A decision framework to repair or eliminate failures in engineering systems according to their Net present value

Luna Burgos Moreno

Master Thesis

Thesis Advisor: Dr. Karen Marais, Purdue School of Aeronautics and Astronautics

Abstract.

This thesis considers the impact of failures in terms of the effect they have in the value of the system. We begin with a qualitative discussion on the value of failure acceptance or reduction. Next, we present a decision framework based on net present value. Then, we develop an analytical model of net present value given a stochastic failure process, quantify the net present value obtained of the system under certain decisions and define the Value of Redesign. To end, we give some guidance to interpret the results and prove the dynamism of the optimal decision process with time, and system and market costs. Moreover, we conclude that although redesigning the system may or may not be the best solution, it always removes uncertainty.

Table of Contents

FIGURES

TABLES

1 Introduction

When failures occur, organizations are faced with a range of choices, the two extremes of which are to act to remove a particular failure mode, or to accept the failure and, if necessary, replace failed or faulty components. What is the best choice in a given situation? Here, we present a decision framework based on net present value to aid in such decisions.

We begin chapter 2 with a qualitative discussion on the value of failure acceptance or reduction. Next, in subsection 2.2, we develop an analytical model of net present value given a stochastic failure process, and define the Value of Redesign. Section 2.3 introduces a case of study and provides some results that show the strong effect of time and costs in the Value of Redesign. Chapter 3 concludes the thesis.

2 Theory: A Value Perspective on Failure Reduction

Figure 1 shows the decision tree that we establish, to show us the methodology followed to decide how to act efficiently according to different situations. First, we get a failure reported. Once we have that failure, if we have enough information, we make an estimation of the parameters that define the process we think these failures will follow. With that estimation we predict future failure rates, estimate the Net Present Value when we decide to repair the failures reported, and estimate the Net Present Value if we do not only repair but also decide to redesign the system so it does not fail again this way. After these predictions, we decide to act removing the reported failures, or repair and redesign the system, or just ignore these failures and do not repair or redesign anything.

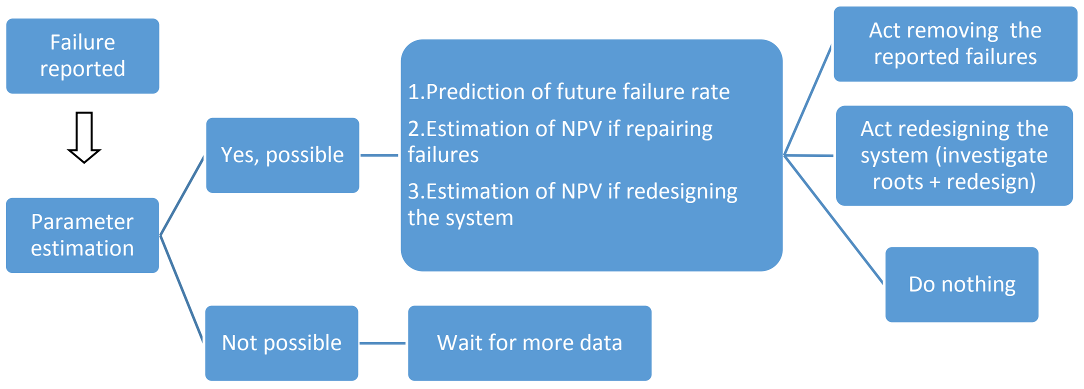
Failure reported
Parameter estimation
Yes, possible
Not possible
1.Prediction of future failure rate
2.Estimation of NPV if repairing failures
3.Estimation of NPV if redesigning the system
Wait for more data
Act removing the reported failures
Act redesigning the system (investigate roots + redesign)
Do nothing

Figure 1: Decision tree

2.1 The value perspective on failure reduction: a qualitative discussion

Failures have several associated impacts on profit, including direct costs like the cost of new components to replace failed components, downtime associated with the failure and subsequent repair, and indirect costs like reduced customer satisfaction. Preventing failures, whether by eliminating them completely or through preventive maintenance, also has its own costs, such as the labor cost of troubleshooting, downtime associated with troubleshooting, the cost of replacing components, and the cost of preventive maintenance. If a component (e.g., a lightbulb) is cheap, difficult to inspect, easy to replace, and does not have a safety critical function, replacement on failure may be the best option. If a component's failure entails significant downtime and cost (e.g., an aircraft empennage), but the root causes (e.g., metal fatigue) of the failure are difficult or impossible to eliminate, inspection and preventive maintenance may be the best option. And if a component's failure entails significant and possibly frequent downtime and cost (e.g., a software bug), but it is possible that the root causes (e.g., incorrect specification) of the failure can be found, eliminating the failure mode may be the best option.

Further complicating this choice is the stochastic nature of failure—a failure may recur frequently, or rarely. When a new failure mode appears, it may not be possible to determine immediately whether it will be a frequent or rare occurrence. This problem is exacerbated in the case of NFFs, where it may not even be clear what, if anything, failed.

Each year, commercial airlines in the United States spend about $185,000 each unsuccessfully attempting to replicate reported failures on electronic devices installed on airliners [Werner, 2015]. Such events are variously referred to as *no fault found* (NFF), *trouble-not-identified* (TNI), *cannot duplicate* (CND), *no-trouble-found* (NTF), and *retest OK* (RTOK).

An NFF event implies that a failure (fault) either *occurred* or was *reported to have occurred* during a system's use, but upon subsequent investigation there was either no evidence of the failure (e.g., a burnt-out circuit), or the failure could not be replicated [Qi. et al., 2008]. Most computer users have experienced NFFs in the form of a computer crash or hang, which is often easily addressed by rebooting the computer. NFFs are found anywhere electronics are used, including the automotive, avionics, telecommunications, computer, and consumer industries [Qi et al., 2008].

NFFs can be dangerous [e.g., Hockley and Lacey, 2015]. In September 2010, during final approach, the crew of a Bombardier Dash 8 Q400 was so distracted by a non-functioning flight display, that they inadvertently disabled the autopilot and would likely have hit the ground, had the ground proximity warning not been activated [Werner, 2015]. The same underlying problem with the input/output processor occurred on several other flights, but each time technicians were unable to replicate the problem.

Even when they do not directly impact safety, NFFs can still have significant costs, both tangible (decreased reliability and availability, the costs of attempting to replicate and correct the fault, and warranty costs) and intangible (e.g., customer perception of inadequate quality).

Thus, there is also a time dimension to the previous choices—act early with limited information, possibly making a poor choice, or, wait for more information, possibly incurring frequent costly failures.

Here we show how a value-based perspective can help address this problem. We follow the approach presented in [Marais, 2013].

Our argument is based on four main ideas:

1. We assume that numbers of failures corresponding to different failure modes are reported each month, and that each month offers an opportunity to decide whether to eliminate the failure, or wait for another month. We specify that each failure count is associated with a particular failure and failure mode.
2. We consider that the system generates a set amount of revenues per unit time. When failures occur, the revenue stream is interrupted by downtime, and costs may be incurred to replace failed components, as discussed above. Similarly, eliminating a failure may also interrupt the revenue stream, and incur costs to replace components.

3. At each time step, we calculate the net present value associated with eliminating the failure mode, or waiting for the next month. We bring all values into the present using discounting.
4. We formulate a decision tree that can be used to determine the optimal action to carry out to obtain maximum NPV.

In the next section, we set up the analytical framework that corresponds to this qualitative discussion.

2.2 An analytical model of the value of failure resolution decisions

In developing our value model of failure resolution decision-making, we make some assumptions that will help to keep the focus on the main argument of this work. The assumptions are the following:

1. We consider discrete time steps of one month, and we account for downtime as a fraction of this time.
2. The problem time horizon is finite and is set to five year period.
3. Failures are reported per time step, and our data is collected over a fraction of the time horizon, being set to one year.
4. We assume that failures are evenly distributed over the time step in which they are reported.
5. We reflect downtime associated with a failure as a decrease in system revenues.
6. We associate a cost with each failure occurrence to account for replacement of components.
7. We also consider a fix operating cost for each time step independent from the failure rate, and a specific hardware cost for each type of failure that we are accounting.
8. We assume a redesign cost independent from the previous costs defined. To simplify the problem, this cost will be assumed over one time step.
9. We consider each month to have 30 days.
10. We consider to have already paid for the cost of discovering the failures that we are interested in, in this case NFFs.

In the following, we consider that failures happen at each time step, once we have that data we estimate, if possible, the distribution parameters. Then we predict the future failure rate over the time horizon estimating the NPV of the system and compare it with the NPV of removing those failures. Finally, according to these results, we decide not to act (if the cost of these failures is so small that it does not even worth to repair them), or act by repairing them, or redesign the system so it does not fail that way again. Figure 2 shows a detailed decision tree with these three possible actions at each time, being the "New System" the system we obtain once we repair and remove a type of failure. This "New System" it is not perfect, it has other type of failures that have not been considered to study before yet.

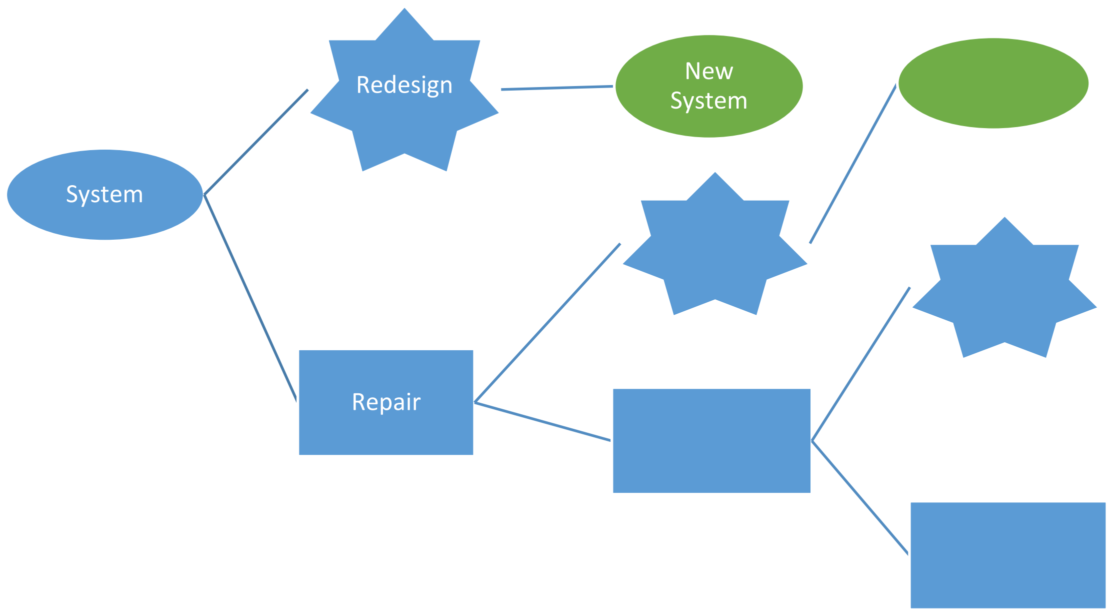

Figure 2: Detailed decision tree

Figure 2 shows an initial system and the three main decisions we can take once a failure has been reported. First, we can decide to redesign the system (for example, if the failure rate reported during the first month is so big that we need to get rid of these failures to make profit). If we decide to redesign, we will get a "New system" right away.

We can also choose to repair the reported failures, and after another month of data, make the same decision again, repair or redesign.

2.2.1 Input failures

We consider our failures to follow an exponential process. Let us denote that any other failure process could have been addressed as this model can be used for any failure type but, because of its simplicity and its good approximation to failures of type NFFs the exponential process has been chosen.

NFFs occur continuously and independently at a constant average rate. Considering the exponential process, allow us to describe the time between events in a Poisson process that perfectly address that problem.

2.2.2 Parameter Estimation

This subsection focus on the parameters' estimation, but before deepening in this idea we must take into consideration the uncertainty factor.

The uncertainty factor is the fact that we never know for sure if the distribution that we assume to have for the existent data is really followed by that data. Even if it happens that we assume the proper distribution, it may not be well estimated if we do not have enough input data. Both uncertainties can lead to erroneous parameter estimation.

In practice, we can rarely, if ever, be certain that an observed quantity is drawn from a specific distribution. The most we can say is that our observations are consistent with the hypothesis that x is drawn from a distribution of that form.

Being said that, in this section we show how to estimate the distribution parameters associated to the exponential process and how to calculate the confidence intervals to define the variation bounds of our results.

Parameter estimation for a Homogeneous Poisson Process.

This model comes about when the inter-arrival times between failures are independent and identically distributed. The cumulative distribution function for Homogeneous Poisson Process is given by

$$F(t) = 1 - e^{-\lambda \cdot \mathrm{t}} \qquad (1)$$

Where λ is the failure rate per unit time (month), being constant for this distribution.

Thus, the failure rate distribution over time is:

$$h(t) = \lambda \cdot \mathrm{t} \qquad (2)$$

The mean time to fail following an exponential distribution is:

$$MTTF = \frac{1}{\lambda} \qquad (3)$$

For our predicted data, we estimate a failure rate$\hat{\lambda}$ that is equal to the average of failures reported during the months of data considered (n).

$$\hat{\lambda} = \frac{1}{\mathrm{n}}\sum_{i=1}^{n} \lambda_i \qquad (4)$$

Where λ_i is the number of failures reported each month.

The table below (Table 1) shows and example of how the failure rate estimation varies each month according to the failures reported.

Table 1: Example of estimation of lambda

n	Month 1	Month 2	Month 3

λ_i	0	1	2
$\hat{\lambda}$	0	$0.5 = \frac{0+1}{2}$	$1 = \frac{0+1+2}{3}$

The predicted failure rate predicted follows this equation

$$\hat{\mathrm{h}}(t) = \hat{\lambda} \cdot t \tag{5}$$

Let us illustrate with the next example given by table 2 and Figure 2.

Table 2: Example of a case generated following a HPP process with λ=4/3

Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$\boldsymbol{\lambda_i}$	1	3	2	2	3	0	1	1	0	0	0	2

Figure 3 shows the example of the estimation of the projected failures over time that the HPP applies to the example given by table 2. Each line represents the number of months of data available and the projection with that many months of data. The first prediction assumes that we only have one month of data reported, the second one assumes we have two months of data, and so on.

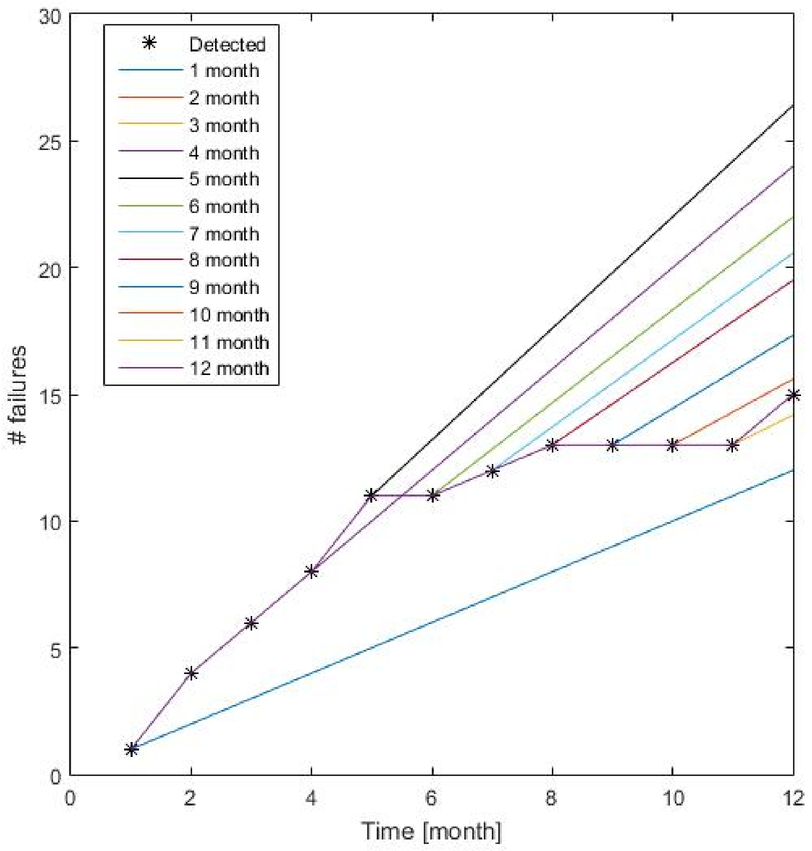

Figure 3: Example of failure estimation following an HPP for the case shown in table 2

2.2.2.1.1 Confidence intervals calculation

To define the bounds within which the results may vary in our study we apply confidence intervals to our results.

We calculate the Maximum Likelihood Estimate for a Poisson distribution, considering a confidence level of 95% for the two-sided confidence bounds.

Once we have the failure rate estimated $\hat{\lambda}$, we calculate the sum of all the failures estimated r_n for the time considered (n months):

$$r_n = \hat{\lambda} \cdot n \tag{6}$$

For $r_n < 100$ the failure bounds are calculated as follows:

$$\widehat{\lambda_{upper_n}} = \mathrm{G}^{-1}\left(1 - {}^{\alpha}/_{2}\,, 2(r_n + 1)\right) \tag{7}$$

$$\widehat{\lambda_{lower_n}} = \mathrm{G}^{-1}\left({}^{\alpha}/_{2}\,, 2r_n\right) \tag{8}$$

Where *G(q,v)* is the Chi Square distribution function.

On the contrary, for values $r_n > 100$ the failure rate bounds are calculated following a Normal inverse cumulative distribution function:

$$\widehat{\lambda_{upper\_n}} = \mathrm{N}^{-1}\left(1 - {}^{\alpha}/_{2}\, , r_n, \sqrt{r_n}\right) \tag{9}$$

$$\widehat{\lambda_{lower\_n}} = \mathrm{N}^{-1}\left({}^{\alpha}/_{2}\, , r_n, \sqrt{r_n}\right) \tag{10}$$

2.2.3 NPV Calculation

For a better approximation of reality, we consider to divide all the failures appearing in our system under two types: A and B. Being A the failure type that we are interested in, e.g. NFFs, and B all the other existing failures, as the mechanical or structural ones.

In this section, we explain the differences in NPV calculation depending on if we are just repairing the failures reported or if we repair the reported ones but also redesign the system so that type of failure does not appear again.

Table 3 shows the parameters that we consider for calculating the different present values.

Table 3: Nominal parameters definition

Parameters	Symbol
Repair Time failure A [-/month]	RT_A
Repair Time failure B [-/month]	RT_B
Repair Cost failure A [$/month]	RC_A
Repair Cost failure B [$/month]	RC_B
Benefits [$/month]	u
Operating cost [$/month]	C_{op}
Hardware cost failure A [$/failure]	C_{hard_A}
Hardware cost failure B [$/failure]	C_{hard_B}
Monthly discount rate [%]	$\mathrm{r}_{\Delta t}$
Redesign cost [$/month]	C_{design}

2.2.3.1 NPV when just repairing.

Benefits are defined as the time step revenue that we obtain from a system when no failures are reported. If the failure rate reported at a specific time step is different from zero, the revenues accounted by the system are equal to the monthly benefits (as if nothing fails) minus a variable time step fraction. This fraction varies with the number of failures reported and the mean repair time that failure needs to let the system be operative again.

Saying that, the following equation gives us the benefits over the time step period:

$$Revenues\ \left[{}^{\$}/_{month}\right] = u - \lambda_{i_A} \cdot RT_A \cdot \frac{u}{30} - \lambda_{i_B} \cdot RT_B \cdot \frac{u}{30} \tag{11}$$

In terms of costs, we consider not only the only the operating and repairing costs but also a hardware cost for each failure reported. Leading to the following equation:

$$Costs\left[{}^{\$}/_{month}\right] = \lambda_{i_A} \cdot RT_A \cdot RC_A + \lambda_{i_A} \cdot \mathrm{C}_{hard_A} + \lambda_{i_B} \cdot RT_B \cdot RC_B + \lambda_{i_B} \cdot \mathrm{C}_{hard_B} + \mathrm{C}_{op} \tag{12}$$

The Present value then can be obtained from the subtraction of benefits minus costs:

$$\mathrm{PV}_i = \frac{\left(u-\lambda_{i_A}\cdot RT_A\cdot\frac{u}{30}-\lambda_{i_B}\cdot RT_B\cdot\frac{u}{30}\right)-\left(\lambda_{i_A}\cdot RT_A\cdot RC_A+\lambda_{i_A}\cdot\mathrm{C}_{hard_A}+\lambda_{i_B}\cdot RT_B\cdot RC_B+\lambda_{i_B}\cdot\mathrm{C}_{hard_B}+\mathrm{C}_{op}\right)}{(1+\mathrm{r}_{\Delta t})^{t_i}} \tag{13}$$

And thus, the Net Present Value

$$\mathrm{NPV} = \sum_{i=1}^{n} \mathrm{PV}_i \tag{14}$$

2.2.3.2 NPV when redesigning.

When we decide to redesign the system there will be two different Present Value estimations. These are:

- The Present Value over the month where we repair failure type A, invest and redesign the system. Being the benefits over that time step period calculated as:

$$Revenues\ \left[{}^{\$}/_{month}\right] = u - \lambda_{i_B} \cdot RT_B \cdot \frac{u}{30} \tag{15}$$

And its costs, considering its redesign cost:

$$Costs\left[{}^{\$}/_{month}\right] = \lambda_{i_B} \cdot RT_B \cdot RC_B + \lambda_{i_B} \cdot \mathrm{C}_{hard_B} + \mathrm{C}_{op} + \mathrm{C}_{design} \tag{16}$$

- The Present Value after the introduction of the new design where only type B of failures remains appearing. Being the revenues calculated as in equation 15, and its cost calculated as:

$$Costs\left[{}^{\$}/_{month}\right] = \lambda_{i_B} \cdot RT_B \cdot RC_B + \lambda_{i_B} \cdot \mathrm{C}_{hard_B} + \mathrm{C}_{op} \qquad (17)$$

Once both revenues and costs are defined, we calculate the present value as

$$\mathrm{PV}_i = \frac{Revenues - Costs}{(1 + \mathrm{r}_{\Delta t})^{t_i}} \qquad (18)$$

And thus, the Net Present Value is calculated as in (14).

2.3 Model Assessment

Here, we use a Monte Carlo approach to generate different failure sequences, estimate the corresponding distribution parameters, and then compare the predicted NPVs of resolving or eliminating failures.

We determined the necessary number of runs as follows.

[Driels and Shin, 2004] define an equation to determine the number of iterations that need to be performed in order to obtain a specified accuracy in the result. The following formula provides the answer for that question.

$$p = \left[\frac{100 \cdot z_c \cdot S_\lambda}{E \cdot \bar{\lambda}}\right]^2 \qquad (19)$$

Where p is the number of iterations we need to perform, z_c is the confidence coefficient, S_λ is the standard deviation of the sample, E is the error assumed, and $\bar{\lambda}$ is the average of the sample.

The value of the confidence coefficient corresponds to the one given by a normal distribution, when the sample of data available is bigger than 25 values.

We set, as a first estimation for our Monte Carlo study, to run 900 different cases. Once obtained that data, we apply the previous equation to obtain the real number of runs needed to get a sample whose results fall within the confidence interval that we set. Table 4 shows the main input parameters that we first estimate.

Table 4: Parameters of our first estimation

Parameter	Value
λ_{input}	4/3
Confidence interval [%]	95
$p_{estimated}$	900
z_c	1.96

Table 5 shows the values obtained from the previous table (Table 4) of the average, the standard deviation, and the number of iterations needed to obtain a sample of data that falls within the confidence intervals.

Table 5: Average, Standard deviation, and iterations for λ_{input}=4/3

n	1	2	3	4	5	6	7	8	9	10	11	12
$\bar{\lambda}$	1.328	1.330	1.316	1.319	1.304	1.314	1.318	1.321	1.321	1.324	1.326	1.327
S_λ	1.189	0.827	0.655	0.578	0.505	0.456	0.425	0.393	0.375	0.356	0.346	0.327
p	1232	595	381	296	231	186	160	137	124	112	105	94

Table 5 shows the number of iterations estimated according to [Driels and Shin, 2004] when the monthly failure rate is 1.33 but, for the purpose of this thesis, we are interested in studying the variation in the results when the failure rate increases or decreases. To do so, we also ran our Monte Carlo study for different input failure rates, obtaining different number of iterations, and we set, as our number of iterations needed, the maximum over all the values obtained.

We observed that for bigger failure rates, the estimated number of iterations was very low. That unexpected result gave us the idea to apply another methodology to check the number of runs needed to obtain a good approximation to reality. We call it, the cumulative average method, and it determines the number of runs as follows.

First, it takes the estimated parameter of the process, and we calculate the cumulative average of its value among all the runs. Meaning that, for 2 months of data, and 8 runs, we estimate the cumulative average that those 8 different cases give us for 2 months of data.

Table 6 illustrates and example:

Table 6: Example of the cumulative average calculation of lambda estimated

Cases considered	Case 1	Case 2	Case 3
λ_2	0	1	2
$\bar{\lambda}$	0	$0.5 = \frac{0+1}{2}$	$1 = \frac{0+1+2}{3}$

Once all cumulative averages have been calculated, we graph them in a plot where each cumulative average corresponds to a number of runs. Figure 4 shows the result for an input failure rate of 1.33. Once each estimation stops oscillating and converges to a constant value, we have reached the number of runs needed to obtain a good approximation.

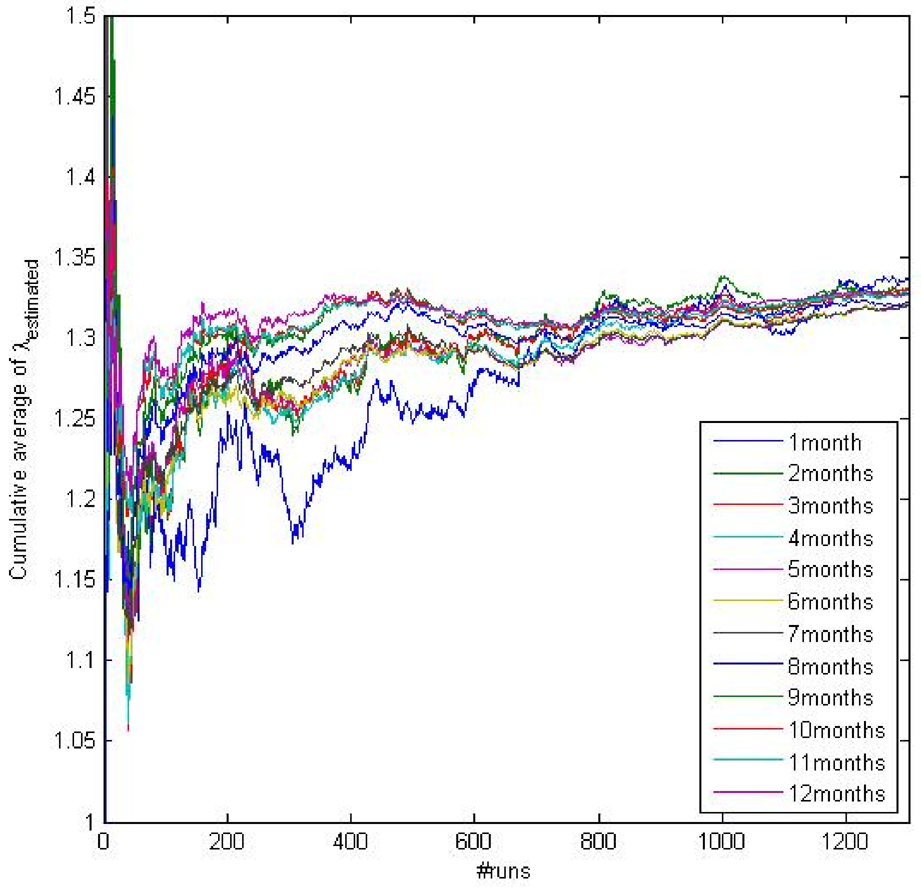

Figure 4: Cumulative average of λ estimated for the exponential distribution, for λ_{input}=4/3

Comparing the results obtained by both methodologies, the number of runs needed to obtain a good sample of data differs from one method to another. Due to this, we chose to stablish as the number of runs, the lowest value obtained by both methodologies, which covers all possibilities. Being said this, we stablish that 1300 runs are needed.

2.3.1 Input failure sequences

As already specified in 1.2.1, we consider a type of failure process—the exponential process. Table 7 shows the values we consider for this distribution. As mentioned before, we model both failures to follow an exponential process being then its failure rate constant.

Table 7: Input distribution parameters

Exponential distribution (HPP)	λ_{input} [-/month]
Failure type A	[0.5,18]
Failure type B	[2,54]

Figure 5 shows examples of exponential failure sequences generated in our Monte Carlo study.

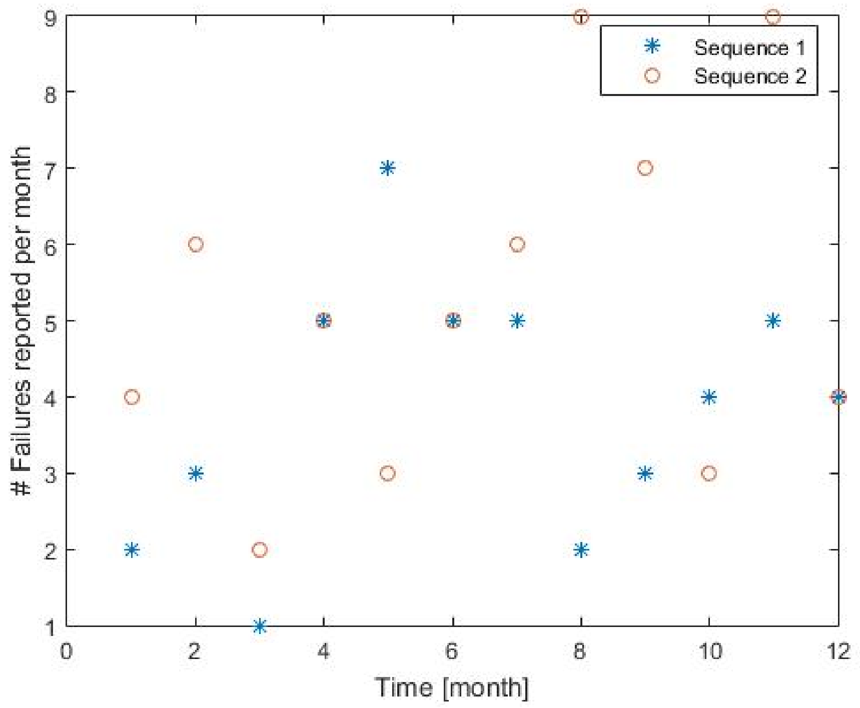

Figure 5: Example of sequences generated for $\lambda_{input} = 5$

2.3.2 NPV Calculation and the value of redesign

Table 8 shows the values assigned to the nominal parameters.

Table 8: Nominal parameters values

Parameters	Symbol	Value	Notes
Repair Time failure A [-/month]	RT_A	0.0042	~3h
Repair Time failure B [-/month]	RT_B	0.0084	~6h
Repair Cost failure A [$/month]	RC_A	[10, 4000]	
Repair Cost failure B [$/month]	RC_B	50	
Benefits [$/month]	u	1500	
Operating cost [$/month]	C_{op}	50	
Hardware cost failure A [$/failure]	C_{hard_A}	10	
Hardware cost failure B [$/failure]	C_{hard_B}	20	
Monthly discount rate [%]	$r_{\Delta t}$	0.4167	5% anual
Redesign cost [$/month]	C_{design}	[1000,8000]	

We consider a range of repair costs and redesign costs so we can measure the impact in NPV when our failures have a low cost to be repaired/redesign or, on the contrary, when their cost is high. Figures 6 and 7 give us an example of how these costs affect our decisions.

While Figure 6 shows an example where each failure incur in a low cost to repair or redesign, figure 7 shows the same scenario when those costs are high.

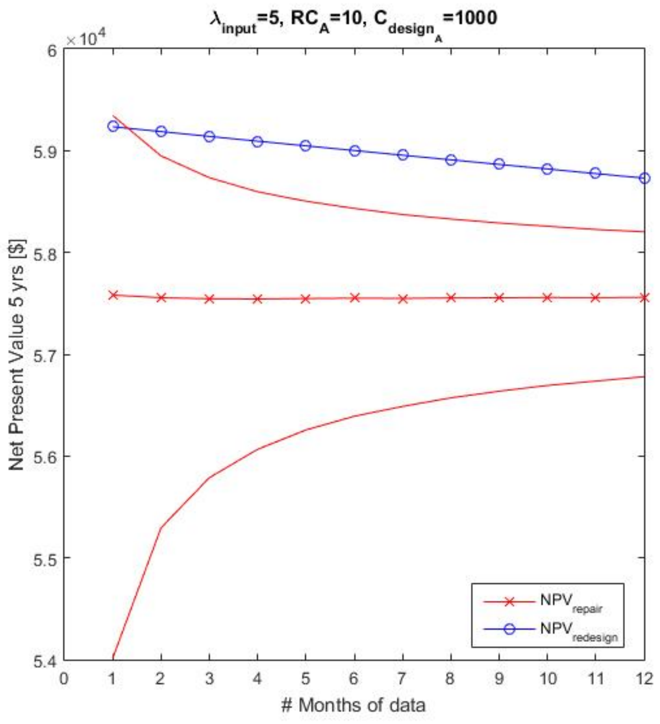

Figure 6: NPV estimation for a 5 year period horizon, low cost assumed

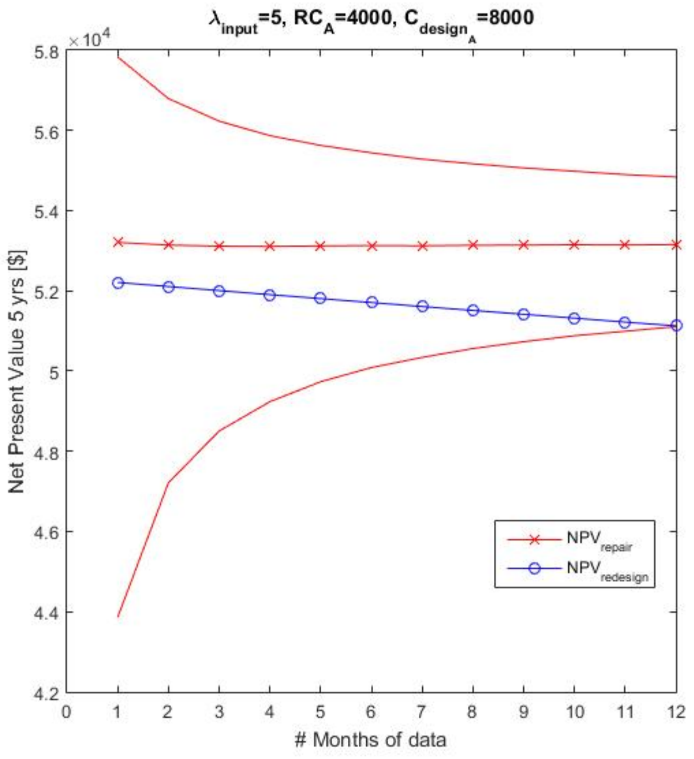

Figure 7: NPV estimation for a 5 year period horizon, high cost assumed

A better way to see when it is worth to invest in a new system design, is looking at the Value of Redesign (VoR). The value of redesign defines when it is worth to investigate the roots of our failure and redesign the system so it does not fail this way again. It is expressed as

$$VoR = NPV_{redesign} - NPV_{repair} \tag{20}$$

Figures 8 and 9 serve as an introduction of what we denominate the value of redesign. These figures (8 and 9) consider also the values of table 8. When its value is positive (Figure 8), redesigning is the best decision. On the contrary, when its value is negative (Figure 9), repairing the detected failures happens to be the best one.

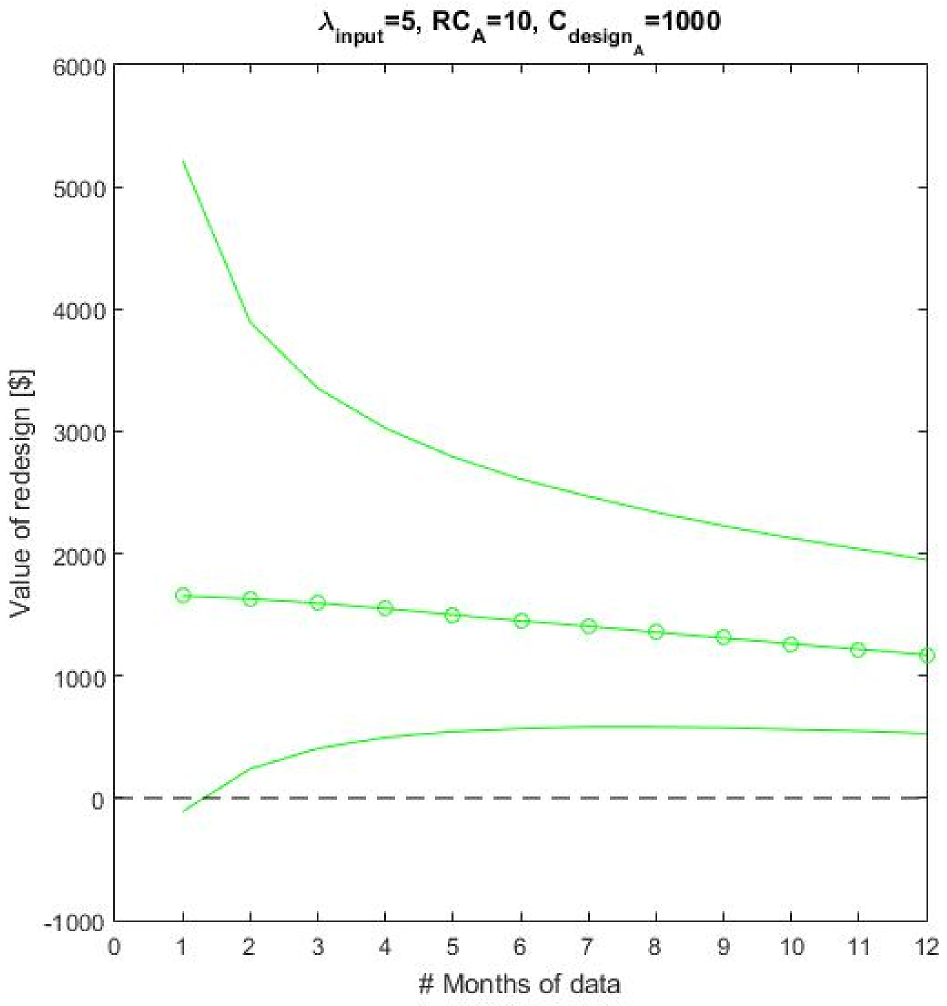

Figure 8: Value of redesign for a 5 year period horizon, low cost assumed

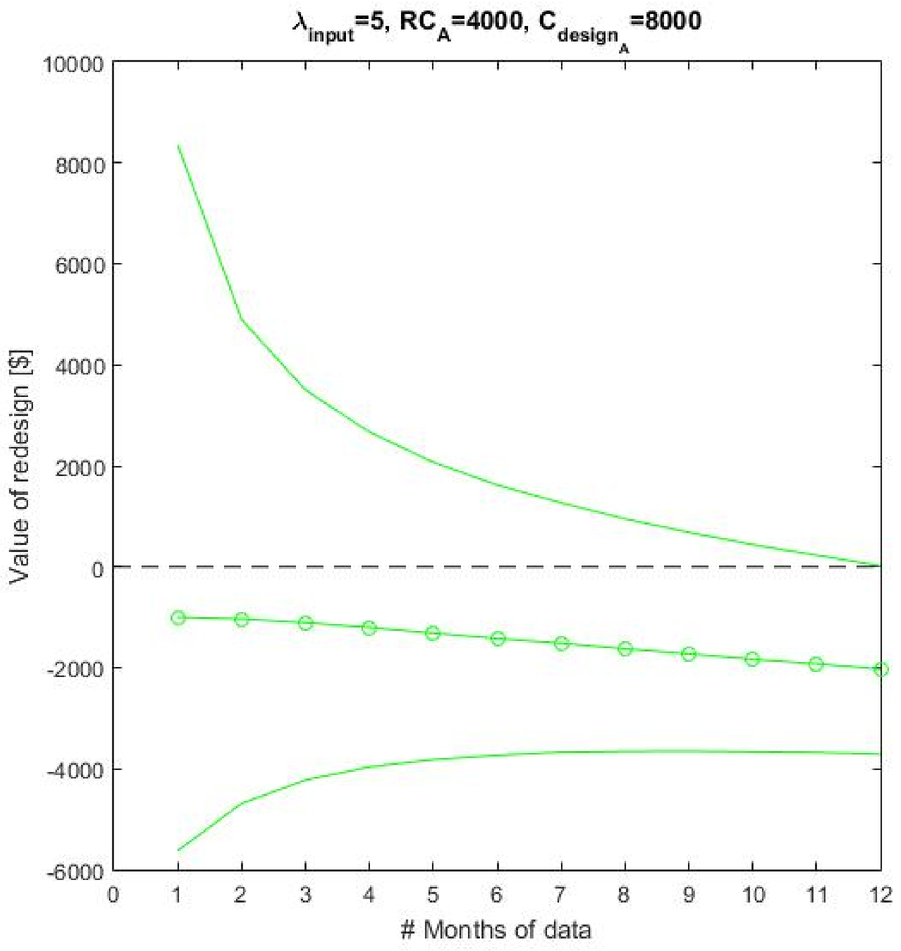

Figure 9: Value of redesign for a 5 year period horizon, high cost assumed

However, the value of redesign is dynamic with time, and it may also happen that there exists a decision point at which we must no longer think about redesigning, but we should start thinking about repairing the failures detected. Figures 10 and 11 show an example where the failure rate considered is the same as previous examples (figures 6 to 7), the time horizon is 5 years, but the costs have increased. Due to this cost increment, the decision varies from one month to another. For this specific example, there is a decision point at which redesigning stops being the best decision that moment occurs between months 4 and 5.

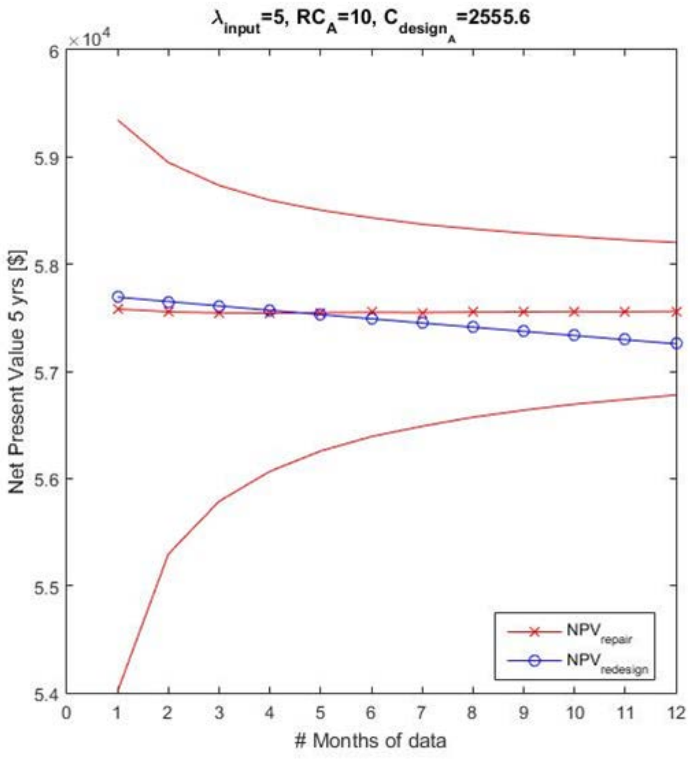

Figure 10: NPV estimation for a 5-year period horizon, higher costs

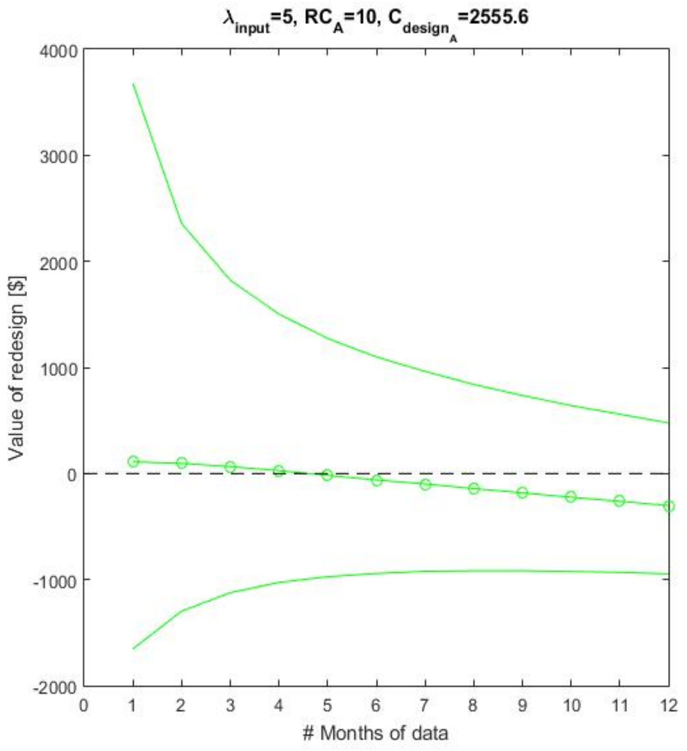

Figure 11: Value of redesign for a 5-year period horizon, higher costs

This dynamism of the Value of redesign within time can also be observed when we have bigger operating lives. Let's assume that our system has an operating life longer than 5 years, it would make sense to estimate the NPV of our system over its operating life. For the sake of this explanation, let us consider a system whose operating life is 15 years, and it has the same properties as of the system defined in tables 7 and 8.

Figure 12 shows the evolution in the NPV when we assume a 15 year time horizon. We can recognize that besides being the same system as before, now the NPV of redesigning remains almost constant over the time. This result claims that short operating lives tend to define a point at which it makes no sense to invest in a new system. It would be cheaper just to repair the existing failures that appear each month until the end of the useful life of the system. On the contrary, longer operating lives decrease the variation in the NPV of redesign, defining decisions that can change completely depending on the time horizon.

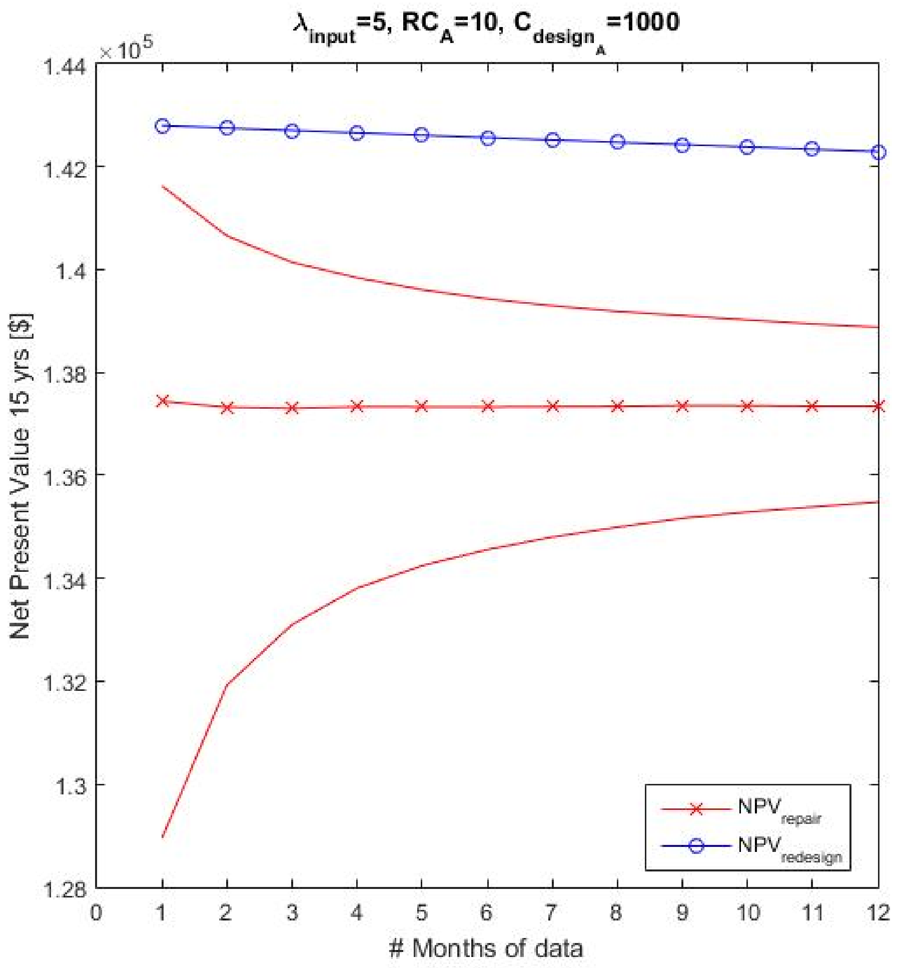

Figure 12: NPV estimation for a 15-year period horizon

Figure 13 reinforces the previous idea of figure 12 showing the variation of the value of redesign, being always positive for this time period.

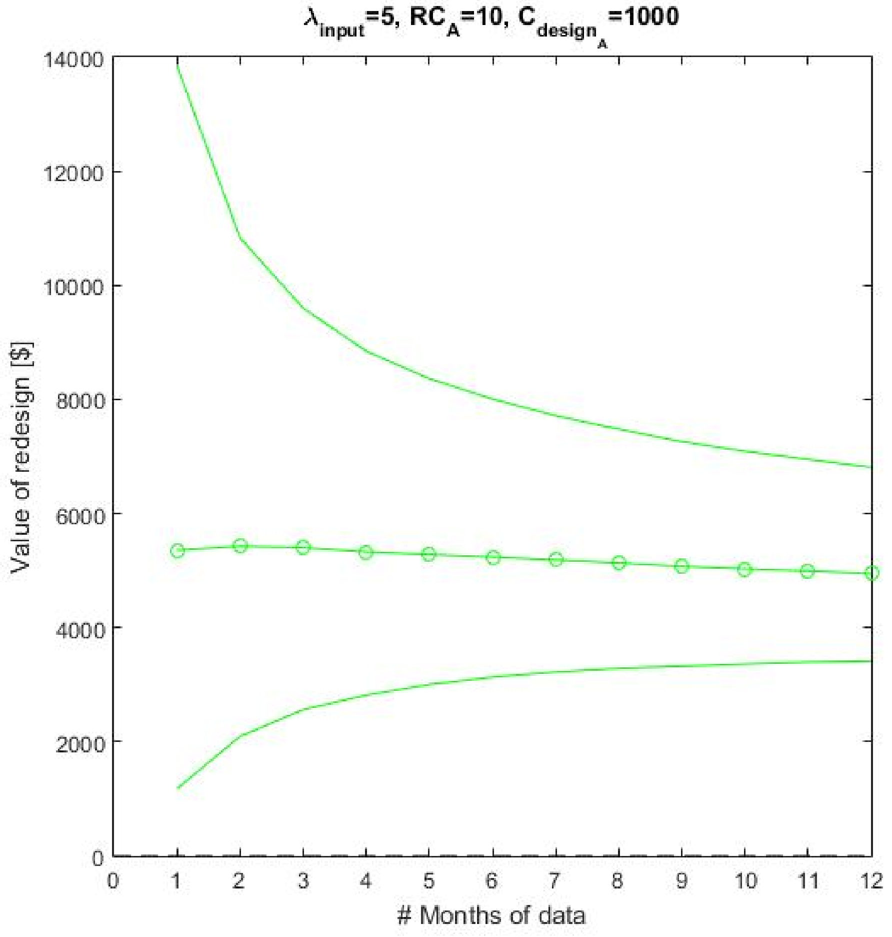

Figure 13: Value of redesign for a 15-year period horizon

To summarize this first part of the section, we can say that it gives us an idea of how dynamic the decision process is. There is a high effect of time and costs over our decisions. It also proves that redesign loses its value, as the system is closer to the end of its operating life. Meaning by this, that it may or may not be better to redesign the system at the very beginning but the farther we go into the future, the less value the redesign has, as the system is closer to the end of its operating life, and investing in a new design may not compensate the revenues generated. On the contrary, it has been proved that redesigning the system always removes uncertainty as our predictions are based only in past data.

The next part of this section focuses on making decisions. It also defines the bound failure rate values that lead to a straightforward decision. We provide the reader with the tools needed to make a decision at each point in time, basing the whole study in the parameters defined in table 8.

Figure 14 gives an example of how a decision changes depending our conditions (failure rate estimated and costs). It shows a graph with two lines, each of them corresponding to a different failure rate. The three points represented (A, B, and C) are pair of costs that we may have in this hypothetical example. Each line divides the area in twice. The area above the line defines those cases where repairing is the best option. On the contrary, the area below, defines those cases where redesigning is the more efficient decision. Figure 14 shows how, depending on our estimated failure rate, the best decision can vary from one to another. Table 9 shows these decisions.

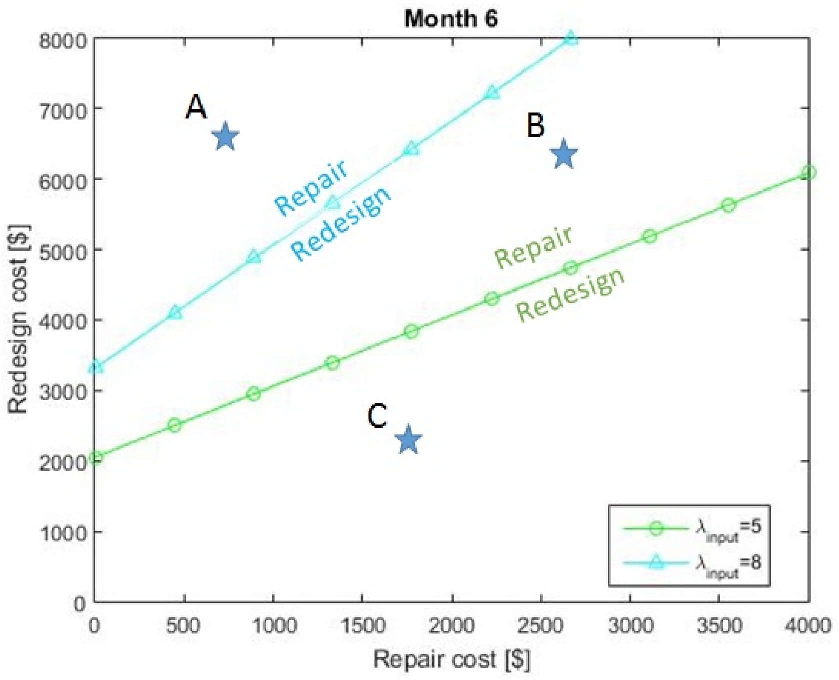

Figure 14: Decision plots

If it happens that our failure rate is 8 events per month, and our costs correspond to pair A, the best solution is repair, while for B and C is redesign, as higher failure rates generate greater costs and, the sooner we get rid of them, the better will go in terms of profits. If, on the contrary, we have a smaller failure rate, 5 events per month, pairs A and B turn to be repairing solutions while C stays as a redesigning one.

Table 9: Example of decisions

Cost	$\lambda_{input} = 8$	$\lambda_{input} = 5$
A	Repair	Repair
B	Redesign	Repair
C	Redesign	Redesign

Now, we wonder if there exists an upper/ lower limit that makes our decision always to be either redesign or repair, and if these limits are invariables over time or if they change accordingly.

After research, we have found the limit failure rates for these conditions (Table 8) to variate with time. The values of these limit failure rates are shown in Table 10 and figure 14. Once these limits (failure rates) have been surpassed, the solution is always either repair or redesign.

Table 10: Limit failure rates and their decisions

Month	Lower bound	Upper bound
1	2.9	7.3
2	2.1	9.9
3	1.8	11.2
4	1.6	12.1
5	1.5	12.9
6	1.4	13.6
7	1.4	14.5
8	1.4	15.2
9	1.3	15.9
10	1.3	16.5
11	1.2	17.1
12	1.2	17.8

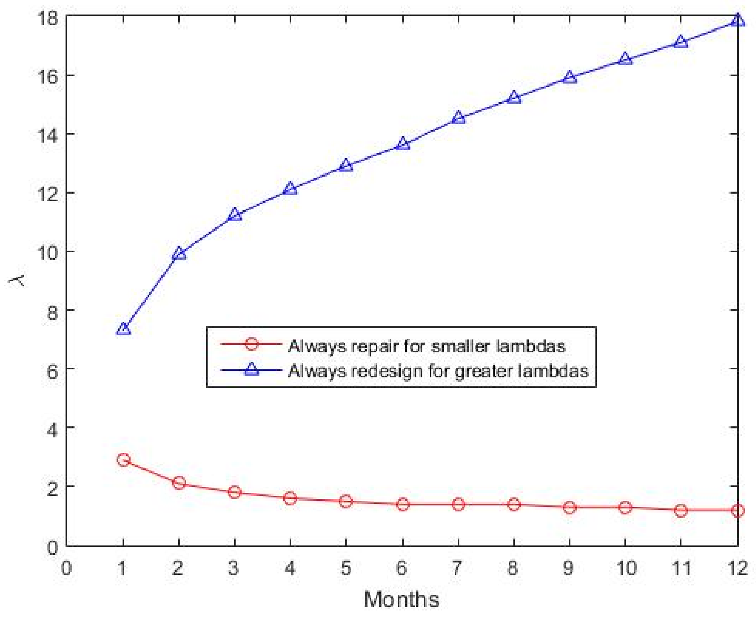

Figure 15: Limit failure rates and their decisions

In next figures, from Figure 16 to 27, we represent the dynamism that the decision process has with time. In these figures, three main lines appear. One of the lines represented corresponds to a generic failure rate chosen to be 5. The two others belong to the previous values to the limiting ones that still show that the best decision depends on lambda and the costs.

As previously commented, the blue lines represent the higher failure rate that give us two possible solutions, being this one, the previous to the upper bound (e.g. upper bound 17.1, blue line represents 17). On the contrary, the red line represents the lower failure rate that give us 2 possible solutions, being this one, the next value to the lower bound (e.g., lower bound 1.3, red line 1.4). As observed in previous figures (Figure 15-26), the farther we go in time, the wider these limits become, meaning that a bigger range of failure rates are within bounds, and the decision for them will be either redesign or repair .

Moreover, the farther we go in time, the lower the slope becomes. Meaning that there are more chances for the decision to be repair. The farther we go in time, the closer we are to the end of the operating life, and thus, less cost combinations state “redesign” as the optimal solution.

To end, in the last months (figures from 23 to 27) the boundaries are so close to the limits that they become almost horizontal (for the lower bound), and vertical (for the upper bound).

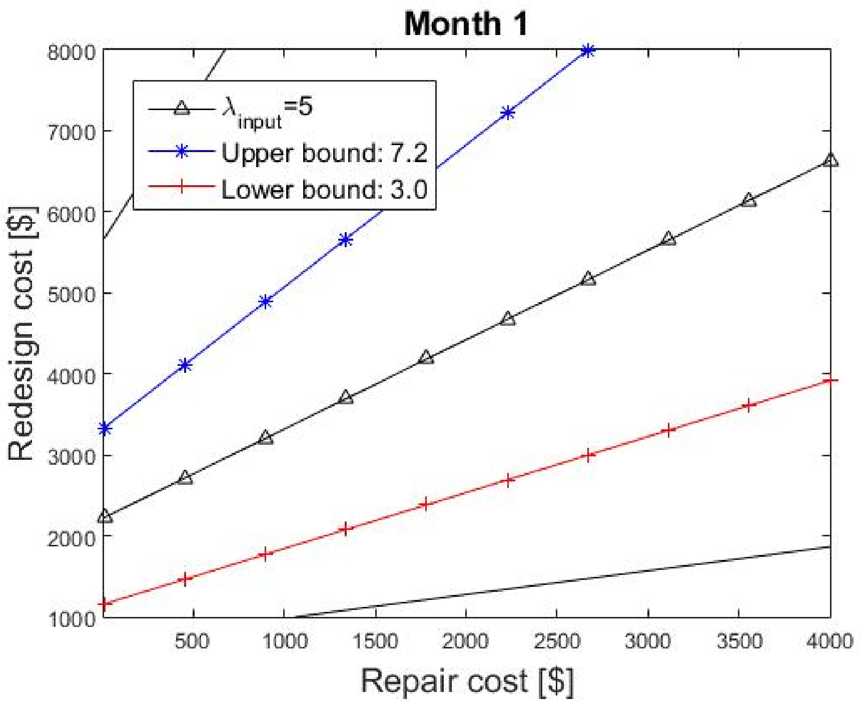

Figure 16: Decision evolution 1

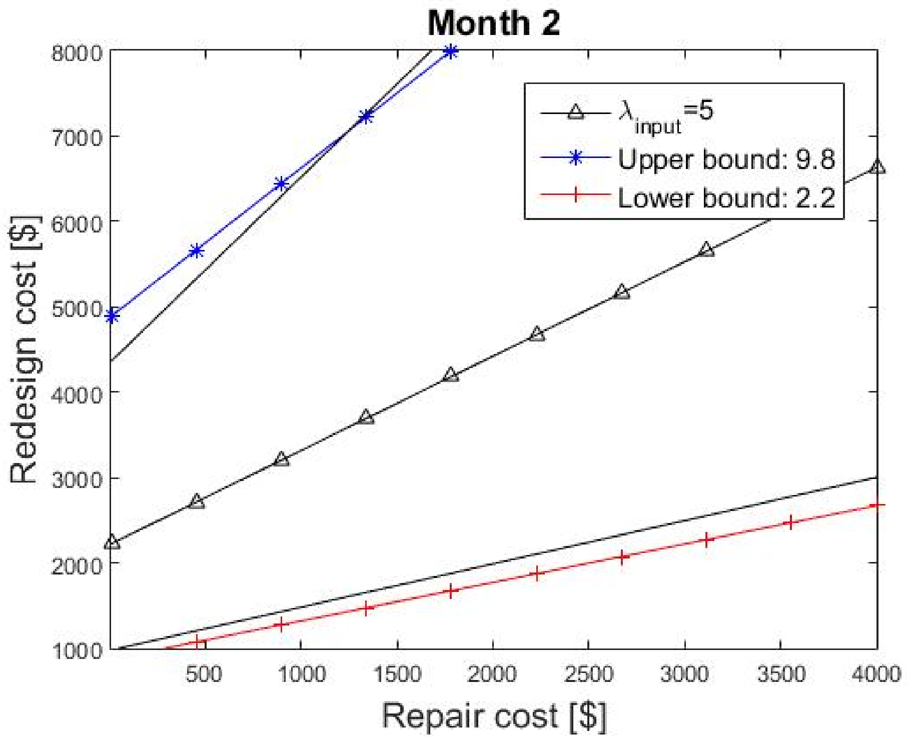

Figure 17: Decision evolution 2

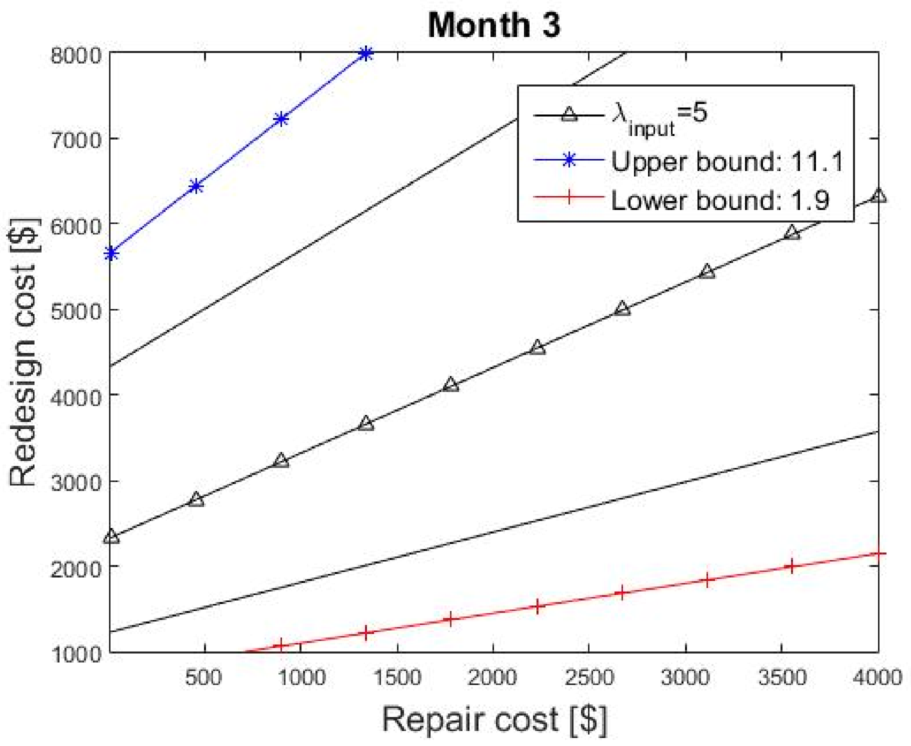

Figure 18: Decision evolution 3

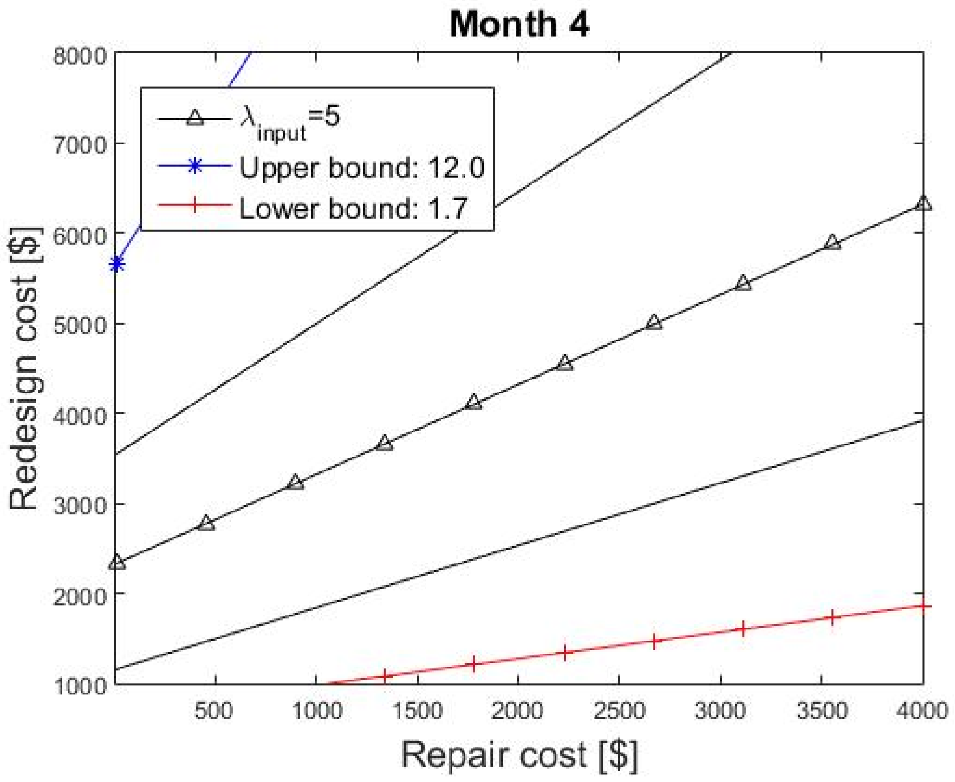

Figure 19: Decision evolution 4

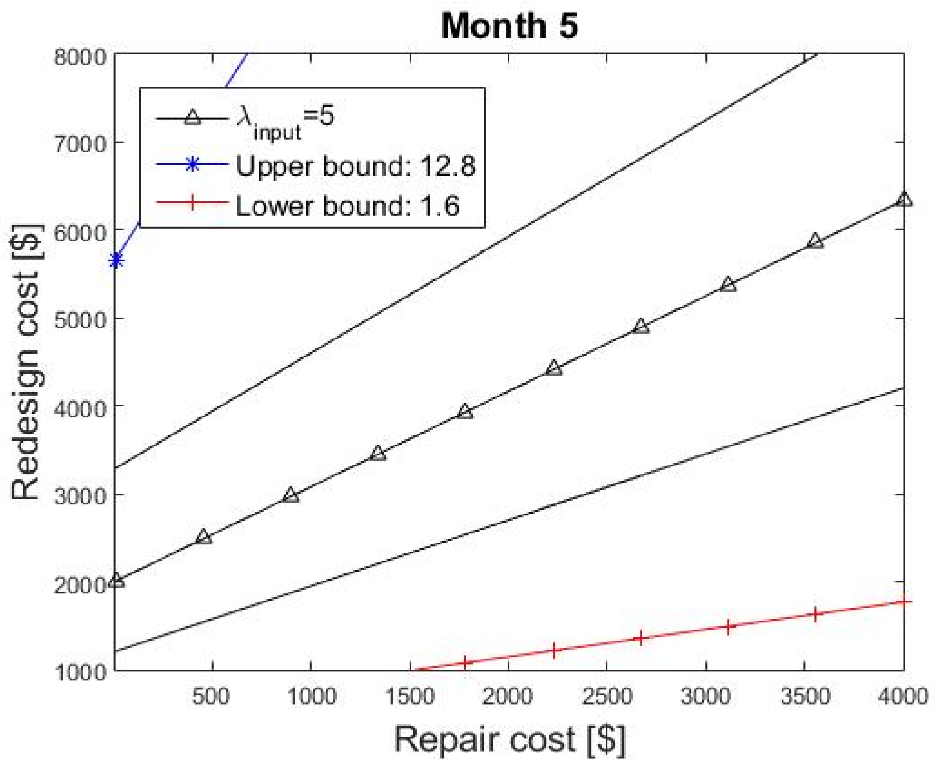

Figure 20: Decision evolution 5

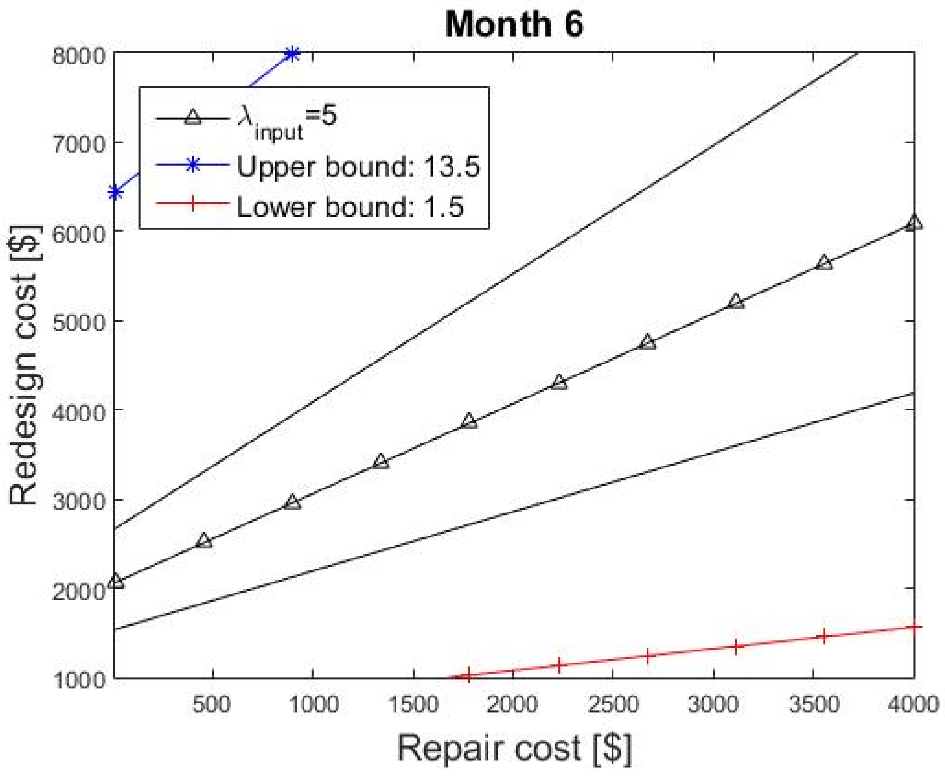

Figure 21: Decision evolution 6

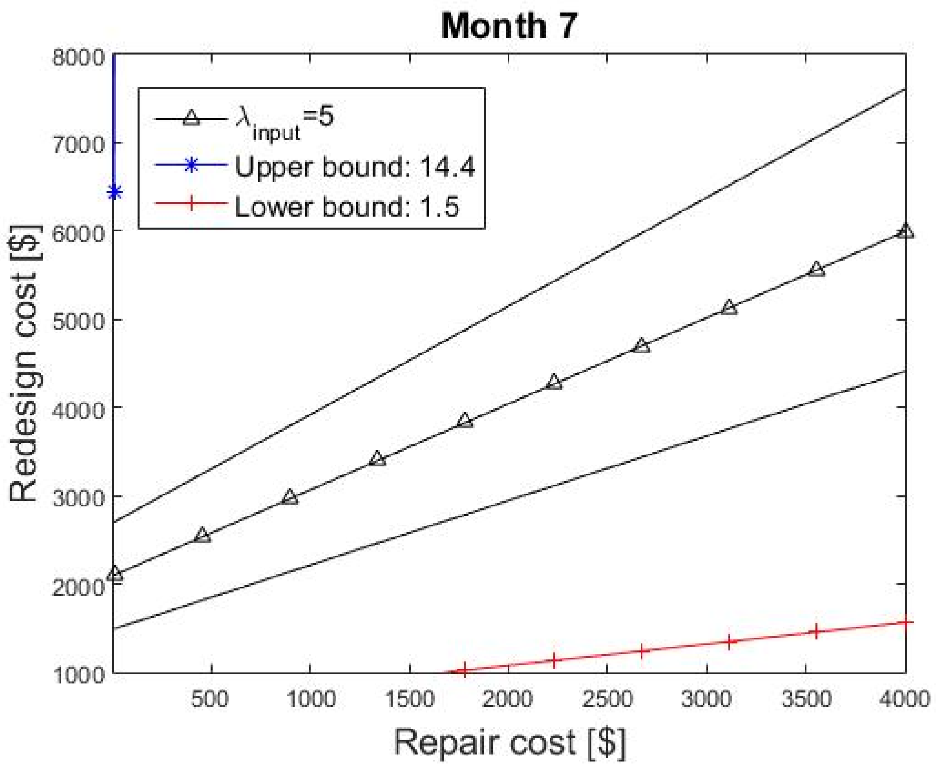

Figure 22: Decision evolution 7

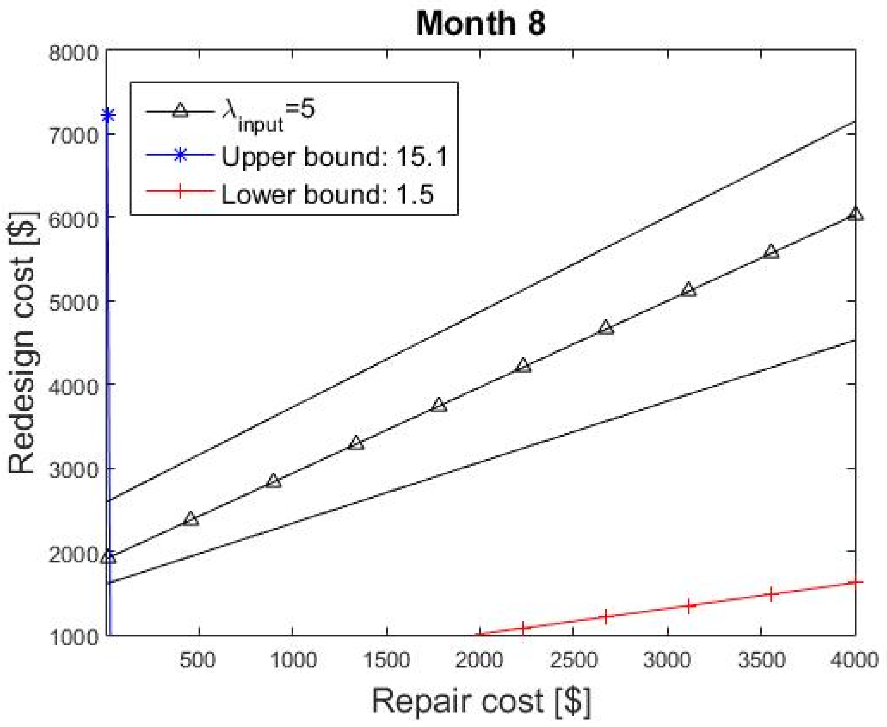

Figure 23: Decision evolution 8

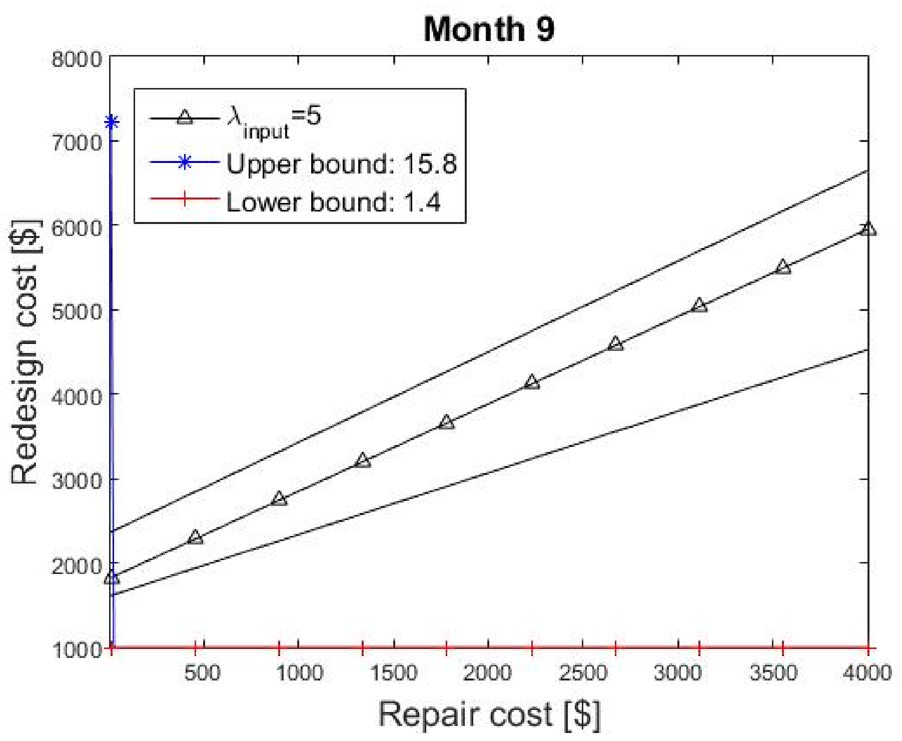

Figure 24: Decision evolution 9

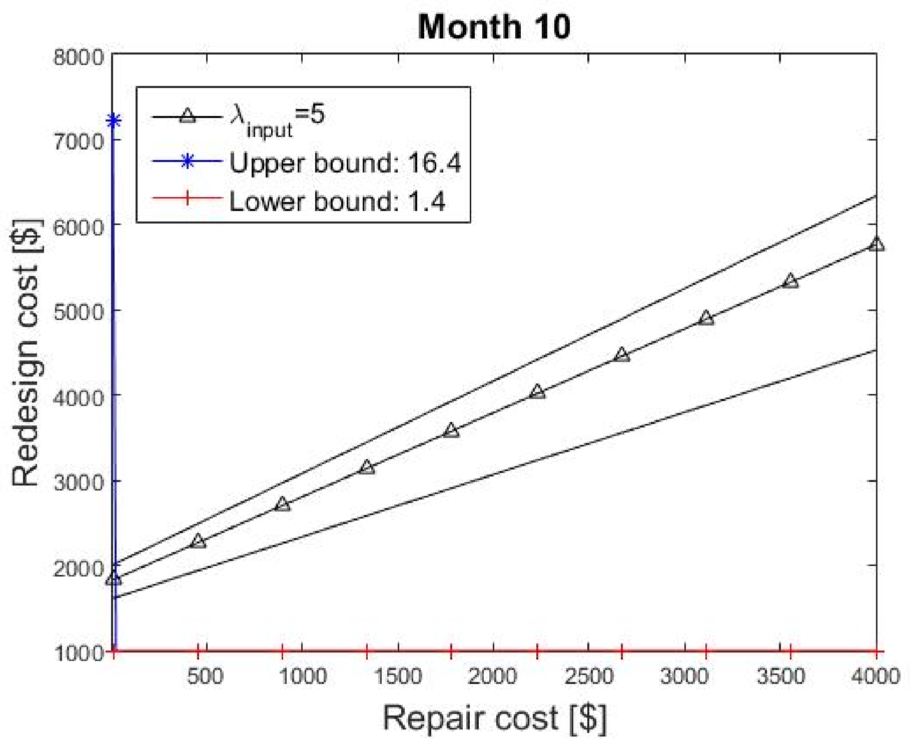

Figure 25: Decision evolution 10

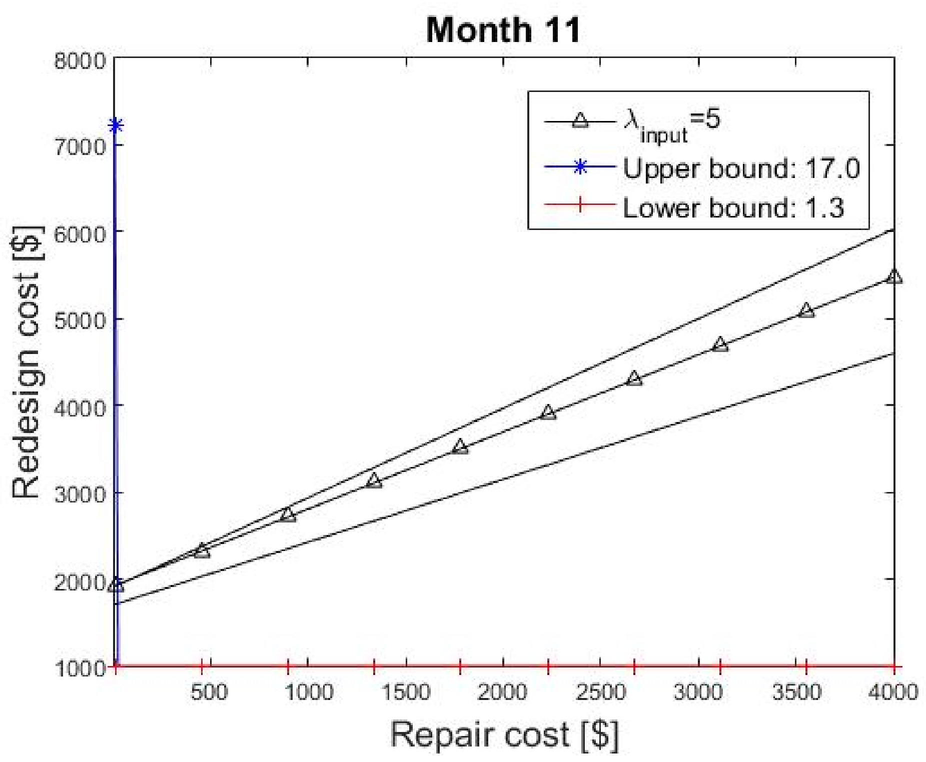

Figure 26: Decision evolution 11

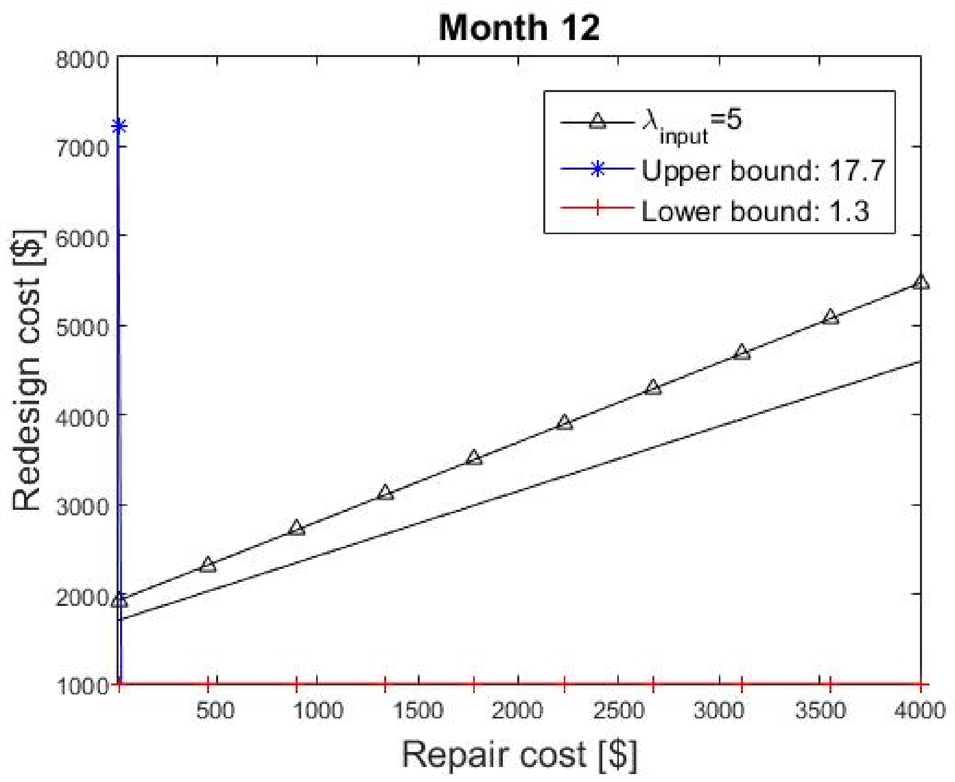

Figure 27: Decision evolution 12

3 Conclusions

While failure resolution decision-making techniques are common in the literature, most of these models focus on minimizing costs or maximizing system availability, but they seldom consider the “value” of the system. We considered “value” as the net revenue generated by the system over a given planning horizon.

Our first concern was the study of NFFs as we found difficult to obtain clear information about them. We made several simplifying assumptions in order to advance our main argument. In future work these assumptions will be validated or relaxed. The work can be used as the basis of a failure resolution decision-making process for other types of failure, where different distribution processes may appear.

One important implication of this work is that the decision-making should be tied to the expected utility profile (system’s operating life) and the market conditions (costs of repair, investigate roots, and redesign the system). In other words, an optimal decision process is dynamic.

Finally, we believe that the framework presented here offers rich possibilities for future work in failure resolution decision-making process.

References

Aircraft Accident Investigation Branch (AAIB) (2012)
http://www.skybrary.aero/bookshelf/books/1853.pdf [accessed 04.30.17]

Bajenescu, T. and Bazu, M. (2010) Component Reliability for Electronic Systems, Artech House, Boston and London.

Beniaminy, I., Joseph,D., (2002) Reducing the "No Fault Found" problem: Contributions from expert-system methods. Aerospace Conference Proceedings, IEEE 6, 2971–2973.

Braaksma, A.J.J., Klingenberg, W., Veldman, J., (2013). Failure mode and effect analysis in asset maintenance: a multiple case study in the process industry. Int. J. Prod. Res. 51 (4), 1055–1071.

Dawn A. Thomasa, Ken Ayersb, Michael Pechtc, April–May 2002. Microelectronics Reliability, Volume 42, Issues 4–5, Pages 641–651. The “trouble not identified” phenomenon in automotive electronics.

De Wolf, I. (2006). Reliability issues in MEMS: physics of failure and design for reliability. Patent DfMM & Memunity Workshop, Milan, Italy, November 27.John Ahmet Erkoyuncu, Samir Khan, Syed Mohammed Fazal Hussain, Rajkumar Roy (2015). Int. J. Production Economics. A frame work to estimate the cost of No-Fault Found events.

Engineering Statistics Handbook, NIST/SEMATECH 2003

Fred K. Geitner and Heinz P. Bloch, (2012). Machinery Failure Analysis and Troubleshooting (Fourth Edition), Chapter 1

Friedel, Evelyn. Schriften zu Marketing und Handel (2014). Price Elasticity: Research on Magnitude and Determinants. Frankfurt am Main, DE: Peter Lang GmbH, Internationaler Verlag der Wissenschaften.

Hockley, C.J., Lacey, L., (2015). No fault found and air safety. In: Proceedings of the 9th WCEAM Research Papers. Springer International Publishing, pp.165–174.

Ishikawa K. (1985) What is total quality control? The Japanese way. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Joseph Homer Saleh, Jean-Francois Castet, (2011). Spacecraft reliability and multi-state failures: A statistical approach. Georgia Institute of Technology

Karen B. Marais (2013). Value maximizing maintenance policies under general repair. Reliability engineering and System safety 119, 76-87.

Khan, S., Phillips, P., Hockley, C., Jennions, I., (2015). No Fault Found: the search for the root cause. SAE Int., ISBN: 978-0-7680-8122-0

Marius I. Bazu and Titu-Marius I. Bajenescu, (2011). Failure Analysis: A Practical Guide for Manufacturers of Electronic Components and Systems, First Edition

Morris R. Driels and Young S. Shin (2004). Determining the number of iterations for Monte Carlo simulations of weapon effectiveness.

O'Connor, Patrick D. T. (2000) Commentary: Reliability—Past, Present, and Future, IEEE TRANSACTIONS ON RELIABILITY, VOL. 49, NO. 4, DECEMBER

Pecht, M. and Ramappan, V. (1992) Are components still the major problem: a review of electronic system and device field failure returns. *IEEE Trans. CHMT*, **15** (6), 1160–1164.

Söderholm, Peter (2006). A system view of the No Fault Found (NFF) phenomenon

Qi, H. et al. (2008). No-fault-found and intermittent failures in electronic products. Microelectron. Reliab., 48, 662–674.

Sorensen B. (2003) Digital averaging – the smoking gun behind 'No-Fault-Found', air safety week, February 24. Vol.17(8)

Strawbridge, Z. et al. (2002) A computational approach to conceptual design. Proceedings of the 2002 ASME Design Engineering Technical Conference, Design Theory and Methodology Conference, Montreal, Canada, DETC02/DTM-34001.

Uday Kumar, Alireza Ahmadi, Ajit Kuma Verma, Prabhakar Varde (2016). Current Trends in Reliability, Availability, Maintainability and Safety, Lecture Notes in Mechanical Engineering

Verma AK, Srividya A, Rana A (2011). Approximation of MTTF calculation of a non-stationary gamma wear process. Int J Syst Assur Eng Manag 2:282–285.

Werner, Debra (2015). A maddening, costly problem. Aerospace America, February 2015, pp. 28–33.

Williams, R., Banner, J., Knowles, I., Dube, M., Natishan, M., Pecht, M., (1998). An investigation of 'cannot duplicate' failures. Qual. Reliab. Eng.Int. 14 (5), 331–337.

Morris R. Driels, Young S. Shin (2004). Determining the number of iterations for Monte Carlo simulations of weapon effectiveness, April.

APPENDIX I: MATLAB CODE INSTRUCTIONS

This document explains the procedure to use the matlab code. Let us note that the code itself is also explained in each .m file. Next figure (Figure 28) shows the structure we need to follow when we want to calculate each study, then figure 29 shows the .m files that correspond to each box of figure 28. That way, if we are interested in just a part of the calculus process, we can directly go to the .m file we are interested in .

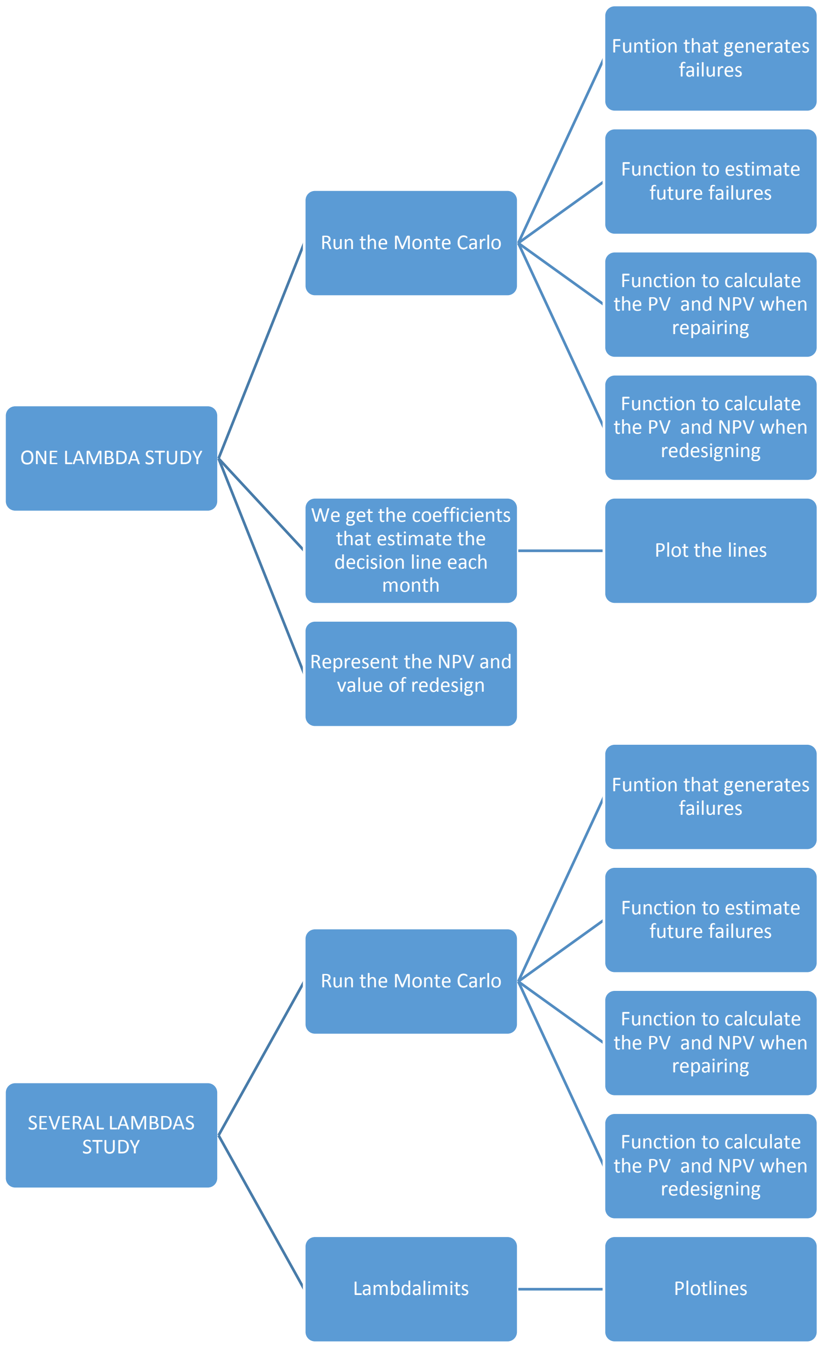
ONE LAMBDA STUDY
Run the Monte Carlo
Funtion that generates failures
Function to estimate future failures
Function to calculate the PV and NPV when repairing
Function to calculate the PV and NPV when redesigning
We get the coefficients that estimate the decision line each month
Plot the lines
Represent the NPV and value of redesign
SEVERAL LAMBDAS STUDY
Run the Monte Carlo
Funtion that generates failures
Function to estimate future failures
Function to calculate the PV and NPV when repairing
Function to calculate the PV and NPV when redesigning
Lambdalimits
Plotlines

Figure 28 Procedure followed

The files corresponding to each box are the following:

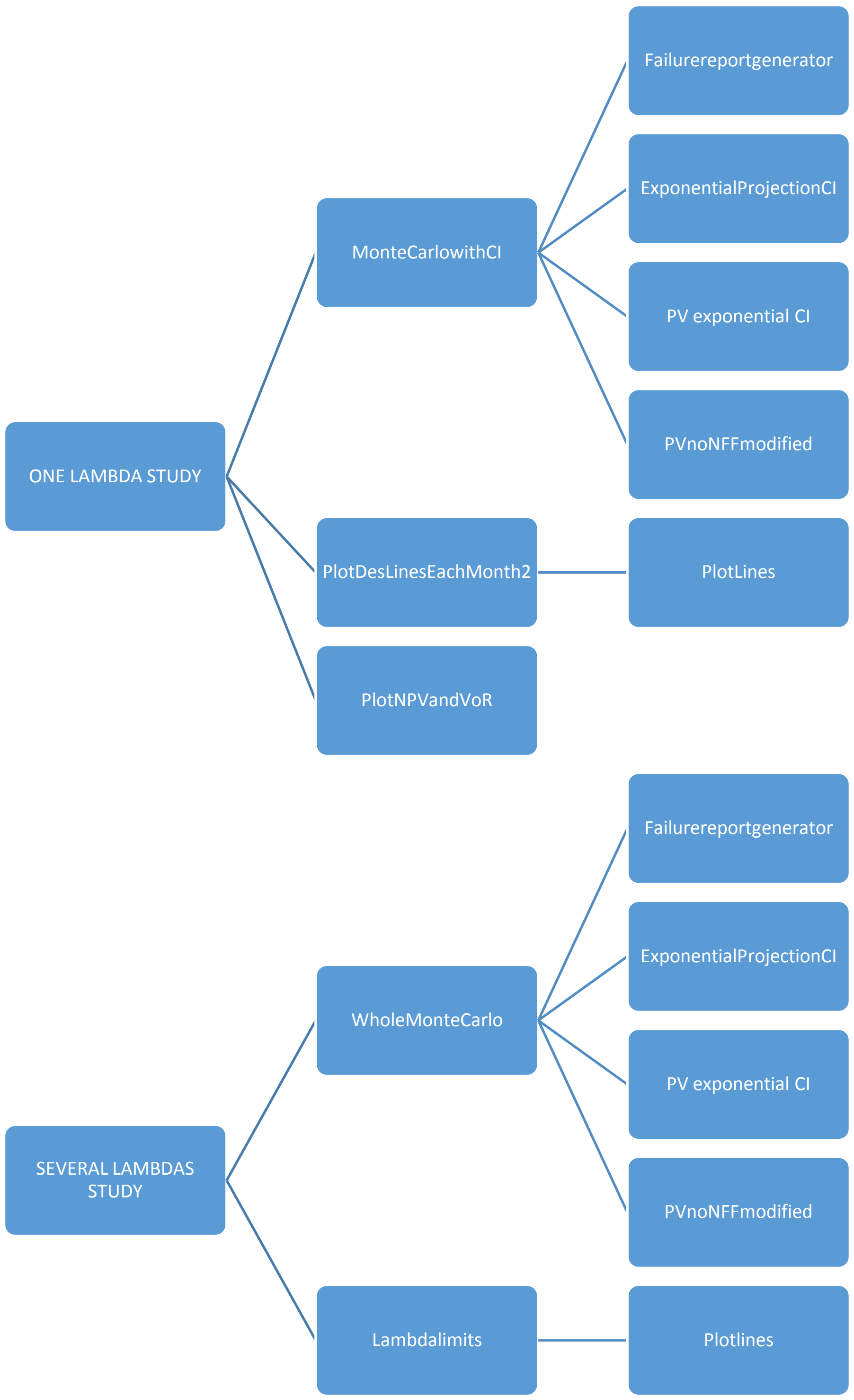

Figure 29 Corresponding .m files

1 One Lambda Study

This section is only useful if we want to run a Monte Carlo for a single value of failure rate, if that is not your interest and you want to run the Monte Carlo for multiple failure rates, then skip this section and go to chapter 2.

Here there are the 4 basic steps we need to follow in order to run our study properly:

1. Open the MonteCarlowithCI
   a. Input the data that can be modified (it says input) with the values we want to
   b. Set a name for our study in the end of the document
   c. Run the file
2. Open the MeanResultsfrom MonteCarlo
   a. Write the name of our study
   b. Set the costs that we want to graph (it says inputs)
   c. Run the file
3. Open PlotDesLinesEachMonth2
   a. Input the name of our study
   b. Set a name for our coefficient matrix in the end of the document
   c. Run the file
4. Open PlotLines
   a. Set the number of the month we want to plot
   b. Input the name of the files previously generated in 3. For different lambdas
   c. Run the file

The next subsections explain what each file does at each moment, in a more detailed way. Remember that everything is also explained in the .m files themselves.

1.1 File: MonteCarlowithCI

This file runs the Monte Carlo study by running each function a number of times that we define as an input of the variable called “iterations”. Once all has been calculated, it is saved in a .mat file to be used by the MeanResultsfromMontecarlo.m file.

Calculation process:

1. Nomenclature
2. Input data: These are all the variables we need to define by given them a single value or a range of values.
3. Main code:
   a. First we generate the sequences of failures detected
   b. Failure report generator displays the information needed to represent the data reported (failures detected over time).

c. Prediction of failures following Exponential distribution (HPP):
Comments: The failures_exp is the total number of failures accumulated per period. lambdas_exp is the counter of failures we have per month
d. Estimation NPV for Exponential distribution: For each MRC there is a redesign cost associated. Here we calculate all those combinations of costs, to get the NPV and the PVs.
e. Cost of eliminating all NFFs (cost of redesign)

4. Save the data.

```
%% NOMENCLATURE
% random_sample  gives us a vector of failures detected following a normal
distribution with lambda as an input
% n              MONTHS OF AVAILABLE DATA
% av             is the mean of the failures detected defined by the
random_sample
% t              is a vector that counts the months of data
% acc_l          is a vector with the accumulated number of failures
% months         MONTHS OF PROYECTED DATA
% alpha          confidence coefficient to calculate the confidence intervals
% t_year         vector that counts all months of projection
% mrt_nff        MEAN REPAIR TIME OF NFF, (-/MONTH)
% mrc_nff        MEAN REPAIR COST OF NFF, IN $/MONTH
% mrt_b          MEAN REPAIR TIME OF FAILURE TYPE B,(-/MONTH)
% mrc_b          MEAN REPAIR COST OF FAILURE TYPE B,($/MONTH)
% oper_cost      OPERATING COSTS, IT IS A FIXED QUANTITY, $/MONTH
% remov_cost     COST OF REDESIGN $/MONTH
% hard_cost_nff  COST OF HARDWARE WHEN NFFS HAPPEN $/FAILURE
% hard_cost_b    COST OF HARDWARE WHEN FAILURE TYPE B HAPPEN $/FAILURE
% u              NET REVENUES OF THE SYSTEM (IF NO FAILURES) PER MONTH
% r_At           MONTHLY DISCOUNT RATE
% lambda         FAILURE RATE OF TYPE A (NFFs)
% lambda_b       FAILURE RATE OF TYPE B,ESTIMATED TRIPLE OF lambda
% pred_year      YEARS OF PROYECTED DATA
% iterations     RUNS OF OUR STUDY


%% INPUT DATA
lambda=5;          % INPUT lambda mean per month
alpha= 0.05;         % COnfident interval of 95%
n=12;                % INPUT months of data
pred_year=5;         % years of estimated NPV data
mrt_nff = 3/(24*30);            % Invented value 3 h to repair a failure
expressed as a fraction of a month
values=10;
mrc_nff = linspace(0,0,values);                  % Invented value $/month
remov_cost = linspace(1000,8000,values);                  % Invented value
$/month
oper_cost = 50;              % I will assume to have a 60$ cost of utilities
per month
hard_cost_nff = 10;              % $/failure cost of hardware
hard_cost_b = 20;
u = 1500;                    %Benefits that each unit reports per month when
working
```

```
r_At = 0.05/12;                    % Discount rate of 5% per year, meaning
0.05/12 per month
lambda_b=round(lambda*3);
%lambda_b=15;
mrt_b = 6/(24*30);             % Invented value 3 h to repair a failure
expressed as a fraction of a month
mrc_b = 50;
iterations=1;

%% Define Montecarlo Study
months = pred_year*12;          % months of projected data
years = months/12;               % years of estimated NPV data
[p,q] = meshgrid(mrc_nff, remov_cost);
repair_remove_pairs = [p(:) q(:)];
[xcol,ycol]=size(repair_remove_pairs);
% First define the empty matrixes
A=[];
AU=[];
AL=[];
C=[];
CU=[];
CL=[];
E=[];
EU=[];
EL=[];
G=[];
GU=[];
GL=[];
I=[];
J=[];
NPVexpbig=[];
NPVexpbig_U=[];
NPVexpbig_L=[];

NPV_noNFFbig=[];

Costexpbig=[];
Costexpbig_U=[];
Costexpbig_L=[];

for l=1:iterations
%% Random scenarios following a Poisson model
random_sample = poissrnd(lambda,1,n);
lambda_seq = transpose(random_sample);
lambda_ci = zeros(2,n);
t=(1:1:n);
% We generate the confidence intervals
    for q = 1:n
        [le(q), lci(:, q)] = poissfit(lambda_seq(1:q));
    end
lambdas_exp=random_sample;
lambdas_exp_upper=[lci(2,1),diff(lci(2,:).*t)];
lambdas_exp_low=[lci(1,1),diff(lci(1,:).*t)];

acc_l = cumsum(random_sample);
av = acc_l./t;
```

```
av_upper = lci(2, :);
av_low = lci(1, :);

% Representation of the data reported
%[l_sample, t_report,l_graph,t_graph] =
FailureReportGenerator1(t,random_sample);

%% Projection following Exponential distribution (HPP)
[data_t,failures_exp,lambdas_exp] =
ExponentialProjectionCI(av,t,months,random_sample);
[data_t,failures_exp_upper,lambdas_exp_upper] =
ExponentialProjectionCI(av_upper,t,months,lambdas_exp_upper);
[data_t,failures_exp_low,lambdas_exp_low] =
ExponentialProjectionCI(av_low,t,months,lambdas_exp_low);
%Comments
% The failures_exp is the total number of failures accumulated per period
% lambdas_exp is the counter of failures we have per month

%% Estimation NPV for Exponential distribution
for p=1:xcol
    f_cost=repair_remove_pairs(p,1); %Cost of repair
    f_redesign=repair_remove_pairs(p,2); % Cost of redesign
    %% NPV when just repairing
    [NPV_year,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp,lambda_b,months,n,pred_year);
    [NPV_year_L,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp_low,lambda_b,months,n,pred_year);
    [NPV_year_U,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp_upper,lambda_b,months,n,pred_year);

    %% Cost of eliminating all NFFs
    [NPV_noNFF_year,counter_l2] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,random_sample,months,n,pred_year);
    %[NPV_noNFF_year_L,counter_l2_L] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,lambdas_exp_low,months,n,pred_year);
    %[NPV_noNFF_year_U,counter_l2_U] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,lambdas_exp_upper,months,n,pred_year);

    % Cost if assumed a HPP model
    Costexp=NPV_noNFF_year-NPV_year;
    Costexp_U=NPV_noNFF_year-NPV_year_U;
    Costexp_L=NPV_noNFF_year-NPV_year_L;

    %% Matrixes for each cost 3D
    NPVexpbig(l,:,p)=NPV_year;
    NPVexpbig_U(l,:,p)=NPV_year_U;
    NPVexpbig_L(l,:,p)=NPV_year_L;

    NPV_noNFFbig(l,:,p)=NPV_noNFF_year;
    %NPV_noNFFbig_U(l,:,p)=NPV_noNFF_year_U;
    %NPV_noNFFbig_L(l,:,p)=NPV_noNFF_year_L;

    Costexpbig(l,:,p)=Costexp;
    Costexpbig_U(l,:,p)=Costexp_U;
    Costexpbig_L(l,:,p)=Costexp_L;

    %%Comments:
    %x axis: number of cases
    % y axis: # months of data available
    % z axis: diff costs studied
end

%% Saving each test
% NPV of all the cases
% Matrixes of the exponential distribution:
% A=[A;NPVexp];
% AU=[AU;NPVexp_U];
% AL=[AL;NPVexp_L];
% Failures estimation with exponential distribution:
C(:,:,l)=lambdas_exp;
CU(:,:,l)=lambdas_exp_upper;
CL(:,:,l)=lambdas_exp_low;
% % PV estimation with exponential distribution:
% % E(:,:,l)=PVexp;
% % EU(:,:,l)=PVexp_U;
% % EL(:,:,l)=PVexp_L;
% %Cost estimation with exponential distribution:
% G=[G;Costexp];
% GU=[GU;Costexp_U];
% GL=[GL;Costexp_L];
% Average of failures
I=[I;av];
% random sample
J=[J;random_sample];

end
% m= num2str(n);
% p= num2str(iterations);
Time=data_t;

save('lamb5_0cost5yrs.mat')
toc
```

1.1.1 Function FailureReportGenerator1

This function is just useful in terms of representing a specific scenario where l_graph, t_graph show over time the failures detected at each month. As we run a Monte Carlo this data is not saved but the function is still here in case we want to apply it to represent a specific case.

```
function [l_sample, t_report,l_graph,t_graph,random_sample,l_samp] =
FailureReportGenerator1(t,random_sample)
```

```
%% SPECIFIC ASSUMPTION:
% FOR THE SAKE OF THIS PROJECT WE ASSUME THAT IF , FOR EXAMPLE, 3 FAILURES
% ARE FOUND IN A MONTH, WE ASSUME TO HAVE OCCURED EQUALLY DISTRIBUTED OVER
% TIME, BEING CALCULATED THE frec to estimate the time at which they were
% reported ( e.g. 0.33,0.66,1)

t_report=[];
l_samp=[];
l_sample=[];

for i=1:length(t)
    if random_sample(i)==0
        t_reported=t(i);
        l_sample1 = 0;
    elseif random_sample(i)==1
        t_reported=t(i);
        l_sample1 = 1;
    else
        frec= 1/random_sample(i);
        l_sample1 = ones(1,random_sample(i));
        multiplier = (1:1:random_sample(i));
        if i==1
            t_reported = frec*multiplier;
        else
            t_reported = t(i-1)+frec*multiplier;
        end
    end
    l_samp=[l_samp,l_sample1];
    t_report=[t_report, t_reported];
end
l_sample=transpose(cumsum(transpose(l_samp)));
l_graph=l_sample;
t_graph=t_report;
l_graph(l_samp == 0 ) = [];
t_graph(l_samp == 0 ) = [];

end
```

1.1.2 Function ExponentialProjectionCI

This function generates the matrix will all the failures (detected and predicted) according to the months of data available, and puts them all together in a matrix, so we can use it for NPV calculation. In case we would like to represent the data, we should use matrixes data_t and failures_exp (accumulated failures over time).

```
function [data_t,failures_exp,lambdas_exp] =
ExponentialProjectionCI(av,t,months,random_sample)

%% NOMENCLATURE %%
% random_sample gives us a vector of failures detected following a normal
distribution with lambda as an input
% n            ·months of data available
% av              is the mean of the failures detected defined by the
random_sample
% t               is a vector that counts the months of data
% acc_l           is a vector with the accumulated number of failures
% months          # of months that we want to proyect our results, eg.24 months
% alpha           confidence coefficient to calculate the confidence intervals
% t_year          vector that counts all months of projection
% lambdas_exp     matrix with n rows and months columns. each row's position
% defines how many data we have , and it continues with the predicted value

t_year = linspace(1,months,months);
data_t =[];
% Matrix of failures reported
for i=1:length(t)
    lambdas_exp(i,1:i)=random_sample(1:i);
    lambdas_exp(i,(i+1):months)=av(i);
    data_t =[data_t;t_year];
end
% Matrix for graphs ( it is just the accumulated value of failures over
% time) so we can graph the points if wanted
failures_exp=transpose(cumsum(transpose(lambdas_exp)));
end
```

1.1.3 Function PVExponentialCI

This function estimates the NPV when we just repair the failures detected each month.

```
function [NPV_year,t_year,PV] =
PVexponentialCI(mrt_nff,mrc_nff,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost
_b,u,r_At,lambdas_exp,lambda_b,months,n,pred_year)

%% NOMENCLATURE:
% mrt_nff         MEAN REPAIR TIME OF NFF, IS A FRACTION OF A MONTH
% mrc_nff         MEAN REPAIR COST OF NFF, IN $/MONTH
% mrt_b           MEAN REPAIR TIME OF FAILURE TYPE B, IS A FRACTION OF A MONTH
% mrc_b           MEAN REPAIR COST OF FAILURE TYPE B, IS A FRACTION OF A MONTH
% oper_cost       OPERATING COSTS, IT IS A FIXED QUANTITY, $/MONTH
% hard_cost_nff   COST OF HARDWARE WHEN NFFS HAPPEN $/FAILURE
% hard_cost_b     COST OF HARDWARE WHEN FAILURE TYPE B HAPPEN $/FAILURE
% u               NET REVENUES OF THE SYSTEM (IF NO FAILURES) PER MONTH
% r_At            MONTHLY DISCOUNT RATE
% lambdas_exp     MATRIX WITH FAILURES OF TYPE A (NFFs)
% lambda_b        FAILURE RATE OF TYPE B
% months          MONTHS OF PROYECTED DATA
% n               MONTHS OF AVAILABLE DATA
% pred_year       YEARS OF PROYECTED DATA

t_year = linspace(1,months,months);
PV_all=[];
NPV_all=[];
for i=1:n
    PV(i,:)=((u-lambdas_exp(i,:).*mrt_nff*u/30-lambda_b*mrt_b*u/30)-
(mrt_nff*mrc_nff.*lambdas_exp(i,:)+mrt_b*mrc_b.*lambda_b+oper_cost+hard_cost_
nff.*lambdas_exp(i,:)+hard_cost_b*lambda_b))./((1+r_At).^t_year);
```

```
end
NPV=transpose(cumsum(transpose(PV)));
NPV_year=transpose(NPV(:,pred_year*12));

end
```

1.1.4 Function PVnoNFFmodified

This function estimates the NPV when we repair the existing failures and we invest in a new design solving future failures of the type that we are interested in (Type A /NFFs).

Calculation process:

1. PV of repairing for the past months of data
2. PV of eliminating future failures and repairing the previous failures
3. PV Once we have paid for the new system and we only have failures of type B

```
function [NPV_noNFF_year,counter_l2] =
PVnoNFFmodified(mrt_nff,mrc_nff,mrt_b,mrc_b,remov_cost,oper_cost,hard_cost_nf
f,hard_cost_b,u,r_At,lambda_b,random_sample,months,n,pred_year)

%% NOMENCLATURE:
% mrt_nff        MEAN REPAIR TIME OF NFF, IS A FRACTION OF A MONTH
% mrc_nff        MEAN REPAIR COST OF NFF, IN $/MONTH
% mrt_b          MEAN REPAIR TIME OF FAILURE TYPE B, IS A FRACTION OF A MONTH
% mrc_b          MEAN REPAIR COST OF FAILURE TYPE B, IS A FRACTION OF A MONTH
% oper_cost      OPERATING COSTS, IT IS A FIXED QUANTITY, $/MONTH
% hard_cost_nff  COST OF HARDWARE WHEN NFFS HAPPEN $/FAILURE
% hard_cost_b    COST OF HARDWARE WHEN FAILURE TYPE B HAPPEN $/FAILURE
% u              NET REVENUES OF THE SYSTEM (IF NO FAILURES) PER MONTH
% r_At           MONTHLY DISCOUNT RATE
% lambdas_exp    MATRIX WITH FAILURES OF TYPE A (NFFs)
% lambda_b       FAILURE RATE OF TYPE B
% months         MONTHS OF PROYECTED DATA
% n              MONTHS OF AVAILABLE DATA
% pred_year      YEARS OF PROYECTED DATA
% counter_l      GENERATES A NEW VECTOR WHERE FUTURE FAILURES ( COUNTING
% FROM THE MOMENT WHERE WE ARE, HAVE BEEN REMOVED, BEING THEIR VALUE 0 )
t_year = linspace(1,months,months);
counter_l2=[];
for i=1:n
    counter_l=[random_sample(1:i),zeros(1,(months-i))];
    %PV of repairing
    PV1(i,1:i)=((u-lambda_b*mrt_b*u/30-mrt_nff*u/30.*counter_l(1:i))-
(mrt_nff*mrc_nff.*counter_l(1:i)+mrt_b*mrc_b*lambda_b+oper_cost+hard_cost_nff
.*counter_l(1:i)+hard_cost_b*lambda_b))./((1+r_At).^t_year(1:i));
    % PV of eliminating future failures and repairing the previous failures
        if (i+1)==months
            PV1(i,(i+1))= (((u-lambda_b*mrt_b*u/30)-
(remov_cost+mrt_b*mrc_b*lambda_b+oper_cost+hard_cost_b*lambda_b))./((1+r_At).
^t_year(i+1)));
        elseif (i+2)<=months
            % When we have to pay for the new system
            PV1(i,(i+1))= (((u-lambda_b*mrt_b*u/30)-
(remov_cost+mrt_b*mrc_b*lambda_b+oper_cost+hard_cost_b*lambda_b))./((1+r_At).
^t_year(i+1)));
            % once we have paid for the new system and we only have
            % failures of type B
            PV1(i,(i+2):months)= ((u-lambda_b*mrt_b*u/30)-
(mrt_b*mrc_b*lambda_b+oper_cost+hard_cost_b*lambda_b))./((1+r_At).^t_year((i+
2):months));
        end
    counter_l2=[counter_l2;counter_l];
end
NPVallvalues=transpose((cumsum(transpose(PV1))));
NPV_noNFF_year=transpose(NPVallvalues(:,pred_year*12));
end
```

1.2 File: PlotDesLinesEachMonth2

It gets and saves the coefficients of the linear polynomial that defines the limit line between repair and redesign, according to the months of data available.

```
clc;clear all;

load('lamb10_5yrs.mat')

for i=1:xcol
    % We evaluate the mean over all cases, with the same cost
    Mean_NPV(1,:,i)=mean( NPVexpbig(:,:,i));
    Mean_NPV(2,:,i)=mean( NPVexpbig_U(:,:,i));
    Mean_NPV(3,:,i)=mean( NPVexpbig_L(:,:,i));
    Mean_Cost(1,:,i)=mean(Costexpbig(:,:,i));
    Mean_Cost(2,:,i)=mean(Costexpbig_U(:,:,i));
    Mean_Cost(3,:,i)=mean(Costexpbig_L(:,:,i));
    Mean_noNFF(1,:,i)=mean(NPV_noNFFbig(:,:,i));
end
% TO GET THE BIG MATRIXES:
% MATRIX DECISION = 1 REDESIGN  0 REPAIR
% MATRIX MONTH = SAYS EVERY POSITION
Mean_Cost(Mean_Cost(:,:,:)>0)=1;
Mean_Cost(Mean_Cost(:,:,:)<=0)=0;
Coef_Matrix=[];
decision=[];
for j=1:n
    % FOr each month

    for i=1:values
        %Matrix_Month(i,:)=Month((values*(i-1)+1):(values*i));
        Matrix_Decison1(i,:)=Mean_Cost(1,j,(values*(i-1)+1):(values*i));
        Matrix_Decison2(i,:)=Mean_Cost(2,j,(values*(i-1)+1):(values*i));
        Matrix_Decison3(i,:)=Mean_Cost(3,j,(values*(i-1)+1):(values*i));
    end
BIG_matrix(:,:,j)=Matrix_Decison1(:,:);
BIG_matrix_U(:,:,j)=Matrix_Decison2(:,:);
BIG_matrix_L(:,:,j)=Matrix_Decison3(:,:);
% TO GET THE LINES EACH MONTH:
% Unique give us the different numbers that we have in the matrix in a
% column vector
nonzero1 = nnz(BIG_matrix(:,:,j));
nonzero2 = nnz(BIG_matrix_U(:,:,j));
nonzero3 = nnz(BIG_matrix_L(:,:,j));
    if (nonzero1==0)||(nonzero2==0)||(nonzero3==0)
        coeff1=[0,0];
        coeff2=[0,0];
        coeff3=[0,0];
    elseif (nonzero1==values^2)||(nonzero2==values^2)||(nonzero3==values^2)
        coeff1=[9999999999,0];
        coeff2=[9999999999,0];
        coeff3=[9999999999,0];
    else
        coeff1= GetLineCoef(BIG_matrix(:,:,j),mrc_nff,remov_cost);
        coeff2= GetLineCoef(BIG_matrix_U(:,:,j),mrc_nff,remov_cost);
        coeff3= GetLineCoef(BIG_matrix_L(:,:,j),mrc_nff,remov_cost);
%     elseif (nonzero==0)||(nonzero==values^2)
%         coeff=[0,0,0];
    end
    Coef_Matrix=[Coef_Matrix;coeff1,coeff2,coeff3];

end

save('coeff_lines2_lamb10_5yrs.mat','Coef_Matrix','BIG_matrix','BIG_matrix_L'
,'BIG_matrix_U')
```

1.2.1 Function: GetLineCoef

It generates the coefficients for the decision lines, according to the approximation we choose. We can change the function we want our data to be approximated to modifying this line of code:`c=fit(Xcoord,Ycoord,'poly1');`

```
function coeff= GetLineCoef(matrix,mrc_nff,remov_cost)
[size1,size2]=size(matrix);
X_line=[];
Y_line=[];
x_point=0;
y_point=0;
for i=1:size1
    for j=1:(size2-1)
        if matrix(i,j+1)~=matrix(i,j)
            x_point=mrc_nff(i);
            y_point=remov_cost(j);
        end
        X_line=[X_line,x_point];
        Y_line=[Y_line,y_point];
    end
end
Xcoord=transpose( X_line( X_line(:)~=0));
Ycoord=transpose( Y_line( Y_line(:)~=0));
c=fit(Xcoord,Ycoord,'poly1');
coeff=coeffvalues(c);
end
```

1.3 File: PlotLines

It plots the decision lines for:

1. A generic failure rate that we can change if we want to(Lambda)
2. The two others belong to the previous values to the limiting ones that still show that the best decision depends on lambda and the costs. (Lambda_U, Lambda_L)

We need to input the month of data that we want to represent in the variable plotMonth.

NOTE: the represented values must have been calculated previously. If the value of Lambda chosen is not included in the Coefficients.mat file, then, load the proper file generated in section 1.2.

NOTE2: This code is also valid for our multiple lambda study, as it gets the coefficients from the .mat generated from all the lambdas run.

```
clc;clear all;

%% Plots lines
plotMonth=12;                                   %% CHOOSE THE MONTH TO PLOT

load('Coefficients.mat','Lambda_redesign','Lambda_repair')
Lambda_U=Lambda_redesign(plotMonth)-0.1;
Lambda_L=Lambda_repair(plotMonth)+0.1;

Lambda=5;
%Lambda2=5;
values=10;
mrc_nff = linspace(0,4000,values);                % Invented value $/month
remov_cost = linspace(0,8000,values);          % Invented value $/month
%% Lambda upper LimiT
filenameU=['A_' num2str(Lambda_U*10)];
load(filenameU)
p1=Coef_Matrix(plotMonth,1);
p2=Coef_Matrix(plotMonth,2);
% p1U=Coef_Matrix(plotMonth,3);
% p2U=Coef_Matrix(plotMonth,4);
% p1L=Coef_Matrix(plotMonth,5);
% p2L=Coef_Matrix(plotMonth,6);
function1 = @(x) (p1*x + p2 );
% function1U = @(x) (p1U*x + p2U);
% function1L = @(x) (p1L*x + p2L );
%% Lambda lower limit
filenameL=['A_' num2str(Lambda_L*10)];
load(filenameL)
p1=Coef_Matrix(plotMonth,1);
p2=Coef_Matrix(plotMonth,2);
% p1U=Coef_Matrix(plotMonth,3);
% p2U=Coef_Matrix(plotMonth,4);
% p1L=Coef_Matrix(plotMonth,5);
% p2L=Coef_Matrix(plotMonth,6);
function2 = @(x) (p1*x + p2 );
```

```
% function2U = @(x) (p1U*x + p2U );
% function2L = @(x) (p1L*x + p2L );
%% Lambda value 1
filename=['A_' num2str(Lambda*10)];
load(filename)
p1=Coef_Matrix(plotMonth,1);
p2=Coef_Matrix(plotMonth,2);
p1U=Coef_Matrix(plotMonth,3);
p2U=Coef_Matrix(plotMonth,4);
p1L=Coef_Matrix(plotMonth,5);
p2L=Coef_Matrix(plotMonth,6);
function3 = @(x) (p1*x + p2 );
function3U = @(x) (p1U*x + p2U );
function3L = @(x) (p1L*x + p2L );

%% Lambda value 2
% filename=['A_' num2str(Lambda2*10)];
% load(filename)
% p1=Coef_Matrix(plotMonth,1);
% p2=Coef_Matrix(plotMonth,2);
% p1U=Coef_Matrix(plotMonth,3);
% p2U=Coef_Matrix(plotMonth,4);
% p1L=Coef_Matrix(plotMonth,5);
% p2L=Coef_Matrix(plotMonth,6);
% function4 = @(x) (p1*x + p2 );
% function4U = @(x) (p1U*x + p2U );
% function4L = @(x) (p1L*x + p2L );

%% plot
str1 = sprintf('Upper bound: %.1f',Lambda_U);
str2 = sprintf('Lower bound: %.1f',Lambda_L);
%str3 = sprintf('= %f',Lambda);

% plot(mrc_nff,function1(mrc_nff),'-ob',mrc_nff,function2(mrc_nff),'-
*r',mrc_nff,function3(mrc_nff),'-+g',mrc_nff,function3U(mrc_nff),'--
g',mrc_nff,function3L(mrc_nff),'--g')
% % hold on
% % fplot(function2 (mrc_nff),'--or')
% xlabel('Repair cost [$]')
% ylabel('Redesign cost [$]')
% legend({str1,str2,'{\lambda}_{input}=5'},'Location','best')
% axis([0,4000,0,8000])
% str4 = sprintf('Month %.0f',plotMonth);
% title(str4);

plot(mrc_nff,function3(mrc_nff),'-^k',mrc_nff,function1(mrc_nff),'-
*b',mrc_nff,function2(mrc_nff),'-+r',mrc_nff,function3U(mrc_nff),'-
k',mrc_nff,function3L(mrc_nff),'-k')
% hold on
% fplot(function2 (mrc_nff),'--or')
xlabel('Repair cost [$]','FontSize',15)
ylabel('Redesign cost [$]','FontSize',15)
legend({'{\lambda}_{input}=5',str1,str2},'Location','best','FontSize',12)
axis([0,4000,0,8000])
str4 = sprintf('Month %.0f',plotMonth);
title(str4,'FontSize',15);
```

```
% plot(1:12,Lambda_repair,'-or',1:12,Lambda_redesign,'-^b')
% xlabel('Months')
% ylabel('{\lambda}')
% legend({'Always repair for smaller lambdas','Always redesign for greater
lambdas'},'Location','best')
% set(gca,'XTick',[0:1:12])
```

1.4 File: PlotNPVandVOR

We evaluate the NPV when we have different costs and different months of data available, and we plot the NPV and the Value of Redesign at each month, during a year period

```
clc;clear all;

load('lamb5_0cost5yrs.mat')

%% INPUT DATA:
% We will evaluate the NPV when we have different costs and different
% months of data available. The costs of the main code are:
% mrc_nff = linspace(10,4000,values);                    % Invented value
$/month
% remov_cost = linspace(1000,8000,values);                    % Invented value
$/month

pair_costs_eval=10;    % INPUT the ROW position of the pair of costs that we
%want to evaluate of the repair_remove_pairs

for i=1:xcol
    % We evaluate the mean over all cases, with the same cost
    Mean_NPV(1,:,i)=mean( NPVexpbig(:,:,i));
    Mean_NPV(2,:,i)=mean( NPVexpbig_U(:,:,i));
    Mean_NPV(3,:,i)=mean( NPVexpbig_L(:,:,i));
    Mean_Cost(1,:,i)=mean(Costexpbig(:,:,i));
    Mean_Cost(2,:,i)=mean(Costexpbig_U(:,:,i));
    Mean_Cost(3,:,i)=mean(Costexpbig_L(:,:,i));
    Mean_noNFF(1,:,i)=mean(NPV_noNFFbig(:,:,i));

end

% This code looks for the position of those costs in the cost vector, their
% value has to be an existent value in the vector, otherwise it won't work

Mean_NPV_plot=Mean_NPV(:,:,pair_costs_eval);
Mean_noNFF_plot=Mean_noNFF(:,:,pair_costs_eval);
Mean_cost_plot=Mean_Cost(:,:,pair_costs_eval);

%% PLOTS
FigWidth = 6;    %INPUT
FigHeight = 6;   %INPUT
```

```
% FIRST GRAPH TO REPRESENT THE MEAN NPV FOR DIFFERENT MONTHS OF DATA, AT A
% SPECIFIC COST THAT WE DEFINED BEFORE
hFig1 = figure('Units','Inches','Position',[2 2 FigWidth FigHeight]);
set(hFig1,'Name','1');
plot(t,Mean_NPV_plot(1,:,:),'-xr',t,Mean_noNFF_plot(1,:,:),'-
ob',t,Mean_NPV_plot(2,:,:),'-r',t,Mean_NPV_plot(3,:,:),'-r')
xlabel('# Months of data')
ylabel('Net Present Value 5 yrs [$]')
legend({'{NPV}_{repair}','{NPV}_{redesign}'},'Location','best')
title('{\lambda}_{input}=5, {RC}_{A}=10, {C}_{design_A}=1000')
set(gca,'XTick',[0:1:12])

% GRAPH THAT SHOWS THE VALUE OF REDESIGN
hFig2 = figure('Units','Inches','Position',[2 2 FigWidth FigHeight]);
set(hFig2,'Name','2');
t2=0:1:n;
worthit=zeros(1,n+1);
plot(t,Mean_cost_plot(1,:,:),'k--o',t,Mean_cost_plot(2,:,:),'--
k',t,Mean_cost_plot(3,:,:),'--k',t2,worthit,'--r')
xlabel('# Months of data')
ylabel('Value of redesign [$]')
%legend({'{NPV}_{repair}, {MRC}_{A}=100,
{C}_{design_A}=2500','{NPV}_{repair}, {MRC}_{A}=1000,
{C}_{design_A}=3500'},'Location','best')
set(gca,'XTick',[0:1:12])
title('{\lambda}_{input}=5, {RC}_{A}=10, {C}_{design_A}=1000')
```

2 Multiple Lambdas study

This section is only useful if we want to run the Monte Carlo for multiple failure rates.

Here there are the 3 basic steps we need to follow in order to run our study properly:

1. Open the WholeMonteCarlo
    a. Input the data that can be modified (it says input) with the values we want to
    b. Set a name for our study in the end of the document
    c. Run the file
2. Open the LambdaLimits
    a. Write the name of our study
    b. Set the costs that we want to graph (it says inputs)
    c. Run the file
3. Open PlotLines
    a. Set the number of the month we want to plot
    b. Input the name of the files previously generated in 3. For different lambdas
    c. Run the file

The next subsections explain what each file does at each moment, in a more detailed way. Remember that everything is also explained in the .m files themselves.File: WholeMonteCarlo

This file runs for multiple failure rates, sections 1.1 and 1.2, saving each test with a specific name.

```
%% INPUT DATA
iterations=1300;
values=10;
alpha= 0.05;          % COnfident interval of 95%
n=12;                 % INPUT months of data
pred_year=5;          % years of estimated NPV data
mrt_nff = 3/(24*30);           % Invented value 3 h to repair a failure
expressed as a fraction of a month
mrc_nff = linspace(10,4000,values);                  % Invented value
$/month
remov_cost = linspace(1000,8000,values);                  % Invented value
$/month
oper_cost = 50;                % I will assume to have a 60$ cost of utilities
per month
hard_cost_nff = 10;                % $/failure cost of hardware
hard_cost_b = 20;
u = 1500;                      %Benefits that each unit reports per month when
working
r_At = 0.05/12;                     % Discount rate of 5% per year, meaning
0.05/12 per month
```

```
mrt_b = 6/(24*30);            % Invented value 3 h to repair a failure
expressed as a fraction of a month
mrc_b = 50;


for lambda=17.1:0.1:18
    % INPUT lambda mean per month

%% Define Montecarlo Study
lambda_b=round(lambda*3);
months = pred_year*12;          % months of projected data
years = months/12;               % years of estimated NPV data
[p,q] = meshgrid(mrc_nff, remov_cost);
repair_remove_pairs = [p(:) q(:)];
[xcol,ycol]=size(repair_remove_pairs);
% First define the empty matrixes
A=[];
AU=[];
AL=[];
C=[];
CU=[];
CL=[];
E=[];
EU=[];
EL=[];
G=[];
GU=[];
GL=[];
I=[];
J=[];
NPVexpbig=[];
NPVexpbig_U=[];
NPVexpbig_L=[];

NPV_noNFFbig=[];

Costexpbig=[];
Costexpbig_U=[];
Costexpbig_L=[];

for l=1:iterations
%% Random scenarios following a Poisson model
random_sample = poissrnd(lambda,1,n);
lambda_seq = transpose(random_sample);
lambda_ci = zeros(2,n);
t=(1:1:n);
% We generate the confidence intervals
        for q = 1:n
            [le(q), lci(:, q)] = poissfit(lambda_seq(1:q));
        end
lambdas_exp=random_sample;
lambdas_exp_upper=[lci(2,1),diff(lci(2,:).*t)];
lambdas_exp_low=[lci(1,1),diff(lci(1,:).*t)];

acc_l = cumsum(random_sample);
av = acc_l./t;
```

```
av_upper = lci(2, :);
av_low = lci(1, :);

% Representation of the data reported
%[l_sample, t_report,l_graph,t_graph] =
FailureReportGenerator1(t,random_sample);

%% Projection following Exponential distribution (HPP)
[data_t,failures_exp,lambdas_exp] =
ExponentialProjectionCI(av,t,months,random_sample);
[data_t,failures_exp_upper,lambdas_exp_upper] =
ExponentialProjectionCI(av_upper,t,months,lambdas_exp_upper);
[data_t,failures_exp_low,lambdas_exp_low] =
ExponentialProjectionCI(av_low,t,months,lambdas_exp_low);
%Comments
% The failures_exp is the total number of failures accumulated per period
% lambdas_exp is the counter of failures we have per month

%% Estimation NPV for Exponential distribution
    for p=1:xcol
        f_cost=repair_remove_pairs(p,1); %Cost of repair
        f_redesign=repair_remove_pairs(p,2); % Cost of redesign
        %% NPV when just repairing
        [NPV_year,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp,lambda_b,months,n,pred_year);
        [NPV_year_L,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp_low,lambda_b,months,n,pred_year);
        [NPV_year_U,t_year,PV] =
PVexponentialCI(mrt_nff,f_cost,mrt_b,mrc_b,oper_cost,hard_cost_nff,hard_cost_
b,u,r_At,lambdas_exp_upper,lambda_b,months,n,pred_year);

        %% Cost of eliminating all NFFs
        [NPV_noNFF_year,counter_l2] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,random_sample,months,n,pred_year);
        [NPV_noNFF_year_L,counter_l2_L] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,lambdas_exp_low,months,n,pred_year);
        [NPV_noNFF_year_U,counter_l2_U] =
PVnoNFFmodified(mrt_nff,f_cost,mrt_b,mrc_b,f_redesign,oper_cost,hard_cost_nff
,hard_cost_b,u,r_At,lambda_b,lambdas_exp_upper,months,n,pred_year);

        % Cost if assumed a HPP model
        Costexp=NPV_noNFF_year-NPV_year;
        Costexp_U=NPV_noNFF_year_U-NPV_year_U;
        Costexp_L=NPV_noNFF_year_L-NPV_year_L;

        %% Matrixes for each cost 3D
        NPVexpbig(l,:,p)=NPV_year;
        NPVexpbig_U(l,:,p)=NPV_year_U;
        NPVexpbig_L(l,:,p)=NPV_year_L;

        NPV_noNFFbig(l,:,p)=NPV_noNFF_year;
        NPV_noNFFbig_U(l,:,p)=NPV_noNFF_year_U;
        NPV_noNFFbig_L(l,:,p)=NPV_noNFF_year_L;

        Costexpbig(l,:,p)=Costexp;
        Costexpbig_U(l,:,p)=Costexp_U;
        Costexpbig_L(l,:,p)=Costexp_L;

        %%Comments:
        %x axis: number of cases
        % y axis: # months of data available
        % z axis: diff costs studied
    end

%% Saving each test
C(:,:,l)=lambdas_exp;
CU(:,:,l)=lambdas_exp_upper;
CL(:,:,l)=lambdas_exp_low;
I=[I;av];
% random sample
J=[J;random_sample];
end

    for i=1:xcol
        % We evaluate the mean over all cases, with the same cost
        Mean_NPV(1,:,i)=mean( NPVexpbig(:,:,i));
        Mean_NPV(2,:,i)=mean( NPVexpbig_U(:,:,i));
        Mean_NPV(3,:,i)=mean( NPVexpbig_L(:,:,i));
        Mean_Cost(1,:,i)=mean(Costexpbig(:,:,i));
        Mean_Cost(2,:,i)=mean(Costexpbig_U(:,:,i));
        Mean_Cost(3,:,i)=mean(Costexpbig_L(:,:,i));
        Mean_noNFF(1,:,i)=mean(NPV_noNFFbig(:,:,i));
    end
% TO GET THE BIG MATRIXES:
% MATRIX DECISION = 1 REDESIGN  0 REPAIR
% MATRIX MONTH = SAYS EVERY POSITION
Mean_Cost(Mean_Cost(:,:,:)>0)=1;
Mean_Cost(Mean_Cost(:,:,:)<=0)=0;
Coef_Matrix=[];
decision=[];
    for j=1:n
        % FOr each month

        for i=1:values
            %Matrix_Month(i,:)=Month((values*(i-1)+1):(values*i));
            Matrix_Decison1(i,:)=Mean_Cost(1,j,(values*(i-1)+1):(values*i));
            Matrix_Decison2(i,:)=Mean_Cost(2,j,(values*(i-1)+1):(values*i));
            Matrix_Decison3(i,:)=Mean_Cost(3,j,(values*(i-1)+1):(values*i));
        end
    BIG_matrix(:,:,j)=Matrix_Decison1(:,:);
    BIG_matrix_U(:,:,j)=Matrix_Decison2(:,:);
    BIG_matrix_L(:,:,j)=Matrix_Decison3(:,:);
    % TO GET THE LINES EACH MONTH:
    % Unique give us the different numbers that we have in the matrix in a
    % column vector
    nonzero1 = nnz(BIG_matrix(:,:,j));
    nonzero2 = nnz(BIG_matrix_U(:,:,j));
    nonzero3 = nnz(BIG_matrix_L(:,:,j));
        if (nonzero1==0)||(nonzero2==0)||(nonzero3==0)
            coeff1=[0,0];
            coeff2=[0,0];
            coeff3=[0,0];
        elseif
(nonzero1==values^2)||(nonzero2==values^2)||(nonzero3==values^2)
            coeff1=[9999999999,0];
            coeff2=[9999999999,0];
            coeff3=[9999999999,0];
        else
            coeff1= GetLineCoef(BIG_matrix(:,:,j),mrc_nff,remov_cost);
            coeff2= GetLineCoef(BIG_matrix_U(:,:,j),mrc_nff,remov_cost);
            coeff3= GetLineCoef(BIG_matrix_L(:,:,j),mrc_nff,remov_cost);
    %     elseif (nonzero==0)||(nonzero==values^2)
    %         coeff=[0,0,0];
        end
        Coef_Matrix=[Coef_Matrix;coeff1,coeff2,coeff3];

    end

filename=['A_' num2str(lambda*10)];
save(filename,'Coef_Matrix','BIG_matrix','BIG_matrix_L','BIG_matrix_U');

end
toc
```

2.1.1 File: LambdaLimits

It gets the the upper and lower bounds for the failure rates at each month according to the data generated in the tests generated in section 2.1.

```
clc;clear all;
n=12;                       % DO NOT CHANGE THIS VALUE
lambda=0.5:0.1:18;          %Range of lambdas that we are studying
for i=1:n
    for j=1:length(lambda)
        filename=['A_' num2str(lambda(j)*10)];
        load(filename)
        B(j,:)=Coef_Matrix(i,1:2);
    end
    Coefficient(:,:,i)=B(:,:);

    Repair=[0,0];
    Redesign=[9999999999,0];

    if strmatch(Repair,B(:,:))~=0
       % The last repair pair will be the lower limit
        Pos_repair=strmatch(Repair,B(:,:));
        Lambda_repair(i)=lambda(Pos_repair(length(Pos_repair)));
    else
        Lambda_repair(i)=0;
    end
    if strmatch(Redesign,B(:,:))~=0
```

```
        % The first redesign pair will be the upper limit
        Pos_redesign=strmatch(Redesign,B(:,:));
        Lambda_redesign(i)=lambda(Pos_redesign(1));
    else
        Lambda_redesign(i)=0;
    end
end


save('Coefficients')
```