1 Problem 2.1, The traveling salesman problem (TSP)

Parameters used: `numberOfAnts` $= 50$, $\alpha = 1$, $\beta = 3.0$ and $\rho = 0.35$.
Best path found: `bestPath` $= 97.7751$

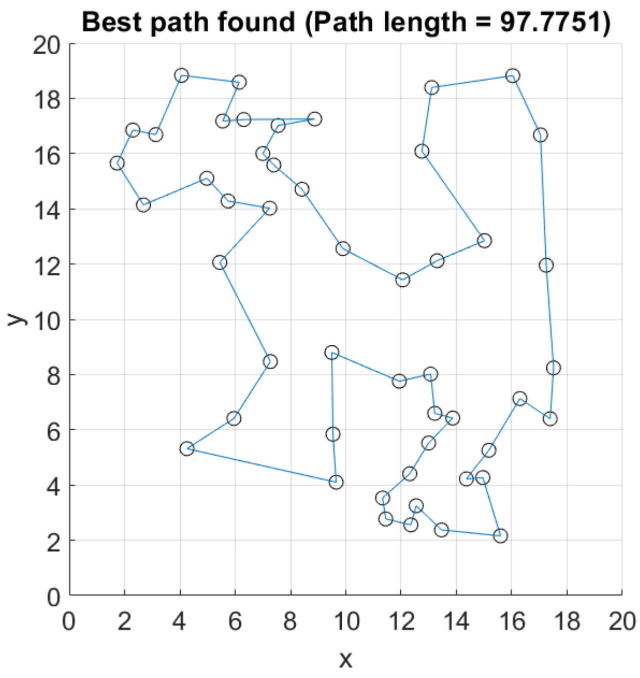

Figure 1: Best path found with length 97.7751.

2 Problem 2.2, Particle swarm optimization (PSO)

The objective is to find the global minimum to

$$f(x,y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right), \quad (x,y) \in [-5,5]. \tag{1}$$

contours in figure 2 is drawn using $log(0.1 + f(x,y))$ to make the minima more visible in the image.

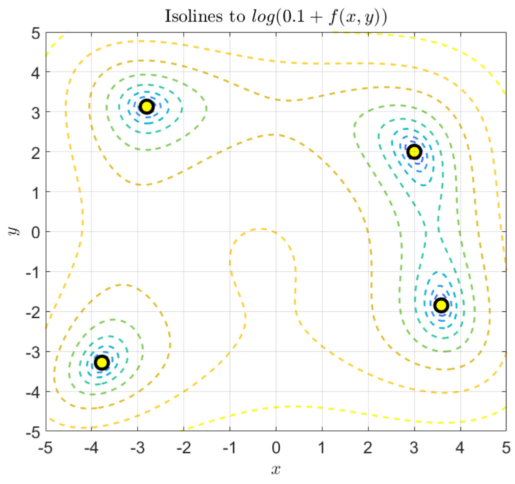

Figure 2: Contours of $log(0.1 + f(x,y))$ with four minimum marked as yellow dots.

x	y	$f(x,y)$
-2.8051	3.1313	0
-3.7793	-3.2832	0
3.5844	-1.8481	0
3.0000	2.0000	0

Table 1: Minima found using PSO. Several runs of the algorithm were needed to find all four points, the value of $f(x,y)$ in this points is in the order of 10^{17} and therefore set to zero.

3 Problem 2.3, Optimization of braking systems

To optimize the braking system a feedforward neural network (FFNN) was trained with a basic genetic algorith, GA, (Algorithm 3.1 in the course book). The FFNN consists of a first hidden layer with 6 neurons with input consisting of three normalized signals, velocity v/v_{max}, slope downward angle α/α_{max} and brake temperature T_b/T_{max} together with a bias term b. The hidden layer was then followed by an output layer consisting of two neurons which was trained to act as pedal pressure control and gear shifting signal. The logistic sigmoid function $\sigma(s) = (1 + e^{-s})^{-1}$ was used as activation function in both layers.

3.1 Training the FFNN

r To train the FFNN with GA the weight metricise in both layers was encoded to one single chromosome (according to the coding scheme in lecture 9), this was done for 500 randomly initialised FFNN's to get a population of 500 truckers to work with. Each trucker was evaluated by decoding the chromosome to a FFNN and use it in a truck model `EvaluateTrucker.m` which returns a fitness score based on the performance. The fitness score was computed as a function of distance travelled and average speed. The computation scheme for fitness is presented below as a `Matlab` function where `x` is distance traveled until termination, `xMax` maximum distance, maximum speed `vMax` and `vMean/iDrive` as the average speed computed as speed every iteration divided by the number of iterations. For a given network this was computed with all slopes in a data-set, then averaged to get a fitness score $\bar{F}$, i.e $\bar{F} = N^{-1}\sum_{slopes} F$, where $N = \#slopes$.

```
lengthFitness = x/xMax;
if x ≥ xMax
    velocityFitness = 1.1*(vMean/iDrive)/vMax;
else
    velocityFitness = 0;
end
fitness = (lengthFitness + velocityFitness)/2;
end
```

This fitness function penalises a uncompleted trip by setting `velocityFitness` to zero and rewards a trucker with a 10% bonus if the full slope is completed successfully.

To avoid overfitting the network holdout validation was used by plotting the fitness score of the best performing network (measured ob the training set) and then used to compute a score on the validation set each generation. In figure 3 the fitness score is shown for each generation. And in figure 4 an example run by the best chromosome/network on a slope from the test set.

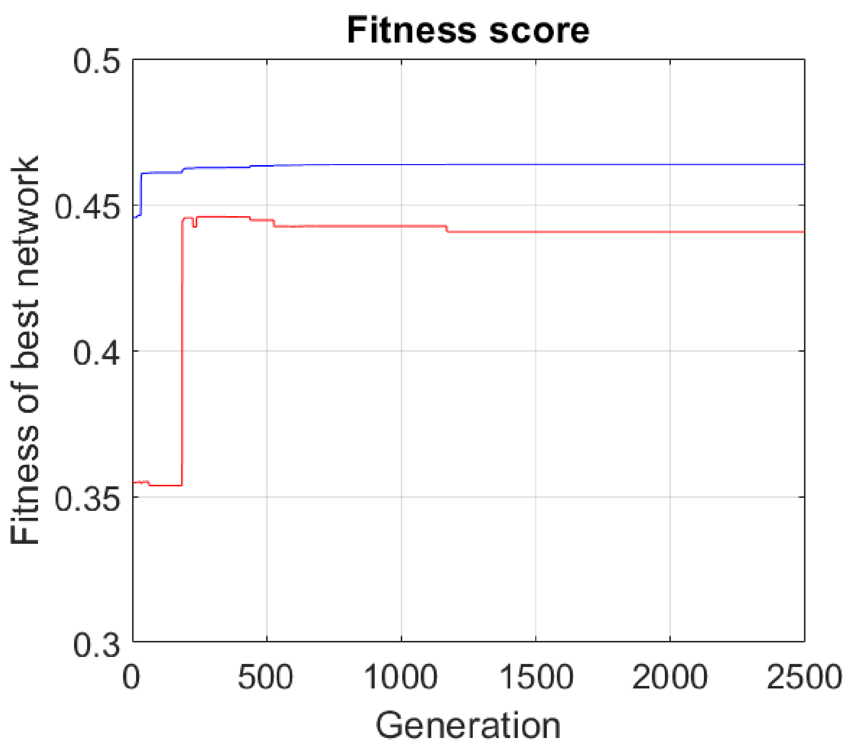

Figure 3: Fitness score as a function of generations iterated (2500 generations shown). Performance on training set in blue and validation set in red. The decrease of performance on the validation set at generation number 500 is a indication of that the training is done. The strength and shape of this signal varies depending on simulation parameters, e.g GA probabilities.

4 Problem 2.4, Function fitting using LGP

The LGP program used two point crossover and a basic GA with tournament selection, two point crossover and the best chromosome was saved each generation. A small number of registers (variable and constant) was used as suggested in the problem description, i.e four variable registers ([x, 0, 0, 0]) and 6 constant registers ([5, 5, 5, 5, 5, 5]). To avoid premature convergence the best individual was mutated ever 1000th generation with a mutation rate of 0.1. The population size was set to 1000, this doesn't increase computation time of one generation that much since `parfor` in Matlab can evaluate individuals in parallel. Figure 5 shows the curve produced by the chromosome fitted to data.

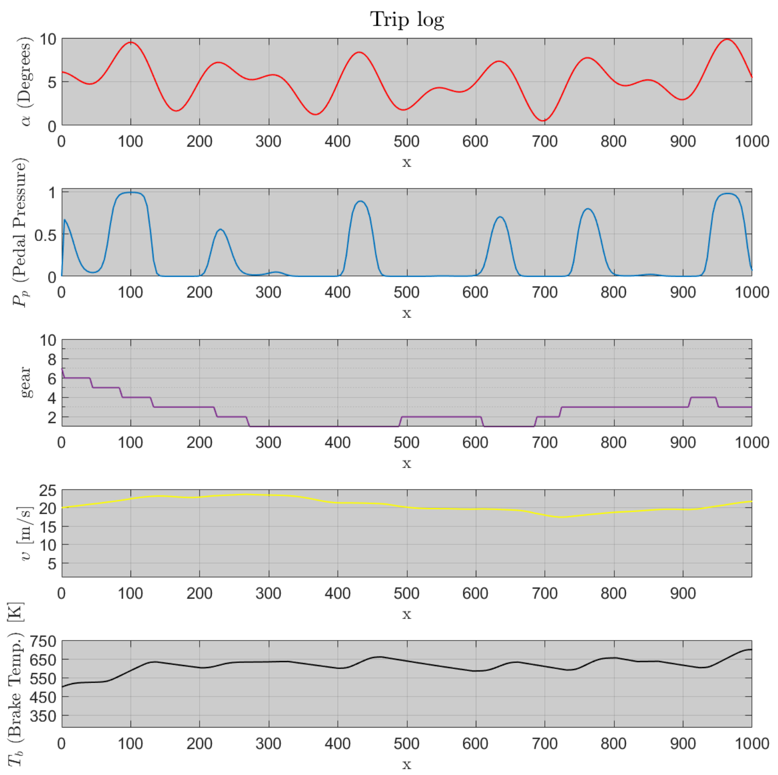

Figure 4: Trip log of a run with the best chromosome on a slope from the test data-set. Slope angle (α) in red, brake pedal pressure P_p in blue, gear in purple, speed v in yellow and the brake temperature T_b in black as functions of the horizontal distance travelled x.

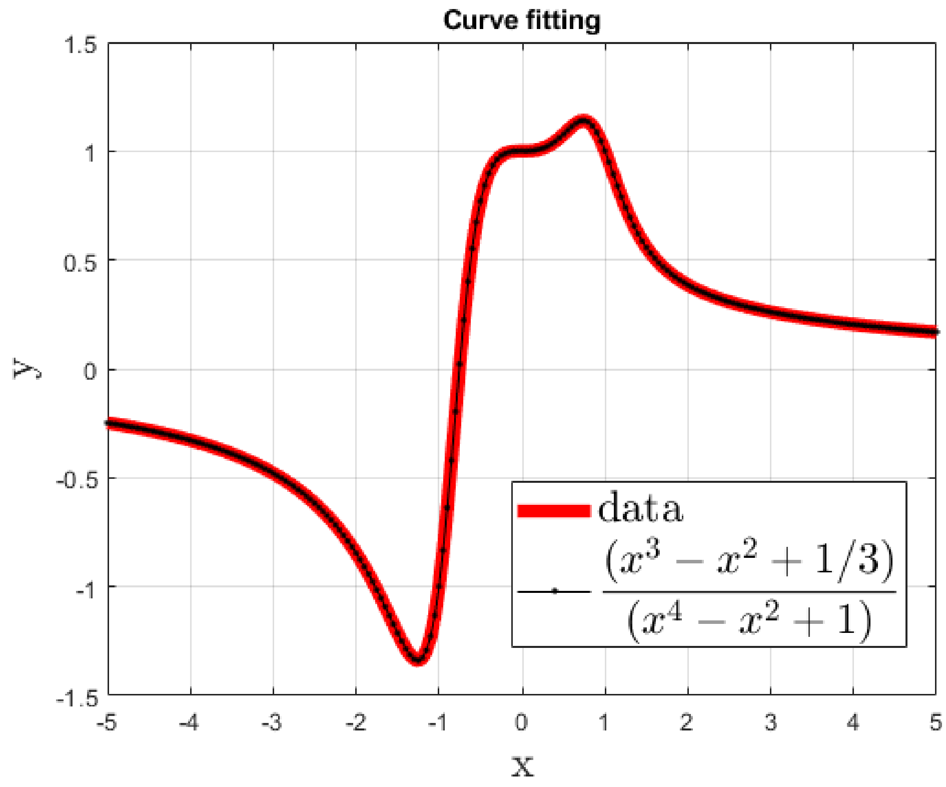

Figure 5: Provided data in red with a fitted curve in black. With an error $E_{rms} = 2.8480 \cdot 10^{-9}$. The symbolic output of the sequence of instructions decoded by the best chromosome was simplified with symbolic toolbox in matlab.