Universidad de Alcalá
Escuela Politécnica Superior

Curso de adaptación al Grado en Ingeniería en Sistemas de Telecomunicación

Trabajo de fin de grado

Improved crash detection system for motorcycles based on neural networks

Autor: César Gay Nieto

Tutor: Sancho Salcedo Sanz

2013

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

CURSO DE ADAPTACIÓN AL GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

Trabajo Fin de Grado

Improved crash detection system for motorcycles based on neural networks

Autor: César Gay Nieto

Director: Sancho Salcedo Sanz

TRIBUNAL:

Presidente: .....

Vocal 1º: .....

Vocal 2º: .....

FECHA: .....

Preface

This Graduate's Project is written on the Course 2012 / 2013 during the Graduate in Telecommunication Systems Engineering Programme. The supervisor for the project is Sancho Salcedo Sanz.

I would like to acknowledge the supervisor for his support throughout the project.

Figures and tables are referred to by the type and number of the object and can be consulted in the sections Figure and Table index, respectively.

Last pages of this report contain references and appendices and the CD can be found on the back page. Appendixes contain information to support elements in the report. Source codes, documentation and other files are included in the CD.

Escuela Politécnica (Universidad de Alcalá), September 2013

César Gay Nieto

Contents

LIST OF FIGURES 3
LIST OF TABLES 7
GUIDELINES 9
SUMMARY 11
1. INTRODUCTION 13
1.1 Background and motivation 13
1.2 Previous work 14
1.3 Contributions of this Report 15
2. ACCIDENT DETECTION 17
2.1 Crash tests 17
2.1.1 First test 18
2.1.2 Second test 19
2.1.3 Third test 21
2.1.4 Fourth test 22
2.2 Common factors in an accident 24
2.2.1 Deceleration 24
2.2.2 Overturning 26
2.2.3 Falling of the driver 27
3. MOTORCYCLE CRASH DETECTION SYSTEM 29
3.1 Moto eCall algorithm 29
3.2 Testing the algorithm 32
3.2.1 Acceleration 34
3.2.2 Deceleration 35
3.2.3 Curve 36
3.2.4 Frontal impact at 120 Km/h 37
3.2.5 Frontal impact at 50 Km/h 38
3.2.6 Rear impact while the motorcycle is stopped 39

3.2.7 Rear impact while the motorcycle in motion 40
3.2.8 Side impact while the motorcycle is stopped 41
3.2.9 Side impact while the motorcycle is in motion 42
3.2.10 Front-side impact in an angle of 30° 43
3.2.11 Front-side impact in an angle of 45° 44
3.2.12 Front-side impact in an angle of 70° 45
3.2.13 Slide side crash 46
3.2.14 Road exit 47
3.3 Discussion of tests results 48
4. IMPROVING THE SYSTEM 49
4.1 Artificial Neural Network 49
4.1.1 What is an artificial neural network? 49
4.1.2 Suitability study 52
4.2 Extreme Learning Machine 54
4.2.1 What is Extreme Learning Machine? 54
4.2.2 ELM Training Algorithm 56
4.2.3 Suitability study 59
4.2.4 Testing the algorithm 60
4.2.5 Discussion of tests results 90
5. CONCLUSIONS 93
6. BUDGET 95
6.1 Material cost 95
6.2 Development cost 95
6.3 Total cost 95
7. REFERENCES 97
APPENDIX A. MOTO eCALL SYSTEM 99

LIST OF FIGURES

Figure 1. Kind of accidents 13
Figure 2. Real crash test 17
Figure 3. Frontal impact at 120 Km/h 18
Figure 4. Frontal impact at 50 Km/h 20
Figure 5. Rear impact while the motorcycle is in motion 21
Figure 6. Rear impact while the motorcycle is stopped 23
Figure 7. Different impacts 25
Figure 8. Dynamic and static forces 26
Figure 9. Passenger and rider sensors location 27
Figure 10. Sensor, gyroscope and accelerometer flow diagrams 30
Figure 11. Main flow diagram 31
Figure 12. The three Cartesian axes on a motorbike 32
Figure 13. First situation, acceleration 34
Figure 14. First situation, global sum 34
Figure 15. Second situation, deceleration 35
Figure 16. Second situation, global sum 35
Figure 17. Third situation, curve 36
Figure 18. Third situation, global sum 36
Figure 19. Fourth situation, frontal impact at 120 Km/h 37
Figure 20. Fourth situation, global sum 37
Figure 21. Fifth situation, frontal impact at 50 Km/h 38
Figure 22. Fifth situation, global sum 38
Figure 23. Sixth situation, rear impact while the motorcycle is stopped 39
Figure 24. Sixth situation, global sum 39
Figure 25. Seventh situation, rear impact while the motorcycle is in motion 40
Figure 26. Seventh situation, global sum 40
Figure 27. Eight situation, side impact while the motorcycle is stopped 41
Figure 28. Eight situation, global sum 41
Figure 29. Ninth situation, side impact while the motorcycle is in motion 42
Figure 30. Ninth situation, global sum 42
Figure 31. Tenth situation, front-side impact in an angle of 30° 43
Figure 32. Tenth situation, global sum 43
Figure 33. Eleventh situation, front-side impact in an angle of 45° 44
Figure 34. Eleventh situation, global sum 44
Figure 35. Twelfth situation, front-side impact in an angle of 70° 45
Figure 36. Twelfth situation, global sum 45
Figure 37. Thirteenth situation, slide side crash 46
Figure 38. Thirteenth situation, global sum 46
Figure 39. Fourteenth situation, road exit 47

Figure 40. Fourteenth situation, global sum 47
Figure 41. Neural network layers 50
Figure 42. Sigmoidal function 57
Figure 43. Hardlim function 57
Figure 44. Sine function 58
Figure 45. Triangular basis function 58
Figure 46. First situation, approx. function='sig' 62
Figure 47. First situation, approx. function='hardlim' 62
Figure 48. First situation, approx. function='sin' 63
Figure 49. First situation, approx. function='tribas' 63
Figure 50. Second situation, approx. function='sig' 64
Figure 51. Second situation, approx. function='hardlim' 64
Figure 52. Second situation, approx. function='sin' 65
Figure 53. Second situation, approx. function='tribas' 65
Figure 54. Third situation, approx. function='sig' 66
Figure 55. Third situation, approx. function='hardlim' 66
Figure 56. Third situation, approx. function='sin' 67
Figure 57. Third situation, approx. function='tribas' 67
Figure 58. Fourth situation, approx. function='sig' 68
Figure 59. Fourth situation, approx. function='hardlim' 68
Figure 60. Fourth situation, approx. function='sin' 69
Figure 61. Fourth situation, approx. function='tribas' 69
Figure 62. Fifth situation, approx. function='sig' 70
Figure 63. Fifth situation, approx. function='hardlim' 70
Figure 64. Fifth situation, approx. function='sin' 71
Figure 65. Fifth situation, approx. function='tribas' 71
Figure 66. Sixth situation, approx. function='sig' 72
Figure 67. Sixth situation, approx. function='hardlim' 72
Figure 68. Sixth situation, approx. function='sin' 73
Figure 69. Sixth situation, approx. function='tribas' 73
Figure 70. Seventh situation, approx. function='sig' 74
Figure 71. Seventh situation, approx. function='hardlim' 74
Figure 72. Seventh situation, approx. function='sin' 75
Figure 73. Seventh situation, approx. function='tribas' 75
Figure 74. Eighth situation, approx. function='sig' 76
Figure 75. Eighth situation, approx. function='hardlim' 76
Figure 76. Eighth situation, approx. function='sin' 77
Figure 77. Eighth situation, approx. function='tribas' 77
Figure 78. Ninth situation, approx. function='sig' 78
Figure 79. Ninth situation, approx. function='hardlim' 78
Figure 80. Ninth situation, approx. function='sin' 79
Figure 81. Ninth situation, approx. function='tribas' 79
Figure 82. Tenth situation, approx. function='sig' 80
Figure 83. Tenth situation, approx. function='hardlim' 80
Figure 84. Tenth situation, approx. function='sin' 81
Figure 85. Tenth situation, approx. function='tribas' 81
Figure 86. Eleventh situation, approx. function='sig' 82

Figure 87. Eleventh situation, approx. function='hardlim' 82
Figure 88. Eleventh situation, approx. function='sin' 83
Figure 89. Eleventh situation, approx. function='tribas' 83
Figure 90. Twelfth situation, approx. function='sig' 84
Figure 91. Twelfth situation, approx. function='hardlim' 84
Figure 92. Twelfth situation, approx. function='sin' 85
Figure 93. Twelfth situation, approx. function='tribas' 85
Figure 94. Thirteenth situation, approx. function='sig' 86
Figure 95. Thirteenth situation, approx. function='hardlim' 86
Figure 96. Thirteenth situation, approx. function='sin' 87
Figure 97. Thirteenth situation, approx. function='tribas' 87
Figure 98. Fourteenth situation, approx. function='sig' 88
Figure 99. Fourteenth situation, approx. function='hardlim' 88
Figure 100. Fourteenth situation, approx. function='sin' 89
Figure 101. Fourteenth situation, approx. function='tribas' 89

LIST OF TABLES

Table 1. Description of first crash test 18
Table 2. Results of first test 19
Table 3. Description of second crash test 19
Table 4. Results of second test 20
Table 5. Description of third crash test 21
Table 6. Results of third test 22
Table 7. Description of fourth crash test 22
Table 8. Results of fourth test 23
Table 9. Moto eCall test results 48
Table 10. ANN suitability study 54
Table 11. ELM suitability study 60
Table 12. Extreme Learning Machine test results 91
Table 13. Material cost 95
Table 14. Development cost 95
Table 15. Total cost 95

GUIDELINES

This section presents the project structure and explains the different areas to give the reader a good understanding of the subject.

- **Summary**
  Here an abstract (written on English and Spanish) and a extended summary of the report can be found. Its purpose is to summarise the project, identify what was achieved, the major changes that happened and what is still outstanding. Besides, this section also includes Keywords contains an extended summary of the project.

- **Introduction**
  This area provides details about the subject and purpose of the project. It includes the background, motivation for writing the report, previous work and contributions of the document.

- **Accident detection**
  Crash tests simulations made by "Working model" program and its results are shown in this section. Besides, the common factors in a motorcycle accident are explained.

- **Motorcycle crash detection system**
  The aforementioned results were used as a baseline for the elaboration of an algorithm capable of detecting crashes. This sections explains the current motorcycle crash detection system. Furthermore, an intensive analysis of the system has been develop in order to find errors and performance issues.

- **Improving the system**
  Here the possible improvements of the current system are explained and analysed. This section includes a deep study of Artificial Neural Networks and the new learning technique referred to as Extreme Learning Machine.

- **Conclusions**
  Here a general conclusion of the project work is presented. In addition, the initial goals are compared with those obtained according to the results that are mentioned in the earlier sections of the report. Besides, it is included some suggestions to improvements identified throughout the work.

- **Budget**
  Here it is presented the total budget of the project, where tools utilized (both, software and hardware) and labor cost are included.

- **References**
  All references of the document and the content of the attached CD-ROM are explained in this section.

SUMMARY

This project optimizes the accident detection mechanism of the system called "Moto eCall", developed in June 2011 at the Engineering College of Aarhus (Denmark). After a deep study of different solutions, a feedforward artificial neural network (ANN) trained the new learning technique referred to as Extreme Learning Machine (ELM) is proposed.

Among others, ANNs present numerous advantages such as adaptive learning, self-organization, failover and real-time operation that make them a suitable alternative to the current crash detection technique. Besides, ELM algorithm tends to provide the best generalization performance at extremely fast learning speed. Therefore, ANNs and ELM are the perfect combination to optimize Moto eCall performance.

Thanks to crash test developed with the simulation program called Working Model, the previously mentioned combination has been tested. The results show an optimal performance due to ANN reacts almost immediately when a crash happens and also ELM training and predicting times are shorter than currently algorithm. Consequently, it is confirmed that the proposal improves the weaknesses of the actual motorcycle crash detection system.

Este proyecto optimiza el mecanismo de detección de accidentes del sistema denominado "Moto eCall", finalizado en Junio del año 2011 en el Colegio de Ingeniería de Aarhus (Dinamarca). Para ello, y después de un largo proceso de búsqueda de soluciones, se propone el uso de una red neuronal artificial (RNA) de propagación hacia delante junto con la nueva técnica de aprendizaje denominada ELM (Extreme Learning Machine).

Las RNAs ofrecen numerosas ventajas como aprendizaje adaptativo, auto-organización, tolerancia a fallos y operación en tiempo real. Además, el algoritmo ELM proporciona el mejor rendimiento de generalización a una velocidad de aprendizaje muy elevada. Con todo esto, las RNAs y el algoritmo de aprendizaje ELM son la combinación perfecta para optimizar el rendimiento de Moto eCall.

Gracias a los "crash test" realizados con el programa de simulación llamado "Working Model", la susodicha combinación ha sido probada. Los resultados muestran un rendimiento óptimo ya que la red neuronal reacciona casi inmediatamente cuando se produce un accidente y, además, los tiempos de entrenamiento y de previsión son más pequeños que los del algoritmo actual. Por lo tanto, se confirma que la propuesta de este proyecto mejora los puntos débiles del sistema actual de detección de accidentes.

Keywords: Motorcycle Emergency Call; Feedforward Artificial Neural Network; Extreme Learning Machine; Crash Tests; Working Model: Matlab

1. INTRODUCTION

This section provides details about the subject and purpose of the project. It includes the background, motivation for writing the report, previous work and contributions of the document.

1.1 Background and motivation

The human being has always been characterized by constantly seeking new ways to improve their living conditions. Thus, work where force plays a major role can be reduced and lead efforts to other fields, such as creating electronic calculators or computers that allow easily implement algorithms and solve many problems that were previously difficult to develop by hand.

However, these machines have limitations to face the problems that do not support algorithmic treatment. If these issues are carefully considered, it will be observed something in common, the experience. Man is capable of resolving these situations thanks to the experience gained.

For this reason artificial intelligence arises, a discipline that try to discover aspects of human intelligence that can be simulated by machines. Thus, neural networks emulate certain human characteristics, such as the ability to memorize and associate facts. Summarizing, they are an artificial and simplified model of the human brain, which is the most perfect example available for a system capable of acquiring knowledge through experience.

Neural networks can be used in a large number and variety of applications in diverse fields such as biology, economics, finance or medicine. The objective of this project is to develop an algorithm capable of detecting motorcycle accidents by using neural networks so that could be used in an emergency in these vehicles.

Among others, a recent study of the Winterthur Foundation (in collaboration with the Traffic Institute at the University of Valencia - INTRAS) analyzes the causes, its consequences and possible solutions to improve safety of motorcycle accidents in Spain. According to this, the motorbike accident rate has declined over the last decade by 22%, although, proportionally, their mortality remains ten times higher than cars.

In 52% of cases the accident occurred against a car, mostly frontal and front-lateral collisions. However, although there were many fewer accidents alone, the death toll is quite high. More than half of sinister are between a motorcycle and a car, being the infringer the driver of tourism, especially for not respecting the priority and invading the opposite direction. Following figure shows statistics about the kind of accident obtained from this study (for further information, the document is attached on the CD-ROM).

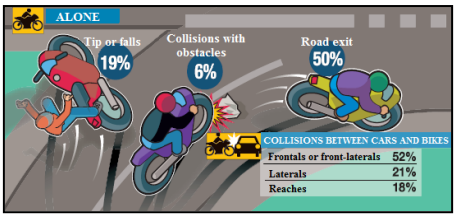


Figure 1. Kind of accidents

1.2 Previous work

This project will develop an Artificial Neural Network (ANN) capable of detecting motorcycle accidents. The objective is to optimize the crash detection mechanism of the project called "Moto eCall", made during the course 2010/2011 at the Engineering College of Aarhus (Denmark).

The idea of "Motorcycle Emergency Call" emerged with the news that European Commission expected the implementation of a project called "eCall" by 2014. This project, which is also supported by the European Automobile Manufacturers Association (ACEA), pretends to bring rapid assistance to drivers involved in a crash anywhere in the European Union.

"eCall" sends airbag deployment, sensor information and GPS coordinates to local emergency agencies and directly establishes a voice connection with the relevant Public Service Answering Point (PSAP) in order to reduce the response time of emergency services. Therefore, it is possible to save more lives or reduce the effects of injuries by increasing the rapidity which rescue services are mobilized.

Thus, "Moto eCall" is a hardware prototype installed on the bike able to send key data (exact time, g forces, location and number of passengers) via SMS to local emergency services (E112) if an accident happens. It consists of an accelerometer (which gives information about the acceleration undergone by the bike), a gyroscope (that sends the rate of change of motorcycle tilt), sensors in handlebars and brackets (which give information about the rider and passenger) and a GPS receiver (that provides the coordinates of its position), coordinated by a microcontroller. Besides, the message can be generated manually by the rider or automatically if a crash is detected by sensors.

During "Moto eCall" development, the following activities related to the crash detection algorithm were developed:

- **Study of real accidents**
  A deep research and study were made about the sequence of events that takes place in an accident. In order to do this and since there were no adequate means to perform crash tests, simulations were made by using "Working model" program. The results were used as a baseline for the elaboration of an algorithm capable of detecting crashes.

- **Crash detection system**
  Thanks to these virtual crash tests, an algorithm which tries to represent the probability that a crash happens could be developed. This algorithm assigned values to some variables according to data received every millisecond. Then, these variables were summed and added to a global sum. If this global sum reached a certain level, it meant that an accident had occurred and the algorithm proceeded to send a message to emergency services.

- **Possibility to change decision levels**
  All levels and variables were used due to the results obtained from "Working Model" simulation program. Therefore, if real measurements are carried out, these decision levels, which are stored in memory, can be easily changed.

1.3 Contributions of this Report

As said before, this project pretends to optimize the crash detection mechanism of the project called "Moto eCall" which was developed in the course 2010/2011 at the Engineering College of Aarhus (Denmark). Here is a list of upgrades this project offers:

- **Study and analysis of current crash detection system**
  An intensive analysis of the current algorithm has been developed in order to find errors and performance issues. Furthermore, this report tries to find possibilities to increase system efficiency, among which artificial neural networks (ANN) stands out.

- **Development and programming of an ANN capable of detecting accidents**
  Thanks to its special features, neural networks are the best option to increase Moto eCall performance, since they are inspired in the interconnection of the neurons in the nervous systems of the human brain. ANNs have a number of properties that make them an attractive alternative to traditional problem-solving techniques. Among others, they present numerous advantages such as adaptive learning, self-organization, failover, real-time operation and easy to integrate into existing technology.

- **Application of the new learning technique referred to as ELM (Extreme Learning Machine)**
  The ELM modelling scheme is a new framework unlike the standard neural network, it is a Single-hidden Layer Feedforward neural Networks (SLFNs) which randomly chooses the input weights and analytically determines the output weights of SLFNs. This algorithm tends to provide the best generalization performance at extremely fast learning speed. Also ELM tends to reach the minimum training error as well as it considers magnitude of weights which is opposite t the classic gradient-based learning algorithms which only intend to reach minimum training error but do not consider the magnitude of weights.

- **Development of a database to setup the neural network**
  With the data taken from the simulation program called "Working Model", it has been created a complete database to train the ANN and test the ELM algorithm.

Furthermore, the proposed project involves different specialties and application fields which are listed below.

- **Analysis**
  - Study of current crash detection system
- **Artificial Intelligence**
  - Development of an Artificial Neural Networks
  - Extreme Learning Machine algorithm
- **Software**
  - Simulation of ELM algorithm carried out in Matlab environment
  - Development of a database to setup the neural network
- **Road safety**
  - Reduction of deaths in crashes

2. ACCIDENT DETECTION

Motorcycle Emergency Call (Moto eCall) is a system divided into three distinct blocks: accident detection, decision-making and emergency services warning. The report is focused on first and second parts, the third one is out of scope. This section details the aforementioned stage "accident detention", and it is divided into two subsections:

- **Crash tests:** Here it is explained crash tests simulations made with program called Working model and the results are shown.
- **Common factors in an accident:** Here, the common factors that take place in an accident are explained in detail.

For further information regarding the second block "decision-making", please take a look at section "3.Motorcycle crash detection system". More details related to third block and Moto eCall architecture can be found in *Apendix A.Moto eCall System* at the end of the report.

2.1 Crash tests

It is essential to understand the behaviour of a motorcycle under a crash situation, as well as G-force levels involved. For this purpose, so called crash test are necessary. A crash test is a form of destructive testing performed in order to ensure safe design standards in crashworthiness and crash compatibility for various models of transportation and related systems and components.

Since there are no means or facilities to carry out crash tests in real conditions, the solution is to reproduce them using the simulation program called Working Model. Working Model is a conceptual design tool that allows professional engineers to create and analyze real-life mechanical systems. Among others, the capabilities of this program are the following:

- Fast “run-analyze-refine” cycle helps to optimize designs before building physical prototypes
- Quickly build, run, and refine simulations with pre-defined objects and constraints
- Run, stop, reset, single step, or pause the simulation at any time
- View output as vectors or in numbers and graphs in English or metric units
- Import the 2D CAD drawings in DXF format
- Input values from equations, sliders and DDE links to MATLAB and Excel
- Create bodies and specify its mass properties, initial velocity, electrostatic charge, etc.
- Simulate contact, collisions, and friction

Following figure shows the sequence of images of a real crash test.

Figure 2. Real crash test

This chapter treats about the four crash tests developed which will provide essential information such as: the sequence of events that takes place in an accident, kind of forces involved at the moment of the crash and the forces level experienced by the motorcycle and rider. The aforementioned crash test are explained below:

- **First test:** Frontal-impact upon a solid wall at a initial speed of 120 km/h
- **Second test:** Frontal-impact upon a solid wall at a initial speed of 50 km/h
- **Third test:** Impact of a car on a motorcycle in motion at a initial speed of 112 km/h
- **Fourth test:** Impact of a car on a motorcycle stopped at a initial speed of 120 km/h

2.1.1 First test

Table below describes the first crash test developed.

Description	Impact upon a solid wall at a initial speed of 120 km/h		
Kind of impact	Frontal-impact	**Initial speed (m/s)**	33.30
Motorcycle mass (kg)	150	**Speed at which impact occurs (m/s)**	18.87
Ryder mass (kg)	79	**Kind of surface**	Asphalt
Distance between motorcycle and wall (m)	13	**Test duration (s)**	2.50

Table 1. Description of first crash test

Next figure illustrates the initial position and the moment of the impact against the wall.

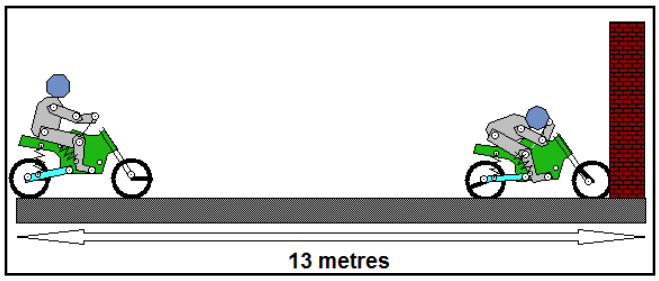


Figure 3. Frontal impact at 120 Km/h

Three parameters (position, speed and acceleration) are measured in motorcycle's body with a precision of 1 millisecond and the results, are shown below (for further information, please consult the document "crash situations.xls" included in the CD).

t (s)	Px (m)	Sx (m/s)	Ax (m/s^2)	Ax (G)	Pz (m)	Sz (m/s)	Az (m/s^2)	Az (G)
0.44	11.98	32.44	81.26	8.28	0.17	0.45	-28.67	-2.92
0.44	12.01	32.43	67.12	6.84	0.17	0.44	-25.94	-2.64
0.44	12.04	31.26	-4179.84	-426.1	0.17	0.84	1395.92	142.30
0.45	12.07	28.95	-2065.59	-210.6	0.17	0.72	202.33	20.62
0.45	12.1	26.27	-2440.87	-248.8	0.17	-1.29	-461.55	-47.05
0.45	12.13	24.41	-1399.07	-142.6	0.17	-1.68	-317.63	-32.38
0.45	12.15	23.32	-839.48	-85.57	0.16	-1.92	-178.44	-18.19
0.45	12.17	22.66	-514.54	-52.45	0.16	-2.06	-94.94	-9.68
0.45	12.2	22.25	-318.11	-32.43	0.16	-2.12	-39.62	-4.04
0.45	12.22	21.79	-246.75	-25.15	0.16	-1.4	747.92	76.24
0.45	12.24	20.43	-1832.6	-186.8	0.16	-0.53	901.68	91.91
0.45	12.26	18.74	-782.93	-79.81	0.16	0.25	340.97	34.76
0.45	12.28	17.53	-1077.23	-109.8	0.16	0.99	691.19	70.46
0.46	12.3	16.46	-1038.57	-105.9	0.16	1.65	603.95	61.56
0.46	12.31	15.46	-945.71	-96.40	0.16	2.24	583.98	59.53
0.46	12.33	14.54	-901.91	-91.94	0.16	2.8	523.59	53.37
0.46	12.34	13.68	-815.84	-83.16	0.17	3.31	514.52	52.45
0.46	12.35	12.89	-764.79	-77.96	0.17	3.8	458.63	46.75
0.46	12.37	12.17	-676.42	-68.95	0.17	4.25	444.93	45.35
0.46	12.38	11.52	-627.71	-63.99	0.18	4.67	391.04	39.86

Table 2. Results of first test

Note: *positive z-axis on table above corresponds to negative z-axis on figure in section "3.2. Testing the algorithm" and vice versa.*

In table above, red row shows the collision instant, where the body reaches an acceleration of -426.1 G in x-axis. The next 17 samples, acceleration is between -25.15 and -248.8 G.

2.1.2 Second test

Next table explains in detail the second crash test developed.

Description	Impact upon a solid wall at a initial speed of 50 km/h		
Kind of impact	Frontal-impact	**Initial speed (m/s)**	13.89
Motorcycle mass (kg)	150	**Speed at which impact occurs (m/s)**	10.16
Ryder mass (kg)	79	**Kind of surface**	Asphalt
Distance between motorcycle and wall (m)	13	**Test duration (s)**	2.50

Table 3. Description of second crash test

In the following figure can be seen the initial position and the moment of the impact against the wall.

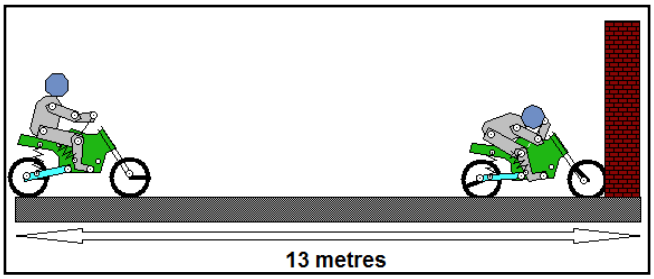


Figure 4. Frontal impact at 50 Km/h

Three parameters (position, speed and acceleration) are measured in motorcycle's body and the results are shown below (for further information, please refer to document "crash situations.xls" included in the CD).

t (s)	Px (m)	Sx (m/s)	Ax (m/s^2)	Ax (G)	Pz (m)	Sz (m/s)	Az (m/s^2)	Az (G)
1.05	11.78	13.48	0.55	0.06	0.05	0.01	-3.08	-0.31
1.05	11.79	13.49	-53.86	-5.49	0.05	0.01	-43.16	-4.40
1.05	11.81	11.84	-1319.34	-134.5	0.05	-0.25	-94.24	-9.61
1.06	11.82	10.65	-1050.58	-107.1	0.05	-0.41	-191.98	-19.57
1.06	11.83	9.75	-755.38	-77	0.05	-0.59	-152.16	-15.51
1.06	11.84	9.13	-369.85	-37.70	0.05	-0.71	177.78	18.12
1.06	11.84	8.21	-285.61	-29.11	0.05	-0.23	193.96	19.77
1.06	11.85	7.99	-587.09	-59.85	0.05	-0.25	200.92	20.48
1.06	11.86	7.47	-511.49	-52.14	0.05	-0.10	180.60	18.41
1.06	11.87	7.02	-473.45	-48.26	0.05	0.03	180.25	18.37
1.06	11.88	6.59	-442.33	-45.09	0.05	0.15	166.74	17.00
1.06	11.88	6.19	-370.13	-37.73	0.05	0.29	103.25	10.52
1.06	11.89	5.83	-403.53	-41.13	0.05	0.37	179.74	18.32
1.07	11.89	5.44	-387.12	-39.46	0.05	0.53	118.67	12.10
1.07	11.90	5.08	-332.06	-33.85	0.05	0.65	151.86	15.48
1.07	11.90	4.75	-336.45	-34.30	0.05	0.79	95.39	9.72
1.07	11.91	4.43	-285.13	-29.07	0.05	0.89	137.00	13.97
1.07	11.91	4.15	-294.52	-30.02	0.05	1.01	76.99	7.85
1.07	11.92	3.87	-240.09	-24.47	0.06	1.10	123.88	12.63
1.07	11.92	3.64	-254.25	-25.92	0.06	1.21	58.65	5.98

Table 4. Results of second test

Note: *positive z-axis on table above corresponds to negative z-axis on figure in section "3.2. Testing the algorithm" and vice versa.*

Above, the red row shows the collision instant, where the body reaches an acceleration of -134.5 G in x-axis. The next 17 samples, acceleration is between -25.92 and -107.1 G in x-axis.

2.1.3 Third test

Following table describes the third crash test developed.

Description	Impact of a car on a motorcycle in motion at a initial speed of 112 km/h			
Kind of impact	Rear-impact	**Distance between car and motorcycle (m)**	12	
Motorcycle mass (kg)	150	**Initial speed (m/s)**	**Car**	31
			Motorcycle	8
Ryder mass (kg)	79	**Speed at which impact occurs (m/s)**	**Car**	25.45
			Motorcycle	7.79
Car mass (kg)	510	**Kind of surface**	Asphalt	
		Test duration (s)	2.50	

Table 5. Description of third crash test

In the following figure can be seen the initial position of the motorcycle and the car.

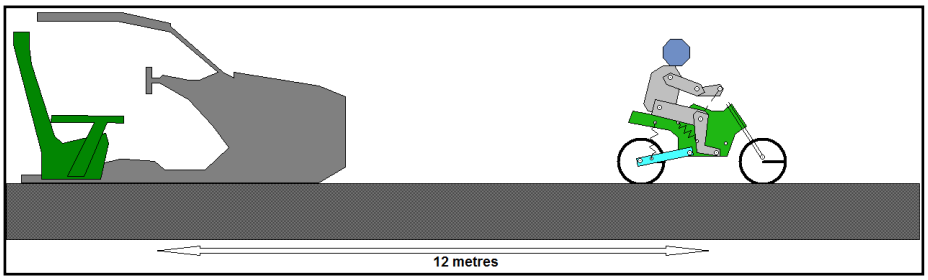


Figure 5. Rear impact while the motorcycle is in motion

Three parameters (position, speed and acceleration) are measured in motorcycle's body and the results are shown below (for further information, consult the document "crash situations.xls" included in the CD).

t (s)	**Px (m)**	**Sx (m/s)**	**Ax (m/s^2)**	**Ax (G)**	**Pz (m)**	**Sz (m/s)**	**Az (m/s^2)**	**Az (G)**
0.33	20.09	7.67	-0.16	-0.02	0.19	-0.46	-2.27	-0.23
0.33	20.1	7.67	82.86	8.45	0.19	-0.46	6.64	0.68
0.33	20.11	11.63	899.06	91.65	0.19	-0.6	1471.09	149.96
0.33	20.12	13.1	1408.62	143.59	0.19	0.62	1128.76	115.06
0.33	20.13	14.43	1243	126.71	0.19	1.66	946.87	96.52
0.33	20.15	15.47	832.21	84.83	0.19	2.55	843.05	85.94
0.33	20.17	16.25	792.02	80.74	0.2	3.3	632.94	64.52

t (s)	Px (m)	Sx (m/s)	Ax (m/s^2)	Ax (G)	Pz (m)	Sz (m/s)	Az (m/s^2)	Az (G)
0.33	20.18	16.89	469.59	47.87	0.2	3.9	567.75	57.87
0.33	20.2	17.36	534.37	54.47	0.2	4.39	398.92	40.66
0.33	20.22	17.76	264.96	27.01	0.21	4.77	378.16	38.55
0.34	20.23	18.04	357.93	36.49	0.21	5.1	253.75	25.87
0.34	20.25	18.29	119.74	12.21	0.22	5.35	261.73	26.68
0.34	20.27	18.45	259.45	26.45	0.22	5.57	155.65	15.87
0.34	20.29	18.6	37.36	3.81	0.23	5.74	182.47	18.60
0.34	20.31	18.69	190.09	19.38	0.24	5.88	94.15	9.60
0.34	20.33	18.78	-17.49	-1.78	0.24	5.99	131.1	13.36
0.34	20.35	18.82	139.88	14.26	0.25	6.09	56.79	5.79
0.34	20.36	18.86	-65.54	-6.68	0.25	6.17	100.48	10.24
0.34	20.38	18.87	111.5	11.37	0.26	6.24	30.36	3.09
0.34	20.4	18.88	-68.15	-6.95	0.27	6.29	78.76	8.03

Table 6. Results of third test

Note: *positive z-axis on table above corresponds to negative z-axis on figure in section "3.2. Testing the algorithm" and vice versa.*

In table above, red row shows the collision instant, where the body reaches an acceleration of 91.65 G in x-axis. The next 8 samples, acceleration is between 27.01 and 143.59 G in x-axis.

2.1.4 Fourth test

Next table explains the fourth crash test developed.

Description	Impact of a car on a motorcycle stopped at a initial speed of 120 km/h			
Kind of impact	Rear-impact	**Distance between car and motorcycle (m)**	12	
Motorcycle mass (kg)	150	**Initial speed (m/s)**	**Car**	31
			Motorcycle	0
Ryder mass (kg)	79	**Speed at which impact occurs (m/s)**	**Car**	16.06
			Motorcycle	0
Car mass (kg)	510	**Kind of surface**	Asphalt	
		Test duration (s)	2.50	

Table 7. Description of fourth crash test

Next figure illustrates the initial position and the moment of the impact against the wall.

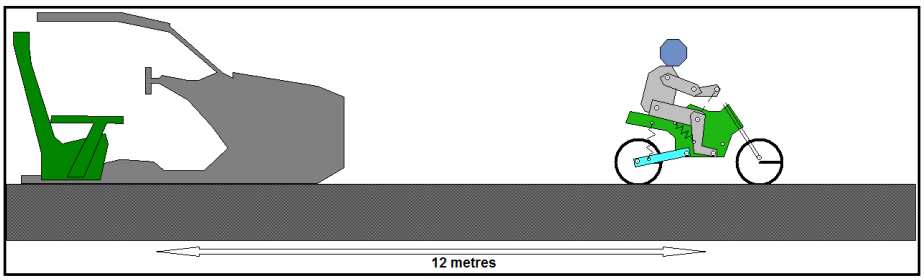


Figure 6. Rear impact while the motorcycle is stopped

Three parameters (position, speed and acceleration) are measured in motorcycle's body and the results are shown below (for further details, consult the document "crash situations.xls" included in the CD).

t (s)	Px (m)	Sx (m/s)	Ax (m/s^2)	Ax (G)	Pz (m)	Sz (m/s)	Az (m/s^2)	Az (G)
0.22	17.54	-0.07	-6.25	-0.64	0.21	-0.22	1.94	0.20
0.22	17.54	-0.02	-30.87	-3.15	0.21	-0.2	45.66	4.65
0.22	17.54	7.99	2319.67	236.46	0.21	-0.03	2150.84	219.25
0.22	17.55	9.79	1520.21	154.97	0.21	1.96	1841.12	187.68
0.22	17.56	11.19	1415.86	144.33	0.21	3.6	1415.54	144.30
0.22	17.58	12.37	621.64	63.37	0.21	4.91	1340.78	136.67
0.23	17.59	13.25	816.51	83.23	0.22	6	971.81	99.06
0.23	17.6	13.97	657.7	67.04	0.23	6.88	788.05	80.33
0.23	17.62	14.54	492.05	50.16	0.23	7.6	658.55	67.13
0.23	17.63	14.98	392.82	40.04	0.24	8.2	541.58	55.21
0.23	17.65	15.31	284.79	29.03	0.25	8.7	455.95	46.48
0.23	17.66	15.56	211.27	21.54	0.26	9.12	381.48	38.89
0.23	17.68	15.71	589.15	60.06	0.27	9.48	190.35	19.40
0.23	17.69	15.85	76.6	7.81	0.28	9.77	272.91	27.82
0.23	17.71	15.92	-20.37	-2.08	0.29	10.02	251.77	25.66
0.23	17.73	15.93	89.96	9.17	0.3	10.24	177.54	18.10
0.24	17.74	15.93	-130.34	-13.29	0.31	10.43	203.32	20.73
0.24	17.76	15.87	30.58	3.12	0.32	10.59	128.48	13.10
0.24	17.77	15.81	-169.79	-17.31	0.33	10.73	165.38	16.86
0.24	17.79	15.7	-24.2	-2.47	0.34	10.87	95.74	9.76

Table 8. Results of fourth test

Note: *positive z-axis on table above corresponds to negative z-axis on figure in section "3.2. Testing the algorithm" and vice versa.*

Above, red row shows the collision instant, where the body reaches an acceleration of 236.46 G in x-axis. The following 10 samples, acceleration is between 21.54 and 154.96 G in x-axis.

2.2 Common factors in an accident

After a deep research and study about motorcycle dynamics, kinematics and its motion, the previously-explained crash tests have been performed. Thanks to these studies and simulations made with the program called Working model, the common variables in an accident has been defined. Although there are no definite sequences of events in an accident, because each crash is unique, there are at least three common factors that happen in most accidents. These factors are a strong deceleration of the body, overturning of bike and the falling of the driver. These common factors are explained below.

2.2.1 Deceleration

As said before, when an accident occurs the motorcycle suffers a strong deceleration, which is usually expressed in G-forces. The G-force associated with an object is its acceleration relative to free fall and is expressed by the following equation:

$$\mathrm{a(G)} = \frac{\mathrm{a(m/s^2)}}{9.81(\mathrm{m/s^2})}$$

The device able to measure G-forces is called accelerometer, which outputs a voltage proportional to the speed change with time (acceleration) suffered. Conceptually, an accelerometer behaves as a damped mass on a spring, when it experiences an acceleration, the mass is displaced to the point that the spring is able to accelerate the mass at the same rate as the casing. Then, the movement is measured to give the acceleration.

Now, it is necessary to know the acceleration levels involved in an accident. For this purpose, so called crash tests are carried out. A crash test is a form of destructive testing which provides real measurements of the deceleration suffered by the driver and the motorcycle. Besides, driver trajectory is analyzed to introduce safety improvements in vehicles.

Since there are not enough facilities to carry out crash tests in real conditions, the solution is to simulate them using the simulation program called "Working Model". Thus, all the forces involved in an accident can be calculated (the four crash tests simulated are explained in section *2.1.Crash tests*).

Assuming the previous experiments as certain, with a sampling frequency of 1 kHz, it is observed that at the collision instant G-forces increase considerably (reaching values above 200 G in some cases) and, during at least 8 samples later, are still very high. For this reason, it is indispensable to define a threshold that will vary depending on each axis.

On one hand, as said before, frontal and rear impacts are related to x-axis. Therefore, after the experiments, the threshold will be located at 20 G. On the other hand, side crashes are connected to y-axis. This axis must take into account the stability and the tilt of the motorcycle, in addition to the referred side impacts. "Working model" cannot simulate this kind of impact, therefore, by estimating, the threshold will be 15 G.

Each type of crash is defined by its threshold, axis and number of consecutive samples (if sampling frequency is 1 kHz):

- **Frontal impact**
  Acceleration is greater than 20 G during at least 8 samples in the negative x- axis (shown in yellow in figure below).

- **Rear impact**
  Acceleration is greater than 20 G during at least 8 samples in positive x- axis (shown in red in the following figure).

- **Side impact**
  Acceleration is greater than 15 G during at least 8 samples in y- axis (in blue in next figure).

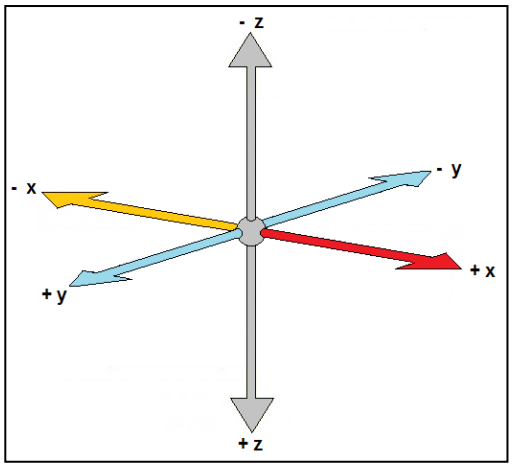


Figure 7. Different impacts

Airbag sensors currently used in vehicles are one axis accelerometers capable of measuring around ±70 G in longitudinal and transversal axes. However, since the test will not be performing due to the lack of resources, the final solution is to use a three axis accelerometer which makes easier the integration in the board.

Besides, some factors such as vibrations or motorcycle internal temperature should be taken into account, because they can alter the measurement and some calibrations would be needed. Owing to our limitations previously discussed, such calibrations are obviated since it are laboratory simulations.

Thereby, the third axis added to the system in first approach is not used, but it can provide useful information in the future.

For further information, please refer to *Apendix A.Moto eCall System* at the end of the report.

2.2.2 Overturning

In the majority of motorcycle accidents the result is that the vehicle lies on the road. The reason for this is the balance of the bike. As mentioned earlier, in an accident there is a strong deceleration, so that the speed decreases drastically. At low speeds, two-wheeled vehicles loose balance if the centre of gravity shifts from the line of the wheels and, therefore, the bike tends to fall.

In order to determine if the bike has overturned, it is necessary to know the motorcycle tilt degree. The first approach to measure this angle has been the use of the third axis of the accelerometer. By using the gravity force as reference, tilt can be calculated using basic geometry. Next figure illustrates both dynamic and static forces.

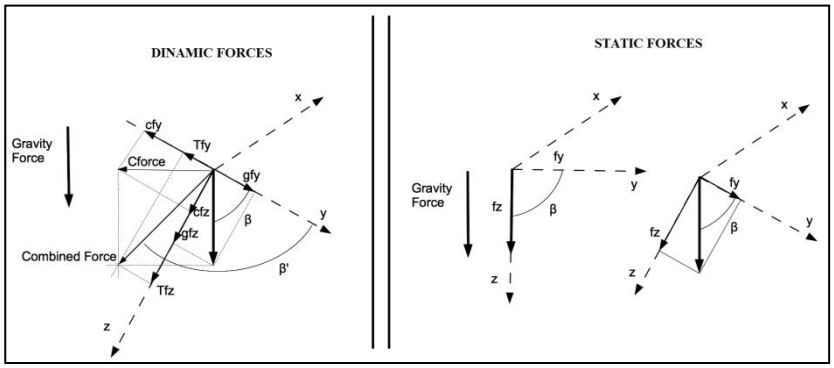


Figure 8. Dynamic and static forces

The problem of this approach is the other forces presents in the system when the motorbike is running, such as centrifugal force in a turn. Therefore, this method is only valid while the motorbike is stopped.

The next idea to obtain tilt angle has been the use of a device called gyroscope. The gyroscopes available in the market give information about the angular speed of the instrument, which unit of measure is called dps (degrees per second). Thus, the angle could be obtained by making the integral over time.

As it is said before, in real conditions some factors such as vibration or motorcycle internal temperature must be taken into account, because they can alter the measurement and some calibrations would be necessary. Because of our restrictions previously discussed, such calibrations are obviated since it is a laboratory simulation.

For further information, please refer to *Apendix A.Moto eCall System* at the end of the report.

2.2.3 Falling of the driver

Besides the two common factors explained above, there is a third one which consists in the falling of the driver. On the one hand, due to the deceleration experienced by the body in an accident, the driver will be violently thrown from the bike. On the other hand, if the rider loses bike control and overturns, the driver will inevitably drop handlebars. Therefore, in any type of accident the probabilities that the rider hold on to the handlebars are virtually nil.

There are bikes that have a seat for a passenger, so that it would be interesting to know if this seat is taken. The information about number of people on the vehicle is vital because if there is an accident, emergency services will send one or two ambulances.

For driver and passenger detection, it has been thought about some sensors that, in a real implementation, would be placed in key parts of the bike such as handlebars and brackets. Therefore, there are four sensors for detecting the rider and two more for passenger detection, as seen in the following figure.

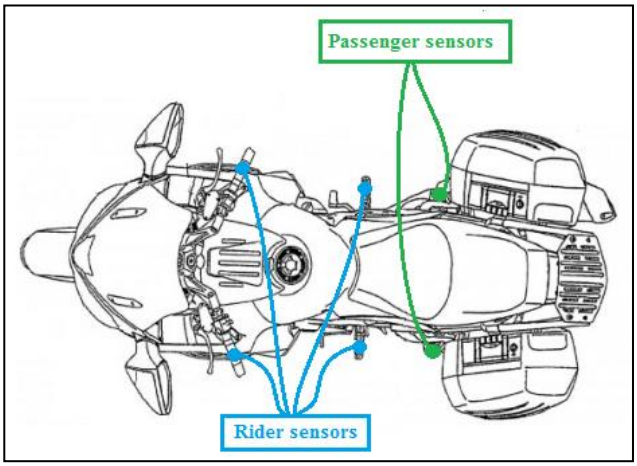


Figure 9. Passenger and rider sensors location

In Moto eCall prototype, as there was no real motorcycle, six switches were acting as sensors. So that, when the system starts, it checks how many of them are asset in order to determine the number of people on the bike. Thus, if an accident occurs, 6 sensors will jump abruptly.

The problem of these sensors is given in cities, the driver leans on the road with one foot (or even two) when he is stopped in a traffic light and he can drop the handlebars. So, unless the bike has a speed of 5 km / h, information send by sensors will not be taken into account.

For further information, please refer to *Apendix A.Moto eCall System* at the end of the report.

3. MOTORCYCLE CRASH DETECTION SYSTEM

Motorcycle Emergency Call (Moto eCall) is a system divided into three distinct blocks: accident detection, decision-making and emergency services warning. The report is focused on first and second parts, the third one is out of scope. This section details the aforementioned stage "decision-making", and it is divided into two subsections:

- **Moto eCall algorithm:** Here it is explained the current crash detection system
- **Testing the algorithm:** Moto eCall system is analysed in order to find errors and performance issues.

For further information regarding the first block "accident detection", please refer to section *2.Accident detention*. More details related to third block and Moto eCall architecture can be found in *Apendix A.Moto eCall System* at the end of the report.

3.1 Moto eCall algorithm

As said before, there are three common factors in a accident: a sudden deceleration, overturning of the bike and the falling of the rider. In most accidents, those common factors will be present but not at the same time because they will probably be consecutively. I.e. if a frontal crash occurs, firstly the accelerometer will suffer a sudden and drastic deceleration; secondly the sensors will jump because the driver will be thrown out and thirdly the bike will finally overturn.

Three bands have been set (whose limits are determined in section *2.2.Common factors in an accident*) and they are divided into normal driving, accident situation or serious accident according to the results taken from crash tests. Thus, the algorithm receives data from sensors every 1 millisecond (so as not to miss vital information) and compares it with the aforementioned limits. Depending on which band is situated, a variable representing each measure takes a value 0, 1 or 2 (0 for normal driving, 1 for accident situation and 2 for serious accident). As seen in following figure, these variables are the following:

- There are two variables for the accelerometer: "ac_data_x" (x-axis) and "ac_data_y" (y-axis).
- The variable for gyroscope ("Value_dps") is the roll angle in dps.
- Rider and passenger sensors are represented by a variable ("sum_sensor") that goes from 0 to 6 indicating the number of active sensors. They will be taken into account only if speed ("Speed") is greater than 1,39 m/s (5 Km/h).

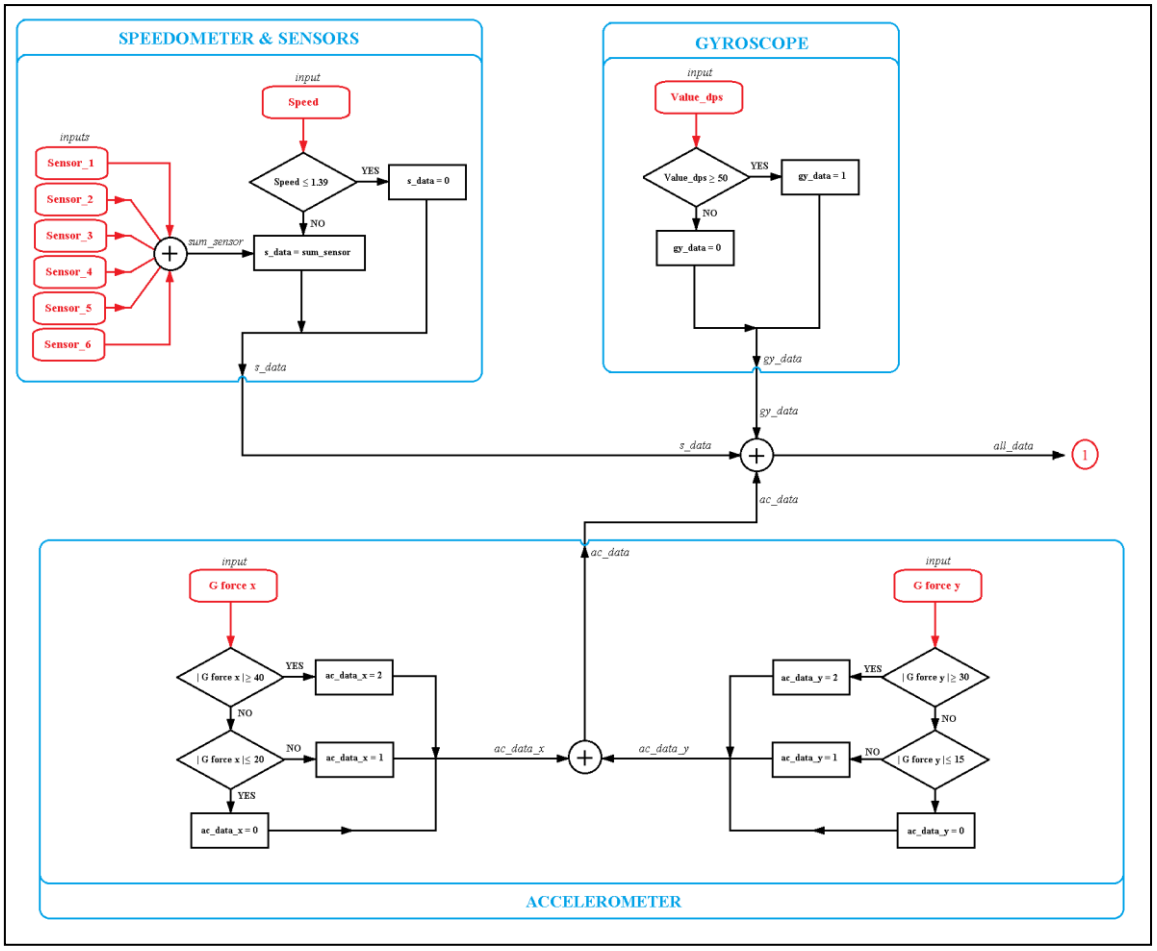


Figure 10. Sensor, gyroscope and accelerometer flow diagrams

Next figure shows main flow diagram, here can be seen that system includes two values: a partial ("partial_sum") and a global sum ("global_sum"). The partial sum is done every time there is a new set of sampled sensor values (every millisecond) and consists of the addition of the previous values, while the global one is obtained by adding the partials. If the sum reaches a value called "lim"(limit), then the system assumes there was an accident and makes an emergency call. In this case, the limit is set on 20.

In order to reset the global sum it is established a variable called "Samp_to_reset" (samples to reset) which defines how many consecutive partial sums must be zero for reset the global one. This variable avoids false alarm due to vibrations or sudden movements which are not accidents.

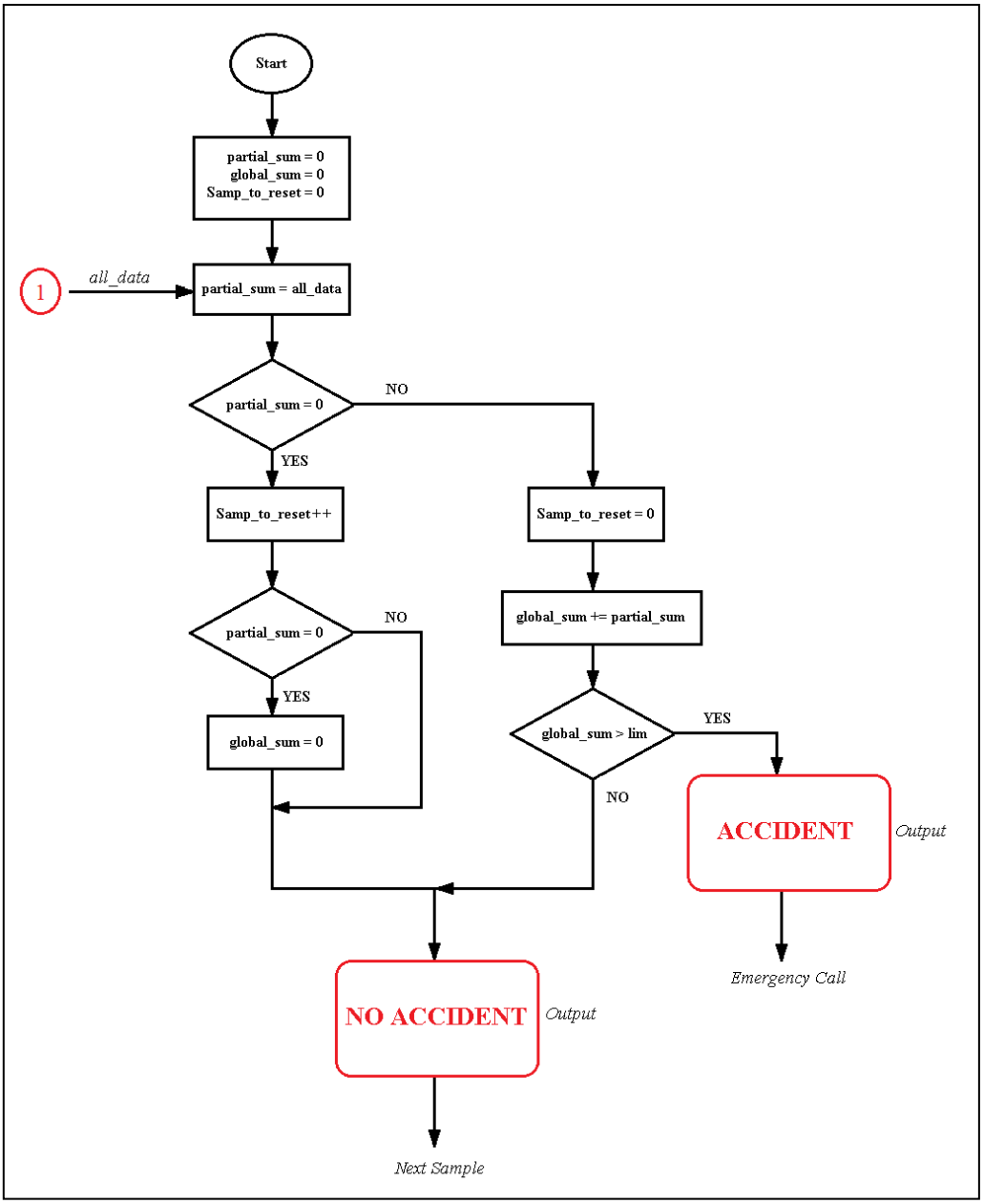


Figure 11. Main flow diagram

3.2 Testing the algorithm

The algorithm is going to be tested by using some situations (both normal driving and accidents) designed for this purpose. Thanks to crash tests developed using Working Model (for further information about these tests, please look at section *2.1.Crash tests*) and a recent study of the Winterthur Foundation (in collaboration with the Traffic Institute at the University of Valencia) which analyzes the causes, its consequences and possible solutions to improve safety of motorcycle accidents in Spain, it has been possible to draw such situations.

This report says that the accident rate varies depending on the parts involved, and the results are the following:

- **Accidents alone:** the most common is exit route.
- **Accidents between vehicles (motorcycles and cars, motorbikes and truck/van and motorcycle and moped):** the most frequent accident is frontal and front-side crash.
- **Accidents between two motorcycles and motorcycle and bus:** the most common collisions are reach impacts.
- **Multiple collisions:** the prevalent accidents are also frontal and front-side collisions, followed by side impacts.

In conclusion, it can be said there are four types of potential impacts: **frontal, rear and side and a combination of these**. The algorithm should be able to detect any kind of accident in order to provide accurate information to emergency services. Next figure shows the three Cartesian axes used to determine the impacts outlined above. Thus, frontal and rear impacts are related to x-axis, while side crashes are connected to y-axis.

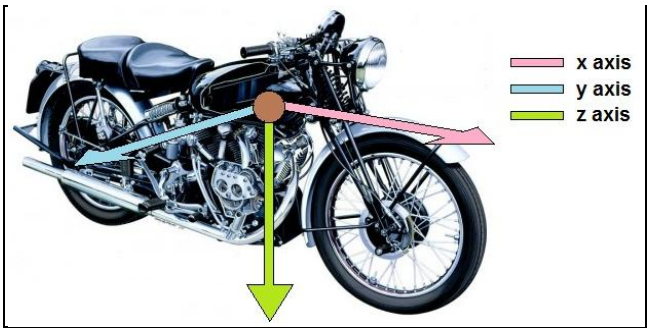


Figure 12. The three Cartesian axes on a motorbike

The algorithm has been firstly designed using Matlab (the code is included in the CD) so it can be tested quickly and efficiently. The Matlab code designed to test Moto eCall is called "test_mec.m" and its basic usage is explained below:

$$[Testing_{Time}] = test\_mec\ (Test_{Situation})$$

where inputs are:

- **Test_Situation:** Situation to be tested.

and outputs are:

- **Testing_Time:** Time spent by test

The aforementioned code reads data from "Driving situations.xls" database (included in the CD). This database contains 14 spreadsheet with different situations created to test the algorithm. These situations are the following:

- **First situation:** Acceleration
- **Second situation:** Deceleration
- **Third situation:** Curve
- **Fourth situation:** Frontal impact at 120 km/h
- **Fifth situation:** Frontal impact at 50 km/h
- **Sixth situation:** Rear impact while motorcycle is stopped
- **Seventh situation:** Rear impact while motorcycle is in motion
- **Eighth situation:** Side impact while motorcycle is stopped
- **Ninth situation:** Side impact while motorcycle is in motion
- **Tenth situation:** Front-side impact in an angle of 30º
- **Eleventh situation:** Front-side impact in an angle of 45º
- **Twelfth situation:** Front-side impact in an angle of 70º
- **Thirteenth situation:** Slide side crash
- **Fourteenth situation:** Road exit

It should be added that for the preparation of curve situations, the following relations are needed:

$$\tan(\alpha) = \frac{v^2}{R \cdot g}$$

where:

- **α:** Lean angle (radians)
- **v**: Speed (m/s)
- **R**: Turn radius of the curve (m)

All spreadsheets have the same structure, six columns with the following information:

- **First column:** Label where situations are defined (0 =No accident / 1 = Accident)
- **Second column:** Speed in metres/second
- **Third column:** Acceleration in x-axis in
- **Fourth column:** G Acceleration in y-axis in G
- **Fifth column:** Angular rate in degrees per second
- **Sixth column:** Number of activated sensors

Thus, the program reads each line and gets the six different values. Then, it assigns values due to the bands explain before and makes the partial and global summation. Finally, it prints on screen this last vector and compares it with the first column of spreadsheet (this value is multiplied by 20 in order to be equivalent to the limit set on previous section 3*.1.Moto eCall algorithm.*

3.2.1 Acceleration

In this situation, it can be seen how a motorcycle accelerates from 0 to 120 Km/h in a straight line, within that is considered a normal driving. Following graphic displays the values taken from all devices.

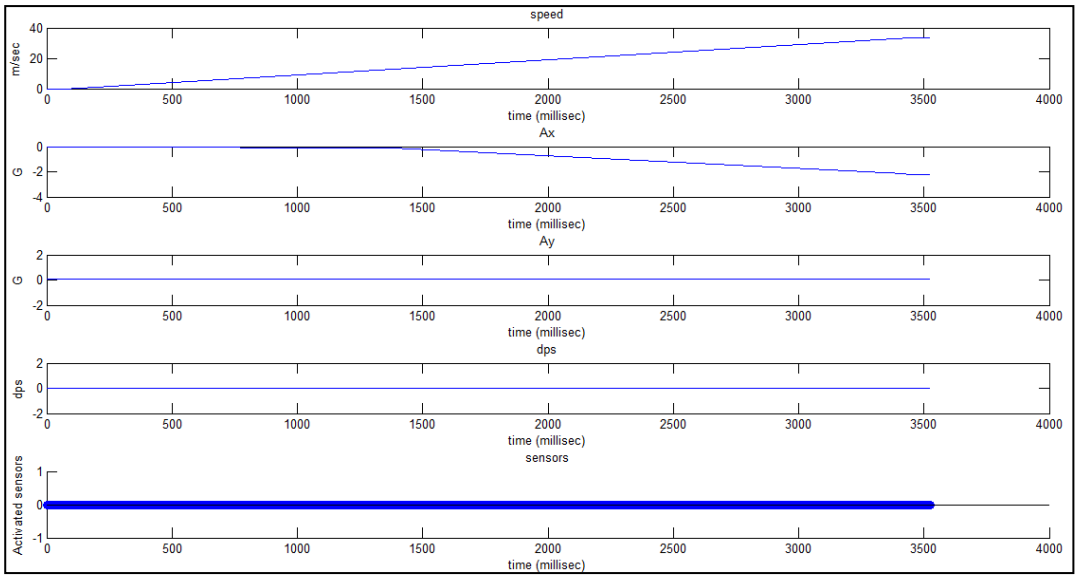


Figure 13. First situation, acceleration

Once the algorithm has been running, it returns a vector called "global sum", as seen in figure below.

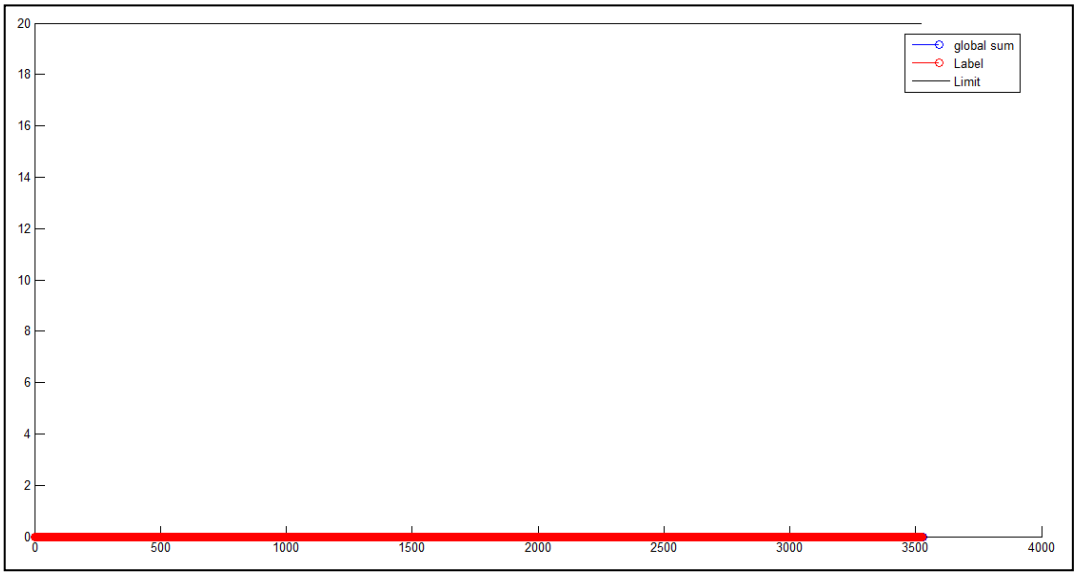


Figure 14. First situation, global sum

3.2.2 Deceleration

Here it can be seen how a motorcycle brakes from 120 to 0 Km/h in a straight line, within that is considered a normal driving. Next illustration presents the values taken from all devices.

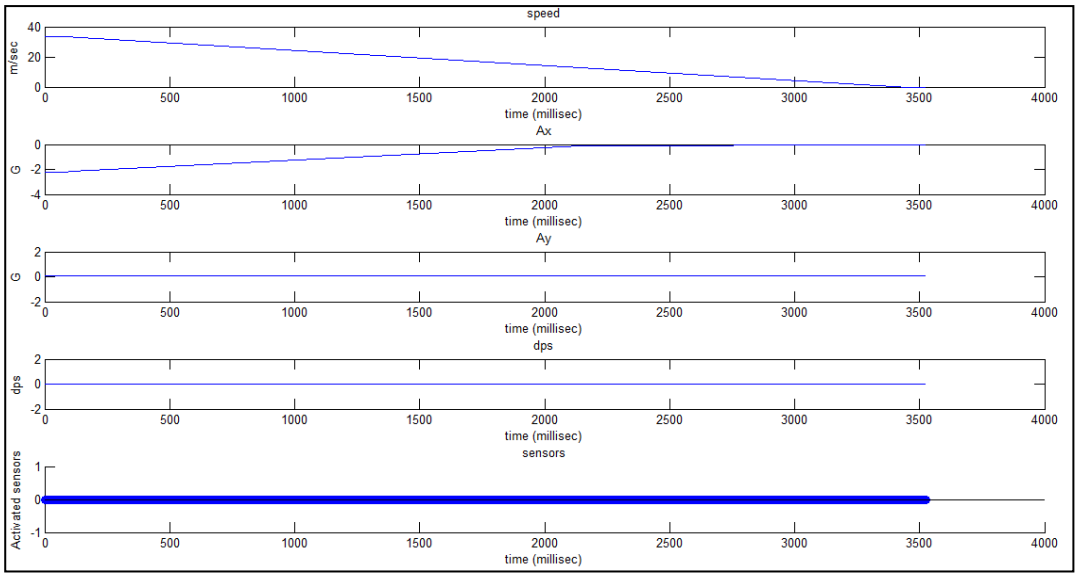


Figure 15. Second situation, deceleration

Once the algorithm has been running, it returns a vector called "global sum". Following figure shows the results.

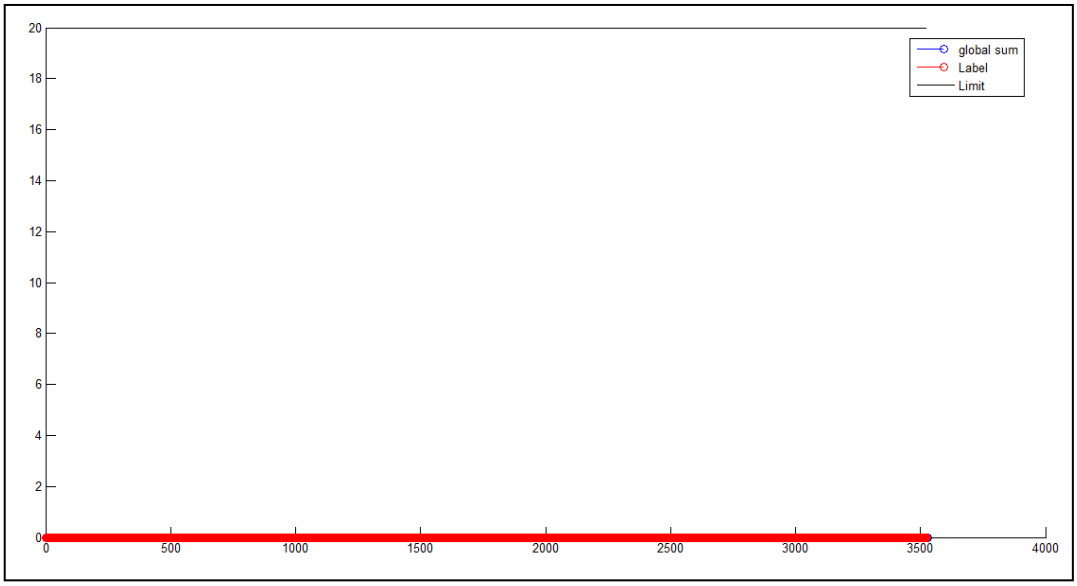


Figure 16. Second situation, global sum

3.2.3 Curve

This scenario shows how a motorcycle takes a curve 200 metres radius at 120 Km/h, within that is considered a normal driving. This situation is shown below.

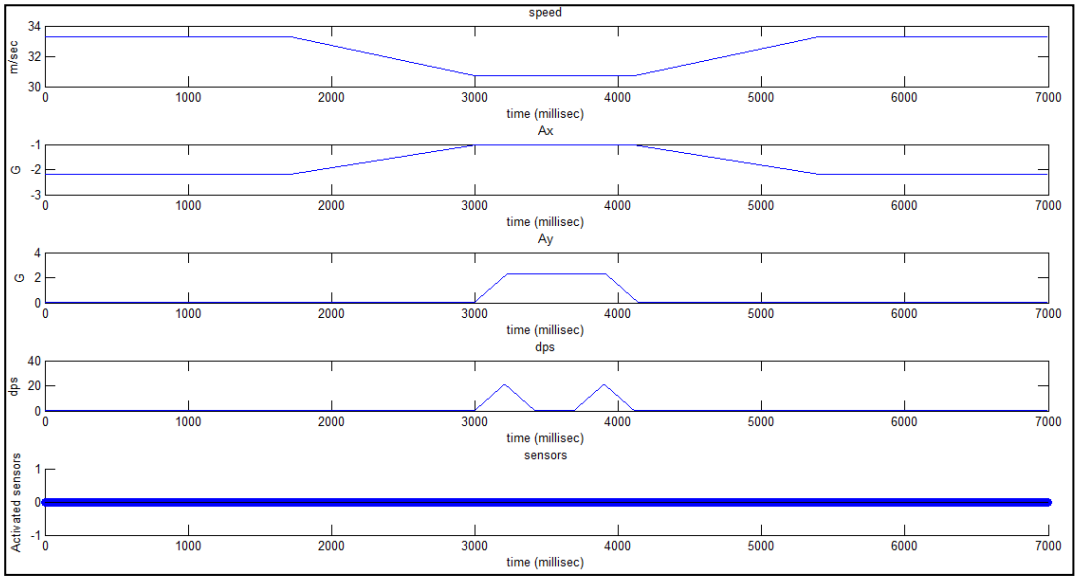


Figure 17. Third situation, curve

Picture below displays the vector called"global sum" returned by the program.

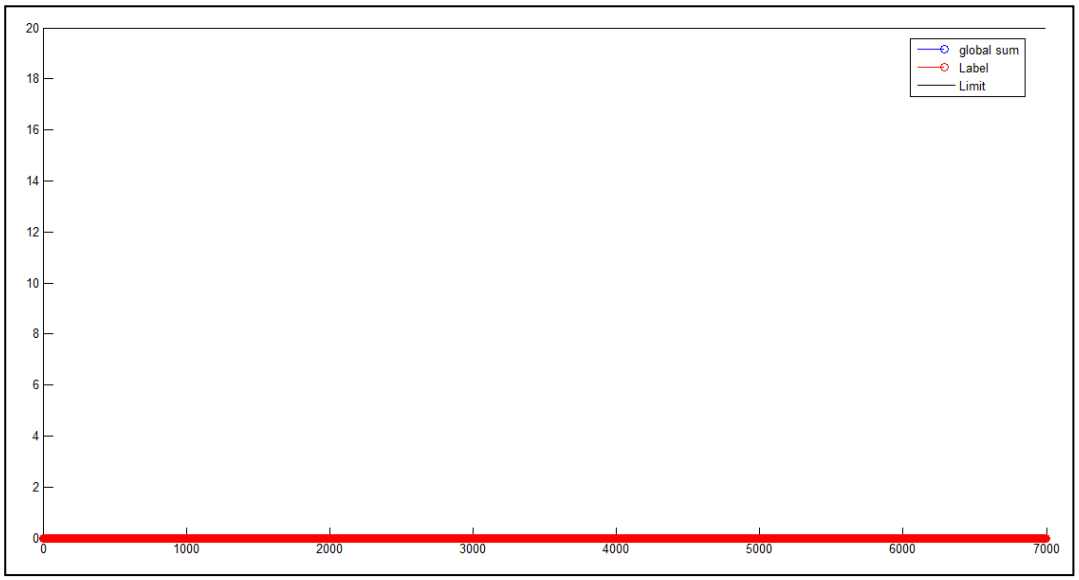


Figure 18. Third situation, global sum

3.2.4 Frontal impact at 120 Km/h

In this situation, it can be seen how a motorcycle suffers a frontal impact while driving at 120 Km/h, within that is considered a crash situation. Figure below shows the values taken from all devices.

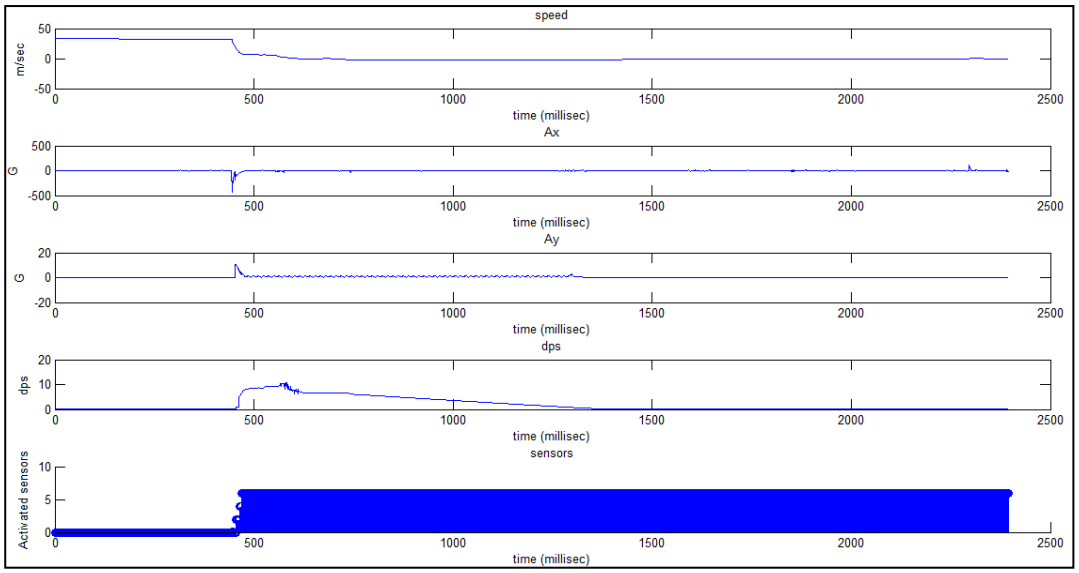


Figure 19. Fourth situation, frontal impact at 120 Km/h

Below, "global sum" it is shown, which is the vector returned after running the algorithm.

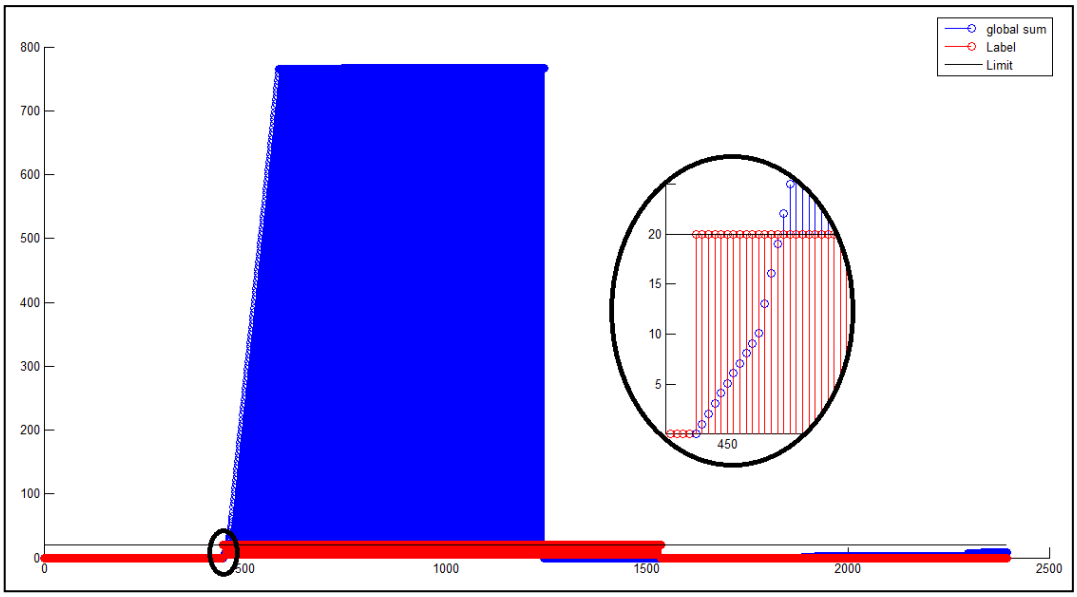


Figure 20. Fourth situation, global sum

3.2.5 Frontal impact at 50 Km/h

Here it can be seen how a motorcycle suffers a frontal impact while driving at 50 Km/h, within that is considered a crash situation. Below, the values taken from all devices are presented.

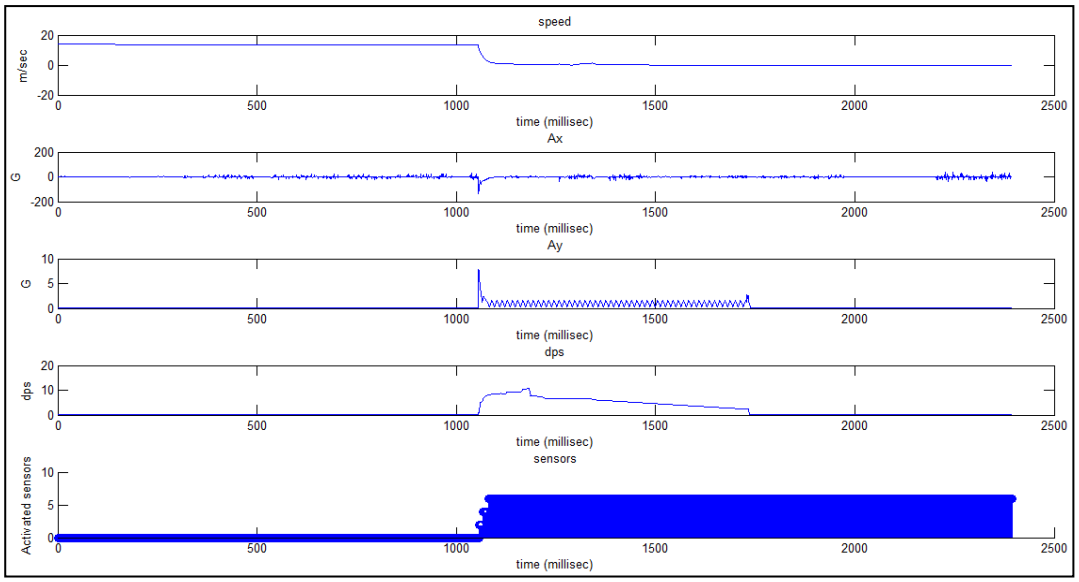


Figure 21. Fifth situation, frontal impact at 50 Km/h

Once the algorithm has been running, it returns a vector called "global sum". Next picture shows the results.

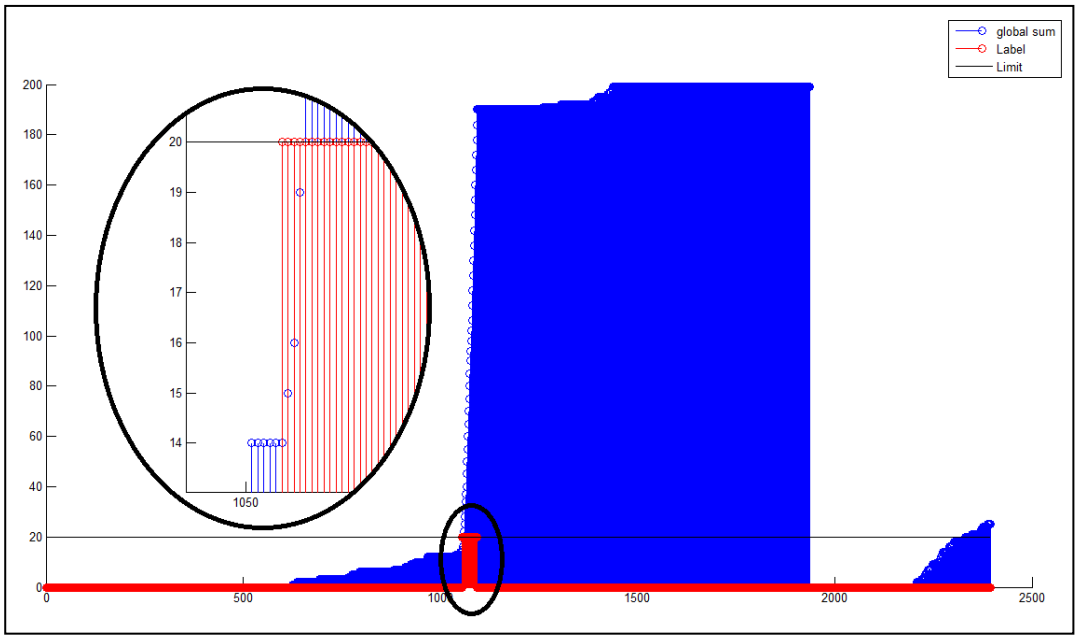


Figure 22. Fifth situation, global sum

3.2.6 Rear impact while the motorcycle is stopped

Here, a motorcycle suffers a rear impact while standing at a traffic light, within that is considered a crash situation. Figure below shows the values taken from all devices.

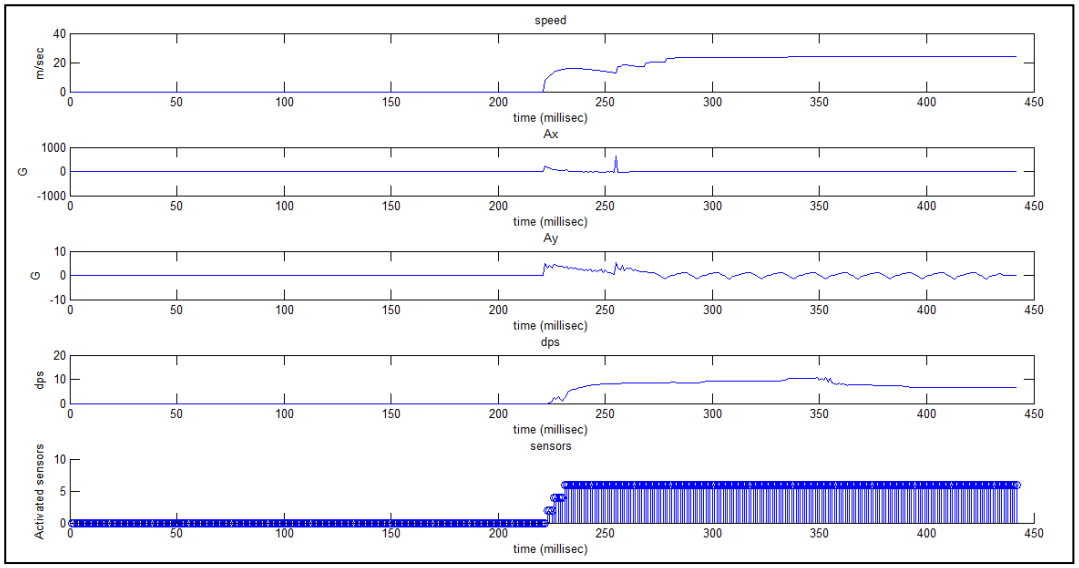


Figure 23. Sixth situation, rear impact while the motorcycle is stopped

Once the algorithm has been running, it returns a vector called "global sum". The following picture shows the results.

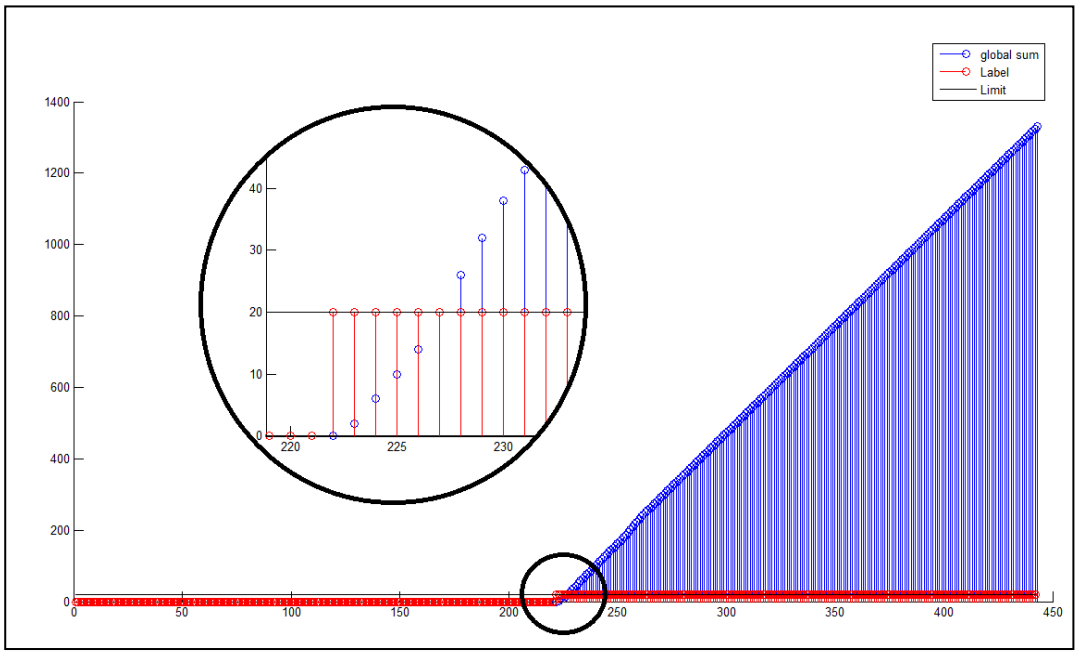


Figure 24. Sixth situation, global sum

3.2.7 Rear impact while the motorcycle in motion

This scenario shows how a motorcycle suffers a rear impact while driving at 25 Km/h, within that is considered a crash situation. Next figure displays the values taken from all devices.

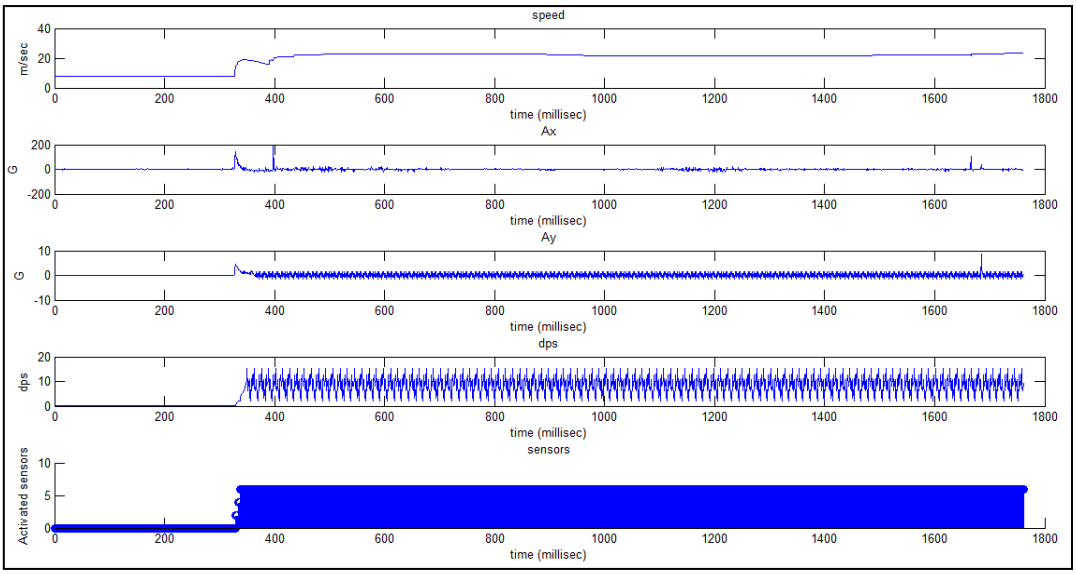


Figure 25. Seventh situation, rear impact while the motorcycle is in motion

Below it is shown the vector called "global sum", which is returned after running the algorithm.

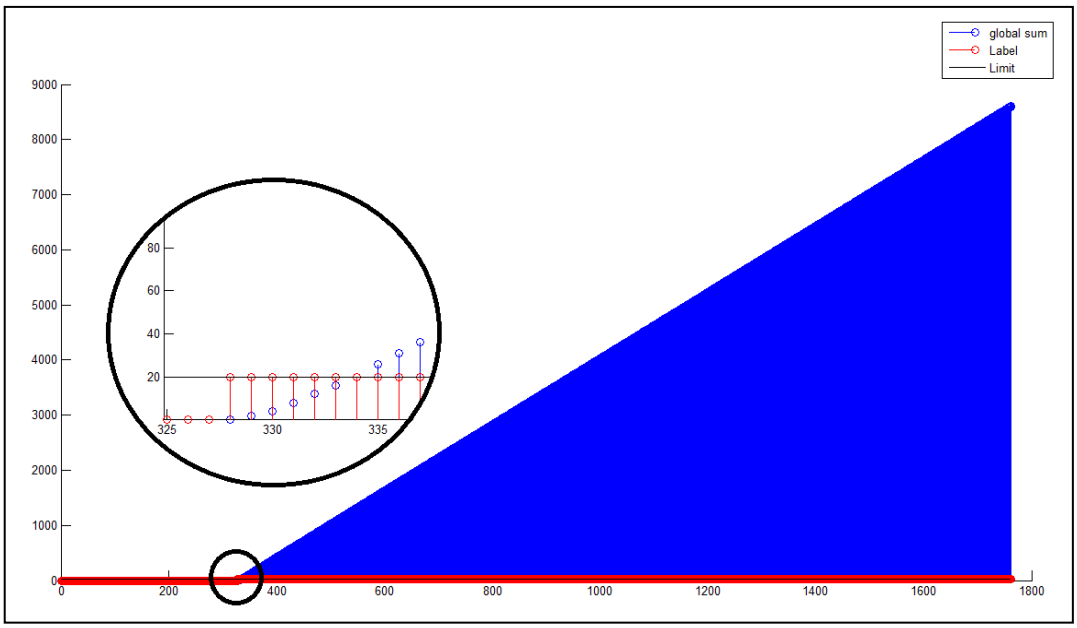


Figure 26. Seventh situation, global sum

3.2.8 Side impact while the motorcycle is stopped

Here, a motorcycle suffers a side impact while standing at a traffic light, within that is considered a crash situation. Values taken from all devices are displayed below.

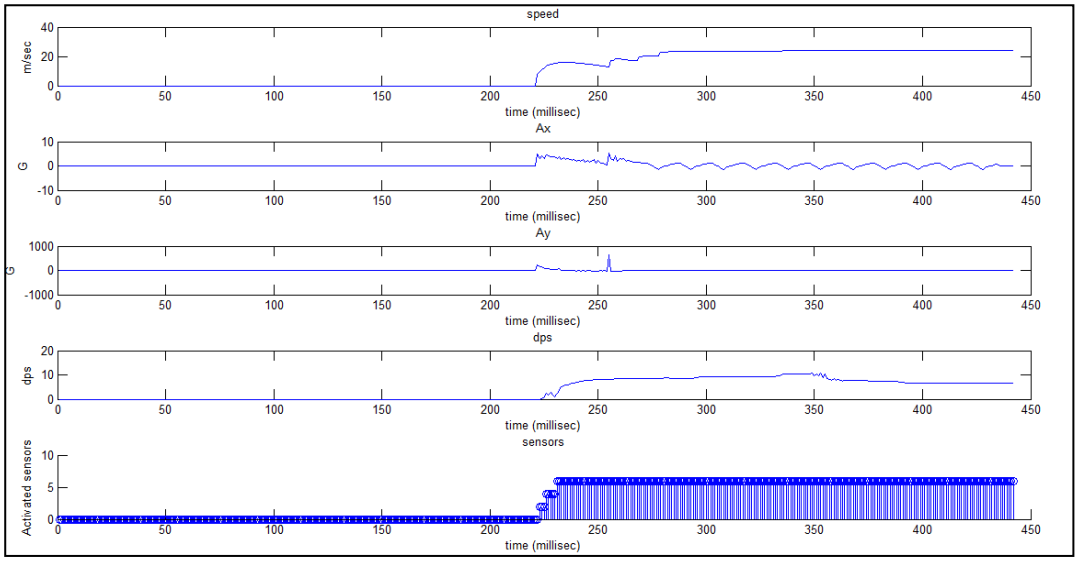


Figure 27. Eight situation, side impact while the motorcycle is stopped

Once the algorithm has been running, it returns a vector called "global sum". Next figure shows the results.

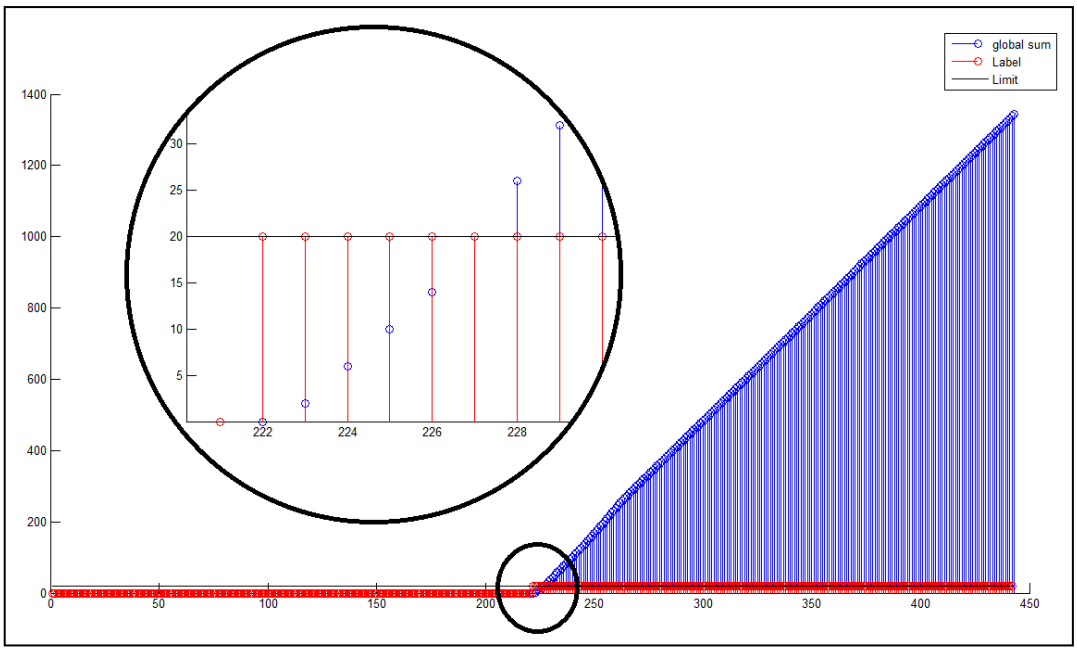


Figure 28. Eight situation, global sum

3.2.9 Side impact while the motorcycle is in motion

In this situation, it can be seen how a motorcycle suffers a side while crossing an intersection, within that is considered a crash situation. Picture below shows the values taken from all devices.

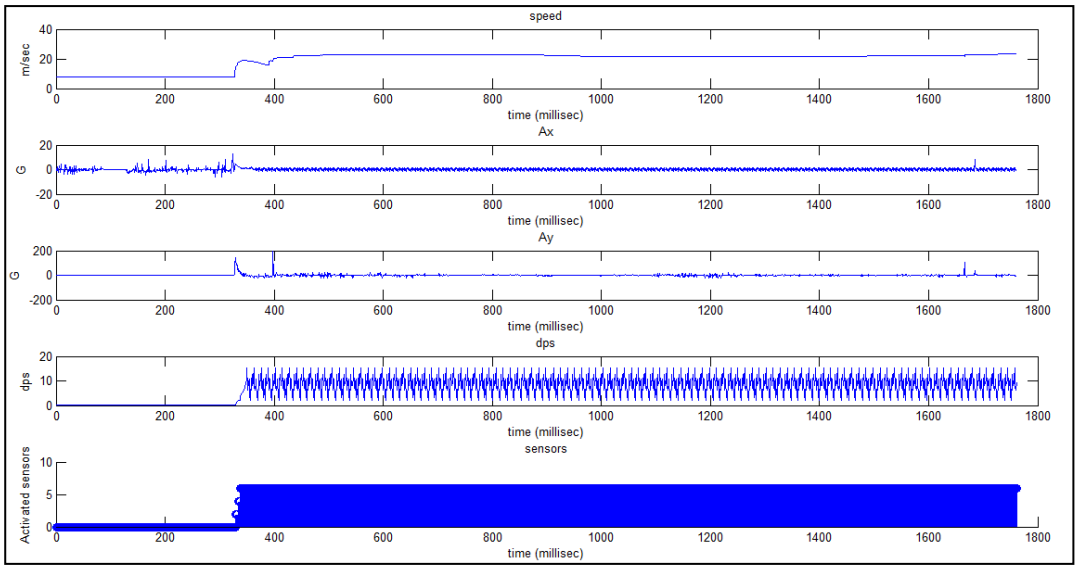


Figure 29. Ninth situation, side impact while the motorcycle is in motion

Once the algorithm has been running, it returns a vector called "global sum". Next figure displays the results.

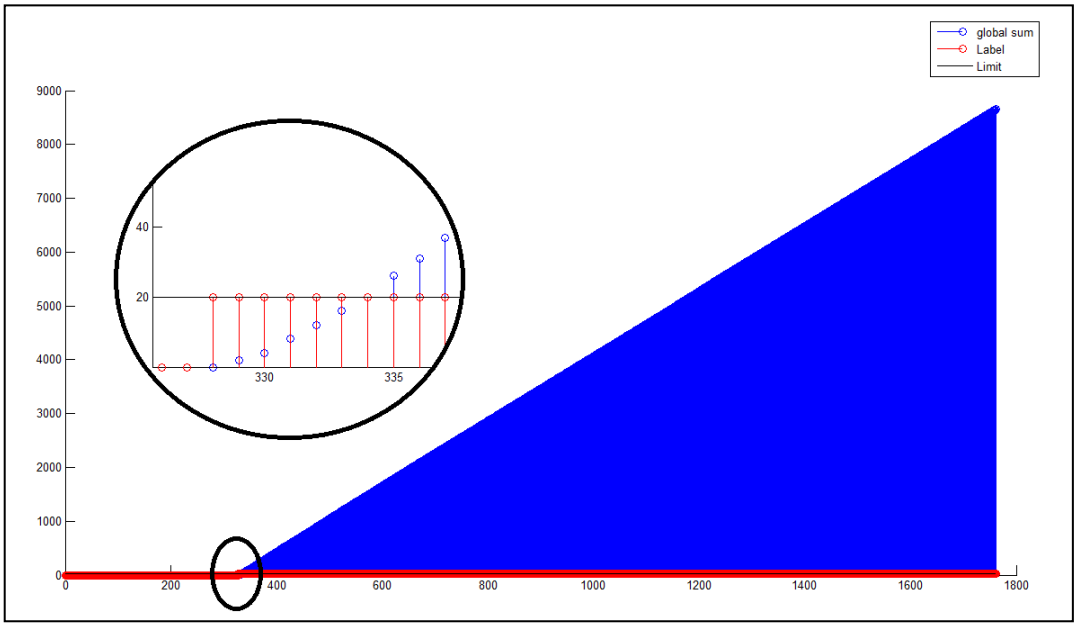


Figure 30. Ninth situation, global sum

3.2.10 Front-side impact in an angle of 30°

In this scenario, it can be seen how a motorcycle suffers a front-side impact in an angle of 30° with the other vehicle while crossing an intersection, within that is considered a crash situation. Below, values taken from all devices are shown.

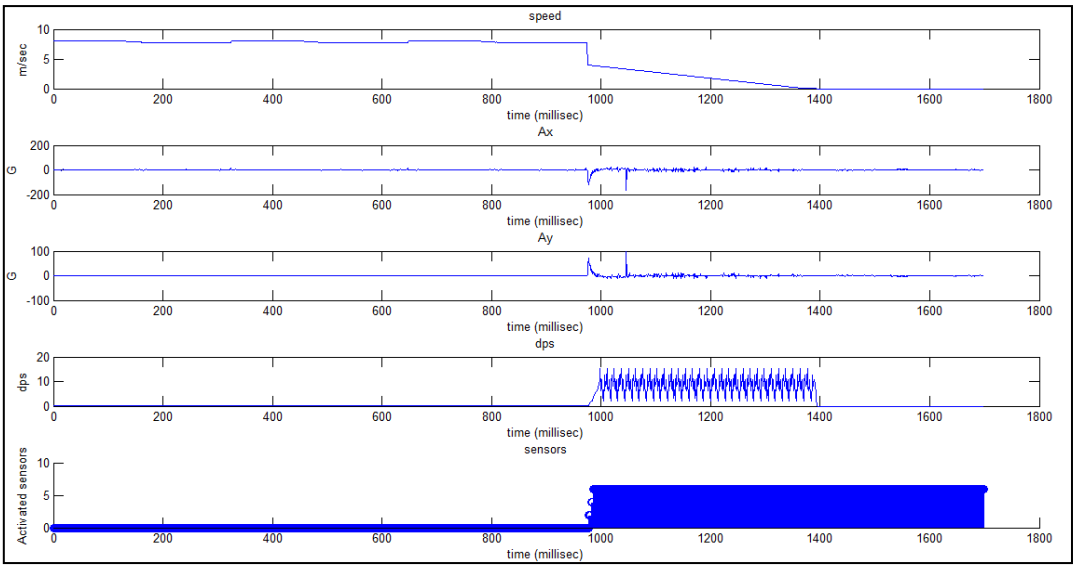


Figure 31. Tenth situation, front-side impact in an angle of 30°

Once the algorithm has been running, it returns a vector called "global sum". The following picture displays the results.

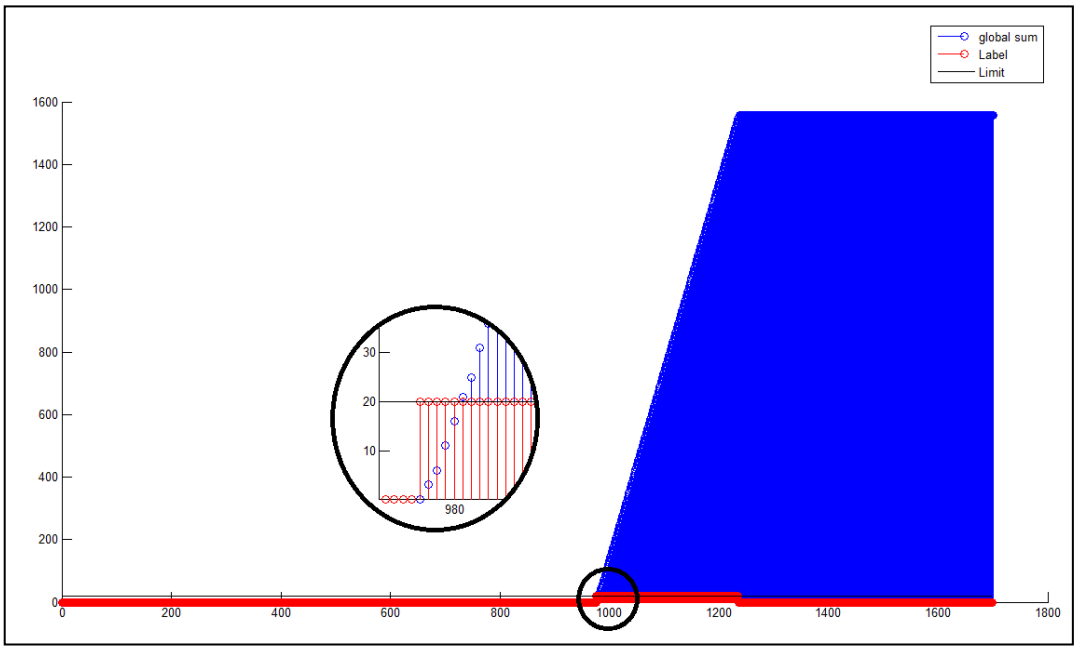


Figure 32. Tenth situation, global sum

3.2.11 Front-side impact in an angle of 45°

In this situation, it can be seen how a motorcycle suffers a front-side impact in an angle of 45° with the other vehicle while crossing an intersection, within that is considered a crash situation. Figure below shows the values taken from all devices.

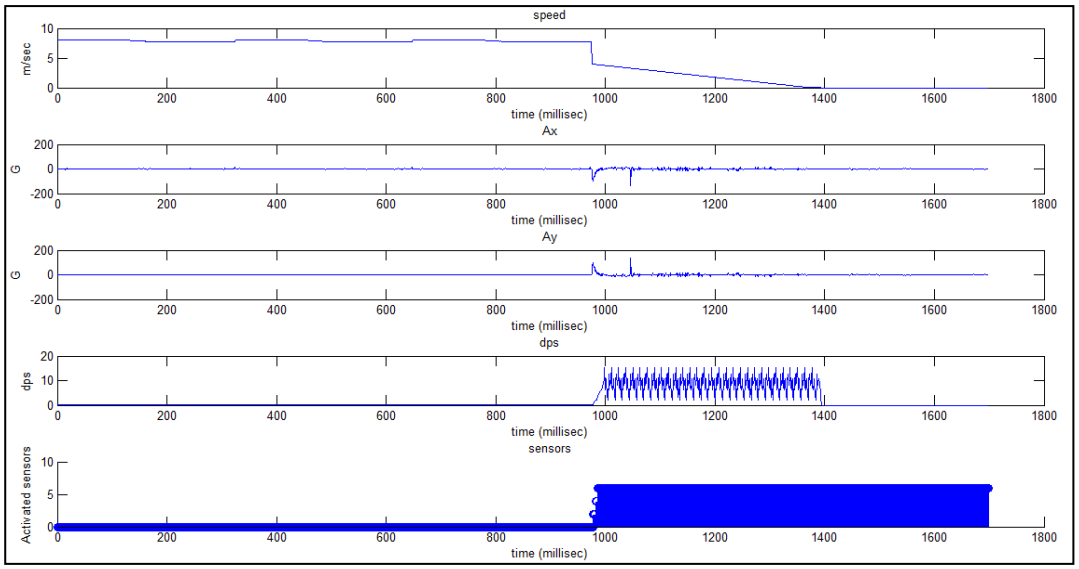


Figure 33. Eleventh situation, front-side impact in an angle of 45°

Below it is shown the vector called "global sum", which is returned after running the algorithm.

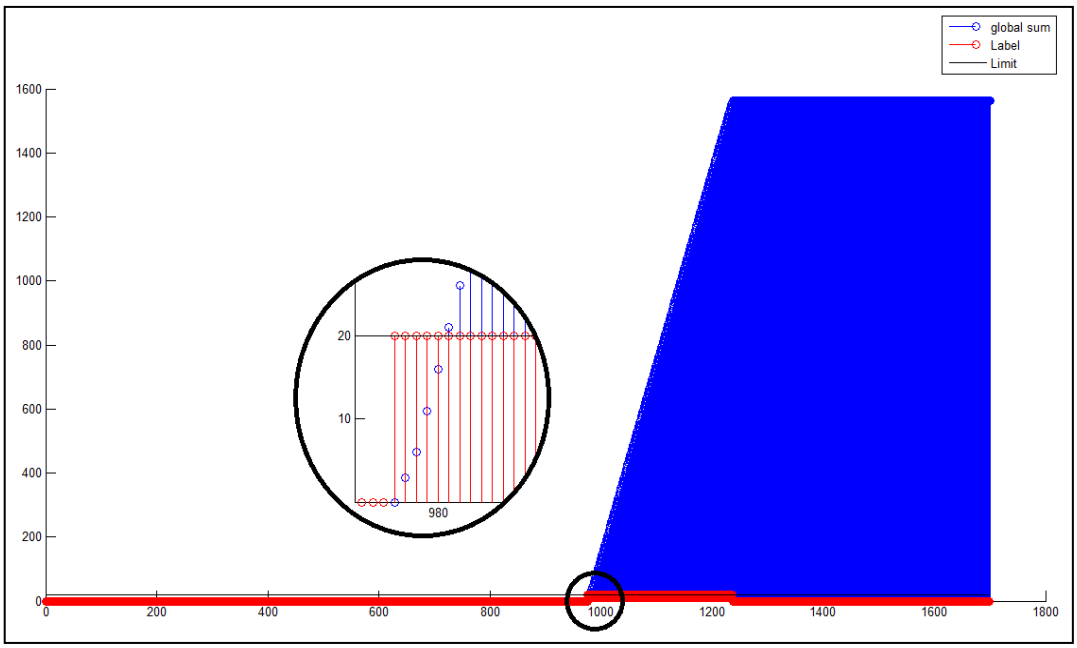


Figure 34. Eleventh situation, global sum

3.2.12 Front-side impact in an angle of 70°

Here, a motorcycle suffers a front-side impact in an angle of 70° with other vehicle while crossing an intersection, within that is considered a crash situation. Next illustration shows the values taken from all devices.

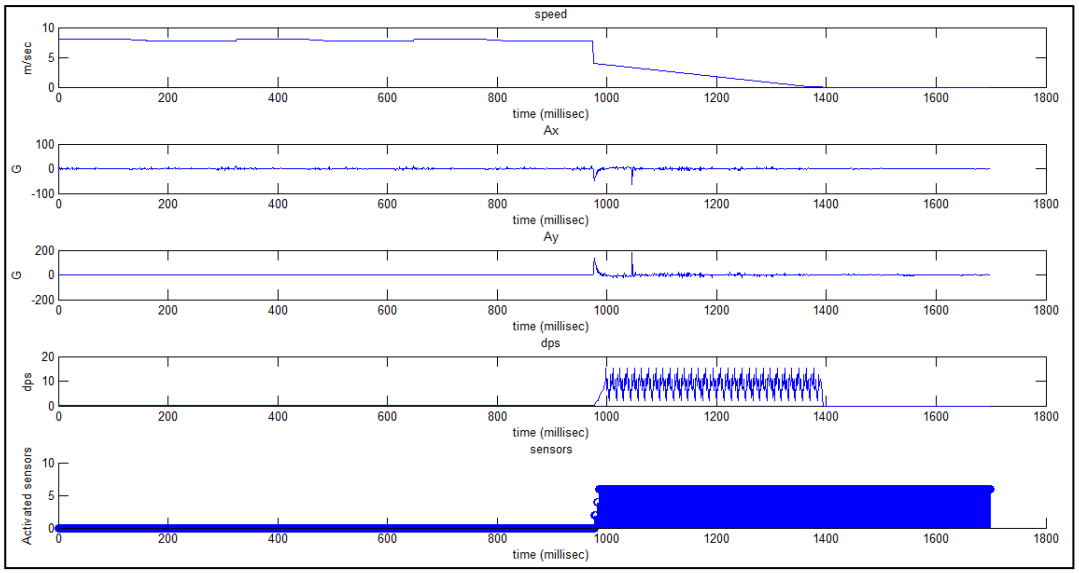


Figure 35. Twelfth situation, front-side impact in an angle of 70°

Once the algorithm has been running, it returns a vector called "global sum" as can be seen below.

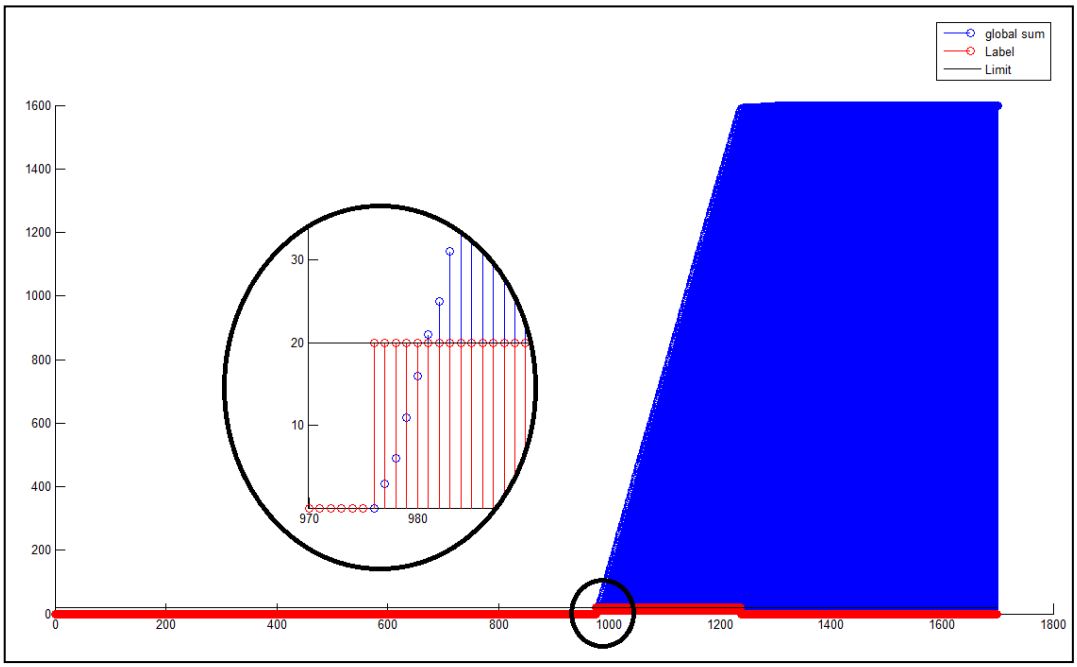


Figure 36. Twelfth situation, global sum

3.2.13 Slide side crash

Thirteenth situation shows how a motorcycle skids and slides in a curve, within that is considered a crash situation. Values taken from devices can be seen below.

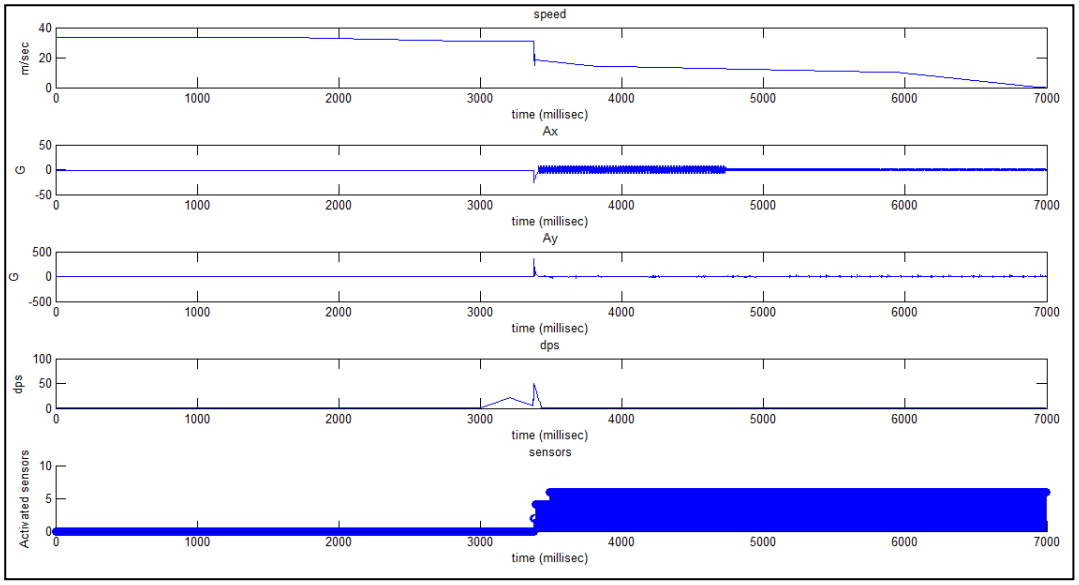


Figure 37. Thirteenth situation, slide side crash

Once the algorithm has been running, it returns a vector called "global sum". Next picture shows the results.

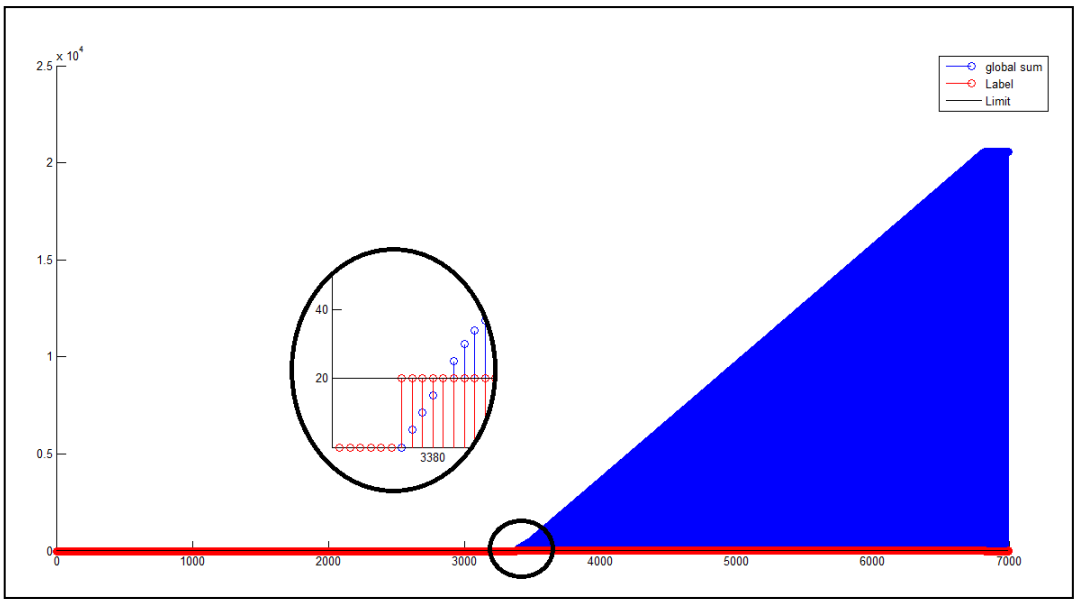


Figure 38. Thirteenth situation, global sum

3.2.14 Road exit

Here, a motorcycle looses control and there is a road exit, within that is considered a crash situation. Following illustration shows the values taken from all devices.

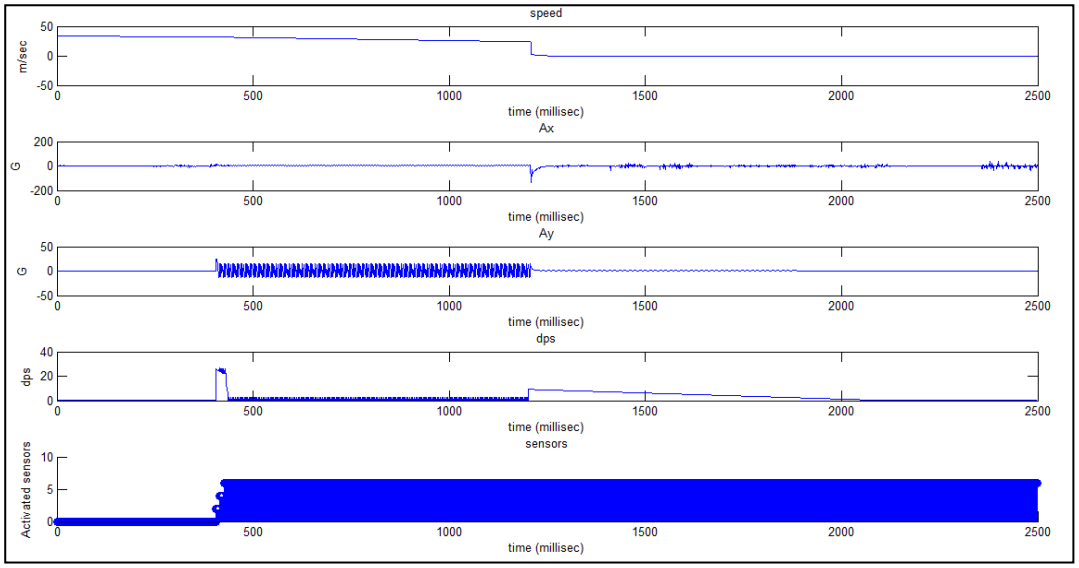


Figure 39. Fourteenth situation, road exit

Once the algorithm has been running, it returns a vector called "global sum". Here, the results are displayed.

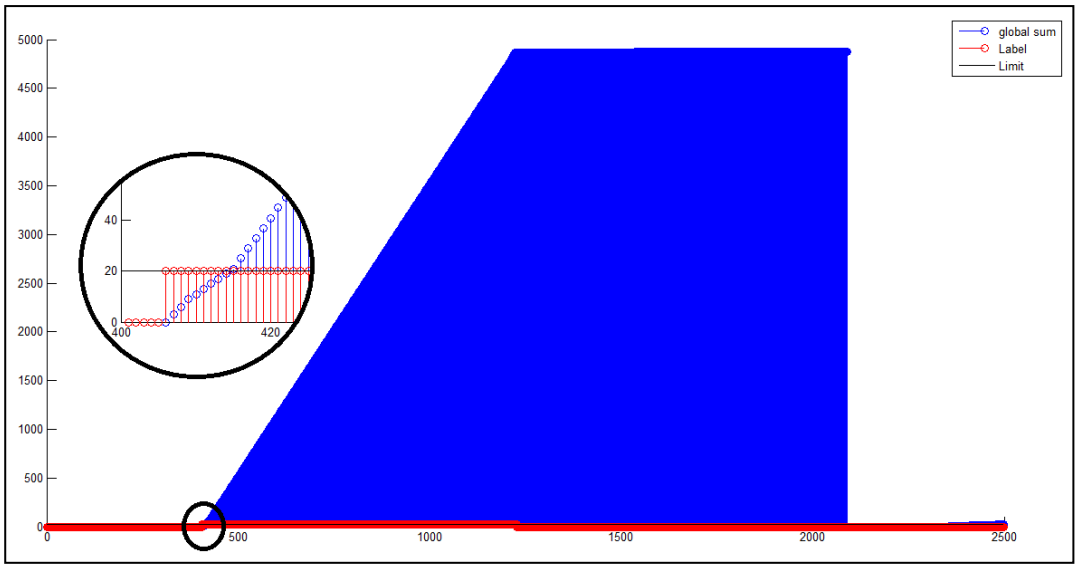


Figure 40. Fourteenth situation, global sum

3.3 Discussion of tests results

After displaying the fourteen situations, the following table summarizes the results. It shows in three columns the maximum values of global summations, time spent on tests and accuracy of each case.

In second column can be seen that the minimum value (obviating normal driving) corresponds with the fifth situation. Thus, if the limit is set at 20 (as explained in section *3.1.Moto eCall algorithm*), when the sum reaches this value, the system assumes that an accident has occurred. Therefore, because the system will detect all possible crash situations.

Third column shows time (in seconds) spent on testing data, which oscillates between 0.60 and 0.78 seconds. In all cases, the algorithm takes less than a second to make all operations. It is fast enough for our system but these duration times can be improvable.

Fourth column represents the accuracy of the algorithm in each tested situation. In this case, accuracy means the time (in milliseconds) to reach the limit. Given that each sample is taken every millisecond, this amount of time is equal to the number of samples to reach the limit. It varies between 4 and 14 milliseconds, which is completely reasonable due to the constitution of the system: as explained in previous section *3.1.Moto eCall algorithm*, the system takes into account the sequence of events in an accident so it does not reach the limit immediately. Nevertheless, this time to reach the limit can be reduced.

Situation	x_global max	Test time (seconds)	Time to reach the limit (milliseconds)
1st	0	0.78	-
2nd	0	0.73	-
3th	0	0.85	-
4th	766	0.68	14
5th	199	0.65	4
6th	1330	0.60	6
7th	8604	0,67	6
8th	1344	0.61	6
9th	8655	0.67	7
10th	1566	0.65	5
11th	1568	0.66	5
12th	1599	0,66	5
13th	20000	0,86	5
14th	4876	0,72	9

Table 9. Moto eCall test results

Summarizing, it can be appreciated that the algorithm works efficiently and is capable to detect the multiple sequences when an accident occurs. However, the system can be improved by reducing testing time and increasing accuracy.

4. IMPROVING THE SYSTEM

Here the possible improvements of the current system are explained and analysed. This section is divided into two subsections:

- **Artificial Neural Networks:** Here, ANNs are explained definition, types, layers, learning process and applications). Besides, the suitability of neural networks for the system will be studied.
- **Extreme Learning Machine:** ELM is described in this subsection. Besides, the suitability of this learning technique will be analysed.

4.1 Artificial Neural Network

4.1.1 What is an artificial neural network?

Definition

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems process information. The key element of this network is the innovative structure of the information processing system. It is composed of a large number of highly interconnected processing elements, called neurons, working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true for ANNs as well.

Types of neural networks

There are several types of neural networks according to the input signals flow, the most important are described below.

- **Feedback networks** can have signals travelling in both directions by introducing loops in the network. These type of networks are very powerful and dynamic, their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.
- **Feed-forward networks** allow signals to travel one way only, from input to output. There is no feedback or loops, i.e. the output of any layer does not affect that same layer. These kind of ANN tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

Network layers

The most common type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units as can be seen in following figure.

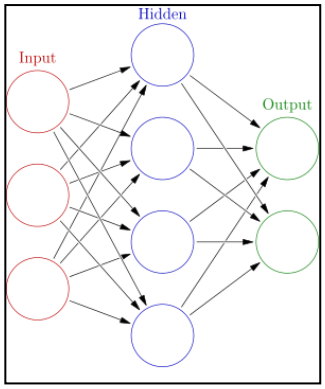


Figure 41. Neural network layers

The activity of the input units represents the raw information that is fed into the network, while the activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units. Lastly, the behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

In this kind of network, the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

Besides, two different architectures can be distinguished: single-layer and multi-layer architectures.

- The **single-layer** organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organisations.
- In **multi-layer** networks, units are often numbered by layer, instead of following a global numbering.

The Learning Process

The memorisation of patterns and the subsequent response of the network can be categorised into two general paradigms:

- **Associative mapping** in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:
  - Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completition, i.e. to produce a pattern whenever a portion of it or a distorted pattern is presented.
  - Hetero-association: the network stores pairs of patterns building an association between two sets of patterns. It is related to two recall mechanisms:
    - ✓ *Nearest-neighbour recall*, where the output pattern produced corresponds to the input pattern stored, which is the closest to the pattern presented.
    - ✓ *Interpolative recall*, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented.

- **Regularity detection** in which units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.

Every neural network possesses knowledge which is contained in the values of the connections weights. Therefore, modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, it can be distinguished two major categories of neural networks:

- **Fixed networks** in which the weights cannot be changed (i.e. dW/dt=0). In such networks, the weights are fixed a priori according to the problem to solve.
- **Adaptive networks** which are able to change their weights (i.e. dW/dt not= 0).

All learning methods used for adaptive neural networks can be classified into two major categories: supervised and unsupervised learning.

- **Supervised learning** which incorporates an external supervisor, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.
  An important issue concerning supervised learning is the problem of error convergence, i.e. the minimisation of error between the desired and computed unit values. The aim is to determine a set of weights which minimises the error.
- **Unsupervised learning** uses no external supervisor and is based upon only local information. It is also referred to as self-organisation, in the sense that it self-organises data presented to the network and detects their emergent collective properties.

A neural network learns **off-line** if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed **on-line**.

Transfer Function
The behaviour of an ANN depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- **Linear** (or ramp): the output activity is proportional to the total weighted output.
- **Threshold:** the output activity is proportional to the total weighted output
- **Sigmoid:** the output varies continuously but not linearly as the input changes

Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

In order to make a neural network that performs some specific task, it is necessary to choose the way the units connected between then and also set the weights on the connections appropriately. On one hand, the connections determine whether it is possible for one unit to influence another. On the other hand, weights specify the strength of the influence.

Applications
The tasks artificial neural networks are applied to tend to fall within the following broad categories:

- Pattern classification: including time series prediction, customer databases, fitness approximation, etc.
- Function approximation: modelling
- Classification, including character and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and image compression.
- Robotics, including directing manipulators, prosthesis.
- Security and control, including credit card fraud and computer numerical control
- Prediction: Stock Market Prediction

4.1.2 Suitability study

Advantages
Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" which can be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

- **Adaptive learning:** An ability to learn how to do tasks based on the data given for training or initial experience.
- **Self-Organization:** An ANN can create its own organization or representation of the information it receives during learning time.
- **Real Time Operation:** ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- **Fault Tolerance via Redundant Information Coding:** Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Limitations
However, like any technique, neural networks have certain drawbacks. Most of the shortcomings are based upon the current state of the art.

- Since the ANN finds a general approximation of a solution, there is a small error usually associated with all the outputs.

- There are many free parameters, such as the number of hidden nodes, the learning rate or minimal error that may influence final result.
- Like with any data-driven models, they cannot be used if there is no or very little data available.
- Neural networks are not good at arithmetic and precise calculations.
- The full nature of neural networks is still not fully understood, and thus current research must take an experimental approach to the problem of performance. There are possibilities of illogical network behaviours.
- Neural networks require lengthy training samples and long processing times.
- At present, there are not any ANN computers available at a reasonable cost.

Neural networks vs. conventional computers

ANNs take a different approach to solve problems than conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem capability of conventional computers to solve problems already understood, but computers would be more useful if they could solve problems which are not understood yet.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (Neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

Pattern recognition

As said before, an important application of neural networks is pattern recognition, which can be implemented by using a feed-forward neural network trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

Conclusions

Table below summarizes the characteristics that an ANN should fulfils to be compatible with current crash detection system.

Characteristics	**ANN suitability study for Moto eCall**
Types of neural networks • *Feedback* • *Feed-forward*	Feed-forward
Network layers • *Single-layer* • *Multi-layer*	Single-layer

Learning Process
• *Associative mapping*
  ○ Auto-association
  ○ Hetero-association
• *Regularity detection*
	Associative mapping with auto-association

Performed learning
• *Fixed networks*
• *Adaptive networks*
  ○ Supervised learning
  ○ Unsupervised learning
	Adaptive networks with unsupervised learning

Transfer Function • *Linear* • *Threshold* • *Sigmoid*	Sigmoid, but all three must be considered rough approximations

Table 10. ANN suitability study

4.2 Extreme Learning Machine

Extreme Learning Machine (Huang et al., 2006) (ELM) is a currently popular neural network architecture based on random projections. It has one hidden layer with random weights, and an output layer whose weights are determined analytically. This sections explains in detail ELM architecture and algorithm.

4.2.1 What is Extreme Learning Machine?

Definition

Neural networks play key roles in machine learning and data analysis (among others, such as support vector machines – SVM). This popular learning technique has to face some challenging issues such as: intensive human intervene, slow learning speed, poor learning scalability.

It is clear that the learning speed of feed-forward neural networks (FFNN) is in general far slower than required and it has been a major bottleneck in their applications for past decades. According to Huang et al. (2006), there are two main reasons behind this

behaviour, one is slow gradient based learning algorithms used to train neural network (NN) and the other is the iterative tuning of the parameters of the networks by these learning algorithms. To overcome these problems, it is proposed a learning algorithm called extreme learning machine (ELM) for single hidden layer feed-forward neural networks (SLFNs) which randomly selected the input weights and analytically determines the output weights of SLFNs. Huang et al. (2006), stated that "In theory, this algorithm tends to provide the best generalization performance at extremely fast learning speed".

This is extremely good as in the past it seems that there had existed an unbreakable virtual speed barrier which classic learning algorithms could not go break through and therefore feed-forward network implementing them would take a very long time to train itself, independent of the application type whether simple or complex. Also ELM tends to reach the minimum training error as well as it considers magnitude of weights which is opposite to the classic gradient-based learning algorithms which only intend to reach minimum training error but do not consider the magnitude of weights. Also unlike the classic gradient-based learning algorithms which only work for differentiable activation functions ELM learning algorithm can be used to train SLFNs with non-differentiable activation functions. According to Huang et al. (2006), "Unlike the traditional classic gradient-based learning algorithms facing several issues like local minimum, improper learning rate and overfitting, etc, the ELM tends to reach the solutions straightforward without such trivial issues".

Applications
ELM has been successfully used in the following applications:

- Biometrics
- Bioinformatics
- Image processing (image segmentation, image quality assessment, image super-resolution)
- Signal processing
- Human action recognition
- Disease prediction and eHealthCare
- Location positioning system
- Brain computer interface
- Human computer interface
- Feature selection
- Time-series
- Real-time learning and prediction
- Security and data privacy

4.2.2 ELM Training Algorithm

Description

ELM is based on the standard SLFN (single-hidden layer feed-forward neural networks). Given N samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n$ and $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^n$, the standard SLFN with N' hidden neurons and activation function g(x) is defended as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = o_j, \qquad j = 1, \dots, N$$

where $w_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ is the weight vector that connects the ith hidden neuron and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$ is the weight vector that connects the ith neuron and the output neurons, and bi is the threshold of the ith hidden neuron. The "·" in $w_i \cdot x_j$ means the inner product of w_i and x_j. SLFN aims to minimize the difference between o_j and t_j. mathematically expressed as:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j, \qquad j = 1, \dots, N$$

or, more in a matrix format as $H\beta = T$, where

$$H(w_1, \dots, w_{\tilde{N}}, \quad b_1, \dots, b_{\tilde{N}}, \quad x_1, \dots, x_N) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_{\tilde{N}} \cdot x_{\tilde{N}} + b_{\tilde{N}}) \\ \dots & \dots & \dots \\ g(w_1 \cdot x_N + b_1) & \dots & g(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{Nx\tilde{N}},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \dots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{Nx\tilde{N}}, \text{ and } T = \begin{bmatrix} T_1^T \\ \dots \\ T_{\tilde{N}}^T \end{bmatrix}_{Nx\tilde{N}}$$

H is called the neural network output matrix.

Thus, according to Huang et al. (2006), the learning method for SLFNs called ELM can be summarized as follows:

Given a training set $N=\{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i=1,\dots,N\}$, activation function g(x), and hidden node number Ñ, ELM

1) Randomly assigns value to input weight w_i and bias b_i, $i = 1, \dots, \tilde{N}$
2) Calculates the hidden layer output matrix H.
3) Finds the output weight β, using $\beta = H\dagger T$, where β, H and T are defined in the same way they were defined in the SLFN specification above.

Usage

ELM algorithm is available in multiple versions, such as Matlab, C/C++, Python or Java, among others. In this project, Matlab version has been used (the code is included in the CD), whose basic usage is explained below:

$$[Train_{Time}, Test_{Time}, Train_{Accuracy}, Test_{Accuracy}] = elm\left(Train_{Data}, Test_{Data}, Elm_{Type}, NHidden_{Neurons}, F_{activation}\right)$$

where inputs are

- **Train_Data:** Filename of training data set
- **Test_Data:** Filename of testing data set
- **Elm_Type:** ELM as functional approximators or classifiers
  - 0 for regression
  - 1 for classification. This type will be used in this project..
- **N_Hidden_Neurons:** Number of hidden neurons assigned to the ELM
  As explained before in section *4.1.1.What is an artificial neural network?*, every ANN has three types of layers: input, hidden, and output. Below, the number of neurons in each layer is defined.
  - *Input layer*
    The number of neurons comprising this layer is equal to the number of features (columns) in the database. Therefore, as explained in section *3. 3.2. Testing the algorithm*, there are the following five columns in the database ("Driving situations.xls", included in the CD): speed in m/s (2^{nd} column), acceleration in x-axis (3^{th} column), acceleration in y-axis (3^{th} column), G Acceleration in y-axis in G (4^{th} column), angular rate in degrees/second (5^{th} column) and number of activated sensors (6^{th} column).
  - *Output layer*
    Like the Input layer, every ANN has exactly one output layer. Its size or number of neurons is determined by the chosen model configuration. In this case, this ANN is a classifier, so this layer has a single node, which can take two values, 0 (No accident) or 1 (Accident), and belongs to the first column (Label) in the database ("Driving situations.xls", included in the CD).
  - *Hidden layer*
    The hidden layer can be configured by using two rules: (1) number of hidden layers equals one; and (2) the number of neurons in this layer is the mean of the neurons in the input and output layers. Therefore, as there are 5 inputs neurons and one output neuron, the number of neurons in hidden layer is equals to three.
- **F_Activation:** Type of activation function:
  - *'sig'* for Sigmoidal function:

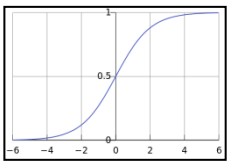


Figure 42. Sigmoidal function

  - *'hardlim'* for Hardlim function:

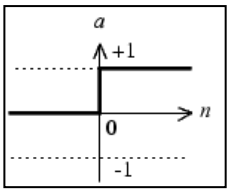


Figure 43. Hardlim function

- *'sin'* for Sine function:

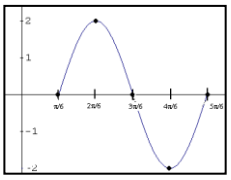


Figure 44. Sine function

- *'tribas'* for Triangular basis function:

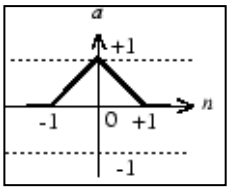


Figure 45. Triangular basis function

and the outputs:

- **Train_Time:** CPU Time (seconds) spent on training ELM
- **Test_Time:** CPU Time (seconds) spent on predicting ALL testing data
- **Train_Accuracy:** Training accuracy: RMSE for regression or correct classification rate for classification
- **Test_Accuracy:** Testing accuracy: RMSE for regression or correct classification rate for classification

Important notes regarding input data format of training and testing files:

- All input attributes (except expected targets) should be normalized into the range: [-1, 1].
- Training and testing files are text files, each raw consisting of information of one instance.
- First column are the expected output (target) for regression and classification applications, the rest columns consist of different attributes information of each instance.

For further information, refer to section *4.2.4.Testing the algorithm*.

4.2.3 Suitability study

Advantages

The ELM has several interesting and significant features different from traditional popular gradient-based learning algorithms for feed-forward neural networks: These include the following:

- The learning speed of ELM is extremely fast. In simulations reported in literatures, Huang et al.(2006), the learning phase of ELM can be completed in seconds or less than seconds for many applications. Previously, it seems that there exists a virtual speed barrier which most (if not all) classic learning algorithms cannot break through and it is not unusual to take very long time to train a feed-forward network using classic learning algorithms even for simple applications.
- ELM has better generalization performance than the gradient-based learning, such as back-propagation, in most cases. Traditional classic gradient-based learning algorithms and some other learning algorithms may face several issues like local minima, improper learning rate and over fitting, etc. In order to avoid these issues, some methods such as weight decay and early stopping methods may need to be used often in these classical learning algorithms.
- ELM tends to reach the solutions straightforward without such trivial issues. It looks much simpler than most learning algorithms for feed-forward neural networks. Unlike the traditional classic gradient-based learning algorithms which only work for differentiable activation functions, as easily observed the ELM learning algorithm could be used to train SLFNs with many non-differentiable activation functions.

Limitations

However, like any technique, ELM has certain drawbacks and open problems. The most relevant are exposed below.

- As observed in experimental studies, the performance of basic ELM is stable in a wide range of number of hidden nodes. Compared to the back-propagation (BP) learning algorithm, the performance of basic ELM is not very sensitive to the number of hidden nodes. However, how to prove it in theory remains open.
- One of the typical implementations of ELM is to use random nodes in the hidden layer and the hidden layer of SLFNs need not be tuned. Generalization performance of ELM turns out to be very stable. How to estimate the oscillation bound of the generalization performance of ELM remains open too.
- It seems that ELM performs better than other conventional learning algorithms in applications with higher noise. How to prove it in theory is not clear.
- It is not clear that ELM always has faster learning speed than Least Squares Support Vector Machine (LS-SVM) if the same kernel is used.
- ELM provides a batch learning kernel solution which is much simpler than other kernel learning algorithms such as LS-SVM. It is known that it may not be straightforward to have an efficient online sequential implementation of SVM and LS-SVM. However, due to the simplicity of ELM, is not clear if it is possible to implement the online sequential variant of the kernel based ELM.

- Is not clear that ELM always provides similar or better generalization performance than SVM and LS-SVM if the same kernel is used (if not affected by computing devices' precision).

Conclusions

Table below summarizes the characteristics that ELM should fulfils to be compatible with current crash detection system.

Characteristics	**ANN suitability study for Moto eCall**	**ELM suitability study for Moto eCall**
Types of neural networks • *Feedback* • *Feed-forward*	Feed-forward	ELM is a learning method for SLFNs (Single-hidden Layer **Feed-forward** neural Networks)
Network layers • *Single-layer* • *Multi-layer*	Single-layer	ELM is a learning method for SLFNs (**Single-hidden Layer** Feed-forward neural Networks)

Learning Process
• *Associative mapping*
  ○ Auto-association
  ○ Hetero-association
• *Regularity detection*
	Associative mapping with auto-association	ELM learning process is auto-associative

Performed learning
• *Fixed networks*
• *Adaptive networks*
  ○ Supervised learning
  ○ Unsupervised learning
	Adaptive networks with unsupervised learning	It randomly selects the input weights and analytically determines the output weights of SLFNs

Transfer Function • *Linear* • *Threshold* • *Sigmoid*	Sigmoid, but all three must be considered rough approximations	Several types of activation functions: sigmoidal, sine, hardlim, triangular basis and Radial basis functions

Table 11. ELM suitability study

4.2.4 Testing the algorithm

In this section, ELM algorithm is going to be tested by using the aforesaid situations used for testing Moto eCall algorithm. Further details and information, refer to sections *2.Accident detection* and *3.Motorcycle crash detection system*.

The Matlab code designed to test ELM is called "test_elm.m" (included in the CD), whose basic usage is explained below:

$$[Train_{Time}, Test_{Time}, Total_{Time}] = test\_elm\ (Test_{Situation}, F_{activation})$$

where the inputs are:

- **Test_Situation:** Situation to be tested:
  - 1 for 1st situation (acceleration)
  - 2 for 2nd situation (deceleration)
  - 3 for 3rd situation (curve)
  - 4 for 4th situation (frontal impact at 120 km/h)
  - 5 for 5th situation (frontal impact at 50 km/h)
  - 6 for 6th situation (rear impact while motorcycle is stopped)
  - 7 for 7th situation (rear impact while motorcycle is in motion)
  - 8 for 8th situation (side impact while motorcycle is stopped)
  - 9 for 9th situation (side impact while motorcycle is in motion)
  - 10 for 10th situation (front-side impact in an angle of 30º)
  - 11 for 11th situation (front-side impact in an angle of 45º)
  - 12 for 12th situation (front-side impact in an angle of 70º)
  - 13 for 13th situation (slide side crash)
  - 14 for 14th situation (road exit)
- **F_Activation:** Type of activation function:
  - *'sig'* for Sigmoidal function
  - *'hardlim'* for Hardlim function
  - *'sin'* for Sine function
  - *'tribas'* for Triangular basis function

and the outputs:

- **Train_Time:** CPU Time (seconds) spent on training ELM
- **Test_Time:** CPU Time (seconds) spent on predicting all testing data
- **Total_Time:** CPU Time (seconds) spent on training ELM and predicting all testing data

The program makes the following operations:

1) Call to the function "create_database.m" (included in the CD) in order to normalize the input attributes according suggestions of ELM authors as explained in section *4.2.2.ELM Training algorithm:*
   - *Training file:* "Train_ELM.xls" normalized to "Training_file.dat"
   - *Testing file:* " Driving situations.xls" normalized to " Training_file.dat"
2) Call to the function "elm.m" (included in the CD), whose usage is explained in previous section *4.2.2.ELM Training algorithm*.
3) Print the results on the screen

4.2.4.1 Acceleration

Here a motorcycle accelerates from 0 to 120 Km/h in a straight line, within that is considered a normal driving. Further information can be found in section *3.2.1.Acceleration*. Following figures show the results of ELM algorithm with different approximation functions.

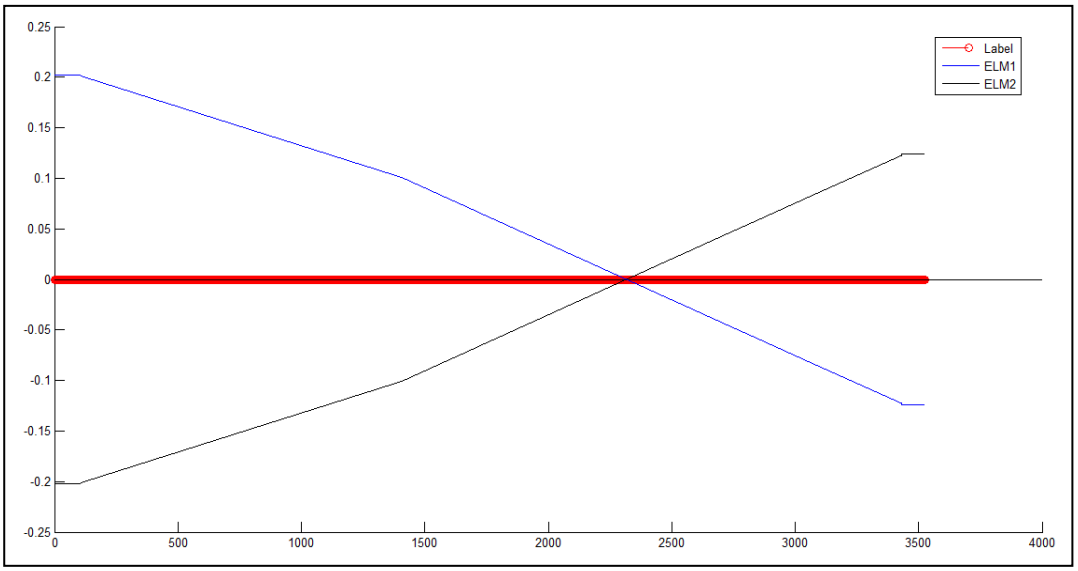


Figure 46. First situation, approx. function='sig'

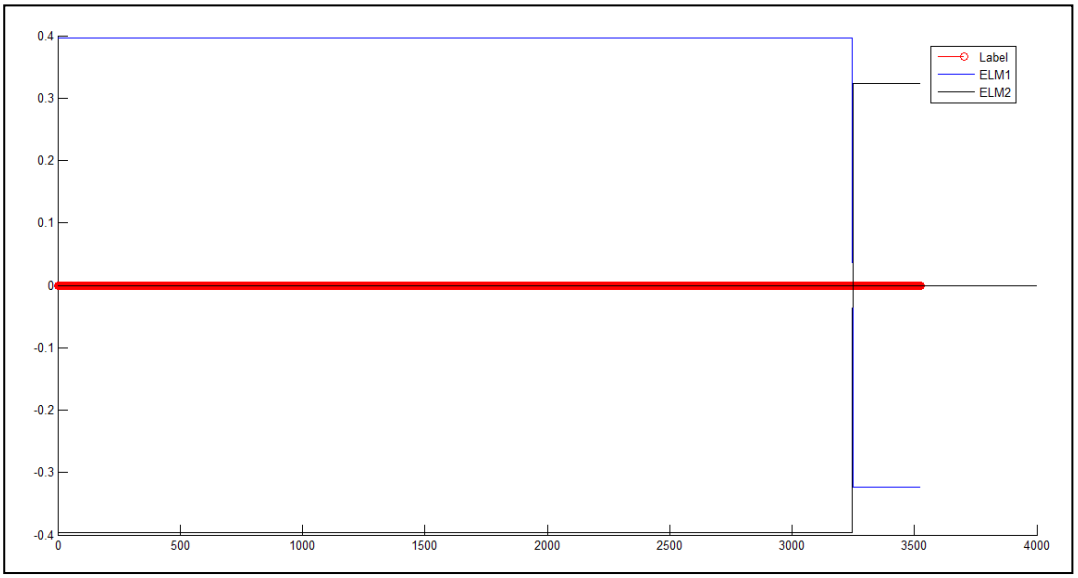


Figure 47. First situation, approx. function='hardlim'

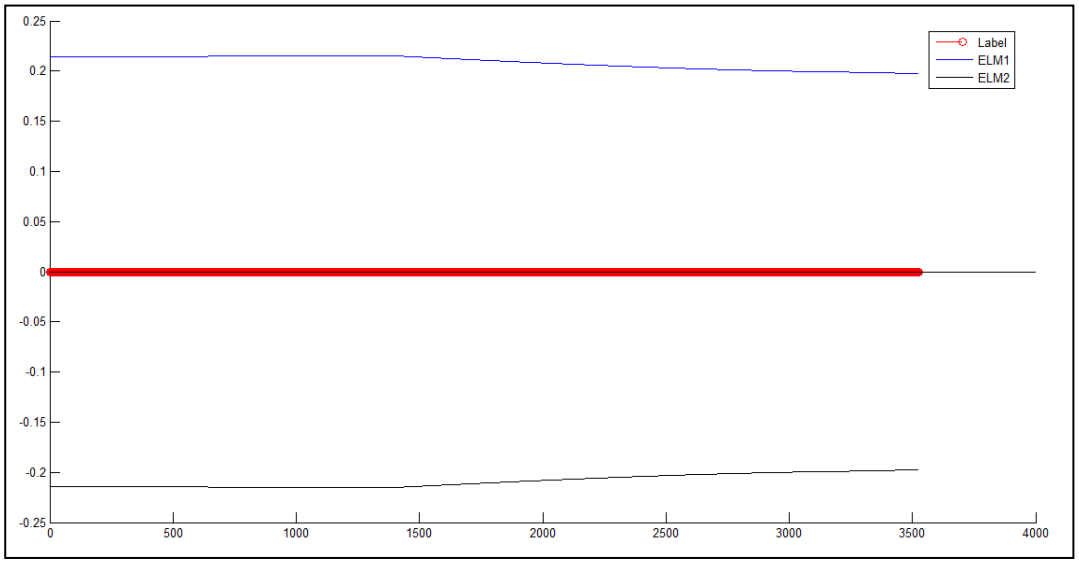


Figure 48. First situation, approx. function='sin'

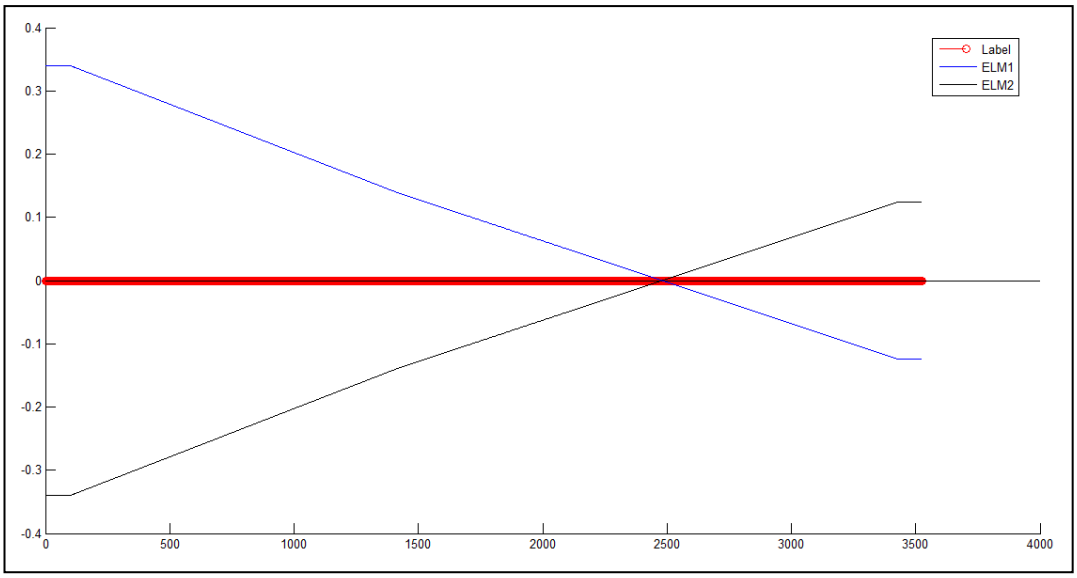


Figure 49. First situation, approx. function='tribas'

4.2.4.2 Deceleration

In this situation, it can be seen how a motorcycle brakes from 120 to 0 Km/h in a straight line, within that is considered a normal driving. For further information, please refer to section *3.2.2.Deceleration*. Figures below show the results of ELM algorithm with different approximation functions.

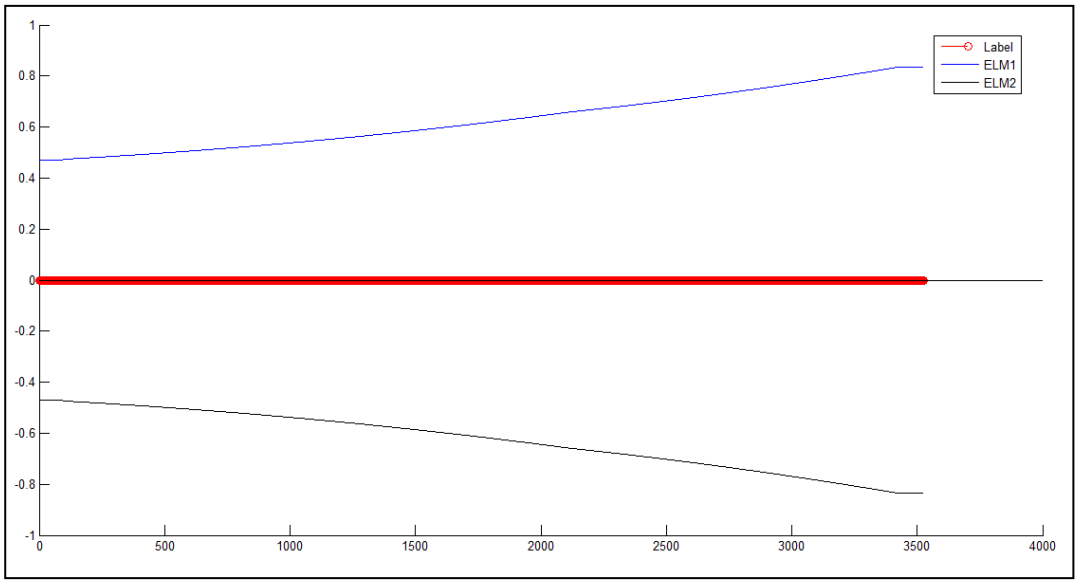


Figure 50. Second situation, approx. function='sig'

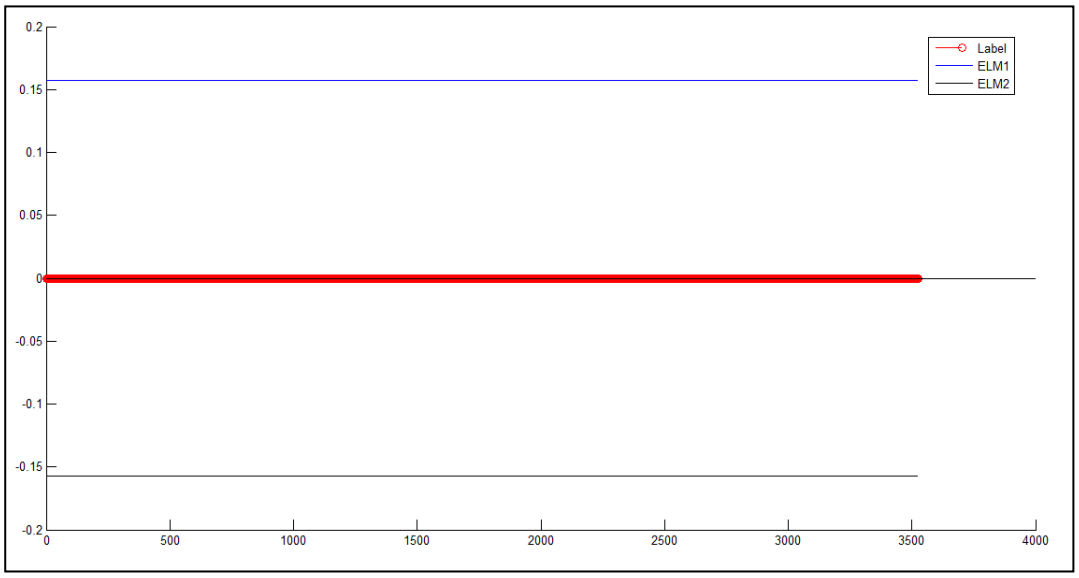


Figure 51. Second situation, approx. function='hardlim'

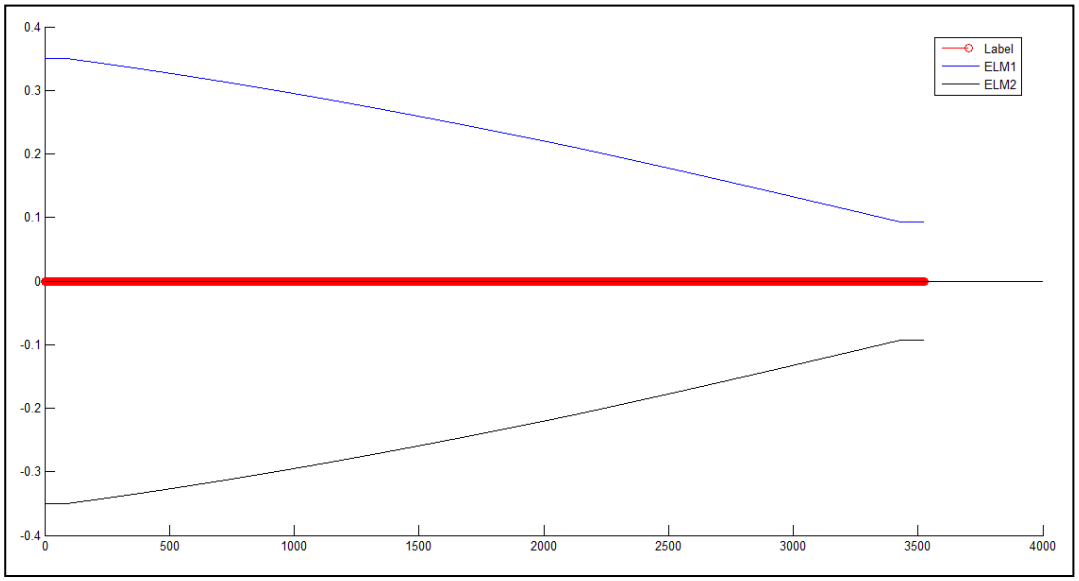


Figure 52. Second situation, approx. function='sin'

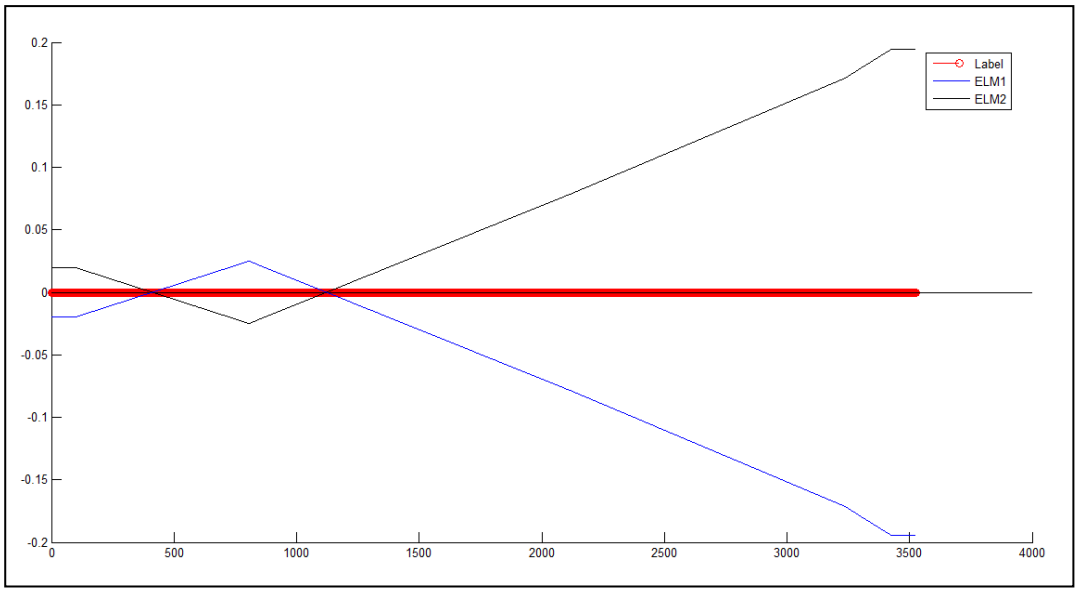


Figure 53. Second situation, approx. function='tribas'

4.2.4.3 Curve

In this scenario, a motorcycle takes a curve 200 metres radius at 120 Km/h, within that is considered a normal driving. More details in section *3.2.3.Curve*. Next figures show the results of ELM algorithm with different approximation functions.

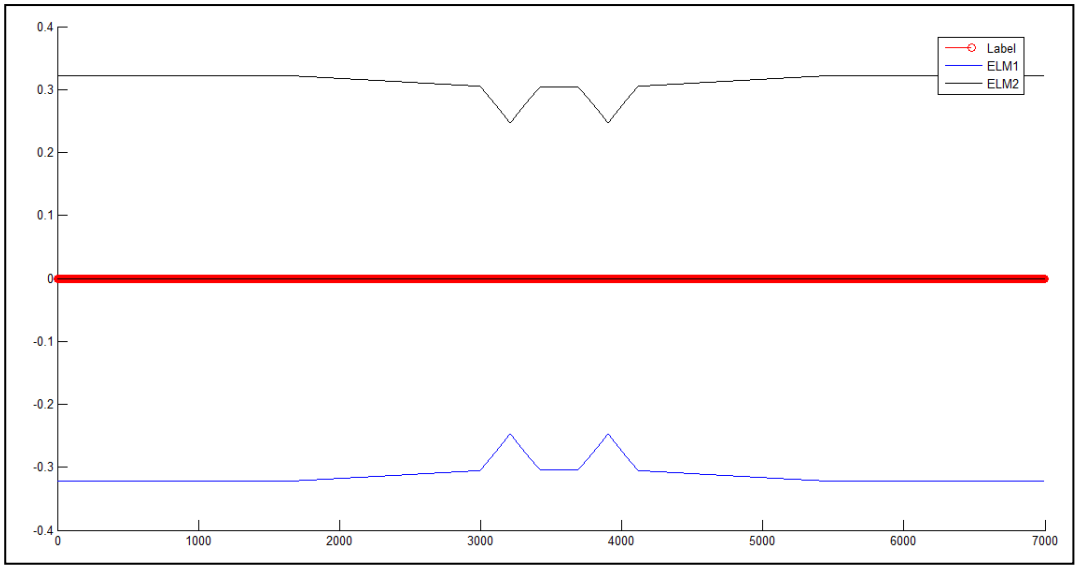


Figure 54. Third situation, approx. function='sig'

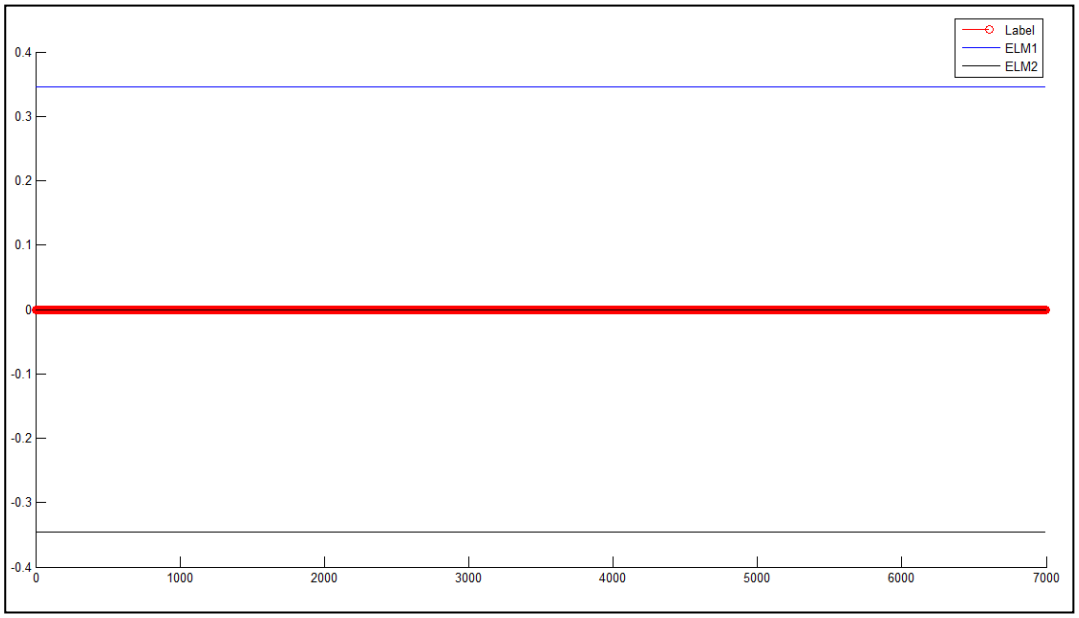


Figure 55. Third situation, approx. function='hardlim'

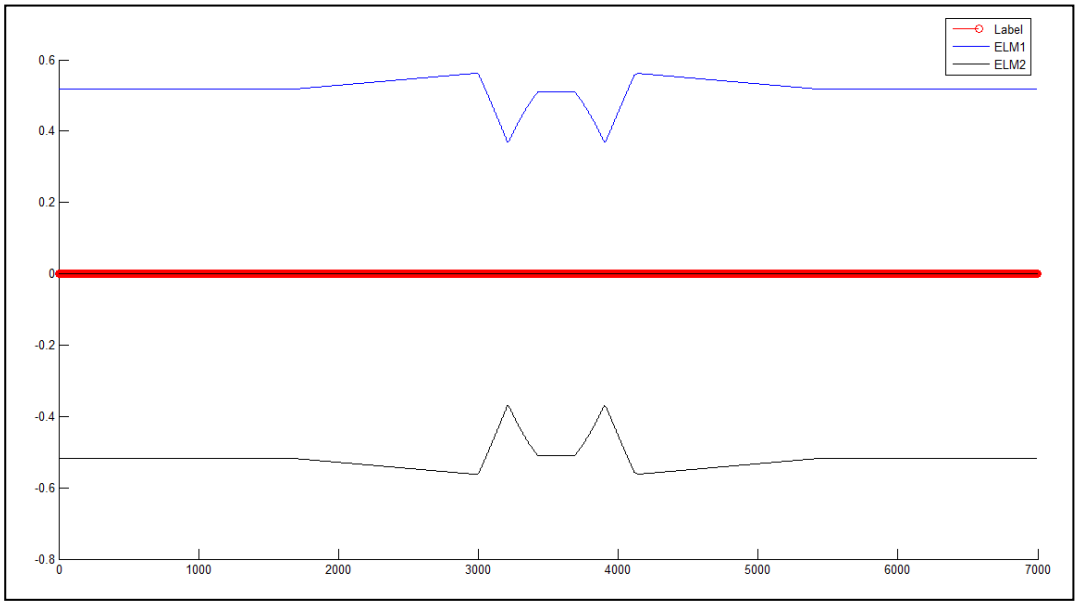


Figure 56. Third situation, approx. function='sin'

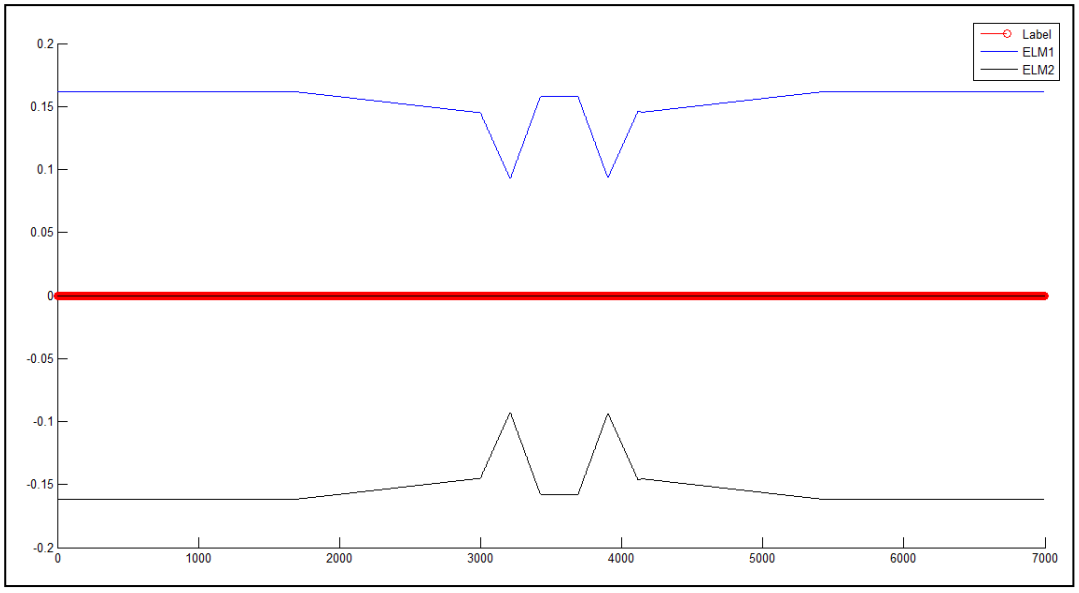


Figure 57. Third situation, approx. function='tribas'

4.2.4.4 Frontal impact at 120 Km/h

Here, a motorcycle suffers a frontal impact while driving at 120 Km/h, within that is considered a crash situation. For further information, please refer to section *3.2.4.Frontal impact at 120 Km/h*. Figures below show the results of ELM algorithm with different approximation functions.

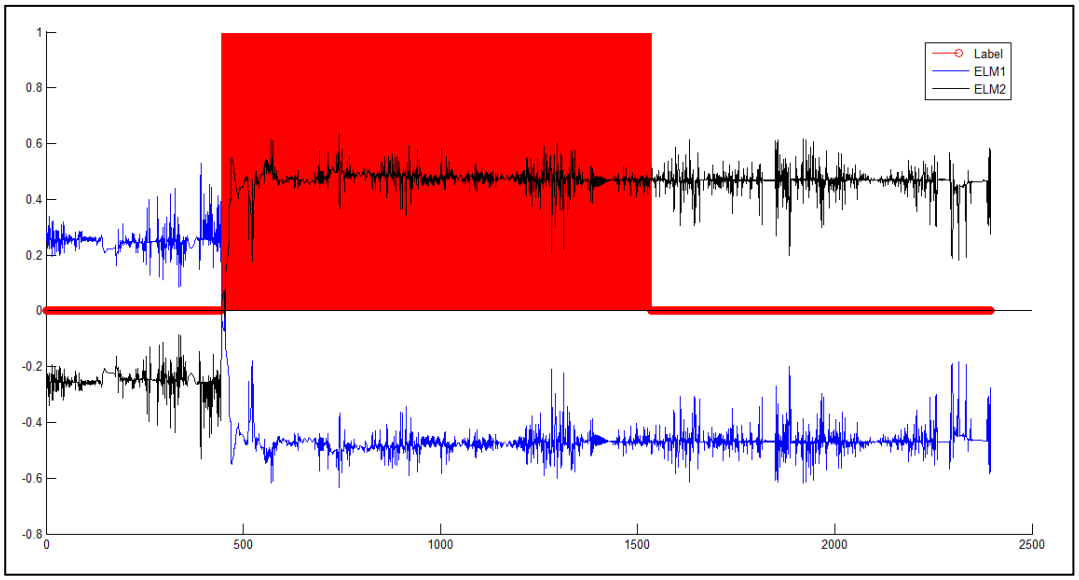


Figure 58. Fourth situation, approx. function='sig'

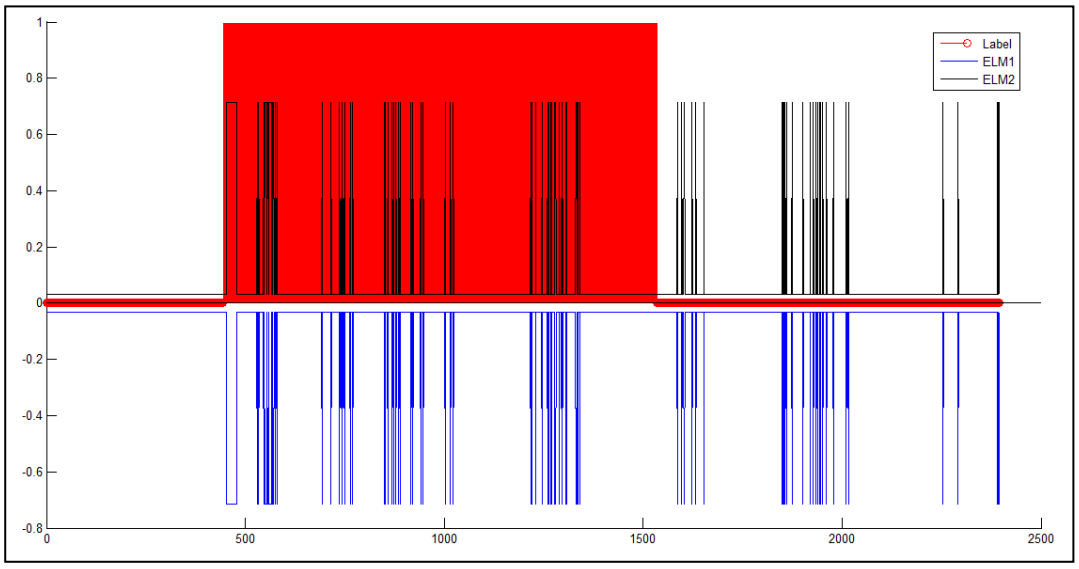


Figure 59. Fourth situation, approx. function='hardlim'

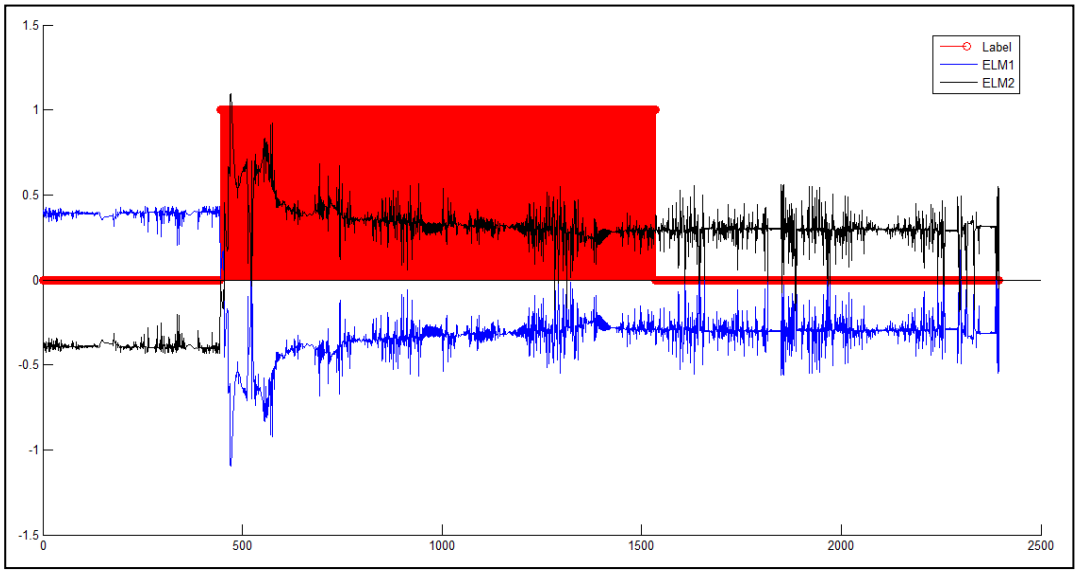


Figure 60. Fourth situation, approx. function='sin'

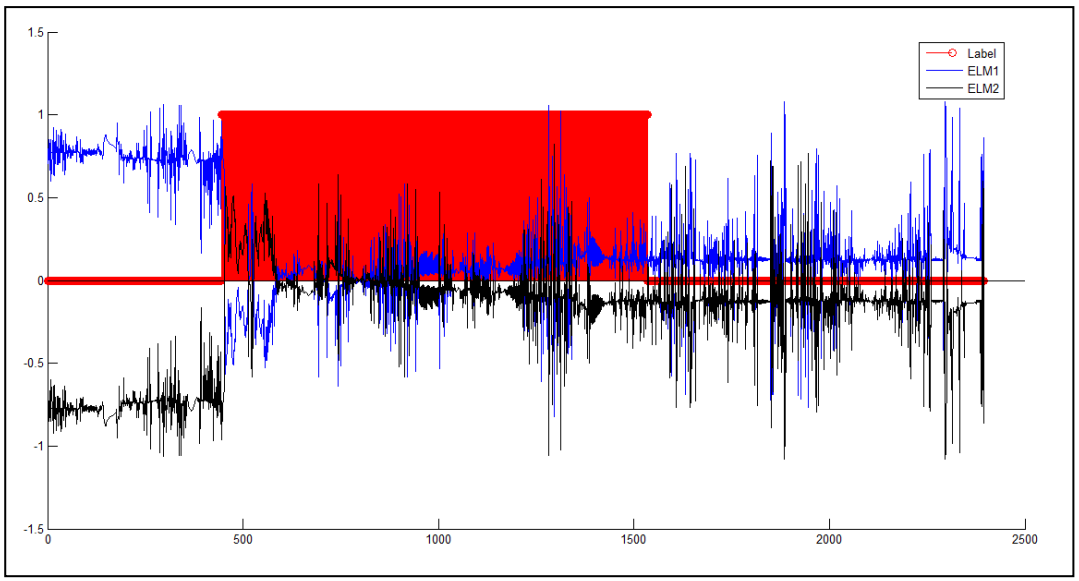


Figure 61. Fourth situation, approx. function='tribas'

4.2.4.5 Frontal impact at 50 Km/h

This situation shows how a motorcycle suffers a frontal impact while driving at 50 Km/h, within that is considered a crash situation. Further details in previous section *3.2.4.Frontal impact at 50 Km/h*. Next figures summarize the results of ELM algorithm with different approximation functions.

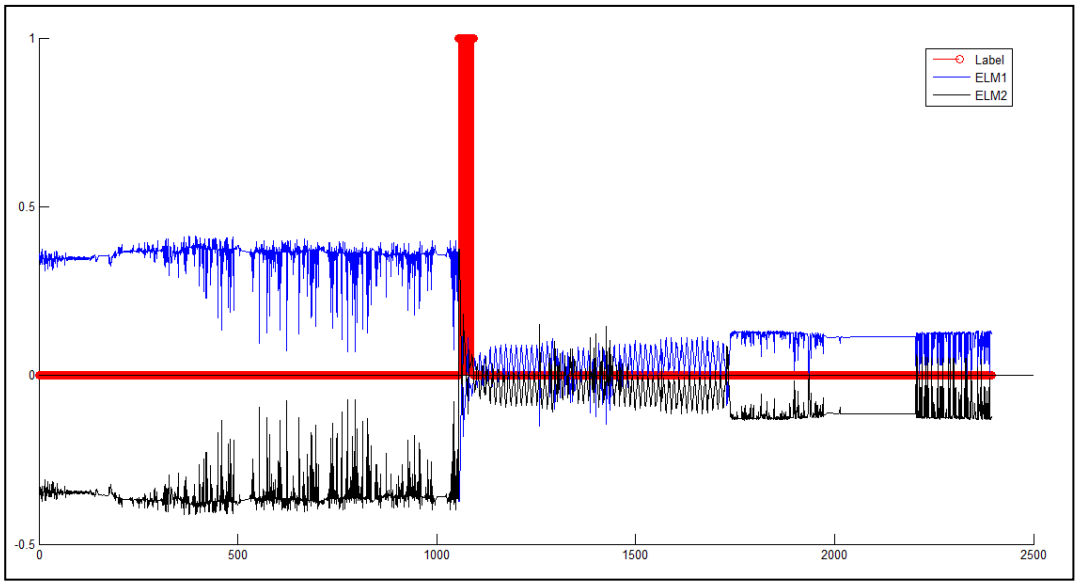


Figure 62. Fifth situation, approx. function='sig'

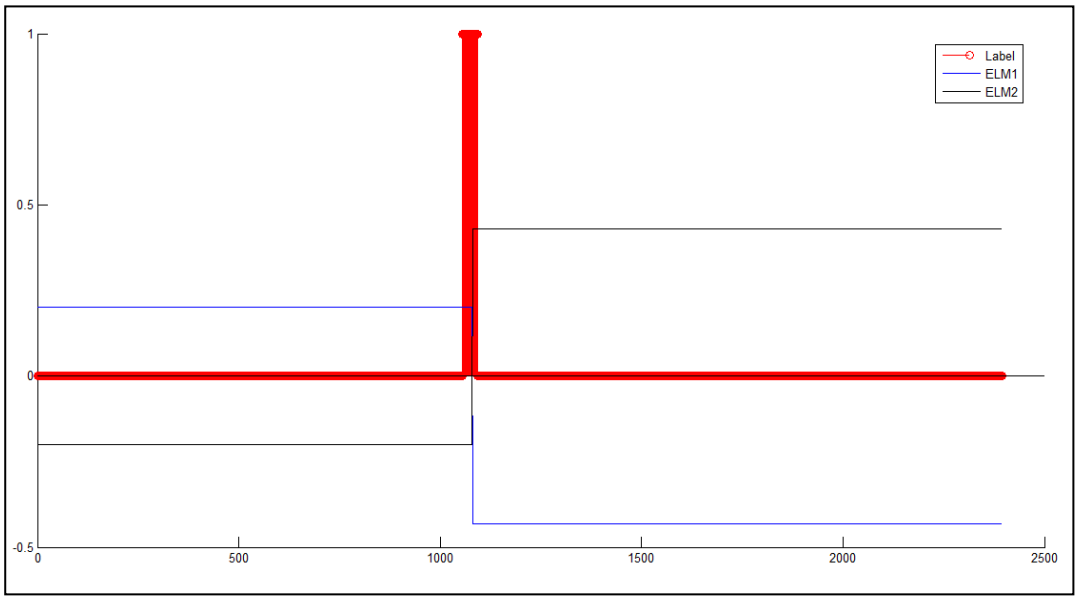


Figure 63. Fifth situation, approx. function='hardlim'

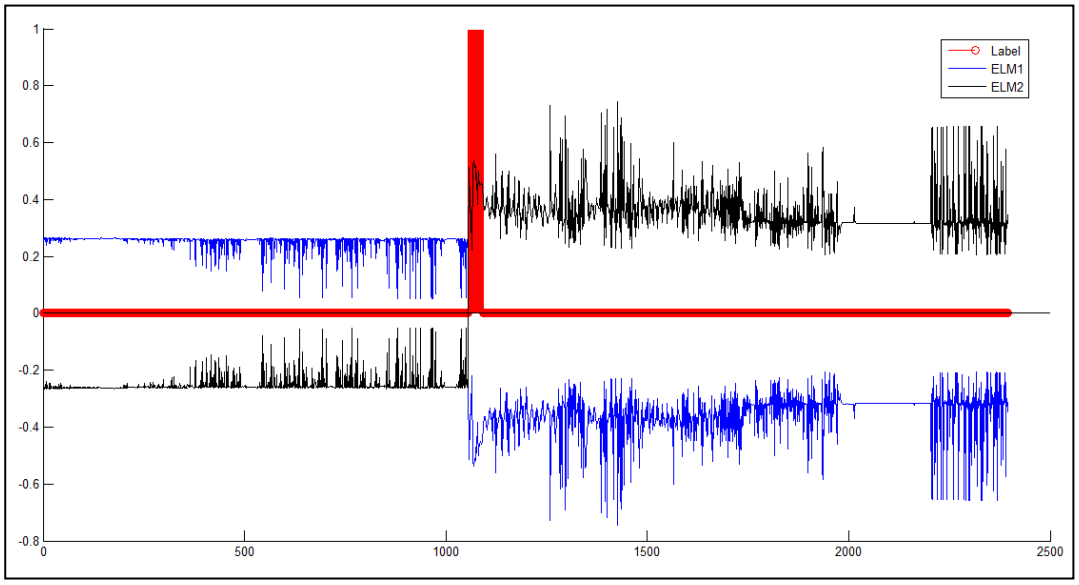


Figure 64. Fifth situation, approx. function='sin'

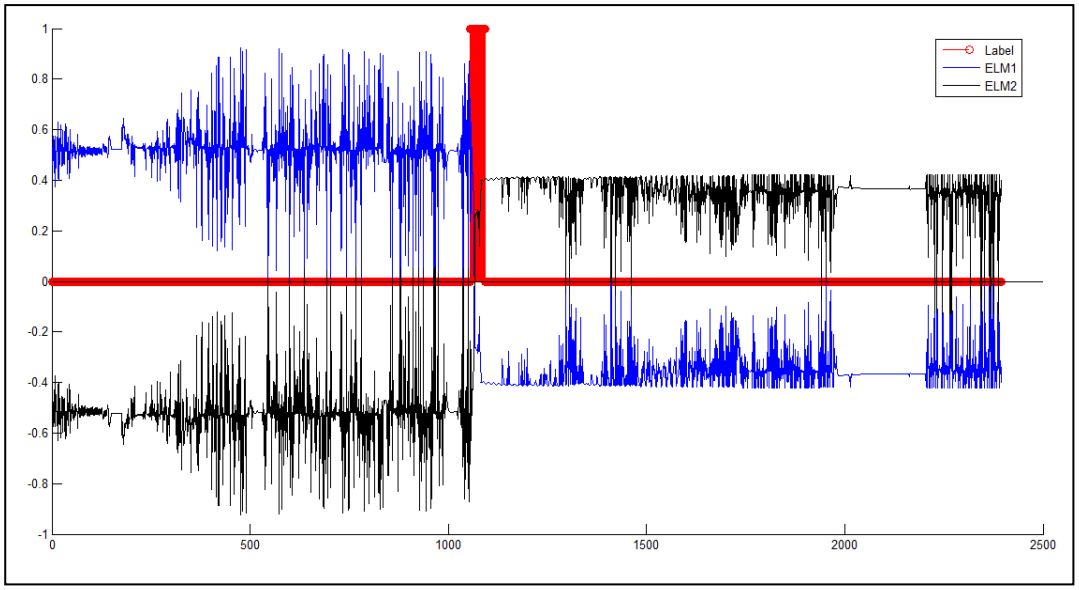


Figure 65. Fifth situation, approx. function='tribas'

4.2.4.6 Rear impact while the motorcycle is stopped

In this scenario, it can be seen how a motorcycle suffers a rear impact while standing at a traffic light, within that is considered a crash situation. For further information, please refer to section *3.2.6. Rear impact while the motorcycle is stopped*. Following figures show the results of ELM algorithm with different approximation functions.

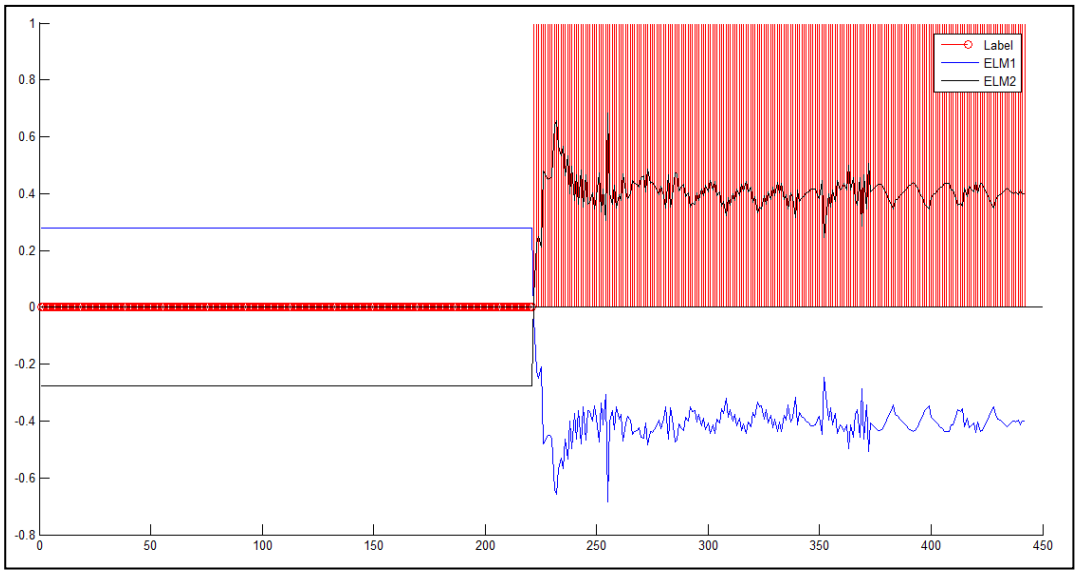


Figure 66. Sixth situation, approx. function='sig'

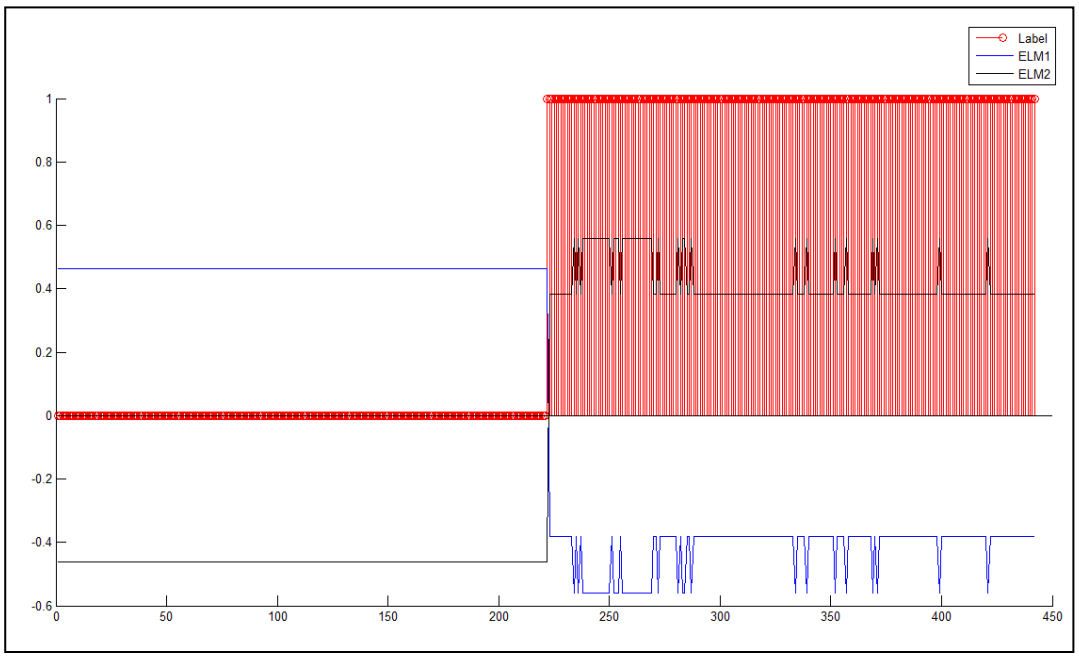


Figure 67. Sixth situation, approx. function='hardlim'

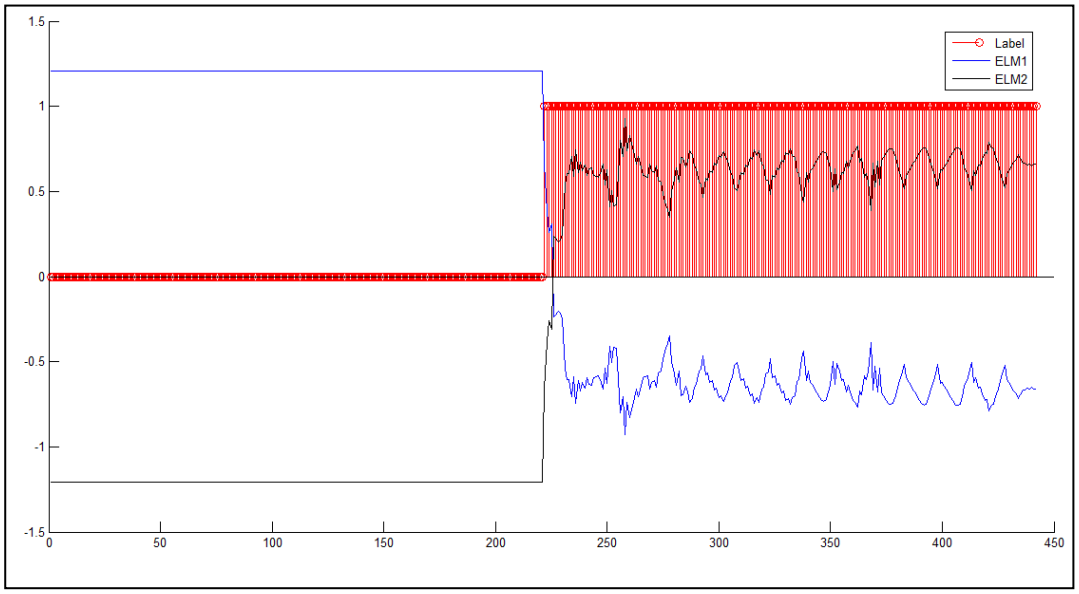


Figure 68. Sixth situation, approx. function='sin'

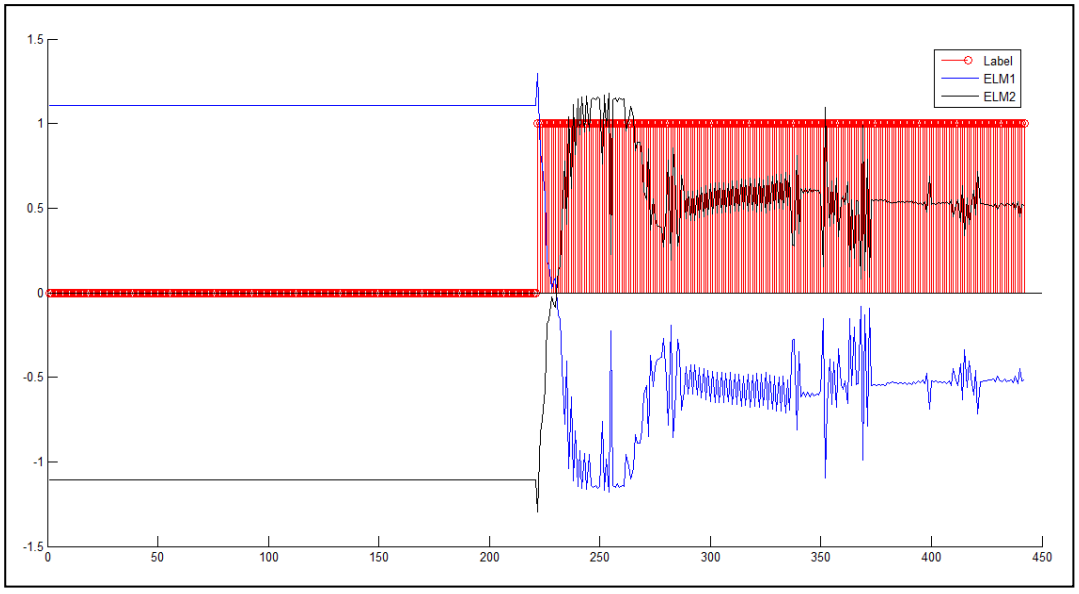


Figure 69. Sixth situation, approx. function='tribas'

4.2.4.7 Rear impact while the motorcycle in motion

Here, a motorcycle suffers a rear impact while driving at 25 Km/h, within that is considered a crash situation. More details in section *3.2.7.Rear impact while the motorcycle is in motion*. Pictures below show the results of ELM algorithm with approximation functions.

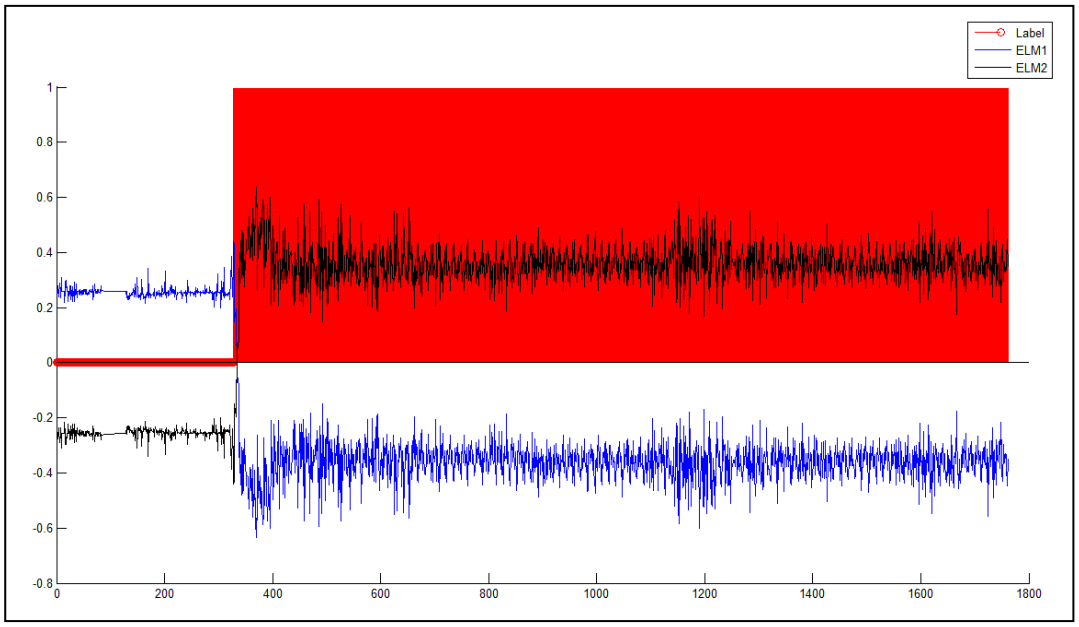


Figure 70. Seventh situation, approx. function='sig'

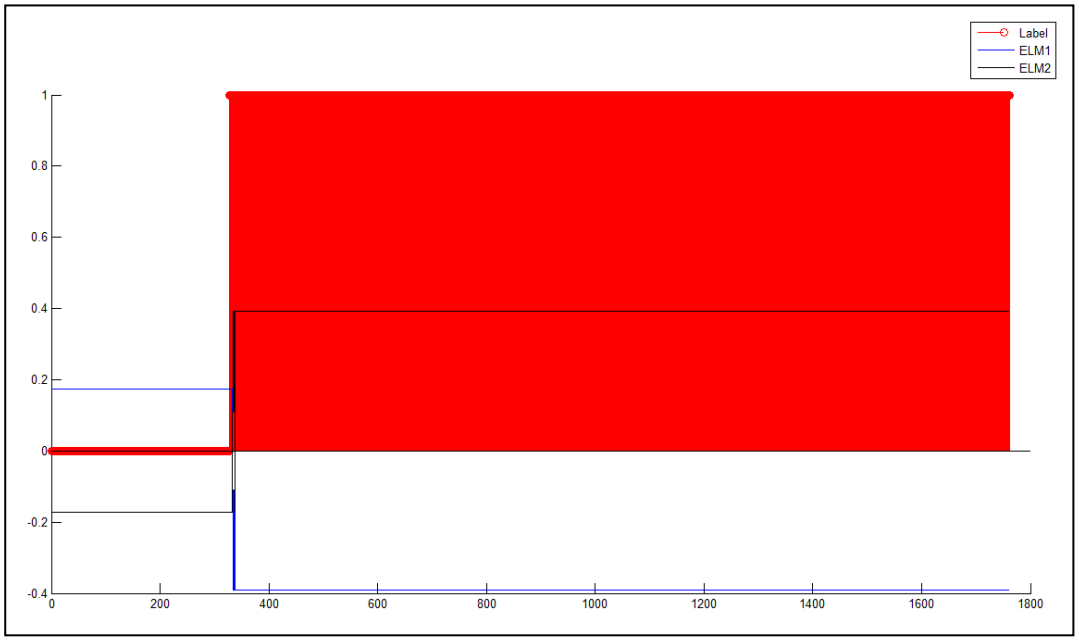


Figure 71. Seventh situation, approx. function='hardlim'

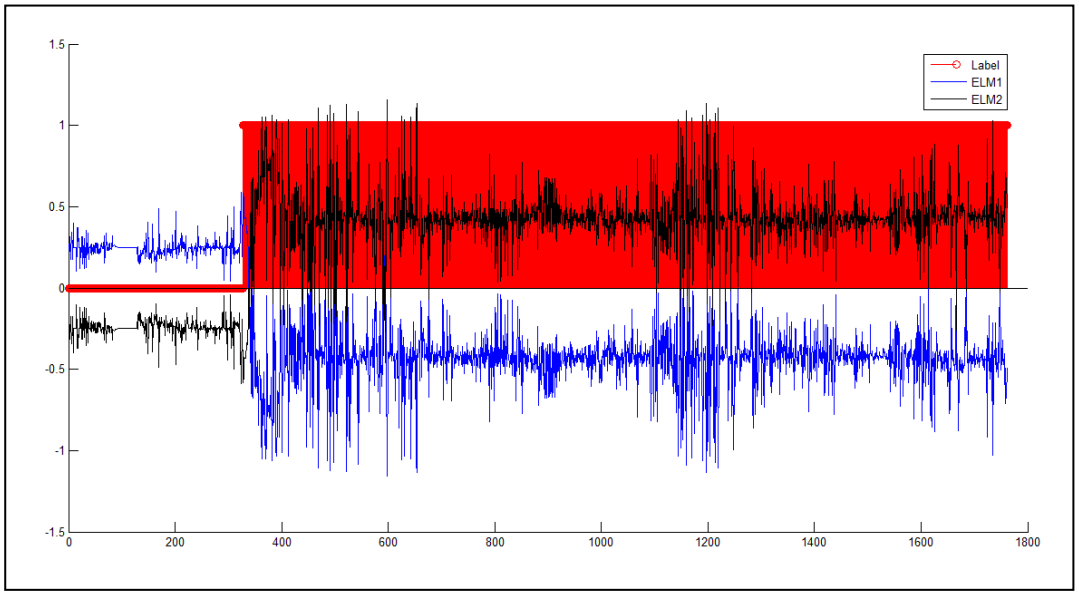


Figure 72. Seventh situation, approx. function='sin'

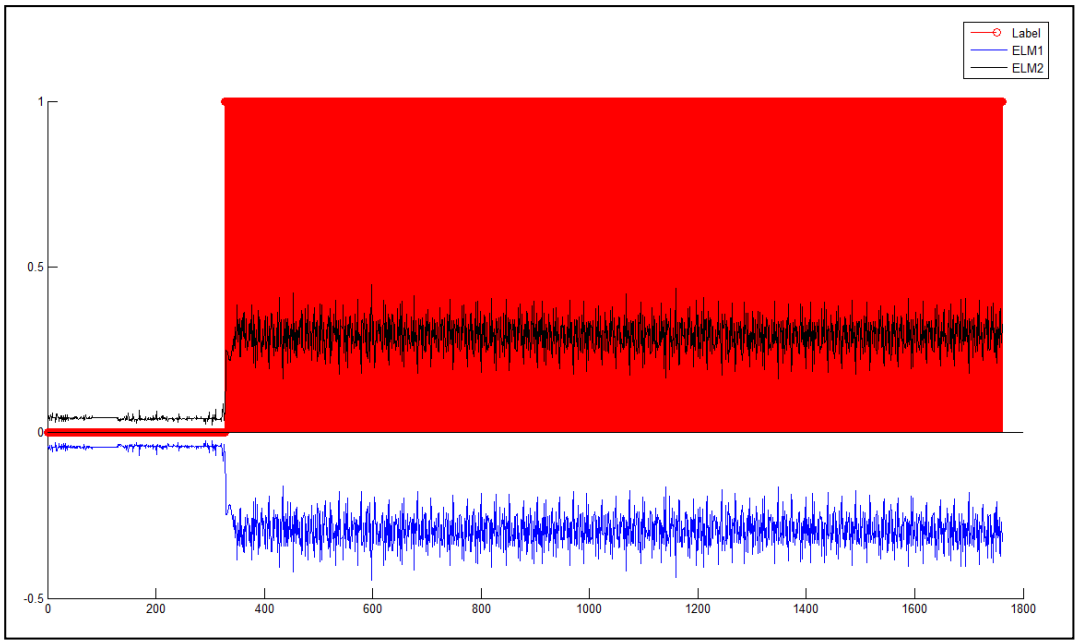


Figure 73. Seventh situation, approx. function='tribas'

4.2.4.8 Side impact while the motorcycle is stopped

Here, a motorcycle suffers a side impact while standing at a traffic light, within that is considered a crash situation. For further information, please refer to section *3.2.8.Side impact while the motorcycle is stopped*. Following figures show the results of ELM algorithm with approximation functions.

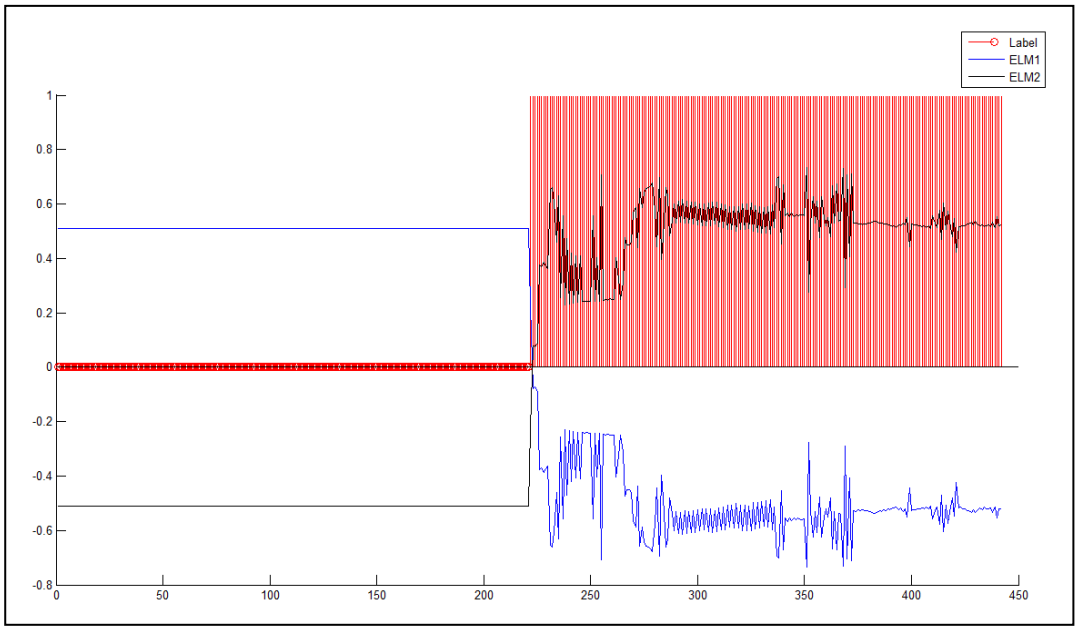


Figure 74. Eighth situation, approx. function='sig'

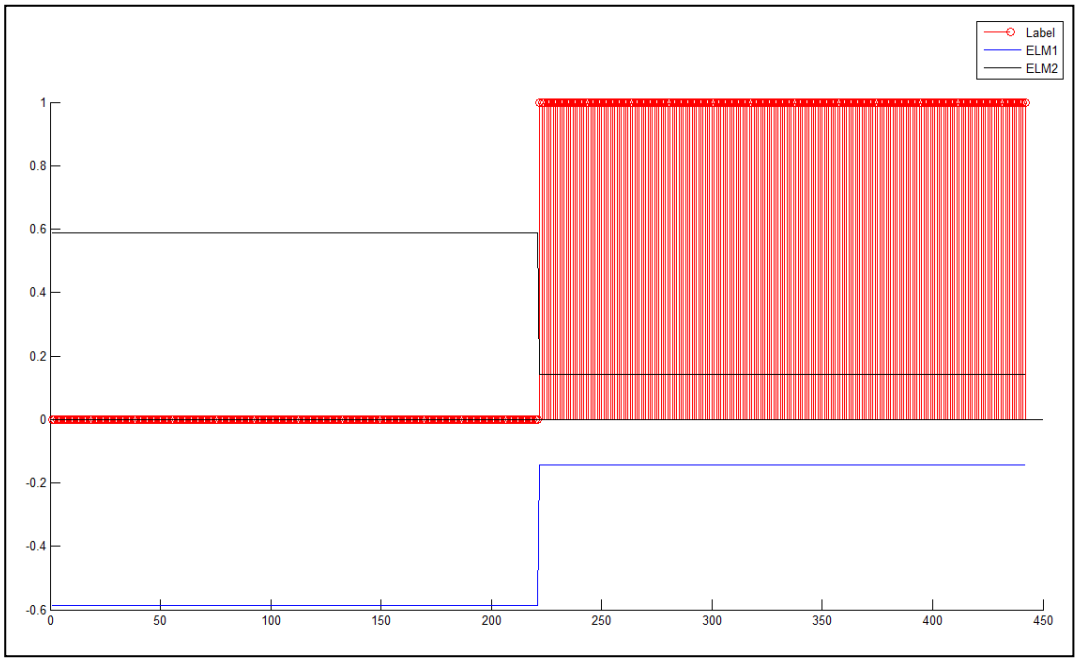


Figure 75. Eighth situation, approx. function='hardlim'

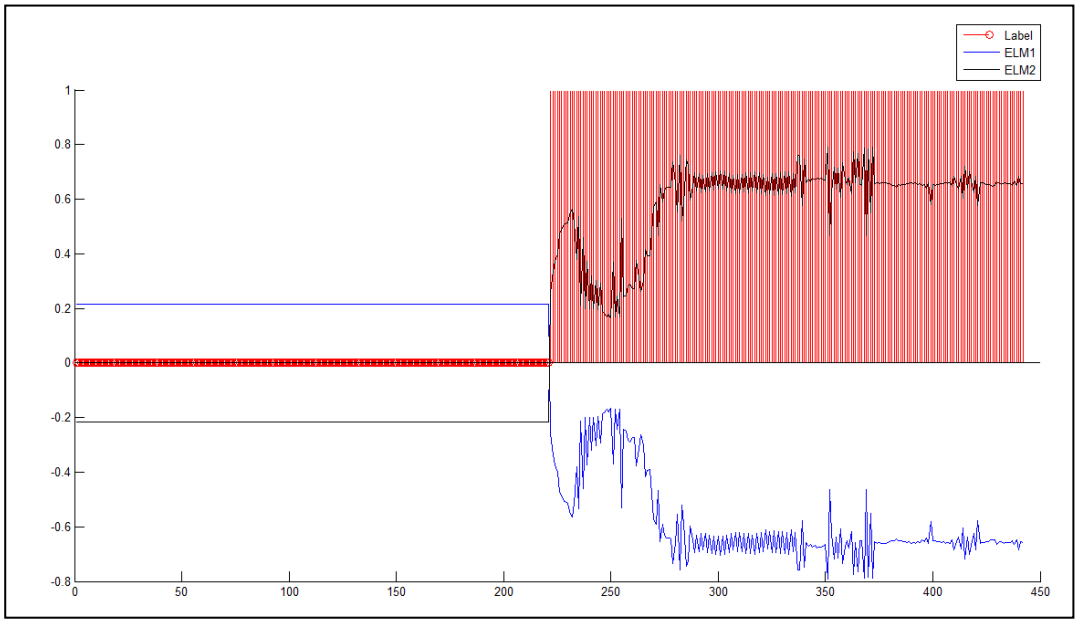


Figure 76. Eighth situation, approx. function='sin'

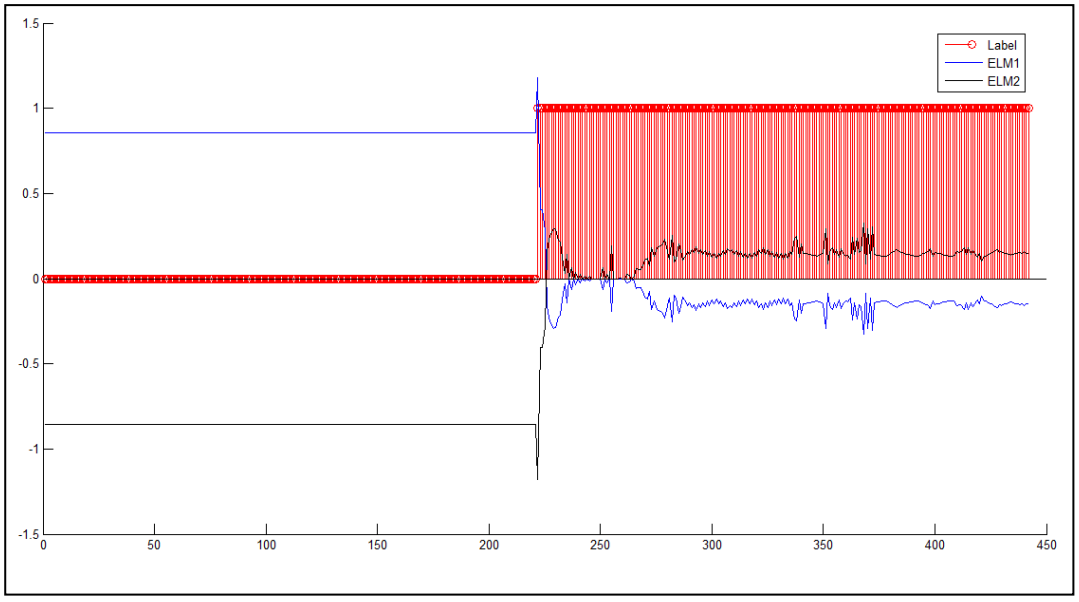


Figure 77. Eighth situation, approx. function='tribas'

4.2.4.9 Side impact while the motorcycle is in motion

In this situation a motorcycle suffers a side while crossing an intersection, within that is considered a crash situation. More details in section *3.2.9. Side impact while the motorcycle is in motion*. Following figures show the results of ELM algorithm with different approximation functions.

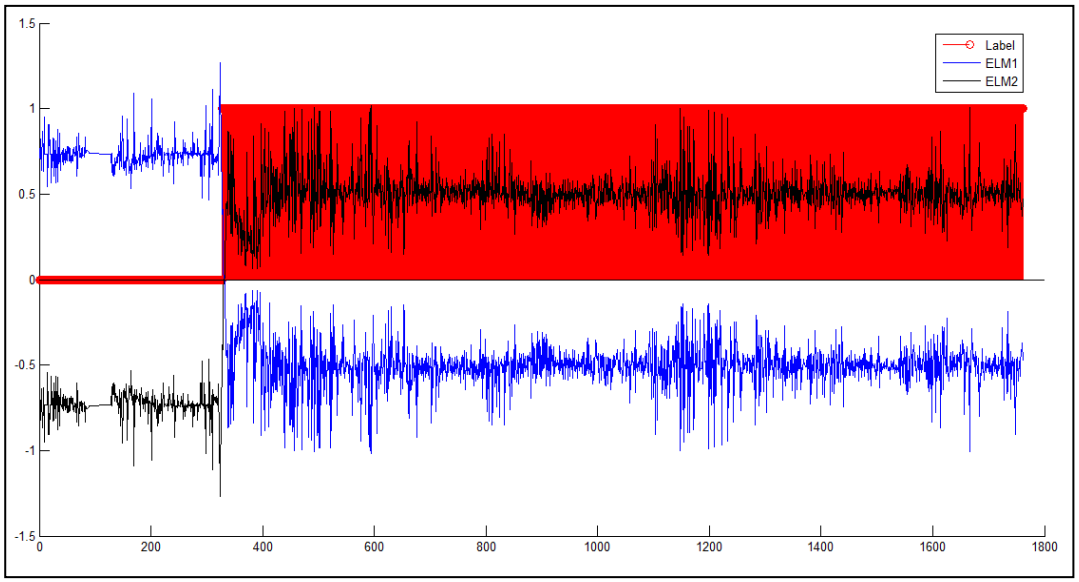


Figure 78. Ninth situation, approx. function='sig'

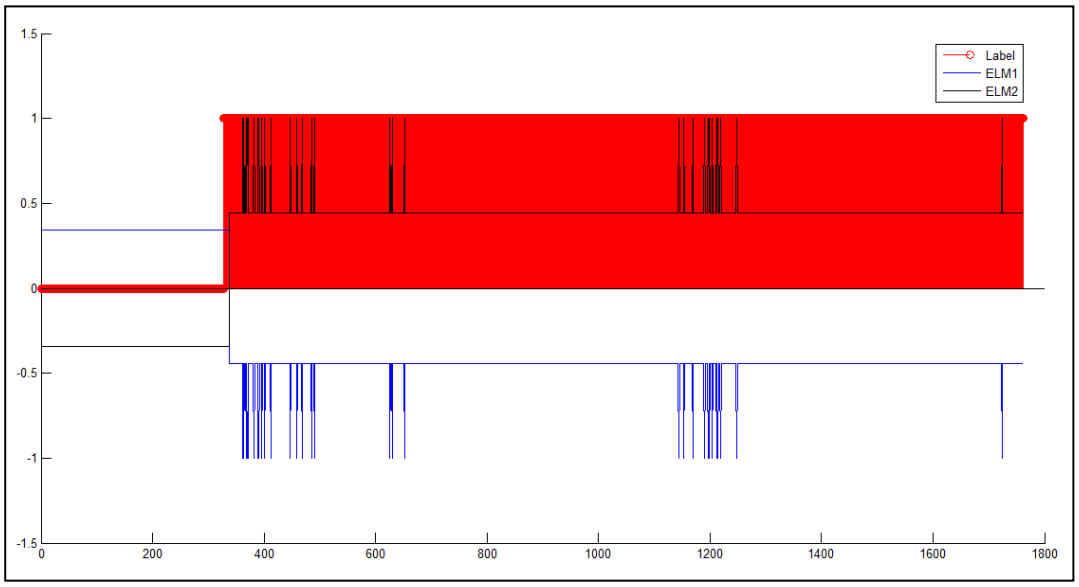


Figure 79. Ninth situation, approx. function='hardlim'

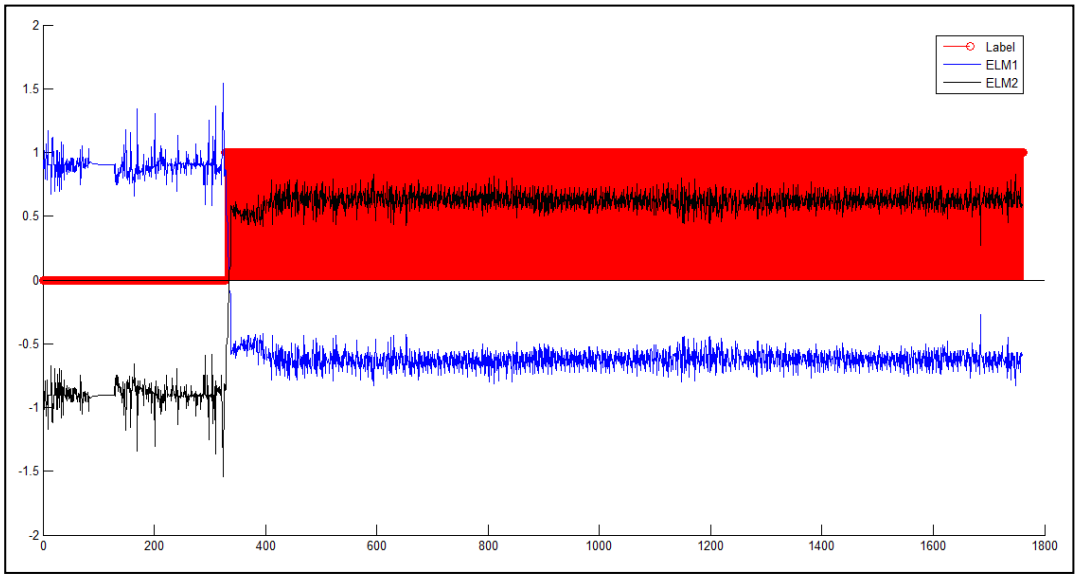


Figure 80. Ninth situation, approx. function='sin'

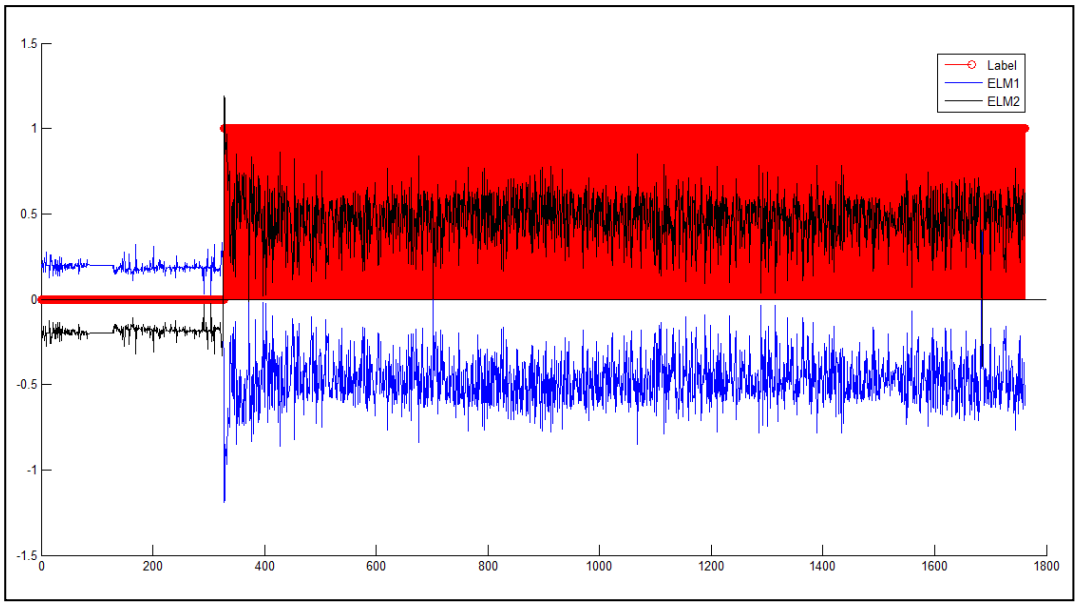


Figure 81. Ninth situation, approx. function='tribas'

4.2.4.10 Front-side impact in an angle of 30°

In this situation, it can be seen how a motorcycle suffers a front-side impact in an angle of 30° with the other vehicle while crossing an intersection, within that is considered a crash situation. For further information, please refer to *section 3.2.10.Front-side impact in an angle of 30°*. Next graphics represent the results of ELM algorithm with different and approximation functions.

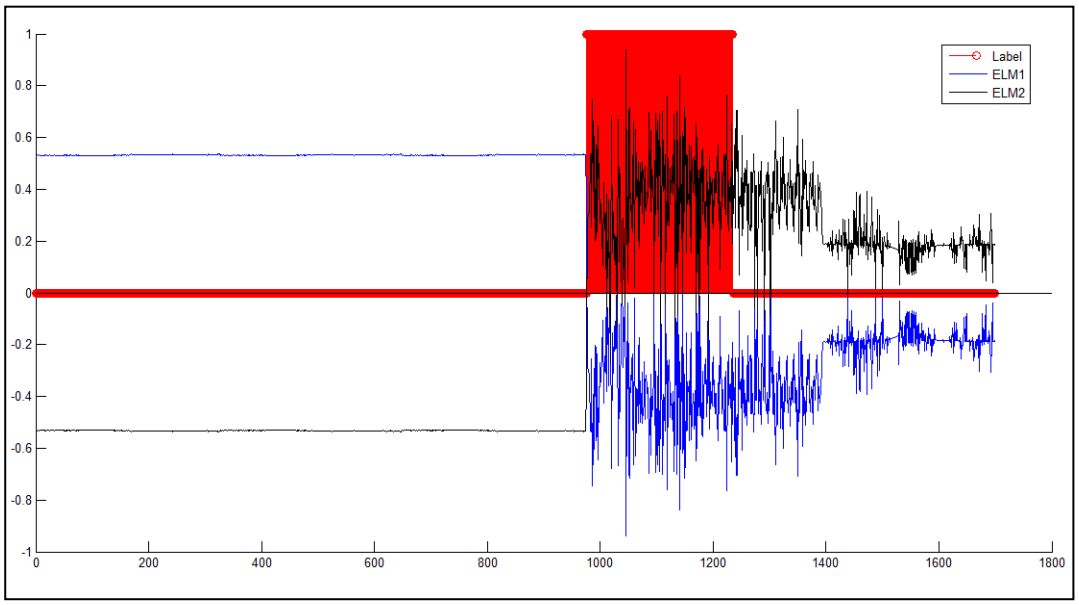


Figure 82. Tenth situation, approx. function='sig'

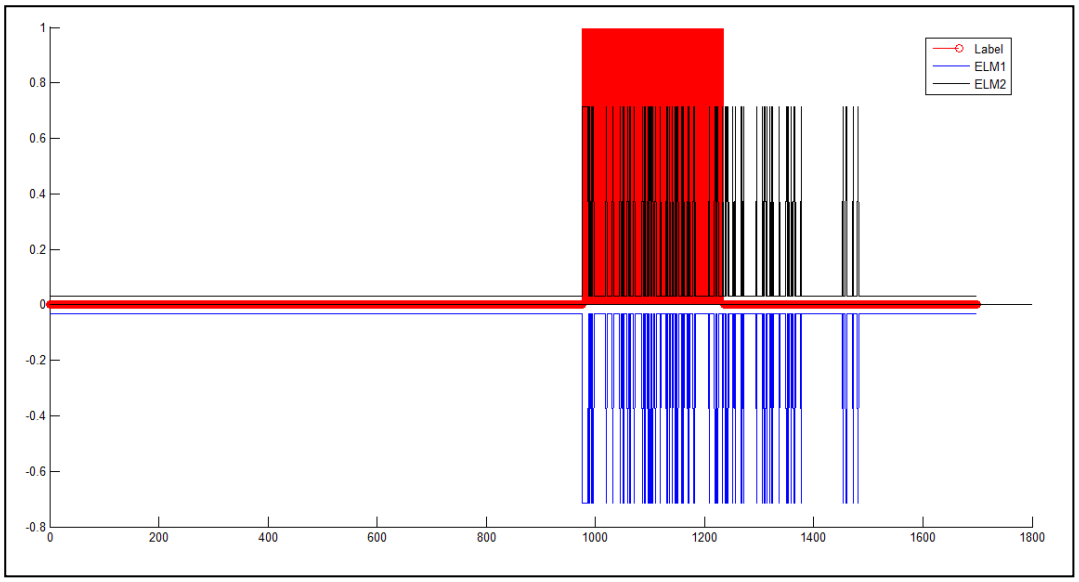


Figure 83. Tenth situation, approx. function='hardlim'

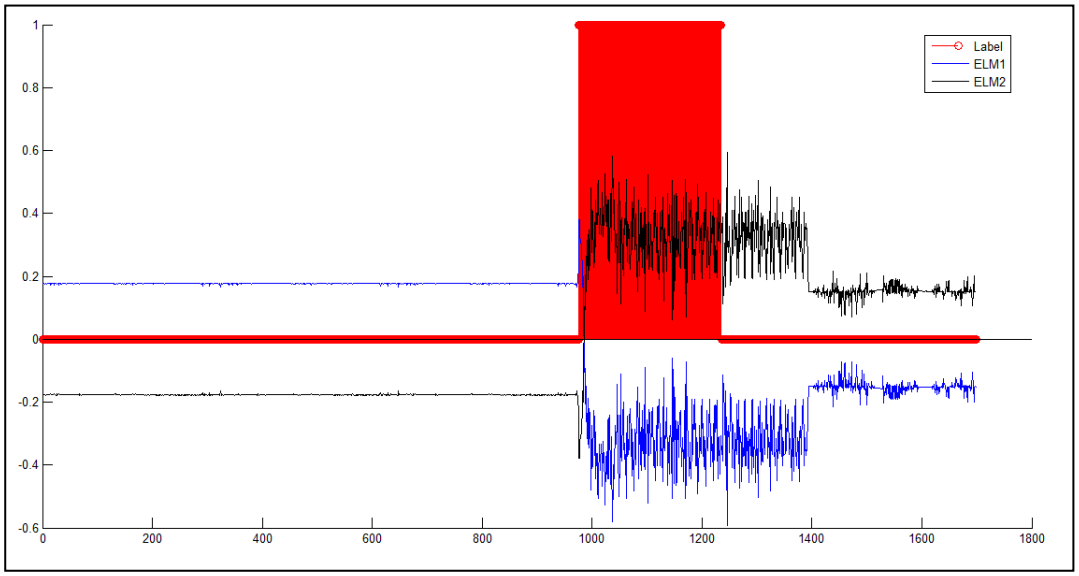


Figure 84. Tenth situation, approx. function='sin'

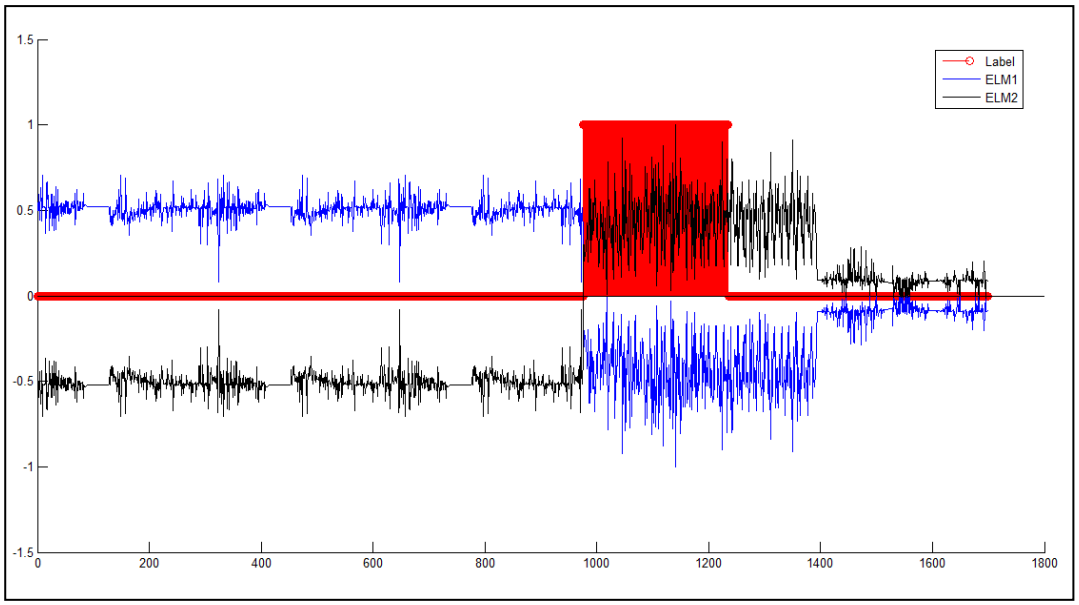


Figure 85. Tenth situation, approx. function='tribas'

4.2.4.11 Front-side impact in an angle of 45°

In this scenario, a motorcycle suffers a front-side impact in an angle of 45° with the other vehicle while crossing an intersection, within that is considered a crash situation. More details in section *3.2.11.Front-side impact in an angle of 45°*.Following figures show the results of ELM algorithm with different approximation functions.

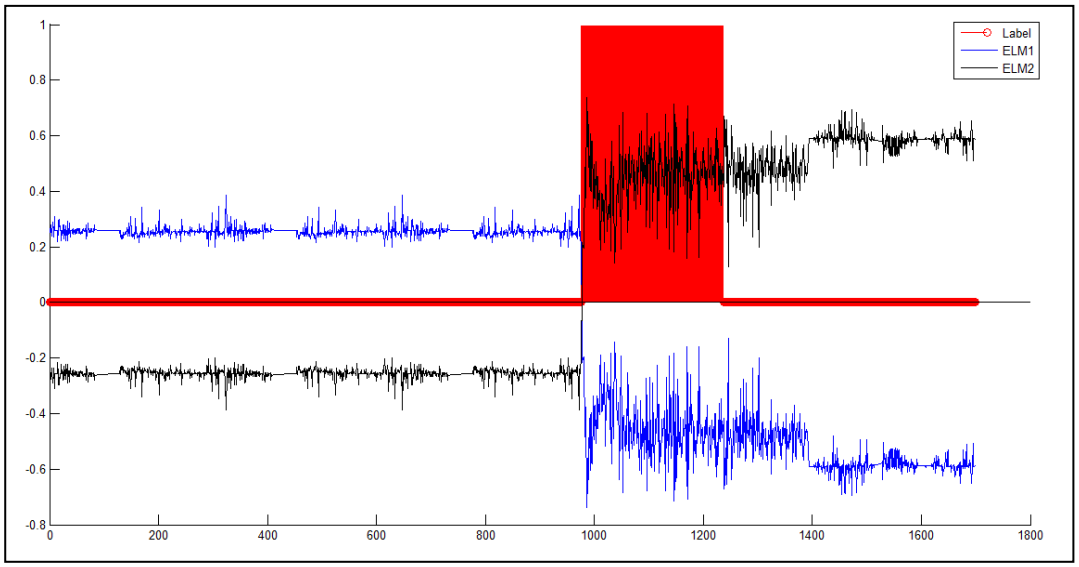


Figure 86. Eleventh situation, approx. function='sig'

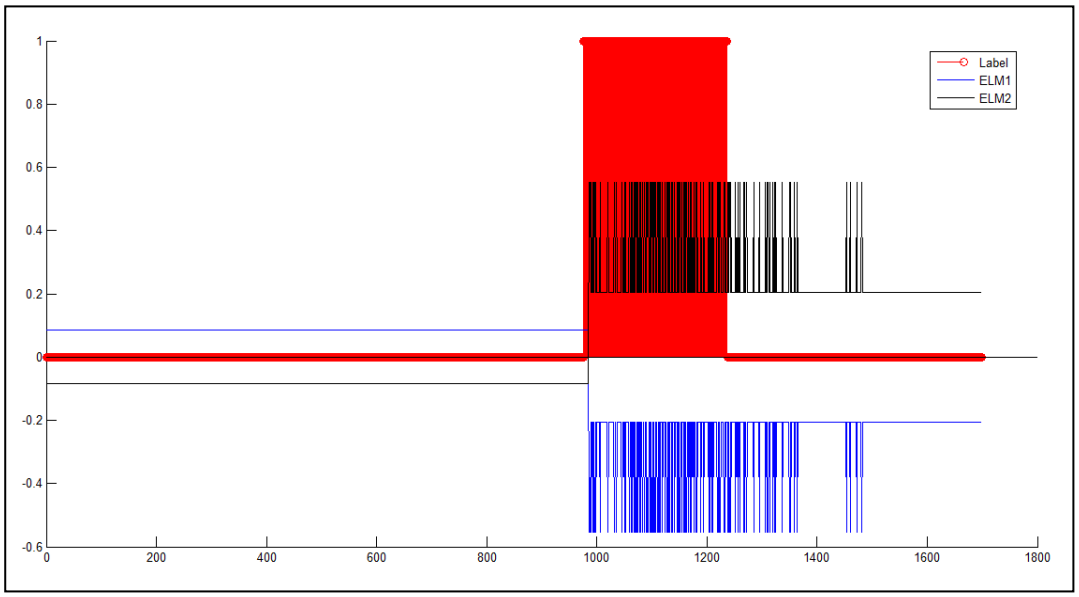


Figure 87. Eleventh situation, approx. function='hardlim'

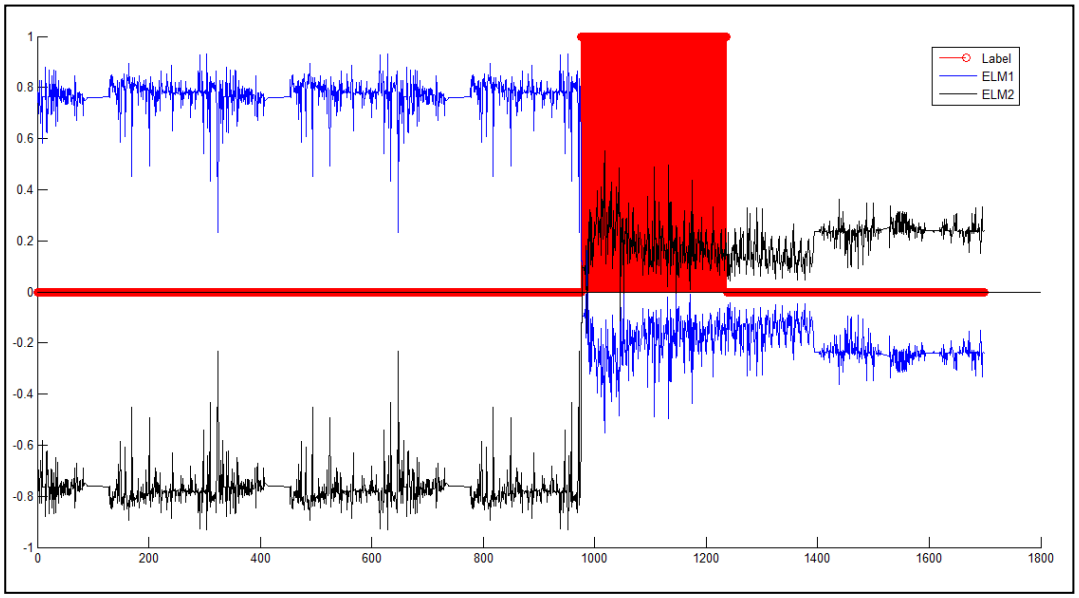


Figure 88. Eleventh situation, approx. function='sin'

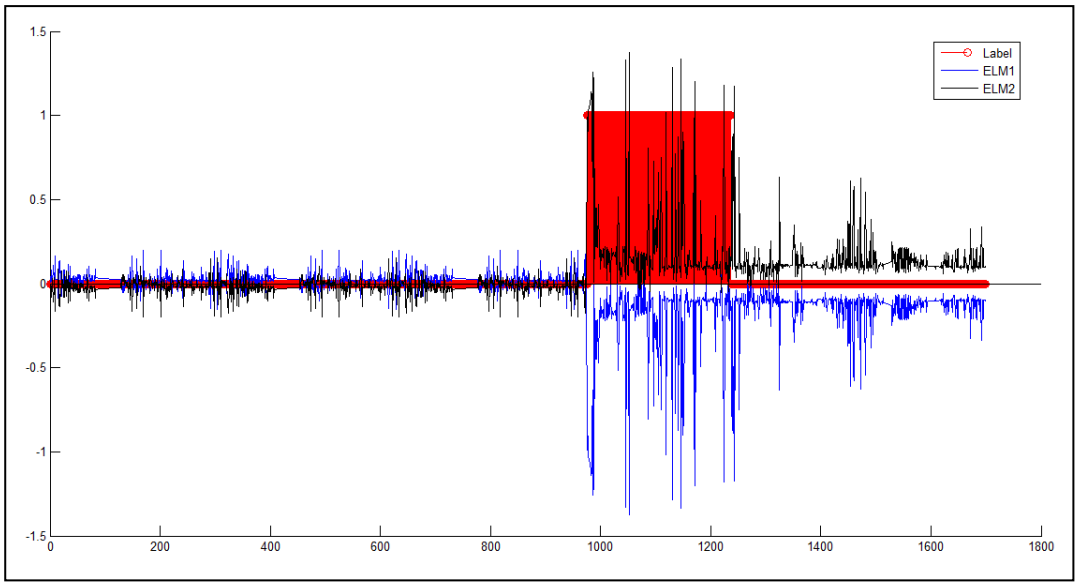


Figure 89. Eleventh situation, approx. function='tribas'

4.2.4.12 Front-side impact in an angle of 70°

Here, a motorcycle suffers a front-side impact in an angle of 70° with other vehicle while crossing an intersection, within that is considered a crash situation. For further information, please refer to section *3.2.12.Front-side impact in an angle of 70°*. Figures below show the results of ELM algorithm with different approximation functions.

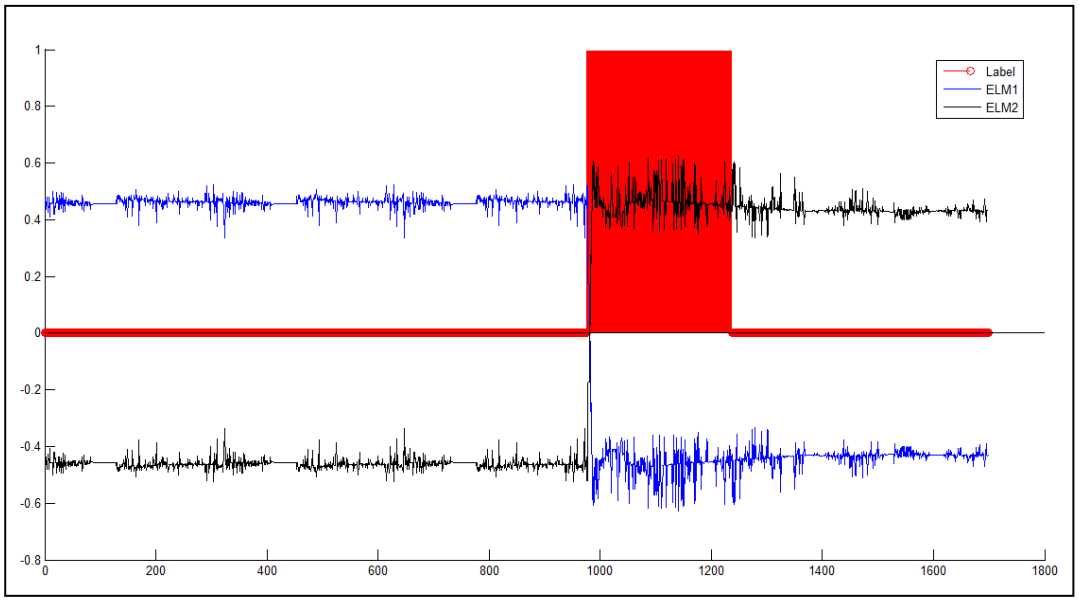


Figure 90. Twelfth situation, approx. function='sig'

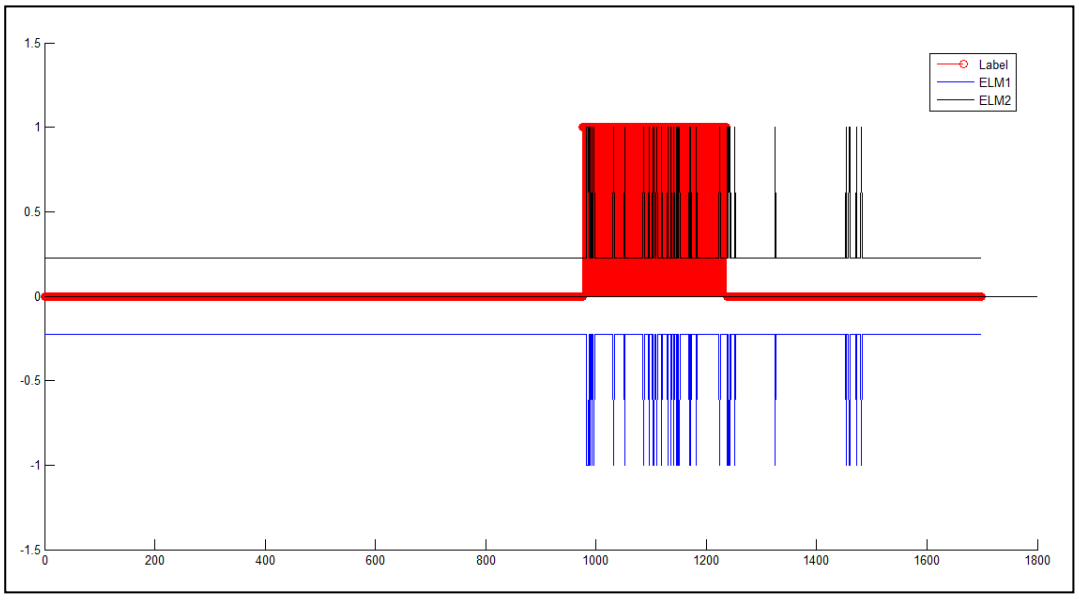


Figure 91. Twelfth situation, approx. function='hardlim'

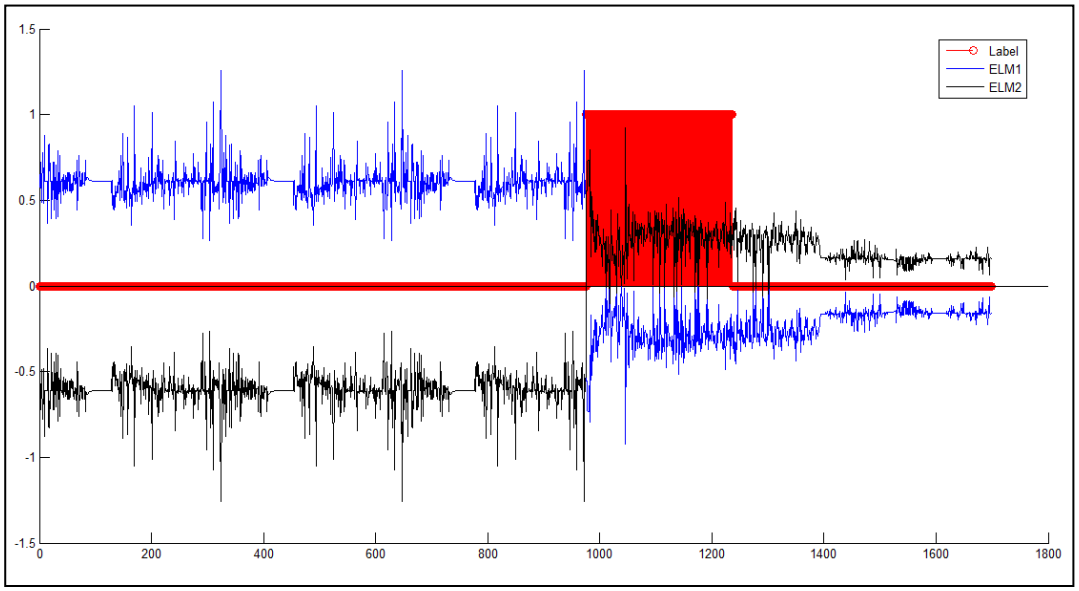


Figure 92. Twelfth situation, approx. function='sin'

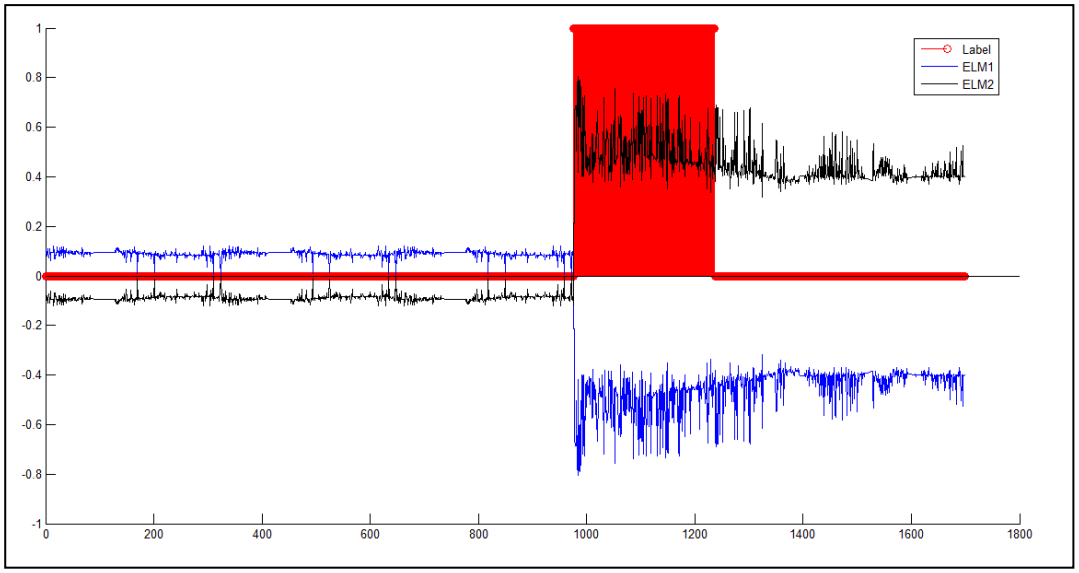


Figure 93. Twelfth situation, approx. function='tribas'

4.2.4.13 Slide side crash

In this situation, a motorcycle skids and slides in a curve, within that is considered a crash situation. More information in section *3.2.13.Slide side crash*. Below, following figures show the results of ELM algorithm with different and approximation functions.

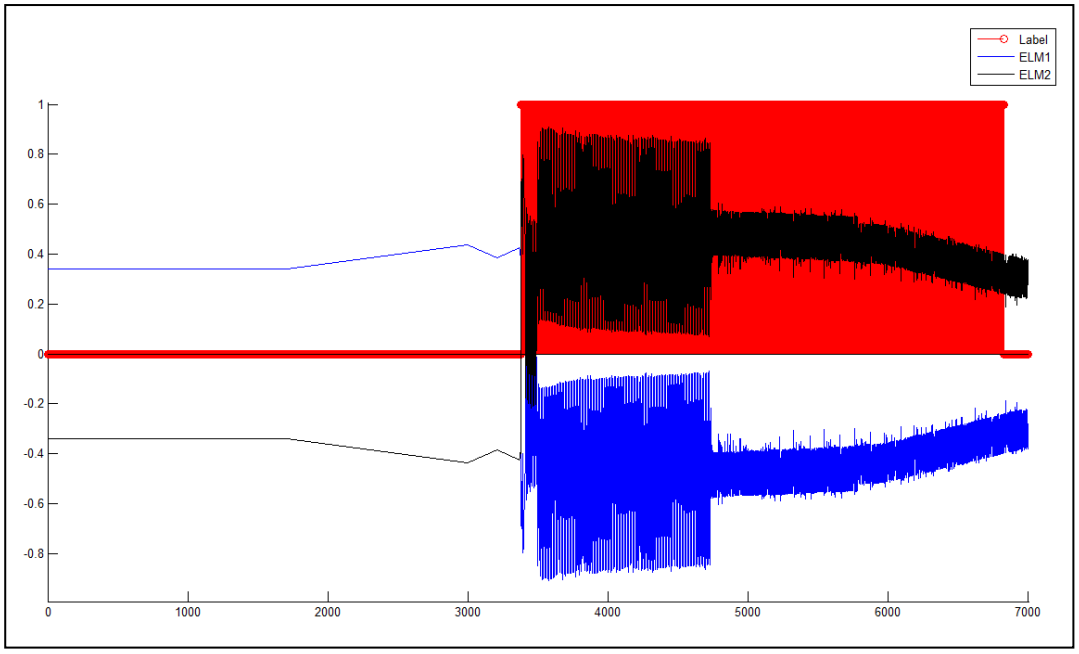


Figure 94. Thirteenth situation, approx. function='sig'

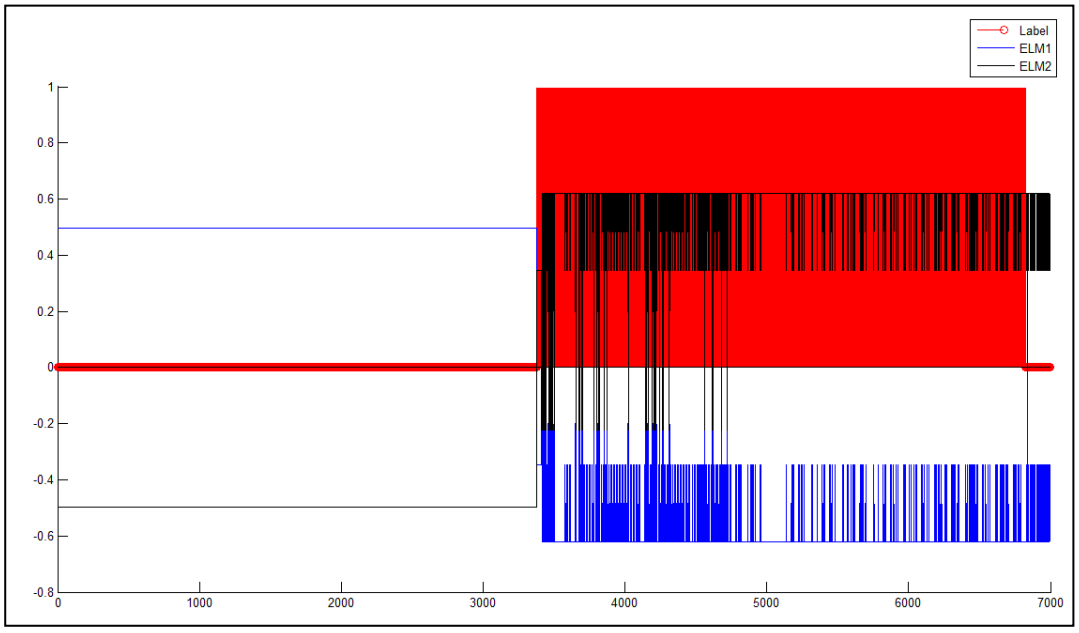


Figure 95. Thirteenth situation, approx. function='hardlim'

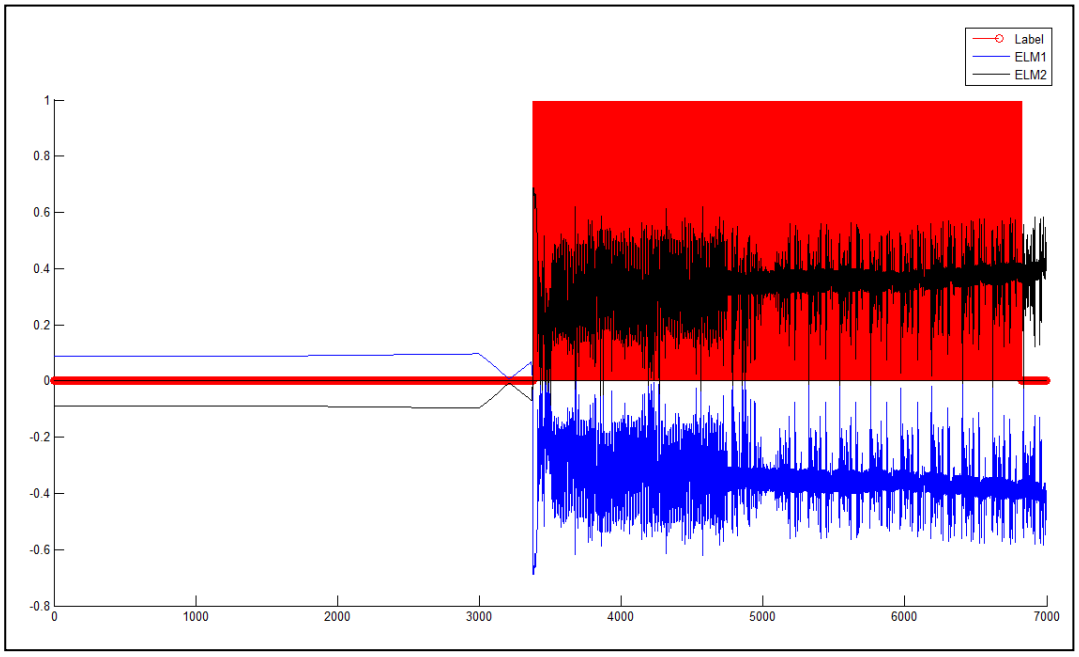


Figure 96. Thirteenth situation, approx. function='sin'

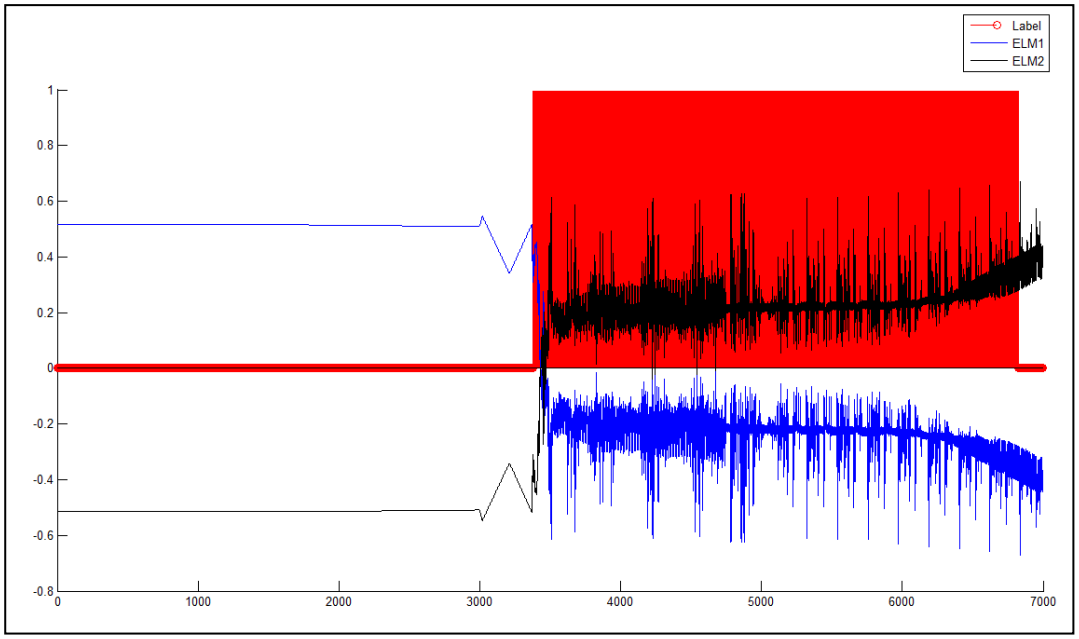


Figure 97. Thirteenth situation, approx. function='tribas'

4.2.4.14 Road exit

Here, a motorcycle looses control and there is a road exit, within that is considered a crash situation. For further information, please refer to section *3.2.14.Road exit*. Below, next figures show the results of ELM algorithm with different approximation functions.

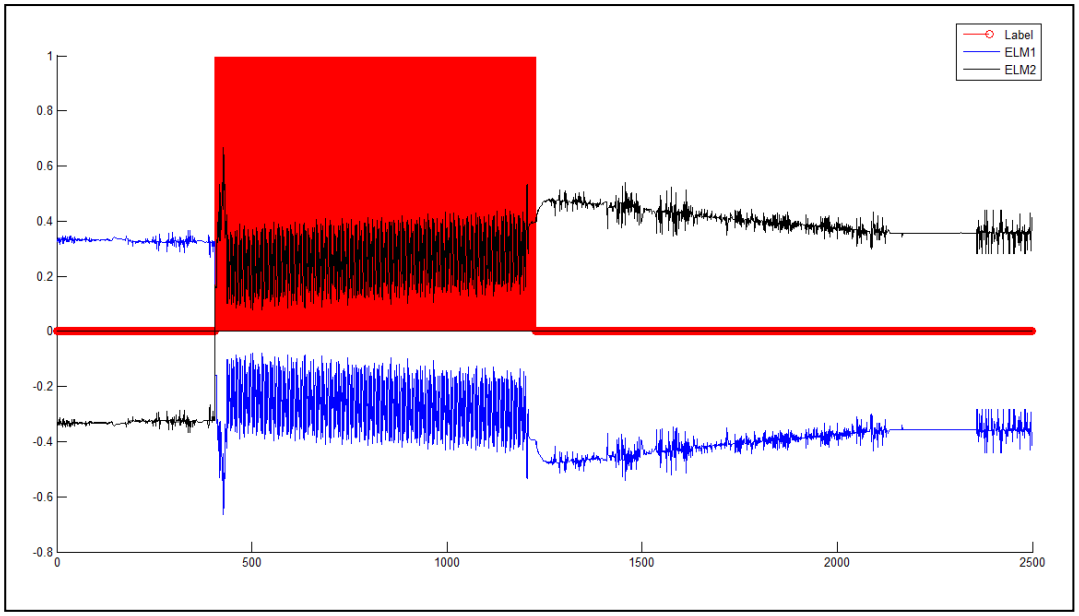


Figure 98. Fourteenth situation, approx. function='sig'

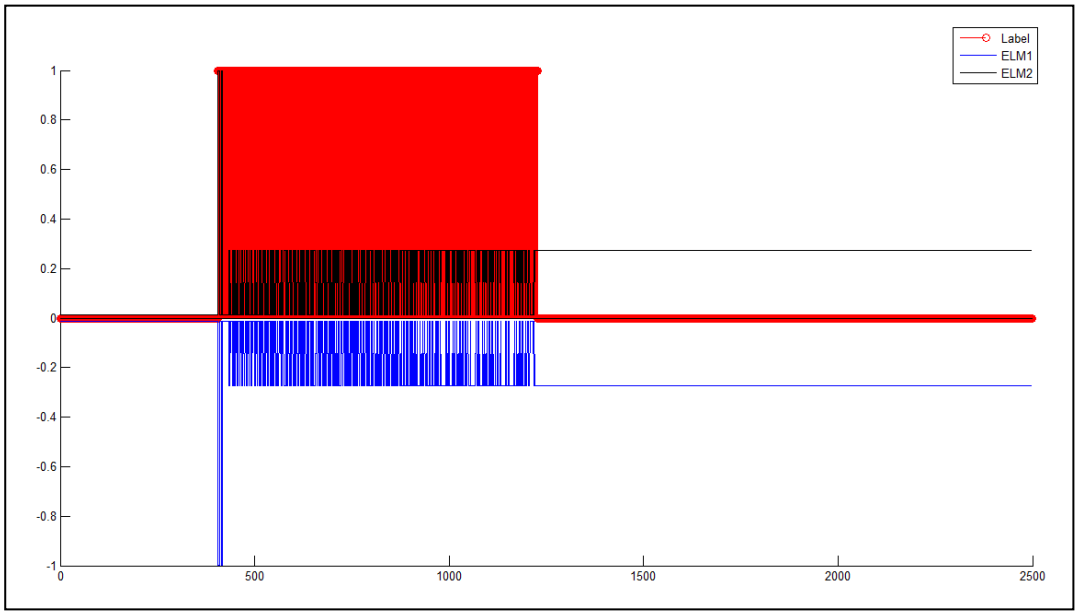


Figure 99. Fourteenth situation, approx. function='hardlim'

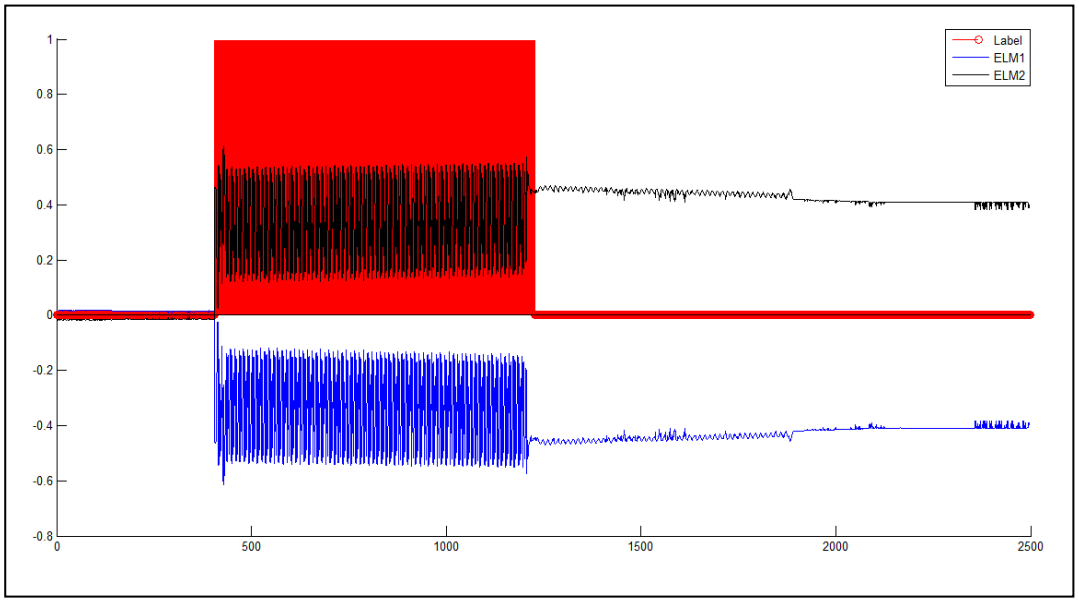


Figure 100. Fourteenth situation, approx. function='sin'

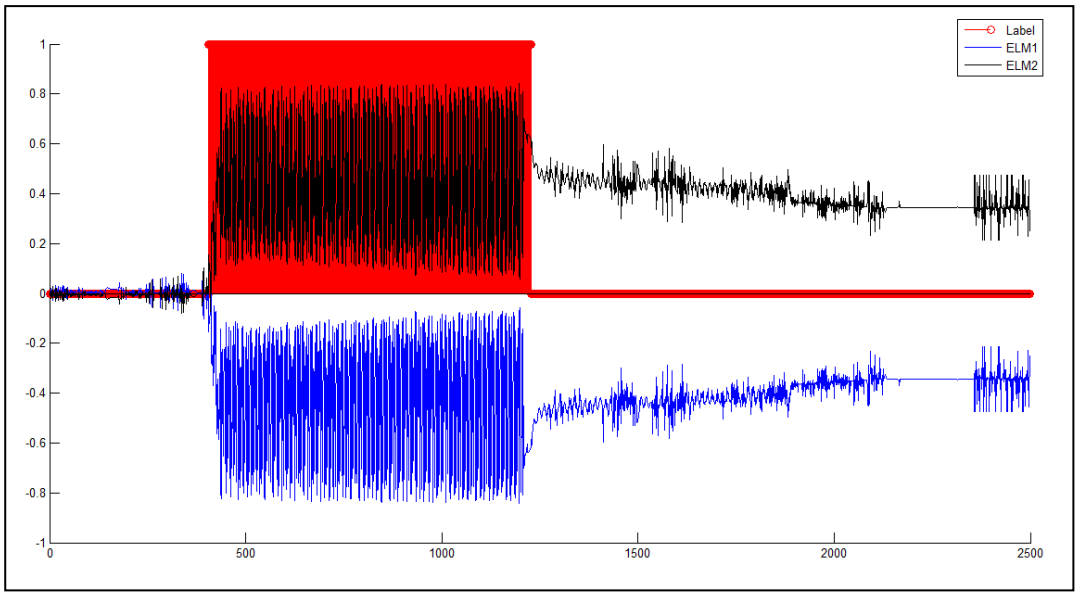


Figure 101. Fourteenth situation, approx. function='tribas'

4.2.5 Discussion of tests results

After displaying the fourteen situations, the following table summarizes the results. It shows in six columns the approximation functions used (sigmoid, hardlim, sine and triangular basis), time spent (in seconds) by "elm.m" on training and testing all data, total time spent (in seconds) by "test_elm.m".

Regarding the approximation functions, it seems that the four are able to react when a crash situation is running. In terms of time spent in training and predicting, the four employ very little time. Nevertheless, hardlim function shows a worse performance than the rest, while the others react better to critical situations.

The time which the program "elm.m" spent on training and predicting all data is almost null. These results confirm that the learning speed of ELM is extremely fast. Therefore, ELM learning algorithm makes an improvement in testing times by reducing them noticeably.

Looking all previous figures in section *4.2.4.Testing the algorithm*, it can be seen that the output of the algorithm suits with the expected results. There is an almost immediately level change when an accident occurs, which means that ANN responds quickly when a crash happens and it has an optimal performance. This means that ELM works efficiently and shows improvements in the accuracy of the crash detection system.

Situation	Approx function	Training time (secs)	Testing time (secs)	Total time (secs)
1st	sig	0	0.02	0.94
	hardlim	0	0.02	0.98
	sin	0	0	0.92
	tribas	0	0.02	0.92
2nd	sig	0	0	0.92
	hardlim	0	0	0.86
	sin	0	0	0.94
	tribas	0	0	0.87
3th	sig	0	0	1.20
	hardlim	0	0	1.31
	sin	0	0	1.18
	tribas	0	0	1.20
4th	sig	0.02	0	0.78
	hardlim	0	0	0.80
	sin	0	0	0.81
	tribas	0	0	0.81
5th	sig	0	0	0.90
	hardlim	0	0	0.87
	sin	0	0	0.89
	tribas	0	0	0.87

Situation	Approx function	Training time (secs)	Testing time (secs)	Total time (secs)
6th	sig	0	0.02	0.61
	hardlim	0	0	0.64
	sin	0	0	0.61
	tribas	0	0	0.61
7th	sig	0.02	0	0.73
	hardlim	0	0	0.80
	sin	0	0	0.81
	tribas	0	0	0.76
8th	sig	0	0	0.59
	hardlim	0	0	0.62
	sin	0	0	0.67
	tribas	0	0	0.56
9th	sig	0	0	0.72
	hardlim	0	0	0.75
	sin	0	0	0.64
	tribas	0	0.02	0.67
10th	sig	0	0.02	0.70
	hardlim	0	0	0.80
	sin	0	0	0.73
	tribas	0	0	0.75
11th	sig	0	0	0.80
	hardlim	0	0	0.72
	sin	0	0	0.70
	tribas	0	0	0.76
12th	sig	0	0	0.73
	hardlim	0	0.02	0.78
	sin	0	0	0.72
	tribas	0	0	0.72
13th	sig	0	0.02	1.25
	hardlim	0	0	1.23
	sin	0	0	1.28
	tribas	0	0.02	1.26
14th	sig	0	0.02	0.87
	hardlim	0	0	0.80
	sin	0	0	0.83
	tribas	0	0	0.95

Table 12. Extreme Learning Machine test results

5. CONCLUSIONS

The main objective of this project was to optimize the crash detection mechanism of the system called "Moto eCall". For this purpose, it was intended to develop an Artificial Neural Network (ANN) capable of detecting motorcycle accidents. Below, main findings and conclusions are presented.

An intensive analysis of the current crash detection system has been developed in order to find errors and performance issues. To this end, a new Matlab code has been created in order to compare the output of Moto eCall system with the expected values. It can be appreciated that the algorithm works efficiently and is capable to detect the multiple sequences when an accident occurs. However, the system can be improved by reducing testing times and also improve accuracy by reducing reaction times when an accident happens.

After a deep analysis of methods to improve Moto eCall performance, artificial neural networks (ANNs) are the best option to do this. ANNs have a number of properties that make them an attractive alternative to traditional problem-solving techniques. Among others, they present numerous advantages such as adaptive learning, self-organization, failover, real-time operation and easy to integrate into existing technology. The type of neural network that better suits with Moto eCall system is a single-layer feed-forward network, with auto-associative, adaptive and unsupervised learning process.

Thanks to the aforementioned characteristics of neural network, a new learning technique referred to as ELM (Extreme Learning Machine) has been used. The ELM modelling scheme is a new framework unlike the standard neural network, it is a Single-hidden Layer Feedforward neural Networks (SLFNs) which randomly chooses the input weights and analytically determines the output weights of SLFNs. This algorithm tends to provide the best generalization performance at extremely fast learning speed, besides its properties satisfy all Moto eCall requirements.

The new neural network and ELM algorithm have been examined to check if not only the characteristics but the performance perfectly suits with Moto eCall system. For this purpose, a complete database has been created (with the data taken from the simulation program called "Working Model") to firstly train the ANN and secondly test ELM. The outcomes show an optimal performance, ANN responds almost immediately when a crash happens. Besides, training and testing times are shorter than currently algorithm. Therefore, these results confirms that neural networks and ELM algorithm satisfy Moto eCall requirements and improve the current system.

In general lines, it can be considered that the main initial goals of the project have been fulfilled and the new technology of neural networks give the possibility to improve crash detection system for motorcycles. However, it would be necessary to carry out real crash test in order to completely understand the real behaviour of a motorcycle. Thereby, all suppositions and simulations made during the project could be verified.

6. BUDGET

This section presents the total budget of the project, where tools utilized (both, software and hardware) and labor cost are included. Below, the aforementioned costs can be found divided in material, development and total costs.

6.1 Material cost

Material	Price (€)	Amortization (years)	Usage (months)	Total (€)
Laptop	600	5	12	120
Windows 7 OS	150	5	12	30
Matlab R2009b	2420	5	12	484
Working Model 2D Student License (1 year license)	49.95	1	12	49.95
Microsoft Excel 2007	229	5	12	45.8
Microsoft Office 2007	289	5	12	57.8
Total				**787.55**

Table 13. Material cost

6.2 Development cost

Function	Price (€/h)	Hours	Total (€)
Engineering	52	250	13000
Document drafting	25	100	2500
Printing expenses	-	-	130
Electricity expenses	0.017	1500	25.5
Total			**15655.5**

Table 14. Development cost

6.3 Total cost

Cost	Total (€)
Material cost	787.55
Development cost	15655.5
VAT (21%)	3453.04
Total	**19896.09**

Table 15. Total cost

7. REFERENCES

[1] Samuel Asensio, César Gay Nieto; *Motorcycle Emergency Call (Moto eCall);* Engineering College of Aarhus (Aarhus, Denmark, June 2011)

[2] Juhani Jääskelaäinen; *eCall: The Pan-European invehicle emergency call*; The 15th ITS World Congress (New York, New York, USA November 20, 2008)
http://www.ecall.fi/SS_on_ACN_-_eCall_20Nov08_Final.pdf

[3] iMobility Support; *eCall Toolbox*
http://www.esafetysupport.org/en/ecall_toolbox/index.html
hhttp://www.esafetysupport.org/download/ecall_toolbox/049-eCall.pdf

[4] Vittore Cossalter; *Motorcycle Dynamics* (Second Edition).

[5] Ivo Boniolo, Mara Tanelli, Sergio M. Savaresi; *Roll Angle Estimation in Two-Wheeled Vehicles*; 17th IEEE International Conference on Control Applications, part of 2008 IEEE Multi-conference on Systems and Control (San Antonio, Texas, USA, September 3-5, 2008).
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4629599
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4745897

[6] Mara Tanelli, Francesco Schiavo, Sergio M. Savaresi And Gianni Ferretti; *Object–Oriented Multibody Motorcycle Modelling for Control Systems Prototyping*; Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design (Munich, Germany, October 4-6, 2006).
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4777065

[7] MOTOSIM (a motorcycle simulator design proyect); *Forces acting on motorcycle*.
http://motosim.blogspot.com/2008/07/report-1.html

[8] INTRAS, Winterthur Foundation; *Análisis de la Accidentalidad de Motocicletas en España* (February 2005)
http://gembmw.files.wordpress.com/2010/12/informeintras.pdf

[9] Carlos Nicolás Fraile; *Motos: menos siniestralidad*; DGT (April 2005)
http://www.dgt.es/revista/archivo/pdf/num171-2005-Motos.pdf

[10] Joanne Hill, Caroline Starrs; *Saving Lives, Saving Money: The costs and benefits of achieving safe roads*; Road Safety Foundation/ RAC Foundation (April 2011)
http://www.roadsafetyfoundation.org/media/11070/saving%20lives_saving%20money.pdf

[11] James Smith, Todd Frank, Graeme Fowler, Stephen M.Werner; *Motorcycle Crash Testing and Reconstruction: Selected Exponent Publications*; Exponent

http://www.exponent.com/files/Uploads/Documents/practices/vehicles/motorcycle_publication_handout.pdf

[12] Guang-Bin Huang, Qin-Yu Zhu, Chee-Kheong Siew; *Extreme learning machine: Theory and applications*; ScienceDirect (16 May 2006)
http://www.ntu.edu.sg/home/egbhuang/

[13] Emad A. El-Sebakhy; Extreme Learning Machine as a New Framework in Predicting Material Properties: Methodology and Comparison; The 12th International Conference of International Association for Computer Methods and Advances in Geomechanics (IACMAG) (Goa, India, 1-6 October, 2008)
http://www.civil.iitb.ac.in/~dns/IACMAG08/pdfs/F12.pdf

[14] Christos Stergiou, Dimitrios Siganos; *Introduction to neural networks*
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Introduction to neural networks

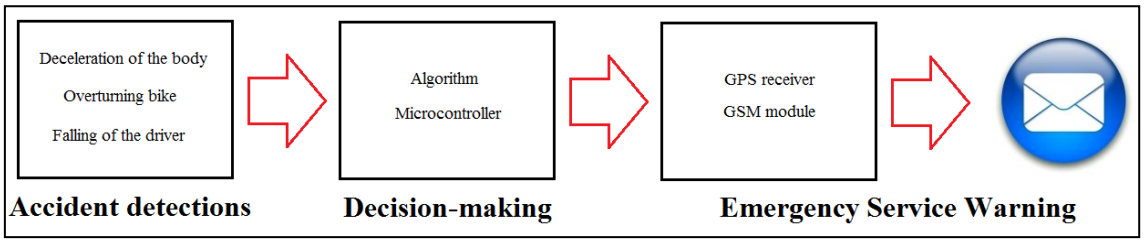
[15] ESCOM; *Redes Neuronales: Algoritmos de aprendizaje* (August, 2002)
http://www.slideshare.net/mentelibre/redes-neuronales-algoritmos-de-aprendizaje

[16] Roman V Belavkin; *Feed-Forward Neural Networks* (BIS3226)
http://www.eis.mdx.ac.uk/staffpages/rvb/teaching/BIS3226/hand11.pdf

APPENDIX A. MOTO eCALL SYSTEM

INTRODUCTION

Motorcycle Emergency Call (Moto eCall) is a system divided into three distinct blocks: accident detections, decision-making and emergency services warning. Following figure shows the system block diagram which helps to understand how it works.



Block diagram of current system

Firstly, for accident detections it has been required a thorough study of the sequence of events. Although there are no definite sequences, because each accident is unique, there are at least three common factors that happen in most accidents. These factors are a strong deceleration of the body, overturning of bike and the falling of the driver, which can be determined by an accelerometer, a gyroscope and detection sensors, respectively.

Secondly, for decision-making it is necessary a microcontroller. This device provides all the peripherals to manage the different components in the system, two USART (universal synchronous/asynchronous receiver/transmitter) to connect with the GPS receiver and GSM module, ADC (Analogue to Digital Converter) for the accelerometer and gyroscope, and digital ports with change level interruptions for the digital sensors and user interface.

Once an accident is detected the microcontroller warns to emergencies. Firstly, it acquires the location through the GPS receiver. Secondly, the emergency message is generated with this data, besides to those obtained from the sensors. Lastly, it sends the SMS through a GSM module.

Below can be found information related to hardware devices, as well as software architecture of Motorcycle Emergency Call project.

HARDWARE

- ## Accident detections

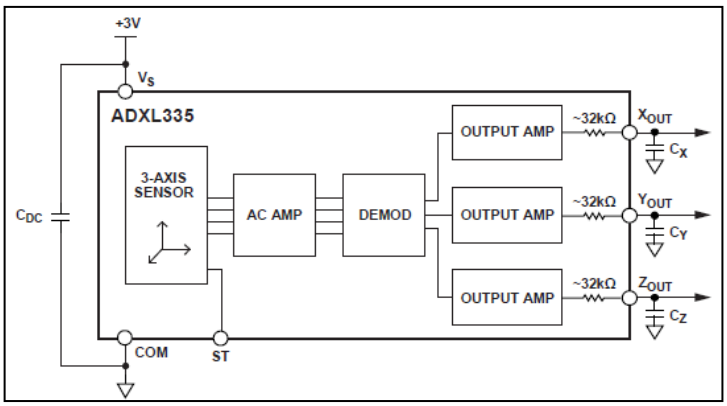
ADXL335 accelerometer

The ADXL335 is 3-axis accelerometer with signal conditioned voltage outputs. This output signals are analogue voltages that are proportional to acceleration suffered. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration, with a range of ±3 g (for further information, the datasheet is included in the CD ROM).

The accelerometer has the following characteristics:

- Three-axis sensing
- Small, low profile package (4 mm x 4 mm x 1.45 mm)
- Low power: 350 µA
- Single-supply operation: 1.8 V to 3.6 V
- 10,000 G shock survival
- Excellent temperature stability
- Bandwidth adjustment with a single capacitor per axis

Regarding the last feature, bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis. Next figure shows the functional block diagram.



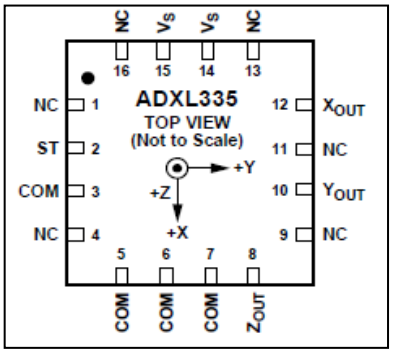
ADXL335 functional block diagram

Following table explains device specifications which are very important to understand how it works. Here it can be seen that the measuring range is ± 3 G, the typical sensitivity is 300 mV/G, the power operation range is between 1.8 and 3.6 V and the zero G bias level is 1.5 V (voltage level output when there is 0 G).

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity			±1		%
SENSITIVITY (RATIOMETRIC)	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		mg/°C
FREQUENCY RESPONSE					
Bandwidth X_{OUT}, Y_{OUT}	No external filter		1600		Hz
Bandwidth Z_{OUT}	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time	No external filter		1		ms

ADXL335 specifications

Next figure illustrates the top view accelerometer and its pin configuration.



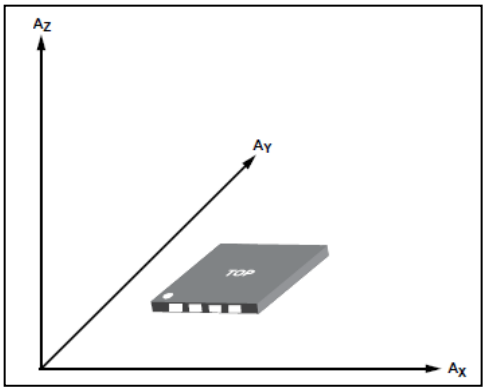
ADXL335 pin configuration

where

- **NC:** No connect (NC pins are not internally connected and can be tied to COM pins, unless otherwise noted)
- **ST:** Self-test
- **COM:** Common
- **Zout:** Z channel output
- **Yout:** Y channel output
- **Xout:** X channel output
- **Vs:** Supply voltage (1.8 V to 3.6 V)
- **EP:** Exposed Pad (Not internally connected, solder for mechanical integrity)

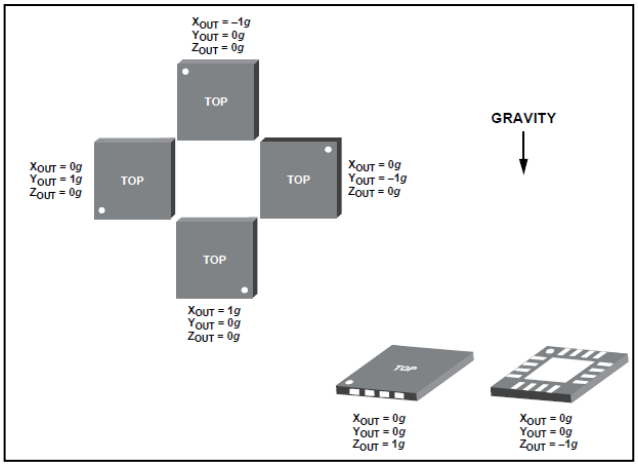
Regarding the way of operation, the acceleration deflects the moving mass and unbalances the differential capacitor resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to determine the magnitude and direction of the acceleration.

Besides, the ADXL335 uses a single structure for sensing the X, Y, and Z axes. As a result, the three axes' sense directions are highly orthogonal and have little cross-axis sensitivity, as shown in figure below, where the corresponding Output Voltage increases when accelerated along the sensitive axis.



Axes of acceleration sensitivity

Finally, next figure shows the relation between the output response and the orientation gravity.



Output response vs. Orientation gravity

IDG500 Gyroscope

Since it was an unknown device, it has been made a thorough study about the output characterization. Therefore, before explaining the characteristics of the gyroscope, the two systems studied will be explained, as well as the results.

Output characterization

The gyroscopes available in the market give information about the angular speed of the device and the measurement unit is called dps, which means degrees per second. This value can be provided in an analogue or digital way. The first one is the most inexpensive and also allows deciding the precision and timing of the samples used. Besides, the most common analogue gyroscopes should be supplied with 3 V, and have a bias which varies between 0.5 and 2.5 V. A voltage of 1.5 V represents 0 dps.

The system being designed should have a very slow angular rate in regular situations. Thereby, if the gyroscope output is modelled with a sin waveform, its frequency should not be greater than 1 Hz as can be seen below.

If gyroscope output signal is considered the following:

$$gyro(t) = maxAngle \cdot \sin(2 \cdot \pi \cdot f_c \cdot t)$$

where

$$f_c \approx 1Hz$$

$$maxAngle \approx 45°$$

Then, the maximum dps value is when $\frac{dgyro(t)}{dt}$ has a maximum value.

With these conditions the maximum angular rate is around 300 dps. This value should be tested in a real environment, but in this case is going to be assumed as real and take it into account.

It is necessary to make an integral to calculate the angle of the system. The problem has to possible approaches: an analogue or a digital system. The first case should be the most accurate because in the second approach the system accuracy is limited due to the approximation of an integral by a summation.

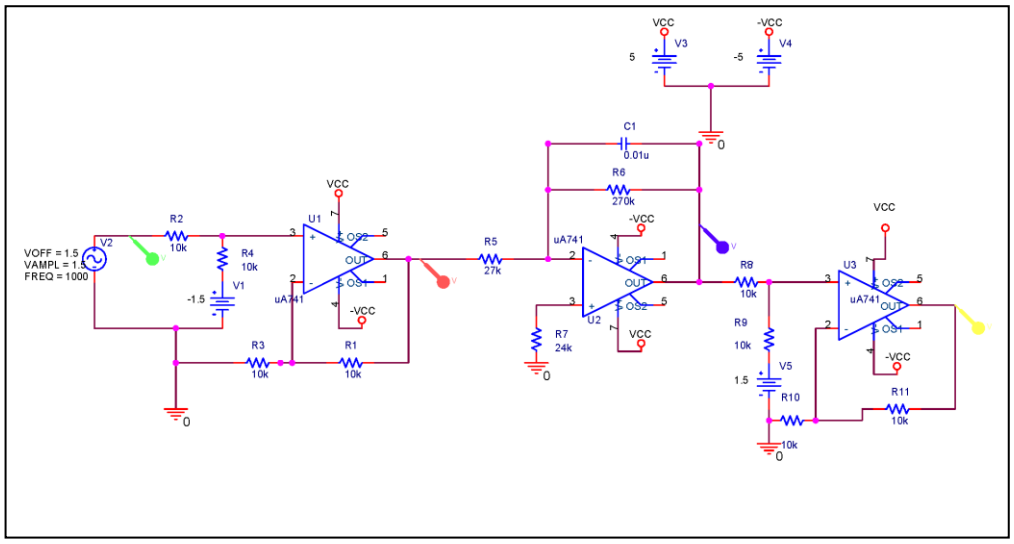
Analogue system

The problem can be solved by using an integration circuit, based in operational amplifiers. Below it is explained the steps taken to design the system.

The first step is to adapt the sensor voltages by subtracting 1.5 V to the output voltage in order to make the integration possible. Thus, there will be positive and negative voltages which are needed to make the integration.

The second stage is solving the integral through the integrator circuit based on operational amplifiers.

The last step is another voltage adaptation to positives voltages in order to make the interpretation possible by a digital circuit. Figure below shows an example of analogue circuit.



Analogue system

In this system two problems have been found, the first one consists of the necessity of obtaining a symmetric power supply from a single one (it would increase the system complexity and size) and the second one is that the lowest operation frequency is when $X_{Ci} = {}^{R_6}/_{10}$. From all this is possible to know that

$$f_{min} = \frac{1}{2 \cdot \pi \cdot C_1 \cdot R_6/10}$$

With the values showed in the circuit, the minimum frequency is 590 Hz. Then, to reduce this frequency and make the system able to integrate the low frequency change of the system output, the capacitor should be enormous. It is not possible to increase R6 value because the gain value is fixed with R5. This means that if R6 increases, R5 will also increase and current system input will decrease.

Digital system

The digital system is based on an analogue to digital converter (ADC) where the output signal is sampled and then the integral is approximated by a summation, as can be seen below:

$$\int \mathrm{dps} \cdot \mathrm{dt} \approx \sum \mathrm{dps}_{\mathrm{sampled}} \cdot \mathrm{T}_{\mathrm{sample}}$$

Where "Tsample" is the time between two samples. This value depends on the ADC and it should be short enough to follow the signal with the precision required.

As said before, the signal generated by the gyroscope in this system is a very low frequency signal, around 1 Hz or less. To ensure its correct interpretation, the sample frequency should be doubled, as Nyquist theory says, but to ensure its correct reconstruction, is necessary to sample with a higher frequency (5 times higher should by enough).

This digital system has several problems with the integration at low frequencies. According to the work "Roll Angle Estimation in Two-Wheeled Vehicles" (attached on the CD), the angle integration has deviations when the integration time is high. Therefore, a possible solution can be to reset the angle in a known condition, i.e. when the motorbike is perpendicular to the ground.

After many attempts to perform this solution and the failure of these, it has not been able to carry out the estimation of the angle due to lack of time.

IDG500 gyroscope

The IDG-500 gyroscope is an integrated dual-axis angular rate sensor which uses two sensor elements that sense the rate of rotation about the X- and Y-axis (in-plane sensing). Besides, it incorporates X- and Y-axis low-pass filters and an EEPROM for on-chip factory calibration of the sensor (for further information, the datasheet is included in the CD ROM).

This device has the following characteristics:

- Integrated X- and Y-axis gyros on a single chip
- Smallest dual axis gyro package (4 x 5 x 1.2mm)
- Two separate outputs per axis for standard and high sensitivity:
    - X-/Y-Out Pins: 500°/s full scale range, 2.0mV/°/s sensitivity
    - X/Y4.5Out Pins: 110°/s full scale range, 9.1mV/°/s sensitivity
- Low power: 350 μA
- 3V single-supply operation
- 10,000 G shock tolerant

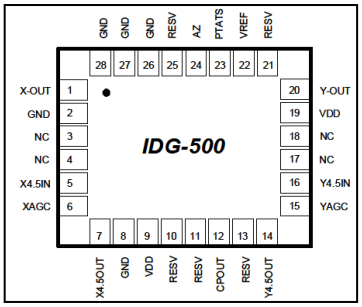
- Integrated amplifiers and low-pass filters
- Auto-Zero function
- On-chip temperature sensor
- High vibration rejection over a wide frequency range

Following table explains device specifications which are very important to understand how it works. Here it can be seen that there are two kinds of output, the first has a measuring range of ± 500 dps and the typical sensitivity is 2 mV/dps, while the second has ± 110 dps and 9.1 mV/dps. Besides, the power operating voltage range is between 2.7 V and 3.3 V.

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
SENSITIVITY						
Full-Scale Range	At X-OUT and Y-OUT			±500		°/s
	At X4.5Out and Y4.5Out			±110		°/s
Sensitivity	At X-OUT and Y-OUT			2.0		mV/°/s
	At X4.5Out and Y4.5Out			9.1		mV/°/s
Initial Calibration Tolerance	At X-OUT and Y-OUT			±6		%
Over Specified Temperature	At X-OUT and Y-OUT			±10		%
Nonlinearity	At X-OUT and Y-OUT, Best Fit Straight Line			<1		% of FS
Cross-axis Sensitivity				±1		%
REFERENCE						
Voltage (VREF)				1.35		V
Tolerance				±50		mV
Load Drive				100		µA
Capacitive Load Drive	Load directly connected to VREF			100		pF
Power Supply Rejection	VDD= 2.7V to 3.3V			1		mV/V
Over Specified Temperature				±5		mV
ZERO-RATE OUTPUT						
Static Output (Bias)	Factory Set			1.35		V
Initial Calibration Tolerance	Relative to VREF	With Auto Zero		±20		mV
		Without Auto Zero		±250		
Over Specified Temperature	Relative to VREF	Without Auto Zero		±50		mV
Power Supply Sensitivity	@ 50 Hz			10		°/sec/V
FREQUENCY RESPONSE						
High Frequency Cutoff	Internal LPF -90°			140		Hz
LPF Phase Delay	10Hz			-4.5		°
OUTPUT DRIVE CAPABILITY						
Output Voltage Swing	Load = 100kΩ to $V_{dd}/2$		0.05		Vdd-0.05	V
Capacitive Load Drive				100		pF
Output Impedance				100		Ω
POWER SUPPLY (VDD)						
Operating Voltage Range			2.7	3.0	3.3	V
Quiescent Supply Current				7		mA
Over Specified Temperature				±2		mA

IDG500 specifications

Next figure illustrates the top view gyroscope and its pin configuration.

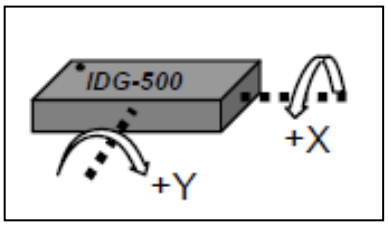


IDG500 pin configuration

where

- **GND:** Ground
- **VDD:** Positive supply voltage
- **X-OUT:** Rate output for rotation about the X-axis
- **X4.5IN:** X-axis input to the 4.5X amplifier
- **XAGC:** Amplitude control capacitor connection
- **X4.5OUT:** X-axis output to the 4.5X amplifier
- **CPOUT:** Charge pump capacitor connection
- **Y4.5OUT:** Y-axis output to the 4.5X amplifier
- **YAGC:** Amplitude control capacitor connection
- **Y4.5IN:** Y-axis input to the 4.5X amplifier
- **Y-OUT:** Rate output for rotation about the Y-axis
- **VREF:** Precision reference output
- **PTATS:** Temperature sensor output
- **AZ:** X & Y auto zero control pin
- **RESV:** Reserved. Do not connect
- **NC:** Not internally connected. May be used for PCB trace routing

Regarding the way of operation, this is a dual-axis rate sensing device which produces a positive output voltage for rotation about the X- or Y-axis, as shown in figure below.



Rate sensitive axis

Rider and passenger sensors

The devices acting as rider and passenger sensors are called “SPST DIP switches”. DIP switches are manual electric switches packaged in a group in a standard dual in-line package (DIP). These kinds of devices are an alternative to jumper blocks which their main advantages are that they are quicker to change and there are no parts to lose. Following figure shows DIP switches used in the prototype laboratory.

SPST DIP switches

DIP switches contain the following specifications:

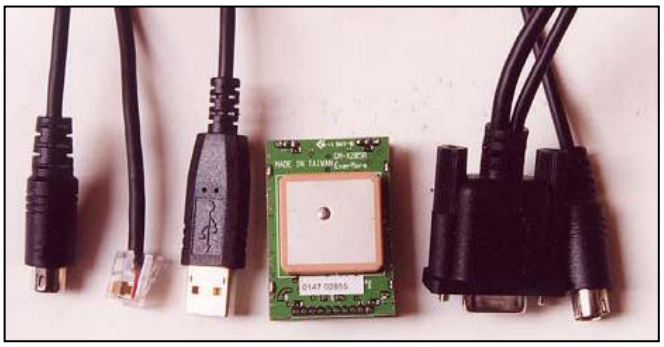
- Contact rating (Switching): 25mA 24V DC
- Contact rating (Non-Switching): 100Ma
- Contact resistance (initial): 50mO Max
- Contact resistance (after life test): 100mO Max
- Insulation resistance: 1000MO Min. at 500V DC
- Dielectric strength: 500V DC Min. for 60 seconds. 1mA
- Capacitance between adjacent switches 5PF Max
- Operating Temperature: -25° to 70°
- Storage Temperature: -40° to 85°
- Operation force: 8N Max
- Resistance to soldering heat: 260±5° for 5±1seconds

There are two positions for the switch which corresponds to OFF (open circuit) or ON (closed circuit). It is necessary to configure the device to ground out the bottom of a pull-up resistor when closed. Then, it generates the low or high logic voltage corresponding to "0" and "1" signals, respectively.

- Emergency service warning

GM-305 GPS Receiver

The GM-305 is a GPS receiver designed for rapid start-up time and high performance in foliage and urban environments. It provides enhanced position and velocity filtering for smooth navigation, an onboard patch antenna and RS-232 driver for simple interfacing (for further information datasheet is included in the CD). Figure below shows the GPS circuit.



GM-305 GPS receiver

This device has the following characteristics:

- General: L1 (1575.42MHz), C/A code, 12-channel, Carrier-Aided with HWTrack
- Dimension: 45mmx30mmx13mm w/o housing
- Weight: 25g w/o housing
- Sensitivity: -165 dBW minimum
- Update Rate: 1Hz
- Accuracy: Position: 25m CEP S/A off
- Velocity: 0.1/sec S/A off
- Time: ± 1μs
- Typical Acquisition Time : Cold start : < 120sec
- Warm start : < 40 sec
- Hot start : < 10 sec
- Reacquisition : < 100 millisecond
- Active Antenna (GM-308): >20 dB (100ft)
  > 10 dB (10 ft)
- Passive Antenna (GM-308): > 2 dBic
- Dynamics : Altitude: -1000m to + 18,000m
  Velocity: 500m/sec
  Acceleration: ±4g
- Primary Power : 3.8V~8V DC, ±10%
- Power Consumption : 125mA @ 3.3V
- 70-90mA with power saving
- Serial Port : Standard RS-232
- IOs : 1 PPS & TX out @TTL level
- Protocols: EverMore binary mode@4800/9600 baud, 8N1
  NMEA-0183 v2.20@4800/9600 baud, 8N1
- NMEA Messages GGA, GSA, GSV, RMC (standard output)
  GLL and VTG (optional)

This subsystem is responsible for all matters related to the location of the vehicle involved in an accident. For reception of satellite signals, the vehicle brings the device 'Evermore GM-305 GPS', which calculates its real coordinates and the current time.

The status of the satellites can be determined in advance, during the first minutes of booting. In this state, the device attempts to locate the minimum number of satellites required to establish the position of the vehicle.

Once the device and the satellites are connected, the GPS receiver starts to get information about the position of the vehicle and the exact time, and sends them to the microcontroller through the serial port.

MC35i GSM Terminal

The MC35i Terminal is a compact GSM modem for the transfer of data, voice, SMS and faxes in GSM networks. Industrial standard interfaces and an integrated SIM card reader allow using MC35i Terminal easily as a dual band GSM terminal. Besides, it offers the full benefits of GPRS technology, such as permanent online connections plus cost-efficient, high-speed data transmission (for further information datasheet is included in the CD). Next figure shows the GSM module.

GSM module

This device has the following features:

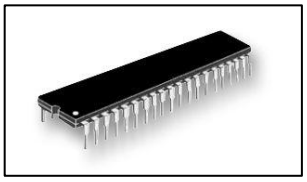
- Dual-Band EGSM900 and GSM1800
- Dimensions: 65 x 74 x 33 mm
- Weight: 130 g
- Output power: – Class 4 (2 W) for EGSM900
  – Class 1 (1 W) for GSM1800
- Control via AT commands
- SIM Application Toolkit
- Multiplex RS232 interface
- Supply voltage range 8...30 V
- Specifications for SMS:
  - Point-to-point MO and MT
  - SMS cell broadcast
  - Text and PDU mode

The microcontroller sends AT commands through a serial port for giving instructions to the modem, and then reads the information that it sends back in order to know if everything works as expected. Besides, the GSM modem contains a SIM card inside, so basically works as a common mobile phone. The communication between the modem and the emergency services is done via GSM technology, sending a SMS containing key information.

- Acquisition and data processing

ATMega 324P microcontroller

ATmega 324P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, this microcontroller achieves throughputs approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed (for further information, the datasheet is included in the CD ROM). Following figure shows an image of this component.

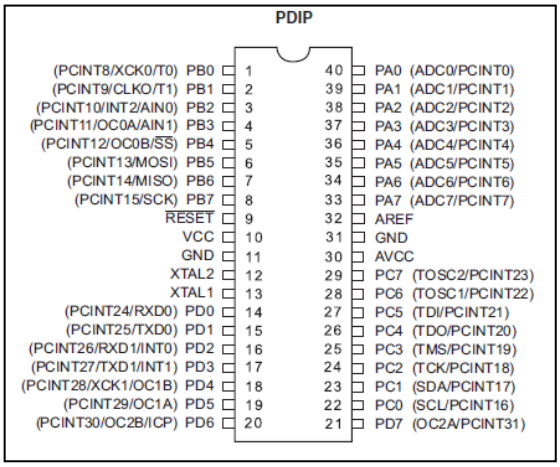


ATMega 324P

The chip has the following features:

- High-performance, Low-power Atmel AVR 8-bit Microcontroller
- Advanced RISC Architecture:
  - 131 Powerful instructions
  - 32 × 8 General purpose working registers
  - On-chip 2-cycle Multiplier
- High Endurance non-volatile memory segments
  - 32 Kbytes of in-system self-programmable flash program memory
  - 1 Kbyte EEPROM
  - 2 Kbytes internal SRAM
- Peripheral Features
  - Two 8-bit Timers with separate prescalers and compare modes
  - 8-channel, 10-bit ADC
- Differential mode with selectable gain at 1×, 10× or 200×
  - Byte-oriented Two-wire Serial Interface
  - Two Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analogue Comparator
- Operating Voltage: 1.8V - 5.5V
- Speed Grades: 0 - 4 MHz @ 1.8V - 5.5V
- Power Consumption at 1 MHz, 1.8V when active: 0.4 mA

Next figure illustrates the microcontroller pin configuration.



ATMega324P pin configuration

where

- **VCC:** Digital supply voltage
- **GND:** Ground
- **Ports (A, B, C, D) inputs and outputs:** Ex:PA7-PA0
- **RESET:** Reset input
- **XTAL1:** Input to the inverting Oscillator amplifier and input to the internal clock operating circuit
- **XTAL2:** Output from the inverting Oscillator amplifier
- **AVCC:** Supply voltage pin for Port A and the ADC (it should be externally connected to VCC, even if the ADC is not used)
- **AREF:** Analogue reference pin for the ADC

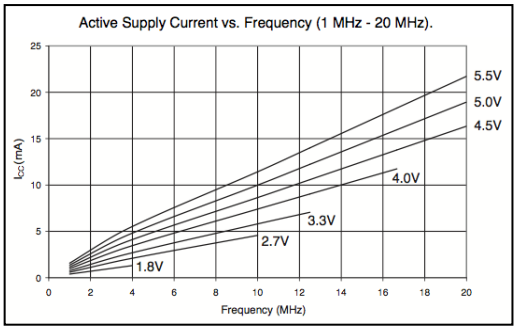
As said before, this microcontroller coordinates the peripherals and it is responsible for making decisions. The following section explains the prototype design.

Board development

Once all hardware has been selected, the following step is to start with the laboratory prototype. This develop has been made to ensure the possibility of interconnecting all the components but it is not final version for use or exhaustive testing.

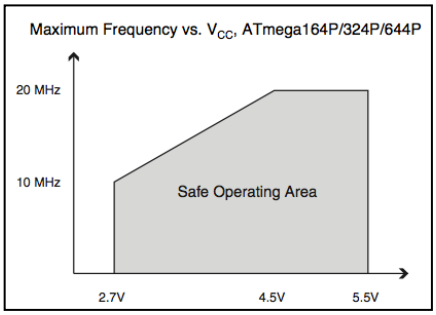
The first decision to make is the main crystal resonator frequency. This value defines the process speed, but also the devices consume and it has some implications in the internal functionality. Therefore, the selected value is 3.6864 MHz which is appropriated for the UART baud rate generation. Since it is not the highest frequency supported by the microcontroller, this means that is possible to upgrade it if is needed to increase the speed.

Figure below shows device consume related with frequency and voltage. This increment in speed can increase consume by almost 4 factor at maximum speed.



Microcontroller consume

The next issue is about the power rails, which voltage to power the system. As said before, the system is designed to be integrated in a motorbike so that the common voltage used is 12 volts DC. There are several elements to be powered, being both gyroscope and accelerometer the most restrictive. These elements need a 3.3 V supply, therefore the microcontroller can be supplied from 1.8V to 5.5V, but this election could restrict the main clock frequency range achievable safe. Below it is showed the maximum frequency related with voltage.

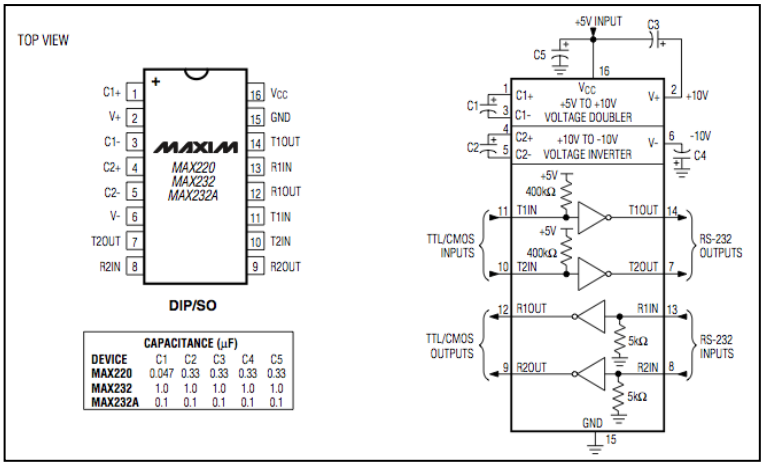


Maximum frequency vs. voltage

The other element to be supplied is the RS-232 driver, which transforms TTL level from microcontroller to serial port (RS-232) standard levels. To do this, the most common device for this task is the MAX232 which needs 5 V of power supply. There are another devices in the market able to do the task with 3V supply but most of them are supplied in surface mount encapsulation (SMD), making more complicated to use in this prototype.

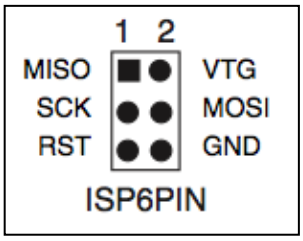
Considering the above, it has been decided the use of two different power rails, 5V and 3.3V. Thus, this configuration allows the most flexible solution during the development. This means that ADC has to be supplied with two different voltage levels: 5 V for AVCC pin because this voltage cannot be different from the main voltage and 3.3V for AREF pin (reference voltage for the conversion) in order to get the maximum number of quantification. To do this, it has been used two voltage regulators: LM1117T3.3 (from 12 V to 3.3 V) and LM7805C (from 12 V to 5 V). In a real implementation (with a definitive CPU speed and using SMD elements), it would be possible to select the lowest one and made the power distribution system easier.

Next figure shows MAX232 device, which is the circuit proposed by the manufacturer to supply RS-232. The only difference is that two serial ports are driven by this circuit using the channels usually destined to flow control for driving another serial port.



MAX232 device

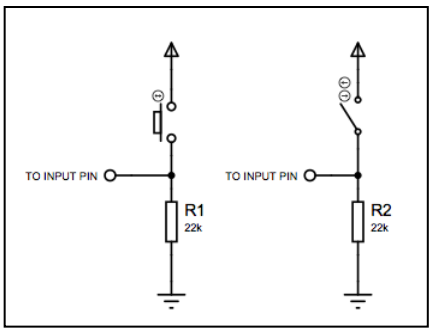
AVR STK 500 kit has been used to programme the prototype. Besides, a 6 pin header, which configuration is shown below, has been included in the design because it connects the programmer with the SPI port of the device used for programming. In this case, the system has its own power supply and the VTG it is not connected in the prototype board.



Programming header

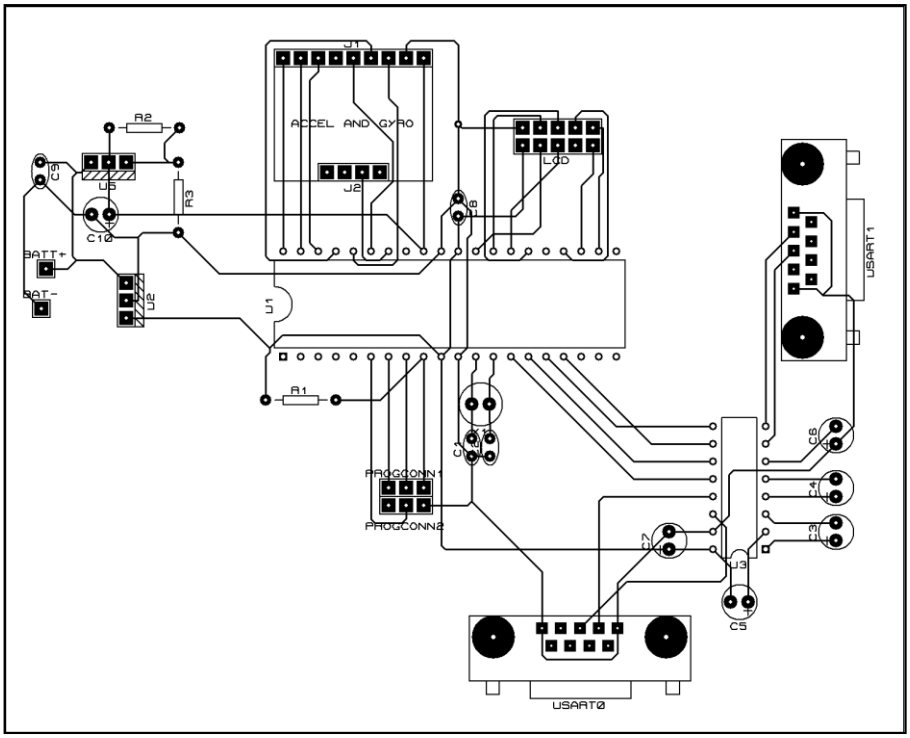
The gyroscope used is able to provided two kind of outputs per axis, the basic ones has the maximum DPS range ± 500 dps, while the second ones reduce his output rating to ± 110 dps (4 times amplified version from the first one). Therefore, a set of jumpers has been included to select the kind of output.

Lastly, the circuit used for the buttons and switches is designed to avoid floating pins in any conditions. Following figure shows examples of these circuits.



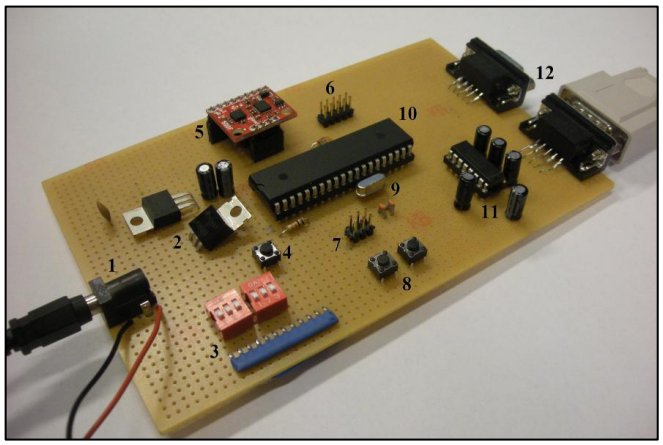
Buttons and switches circuits

As a result, the following design shown below has been made by using "Proteus". Although this program is used to design printed circuit boards, it has been used in order to follow the connections and place the components.



PCB design

After soldering the wires and placing the components, the laboratory prototype board has been built with dimensions 16 x 10 cm. This is the result:



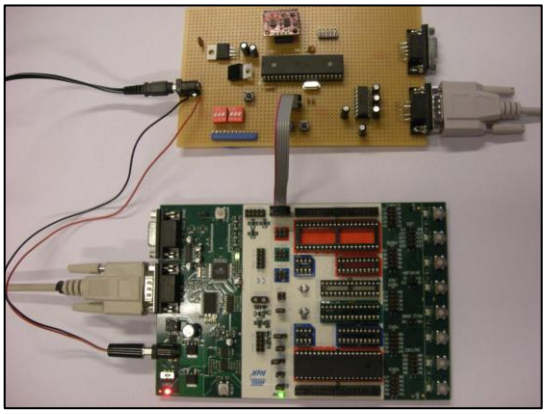
Prototype Board

where:

1. Power supply
2. Regulators circuit
3. Sensors circuit
4. Reset
5. Unit Combo Board (ADXL335 accelerometer & IDG 550 gyroscope)
6. LCD pins
7. Programming header
8. Buttons and switches circuit
9. Xtal oscillator

10. AtMega324P microcontroller
11. MAX232 driver
12. USARTs

Finally, following figure presents how to connect the AVR STK 500 kit to the prototype board in order to program it.



Motorcycle Emergency Call system

- UART communication

The microcontroller communicates with GPS and GSM through a serial port. This port is used due to its simplicity, low cost and its console function is standardized. Besides, the library was already developed and MIC1 course, completed during the first semester at IHA, so it provided a small background in serial communication.

The serial communication is standardized under the RS-232 norm. It allows for serial communication, asynchronous, and full duplex between the two connected elements. Here it is explained the pins that the serial port uses.

	9 Pin Connector on a DTE device (PC connection)
Male RS232 DB9	① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

9 pin connector on a DTE device

DTE (Data Terminal Equipment) corresponds to the device connected to the computer, while DCE (Data Communication Equipment) is connected to the PC. Note that, since the communication is duplex, this is only a convention and the communication can be reversed.

Voltage sent over the pins can be in one of two states: On and Off. On (binary value "1") means that the pin is transmitting a signal between -3 and -25 volts, while Off (binary value "0") means that it is transmitting a signal between +3 and +25 volts.

The handshake protocol aims at allowing the two devices that are both “on and ready”. The DTR and DSR pins are usually “on” all the time and are used to signal from each end that the other equipment is actually presented and powered-up.

Flow control is very important. With it is possible to stop the data flow from one of the connected elements before it overruns the other element’s capacity to receive the information. The device sending data is constantly sending a signal on the “Request to Send pin”, and checking for a signal on the “Clear to Send pin”. If there is no “Clear to Send” response, it stops sending data, waiting for the “Clear to Send” before it resumes. This allows the flow of data to run smoothly.

Data is sent through the bits 2 and 3 using the UART communication protocol. This protocol takes on byte of data and sends it bit after bit, one bit at a time (thus giving the serial port its name). The byte is enclosed in a UART frame, as seen below.

UART frame

The rest state of the line is a logical 1 state (-3 to -25V). Before each byte of data, a start bit is sent, which is a simple 0 bit. Then, the byte is sent bit after bit, starting with the least significant bit. After the byte is sent, the UART protocol also requires sending a stop bit, telling the receiver that the transmission for this byte is finished. A parity bit can also be used at the end of the frame after the stop bit. This bit is used to unsure the byte has been correctly sent. It corresponds to the number of logical ones in the byte and can be either even or odd. However, most of the time the parity bit is not used.

- AVR Demo board

The device called AVR demo board has been used to view all the information received by the GPS, the number of passengers on the motorcycle and management of Motorcycle Emergency Call. Indeed its size is adequate and can be conveniently installed on the dashboard of a motorcycle.

AVR demo board contains a LCD (Liquid Crystal Display) screen which name is HD44780U. This device has the following features:

- 5 x 8 and 5 x 10 dot matrix possible
- Low power operation support: 2.7 to 5.5V
- Wide range of liquid crystal display driver power: 3.0 to 11V
- Liquid crystal drive waveform: A (One line frequency AC waveform)
- Correspond to high speed MPU bus interface: 2 MHz (when Vcc = 5V)
- 4-bit or 8-bit MPU interface enabled
- 80 x 8-bit display RAM (80 characters max.)
- 9,920-bit character generator ROM for a total of 240 character fonts
    - 208 character fonts (5 x 8 dot)
    - 32 character fonts (5 x 10 dot)

Next figure shows a real HD44780U LCD Screen.

HD44780U LCD Screen

SOFTWARE

- Inputs

In this section two important parts about the communication between the microcontroller and two specific devices: GPS receiver and GSM module. NMEA protocols for GPS and AT commands for GSM are vital to understand the behaviour of these devices.

GPS

NMEA Protocol

GPS receiver communication is defined within the specification NMEA (National Marine Electronics Association), which means that the receiver expects data to be in NMEA format in order to understand real time position information.

This data includes the complete PVT (position, speed, time) solution computed by the GPS receiver. The idea of NMEA is to send a line of data called 'sentence' that is totally self contained and independent from other sentences. All of the standard sentences have a two letter prefix which defines the device that uses this sentence type. For GPS receivers the prefix is GP, which is followed by a three letter sequence that defines the sentence contents.

Each sentence begins with a '$' and ends with a carriage return/line ("\r\n") feed sequence and can be no longer than 80 characters of visible text (plus the line terminators). The data is contained within this single line with data items separated by commas, e.g.:

"$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47"

where:

- GCA: Global Positioning System Fix Data
- 123519: Fix taken at 12:35:19 UTC
- 4807.038,N: Latitude 48 deg 07.038' N
- 01131.000,E: Latitude 11 deg 31.000' E
- 1: Fix quality: 0 = invalid
  1 = GPS fix (SPS)
  2 = DGPS fix
  3 = PPS fix
  4 = Real Time Kinematic
  5 = Float RTK
  6 = estimated (dead reckoning) (2.3 feature)
  7 = manual input mode
  8 = Simulation mode
- 08: Number of satellites being tracked
- 0.9: Horizontal dilution of position
- 545.4, M: Altitude, Meters, above mean sea level

- 46.9, M: Height of geoid (mean sea level) above WGS84 ellipsoid
- (empty field) Time in seconds since last DGPS update
- (empty field) DGPS station ID number
- *47: The checksum data always begins with *

There is a provision for a checksum at the end of each sentence which may or may not be checked by the unit that reads the data. The checksum field consists of a '*' and two hex digits representing an 8 bit exclusive OR of all characters between, but not including the '$' and '*'.

For this application, only the command "GPRMC" is needed. This command corresponds to the version of essential GPS PVT (position, speed, time) data. It is called RMC (The Recommended Minimum) and looks similar to:

"$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A"

where:

- RMC: Recommended Minimum sentence C
- 123519: Fix taken at 12:35:19 UTC
- A: Status A=active or V=Void
- 4807.038,N: Latitude 48 deg 07.038' N
- 01131.000, E: Longitude 11 deg 31.000' E
- 022.4. Speed over the ground in knots
- 084.4: Track angle in degrees True
- 230394: Date – 23rd of March 1994
- 003.1, W: Magnetic Variation
- *6A: The checksum data, always begins with *

GPS library

For this project it has been necessary to develop some functions able to manage with the hardware. These functions have been grouped in libraries, to make the maintenance easier. In this case, Following table shows the library for GPS.

Variable	Type	Size (B)	Comments
status	char	1	0: Fix not available or invalid 1: GPS SPS Mode, fix valid 2: Differential GPS, SPS Mode, fix valid 3-5: Not supported 6: Dead Reckoning Mode, fix valid
tmpMsg	char[]	120	GPS data receive buffer
compMsg	char[]	120	Message completed
tmpMsgIndex	unsigned char	1	Index for tmpMsg
lastMsgComplete	unsigned char	1	Sign that the last message is complete
Error	unsigned char	1	Sign errors conditions
latDeg	unsigned char	1	Latitude degree
latMin	unsigned char	1	Latitude minutes
latDecMin	int	2	Latitude decimal part of minutes

Variable	Type	Size (B)	Comments
latDir	char	1	North (N) or South (S)
lonDeg	unsigned char	1	Longitude degree
lonMin	unsigned char	1	Longitude minutes
lonDecMin	int	2	Longitude decimal part of minutes
lonDir	char	1	East (E) or West (W)
satell	unsigned char	1	Number of satellites fixed
altitude	int	2	Mean sea level altitude
altUnit	char	1	Height unit
timeHour	unsigned char	1	Hours in GPS time
timeMin	unsigned char	1	Minutes in GPS time
timeSec	unsigned char	1	Seconds in GPS time
timeDecSec	unsigned char	1	Tenths of second in GPS time

GPS structure

GSM

AT commands

As it is said before, the microcontroller communicates with the GSM Modem sending different AT commands, which are explained below.

- **AT**
  "AT" is used simply to check if the connection between the microcontroller and the Modem is working. If everything works perfectly the answer is "OK", otherwise if there is something wrong the answer will be "ERROR".

- **AT+CPIN=Pin_Number**
  This command is used for entering the Pin code to the SIM card inside the Modem. If it is the correct number, an "OK" is returned, and if not an "ERROR".

- **AT+CMGF=1**
  It is used for choosing the SMS format. In this case, type "1" is going to be used. It means a text format SMS mode. It returns an "OK" if there is no problem and an "ERROR" if something goes wrong.

- **AT+CREG?**
  "AT+CREG?" is the command used for asking about network registration. The network usually is set up just after the PIN code is introduced. But the main programme is going to be constantly throwing this command in order to be sure if the Modem is still covered. If there is no network or the network is not checked in yet, the answer is "0" or "2", but it will be "1"if the network has already checked in.

- **AT+CMGS="tel_number" \r text CTRL+Z**
  This is the command used for sending the SMS with all the information of the accident. "tel_number" is the telephone number of the emergency services (usually 112) and "text" is the message containing key information.

The way the microcontroller sends the commands to the Modem is creating different character chains (strings) and sending them through the UART. After each command a carriage return is always added (\r).

The answers are processed in the same way. There is an interruption that takes all of the characters that USART receives, and then, various conditions validate if each answer is correct.

GSM library

For this project it has been necessary to develop some functions able to manage with the hardware. These functions have been grouped in libraries, to make the maintenance easier. In this case, following table shows the library for GSM.

Variable	Type	Size (B)	Comments
status	unsigned char	1	0: System not initialized 1: PIN needed to be introduced 2: Modem respond to AT command 3: PIN introduced, need set SMS mode 5: Modem ready 6: Sending a command 7: Sending SMS 10: Network connection not available 11: Network connection available
cmd	char[]	200	Command to send or last command sent
sendIndex	unsigned char	1	Index for cmd
cmdLength	unsigned char	1	Length for the command
response	char[]	200	String for the response
respComp	char	1	Marks if the answer is completed or not
lastCmdStat	unsigned char	1	0: Waiting for first character response 1: Character received >10: Retrying to send the command
statString	char[]	20	String for debug screen
index1	int	2	First index for reception
index2	int	2	Second index for reception
cont	char	1	Counter for periodic check for the modem
maxCont	char	1	Counter for periodic check for the modem
WaitingCmdResp	char	1	System waiting for the command response
CREGvalues	char[]	2	Values for the result of the command AT+CREG
tmr1mode	unsigned char	1	Functionality of tmr1: 0: Not in used 1:Waiting 5 seconds 2: Waiting 60 seconds to resend network status command.

GSM structure

- Acquisition and data processing

Analogue-to-digital converter (ADC)

The analogue signal from the sensors is reading sequentially by the ADC. The frequency for each channel depends on the number of channels being used. Thus, taking values from all the channels the frequency per channel is around 1 kHz.

The reading of this value is made by using the interruptions capabilities provided by the hardware. This makes the system able to do more tasks meanwhile the conversion is being made.

To manipulate the values provides by the ADC, it has been studied different solutions. The first is to convert the value provided in LSB (Less Significant Bit) to a float number and store it making the calculations in the same interruption service routine (ISR).

Since making time measurements used by the ISR is too much time, it has been decided to move this operations to an external function called from the main while(1) loop.

It has been found that this operation is too expensive in time and the final implementation uses directly the LSBs base value to make comparisons. Only if a final value is needed for representation or debugging, it will be calculated.

All sensors used have a bias voltage, this means the 0 value of the variable measurement is not in 0 volts. For this reason is needed to convert the biased or unsigned value, to a signed value.

To interpret the values provided by the sensor is necessary to use the values for bias voltage and sensitivity provided by the manufactures, but, in some occasions, these values suffers variations depending of external conditions, such as temperature, or even from fabrication faults.

This conversion is made during the interruption routines, to ensure the use of the most updated value. Later in the main loop, these values are interpreted to calculate firstly the partialSum and lastly the globalSum.

Configuration structure

One characteristic desirable for this kind of systems is the easiness to adapt or reconfigure the system for the client. For this reason, it has been decided to storage all the configuration values in order to make the access easier, and also allows the possibility to change them without the necessity of access to the source code. Therefore, configuration values are storage in the EEPROM because this memory can be reconfigured with basic tools without changing the code. This means that the system is able to be adapted to multiple environments.

The values are storage in order, and ready to be read and interpreted according of the type of variable It also includes some markers, always defined with ";" (3B in hexadecimal) to facilitate the reading function to determine the correct position of the values. In this part is possible to improve the system adding a CRC code able to check the consistency of the data storage. Next table shows the configuration structure.

Variable	Type	Size (B)	Start byte	Comments
version	char[]	5	0	Software version
plate	char[]	10	5	Vehicle plate
emeNum	char[]	15	15	Number to send the SMS
sec2Call	unsigned char	1	30	Seconds to make the call
lim01AccelX	int	2	31	First limit for this axis
lim12AccelX	int	2	33	Second limit for this axis
lim01AccelY	int	2	35	First limit for this axis
lim12AccelY	int	2	37	Second limit for this axis
lim01AccelXY	int	2	39	First limit for this axis combination
lim12AccelXY	int	2	41	Second limit for this axis combination
lim01DpsY	int	2	43	First limit for this axis
lim12DpsY	int	2	45	Second limit for this axis
lim01DpsX	int	2	47	First limit for this axis
lim12DpsX	int	2	49	Second limit for this axis
GsmPin	int	2	51	Pin for the mobile SIM
limGlobal	unsigned int	2	53	Trigger value for global addition
limPartial	unsigned char	1	55	Trigger value for partial addition
limResetIter	unsigned char	1	56	Times in a row of partial=0 to reset value.
gpsBaudRate	unsigned long	4	57	Baud rate for the GPS module
gsmBaudRate	unsigned long	4	61	Baud rate for the GSM modem
senPreFactor	unsigned char	1	65	Factor for risk algorithm
accelFactor	unsigned char	1	66	Factor for risk algorithm
gyroFactor	unsigned char	1	67	Factor for risk algorithm

Configuration structure

System functions

As said before, for this project it has been necessary to develop some functions able to manage with the hardware. These functions have been grouped in libraries, to make the maintenance easier. In the following lines some functions are explained (the ending, inputs and outputs)

Main file

In this file is declared all the global variables for each type of structure and also the main loop, where the main algorithm is executed, is located here.

Sensors

This file provides all the function for the configuration, reading and conversion of the sensors analogues and digital used by the system for detection of different situation.

interrupt [ADC_INT] void ADC_interrupt();

This interruption function is called when the ADC has finished a conversion. The function changes the channel to convert and restart another conversion, previously saved the result in the vector "ADCresult" and signed the existence of new data to convert.

- **Input:** The value converted by ADC, read from ADCW.
- **Output:** According to the channel converted, the value provided by the ADC (unsigned) is storage in the array called "ADCresult" and a signed version is storage in "lADCconv" array.

interrupt [PCINT1] void PC1_interrupt();

It responds to events in the assignment to the ports related to this interruption. The interruption checks which pins have change and update the variables. In this case the pins are connected to the rider sensors located in handlebars and brackets, but the pins from the passenger are also checked to ensure the most updated information.

- **Input:** Pin status assigned to the interruption.
- **Output:** A number which represents the number of sensors activated.

interrupt [PCINT3] void PC3_interrupt();

It responds to events in the assignment to the ports related to this interruption. The interruption checks which pins have change and update the variables. In this case the pins are connected to the rider sensors located in handlebars and brackets, but the pins from the passenger are also checked to ensure the most updated information.

This interruption is also executed with user interface buttons. The new screen to load in "desiredScreen" is defined in two, one for each button, where this screen is indexed by the number of the actual screen.

- **Input:** Pin status assigned to the interruption.
- **Output:** Number representation of the number of sensors activated. The screen changes if a user interface button is pressed.

interrupt [TIM0_OVF] void tmr0ovf()

It is very common to have rebounds in mechanical buttons and switches. This function, combined with some code in the port change interruption, allows avoiding these effects. When a switch changes the value or a button is pressed, the system can disable it during a short time (around 20 ms) in order to eliminate false calls to the ISR caused by these rebounds.

The goal of this function is to re-enable the ISR for the elements when the fixed time has passed. To configure this function, each element is represented by 1 bit in the

variable called "SENSORSTR.antiRebound". If the bit is one, the interruption for this bit is disabled if is called, and then re-enabled.

unsigned char ADCinit();

The purpose is to configure the ADC and start the first conversion. The following conversions are started by the interruption subroutine.

- **Output:** The returned value is 0 if the configuration has success and 1 if a problem is detected.

unsigned char DigitSensInit();

This function configures the ports able to monitoring the digital sensors and the buttons. In this case, the system is monitoring 6 sensors for brackets and handlebars and buttons for the UI.

unsigned char updateFloat(char slctchann);

The goal is to update the float value in the array for the channel selected. This function is going to be used for representation and debug and it is not called frequently because its high CPU time consume.

To test the efficiency of each kind of conversion, a lab test has been designed where the main loop is converting the same channel alternatively with each method. The results of the test should sign what types of operation are faster and also more precise in the conversion.

- **Input: "**char slctchann" (channel to convert).
- **Output:** The function provides the number of channels converted.

sensorCalibration

This function tries to calibrate the sensors. I.e. in this gyroscope it has been detected some variations between the manufacture output specification and the real ones, such as the 0 bias voltage value which is different in each axis. This function should be called with known conditions of the motorbike to ensure the correct calibration.

uartX

This library provides the basic functions to use the UART peripherals embedded in the microcontroller. It is manly developed in the past semester in the subject MIC1, but it has been updated now and added some functionality for this project.

Many functions are defined in these files, but the used ones are:

InitUartX (unsigned long BaudRate, unsigned long XTALFrequency, unsigned char Databits, unsigned char Parity)

This function is in charge of configuring the UART hardware to work by the parameter specified in the call.

- **Input:** unsigned long BaudRate (Baud rate desired for the communication, this value is expressed in bauds per second and is recommended to used one of the standard baud rates, such as 1200, 2400, 4800, 9600, 14400 or 57600).

unsigned long XTALFrequency

This value is the crystal frequency used by the device as main clock, is expressed in Hz.

unsigned char Databits

Number of bits for the communication, the most common is 8 bits.

- **Input: "**unsigned char Parity" (Enable parity if needed for error detection, the possible values are: 0 for no parity, 1 for even parity and 2 for odd parity).
- **Output: "**unsigned long" (the output is the final baud rate, which is important because only some discrete values are possibility generate by the UART, and depends on the crystal frequency).

UartXSendStringSramInt(char String)*

This function starts to send the character array String and configures the system to make possible to finish the transmission using interruptions. This means that the CPU can make other task meanwhile the UART peripheral is sending the bits of each character. When the shipping is complete, the appropriate ISR is called and loaded the next character to send until the array is completed.

interrupt [USARTX_RXC] void UARTX_RX()

This function is the ISR (Interruption Service Routine) and it is executed when the USART is receiving information (by interruptions) and a new element is complete. This function saves the value to allow the UART to receiving a new one.

These functions have the problematic to detect the end of a frame. Two different solutions are used for each one of the UARTS:

In the case of GPS, the frame always starts with the same character "$". When this one is received, the ISR marks the condition and the frame process function starts.

In the GSM case this is not possible because any character is common to all the transmission and the solution is based on time. If no new character is received in 5 seconds, the function takes the frame as complete and marks it for a later analysis.

interrupt [USARTX_DRE] void UARTX_TX()

This function is the ISR and it is executed when the USART is receiving information (by interruptions) and the outgoing buffer is empty. The function loads a new value in it to continue the operation.

CONFIGlib

char readEepromConfig(struct strConfig str)

It reads the configuration values for the system from the EEPROM. The function returns an error condition when the structure of the data (saved in the EEPROM) is not correct. The function expects to find values separate by semicolon, but not check the validity of the data read.

- **Input:** Structure where the information should be saved.
- **Output:** It returns a 0 if the reading successful and a 1 it there is an error in reading.

Errorlib

char systemError (char errorSource)

The program executes this function if the system suffers and error that invalidate the system. The function warns the user by the LCD and sends the error code by the UART0 serial port to diagnosis.

- **Input:** "errorSource" (it differentiates the sources of error explained in next table).

ID	Devices Source	Diagnosis
1	Memory	System not able to read the configuration from EEPROM or external memory
…		
11	SerialPorts	UART0 obtained baud-rate different from desired one
12	SerialPorts	UART1 configuration function not able to finish correctly
13	SerialPorts	GPS is not sending information
14	SerialPorts	MODEM not responding
…		
21	ADC	ADC not correctly initializated
…		
30-50	GSM	Error + CMD: modem not responding sending CMD
51	GSM	SIM card not inserted
52	GSM	PIN related problem, probably need to unlock with PUK
53	GSM	SMS mode related problem

Error library

- **Output:** Warning the user with a unique configuration of lights in the UI and try to configure the serial ports to send the diagnoses for reparation.

GPSlib

This library is in charge of managing the GPS connection and it uses the function defined in the UART0 library.

InitGps

This function initializes the system to use the GPS, mainly configuring the UART and enable interruption for reception.

GpsStringManage

This function first makes a copy of the GPS frame to another array to avoid lose values by overwrite. Then, it extracts the values from the GPS NMEA frame and storage them in the corresponding variables.

GUI

This library is used to refresh the user interface, which in this case is the LCD screen. To do this, it uses a function defined in the lcd162 library.

Char refreshSCR()

The purpose is to update the status of the user interface. When the function is called, it tests if "desiredScr" is different to "actualScr" (in this case, checks the existence "undesiredScr"). If this is valid, the function copies the value to "actualScr" and refreshes the user interface.

- **Output:** char (0 if the execution was correct and the new screen is valid and 2 if desiredScr is invalid).

GSMlib

This library is used to communicate with the GSM modem. The protocol is called AT-commands and works over a serial port connection.

char GsmInit()

The goal is to configure the hardware to be able to communicate with the modem.

- **Output:** it returns a 0 if the configuration is corrected and a 1 if there is a problem with the baud rate configuration.

char sendCommand(char command)

This function sends the different commands to the modem. The process is executed using interruptions and the function also configures the system to receive the answer also by interruptions.

The output of this function, in case of correct execution, only means that the process to send the command started fine, but it does not mean its correct execution because this can be later than the function return. It depends on the longitude of the command and the time which takes the modem to be executed.

- **Input:** A number defining which command is going to be sent.
- **Output:** It returns a 0 if there is a correct start in the command send process, a 1 if the system is waiting the answer for another command, and a 2 if the command is not recognized.

char resendCmd()

This function is used for resend the last command. I.e. in case of a communication failure.

interrupt [TIM1_OVF] tmr1ovrf()

This ISR is used firstly for monitoring the answer of the modem when characters are sent to the modem. Then, it echoes it in case of the first character in the string. If this not happened in 5 seconds, the functions assumed a problem in the modem and it will retry to send the command a fixed amount of times, depending of the command.

Secondly it is used to determine when the modem finishes the answer. The system resets this timer when a new character is received and when any one new character is received the function assumed the end of the response.

void analyseResponse()

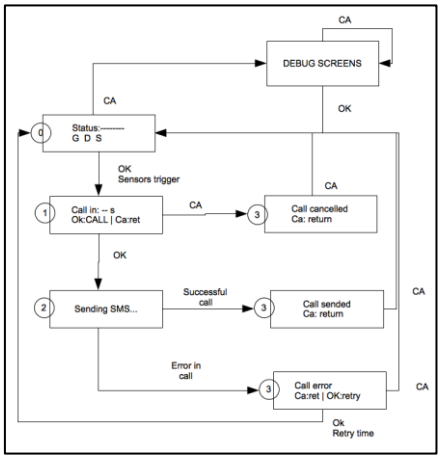
This function analyses the response sent by the modem. To make this possible, the number of the command sent is stored in a variable until this function is called. The function also implements the sequence of commands for the initialization started by the "GSMInit" function. All the major errors occurred in the execution of commands are informed by using the systemError() function and the minor ones are try to be solved.

lcd162

This file contains functions to manage an LCD 16x2 compatible. It is a kind of standard library developed by the authors in the previous semester in MIC1.

- User interface

Figure below shows a diagram which represents the content of the screen and its sequence depending on the user or system actions.



Screen diagram

In the diagram, OK and CA means the user has pressed the button OK or cancel respectively. The design of this screen are the most simplest as possible and only presents the basic information to avoid distraction with it.

The main screen shows in the first line the general status of the system while in the second line is shown the status of the GPS, GSM and sensors subsystem. Spearing a G means the GPS position is available, a D means the GSM modem is working correctly and a S means the sensor are working in the expected values.

When the call procedure is started, the system shows a countdown which can be avoid if the user press OK or even cancelled if was an error.

The software has two arrays indexed by the number of the actual screen and tells the system with screen to load, depending on which buttons is pressed one or the other array is used.

Finally, the screens in the upper right part are designed to see the system running, only for debug purposes.

Universidad de Alcalá