# Introduction

A general introduction goes here.

## Background

* Problem Formulation
* Literature Survey
* What remains to be done.

## Objectives

The main objectives of this Master’s project are:

1. To rewrite, scrutinize, and test the DST
2. To develop new features for the DST

## Limitations

## Approach

Listing the main parts of the methodology, stating why each step was important. Check the disposition note in evernote

## Structure of the Report

# Methodology

Part 1, present; part 2 explain how. Results

## Code Quality

The DST will be an Open Source collaboration. This implies some demands on the code quality of the project.

### Naming Conventions, Commenting and Formatting

### Modularization and Encapsulation

Describe the concept

Describe the implementation

### Testability

The new

Years to go batt example, if tests were ready one would be able to spot a jump in prices.

### Evaluation of Finished Rewrite

In order to make a quantitative comparison of the rewrite and the original code, the bugs from the original DST was replicated in the rewritten DST. This way the changes in architecture is isolated and will produce the exact same data, if the implementation is correct.

Comparison of different outputs was used to prove maintained correctness. The function isequal(A,B) will compare every element in two matrices, and returns true (1) if the matrices are identical, and false (0) if there are one or more element with any kind of difference. Given an input with large span and high resolution, there should be enough data points to establish that the new implementation conserve the functionality from the old with a high degree of certainty. The probability of two matrices being identical coincidentally can be described as follows.

If the average is 50%, and we wish to be 99.999% sure that our implementation is correct, we can see that , and that the required amount of data points is

For one time series in the DST we have 8760 data points in hours, which alone should be sufficient to ensure correctness. When we evaluate the modules with lesser complexity, the data points decrease, but the chance of a data point being coincidentally wrong is presumed lower, since they are derived from the entirety of the previous data points. Given the validity of the calculation above, we can be more than 99.99% sure if we use the combinations of different PV and battery sizes, for good measure we will output combinations / data points.

In the rewritten DST, the calculations are executed in modules that pass parameters between them. These are placed in three classes that describe the key outputs of each module, and the names have the benefit of a prefix to explain them (ex. simulationOutputs.stateOfCharge).

|  |  |
| --- | --- |
| Variables | |
| Before Rewrite | After Rewrite | Module | n Data Points |
| SoC | stateOfCharge | sapv\_plant\_simulation |  |
| LoL | lossOfLoad | sapv\_plant\_simulation |  |
| NPC | netPresentCost | economic\_analysis |  |
| LCoE | levelizedCostOfEnergy | economic\_analysis |  |
| MA\_opt\_norm\_bhut\_jun15\_20\_10(5) | lossOfLoadProbabilities | get\_llp\_constrained\_optimum |  |
| MA\_opt\_norm\_bhut\_jun15\_20\_10(3:4) | [pvKw, battKwh] | get\_llp\_constrained\_optimum |  |

The two versions were tested to see the degree of overhead presented in the two cases.

### The Rainflow Counting Algorithm

The Rainflow Counting algorithm is an algorithm initially used to account for stress exposure in materials, initially in full cycles, and later in partial cycles. One cycle is one instance of full stress exposure, a partial cycle is an instance of a partial stress exposure. The algorithm has since been developed to account for stress in batteries (You, et al., 2011) and other appliances that go through similar wear.

The expense of battery replacement is high, importance If the DST become a widely utilized tool by micro and small scale enterprises, it is necessary which rely on small margins, there should be

#### Original Algorithm: (You, et al., 2011)

1. Initiate a vector in encounter of a stress local minimum.
2. Note increase in stress during rainfall of vector
3. Count occurrences of ranges as one cycle
4. Sum the expended partial cycles for every stress level accounted for

Where n represents the number of bins chosen in the study; Nc(DOD) represents the number of consumed partial cycles at a given DOD level, derived by counting in the corresponding period; No(DOD) represents the maximum number of partial cycles that can be performed before battery failure at that DOD level.

Where ExpL denotes the expected lifetime of the BS, and Tp represents the length of the counting time period.

#### Implemented Algorithm:

1. Discover discharge valleys (can currently only occur after 8 consecutive hours of discharging)
2. Count cycles to failure:
3. Accumulate cycles to failure
4. Estimate battery lifespan

A requirement for finding the lifespan in years, is that the spendage of lifetime fractions are summed over one year precicely. This way we get the amount of years that the battery need.

#### Improving the Implementation

Every input related part of the DST should assume a generic form. This implementation assumes the that the rainflowCounter will keep counting for exactly one year, there was therefore added a year counter to the algorithm as seen when calculating ExpL in the original algorithm.

|  |
| --- |
| Figure 2:1: Plotting DoD(Cycles To Failure) w/initial parameters.  There are not sufficient points of DoD in the intervall 0-20% to represent the Cycles to Failure accurately. |
|  |

The cyclesToFailure is calculated for every instance of DoD valley that occurs, which has an complexity of **O(n).** This resolution in cycle values might not be required considered the massive abstraction level of which we are operating. A future extension of the DST will at some point trigger far more occurences of DoD, before a simulation is considered complete. Users might want to input longer time series or change the 8 hour consecutiveness condition. In earlier versions of the algorithm(Simple Rainflow Counting Algorithms, 1982), the method proposed is preemptively generating a table of maximum cycles before failure for every DoD percentage.

Given that the rate of change in DoD is not too close to zero, so that one has horizontal asymptotes, the points of DoD will sufficiently describe the Cycles To Failure. This is because there will be many points of DoD to describe the different Cycles To Failure values.. By inspecting Figure 2:1, one can plainly see that it would not be safe to approach the problem at 1% granulity, the rate of change is too low until about 20% and we will not accept a DoD lower than 60% due to the excessive loss of cycles this choice imply. This can be solved by increasing the resolution of the DoD values to 0,1% or even smaller in the precalculated array, but seeing that the overall run time of the program is there is a neglible amount of time to gain, compared to the testing that would be needed to ensure satisfactory results. Hence the initial algorithm that calculates on each occurrence of DoD, will be kept in the DST.

The mentioned 8-hour consecutiveness condition is also implemented in (Simple Rainflow Counting Algorithms, 1982), here the condition is 3 points.

## Biomass Power Generation

# Summary and Recommendations for Further Work

## Summary and Conclusions

Through renaming every variable to optimize the readability of the DST, the design flaws and inner working of the tool became apparent. It was discovered that certain parts of the hard coded and hard-to-find parameters were responsible for….

## Discussion

## Recommendation for Further Work