Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Project Status Report

Optimizing parameters of a BLDC motor model

Max Schachtschabel

June 14, 2015

Betreuender Dozent:
Prof. Dr. Kreatzschmar

Contents

Abstract: Simulation and Modelling has become a major part in mechanical and electrical engineering and are tasks which base heavily on mathematical and physical theroy and the ability the write time efficient algorithms that enable full control over a given system. In this particular project, a model for a electrical brushless direct current (BLDC) motor is given based on a set of differential equations and another set of motor specific parameters. Determining these variables however is a task that can not always be done just by looking at the technical specifications of the motor or by meassuring; in most cases even the known values are merely estimated. Thus, machine learning algorithms can be used to optimize these parameters by comparing the behaviour of the system (e.g. the velocity curve w.r.t time) to the real system and adapt take the parameters as variables that need to be adjusted such that the error between simulation and reality becomes arbitrary small.

1 Background

The STELLA project, founded supervisied by Prof. Dr. Asteroth, focuses on the usage of electrical light-weighted vehicles such as E-Bikes and Velomobiles and how to optimize the amount of energy needed drive a certain known track. The optimization mainly regards the shape of the vehicle and the way the energy storage (in our case a lithium based battery pack) is used by the driver; in many cases, there is no need to go full throttle to climb the oncoming hill, and especially no need to use energy to accelerate downhill. What most truck- and train-drivers learn in driving lessons can also be put in algorithms, even more when there exists sensory to determine the slope of the next x meter on a given track. The type of algorithm used here is based on evolutionary strategies and is able to set a „full speed“, „hold speed“or „roll“command based on a given height map and a wanted driving time with respect to the optimal amount of energy needed.
Developing and testing this algorithm is done in simulation; to successfully simulate the driving of the vehicle on a road, there are many different kinds of models needed: The vehicle shape and the resulting drag forces during driving, the wheels, the battery, the driver itself and also the motor and the controller have to be modeled as detailed as possible.
My focus rests on the brushless direct-current(BLDC) motor and the corresponding controller that is currently built in one of the vehicles. The current state of the model is based on a set of differential equations that describe the physical transformation from electrical to mechanical power. To apply this generic model to the specific type of motor used, there exist a couple of motor specific parameters which are used within the calculations. This shall be the entry point for this subproject.

Even though the parameters are neccessary for the correct computation of the physical formulas, some of them are not explicity known. Most of them are given by the specific system the motor is placed in (Supply voltage, wheel radius, air-drag coefficients of the vehicle) and some internal values are given by the manufacturer of the BLDC motor within datasheets (internal resistance, inductivity, amount of magentic poles). These values are also accessible through external meassurements and thus could be validated in the past. However, to fully model the motor, the following parameters are still uncertain and only roughly estimated:

- Voltage constant $k_w[\frac{V}{rpm}]$
- Torque constant $k_w[\frac{Nm}{A}]$

- Magentic Flux constant $B[T]$
- Inertia $J[kg * m^2]$
- Load torque $T_m[Nm]$

Optionally, the following parameters can also be adjusted for further optimization (if neccessary) , since the values are given by the datasheet but may or may not be precise:

- Coil resistance $R[\omega]$
- Coil inductivity $L[T]$

The rest of the parameters was confirmed by a reliable source (either through contact with the manufacturer or through repeated meassurements) or will be made concrete by other subprojects and thus won't be the focus of this particular project (e.g. everything regarding the aerodynamics of the vehicle:

- Supply voltage = $40V$
- Number of magnetic poles = 42
- Radius of the backwheel = $0.23m$
- Mass of the vehicle + driver = $100kg$
- Aerodynamic constants
  - cwA = 0.2
  - crr = 0.01
  - $\rho = 1.2$

2 Current state of the simulation

At this point, rough estimates of these values led to a working motor model, but the process of acceleration still shows that there is a notable difference between simulation and reality. The following graph shows a direct comparison between an acceleration of the whole vehicle including the driver on a flat track in simulation and in reality:

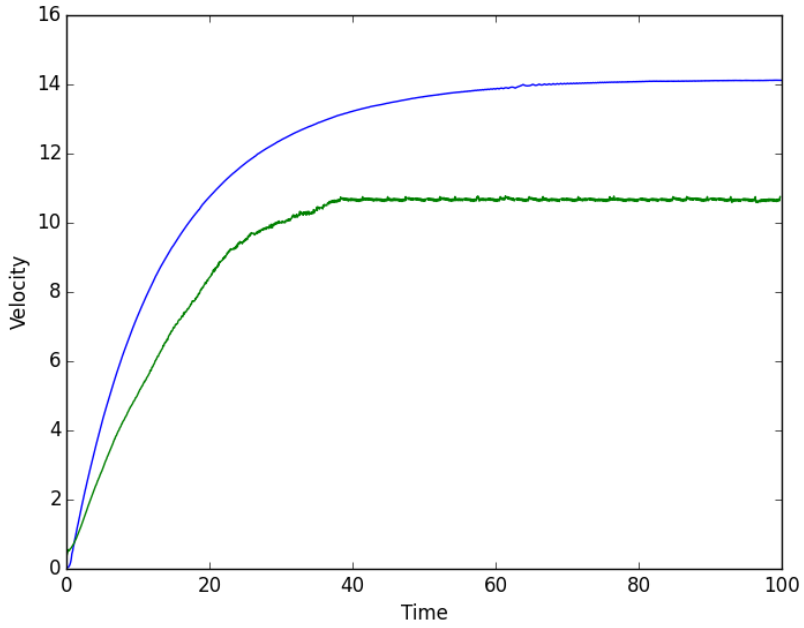


Figure 2.1: Velocity w.r.t. Time in reality (green) and in simulation (blue)

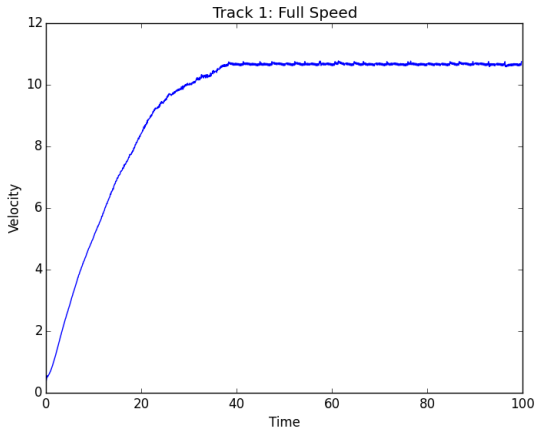
The error between the current set of parameters in the simulation and the behaviour of the real system can be seen very clearly in the picture: There seems to be a difference in acceleration in the first few meters as well as a certain offset in the top speed of the vehicle. To see that small changes in the named parameter values which will optimized really have an effect on the simulation, the red lines show the behaviour of the system with different sets of parameters. The changes made here are random, though they keep a certain bounded range depending on each individual parameter. Of course, there are more output values that need to be compared between simulation and reality; espacially the current power needed to move the vehicle is of interest for the STELLA project.

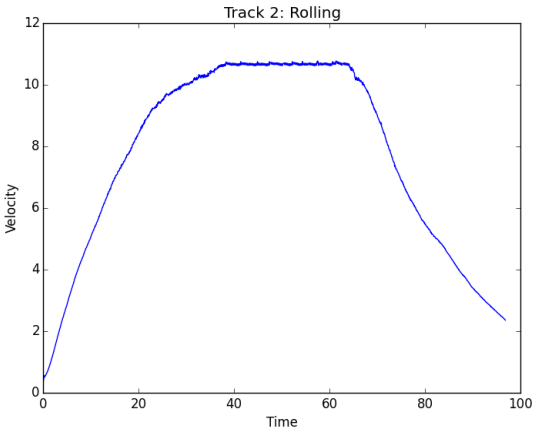
However, during this project, I want to focus mainly on the velocity, since these values can be achieved quite easily from the real system and the error between simulation and reality is reasonable small; other behaviours are still a work in progress and wouldn't lead to a statisfying result regarding the approach of using machine learning until the model is further developed.

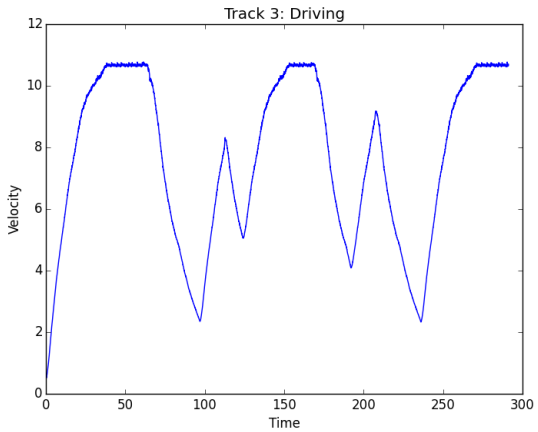
To be fully able to test every possible behaviour of the motor during the driving process, there are three different kinds of real data to represent the reality which are compared to the model:

- A simple acceleration process that holds the maximum speed for some time
- An acceleration with a following rolling process, meaning that the vehicle uses its momentum to keep the velocity but is slowly baffled by the rolling resistance and the air drag
- A sequence of acceleration and rolling phases at variing points in time

The data is obtained by driving the vehicle over a long distance with a minimum amount of slope or other circumstances that could alter the simplified driving process. Since we are interessted in the parameters of the motor as it is running in this particular vehicle on a real street, this data shall be our foundation of further research; errors in the meassurement of velocity and distance are valid, but not of much relevance for our application, since we are able to analyze mutliple runs of the same track and can thus minimize the influence of these errors as well. The curves for each of the listed cases can be seen in the following figures, plus the corresponding output of the current un-optimized model.


Track 1: Full Speed
Velocity
Time
0
2
4
6
8
10
12
0
20
40
60
80
100


Track 2: Rolling
Velocity
Time
0
2
4
6
8
10
12
0
20
40
60
80
100


Track 3: Driving
Velocity
Time
0
2
4
6
8
10
12
0
50
100
150
200
250
300

3 Choice of machine learning technique

The choice of a specific learning technique is not easy, since there doesn't seem to be a standart, exemplary method for curve fitting in general. Minimizing an error function based on a whole simulation is also not an easy task, and even though there are "only" five parameters to be optimized (at first), we are talking about a five-dimensional function that has a global minimum somewhere with the correct set of parameters as coordinates. To determine the error for a given set of parameters, an entire simulation run is needed, so there is an upper limit on how many iterations the technique should need before it produces a satisfying result.

3.1 SMV, RBF and other Neural Network Types

This approach was my first intent to look at, since I already have some experience in implementing and understanding these types of machine learning. All of the named types of Neural Networks and especially the Support Vector Machines are fairly simple to build and in many cases also very effective in their way of adapting to a given input signal and predicting the output to yet unknown data.
The idea would be that I use a network to adapt my model to the real world data, meaning that the network behaves just like the real vehicle. In theory this sounds reasonable and there is also enough data available to train and test the network succesfully; however, there is a problem with this kind of learning technique when looking at the original task of this project: I want to use machine learning techniques to find the best set of parameters so that my exisiting modell fits as good as possible with the real world data. This way of learning would produce a network with a certain set of optimal weights to produce the smallest error possible between the desired and the real output. These weights have nothing in common with the used model and don't give any clues on how to find these; the network has no way of connecting the used simulation and the real vehicle, it would come up with its own model to fit the data, which doesn't bring me closer to the solution of the optimal parameter problem. Since all types mentioned above basically work on the same principle of finding the best weights/kernel functions/support vectors to fit to the input data, I can exclude these techniques from this project and look for other possible approaches.

3.2 Particle Swarm Optimization

This optimization technique can be found in the category of swarm intelligence, which can be seen as a discipline of reinforcement learning. This kind of machine learning is inspired by behaviour analysis, often done on certain kind of animals; many people have studied the group behaviour of birds, ants, bees and many more and came up with ways of adapting these techniques to solve optimization problems. The named algorithm was inspired by the movement of bird flocks and insect swarms when looking for food in an enviroment like a giant corn field. The idea is that each bird is able to remember the best place for food it had found and is also able to know the best solution of the entire flock. This results in a the movement we can witness in a real flock of birds, where each bird is drawn towards its own best location as well as the best location of the flock.
This behaviour is modeled by making a particle consist of three vectors: the current location in the search space, the best location found by it up to this point, and a velocity term. Each location is given a fitness value that needs to be computed, and these values are also memorized and used to compare it to the global best solution. The velocities are adapted each iteration by the following formula:

$$v_i = v_i + \phi_1 R_1(p_i - x_i) + \phi_2 R_2(g_i - x_i) \tag{3.1}$$

where
v_i is the current velocity of particle i,
x_i is the current position of a particle i,
p_i is the best personal location found,
g_i is the flock's best location found until now,
R_1, R_2 are uniformly distributed random values and
ϕ_1, ϕ_2 are constants to control the influence of the personal and the global best solution on the the change of the velocity (similar to a learning rate).

When a personal best solution is found that is better then the global one, the flock's best solution is replaced and the iteration proceeds until a stopping criteria is reached. Particle Swarm Optimization (PSO) implementations in Python can be found within the „ecspy“package, which also involes a number of other different evolutionary approaches to optimization problems. The usage of this implementation is pretty straight-forward, as it can be seen in the following chapter when I apply this approach on my optimization problem.

4 Implementation and Results

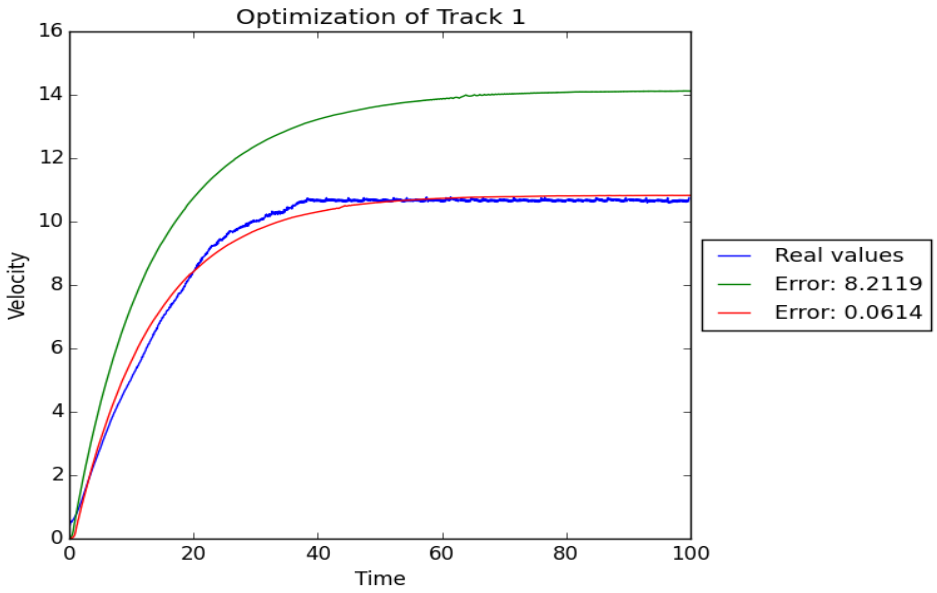
Since the PSO technique has a lot of promise, I the implementation of the „ecspy“package on the search for a global minimum of my 5-dimensional error function. The specific way of merging these two programs and the results of the optimization can be found in this chapter.

4.1 Implementation details

To be able to compute new candidates and to evaluate the fitness of each one ,the algorithm needs a total of two functions: one generator for new possible solutions to add to the swarm, and one evaluator to determine the fitness value. In our case, the generator comes up with a new set of parameters based upon the initial parameters from the previous model and adds a random change to each parameter, based on a normal distribution with a tunable variance for each parameter. This inherits that the solution we are looking for is somewhat related to the initial solution, since we know that these parameters are only estimates of the real ones. It is reasonable to believe that we can come up with a good, if not the best solution if we stick to what we already know and keep the search space limited. Maybe there do exist other solutions, maybe even better ones, outside of our field of view, but these would not be useful for our goal of finding the true set of parameters for our vehicle.
The evaluator takes the candidate as a parameter for calling the simulation by using the Simulation class as mentioned in the previous section. Once the track has been driven using the given parameters, the MSE is returned and can be used by the optimization algorithm. The rest of the optimization parameters covers the bounds of every parameters, the reason of terminating the algorithm, the size of a swarm population, the size of the direct neighbourhood of one solution and the topology of candidates. After some computation time, the algorithm returns the best candidate of the current population. Since there are no stochastic operations in the simulation, this array of parameters can be tested and validated again just by calling the simulation with the resulting set of parameters.

4.2 Plotting and results

The following figures show the results of the algorithm used like explained in the previous section. The few parameters for the optimization are either untouched or slightly adjusted to our problem; however, they did not have a big influence on the outcome of the optimization.

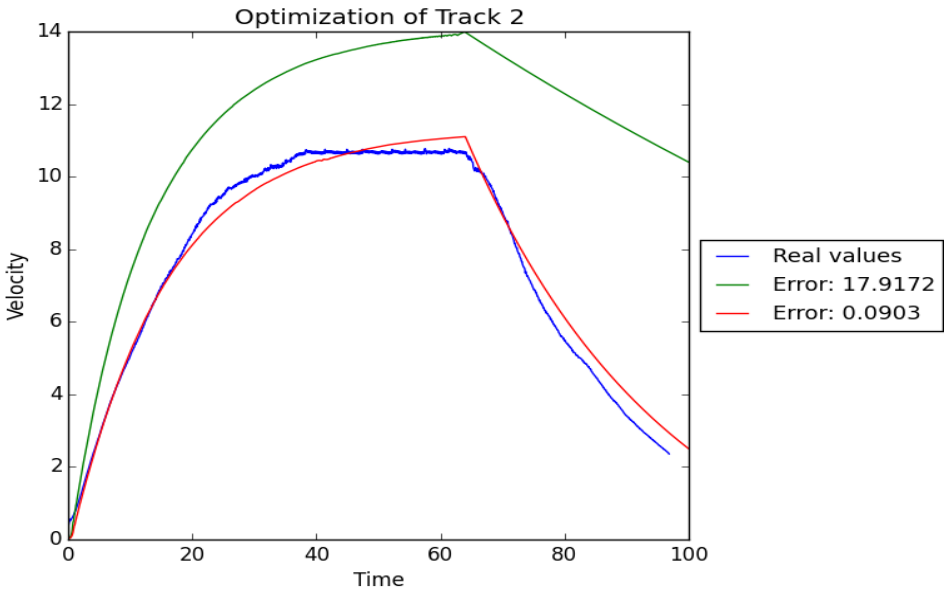


(a) First test case: Full Speed

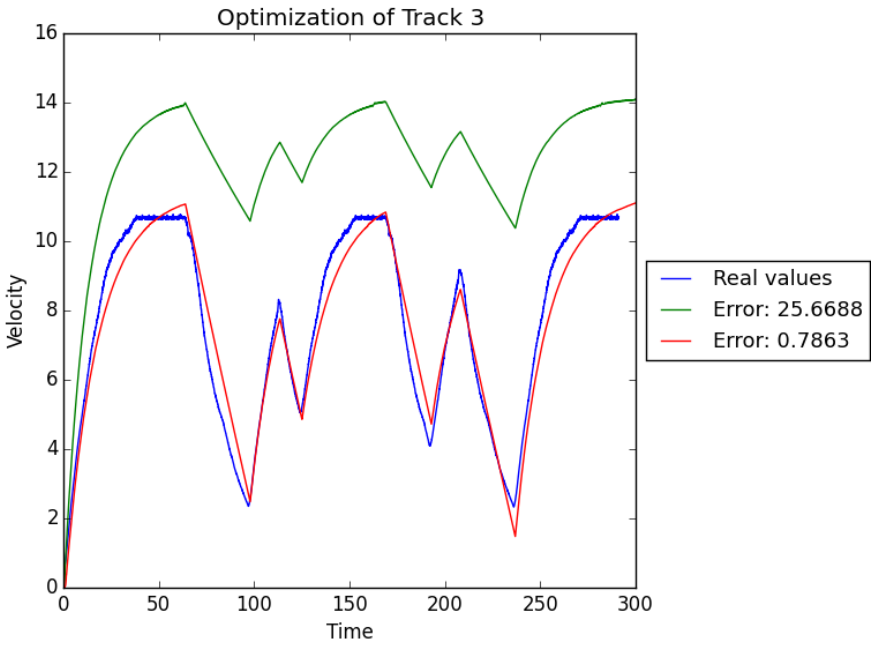
The first two test cases have been approximated quite well, which can also be seen on the improvement of the MSE between the real velocity readings and the model's predictions. The set of parameters manage to fullfil their purpose in simulating the reality, but the two sets are not equal: Since the second case shows a more complex behaviour, more optimization has been done and the algorithms eventually finds the parameters from the first case not statisfing for the new case.

	k_w	k_T	J	B	T_{load}
First track	0.04085	0.68635	0.50199	0.12066	18.2607
Second track	0.04123	0.62803	0.24343	0.06776	23.2637
Third track	0.04283	0.64382	0.59976	0.00385	36.3808

The outcoming set of parameters from the last run speak the same language: A more complex case asks for a better solution; while in the simpler cases, certain parameters don't have that much of an effect on the resulting graph, they could be very meaningful in a more complex case like the third run. This is also the reason that every run was executed individually from the same initial parameters and not sequentially, meaning that the best solution is taken as a new foundation for the next run.



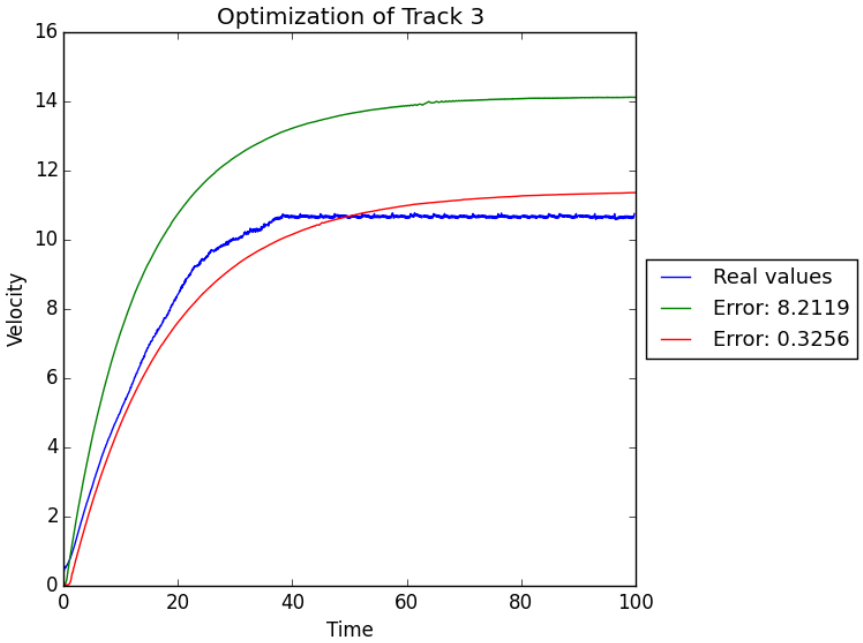
(a) Second test case: Rolling



(a) Third test case: Sequence

This approach can be questioned when looking at the resulting graph of the third test case. Here, we have many changes in the velocity curve based on a lot of changes in the

speed command at different moments in time. This case should be the most realistic one, since speed regulations during the driving process are also quite common compared to a simple „hold speed“or „roll all the way“situation. Since this is only one example of such a driving process, it can be seen that even though the graphs overlap most of the time, the specific set of parameters may not be able to cover every possible scenario that well. This gets clear once we try the resulting parameters of the third test case on one of the easier examples :

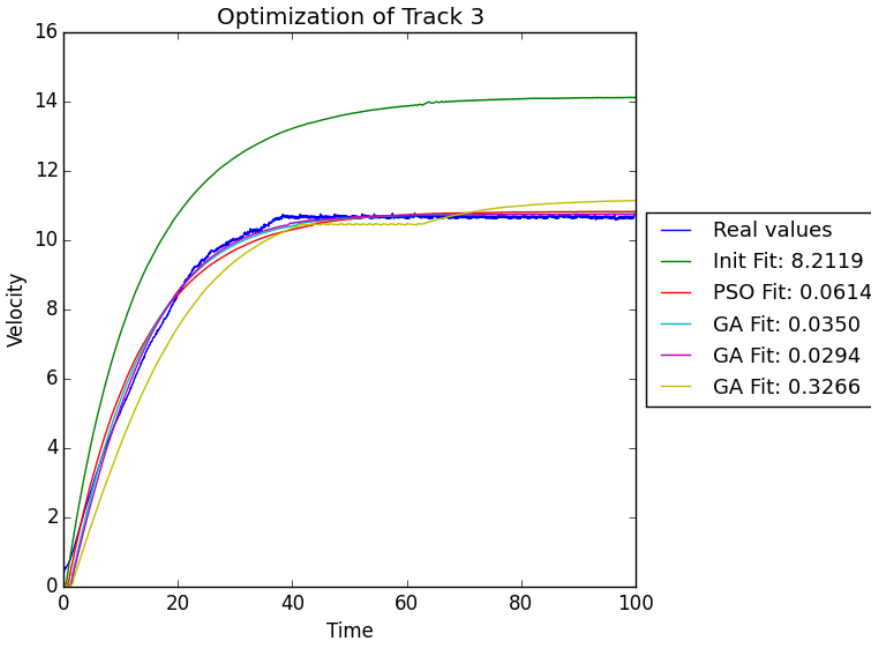


(a) First track with parameters from the third track

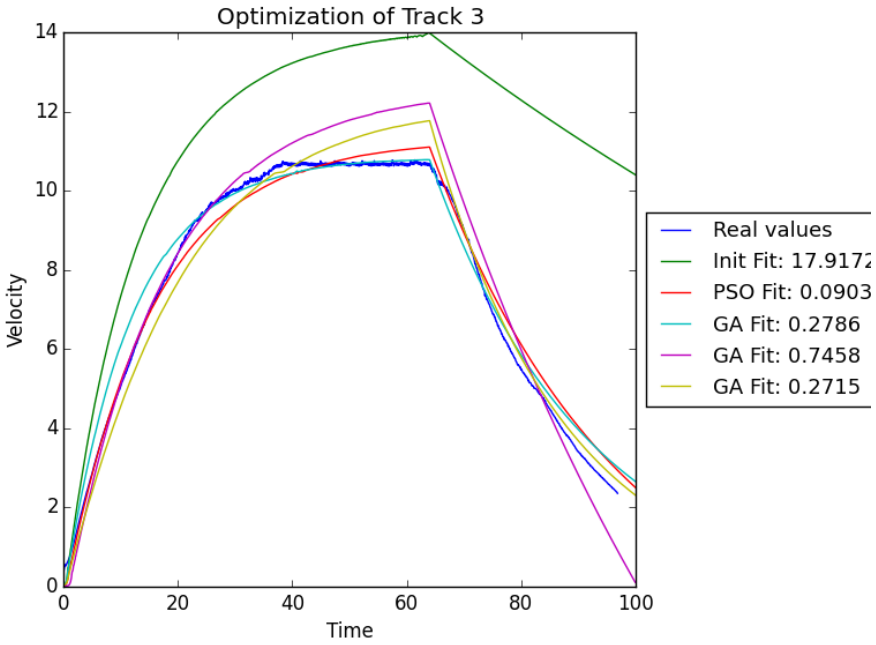
The outcome of this simulation run was to be expected, since the parameters of the third test case create a slightly different motor behaviour which is now not optimal for this case anymore. It can be argued that we fell victim to a case of overfitting our individual solutions to the given test case; to evalute this, we would need to examine all the possible driving procedures possible on a given time slot and see if we can find a solution that optimizes every one of them. Such an examination could be part of a following project; however, it is questionable if this is really neccessary looking at the application field of this model. Even though the solutions may not be optimal, we know that we try to optimize an exisiting model that already has some simplifications built in while trying to simulate a complex real-world process; a true optimal solution to this problem would need to look at every aspect of this very model and not only on a uncertain set of parameters. Since the main purpose is to have a functional and also a fast model of the vehicle that is a good approximation of the real one, further work on this topic may not be neccessary at this point in time.

5 Comparison: Genetic Algorithm

To be able to evaluate the approach chosen in this project, it is appropriate to compare the results to another technique that can be used in a similar matter to obtain a solution for the given problem. Since there exists another project in parallel for the same task, we can compare these results with the other approach, being the usage of a genetic algorihm to evolve a generation of individual solutions in such ways that a good or even the best solution can be obtained.
The details of this approach and the theory behind genetic and evolutionary strategies exceed the limits of this paper. However, some basics overlap with the particle swarm technique used in this project: Both are population based, stochastic methods with individual ways of evolving the population over time and gradually generating better results with time. Genetic Algorithms (GA) and Evolutionary Strategies (ES) are inspired by the natural selection process „survival of the fittest“and model the passing of good *genes* to the next generations to come. The following figure shows the best results of this approach on the first test case from the earlier chapters:



The quality of the results are equal to the ones obtained by the PSO approach. However, the amount of time needed to optimize the runs of the genetic algorithm were much larger then the ones of the PSO; while the GA is very sensitive regarding the optimization parameters such as population size, crossover rate and mutation rate, the technqiue still needs multiple repeated runs to obtain a statisfying solution. The problem here is the vast amount of local minima in the problem space which are hard to overcome using algoritms that are based on the local neighbourhood of the individuals. The next graphs show the limits of the genetic approach, since even the best final solutions from multiple runs, despite the fact that each of them claims to have found a different optimum, were not able to let the simulation approach the reality as well as the PSO:



(a) Second track with the solution parameters of the GA

While the approach using a GA still has some potential by adding further extensions that help to overcome local minima, the current state shows that the PSO brings better or similar results in less computation time and with less implementation and adaptation effort. With the GA approach still being the foundation of another ongoing project, the results of this paper can be used as a good comparison to the other results, as well.