mechanical Engineering'. The report is mainly divided into three main sections: company assessment, internship work and self reflection. The first section talks about the company, its global market and about its products. The next section talks about the work done by the student and results achieved. The final part talks about the student views and experience in the company.

2 PART A - Company assessment

This section will talk about the company 'Siemens PLM software'. It begins with the need for PLM software and then it talks about the global market leaders in the field. Finally it concludes by saying about the division inside the company and its products.

2.1 General Introduction

With the increase in complexity of managing a product from its engineering design to manufacturing and services, some of the companies opt for PLM solutions.
PLM refers to 'Product Life Cycle Management', which is a software that manages entires life cycle of the product by integrating People, data, process, services and business systems. The market growth estimates are around 10%. Some of the major Players in the field of PLM are

- Dassault Systems
- PTC
- Siemens PLM solutions
- SAP
- Oracle

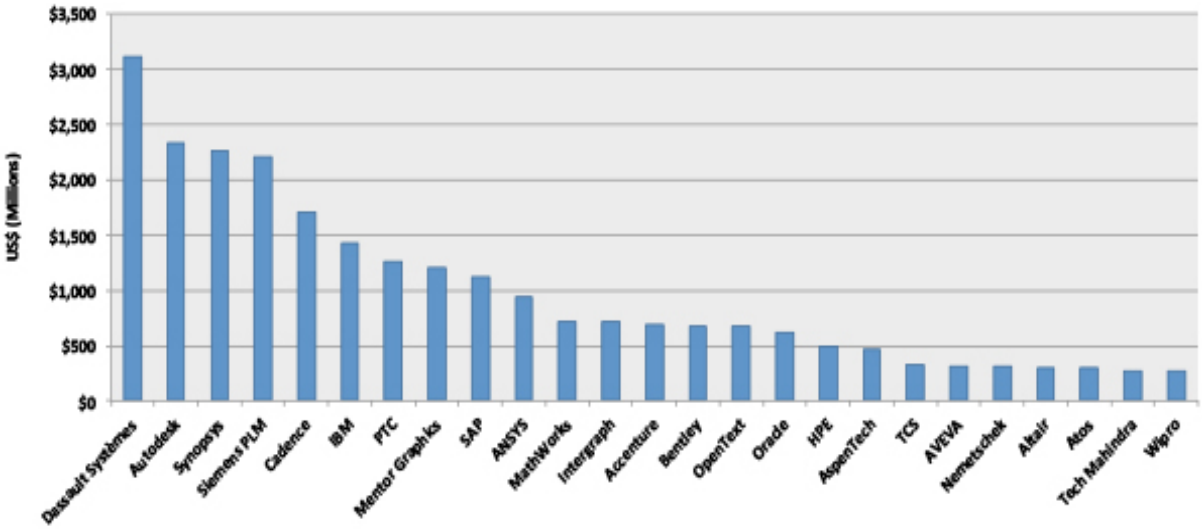

Figure 1: 2015 PLM Market Leaders[1]

Figure(1) shows the Global Market Leaders in PLM during the year 2015, and the figure(2) shows the financial market share by some of the major companies in PLM Market.

CIM data report PLM Revenue Leaders in 2015				
S.No	**PLM Providers**	**Market share**	**Direct revenue (Billion USD)**	**Package included**
1	Dassult Systems	13,10%	3.1	CATIA (CAD), SolidWorks,SIMULIA (CAE,simulation) and ENOVIA (PDM)
2	Siemens	10,00%	2.3	Teamcentre,Caxsoftware NX and Solid Edge, Tecnomatrix (digital manufacturing)
3	Autodesk		2.3	MCAD
4	PTC	5,40%	1.27	Windchill,CREO(CAD) and ThingWorx(IOT software)
5	SAP PLM		1.122	
6	Oracle PLM		0.619	

Figure 2: CIMdata report on PLM Revenue Leaders in 2015[2]

2.2 Siemens Industrial Software

Siemens PLM contains PDM (Product Data Management system) which brings information about CAD Data, models, parts information, manufacturing instructions through **Teamcenter**. **NX** offers integrated solution for CAD/CAM and CAE. During the design process Finite Element Analysis and multi body modeling of mechatronic systems is provided by **LMS Virtual Lab**.

Table 1 shows the growth of Siemens PLM over the years. It shows how Siemens after entering in PLM business by acquiring UGS in 2007 it expanded its function by including various application to its product.

Siemens PLM Growth			
s.no	year	PLM growth	
1	2001	UGS corporation merged with MSC NASTRAN	Nastran is FEA program
2	2007	Siemens acquires UGS	
3	2008	siemens acquires Innotec GmbH	digital engineering software is known for its COMOS platform
4	2009	Siemens acquires Elan software soln (France)	
5	2011	Siemens acquires Active technology in Automation Systems (Brazil)	pharmaceuticals and biotechnology production.
6	2011	Siemens acquires Vistagy, Inc (Massachutes)	Specialized engineering software for manufacturing structures made of advanced composite materials
7	2012	Siemens acquires Vrcontext International S.A (Belgium)	integrated real-time visualization software for the oil and gas, process, energy, architecture/ engineering/construction, and homeland security sectors worldwide
8	2012	Siemens acquires LMS International (Belgium)	Mechatronic simulation and testing software producer
9	2013	Siemens acquires TESIS PLMware (Munich based)	To provide a platform for integration with ERP systems

Table 1: Siemens PLM growth

LMS international is a spin-off from KU Leuven, which provides mechanical simulation and advanced testing in the product development in automobile, aerospace and manufacturing industries. To maintain the technology edge LMS develops methods to validate, together with industries to meet the needs of the end users. Some of its major customers are John Deere, Airbus (for landing gear analysis), Boeing (aileron flap mechanism analysis).
Siemens industrial software (former known as LMS International) located in Interleuvenlaan, Leuven has 4 divisions inside

1. System simulation (1D simulation)
2. 3D simulation
3. Testing
4. Engineering Division

2.3 Products[7]

2.3.1 System simulation

With the increase in mechatronic systems in industrial products, a software with multi-domain dynamic behavior embedded in it is necessary. LMS Imagine Lab is a software used to create and simulate 1D-model of a system to analyze the multi-domain model.

- LMS Imagine Lab Amesim

  Used to simulate functional performance of mechatronic system in its early development stages. Amesim has physical domain libraries in

  - Fluids
  - Thermodynamics
  - Electrics
  - Electromechanical
  - Mechanics
  - Signal Processing

  and application libraries for

  - Cooling systems
  - Air conditioning
  - IC engines
  - Aerospace

- LMS Imagine Lab Sysdm

  Used to store and organize system mechanical and control models in a common Repository which can be shared with in a team/ work-group/ with in a division.

- Embedded software Designer

  Embedded software designer is an IDE(Integrated Development Environment) for the development of on-board software. Its application includes cyber-physical systems such as smart vehicles, home appliances and buildings, transportation and tourism services based on continuous analysis of IOT generated data.

Applications

1. Automotive

   Assesses global vehicle dynamic performances in terms of fuel economy, durability & safety in the early stages of design

   - Balancing
   - Power train transmission
   - Internal combustion
   - Thermal Management
   - Electrical system

2. Aerospace Industry

   Provide mechatronic simulation function requirement to detailed design in landing gears in performance analysis, landing gear extension and retraction, breaking and steering systems.

2.3.2 3D Simulation

LMS Virtual Lab and LMS Samtech mimic the real-world behavior of the product under development.

- LMS Virtual Lab

  It is a 3D Finite element model tool used to simulate real-world performance of mechatronic systems.

  VL Acoustic and Vibration.

  VL Acoustics and Vibration can predict the Noise and Vibration performance of the new design in very early stages rather than at the end of the design.

  VL Motion

  Helps in analyzing and optimizing mechanical and mechatronic systems in a virtual way during its design stages.

  VL Durability

  Used to give durability analysis predictions. It provides feedback regarding critical durability areas, critical loads and critical events so that we can optimize the durability of the product.

2.3.3 Testing Solutions

LMS Testing solutions helps in increasing Test efficiency and Productivity in the field of structural dynamics, acoustics, rotating machinery.
LMS Test Lab along with their Data Acquisition systems helps in collecting the required data in the format required with the PC or autonomous recording.

3 Part B: Internship Work

This section tells about the work done by the student in the company. It starts by giving introduction about the problem and the methods followed to solve the problem and the validation of the solution. It also talks about the structure used in the MATLAB programming for better understanding of the program by the user.

3.1 Introduction

Rigid Body Analysis
A body is considered rigid if the body doesn't deform on the application of force. However in reality no body is perfectly rigid. It will be considered rigid if the frequency range of deformation is within its first deformation mode, so the whole body can be compared to a system with mass attached to a spring and damper elements.

A rigid body has 6 degrees of freedom, 3 vibrational translation and 3 rotation about x,y and z axes. In order to measure the forces acting on such rigid bodies, measurement of all 6 acceleration is required. The measured vibrational acceleration will differ from the theoretical acceleration values of the rigid body due to the following[6]

- Errors in finding the position of measurement
- Error in measurement direction with respect to the desired direction
- Wrong calibration value of the accelerometer
- Other measurement errors

3.2 Problem Definition

The test setup consists of the engine and transmission system of a car. The engine is run down from 2900rpm to 1100rpm and acceleration readings are taken with the help of accelerometer placed on 5 different positions out of 14 possible positions identified. These accelerometer records the acceleration values in all three direction (x,y,z). These readings are used to estimate the vibrational acceleration of the engine and transmission system.

The frequency domain of interest is <100HZ. At this frequency range the body can be considered as rigid. Acceleration readings measured from the sensors (accelerometers) are in the sensor co-ordinate system. These measurements are used to estimate the vibrational acceleration at center of gravity of the body or at some common node (Nodes are the point of measurement or a point of interest). The estimated values may not give the actual vibrational values of the engine due to the causes mentioned in section 3.1. **The main objective of the internship is to develop a program to estimate the possible errors (ref 3.1) in the Euler angles of the sensors and errors in position measurement of the sensors by optimization techniques and finding the best position for the placement of the sensors.**

3.3 Theory

3.3.1 Base coordinate Rotation

The nodes corresponding to the response DOF's used, do not have global directions or when a reference (not coincident with the global origin) is specified, rotation of the measured accelerations to the global/reference axis system will then be needed[6].
Euler angle is used for 3D representation of an object, using a combination of rotations about 3 different axes

Rotation matrix for rotation about z axis (yaw) is given by

$$R_{\mathrm{z}} = \begin{bmatrix} cos(\theta_{xy}) & sin(\theta_{xy}) & 0 \\ -sin(\theta_{xy}) & cos(\theta_{xy}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation matrix for rotation about y axis(pitch) is given by[4]

$$R_{\mathrm{y}} = \begin{bmatrix} cos(\theta_{zx}) & 0 & -sin(\theta_{zx}) \\ 0 & 1 & 0 \\ sin(\theta_{zx}) & 0 & cos(\theta_{zx}) \end{bmatrix}$$

Rotation matrix for rotation about x axis(roll) is given by

$$R_{\mathrm{x}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_{yz}) & sin(\theta_{yz}) \\ 0 & -sin(\theta_{yz}) & cos(\theta_{yz}) \end{bmatrix}$$

and the Rotation matrix is product of three matrices [4]
$[\mathrm{R}]_{\mathrm{o}} = \mathrm{R_x} * \mathrm{R_y} * \mathrm{R_z}$

$$R_{\mathrm{o}} = \begin{bmatrix} C(\theta_{xy})C(\theta_{xz}) & C(\theta_{xy})S(\theta_{yz})S(\theta_{xz}) - C(\theta_{yz})S(\theta_{xy}) & S(\theta_{yz})S(\theta_{xy}) + C(\theta_z)C(\theta_{xy})S(\theta_{yz}) \\ C(\theta_{xz})S(\theta_{xy}) & C(\theta_{yz})C(\theta_{xy}) + S(\theta_{yz})S(\theta_{xy})S(\theta_{xz}) & C(\theta_{yz})S(\theta_{xy})S(\theta_{xz}) - C(\theta_{xy})S(\theta_{yz}) \\ -S(\theta_{xz}) & C(\theta_{xz})S(\theta_{yz}) & C(\theta_y z)C(\theta_{xz}) \end{bmatrix}$$

The three measured (local) accelerations of output node "o":

$$\{\ddot{X}\}_g = [R]_0^{-1}\{\ddot{X}\}_l \tag{1}$$

where:

$\{\mathrm{X}\}_{\mathrm{g}}$ is column vector with the global acceleration values

$\{\mathrm{X}\}_{\mathrm{l}}$ is the column vector with local acceleration values

$[\mathrm{R}]_{\mathrm{o}}$ is the rotation matrix (global to local) of node "o"

$\mathrm{C}(\theta_{\ \mathrm{ij}})$ and $\mathrm{S}(\theta_{\ \mathrm{ij}})$ refers to $\cos(\theta_{\ \mathrm{ij}})$ and $\sin(\theta_{\ \mathrm{ij}})$ respectively

When a reference is specified, which does not coincide with the global direction, the three measured accelerations of output node "o" are also rotated according to the axis of the reference system.

$$\{\ddot{X}\}_r = ([R]_0^{-1}[R]_r)\{\ddot{X}\}_l \tag{2}$$

where:

$[\mathrm{R}]_{\mathrm{r}}$ is the rotation matrix (global to local) of node "r".

Validation of the rotation matrix is done by giving Euler angle of 90 in one of the axis (z axis) and checked whether values in remaining axes (x and y axis) are interchanged with sign value on one axes reversed.

3.3.2 Co-ordinate Transformation

Since the measurements are done at points on the surface of the engine which are at different distances from the Center Of Gravity(COG) of the engine, these measurement will have different moments due to the rotational degrees of freedom. So co-ordinate axes transformation is done.

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{bmatrix} A_t x \\ A_t y \\ A_t z \end{bmatrix} + \begin{bmatrix} \alpha_y * R_z - \alpha_z * R_y \\ -\alpha_x * R_z + \alpha_z * R_x \\ \alpha_x * R_y - \alpha_y * R_x \end{bmatrix}$$

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & R_z & -R_y \\ 0 & 1 & 0 & -R_z & 0 & R_x \\ 0 & 0 & 0 & R_y & -R_x & 0 \end{bmatrix} X \begin{bmatrix} A_t x \\ A_t y \\ A_t z \\ \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}$$

where:

A_i represents the acceleration value measured in i direction at the sensor location

A_{ti} represents the translational acceleration value in i direction

α_i represents the tangential acceleration.

R_i represents the perpendicular distance(along direction i) from the center of rotation to the node where measurement is done.

$$\{\ddot{X}_g\} = [T^{-1}] * \{\ddot{X}_l\} \tag{3}$$

where:

$\{\ddot{X}_g\}$ refers to acceleration values at COG

$\{\ddot{X}_l\}$ is the acceleration values at sensor position

[T] is the Transformation matrix (global to local)

3.4 DOF

The acceleration values at five different sensor position is transformed to the acceleration values at the center of gravity (as mentioned in the section 3.3.2) and five values for acceleration in each direction at different rpm is calculated. Using least square method (*pinv*) a best fit curve for these 5 different values is achieved.

'pinv()' refers to the MOORE-PENROSE Pseudo inverse which is used to find the best fit for the system of equation which doesn't have a unique solution. To find the accuracy of estimated acceleration values, DOF's at COG (or at the Reference node) are extrapolated to the nodes where readings are taken and compared with the actual measured values.

The figure (3) shows the estimated (in red color) and actual acceleration (in blue color) values measured at 5 different accelerometer positions in x,y and z direction.

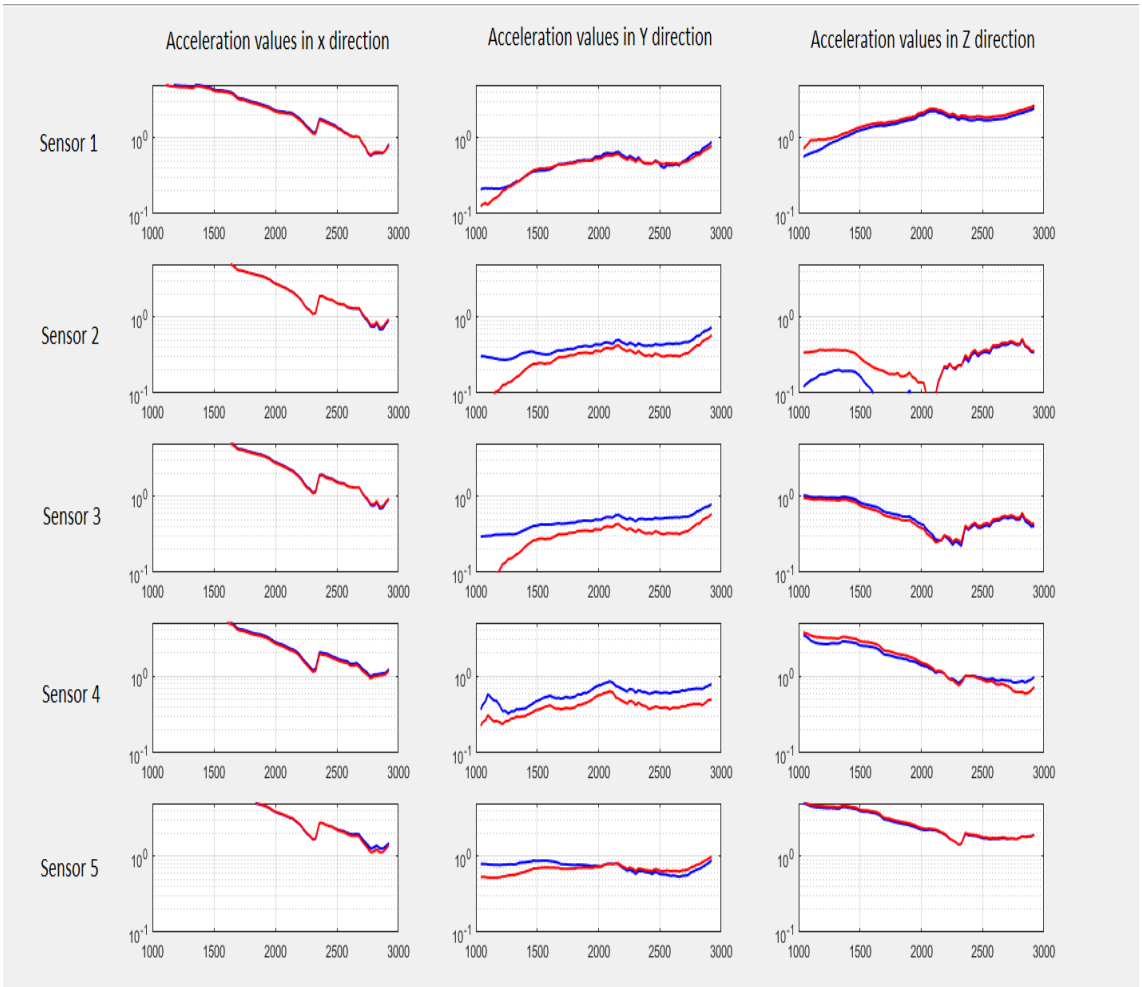

Figure 3: Comparison of estimated(red) vs actual measurement values(blue)

3.5 Optimization

To find the possible error that might be available with the position of sensors, an optimization problem is solved. The cost function that needs optimization is the Euclidean norm of the difference between the actual measured values of acceleration and the acceleration values extrapolated from the center of gravity of the engine to the sensor position values (estimated).

The parameters that need to be optimized are error in Euler angles of each sensors and the error in position variation of each sensor from the given distance. Maximum error in sensor position and sensor angle are set as constraints while solving the optimization problem. Figure(4) shows the upper and lower bound bounds given for the sensors angle and position error.

Since 5 sensors are used in total, the objective function solves for a matrix of dimension 5*6 with each element containing error in the Euler angles in a particular direction or error in the sensor position measurement in a particular angle.

$$min_{\theta_i, e_i} \frac{1}{2} * ||\{\ddot{X}_l\} - \{R\} * \{T\} * \{\ddot{X}_g\}||_2{}^2 \quad (4)$$

$$subjectto - 15 \leq \theta_i \leq 15 \quad (5)$$

$$-0.05 \leq e_i \leq 0.05 \quad (6)$$

where:

$\{X\}_g$ represents the acceleration values at COG

$\{X\}_l$ represents the local acceleration values measured.

$\{R\}$ is the rotation matrix (global to local).

$\{T\}$ is the Translational matrix (global to local).

θ_i refers to the error in Euler angle of sensor in x, y, z direction.

e_i refers to the position error of sensor in x, y, z direction.

3.6 Steps followed in MATLAB

This subsection tells about the steps followed in the MATLAB program

- Read the Universal File and get the following data
  - Nodes at which sensors are placed.
  - x,y,z position of all the nodes.
  - Acceleration values measured by each sensors in x, y, z direction.
  - Rpm values.
  - Gyroscope readings (if any).
- Form the Rotation matrix to orient the sensor direction to the global/Reference direction.
- Form the matrix to convert the acceleration values from COG/Reference node to the nodes at which sensors are placed.
- Calculate the acceleration values at the COG using the measured values and the matrices found in the above 2 steps.
- Using the acceleration value at the COG the acceleration values are estimated at the sensor positions.
- Calculate the difference between the acceleration values estimated and the acceleration values actually measured.
- Objective function which minimizes the differences (by introducing error in EULER angle to each sensors and introducing error in the measurement of the distances of the sensors) is created and fed to the optimization program.
- Optimization subroutine is run several times with different starting points, Error values for which the objective function has minimum cost is stored as the Euler angle Error and Position error of each sensors.

3.7 Finding Best position for sensor placement

Out of 14 possible positions for placement of accelerometers some of them are more suitable as they are less affected by positional and angular error in placement of the accelerometer.The following steps explain the subroutine used to find the best sensor position.

Steps followed to find the best sensor positions:

- Rigid body acceleration $\{\ddot{X}_g\}$=[$\ddot{x}$ $\ddot{y}$ $\ddot{z}$ $\ddot{\theta}_x$ $\ddot{\theta}_y$ $\ddot{\theta}_z$] is fixed to be a constant value and the acceleration at the sensor positions are estimated.
- A small error (e) in position/ angle in one of the sensor is introduced and the acceleration of Rigid body is calculated $\{\ddot{X}_{g\ \text{after error}}\}$
- Euclidean norm of the difference between acceleration values before and after introduction of the error is found $\parallel \{\ddot{X}_g\}\text{-}\{\ddot{X}_{g\ \text{after error}}\} \parallel_2$
- The above said procedure is followed for all possible sensor positions and a vector of Euclidean norm is found.
- The Vector is sorted in the order of best position for placement of accelerometer.

3.8 Structured Programming

A program is a combination of data, functions and control flow. A structured program aims at improving the quality and clarity of the program with the help of subroutines and blocks, loops and structure[3]

Blocks are codes which are grouped together so that they can be treated as if they are one statement. Moreover the variables declared in a function will not conflict with the variable in same name in other function. Functions also help in re-usage of the same block of codes again and again. Group of variables describing particular features are grouped together in a structure for easy usage.

Here data represents the values read by the sensors. The function deals with the domain specific functions such as optimization algorithm, reading data form universal file, or formation of rotation / transformation matrices etc... Separate user defined functions are created for each and every domain specific operation, so that person who deals with a specific domain does not have to know how the other functions work. In order to run the whole program or to modify it, the user has to change the control flow and reconfigure it a little if necessary.

The flowchart explaining the flow of program is shown in fig(12).

3.9 Result

Figure 4 shows the bounds in angular errors are assumed to be from -15 degree to +15 degrees and the error in position measurements is taken from -5 cm to +5 cm for the optimization algorithm. Euclidean norm of error Values before optimization = 4.1572 and norm of the errors after optimization = 0.6247.

The estimated error values are found as shown in fig(5). The plot in figure(6) compares the measured values and estimated values before and after optimization. It shows clear improvement after optimization in sensor 2 (y-direction and z-direction) acceleration measurement and in sensor 4 (y-direction) measurement.

Lower bound						
Sensor Number	Euler Angle Error Bound			Position Error Bound		
	X	Y	Z	X	Y	Z
Sensor 1	-15.00	-15.00	-15.00	-0.05	-0.05	-0.05
Sensor 2	-15.00	-15.00	-15.00	-0.05	-0.05	-0.05
Sensor 3	-15.00	-15.00	-15.00	-0.05	-0.05	-0.05
Sensor 4	-15.00	-15.00	-15.00	-0.05	-0.05	-0.05
Sensor 5	-15.00	-15.00	-15.00	-0.05	-0.05	-0.05

Upper bound						
Sensor Number	Euler Angle Error Bound			Position Error Bound		
	X	Y	Z	X	Y	Z
Sensor 1	15.00	15.00	15.00	0.05	0.05	0.05
Sensor 2	15.00	15.00	15.00	0.05	0.05	0.05
Sensor 3	15.00	15.00	15.00	0.05	0.05	0.05
Sensor 4	15.00	15.00	15.00	0.05	0.05	0.05
Sensor 5	15.00	15.00	15.00	0.05	0.05	0.05

Figure 4: Bounds given on the Position and Angular errors

Error Values after optimization						
Sensor Number	Euler Angle Error			Position Error		
	X	Y	Z	X	Y	Z
Sensor 1	1.219	0.066	-0.918	0.049	-0.050	-0.017
Sensor 2	-0.130	-1.547	-1.266	-0.022	-0.024	0.035
Sensor 3	-9.173	-3.826	-3.851	-0.036	-0.018	0.041
Sensor 4	-0.277	-1.081	-0.141	-0.008	0.050	-0.048
Sensor 5	2.093	-1.189	1.019	-0.025	-0.050	0.020

Figure 5: Error Values of DOF after optimization routine

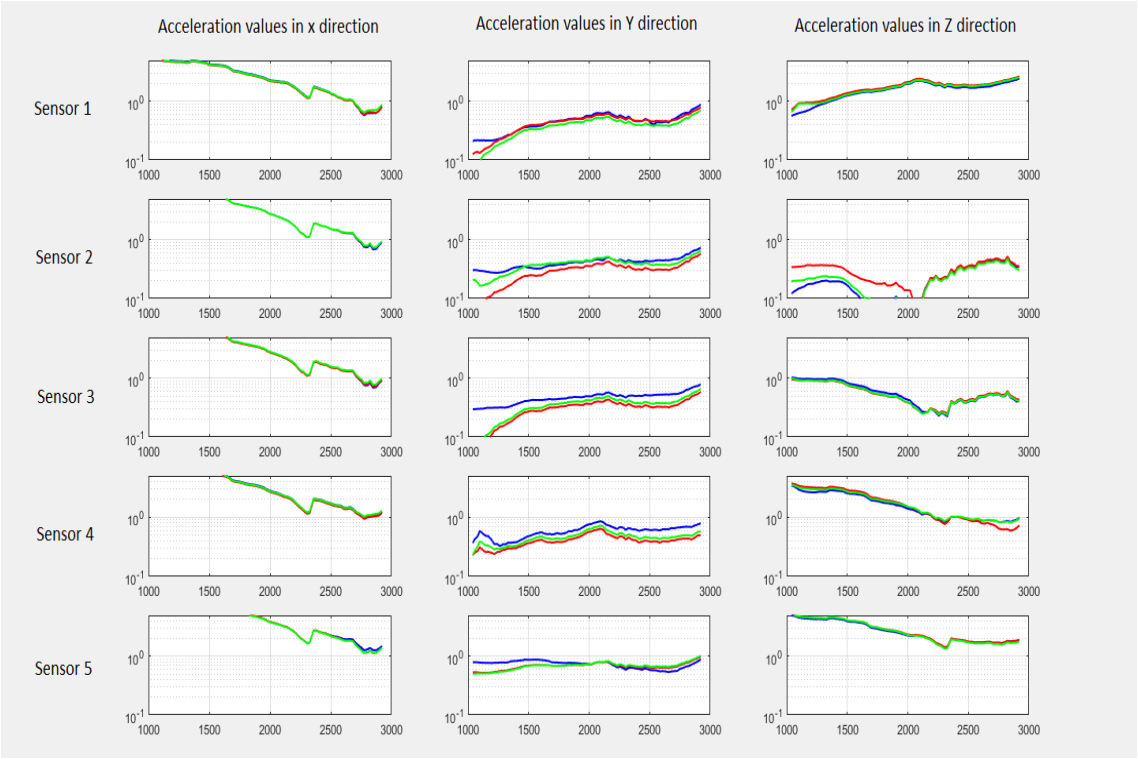

Figure 6: Comparison of acceleration values of Measured(Blue), estimated before (Red) and after Optimization(Green)

Possible Sensor Position details			
Sensor num	X position	Y position	Z position
1	0.0282	0.4217	0.2437
2	0.0982	-0.4383	0.1537
3	0.0982	-0.4383	0.1537
4	0.1982	-0.1333	-0.2963
5	-0.0718	-0.0983	0.3137
6	0.0000	0	0
7	0.0102	0.4217	0.2364
8	0.0602	-0.4283	0.1614
9	0.0402	-0.4283	0.1614
10	0.1752	-0.1233	-0.2336
11	-0.1098	-0.1033	0.2964
12	0.0582	-0.4765	0.1495
13	0.0032	0.4737	0.2149
14	0.1982	-0.1263	-0.2691
15	0.2282	0.5417	0.3837

Figure 7: Available position for placing the sensors

Comparison of Sensor positions for finding best position for placement of sensors															
Norm of Error	0.0364	0.0458	0.0465	0.0488	0.0489	0.0490	0.0490	0.0497	0.0535	0.0543	0.0566	0.0676	0.0748	0.0798	0.0934
Sensor position	6	9	8	5	12	3	2	11	7	1	13	10	14	4	15

Figure 8: Best Position for Sensor Placement

Figure(7) shows the available position for placement of accelerometer sensors, and the best positions for placement of sensor arranged in ascending order (from optimal to least optimal position) is shown in fig(8).

3.10 Validation

To validate the process, gyroscopes are placed in some of the nodes and the experiment is conducted. The angular acceleration estimated by the above method is compared with the angular acceleration values estimated using gyroscopes.

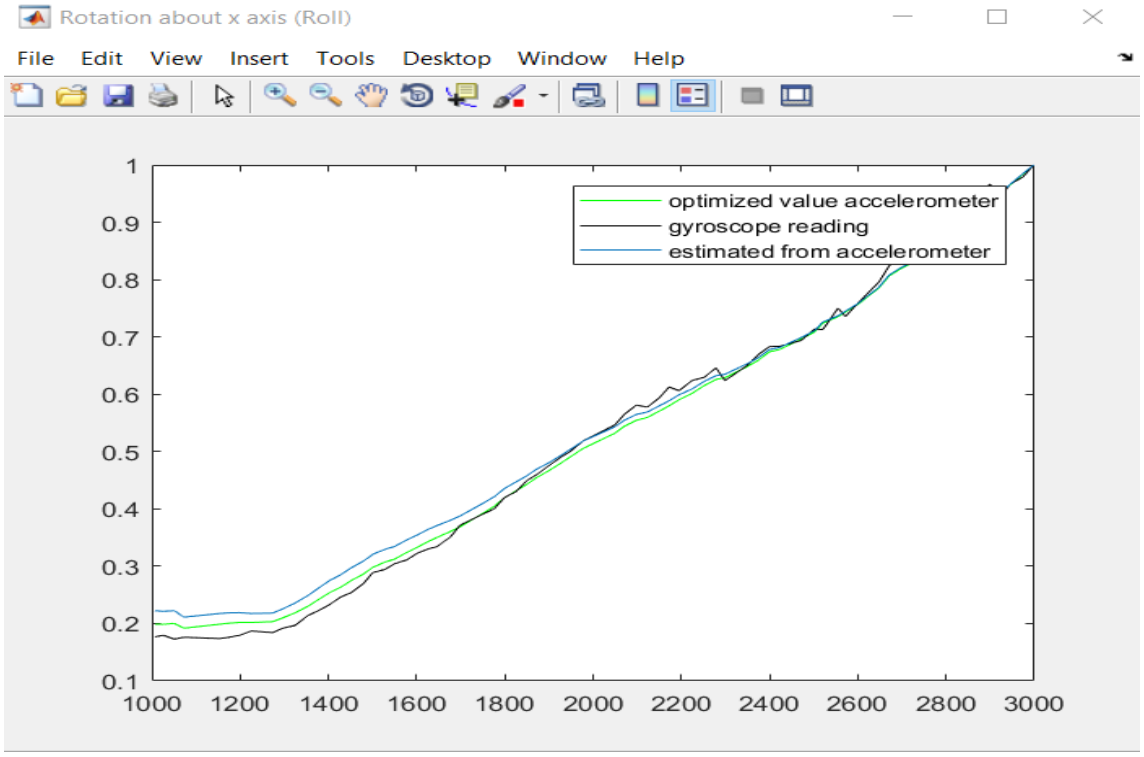

Figure 9: Comparison of Angular acceleration about X axis

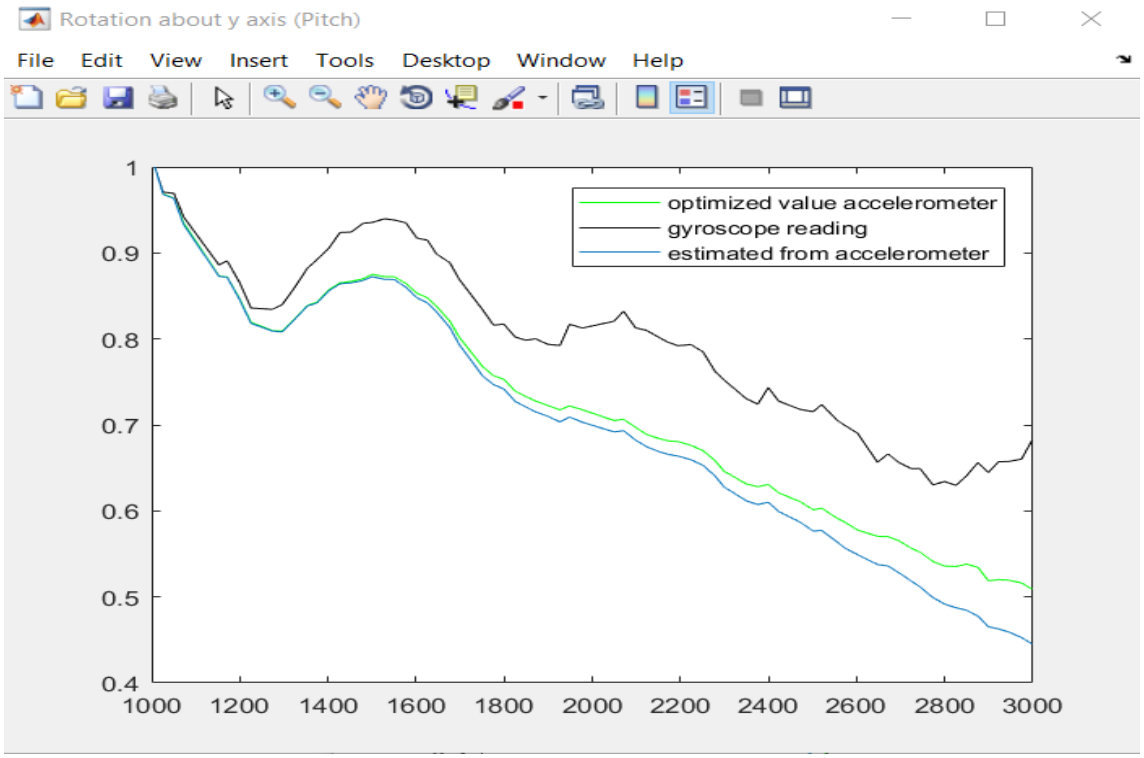

Figure 10: Comparison of Angular acceleration about Y axis

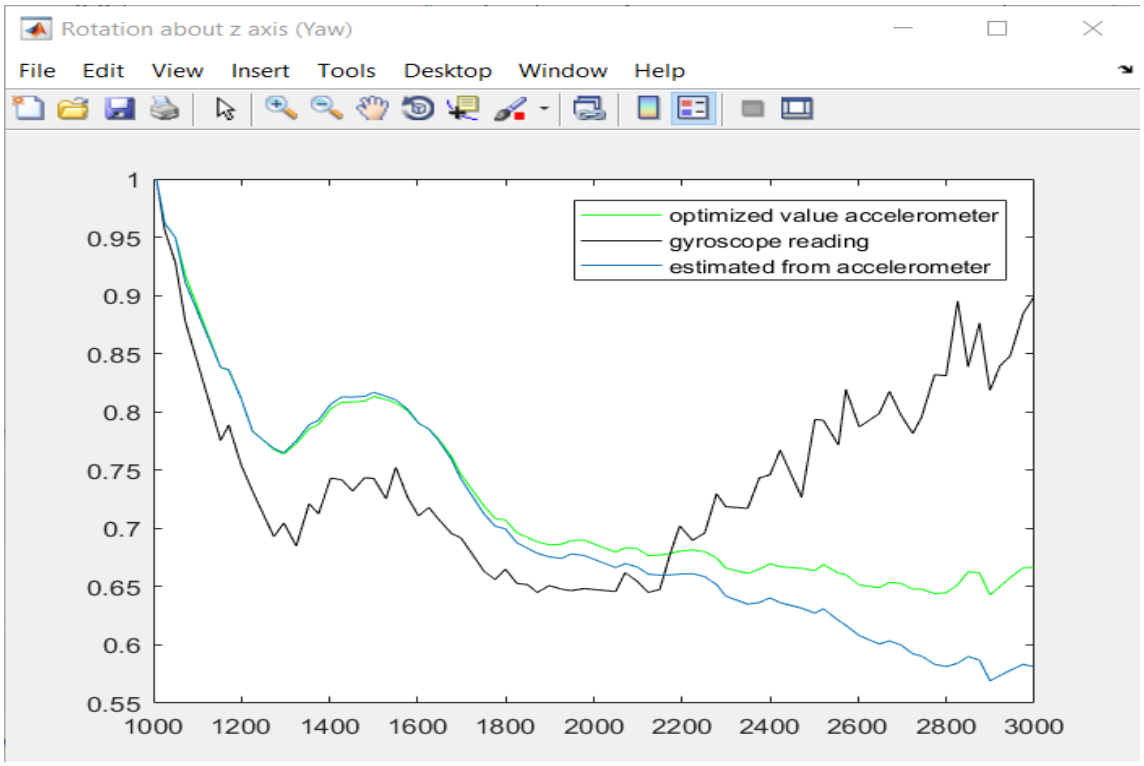

Figure 11: Comparison of Angular Acceleration about Z axis

fig9,10,11 shows the comparison of normalized values of angular acceleration estimated using gyroscope and angular acceleration measured using accelerometers. X axis of the graph shows the rpm values and y axis shows its corresponding acceleration values. It is clearly seen that the optimized acceleration values line remain closer to the gyroscope reading values. Thus proving that the program improves the acceleration measurement.

3.11 Further Improvement to be done

The optimization program only finds a local minimum. The subfunction *'sensoroptim'* feeds random starting points, and then the minimum point among the different results of optimization program is taken as the minimum. This algorithm can be improved by using a global optimization algorithm.

In fig 10,11 the difference between the angular acceleration values measured using gyroscopes and the acceleration values estimated form accelerometer is high at higher rpm levels. This deviations cannot be explained. Validation process need to be done with one more set of data.

4 PART C Self Reflection

Being a part of the engineering group in Siemens industry software is a very knowledgeable experience. Being in the field of engineering and implementing what is learnt in the textbook gives some experiences which are not given by the textbooks. It also gives an overview of the new technologies the companies are working on, which will be part of our life in the near future.

The company is well organized and every person in the company is easily approachable. People are ready to help with understanding of the project. Frequent meetings with the supervisors and mentors helped me whenever I got stuck up in the projects.

Working with Siemens has given me a basic understanding on 'optimization', which I will be following in the course curriculum later in my studies, It also improved my understanding in the subject 'Sensors and Measurement'. My internship work also helped in improving my MATLAB skills.

References

[1] Cimdata.com. (2017). CIMdata Publishes PLM Market and Solution Provider Report - CIMdata. [online] Available at: https://www.cimdata.com/en/news/item/4456-cimdata-publishes-plm-market-and-solution-provider-report [Accessed 1 sep. 2017].

[2] Anon, (2017). [online] Available at: https://www.engineering.com/PLMERP/ArticleID/12946/PLM-This-Week-CIMdata-Report-Shows-Dassault-Systemes-is-a-PLM-Revenue-Leader-in-2015.aspx [Accessed 6 Nov. 2017].

[3] Bruyninckx, Herman, Markus Klotzbùcher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. *"Roc"*. Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13 (2013).

[4] "Understanding Euler Angle." *CH Robotics,* Redshift Labs Pty Ltd, 0ADAD, www.chrobotics.com/library/understanding+-euler-angles.

[5] Latt, W., Tan, U., Riviere, C. and Ang, W. (2011). Placement of accelerometers for high sensing resolution in micromanipulation. *Sensors and Actuators A: Physical, 167(2), pp.304-316.*

[6] Siemens Industry Software NV, PDF on Rigid body modes *LMS Test.Lab*

[7] Software, S. (2017). *LMS: Siemens PLM Software.* [online] Plm.automation.siemens.com. Available at: https://www.plm.automation.siemens.com/en/products/lms/ [Accessed 1 sep. 2017].

5 Appendix

5.1 Flow chart

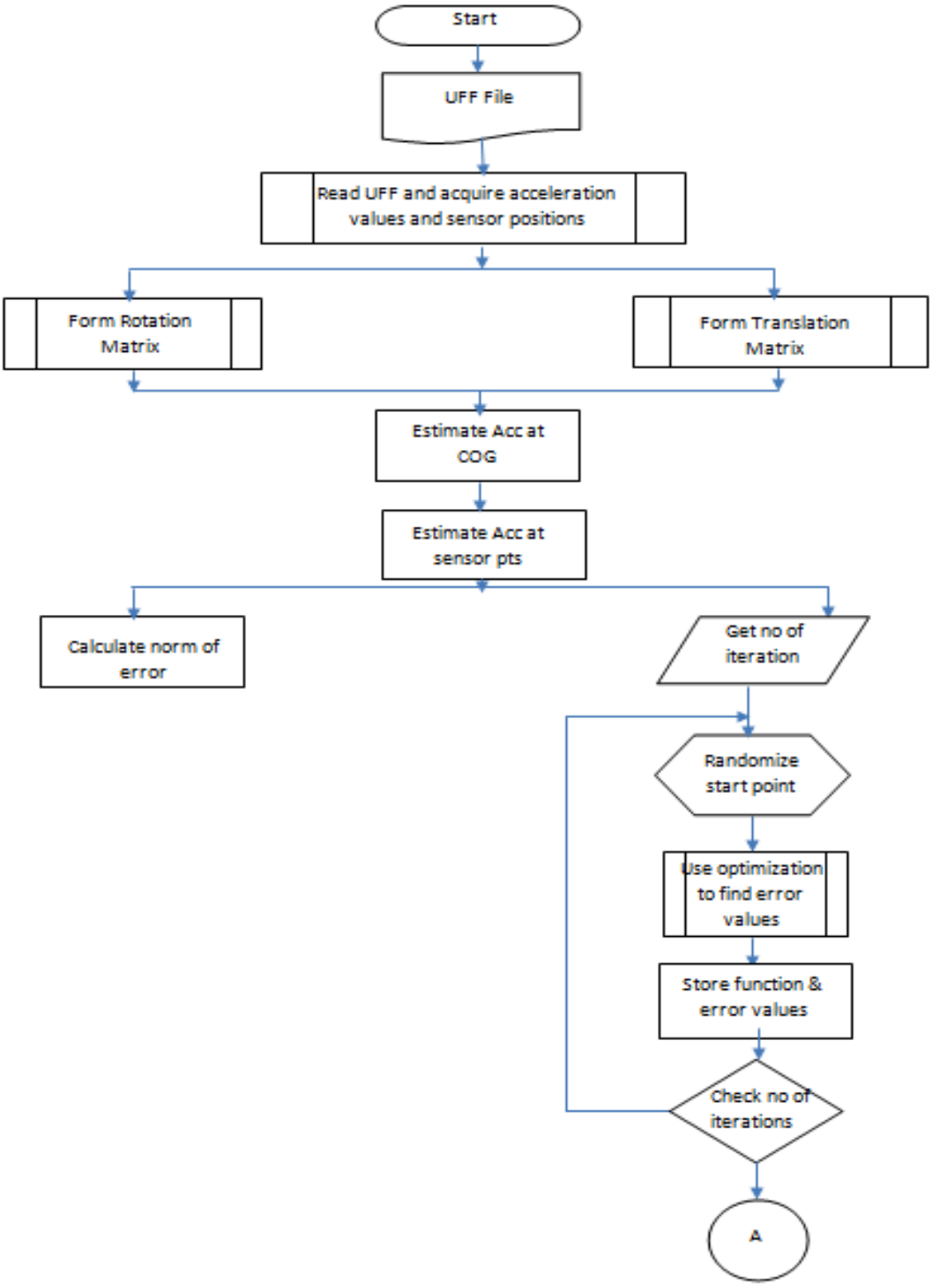

Figure 12: Flow chart Explaining the flow of program

5.2 Flow chart

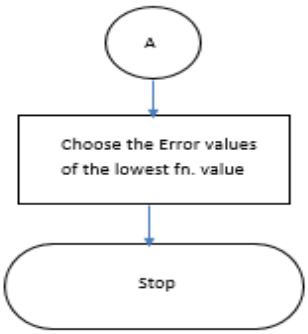

Figure 13: Flow chart Explaining the flow of program cont.

5.3 Matlab code

5.3.1 Main program

```
clear;
close all;
clc;
% cd 'E:\I\matlab';
% file=fopen(which('R376.unv'),'r');
% file=fopen(which('R400.unv'),'r');
 file=fopen(which('order_2_gyro_acc1.unv'),'r');
global F ;
%% reading the UFF files

% All the datas from the file is stored in a structure Data
[Data]=read_UFF(file);
%% manually clearing the noise in certain places

% this block is used for clearing noise from file order_2_gyro_acc1.unv
temp=isempty(Data.RotY);
if temp<1
a=[77,76,71,52,41,40,28,23,16];
for j = 1:numel(a)
    Data.acc_values(a(j),:)=[];
    Data.freq(a(j))=[];
    Data.RotY(a(j),:)=[]
 end
clc;
clear a i j file;
end
%% converting the acceleration values form sensor pos to COG

F=zeros((numel(Data.sen_no))/3,6);
% Function transforms sensor values from sensor axes to global orientation
```

```
Rotation_mat=EULER_ANGLES(Data);
mat=values_at_COG(Data);
% to find 6 DOF acceleration values at COG
acc_at_COG=pinv(mat)*Rotation_mat*Data.acc_values.';

%% Estimating the accelration values at the sensor position

acc_est=(pinv(Rotation_mat)* mat*acc_at_COG).';
% diff =((abs(Data.acc_values).') - abs(mat * pinv(mat)* Rotation_mat*...
%     Data.acc_values.')).^2;
% diff =((abs(Data.acc_values)) - abs(acc_est)).^2;
% rms1=rms(rms(diff,2),1)*10000;
% clear diff
diff1=((abs(Data.acc_values)) - abs(acc_est));
norm2=norm(diff1);
clear diff1;

%% Plotting measured, estimated accelreation values
NODE_uniq=unique(Data.sen_no);

 fig(1)=figure('Name',...
     'comparision of measured,estimated and optimized acceleration values'...
     ,'NumberTitle','off')
for ii=1:numel(Data.sen_no)
    subplot(numel(NODE_uniq),3,ii)
        semilogy(Data.freq,abs(Data.acc_values(:,ii)),'b',...
            Data.freq,abs(acc_est(:,ii)),'r','linewidth',1.5);
    xlim([1000 3000])
    ylim([0.1 5])
    grid
end
mtextbox=uicontrol('style','text','position',[0,0,1500,40]);
set(mtextbox,...
    'string','Blue - Measured Acc values , Red - Estimated Acc value');
clear ii;

%% optimization for EULER angle and Distance

 [F,fnvalue,opt]=sensoroptim(Data);


%% Estimating the acceleration values at the sensor positions

NODE_uniq=unique(Data.sen_no);
Rotation_mat=EULER_ANGLES(Data);
mat=values_at_COG(Data);
acc_at_COG_opt=pinv(mat)*Rotation_mat*Data.acc_values.';
acc_opt=(pinv(Rotation_mat)*mat*acc_at_COG_opt).';
```

```
%% Plotting measured, estimated and optimized accelreation values

 fig(2)=figure('Name',...
     'comparision of measured,estimated and optimized acceleration values'...
     ,'NumberTitle','off')
for ii=1:numel(Data.sen_no)
    subplot(numel(NODE_uniq),3,ii)
        semilogy(Data.freq,abs(Data.acc_values(:,ii)),'b',...
            Data.freq,abs(acc_est(:,ii)),'r',...
            Data.freq,abs(acc_opt(:,ii)),'g','linewidth',1.5);
    xlim([1000 3000])
    ylim([0.1 5])
    grid
end
mtextbox=uicontrol('style','text','position',[0,0,1500,40]);
set(mtextbox,'string',...
    'Blue - Measured Acc values , Red - Estimated Acc value, Green - Optimized A
clear ii;

%% Validation using gyroscope reading
temp=0;
if temp<1
for i=4:6
if i==4
    fig(2)=figure('Name','Rotation about x axis (Roll)','NumberTitle','off')
elseif i==5
    fig(2)=figure('Name','Rotation about y axis (Pitch)','NumberTitle','off')
else
    fig(2)=figure('Name','Rotation about z axis (Yaw)','NumberTitle','off')
end
 plot(Data.freq,(abs(acc_at_COG_opt(i,:)./max(abs(acc_at_COG_opt(i,:))))))...
     ,'g',Data.freq,(abs(Data.RotY(:,i)'./max(abs(Data.RotY(:,i))))),'K');
  plot(Data.freq,(abs(acc_at_COG_opt(i,:)./max(abs(acc_at_COG_opt(i,:))))))...
     ,'g',Data.freq,(abs(Data.RotY(:,i)'./max(abs(Data.RotY(:,i))))),'K',...
 Data.freq,(abs(acc_at_COG(i,:)'./max(abs(acc_at_COG(i,:))))));
%  plot(Data.freq,(abs(acc_at_COG_opt(i,:)).*9.81)...
%      ,'g',Data.freq,(abs(Data.RotY(:,i)')./(2*pi)),'K');
legend('show');
legend('optimized value accelerometer','gyroscope reading',...
    'estimated from accelerometer');
end
end
clear temp;
clear mtextbox fig;

%% to find global minimum
%{
%  Global Optimization Toolbox needed
gs=GlobalSearch;
```

```
problem=createOptimProblem('fmincon','x0',f,'objective',opt.pass,'lb',...
    opt.lb,'ub',opt.ub);
[xmin,fmin,flag,output,allmins]=run(gs,problem);
%}
```

5.3.2 Rotation Matrix subroutine

Program for forming Rotation Matrix

```
function RR =EULER_ANGLES(Data)

Euler=zeros(size(Data.sen_pos));
% Euler(7,2)=8;
% Euler(7,3)=8;
% Euler(8,3)=8;
% Euler(9,3)=8;
global F;

for i=1:(numel(Data.sen_no)/3)
    for j=1:3
        Euler(i+min(Data.sen_no),j)=Euler(i+min(Data.sen_no),j)+F(i,j);
    end
end

NODE_uniq=unique(Data.sen_no);
RR=[];

for ii=1:numel(NODE_uniq);
    idx=find(Data.set_no==NODE_uniq(ii));
    RR=blkdiag(RR,rotation_TL(Euler(idx,:)));
    ii;
end
end

function [ R ] = rotation_TL( E )
%calculate rotation matrix of euler angles from Test.Lab
%   Detailed explanation goes here
c1=cosd(E(1));
c2=cosd(-E(2));
c3=cosd(E(3));
s1=sind(E(1));
s2=sind(-E(2));
s3=sind(E(3));
%  the transformation matrix is got by multiplying the three transformation
% matrices inertial to rotation in yaw and the rotated to roation to pitch...
% and the rotated to roll
% E1 represent rotation about z axis (yaw)
% E2 represent rotation about y axis (pitch)
% E3 represent rotation about z axis (roll)
R=[c1*c2 c1*s2*s3-c3*s1 s1*s3+c1*c3*s2;...
    c2*s1 c1*c3+s1*s2*s3 c3*s1*s2-c1*s3;...
```

```
    -s2  c2*s3  c2*c3];

end
```

5.3.3 Translation Matrix subroutine

Program for finding the acceleration values at the COG

```
function M = values_at_COG(Data)
G=Data.sen_pos;

clear M;
global F;

for i=1:(numel(Data.sen_no)/3)
  for j=1:3
      G(i+6,j)=G(i+6,j)+F(i,j+3);
  end
end

for ii=1:numel(Data.sen_no)
    idx=find(Data.set_no==Data.sen_no(ii));
    switch Data.direction(ii)
        case 1
            M(ii,:)=[1  0  0  0  G(idx,3)  -G(idx,2)];
        case 2
            M(ii,:)=[0  1  0  -G(idx,3)  0  G(idx,1)];
        case 3
            M(ii,:)=[0  0  1  G(idx,2)  -G(idx,1)  0];
    end
end
end
```

5.3.4 Optimization Routine

Program for finding the error in the angle and distance measurement of the sensor position

```
function [F,fnvalue,opt]=sensoroptim(Data)
NODE_uniq=unique(Data.sen_no);
iteration=5;
fsol=zeros(numel(NODE_uniq),6,iteration);
solval=zeros(iteration,1);
f=zeros(numel(NODE_uniq),6);
stpt=zeros(numel(NODE_uniq),6,iteration);
% Z=zeros(iteration,2);
% Z(:,1)=linspace(-10,10,iteration);
% Z(:,2)=linspace(-0.05,0.05,iteration);
for i=1:iteration
% f(:,1:3)=Z(i,1)*ones(numel(NODE_uniq),3);
% f(:,4:6)=Z(i,2)*ones(numel(NODE_uniq),3);
anglelb=-15;
```

```
angleub=15;
distlb=-0.05;
distub=0.05;
f(:,1:3)=(angleub-anglelb)*rand(5,3)+anglelb*ones(5,3);
f(:,4:6)=(distub-distlb)*rand(5,3)+distlb*ones(5,3);
opt=struct('lb',[],'ub',[],'pass',[],'options',[]);
opt.lb=zeros(numel(NODE_uniq),6);
opt.ub=zeros(numel(NODE_uniq),6);
opt.lb(:,1:3)=anglelb;
opt.lb(:,4:6)=distlb;
opt.ub(:,1:3)=angleub;
opt.ub(:,4:6)=distub;
opt.pass=@(f)optim(f,Data);
nonlcon=[];
opt.options=optimoptions('fmincon','Algorithm',...
    'interior-point','MaxFunEvals',20000,'MaxIter',1000);
% options = optimoptions('fmincon','Algorithm',...
% 'trust-region-reflective','Hessian','user-supplied');
[F1,fnvalue1]=fmincon(opt.pass,f,[],[],[],[],...
    opt.lb,opt.ub,nonlcon,opt.options);
stpt(:,:,i)=f(:,:);
solval(i,1)=fnvalue1;
fsol(:,:,i)=F1;
end
[u,u1]=min(solval);
F(:,:)=fsol(:,:,u1);
fnvalue=u;
end
```

5.3.5 Optimal Sensor Position

Program for finding the Best position for Sensor placement

```
%% finding best location for sensor placement
engine_points=15;
e =zeros(engine_points,6);
temp = points(Data,e,engine_points);
a_global= [1 1 1 1 1 1]';
a_local = temp * a_global;
clear temp;
for i=1:engine_points
    for j=1:6
        if j<=3
        e(i,j)=0.1;
        else
        e(i,j)=1;
        end
temp=points(Data,e,engine_points);
temp1=a_global-(pinv(temp)*a_local);
a_new(i,j)=norm(temp1);
```

```
clear temp1 temp;
e = zeros(engine_points,6);
    end
arrange(i,1) = norm(a_new(i,:));
end
[sensorno(:,1),sensorno(:,2)]=sort(arrange);
clear engine_points i j e a_global a_local arrange a_new file;
```

Subroutine used

```
function new = points(Data,e,engine_points)
G=Data.sen_pos;
%% forming translation matrix
j=1;
M=zeros((engine_points*3),6);
for i=1:engine_points
    M(j,:)   =[1 0 0 0 G(i,3)+e(i,3) -G(i,2)-e(i,2)];
    M(j+1,:)=[0 1 0 -G(i,3)-e(i,3) 0 G(i,1)+e(i,1)];
    M(j+2,:)=[0 0 1 G(i,2)+e(i,2) -G(i,1)-e(i,1) 0];
    j=j+3;
end

%% forming Rotation matrix
Euler =zeros(engine_points,3);
for i=1:engine_points
    for j=1:3
        Euler(i,j)=Euler(i,j)+e(i,(j+3));
    end
end
RR=[];
for i=1:engine_points
    RR=blkdiag(RR,rotation_TL(Euler(i,:)));
end
new=pinv(RR)*M;
end
```