CHALMERS

Optimization of load distribution in washing machines using bio-inspired computational methods

APPLE MAHMUD
EDGAR CUELLAR MONDRAGÓN

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems and Division of Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010
Master's Thesis 2010:55

MASTER'S THESIS 2010:55

Optimization of load distribution in washing machines using bio-inspired computational methods

APPLE MAHMUD
EDGAR CUELLAR MONDRAGÓN

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems and Division of Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2010

Optimization of load distribution in washing machines using bio-inspired computational methods
APPLE MAHMUD
EDGAR CUELLAR MONDRAGÓN

Master's Thesis 2010:55
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems and Division of Dynamics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2010

Optimization of load distribution in washing machines using bio-inspired computational methods

APPLE MAHMUD
EDGAR CUELLAR MONDRAGÓN
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems and Division of Dynamics
Chalmers University of Technology

Abstract

This thesis deals with the optimization of the laundry load distribution within front loaded washing machines in order to avoid vibration noise and, in consequence, increase the performance and duration of such systems. The washing machine is programmed to follow several cycles in order to clean the laundry. The higher levels of vibration are reached when the motor executes the drying cycle where the laundry will spin at a very high speed in order to remove the water by using the generated centrifugal force.

The aim of this thesis is to find a method which, when used, obtains a new distribution scheme that will evenly distribute the laundry load. This scheme will reduce the vibration with a low cost for the implementation since the change will be made only in the software of the system. In order to evaluate a possible solution scheme, a hardware-in-the-loop (HIL) set up was created. The HIL set up makes possible to avoid the complex modeling of the system as the washing machine itself will run each scheme and give a real time feedback of the unbalance. To improve the load distribution a novel approach is explored by using Evolutionary Algorithms that have been proved to solve problems with similar complexity involving a high number of variables. A scheme was obtained after the execution of the evolutionary algorithm that achieved a better performance of 12% less unbalance with 9.8% lower standard deviation than the current distribution scheme used commercially. With these results the validation of this method to obtain optimal distribution schemes is demonstrated.

Keywords: Stochastic Optimization, Evolutionary Algorithms, Washing Machine, Hardware-in-the-loop

Contents

Preface

This work has been carried out from January to September in 2010 at the Department of Applied Mechanics, Division of Vehicle Engineering and Autonomous Systems and Division of Dynamics, Chalmers University of Technology, Sweden.

Aknowledgements

We would like to say our thanks to our supervisors at the Department of Applied Mechanics for their outstanding support and patience. Without their help this project would not have been possible.

The regular weekly meetings with Krister Wolff and Thomas Nygårds together was immeasurable. Thomas, with his prior knowledge about washing machines provided outstanding support regarding the hardware and many issues other than those. That always kept us one step ahead. On the other side when we got stuck with algorithm related confusion, Krister made the solution in an easy and supported manner.

Special thanks to Patrik Jansson from Asko Appliances AB who helped time to time by giving washing machine related technical information for experiment and software development. Professor Vikor Berbyuk was helpful through the entire project. He kept an open eye what was going on in the project and tried to guide in a proper way so that we can follow the path to the success.

Also we are grateful to our families and friends for supporting and encouraging us.

Göteborg September 2010
Apple Mahmud,Edgar Cuellar Mondragón

1 Introduction

Vibration dynamics is often one of the key issues in machine design due to the high impact on the lifetime and performance. It is known that vibration plays an important role in the lifetime of a system because as the vibration increases, the stress applied to the parts under such vibration also increases. A system whose vibration surpasses specified limit may fail to perform at its full capacity, become less accurate and less reliable and eventually stop to function properly.

The washing machine, a very common home appliance in most of the countries, is a system subject to vibration due to the nature of its task, which is to clean the laundry using centrifugal force and, if it is not taken handled correctly, high vibration noise can be present. Washing machines consist of a rotating inner chamber (better known as drum) where the laundry is loaded, a motor that will rotate the drum, suspension, an outer chamber (also known as tub), water and detergent inlets and outlets and an electronic control system consisting of sensors, controllers, actuators, and other auxiliary components.

According on how the laundry is loaded into the drum, washing machines can be divided into two main groups, the front loaded (FLWM) and top loaded (TLWM). Apart from the previous categorization, they can also be divided into horizontal and vertical axis washing machines, according to the orientation of the rotational axis of the drum.

Vibration is caused by the unbalance load in the drum during the spinning cycles, such as drying or rinsing cycle, where high speed is demanded. This unbalance within the drum will shift the main axis of inertia away from the spinning axis causing an irregular movement, and hence, vibration.

Laundry load is composed by single cloth pieces which can vary in size, weight, shape, and materials from each other. Usually when they are loaded into the washing machine they are mixed in a stochastic way. When the machine starts running the load will not be distributed evenly due to these variable initial conditions involved in the dynamics of the process. The unbalance load originated by the uneven initial distribution creates such vibration that the motor cannot reach the higher or desired speed and it will abruptly stops, resulting in an incomplete function, an error or even a machine failure. This is a worst-case scenario, many times the machine will complete the washing cycle with good results, but high vibration will wear off the parts faster than low vibration and the result will be a broken washing machine that will need to be repaired.

In an attempt to eliminate or reduce the vibration into acceptable limits, washing machine designers have introduced mechanisms or components in order to enhance stability such as dampers [4] making the design more complex and thus more expensive. These approaches make the total machine heavier by adding extra weights and most likely leads to additional space requirement inside the machine. Before the high-speed spinning can start, the laundry load needs to be redistributed inside the drum. In current front-loaded washing machines, the distribution stage is made by a fixed speed versus time motor scheme in order to rotate the drum in such a way that the load is redistributed evenly and therefore minimizing the imbalance in the drum.

1.1 Motivation

Nowadays, the demand of washing machines that will be placed in narrow and small specific spaces within the house is increasing, and also the interest in reducing to the lowest possible level the vibrations and noise of this appliances.

A current common tendency is to move the washing machine from basement to bedrooms, bathrooms or kitchens, therefore, the washing process should be quiet. The method

or approach to minimize the vibration in this project will focus on reducing the unbalance by evenly redistribute the load in the drum. In order to achieve this goal, it is necessary to find a more satisfactory speed scheme than the already used, which is the result of manual tests. This approach will have a low cost[1] and will increase the performance of the system.

The distribution schemes currently included in the washing machine have been developed over time by washing machine manufacturers and are based on experiences and manual testing.This has worked well but since the company changed the geometry of the drum and new bigger machines are being developed all the time, there is no guarantee that this old speed scheme is optimal for new geometry of the drum. In this thesis a novel approach to automate the generation of a speed scheme for washing machines with the aim of optimizing the load distribution and thus reducing vibration is investigated.

Optimization algorithms inspired of biological phenomena are stochastic methods that try to emulate or follow the behavior of natural biological processes. Evolutionary Algorithms (EAs)[2] are part of these stochastic optimization algorithms and they are well established methods for searching a good or satisfactory solution where the solution space is multidimensional and where there are many local optima. This type of algorithms has been used in many multivariable problems with outstanding results [2][3][5][6][8] making them a good approach to obtain a satisfactory result.

1.2 Purpose

The purpose of this project is to apply stochastic optimization algorithms, in order to find an optimal motor speed scheme that will evenly distribute the laundry load inside the drum. This optimal distribution will avoid drum vibration due to the unbalance load when high rotational speed is requested. A hardware-in-the-loop set up is going to be used where an external computer will communicate with the motor control unit, send the proper speed requests and receiving feedback from the motor as it has a built-in feature that can measure the unbalance.

The project consists of the integration of the washing machine hardware with an external computer that will be in charge of the data processing and the software to control and process data. This integration is made by an interface program that has to be developed under the specifications of the washing machine that is used for the experiments.

1.3 Limitations

Designing an optimal redistribution strategy for washing machine is quite complex due to multiple variables involved in the real washing process like laundry amount, washing water, laundry location, laundry dynamics or motor properties. Therefore it is not likely that the found solution to the load distribution problem is optimal one. Any solution which is better than current distribution will be considered sufficient.

The lumpy nature of soaked cloth made it hard to establish a mathematical formula regarding its motion against various force involved in the process. Even if the equation is formed that will not results similarly due to changed initial condition after each washing cycle run.Because each time after completing the washing cycle initially cloths location, orientation, moisture content every thing will be different from another run.

Other limitation in this project is that each time a speed scheme[3] is executed the laundry load placed in the drum will have a different initial position for each cloth. The

[1] As the implementation of the solution will consist in only changing the software in the washing machine.
[2] For a detailed description of EAs see section 4.1 in the next chapter.
[3] A set of speed commands sent at a determined time.

reason to set up random initial conditions is due to the fact that laundry users always put the laundry inside the washing machine randomly piled. Also to set the same initial conditions for all experiments is quite difficult and time-consuming which will make the experiments last way longer. Because of these conditions each execution, even if it is the same speed scheme, will give different outputs.Also the measurements given by the microcontroller is not directly related to a known unit which makes the interpretation of the results unclear but since the optimization only requires reducing the imbalance, this feedback shall work fine.

A washing machine is quite flexible when it comes to the amount of laundry load it can handle because normally it can be used with loads up to 8 kg. Each of these load conditions will probably have different optimal speed control scheme related to them. It is beyond the scope of this master's thesis project to evaluate all these possible cases, therefore only one representative case was examined. Nevertheless, the proposed method could be applied to other load cases as well without any major difficulties.

1.4 Related work

Modern washing machine performs the high speed spinning to extract water from the wet load leading to faster drying and less energy consumption during heated drying phase. The problem is that this high speed spinning is susceptible to unbalanced load due to uneven distribution inside the drum. This causes high vibration and noise which is unpleasant for the end user whom everyday is demanding better performance out of its appliances. That's why, the general vibration issue got researcher's and manufacturer's attention[1][7].

Yuan's work [11] was directed to accurately estimate the unbalanced loads, including their magnitudes and locations so that corrective action could be taken to reduce the unbalance and minimize vibration.Yuan used multiple mechanical and measurement instruments such as laser displacement sensors, accelerometers and load cells. He installed these components in different location of the washing machine to determine the unbalanced load information inside the spinning drum. At the same time a neural network based algorithm was used to estimate the magnitude and location of the load imbalance.

Sonoda was trying to develop a system to reduce the cloths imbalance while rotating in a drum type washer during spin-drying cycle. He proposed an extra additional balance box added to the spin drum containing an aqueous solution of calcium chloride was proposed *et al.* [10].They named their system as "G Fall- balancer". This control system has balance boxes added at the two ends of the spin drum. When the drum spins the liquid enters in the boxes and tries to minimize the imbalance depending on the amount and magnitude of laundry loads.

Another counterbalancing attempt was taken by Papadopoulos[7] in the specific case of portable washing machine vibration control. He addressed the problem in two ways. The first one is active balancing using only one balancing mass and the second one is by using two balancing masses.

2 Theory

A horizontal axis washing machine is loaded with laundry through the front-end hatch. During rotation of the drum cloths gets stuck to the drum's wall due to centrifugal force. The masses of the cloths are also subject to gravitational force as like any mass in the world.

When the drum starts rotation the load starts to show the dynamic behavior of mass's circular motion. To determine how the load affects the dynamics it can be start with draw-

ing the free body diagram (FBD) of the load mass inside the rotating drum as Figure 2.1. Here m represents the lumped mass of the load that remains unbalanced after distribution over the drum inside wall and ω represents the constant rotational velocity of the drum.

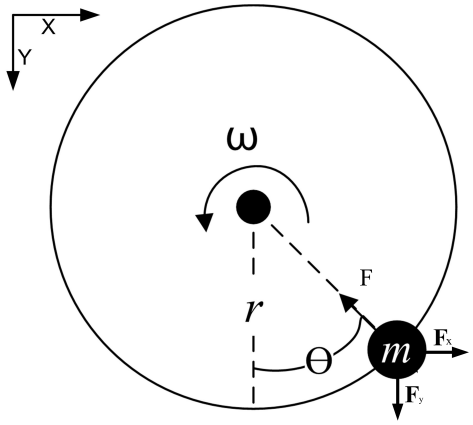

Figure 2.1: The free body diagram for the unbalanced mass in a horizontal axis washing machine.

Starting from the drum, the forces exerted from the rotating laundry mass on the drum are F_y and F_x in the vertical and horizontal direction respectively and g is the acceleration of gravity. [7]

$F_y = m\omega^2 r\cos\theta + mg$

$F_x = m\omega^2 r\sin\theta$

The vertical direction force that comes from this unbalanced mass is added to the gravitational force. In case of horizontal direction it is only affected by the amount of unbalance mass and rotational velocity. At high constant rotational speed ω the gravitional forces are comparatively small. So the forces acting on the unbalanced load can be written as follows

$$F \approx mr\omega^2 \tag{2.1}$$

In the equation 2.1 there are three factors, the load mass m, the eccentric distance r and the angular velocity ω which influence the unbalanced force that causes vibration. When the drum rotates at speed, ω remains constant and also the eccentric distance r will converge due to the load sticking against the wall within the drum. So the only factor that could be minimized to get low unbalance force F is the unbalanced mass m. In conclusion, by keeping the whole washing machine structure unchanged and minimizing the unbalance load we can reduce vibration.

There are two main solutions to minimize the unbalance as shown in Figure 2.2.

1. Removing the effective imbalance by counterbalancing.

2. Preventing the imbalance formation by distributing into an uniform layer around the drum wall.

This thesis focuses on the second option for minimizing vibration. In essence, if all the loaded cloths get uniformly distributed forming a layer of cloths against the drum wall, in such a way that, each unit of laundry mass is counterbalanced by another piece of laundry on the opposite side of the drum wall then the total system is balanced. Then unbalanced

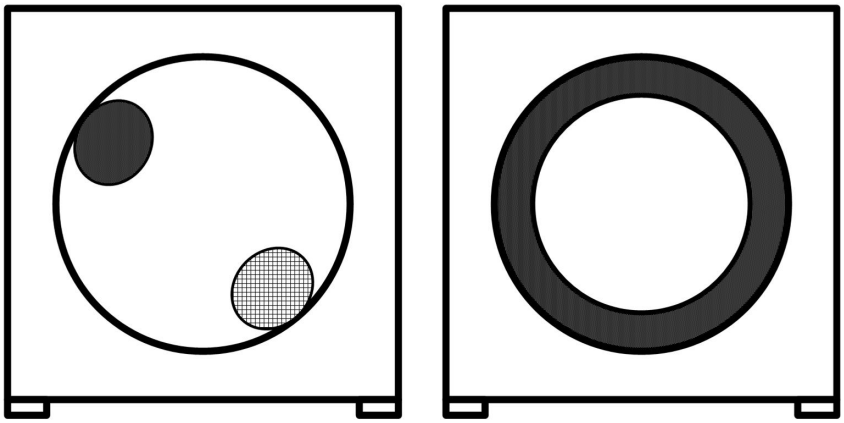

Figure 2.2: Example of solutions to minimize the unbalance: at left, Counter-balancing the load and at right, Uniformly distributing the load.

mass m gets a value of zero and the simplified system stops vibrating, as shown in Figure 2.2. Practically getting a zero unbalance would be almost impossible due to the lumpy nature of the cloths. We can only try to distribute those cloths as uniformly as possible around the drums inside wall so that the total of the unbalanced mass get a minimum.The search for this method for distribution is optimization of the load distribution in washing machine.

2.1 Speed scheme optimization

This thesis aims to achieve a uniform distribution of the cloths inside the washing machine drum by varying the rotational speed gradient. From experience it has been observed that depending on the rotation direction and gradient of the speed, cloths inside the drum get a different distribution. In a previous study it has been found that with a certain amount of load, different gradient follows different unbalance values. Table 2.1 and Figure 2.3 show that gradient variation affects the mean, median and standard deviation of the unbalance value.

The Table 2.1 shows the results for rotating same amount of cloths inside the drum with a speed gradient equal to 5 rpm/s several times. The motor measured unbalance value is dependent on the speed gradient. For the gradient 5 rpm/s the imbalance got highest value of 129 lowest value of 6 and average 72. Figure 2.3 shows how these values are roughly normally distributed over the range. With a speed gradient of 10 rpm/s the average of the observed values is 77 which is different from 5 rpm/s. Similarly 15, 20, 25 rpm/s also shows gradient effects on the cloths distribution and their corresponding resulted unbalance.

The objective is to find a specific speed scheme so that the cloths inside the drum get a reorientation resulting in a minimum unbalance. This speed control scheme can be represented by a certain number of sets of direction, speed and acceleration. Figure **??** shows the genes of the individual or chromosome which consist of 4 genes and each gene consists of 4 alleles[4]. Each allele is a variable in the system (direction, speed, gradient or acceleration and time) and each one limit its value to fit into a range that the software in the motor controller can interpret as correct[5]. The execution of the algorithm shall stop

[4]For further explanation relate to Section 4.2.

[5]Even though the motor controller make valid a parameter value does not mean that it can execute it

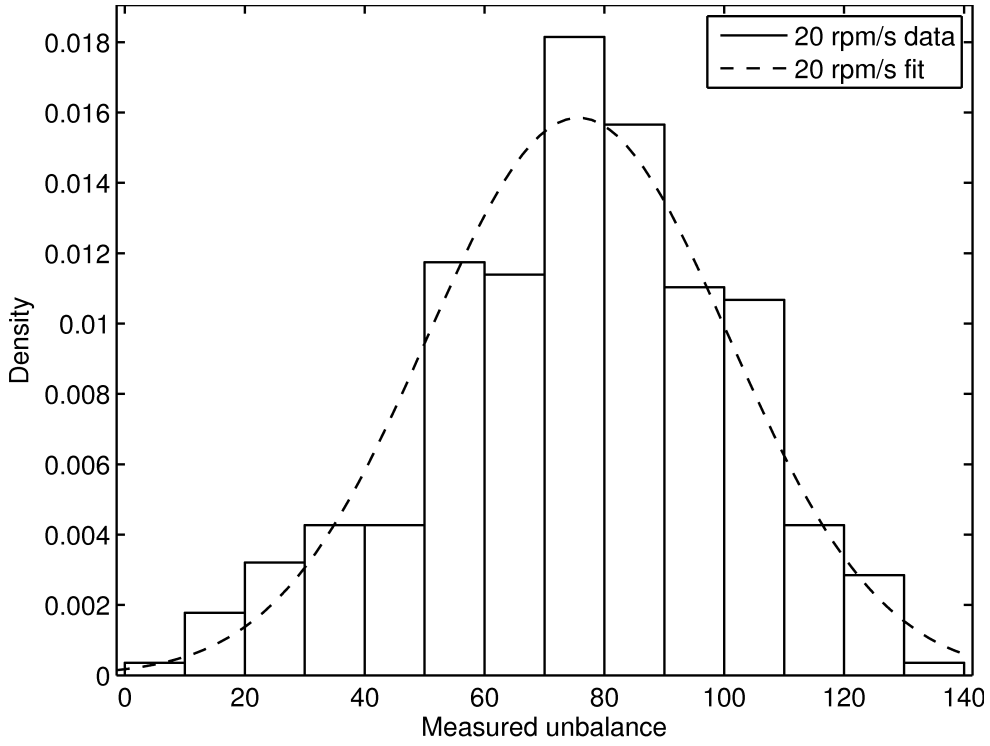

Figure 2.3: This figure shows a histogram, based on 280 data points, of measured unbalance data for speed gradient 20 rpm/s (solid lines). As seen in the figure the collected data resembles a Normal distribution (dotted line). The situation looks similar for the speed gradients 5, 10, 15 and 25 rpm/s, respectively. The data was provided by Asko-Cylinda.

Table 2.1: Mean, median and standard deviation value for unbalance measurements at different speed gradients (5, 10, 15, 20 and 25 rpm/second).

Gradient (rpm)	Measured unbalance value (unavailable unit)			
	Max	Min	Mean	Standard Deviation
5	129	6	72	37
10	135	8	77	31
15	127	9	77	15.5
20	135	9	76	28.9
25	131	8	74	39.3

after when the best individual of a generation satisfies the desired goal or after a certain amount of generations evaluated and the decision for choosing one or the other will depend on the problem in which the EA will be applied.

2.2 Hardware-in-the-loop

The system of freely moving laundry inside a rotating drum is hard to model with satisfactory results. While performing the EAs, the selection process of the speed scheme is not based on an analytical fitness function, rather it is a fitness function evaluated on hardware. A hardware-in-the-loop (HIL) system has been implemented to evaluate each individual. HIL is suitable to dynamical systems where the output is not simply a function of inputs, instead a function of the present inputs and some combination of past inputs.

mainly because of physical constrains within the hardware

Those systems also have numerous outputs for one specific type of inputs. As the output is correspondent to many factors so changing one specific input will results differently depending on the other factors.

HIL could be the best solution when it is really difficult to predict the real-time behavior and characteristic of the physical system. Because HIL will take the out put measurement directly from the hardware equipped with all the contributing factors in an actual operating condition. During washing process, rotating cloths dynamics is influenced by several variables like laundry amount, water content, gravitational force, drum rotation direction, rotation speed, laundry behavior, etc. So, it is really very much difficult to set a function to model how the cloths will be distributed inside the drum. This problem could be easily managed if the washing machine is directly connected in a loop with controller run by in interface software. The measured output will be the resultant of all the contributory factors associated in the washing process.

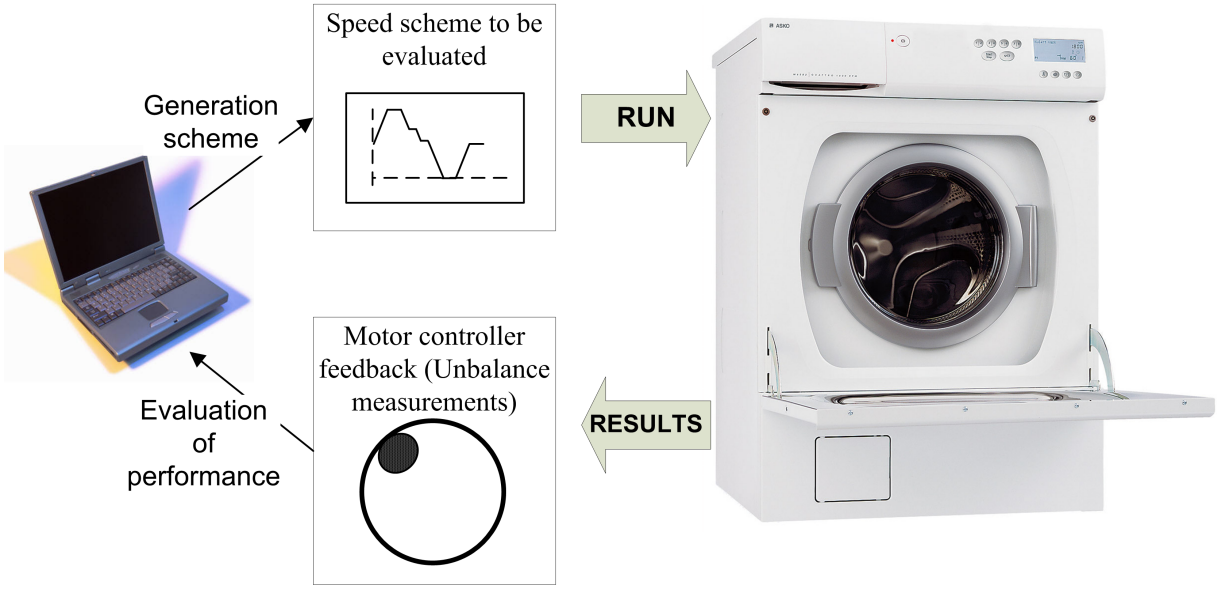

Figure 2.4: A physical washing machine in the fitness evaluation loop. It executes the speed scheme from interface software and motor controller sends the feedback of measured unbalance to the software.

Figure 2.4 shows the concept of HIL in washing machine speed scheme selection process. One typical HIL system consist of a controller, actuator, human machine interface (HMI) and an analysis platform. The idea in general term is to feed the motor controller with individuals (speed schemes) and record the unbalance feedback from the motor.

3 Experimental sensivity analysis

Before setting the experimental policy and designing the final experimental setup, some issue has to be decided first for an appropriate sensitivity analysis.

The user interface software (see Appendix A) can request unbalance measurements from the washing machine motor and each request contains two fields: unbalance at constant speed and unbalance at speed ramp. It was very important to take a decision which unbalance value should be considered for the speed scheme evaluation.

A problem arises because those measured unbalance values[6] were unavailable. So, it is hard to explain the unbalance value obtained from motor controller. The value should

[6] Measurement units was unavailable to the authority.

correlate the actual unbalance of the load inside the washing machine. It was important to interpret any obtained value from the motor as current unbalance inside the drum.

And also it was necessary to know whether the unbalance loads position inside the drum influence the unbalance value that is provided from motor intelligence system. If the load position inside the drum influences the unbalance value then this factor should be taken care of properly in the final experimental setup. The load should be properly placed at the front, mid or back position inside the drum so that positional factors are included in the unbalance analysis.

3.1 Method

This study was performed to determine how the measured unbalance value changes as the actual unbalanced load changes. The unbalance consisted of load weights ranging from 0 grams to 1200 grams placed at three different positions within the drum (front, middle and back). Figure 3.1 shows the position of the loads inside the washing machine drum.

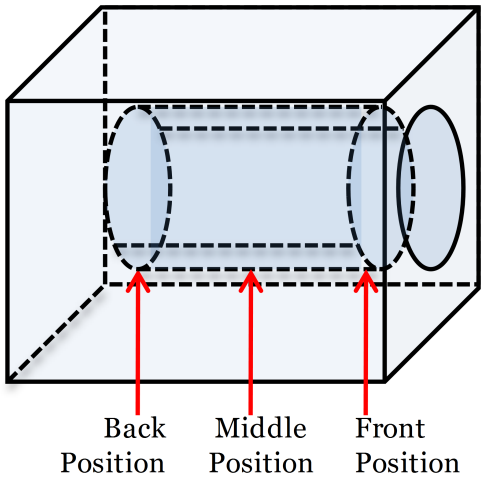

Figure 3.1: Unbalance load positions inside the drum that were used to test the feedback sent by the motor controller.

One specially designed test speed profile shown in figure 3.2 was run for all concentrated unbalanced loads and the three positions in consideration. The speed scheme was designed so that it had both speed gradient of acceleration and deceleration and two different constant speeds. The motor executed the same profile speed for all load ranging from 0 grams to 1200 grams concentrated metallic load and provide the unbalance value.

3.2 Data analysis

The currently available motor controller can request two types of unbalance measurements. These are unbalance value at constant speed and unbalance value at ramp speed. Consequently, all data collected during the experiment have been put in two different groups-

- Constant speed unbalance and
- Ramp speed unbalance.

Each group was then analyzed separately.

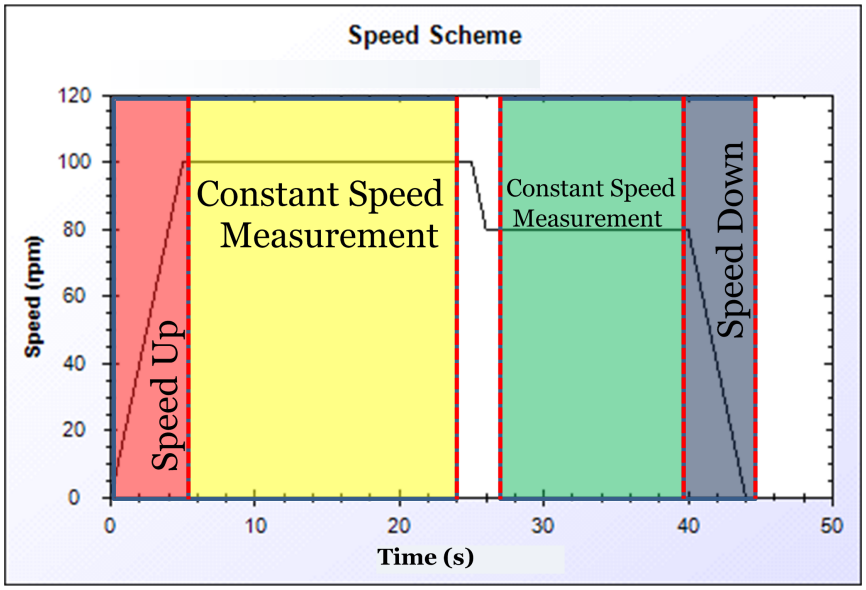

Figure 3.2: Test Speed profile for the unbalance value analysis.

Table 3.1: Unbalance at constant speed (front position).

load	0	100	200	300	400	500	600	700	800	900	1000	1100	1200
min	4	17	30	42	54	69	76	89	101	114	123	135	148
max	12	22	38	58	72	88	103	116	129	141	156	169	180
mean	5	20	34	48	61	77	87	99	111	124	136	147	160
mode	4	21	34	42	69	88	83	116	103	115	156	139	148
median	4	20	34	45	57	72	83	91	103	115	127	139	149

3.2.1 Constant speed unbalances data analysis

The motor controller measures the unbalance value while the motor rerates in a steady angular speed. All the constant speed unbalance value was measured at two different constant speeds. One of the speeds was 100 rpm and another was 80 rpm. It was also interesting to find out whether load position inside the drum really effects the measurement. So experiments were completed with actual load placed at three different positions (front, mid and back).

Unbalance at constant speed (front)

A range of unbalance weights from 100 grams to 1200 grams were placed in the front position inside drum and the experimental speed scheme were run with each load. The Table 3.1 shows all the unbalance values summery obtained loads ranging from 0 grams to 1200 grams for front position. Here, mean is the average of the values that were obtained for a specific load running several times. Similarly mode represents the most commonly available value from those tests. In case of constant speed unbalance value, Figure 3.3 shows a liner relationship between load and corresponding unbalance value. It is also clear that as the amount of load increased the range of measured unbalance value is also increased. Connecting the mean unbalance values of all the load cases from 0 grams to 1200 grams results in a straight line.

Unbalance at constant speed (Mid)

The graph of figure 3.4 indicates that keeping any unbalance weight at the mid position inside the drum and resultant obtained unbalance value has the same characteristics like

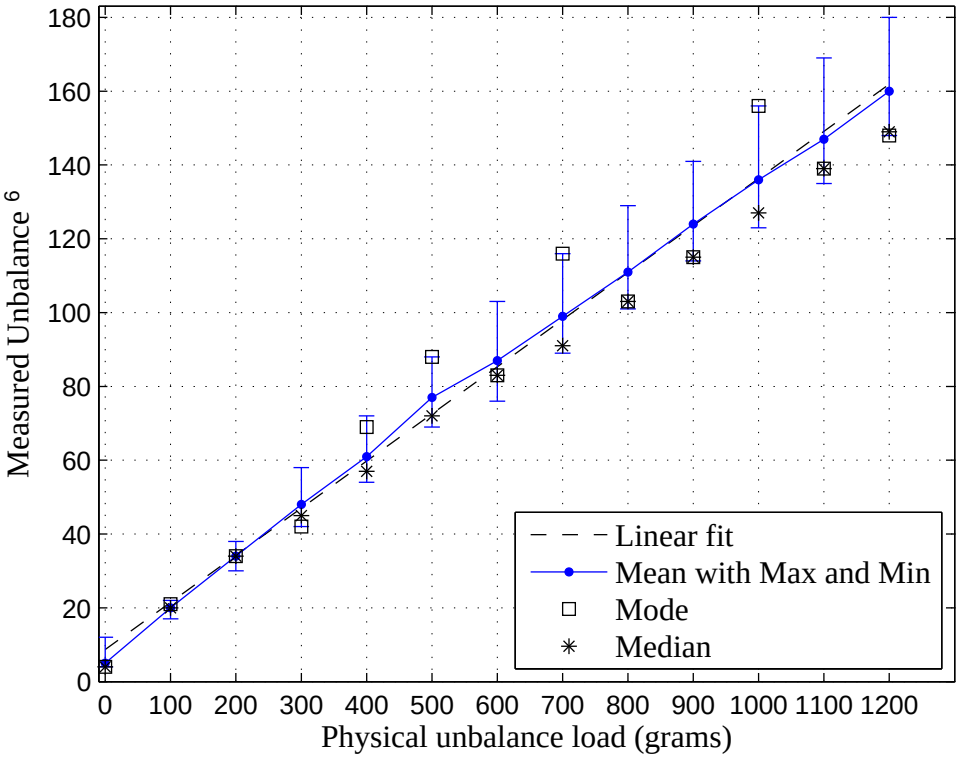

Figure 3.3: Unbalance measurements at constant speed vs. load at front position.

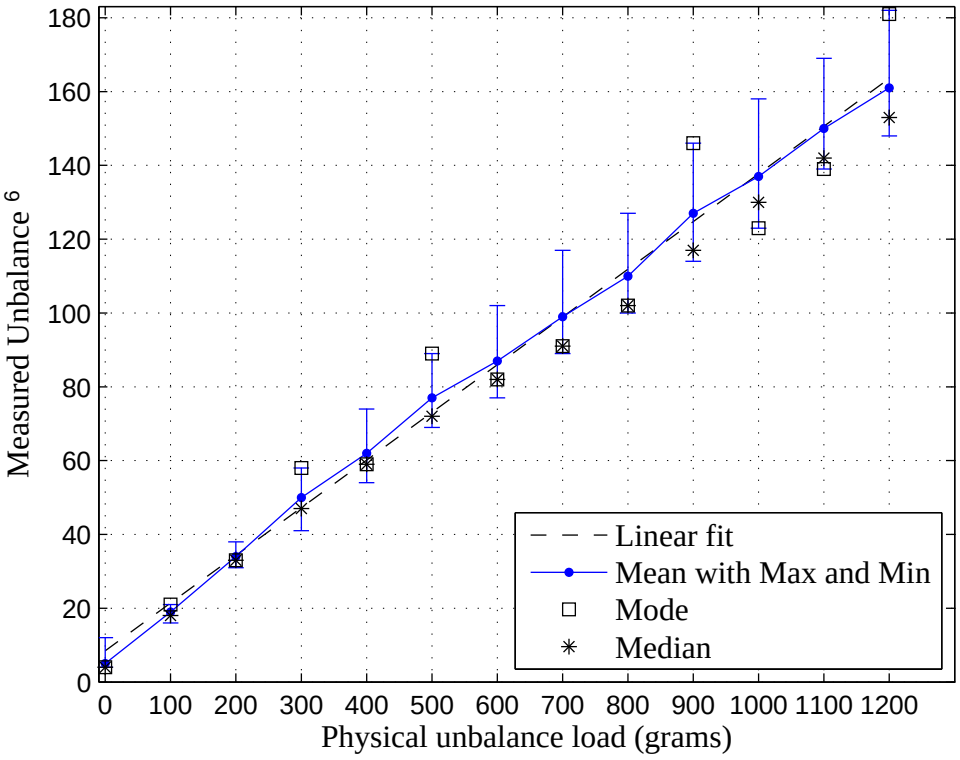

Figure 3.4: Unbalance measurements at constant speed vs. load at middle position.

Table 3.2: Unbalance measurements at constant speed (middle position).

load	0	100	200	300	400	500	600	700	800	900	1000	1100	1200
min	4	16	31	41	54	69	77	89	100	114	123	139	148
max	12	21	38	58	74	89	102	117	127	146	158	169	182
mean	5	19	34	50	62	77	87	99	110	127	137	150	161
mode	4	21	33	58	59	89	82	91	102	146	123	139	181
median	4	18	33	47	59	72	82	91	102	117	130	142	153

in case of front position. The measured unbalance value varies linearly with respect to actual unbalance weight placed inside at the mid position. The values provided by the motor controller were close to each other in case of smaller unbalance weight and as the load amount increase the range of values gets wider. For example the table 3.2 shows for an actual unbalance value of 100 grams the range is 5 (max: 21 – min: 16) where as in case of 800 grams it is 27 (max: 127 - min: 100).

Table 3.3: Unbalance measurements at constant speed (back position).

load	0	100	200	300	400	500	600	700	800	900	1000	1100	1200
min	4	17	29	42	54	69	77	89	100	111	124	135	148
max	12	21	40	58	74	89	102	117	127	143	157	168	182
mean	5	20	34	48	63	77	88	99	110	124	137	148	161
mode	4	21	30	42	60	88	102	89	127	116	157	168	149
median	4	20	30	45	60	73	83	91	101	116	126	139	149

Unbalance at constant speed (back position)

Putting the concentrated unbalance weight at the back position inside the drum did not change the unbalance values much. The measurement signal shows similar behavior as in the case of front and mid position placement. The Figure 3.5 shows the relation between actual unbalance weights placed at the back position with the corresponding measured values by the motor controller. This relationship is linear.

3.2.2 Ramp speed unbalances data analysis

All those ramp speed unbalance values were analyzed in two group of ramp-acceleration and ramp-deceleration.

Ramp-acceleration

Ramp means speed with a gradient of increment or decrement. The motor controller can measure the unbalance present inside the drum while the motor rotates in a ramp speed. Figure 3.6 measured unbalance value is linearly dependent to the actual unbalance value. For all the unbalance weight position inside the drum the measured values are very similar and the difference is not significant. That is why measuring the current unbalance present inside the drum can also be effective if it were measured by looking at the ramp speed unbalance value especially when the drum rotates with speed acceleration.

Ramp-deceleration

The graph in Figure 3.7 depicts the data obtained from the ramp speed unbalance value while the drum rotates with deceleration. Measured unbalance values is plotted against

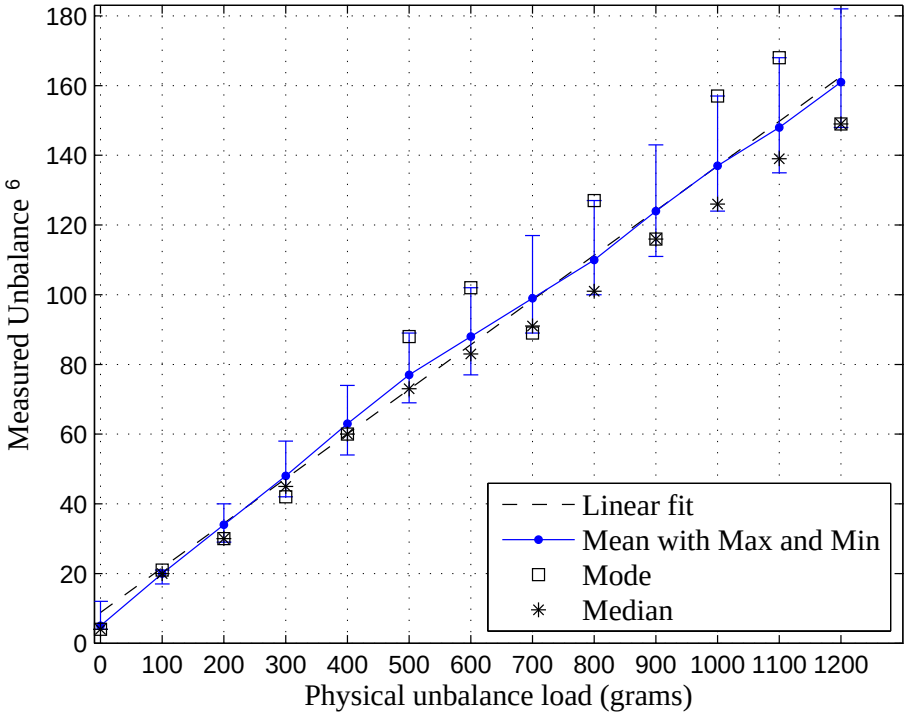

Figure 3.5: Unbalance values at constant speed vs. load at back position.

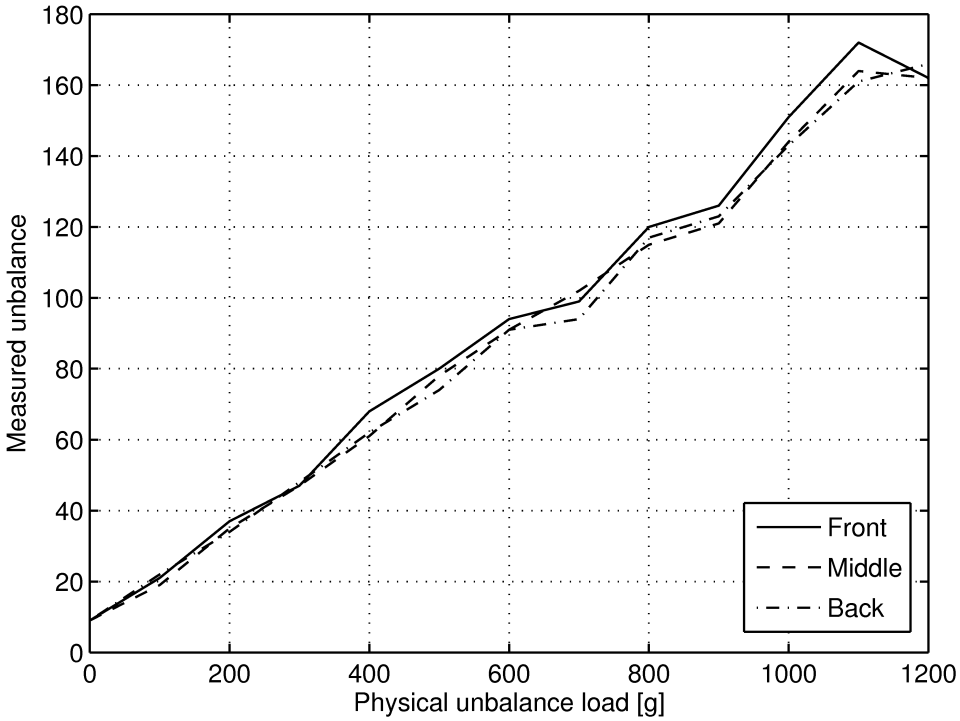

Figure 3.6: Ramp speed unbalance measurements for all load positions (acceleration).

Table 3.4: Unbalance measurements for ramp speed (acceleration).

Load	Unbal(front)	Unbal(mid)	Unbal(back)
0	9	9	9
100	21	19	22
200	37	35	34
300	47	47	48
400	68	61	62
500	80	78	74
600	94	91	91
700	99	102	94
800	120	115	117
900	126	121	123
1000	151	144	143
1100	172	164	161
1200	162	162	166

Table 3.5: Unbalance measurements for ramp speed (deceleration).

Load	Unbal(front)	Unbal(mid)	Unbal(back)
0	40	40	40
100	61	64	54
200	77	39	72
300	92	65	111
400	77	102	115
500	140	129	136
600	140	146	153
700	154	178	155
800	195	183	175
900	163	161	200
1000	182	203	184
1100	220	227	212
1200	254	222	225

the gradual increase of actual unbalance load for all the three positions (front, mid and back). In some cases measured unbalance value is not proportional to actual unbalance value. The Table 3.5 shows, in front position as the actual load increase from 300 grams to 400 grams, the measured value should also increase. Unfortunately, it decreased from 92 to 77 instead of increasing. Similar case is true for other load cases in both mid and back position also. Other than these two specific cases every time with the increase of actual load corresponding measured value increases. So the measured value is linearly dependent on the actual unbalance value. For all the three positions the motor controller detects with similar values of unbalance. This implies that weight position inside the drum does not influence the unbalance value.

Comparison of Acceleration and Deceleration

Figure 3.8 shows the basic difference for unbalance values obtained with two types of ramp speed. From this graph it is clear that for the same actual unbalance weight the motor controller measures two different values for acceleration and deceleration ramp. It

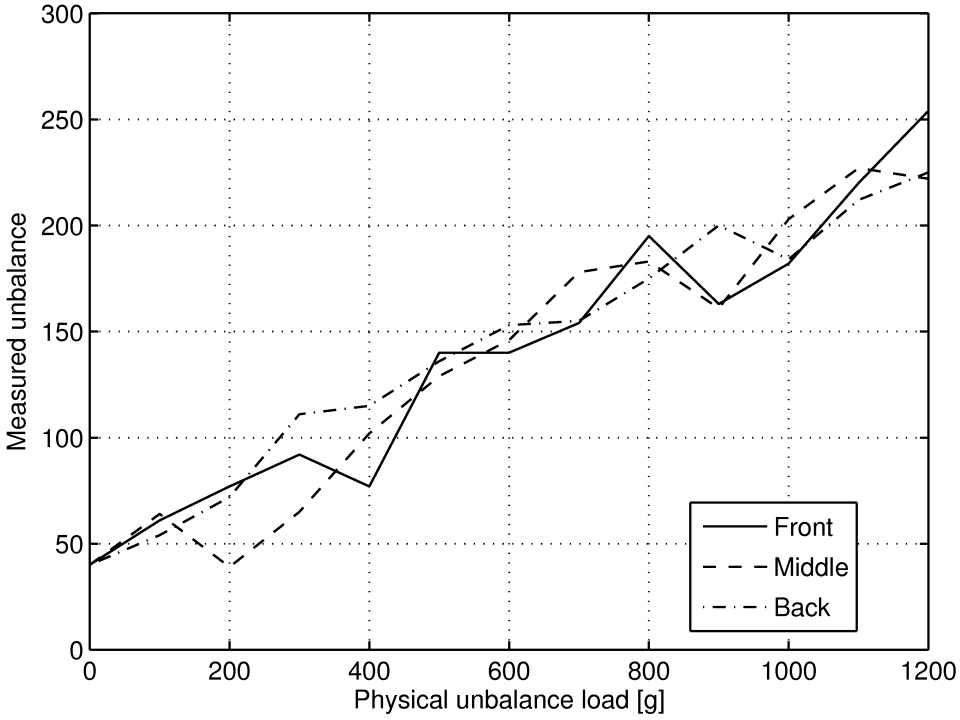

Figure 3.7: Ramp speed unbalance values for all load positions (deceleration).

is also interesting that a deceleration ramp always results in a higher amount of measured value compared to an acceleration ramp. Acceleration ramp speed measured values are linearly related in a better from than the deceleration speed ramp values.

3.2.3 General observation in data analysis

After plotting all those data in a graphical format and analyzing them, there was some common observation over constant speed unbalance value-

- Unbalance at constant speed shows a range instead of any fixed value like ramp speed unbalance.
- Constant speed unbalance with respect to one specific load shows different values with different frequency.
- Constant speed unbalance shows almost similar range of values with different placement of load.

Similarly there was also some common observation for ramp speed data analysis-

- Unbalance at ramp speed varies linearly with load amount
- Unbalance for high speed ramp is not dependent on the load position.

3.2.4 Data analysis conclusion

This analysis was one of the important items of the whole project. While selecting the unbalance value, the following conclusion played key role in value selection criteria.

- Unbalance measurement at constant speed is increased in an almost linear way as the physical unbalanced load is increased.

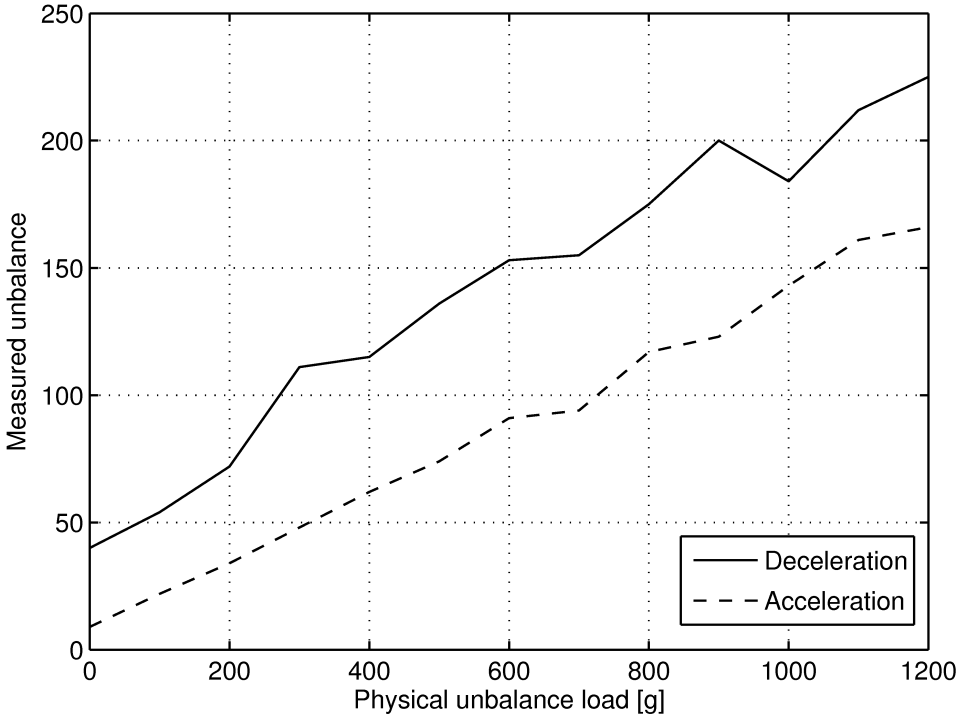

Figure 3.8: Ramp speed unbalance measurements comparison (acceleration vs. deceleration) in the back position.

- The tested load position does not affect significantly the unbalance values obtained from the motor.
- Unbalance measured at constant speed fluctuates less compared to ramp speed measurement.
- Less reliance should be put on the ramp speed as the values vary in a wide range for acceleration and deceleration.

With these conclusions it can be stated that unbalance measurements at constant speed are more reliable and they are the best choice as the main consideration in the performance evaluation function of the system. Also the fact that the unbalance measure is not affected with the load position within the drum makes the evaluation easier as the system does not need to know where the unbalance is placed, hence, avoiding the calculation or even the addition of extra hardware that can determine the placement of the unbalance. And finally the measured value obtained can be optimized no matter if the units are unknow or can not be directly related to a weight unit as it is proportional to the unbalance, the lower unbalance measure, the best system performance for the purpose of this project.

3.3 Evaluation of solution candidates

In the software setup the system requests the motor controller to get the unbalance value once after each individual has been completed.A lot of variables like speed, time, load weight, gravitational force etc. are connected to the resultant unbalance after executing any individual speed scheme. The same speed scheme with same cloths inside the drum will results different orientation each time it is executed. Every new run of a specific individual will have a new orientation of cloths load. So unbalance value will be change every time.

From the previous data analysis it was observed that at constant speed the unbalanced value gets a range of values instead of one specific value of concentrated load. For example

1000 grams of concentrated load placed at the mid of the drum may result in an unbalance range from 123 to 158.

On the other side one specific speed scheme may create different orientation of the clothes inside the drum. After each run a different orientation of load makes different unbalance. To minimize the effect of these two factors and get a suitable average for each individual five samples were collected. Thus, based on these five samples the mean unbalance value of each individual speed scheme has been calculated.

But most of the solution candidates (speed schemes) were getting very high fluctuation in the unbalance value. Due to the small number of samples it was troublesome to set a good judgment over the solution candidates performance. High fluctuation of unbalance value also brought attention into getting more stable solution candidates. From the beginning best candidate selection criteria was based on the lowest average of constant speed unbalance value over five observations. Later looking on the individuals highly fluctuating unbalance value it was also needed to take care of standard deviation of those five samples.

An individual can only be judged as a good candidate when it gets lower unbalance values for several independent runs and also get low variation among those unbalance values. The obvious solution is weighting to be able to use averages and standard deviation of the constant speed unbalance values simultaneously. The question is how much weighting should be given on what factor.

$$\text{Average, } \bar{X} = \frac{(X_1 + X_2 + X_3 + X_4 + X_5)}{5} \tag{3.1}$$

$$\text{Standard deviation, S} = \sqrt{\frac{1}{5}\sum_{i=1}^{5}(X_i - \bar{X})^2} \tag{3.2}$$

$$\text{Individual fitness, F} = \frac{1}{X * \frac{46}{(46+26)} + S * \frac{26}{(46+26)}} \tag{3.3}$$

The ASKO or production speed scheme[7] derived from experience and judgment shows highly stable results with low standard deviation while it is run for a high number of times. A preliminary pilot study of 30 samples with the production speed scheme ended with an average (eqn. 3.1) of $\bar{X}_{ASKO30} = 46$ as an unbalance value with a standard deviation (eqn. 3.2) of $S_{ASKO30} = 26$. So based on this pilot study result all the individuals were evaluated based on the relationship of the resultant mean and standard deviation. The relationship that was used is the sum of the mean and the standard deviation and a weighing factor was determined according to the contribution of each one to the total. The weighting factor for the candidates averages and for standard deviation (for five samples) was determined by the formula stated in equation 3.3.

Best individual was selected based on the highest value attained by this formula which take care both of the averages and the standard deviation

3.4 Experiments plan

The complete experiment has been designed in four comprehensive steps for finding an improve speed scheme. The first step is independent and its purpose is evaluation. The second and third steps are verification and validation. And the final part is decision taking by comparison.

[7]The ASKO or production scheme is the speed scheme that is currently used for production in the ASKO washing machines during the load distribution stage which is part of the drying cycle of the washing machines.

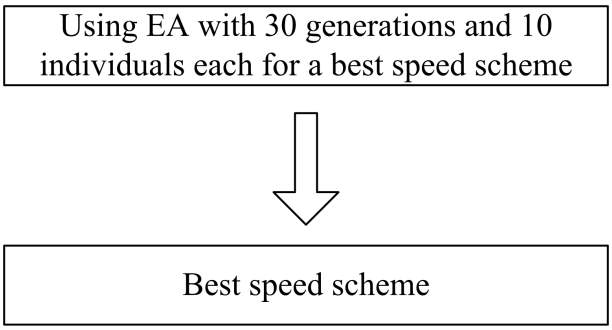

Figure 3.9: Finding the best individual steps.

The first step is shown on figure 3.9. Here the evolutionary algorithm was used to find the best speed scheme. The algorithm was evaluated generation after generation with 10 different individuals (speed schemes) in each generation. The whole study was continued up to 30 generations until the best speed scheme was selected. In total 300 speed schemes were evaluated during the process.

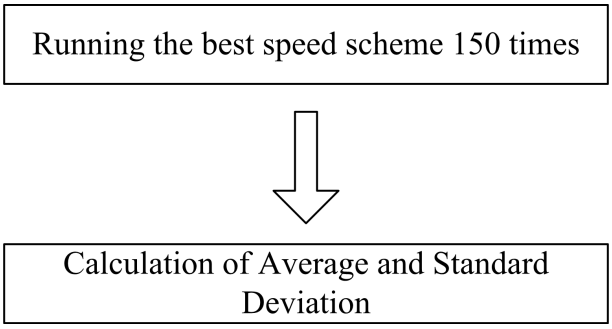

Figure 3.10: Sampling 150 times with the selected speed scheme.

The figure 3.10 and 3.11 show the second and third steps where both of the speed schemes ASKO speed scheme currently installed in the machine and evolutionary algorithm based best individual based on 300 samples both were run for 150 times to determine the average and standard deviation of the unbalance value. 150 samples are not a big size samples. Considering 20 weeks master's thesis with tight time schedule, it was accepted as sufficient.

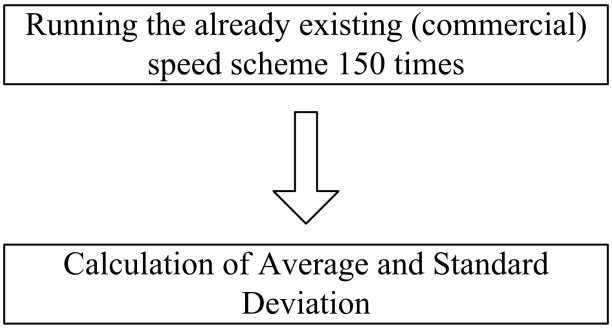

Figure 3.11: Sampling 150 times with the current speed scheme.

The final step is the comparison of the both schemes based on their achieved average (eqn. 3.1) value and standard deviation (eqn. 3.2) value obtained from 150 samples.

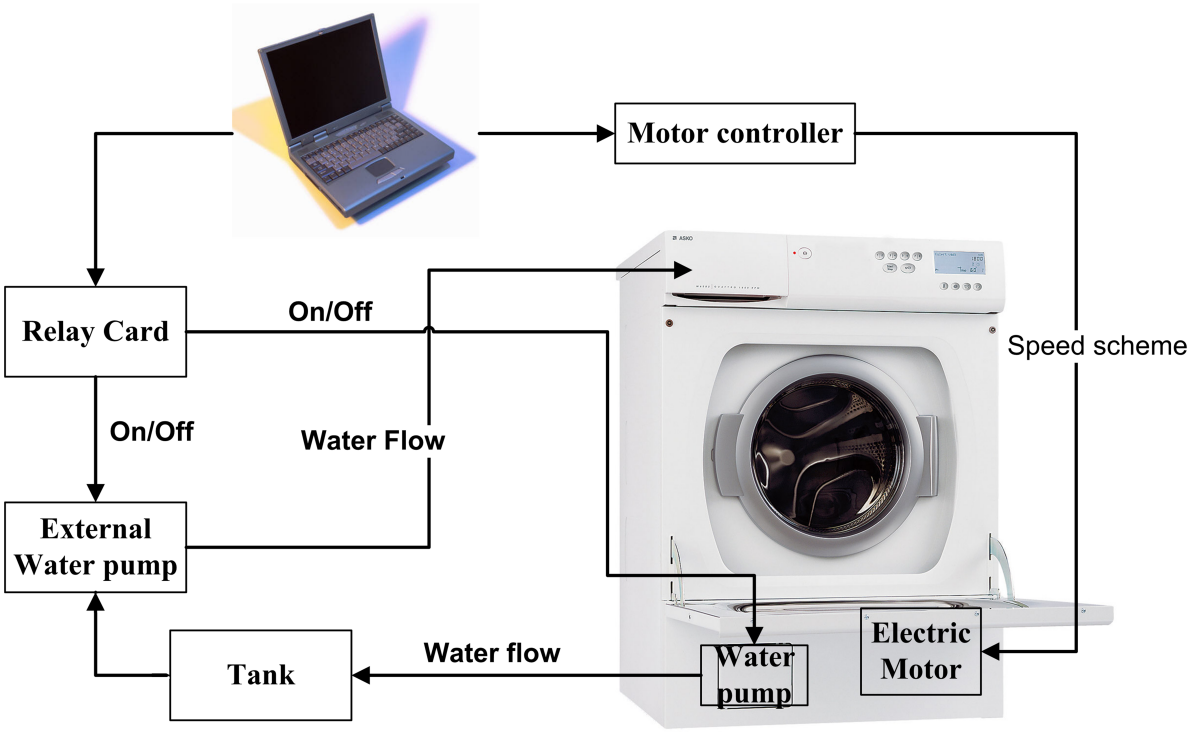

Figure 3.12: Experimental setup

3.5 Experimental setup

The whole experimental setup is controlled and runfrom one single interface on a computer. The complete interface has been shown in Figure **??**. This interface is developed with two main objectives

1. Control hardware related to the experiments.
2. Run the EA to find suitable speed scheme.

The complete experimental setup has been shown in figure 3.12. The experimental setup is mainly composed of two pumps controlled by a relay card (Velleman K8056) and washing machine motor controlled by its built in controller. A closed water circuit is maintained to by hose pipe to get the cloths soaked for each speed scheme. Evaluation of each individual speed scheme through this experimental setup is preformed by the following sequences of operation.

1. External water pump remains started by relay card to pump water into the drum from the tank.
2. Washing machine drum is rotated by the motor controller ensuring the proper mixing of water with cloth.
3. Relay opens water pump of the machine to drain out all the water from the washing machine and back into tank until scheme finishes.
4. Interface software sends the motor controller a speed scheme to be executed.
5. Motor controller sends the unbalance output to the interface to store the value for further evaluation.

   This whole cycle continues for fives times for each speed scheme.

3.6 Time frame for the experiment

During experiment planning the time consumption was one of the important factors in determining the number of speed scheme evaluation that was going to be done. Time taken to pump water into the washing machine, running the machine drum to soak the clothes, together with the water drainage time and the speed scheme evaluation time constitute the whole cycle time. The cycle time also includes the time taken by the motor controller to trigger the unbalance value during the weighing cycle. Cycle time for each individual test,t_{cyc} and is calculated as:

$$t_{cyc} = t_{win} + t_{w\_out} + t_{sc} + t_{wgh} \tag{3.4}$$

where t_{win} is the time spent in pumping in water and soaking the clothes, t_{w_out} is the time spent in pumping out water, t_{sc} is the time spent in running each speed scheme once and t_{wgh} is the weighing triggering time. Considering a cycle of pumping in and soaking clothes takes 35 seconds, pumping out water of washing machine takes 25 seconds, running 1 individual 1 time takes 80 seconds as the worst case scenario and the measurement speed scheme takes 63 seconds, the cycle time for each individual is t_{cyc} is equivalent to 203 seconds as worst case sceneario.

This time frame calculation was aimed to find out the total time required for the whole experiment. As each generation consists of several speed schemes, the number of the speed schemes tested is equal to the total number of generation times population size per generation. And the total time was calculated by multiplying the cycle time with the total number of executed speed schemes. The fact of executing five times each individual makes the process five times longer. If the sampling frequency is s for each individual then g number of generation of p population size will take total time

$$t_{total} = t_{cyc} * s * g * p \tag{3.5}$$

The sampling frequency s is equivalent to 5, the number of generations g is equivalent to 30 and the population size p is equivalent to 10, the total time is T_{total} is equivalent to 101500 seconds or about 28.2 hours. Running the best scheme 150 times takes 7.1 hours and running the ASKO scheme 150 times takes another 7.6 hours. Although theoretically running the experiment takes around 29 hours (around 43 including the execution of the best speed scheme obtained by the optimization algorithm and the execution of the ASKO scheme), in the reality there is a lot of problems (*i.e.* software or hardware issues) that can affect the execution of the experiment and can add overhead time to the total time.

3.7 Load weight

A washing machine causes vibration while rotating with unbalance load inside the washing drum. Higher amount of laundry load gets easily well distributed inside the drum and results in less unbalance. Unfortunately, in case of less amount of cloth it is hard to get a proper distribution which results high unbalance and causes vibration. So as the washing machine has capacity for loads up to 7 kg, it has been decided to take 2 kg of load weight for performing an EA evaluation scheme.

A bundle of 20 pieces cotton cloths which each piece weight was 100 grams of which the accumulated weight was 2000 grams (2 kg) was used to represent the load.

3.8 Water amount calculation

According to ASKO washing machine manufacturer for a load of 2 kg cotton, 9.1 liter of water should be supplied for proper washing. While the experiment was running it was

necessary to supply the same amount of water to ensue same conditions for all competing speed schemes. In reality, after each cycle of operation a small amount of water remains in the pump unit and also in the drain hose. So, following steps were executed so that the water amount during evaluation of each speed scheme9.1 liter became exactly.

1. Provided the washing machine with 9.1 liter of water
2. Starting washing machine pump so that all water is drained out of the washing machine.
3. Collect and measure the water to check how much water is trapped inside the pump and hoses.
4. Replenish with water that has been trapped inside the pump.
5. Iterate the whole process until same amount of water comes out each time.
6. Follow the same procedure for the external water pump also.

During experiment there were some variation in the washing machine pump and external water pump. The Table 3.6 and Table 3.7 shows experiments based water calculation.

Table 3.6: Water correction for washing machine pump.

Step	Addition of Water (liter)	Pumped in(liter)	Pumped out(liter)
1	9.1	9.1	8.6
2	0.5	9.1	9.1
3	0	9.1	9.1

Table 3.7: water correction for external water pump

Step	Addition of Water (liter)	Pumped in(liter)	Pumped out(liter)
1	9.1	9.1	8.3
2	0.8	9.1	9.1
3	0	9.1	9.1

4 Proposed algorithm

4.1 Evolutionary algorithm

Optimization algorithms inspired in biological phenomena are stochastic methods that try to emulate or follow the behavior of natural biological processes. One of these processes is the evolution which is, by its definition, one of the greatest optimization processes in nature. Evolutionary Algorithms (EAs) are part of these stochastic optimization algorithms and, as their name imply, take the evolution as the main concept in their construction. The principle is that the algorithm creates a population of solutions and breeds them to generate a new solutions population and iterating this process until a satisfactory individual is generated. The breed stage refers to the crossover and mutation of individuals, combining

their genetic code and generating new individuals who will be evaluated to determine the best each generation.

EAs are search and optimization algorithm that are inspired by biological adaptation mechanism of natural (or darwinian) evolution. EAs are not the absolute models of biological evolution, rather they are merely the imitation of adaption phenomena captured into a computational model for optimization. In case of any specific problem, EAs implementation concerns first about the solution candidate or individual representation which is how the individual will be structured (quantity, size and content of the genes). Population is the central concept of EAs which means a group of individuals or solution candidates. Instead of using a single candidate solution, a candidate solution population ensures parallel search through the solution space.

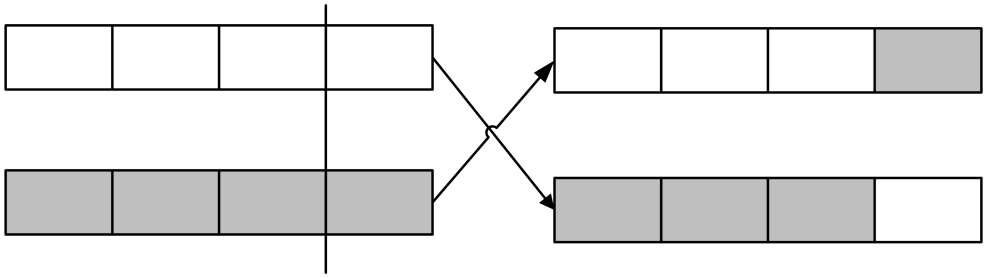

Figure 4.1: The standard single point crossover procedure. After a crossover point has been set, one part of the gene's individual are combined with the other part of the second gene's individual

After determining the initial population a *evaluation* stage is carried out with the help of the fitness function (or the objective function). Evaluation aims to assign a fitness value to each of the individuals for distinguishing one good candidate from an inferior one. The next step is the *selection* process confirms the progenitors for the next generation solution population. In *generational replacement*, all the individuals of the first generation are discarded and second generation comes into consideration. In this stage, an *elitism* step may be added and, it consists in the selection of the best individual[8] which will be pass unchanged to the new generation. In this way good candidates can survive the next stages. To increase the diversity of the candidate solution population a variation operator is used. In EAs such type of variation operator is known as *mutation*. Mutation takes the responsibility of exploring the new aspects of the problem avoiding the algorithm to be stuck in local optima. On the other side, conservation operators are to minimize the population's diversity to consolidate already explored search space. *Crossover* serves as conservation operator. One point, two point, or n-point crossover among two different individual (parents) gives new set of individuals as reproduced offspring. Figure 4.1 shows one point crossover process in the case of binary EAs, but for more realistic representations like real-valued EAs, another process is used (Although it helps to show the basic principal of the crossover). After iterating these steps a new generation of solution candidates is created.

Now two sets of population are available. This second generation follows the same procedure of evaluation that has been done for the first generation. This process of selection, mutation and crossover continues from generation to generation until a satisfactory solution candidate has been found. Figure 4.2 shows a schematic diagram for EA's flow.

As described before, EA is a stochastic optimization method that consist of a set of steps that try to emulate the basic principle of Evolution and which fundamental steps are

[8]It is not restricted to only one individual, can be more depending in the population size or if the problem requires it.

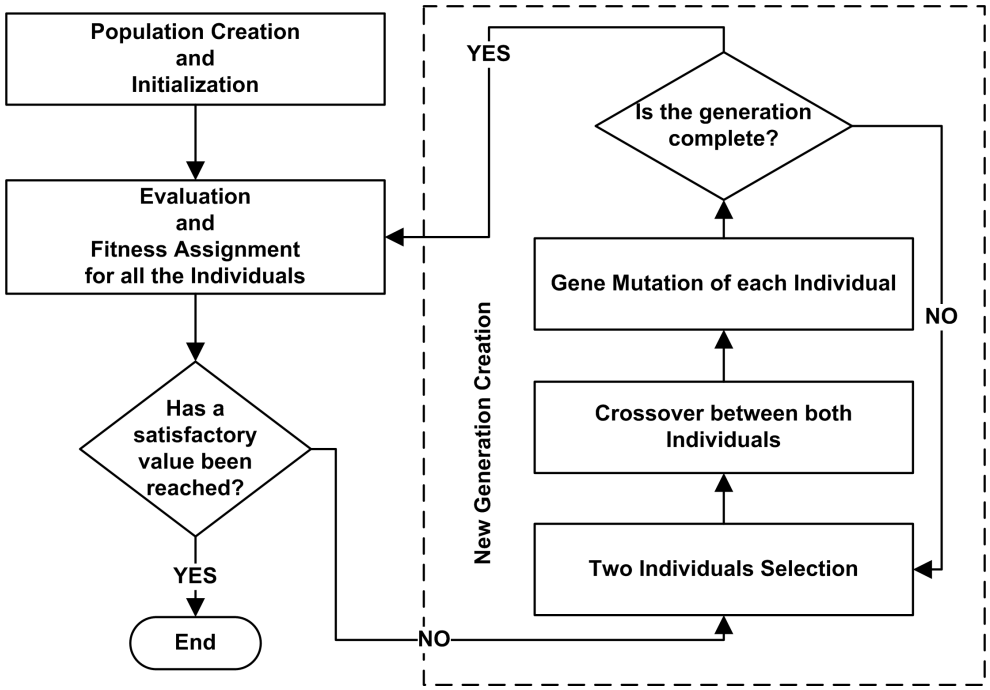

Figure 4.2: A flow chart of a generic EA describing its main stages (Initialization, Evaluation, Selection, Crossover and Mutation).

described in Figure 4.2. The basic principal is already stated but, the interest arises on how to apply each step to the project. For each step there are many kinds of options to choose from for its implementation but, for the current project the selected methods shall be described as follows.

4.2 Structure of the individuals

As stated before the definition of the structure of the individual or chromosome is very important as it will represent the variables that can be controlled in order to optimize the system. As mentioned in section 2.1, the individual shall have four genes and, each

	1st Allele	2nd Allele	3rd Allele	4th Allele
	DIRECTION	**SPEED (rpm)**	**GRADIENT (rpm/s)***	**TIME (ms)**
1st Gene	**Clockwise - Counterclockwise**	**0-400**	**0-80**	**0-20000**
2nd Gene	**Clockwise - Counterclockwise**	**0-400**	**0-80**	**0-20000**
3rd Gene	**Clockwise - Counterclockwise**	**0-400**	**0-80**	**0-20000**
4th Gene	**Clockwise - Counterclockwise**	**0-400**	**0-80**	**0-20000**

Figure 4.3: Structure of the individual or chromosome. Description and range used for each gene and allele in the EA. **Unit used in the software specification of the motor controller.*

gene will contain four alleles[9] which are: the direction, speed, acceleration and time. In Figure 4.3 the structure is shown, as well as the allele's range allowed for this project. Such limits should not be exceeded in order to avoid damage in the system as the mutation can generate really abrupt changes and high speed. Also the time shall not exceed 20 seconds each gene[10], as the new distribution scheme shall not add a significant overhead to the execution time of the drying cycle.

4.3 Initialization

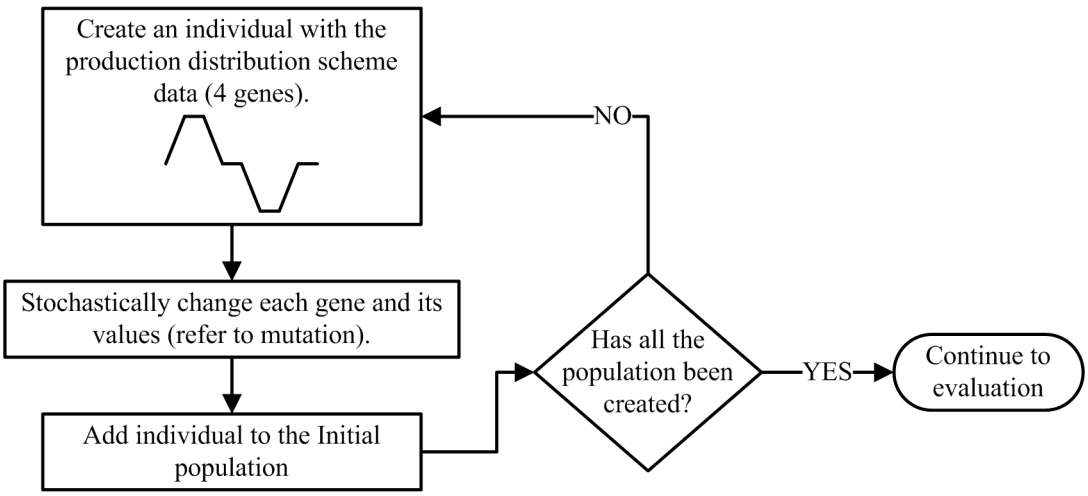

Figure 4.4: Flow chart of the population initialization stage.

As shown in Figure 4.4, the initialization of the individuals is made taking into consideration the current production distribution scheme, since it is known that it is already a good candidate, as a seed. The EA will create a copy of the seed scheme considering it as an individual in which a randomization is made to vary its genes and form a newly individual that can lead to an optimal solution in the long run. This process is repeated until all the individuals in the population are created.

4.4 Evaluation

The next step is the evaluation of each individual. As stated before the evaluation function, the most crucial factor in EAs, will be made using a hardware-in-the-loop set up. Right before each individual is executed, the clothes need to be soaked with water in order to maintain an initial condition for each individual or solution candidate. Each individual of the population will be processed, and the commands necessary will be sent, in order for the washing machine to execute them all (see section 2.2). After the execution of each individual the algorithm also executes the measurement scheme which is a scheme made specially to measure the unbalance at certain points, making the measurement standard for all the individuals. After all the individuals were evaluated and all the information was acquired, the program will calculate the fitness using the statistical method described in section 3.3.

[9] In genetics, Allele is a DNA sequence. In this case it represents a parameter or variable that can be manipulated in order to optimize the system.

[10] Worst case scenario is 80 seconds for the whole individual in comparison with the production scheme which last 30 seconds.

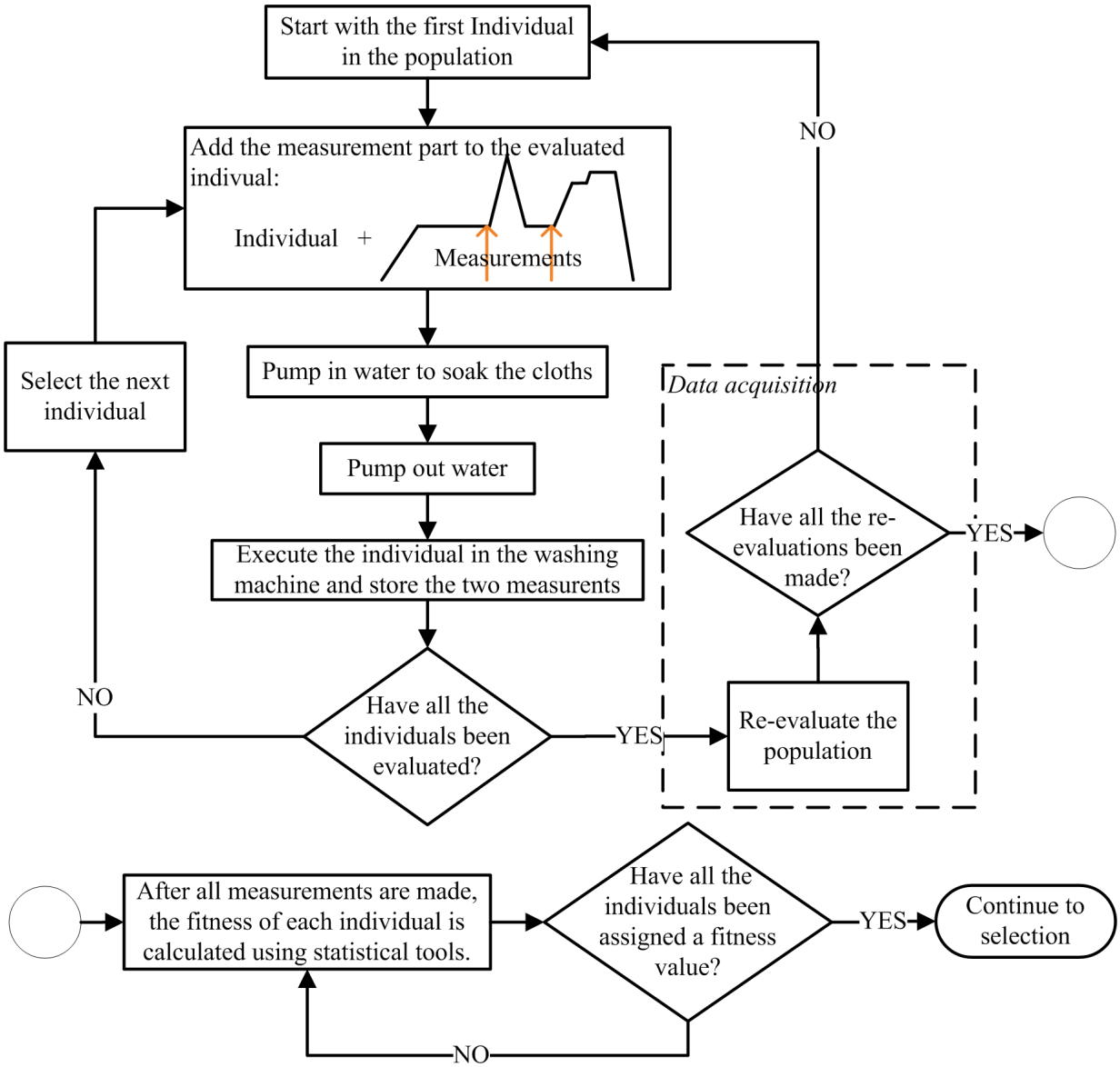

Figure 4.5: Flow chart of the evaluation stage.

4.5 Selection

After evaluating all individuals, the next steps are considered as the breeding, where the individuals of the current population breed in order to generate a new population of solution candidates. As can be seen in Figure 4.6, this part of the algorithm starts with a phase called elitism which selects the best individual of the current generation and adds it to the new generation without any change. Elitism helps to ensure that the new generation has at least one good candidate, in other words, that the best solution candidate has an equal or better fitness value than the previous generation. After elitism, two individuals are randomly selected and they will contend each other in order to be selected for reproduction. The criteria chosen to determine how an individual can be picked is the Boltzman selection process which starts by calculating the Boltzman coefficient as described in equation 4.1:

$$b(F_j, F_k) = \frac{1}{1 + e^{\frac{1}{T}\left(\frac{1}{F_j} + \frac{1}{F_k}\right)}} \tag{4.1}$$

Where F_j and F_k are the fitness values of the two randomly chosen individuals and T is a parameter that will determine the criteria to select the resulting individual. For small

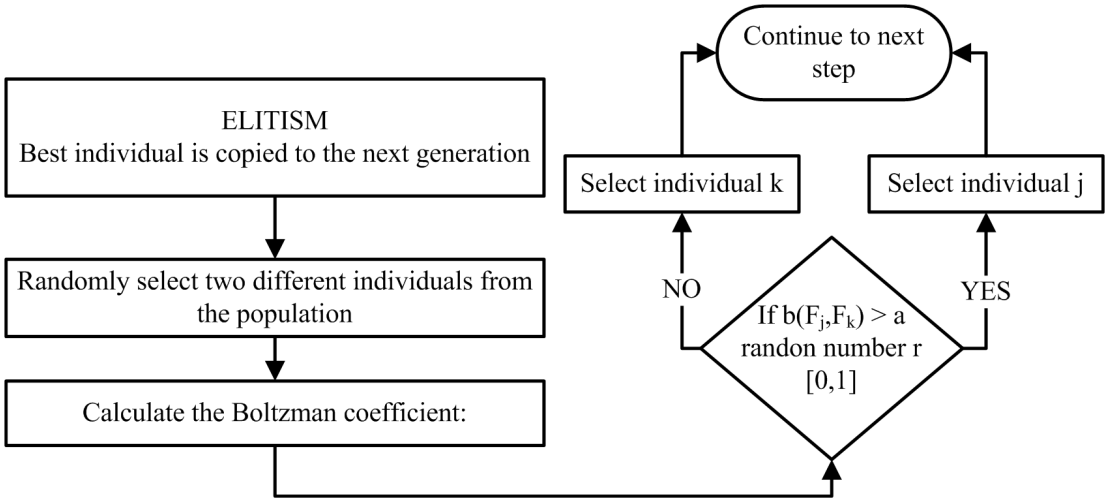

Figure 4.6: Flow chart of the population selection stage.

values of T ($T \to 0$) the resulting individual will be the one with the higher fitness value and for large values ($T \gg 0$) there would be the same probability that any individual is chosen. For the current algorithm the value T was chosen to be high in order to select the individual with the higher fitness value at any generation. After obtaining the Boltzman value $b(F_j, F_k)$ a random number r, with range [0,1], is generated and after the following conditions 4.2 are applied, one of the individuals will be successfully selected.

$$\text{Selected Individual} = b(F_j, F_k) \begin{cases} j & \text{if } r > b(F_j, F_k), \\ k & \text{otherwise.} \end{cases} \tag{4.2}$$

The Boltzman selection was initially chosen because the parameter T can be changed from generation to generation and, by doing that, the criteria can be changed in time. This means that in the first generations the parameter T will be higher in order to select any of both contenders with the same probability so that at the beginning there will be an exploration of individuals. Then for the later generations the parameter T will be low in order to select the best individual from both contenders and, hence, exploit the best candidates on each selection. Unfortunately late in the project , it was discovered that T was assigned a constant low value and did not vary with the generation intex. There was no time to fix the issue and hence the selection effectively became selection of the best individual of the individuals J and K according to equation 3.1 and 3.2

4.6 Crossover

The next step is the crossover, which consist of the combination of the two individuals previously selected using the previously described selection process. As both individuals will combine their genes, the resulting individuals will be a variation of each other, and hence, they will spread their parent's genes trough the next generations. This allows genes to spread out through population and generations but may cause the algorithm to be stuck in a local optima, and for this the crossover shall be applied with certain crossover probability p_c. This means that not all the individuals will be subject to the crossover. Crossover in this project is carried out by randomly selecting a crossover point as seen in Figure 4.7. After selecting the crossover point both individuals will exchange the genes from this point and generate new individuals with mixed genes.

In order to avoid conflicts when a new individual is generated from the crossover, the crossover points were selected in such that they are placed only in the genes limits so that

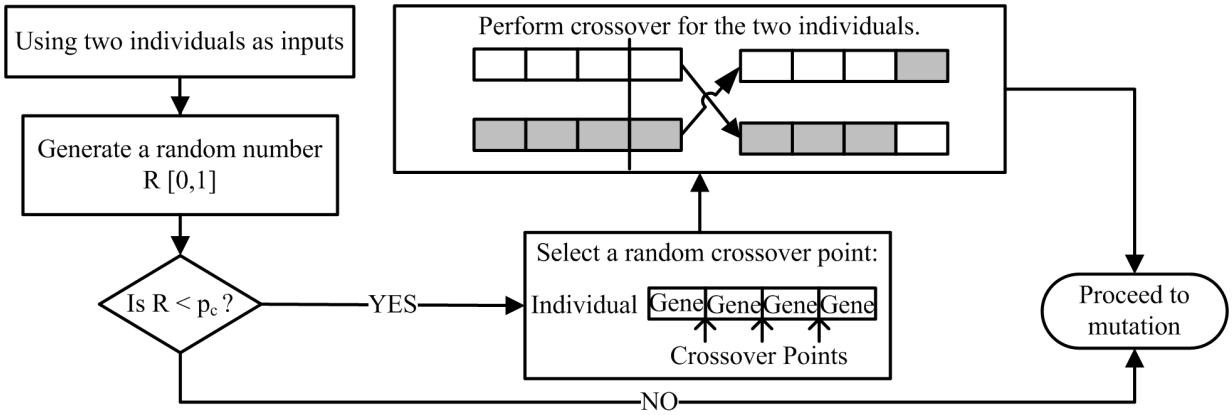

Figure 4.7: Flow chart of the population crossover stage.

the resulting individual will be genetically correct. There can be physical limitations from the hardware that may cause the new individual to fail during the execution but, they are being taking care of take place since the formation and also during mutation stages.

4.7 Mutation

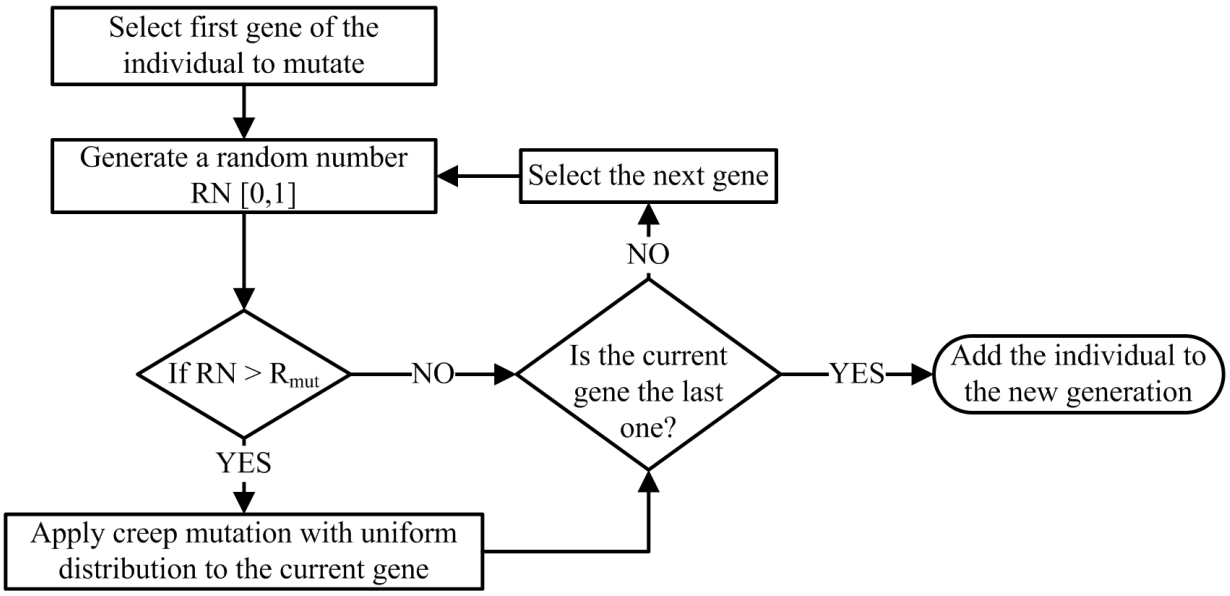

Figure 4.8: Flow chart of the population mutation stage.

The final step is the Mutation, which is the exploration operator, and its main objective is to extend the search space of the solutions candidates. By using the mutation, the algorithm will avoid to get trapped at local optima. As observed in Figure 4.8, the first step is to select the first gene of the individual, then generate a random number in the range [0,1]. If it is higher than the mutation probability p_{mut} [0,1] then the creep mutation is applied. The p_{mut} is established in order to prevent all the population to vary far away from its original values and as a result eliminate potentially good individuals.

In order to perform the creep mutation, a creep rate (C_r) value shall be established; a value which will determine how far the gene will mutate. In equation 4.3, the new value X' is the result of setting a random number [0,1] and then multiplying this number by a default range established for each parameter that conforms the genes, and the result will be the C_r. Figure 4.9 shows the uniform distribution graph used in the algorithm that

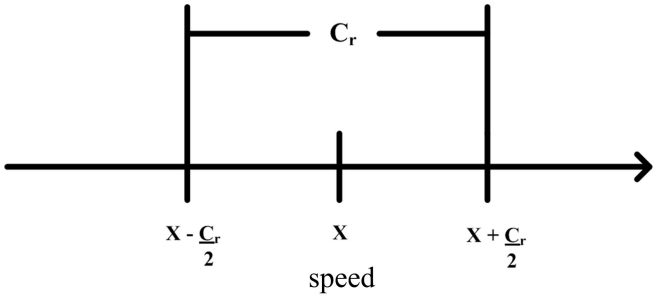

Figure 4.9: Uniform distribution graph applied in creep mutation. The resulting X' can have a value between $\left[\left(X - \frac{C_r}{2}\right), \left(X + \frac{C_r}{2}\right)\right]$ according to formula 4.3.

explains how the value can vary within the established range $\left[\left(X - \frac{C_r}{2}\right), \left(X + \frac{C_r}{2}\right)\right]$. If the result of using the calculated C_r in equation 4.3 exceeds the physical low or high limit of each parameter (*i.e.* for the speed, the limit is 400 rpm), it will wrap around the value and turn back[11]. Each one of the parameters within the gene has its own creep rate, for exameple, for the speed the creep rate was set to be 200 rpm so that a speed value can mutate is ± 100 rpm. And so, there is also one creep rate for gradient and time. The Direction does not have a creep rate since it just have two possible values. If mutation is applied to the direction, it will just switch the direction.

$$X' = X - \frac{C_r}{2} + C_r r, \tag{4.3}$$

After mutating the individuals each one will form part of the new generation. This breeding phase will be performed as many times as necessary in order to form all the individuals required in the population.

5 Results

As described before the set up for the experiment was made for the case of 2 kilograms of laundry. When using EA, on a pure analytical fitness function an evaluated individual always receives the same fitness value making the optimization to be straight forward, in other words, if there is a new individual with better performance than the current best individual, automatically is selected as the new best because its fitness value will not change. In this project, there is the possibility that the evaluated individual can perform better or worse than in previous generations due to the stochastic nature of the initial conditions of each evaluation.

A total of 30 generations were run, and each new individual generated did not take into consideration the previous unbalance values from their parents or if the individual was the best of a previous generation, its previous data was deleted in order to be added to the new generation. As a result the individuals in later generations, being new breeds or best from a previous generation, may not have the same unbalance values as they had before.

Figure 5.1 shows the fitness value assigned to the best individual for each generation but, the figure does not clearly show how the optimization is progressing as the values are scattered due to the reasons just mentioned above. For that purpose a linear fit line was obtained by calculating the coefficients of the first degree polynomial function in a least

[11] *I.e.* if limit is 20 and the value is 24, the software will wrap around the value and the result will be 16.

square sense, and the result is the linear fit line with negative slope, which means, that the fitness value is decreasing generation by generation.

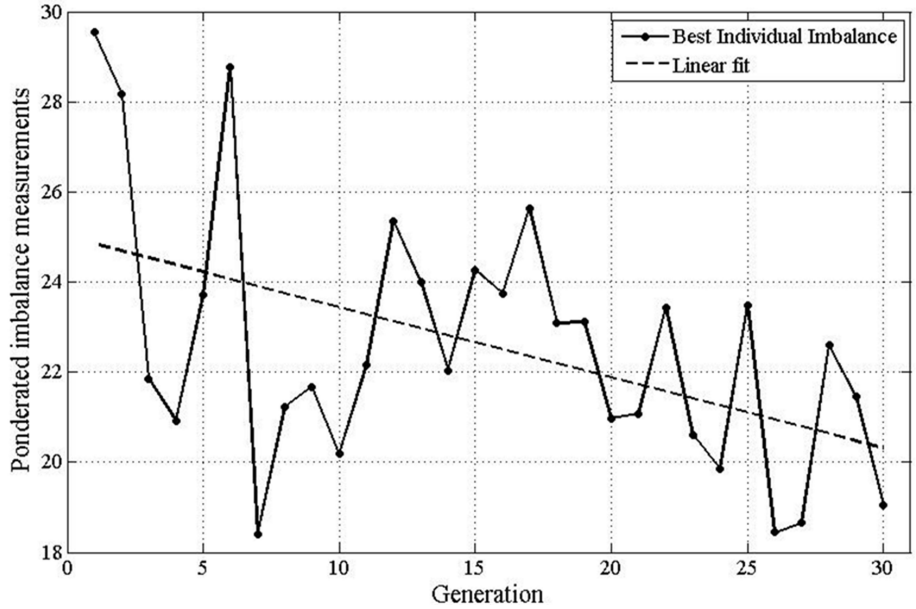

Figure 5.1: Improvement based on fitness function on each generation.

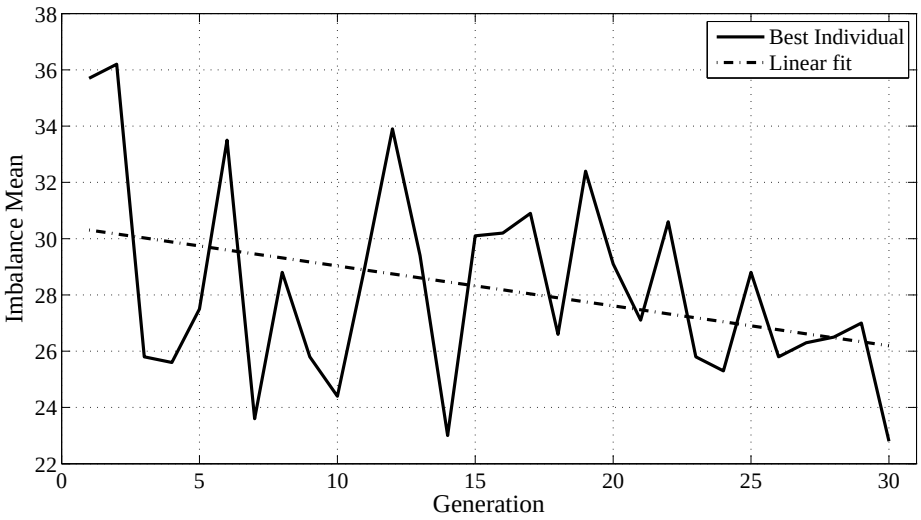

Figure 5.2: Best performance on each generation. Performance calculated using only the mean of the unbalance measurements.

The fitness function is included with standard deviation. So individuals were also evaluated based on their standard deviation. As a result there was improvement in their standard deviation from generation to next generation. The entire best individual's standard deviation of 30 generations is plotted and a trend line is calculated using the same technique as before, as can be seen in Fig. 5.3, a negative slope is obtained which means that the trend is to decrease the standard deviation as the generations are executed.

After obtaining all 30 generations a final speed scheme was obtained which had a low unbalance mean value and a low standard deviation. The parameters for this individual can be seen in the table 5.1 along with the currently used ASKO speed scheme. As can be seen, both schemes contains similarities as both change from one direction to another

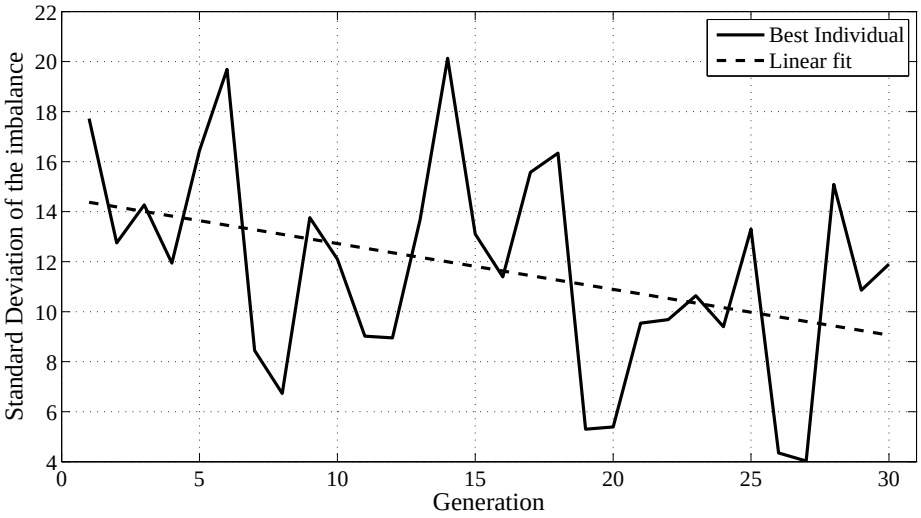

Figure 5.3: Standard deviation of the best performance on each generation.

but the new proposed scheme requires higher speed than its predecessor. Figure 5.4 shows their graphical look side by side.

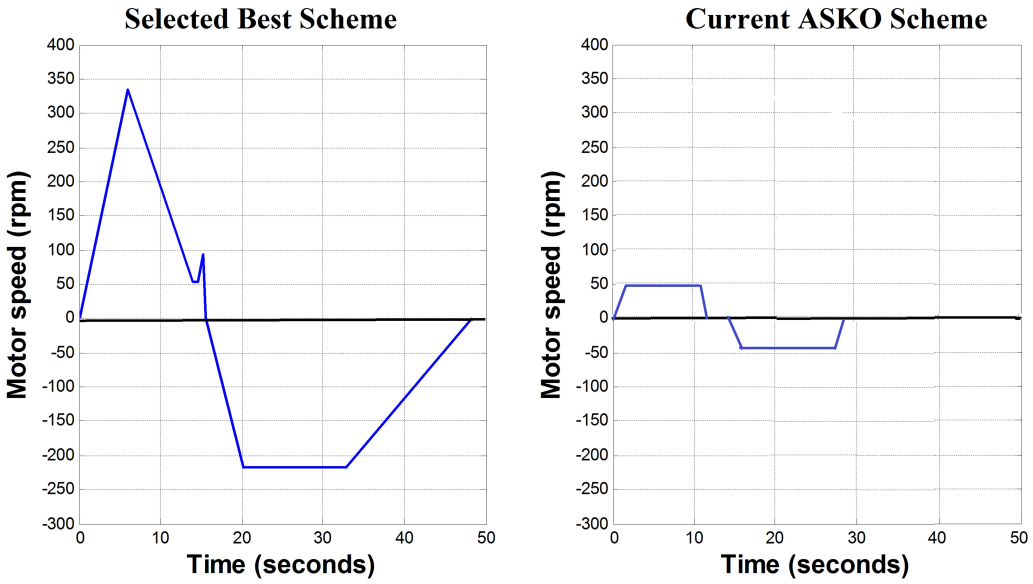

Figure 5.4: Selected best speed scheme from 30 generation at left and currently available ASKO speed scheme at right.

Figure 5.5 shows the speed scheme obtained from the EA, in the left side is the speed scheme as it should be executed if the timing is correct, but as there are timing constrains for the transmission of the messages as well as physical constrains[12], the precise execution of the scheme can not be reached. For that matter, in the right side, the executed speed scheme is depicted using the data obtained from the washing machine.

Due to this difference between the real and the theoretical scheme, some adjustments can be made in the data in order to follow the real scheme. In table 5.2, the adjusted scheme values are shown.

[12]Physical constrains are present because the motor may not be able to follow the gradient requested as it does not have the enough power to rotate the heavy laundry load.

Table 5.1: Comparison between the selected best speed scheme from 30 generation and the current ASKO speed scheme.

	Direction	Speed	Gradient	Time (seconds)
Selected Best scheme	Clockwise	371	56	5.9
	Clockwise	54	35	14.6
	Counterclockwise	340	60	0.6
	Counterclockwise	218	47	17.2
	total time			38.1
	Direction	Speed	Gradient	Time (seconds)
Current ASKO scheme	Clockwise	49	28	12
	Clockwise	0	0	3
	Counterclockwise	49	28	12
	Counterclockwise	0	0	3
	total time			30

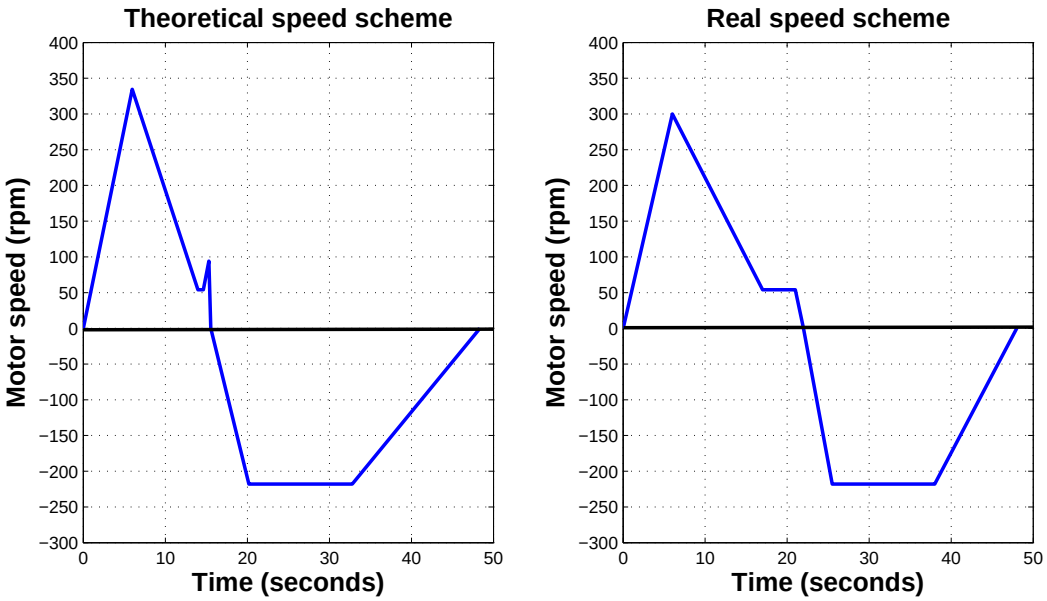

Figure 5.5: Comparison between the theoretical scheme obtained by the program and its execution in the washing machine.

Finally after the best speed scheme obtained from 30 generation was tested 150 times. The currently available ASKO speed scheme was also tested 150 times for making a valid comparison. By calculating the average and standard deviation of all the measurements of each experiment, a comparison was made. The mean unbalance value obtained for the proposed scheme and currently ASKO scheme can be compared in table 5.3.

This demonstrates that the new scheme has a unbalance mean value of 12% less than the previous one and 9.8% less standard deviation.

Table 5.2: Proposed scheme. It was adjusted to follow the real data obtained from the washing machine.

	Direction	Speed	Gradient	Time (seconds)
Proposed scheme	Clockwise	300	50	6
	Clockwise	54	22.36	15
	Counterclockwise	218	62.28	17.2
	Counterclockwise	0	14	15.5

Table 5.3: Comparison in the performance of the proposed distribution scheme and the current distribution scheme.

	Unbalance Mean (Unavailable units)	Standard deviation	Time consumption (seconds)
Actual Scheme	45.8	25.4	30
Proposed Scheme	40.3	22.9	48

6 Conclusions and recomendations

As the results from the experiments the selected best speed scheme performs better than the ASKO speed scheme used in washing machines to distribute the laundry load. This scheme was tested independently for a total of 150 times, it outperformed the ASKO scheme by 12% less unbalance. The standard deviation calculated from this 150 measurements was 9.9% less than the ASKO scheme showing a better performance in different kind initial conditions. Also we can state that the total amount of time required for the new scheme is around 48 seconds, only 18 seconds more than the ASKO scheme, which does not add significant overhead to the total time of the drying cycle. This increment in time can be optimized also if it is included in the fitness function, which means that a new scheme can be found that is executed faster.

At the beginning the fitness function was designed to consider only the mean of the unbalance values received from the motor. Those experimental outcome are summaries in the table 6.1. The values obtained were not as good as expected and as the standard deviation was not being considered it fluctuates quite a lot from generation to generation. After 6 generations the experiment was interrupted looking at those disappointed outcomes. Fitness function was changed by including standard deviation (equation 3.3) which seems more promising.

The overall objective of the project is to get an optimized speed scheme which will result less unbalance. It is also very important that the speed scheme should be stable ensuring similar results at every execution. It was an important decision to set the weightage of the averages and standard deviation for the fitness function. The current ASKO speed scheme is good scheme with minimized average and standard deviation. So, the weightage for averages and standard deviation in fitness function were set based on 30 samples from ASKO scheme.

If the EA should be run more generations, it is likely that an even better scheme could be found. The evaluation function that determines the performance of each candidate scheme, and thus its fitness value, is not a mathematical model but a real time evaluation[13] of the scheme in the washing machine. This condition makes possible that a candidate scheme may result better or worse if it is executed in more than one generation since its previous fitness data is erased to generate the new individual.

[13]With stochastic initial conditions for each evaluation.

Table 6.1: Performance of the best individual of each evaluated generation using only average to calculate the fitness value.

Generation	Unbalance Mean (unknown units)	Standard Deviation
1	32.9	37
2	33.6	31
3	31.8	15.5
4	44.7	28.9
5	34.5	39.3
6	39.9	33.5

An improvement in this area could be to keep track of the previous fitness values in order to ensure that if an individual got the best fitness value in a given generation, for the next generation it will part from that value and remain still a good candidate. New data will be added to the previous fitness data and will give more accurate values. In this project, a proposed individual consisting of four genes with a fixed length was used. This was made in order to ensure coherence during crossover and mutation, but with some considerations and limitations, the evolutionary algorithm may be able to change the size of the genes, and thus, expanding the search space of the algorithm.

The criteria for ending the EA was a certain number of generations instead of using a threshold value or desired fitness value that the individuals needed to reach in order consider them a satisfactory solution. This decision was made because the execution of the algorithm may take a lot of time. Runing 30 generations takes around 50 hours (+/- 6 hours) and since the criteria may be reached in more than 30, it could not be run indefinetly. After the experiments we can conclude that adding a threshold exit value may not take many generations since after 30 generations we obtained a satisfactory result.

Within the scope of this project only one load case to be tested, but this work can easily be expanded by running the EA for any other load cases but the parameter limits re adjusted for the new load cases. The motor can execute accurately speed gradient commands higher speed like over 1400 rpm if the load is not that much heavy. But if the drum loaded with higher weight then it risky to run the motor with higher speed as the results could be catastrophe. During the experiment only a load of 2 kg were tested. If the loads are increase to higher amount then the spinning speed needs to be reduced. If this is not taken into consideration the algorithm may create a bad individual that will generate such vibration that the machine may walk[14] or break.

Other further research could be in the parameter settings for the EA, the parameters (i.e. creep rate or mutation probability) were set using common used values and not custom values that may be adequate for the project. Choosing these parameters wisely may lead to that the performance of the EA will be increased.

In the current implementacion of the EA, elitism was used as a method to keep the best solutio candidate unchanged so that it can prevail in many generations if it continues with good results. In this experiment only one candidate was choosen for elitism but also choosing more than one will increase the chances for good candidates to prevail for longer time. For example a good individual may have been the best for 4 generations but in the fifth, another candidate perfoms slightly better. In this case the good candidate may be discarded, but if 2 individuals are selected for elitism, it may still be alive and if it really

[14]Walk in this context refers to the displacement a washing machine does when high vibration makes it to loose the grip of its original position.

was a good solution, will keep preserving and as a result ever spread its genes to four an excelent candidate.

Another follow up area is the exploration of other bio inspired stochastic optimization methods such as particle swarm optimization[9] or even using a neural network [9] to optimize the speed scheme.

Finally, it was demonstrated that the use of Evolutionary Algorithms in the optimization of the unbalance value within the washing machine gave a satisfactory result that may be improved with further execution of the algorithm.

References

[1] D. C. Conrad and W. Soedel. On the problem of oscillatory walk of automatic washing machines. *Journal of Sound and Vibration*, 188:301–314, 1994.

[2] K. Deb and A. R. Reddy. Reliable classification of two-class cancer data using evolutionary algorithms. *Biosystems*, 72:111–129, 2003.

[3] J M. Deutsch. Evolutionary algorithms for finding optimal gene sets in microarray prediction. *Bioinformatics*, 19:45–52, 2003.

[4] Jesse HOAGG Henri GAVIN and Mark DOBOSSy. Optimal design of mr dampers. *Proc. US -Japan Workshop on Smart Structures for Improved Seismic Performance in Urban Regions*, pages 225–236, August 2001.

[5] P. Lingman and M. Wahde. Transport and maintenance effective retardation control using neural networks with genetic algorithms. *Vehicle System Dynamics*, 42:89–107, 2004.

[6] S. Nolfi and D. Floreano. *Evalutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, 2000.

[7] E. Papadopoulos and I. Papadimitriou. Modeling, design and control of a portable wahing machine during the spinning cycle. *Proceedings of the AIM 2001*, pages 899–904, Jul 2001.

[8] M. Wahde. Detennination of orbital parameters of interacting galaxies using a genetic algorithm. description of the method and application to artificial data. *Astronomy and Astrophysics*, 132:417–429, 1998.

[9] M. Wahde. *Biologically Inspired Optimization Methods: An Introduction*. WIT Press, 2008.

[10] H. Yamamotoo Y. Sonoda and Y. Yokoi. Development of the vibration control system g-fall balancer for a drum type washer dryer. *Proceedings of the AIM 2003*, pages 1140–1144, Jul 2003.

[11] Ray Martin Y. Yuan, Ali Buendia and F. Ashrafzadeh. Unbalanced load estimation algorithm using multiple mechanical measurements for horizontal washing machines. Oct 2007.

A Appendix: Optimization Program

In order to make possible this project it was required to develop a control interface software, custom made for the hardware (*i.e.* Motor controller, relay card.). The development of the software was made in C#, an object oriented high level programming language that provides full support for development of graphical interfaces and its native compatibility with windows environment. Also the support provided to all its libraries and functions is extense and easy to find over the internet. As one of the main languages supported by Microsoft^{TM}since they developed.

A.1 Design

The software was designed in a modular way following also the basic layers for software development. Each module was designed to perform a specific task in order to make it simple and avoid complexities that can cause problems. The main characteristics of the system were the need to be easy to maintain, easy to debug, easy to fix, easy to understand. However due to the time constraint of the project some parts were fixed without considering all this characteristics. Figure A.1 shows a diagram displaying the main modules considered since the early stages of the project, and they are:

- Graphical user interface that constructs the graphics and controls that the user will interact with, such as buttons, check boxes, etc.
- General Data Management and Storage which will handle the processing of the input data provided by the user and once processed, it will be available for the rest of the modules that will require it.
- Serial Comunication will take care of the interaction of the software and the washing machine hardware. It will also receive the information to be sent and packeted in the required format. And finally it will send and receive the answers or acknowledgments from the motor controller and decode those messages.
- Optimization algorithm module will take care of the optimization, receiving the processed data and answering with the optimization parameters or commands that will be sent to the motor controller.

With all the interaction through the modules and the structure described, it is ensured the trustability of the system and the optimized code will help to accomplish the tasks deadlines in order to execute the optimization in real time.

A.2 Development

Once the design has been developed and main modules specified, the next stage is the implementation. The implementation stage required an understanding on the project and the hardware since the motor controller uses a serial communication with the optimization software with a specific format that is specified by the manufacturer, in this case ASKO Cylinda AB.

In order to develop the software an incremental design was used. This model that helps create the program from scratch and the functions and/or features that the program does are developed one by one and tested first the initial or basic features from which the more complex ones will be build up. In this case there were 3 stages in which the software was developed.

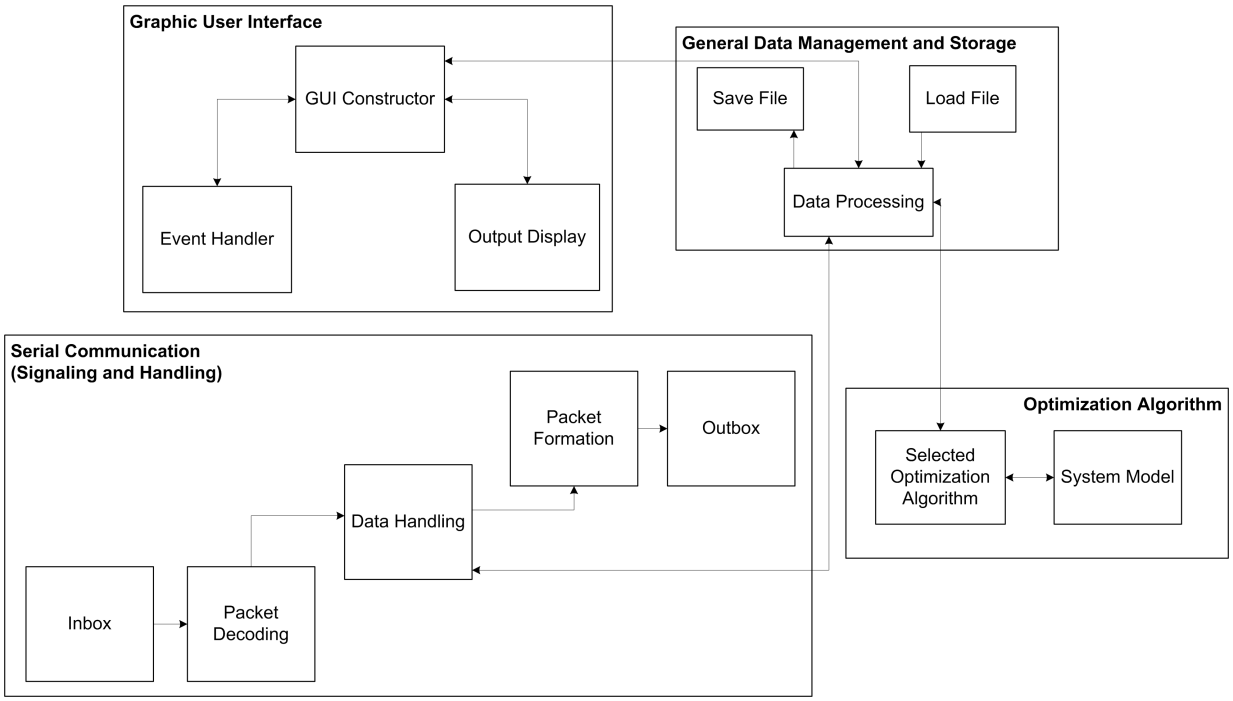

Figure A.1: Module diagram designed for the optimization software.

The first steps to start using the program shall be followed to ensure the right performance. At the beginning the program scans the system searching for serial ports. It requires two serial ports to communicate, one for the washing machine motor and the second for the relay card that controls the water pumps. If there is just one available port it can run only the manual mode. If there are no ports available it can not run any mode. Serial ports are showed in the dropdown boxes A.2(A) and after choosing the right ports the user shall press the *"OPEN PORT"* button (A.2(B)) after that in the system display A.2(E) shall appear the status of the connection. If the ports were connected after starting the program, the *"REFRESH PORTS"* button (A.2(D)) will refresh the ports lists at any time. If the ports need to be closed, the *"CLOSE PORT"* button (A.2(C)) shall be pressed and the conection is terminated. In the general display A.2(F), raw data coming from the motor controller is displayed in order to see the acknowledgements of the messages sent. The implementation works for few messages although it can be extended for any one.

A.2.1 Manual Mode

The first stage in the implementation was the development of a *"manual mode"* that will test the proper function of the commands that will be sent to the motor controller and the response that the program will receive. This is made to ensure that the controller works as specified and that the program also works according to that specification and avoid errors in later stages. Also the purpose is to have a manual setting of parameters in case is required by the experiment.

In figure A.2 we can see where the paremeters are set in order to send commands to the motor controller. In this case the motor controller can receive just a limited set of commands but only few were required by the project to control and interaction and they are:

1. Set reference speed of drum. A command that sets a speed to be reached, the speed gradient that will be used to reach that speed and the direction of rotation. Controls

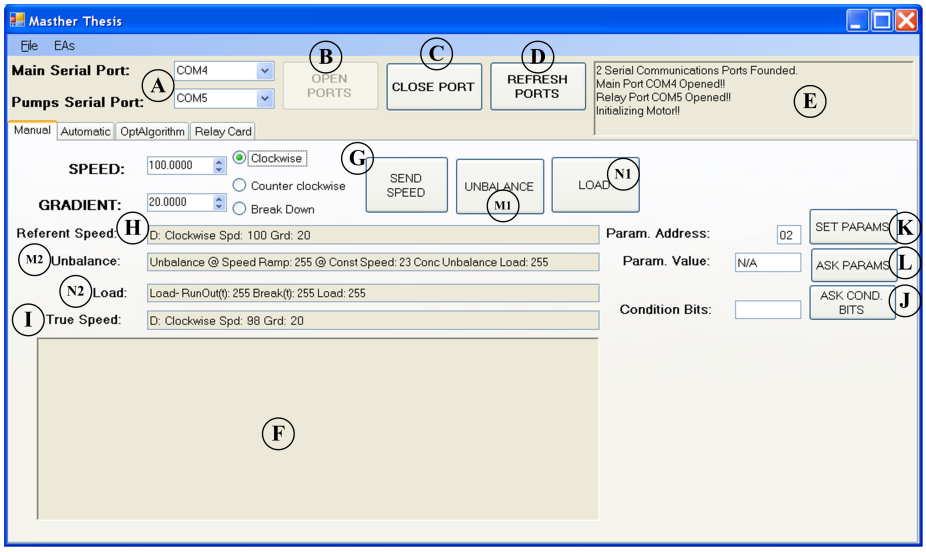

Figure A.2: Graphic user interface from the optimization software developed for this project.

related to this command are showed in A.2(G) where the speed and gradient are set numerically, the direction is set using the ratio buttons and then the *"SEND SPEED"* button is pressed to send the message.

2. Ask for reference speed of drum. It requests the current reference speed to which the motor is trying to reach or follow. This message is not commonly used in the program as the set reference speed acknowledgement is the most common indicator for the reference speed but, it can also be requested within the program if required. The requested data is displayed in A.2(H).

3. Ask for true speed of drum. It requests the instant speed that the motor is performing at the time of the request. This message is sent periodically in order to know the speed of the motor at any time and is used to plot the real performance. The data obtained from this message is displayed in the textbox A.2(I).

4. Ask for control bits. It requests for control flags that are on or off at specified situations. It is requested by pressin the A.2(J) and the bits shall be showed in the *"Conditions Bits"* textbox right next to the button.

5. Set parameters for drive. It specify the parameters for the motor to start working and it is used in the initialization of the system. The parameters are constituted by two fields, the address of the parameter (from 0 to 34) and the value of the parameter (from 0 to 65535). These two parameters are written in the *"Param. Address"* and *"Param.Values"* textboxes which are right next to the *"SET PARAMS"* button (A.2(K)) that sends the message.

6. Ask for parameters for drive. It requests the parameters that the motor currently have for working. As the set parameter the request will return two values that correspond to the address requested and the value that the address has. The button to request the parameters is the A.2(L) and the data is displayed in the corresponding textboxes.

7. Ask for unbalance. It requests the unbalance measured by the motor controller and it contains three fields, the unbalance at constant speed, the unbalance at ramp speed

Figure A.3: Graphic user interface from the optimization software developed for this project.

and the concentrated unbalance load measured in the drum. The *"UNBALANCE"* button (A.2(M1)) send the request and the data is displayed in the unbalance textbox (A.2(M2)).

8. Ask for load. It requests the load measured in the drum, and two parameters used to detect the load; the run out time and the break time per speed difference $\Delta t/\Delta\omega$. In order to start the measurement of the load, a speed scheme with certain characteristics has to be executed and it is not feasable to execute this speed scheme each time to trigger the measurements, so this message was not used for the final stages. Besides the speed scheme characteristics were not clear since it was not specified in the documentation although some empirical schemes made the measurements start, it was not 100% reproducible. The *"LOAD"* button (A.2(N1)) sends the message and the data is displayed in the textbox A.2(N2).

9. Ask for ID-Code software. It requests the software version of the motor controller. The message is sent after the initialization in order to check that the motor controller is comunicating in the right way and to check its software version. It is displayed in the general display A.2(F) every time the port is open.

Other commands are avalilable but, for the purpose of this project they will not be used although the skeletons of their commands are available within the code.

A.2.2 Automatic Mode

The *"automatic mode"'* is where a specific speed scheme is loaded into the program and can be executed as many times as required in an automatic way, where the user only has to wait for the scheme to finish. To load the file, click on the *"OPEN"* menu (FILE→OPEN) and select the file in the stored path. At the end of the scheme execution, the motor will stop when the time is finish, no matter which speed command was sent before. The structure of the file shall be as follows:

Direction,Speed,Gradient,Time
Command1
Command2
⋮
CommandN
where Direction is specified as *'r'* for right direction spin or *'l'* for left direction spin, the speed is specified in an integer from 0 to 2000 rpm, the gradient is specified in an integer from 0 to 200 rpm/s and the time is specified in an integer (miliseconds). And it shall be saved as csv[15] file format.

For example a file named test.csv that contains:
Direction,Speed,Gradient,Time
r,100,20,1000
l,300,40,2500
r,20,47,4000

the program will send the commands in order for the motor to first rotate to the right direction during one second at a speed gradient of 20 rpm per second, then it will stop and rotate to the left and try to reach 300 rpm at 40 rpm/s during 25 seconds and then stop and rotate to the right and try to reach 20 rmp at 47 rpm/s during 4 seconds. The file format is quite straight-forward. After the file is loaded, the speed scheme shall be showed in the scheme data grid A.3Ⓒ. There the scheme can be checked for errors or just to see which commands are going to be sent next. Also after loading the file, a graphical representation of how the speed scheme shall look like is displayed in a graph of time (seconds) versus speed (rpm) in the A.3Ⓓ area. If during the execution the user wants to ask for unbalance in a periodic way, the *"Ask for Unbalance"'* checkbox shall be set. The same applies to the load, if the user needs to know the load during the entire scheme execution, the *"Ask for Load"'* checkbox shall be set. After the data is loaded, the user shall press the *"RUN"'* button (A.3Ⓐ) to start the scheme. At this point only the *"STOP"'* button (A.3Ⓑ) can be used in order to stop the execution. If the current scheme requires to be saved, it can be done by clicking in the save menu (FILE→SAVE) and choose the right path and name.

Finally, a report file is generated by the program, in the a folder named in a date format (YYYYMMMDD) and within a subfolder named with the time in which the scheme starts (HH_MM_SS), that will help to track the experiment results. In this folder, three files are generated: *Speed.csv*, *Unbalance.csv* and *Load.csv*. Each file will contain the data obtained by the program for the specific case and the format for each file shall be described as follows:

Speed file data is stored within the file with the following format:
Speed[0],Time[0]
Speed[1],Time[1]
⋮
Speed[N],Time[N]
where speed is an integer from 0 to 2000 rpm and time is an integer with the relative time in miliseconds.

Unbalance file data is stored within the file with the following format:
RampUnbalance[0],ConstantUnbalance[0],ConcentratedUnbalanceLoad[0],Time[0]
RampUnbalance[1],ConstantUnbalance[1],ConcentratedUnbalanceLoad[1],Time[1]
⋮
RampUnbalance[N],ConstantUnbalance[N],ConcentratedUnbalanceLoad[N],Time[N]

[15]Comma separated values.

where RamUnbalance is the unbalance at ramp speed stored in a integer from 0 to 254^{16}, ConstantUnbalance is the unbalance at constant speed also stored in an integer from 0 to 254^{16}, the ConcentratedUnbalanceLoad is the unbalance load measured within the drum also an integer from 0 to 254^{16} and finally the time in milliseconds.

Load file data is stored within the file with the following format:
ROT[0],BT[0],Load[0],Time[0]
ROT[1],BT[1],Load[1],Time[1]
⋮
ROT[N],BT[N],Load[N],Time[N]
where ROT is the run out time per speed difference $\Delta t/\Delta\omega$ denoted with an integer from 0 to 254^{16}, BT is the break time per speed difference $\Delta t/\Delta\omega$ also stored in an integer from 0 to $254\Delta t/\Delta\omega$, Load is the concentrated load within the drum stored with an integer from 0 to $254\Delta t/\Delta\omega$ and the time in milliseconds.

A.2.3 Optimization Algorithm Mode

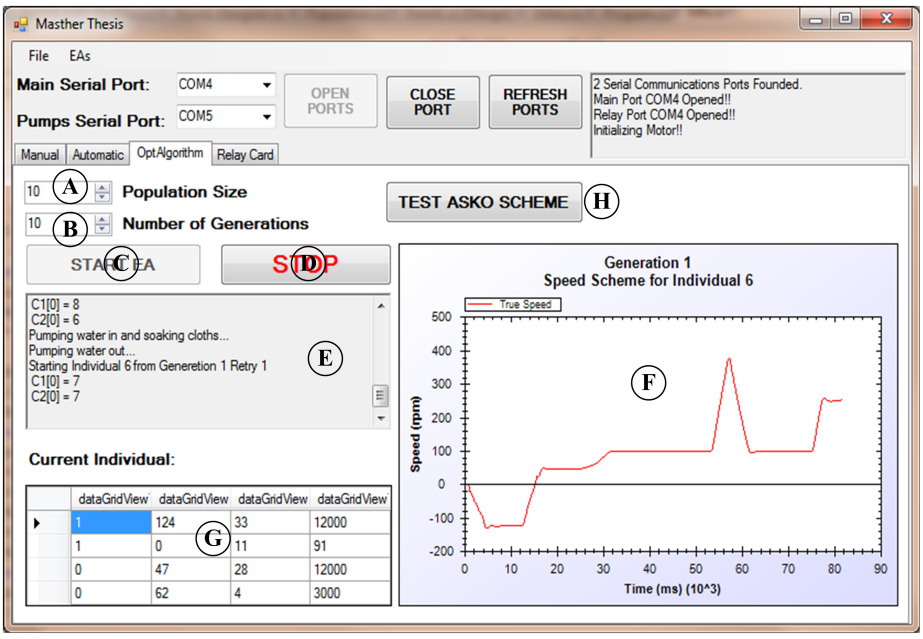

Figure A.4: Graphic user interface from the optimization software developed for this project.

The final stage of the program is the implementation of the optimization algorithm in this case the EA. In figure A.4, the main graphic user interface of this stage is showed. The main functionality is the result of the integration of the two previous stages and the EA. In order to run the algorithm first the population size needs to be specified in its numerical box A.4(A) along with the number of generations in A.4 mycircB. After fixing those settings the program is ready to begin by pressing the *"START EA"'* button (A.4(C)) right after the press the algorithm will start initilizing all the population and start executing the solution candidates (speed schemes). The EA display A.4(E) shall display the status of the algorithm like which candidate is being executed and in which part of the execution, and many other useful information that will help the user to follow the execution. In the EA data grid A.4(G), the solution candidate that is currently being executed is displayed, in order for the user to check the correctness of the execution in real time displayed in the

[16]Value 255 means no measurement.

EA graph area A.4Ⓕ, where, in a graphical way, the true speed values that the motor is giving are showed.

During the execution of the EA, it can be stopped by pressing the *"STOP"'* button A.4Ⓓ. After concluding a succesfull run of the EA, a report is generated in the root folder of the program[17] and it follows a special format that will give all the data from each generation as well as the results of the execution with the measurements data related to each candidate solution. The file is a csv file named *EAStatus.csv*. The first part of the file is the header that specifies the status of the execution like the generation and candidate that is being executed as well as the population size and the total number of generations. After that all the schemes from all the executed generations are showed with all the relevan information like the number of individual within the generation, the fitness value, the speed scheme values (speed, gradient, direction, time) and the measurements.

The EA file format is structured as follows:

CG,TG,CC,CS ← Header
#CC[0]G[0],FV[0]G[0]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5
#CC[1]G[0],FV[1]G[0]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5
⋮
#CC[N]G[0],FV[N]G[0]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5
#CC[0]G[1],FV[0]G[1]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5
#CC[1]G[1],FV[1]G[1]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time

[17] It refers to the folder where the program is stored.

C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5
⋮
#CC[N]G[M],FV[N]G[M]
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
#G,Dir,Speed,Gradient,Time
C1,M1,M2,M3,M4,M5
C2,M1,M2,M3,M4,M5

where CG is the index [18] of the current generation being executed, TG is the total number of generations, CS is the index [18] current candidate (scheme) being executed and CS is the index [18] of the current iteration[19] of the same generation. The body consists of the data of all the candidates in all the generations and is structured by a group of data that represents or model a solution candidate (speed scheme). Each speed scheme contains the same data. #CC[N] is the index [18] of the Nth candidate within its generation M (G[M]), FV[N] is the fitness value of that Nth individual from the generation M (G[M]) and it is stored as a decimal fraction that has a range from 0 to 1. If the fitness value is 0 it means that the candidate has zero means that the current generation haven't been evaluated if and it is the last generation executed because of an error in the program that stopped the program. #G is the index [18] of the gene if that speed scheme, and Dir, Speed (rpm), Gradient (rpm/s) and Time (miliseconds) are the same values as the previous formats although the direction changes the values, *"0"* represents the right direction and *"1"* represents the left direction. Finally the measurements for that Nth individual are stored in the next two lines, C1 represents the first unbalance measurement at constant speed and C2 represents the second unbalance measurement at constant speed. Each execution of a speed scheme gives two unbalance measurements. This data structure is repeated as many times as individuals in the generation times total number of generations and the first generation is stored first, the second next and so on until the last one.

Example of a EA file format:

0,10,10,5
0,0.0194174757281553
0,1,49,28,18529
1,1,77,0,3000
2,1,20,28,12000
3,0,0,6,3000
C1,56,34,39,81,48
C2,57,32,37,83,48
1,0.0264550264550265
0,0,41,42,12000
1,1,46,6,3000
2,1,52,28,12000
3,0,7,0,4345
C1,33,49,27,41,43
C2,37,42,25,42,39
2,0.0112866817155756

[18]The index is an integer starting from 0 as the first instance in order to be directly compatible with the code.

[19]The iteration is required due to the fact that each speed scheme requires five evaluations as stated in chapter 3.3.

0,1,49,28,17405
1,1,22,10,12780
2,0,49,42,18426
3,0,14,0,6452
C1,98,95,57,122,66
C2,99,97,59,122,71
⋮
9,0.0145985401459854
0,0,49,42,12000
1,0,0,0,7117
2,1,29,39,17785
3,1,80,0,8270
C1,100,90,96,35,20
C2,107,92,97,31,17
0,0.02398081
0,0,83,22,19120
1,0,18,0,4640
2,1,49,28,19093
3,0,0,0,912
C1,29,28,28,46,80
C2,28,23,28,42,85
⋮
9,0.02906977
0,0,41,42,12000
1,1,121,6,3000
2,0,52,26,12000
3,0,87,0,4345
C1,73,25,26,35,20
C2,74,21,22,31,17

A.2.4 Failsoft methods

Failsoft methods are the techniques used when a failure is present in software or hardware and helps the program to continue working as normal or at least save the state of the program so that it can resume later when the problem is solved. One of the main failures or the most frequent is the lost of communication between the motor controller and the optimization program. There is some methods that tries to reconnect the serial ports if there is a synchronization problem or something related with the optimization program but these methods were not solving the problems. In order to resume the execution of the algorithm and to avoid wasting time if a failure is present a load state feature was implemented using the EA file as input. If the software fails at any given point the EA file will still have stored the data at the point of failure so accordind to that data, the process can be resumed using the header as the status settings and integrating the candidates data to the program. In order to do this the file has to be loaded by clicking in the EA load menu (EAs $\rightarrow$ Load Previous Values) and open the EAStatus file. NOTE: Good care needs to be taken when loading the file, before loading the file the user needs to be sure that all the information is coherent. This is that if the header states that the last generation executed was the fifth, the last individual executed was the second and the generation was in the third iteration, the body of the file corresponds to that data which means that there

shall be five generations data, four generations with all the data set including fitness values and the last generation with only 2 measurements for C1 and C2, and only the first scheme has 3 measurments as it stopped in the second one. This is made to avoid corruptions in the file that can be present or corruptions in the program while loading this file.