# Appendix B: Matlab functions in ERSim

ERSim is based on the ARSim autonomous robot simulator (see Appendix A for details). Here, only the parts relevant to the artificial evolution of robotic brains will be described. Compared to ARSim, two additional libraries of Matlab functions, RunFunctions and EAFunctions, are available in ERSim. The former library contains functions needed for creating, running, and evaluating an individual, whereas the latter contains functions for generating new individuals. Basically, the flow of the simulation is as follows: (see also Fig. 4.9 and the EvolveRoboticBrain Matlab function) First, an arena and a population is created. In the standard version of ERSim, it is assumed that all chromosomes are of equal length. Next, a loop over generations is initiated, followed by a loop over individuals. For each individual, the corresponding brain is created, and is then inserted into a standard robot (generated using the CreateStandardRobot function in RobotFunctions). Next, a simulation is created. During execution, the simulation evaluates the robotic brain. Note that fitness values can be updated both *during* and *after* a simulation. Thus, it is possible to consider both local and global fitness measures. In the default version of ERSim, the (global) fitness measure is obtained using the GetFitness function. When all individuals have been evaluated, a new generation is created using the MakeNewGeneration function which, in turn, uses Matlab functions for tournament selection, crossover, and mutation. In order to test ERSim, start Matlab and write

```
>> EvolveRoboticBrain
```

The following list shows the Matlab functions that are available in ERSim, but not in ARSim (for a list of those functions, see Appendix A).

### CreateBrainFromChromosome

**Library:** RunFunctions
**Interface:** `b = CreateBrainFromChromosome(chr);`
**Description:** This function decodes a chromosome to generate a robotic brain. The details of the procedure will, of course, vary from case to case.

### CreatePopulation

**Library:** EAFunctions
**Interface:** `pop = CreatePopulation(size, chromosomelength);`
**Description:** This function generates a population. All generated chromosomes are of equal length. Note that this function also sets the tournament size, tournament selection probability, crossover probability, and mutation rate.

### CreateSimulation

**Library:** RunFunctions
**Interface:** `sim = CreateSimulation(robot, arena);`
**Description:** `CreateSimulation` sets up a simulation, and also assigns default values to the time step length and the number of time steps.

### CreateStandardRobot

**Library:** RobotFunctions
**Interface:** `robot = CreateStandardRobot(brain);`
**Description:** `CreateStandardRobot` generates a standard robot, used in the evaluation of robotic brains.

### Crossover

**Library:** EAFunctions
**Interface:** `[chr1, chr2] = crossover(population,i1,i2);`
**Description:** This function carries out crossover between individuals `i1` and `i2` in the population.
**See also:** `MakeNewGeneration`

### GetBestIndividual

**Library:** EAFunctions
**Interface:** `ind = GetBestIndividual(population);`
**Description:** `GetBestIndividual` extracts the best individual from a population. It is assumed that fitness values are available for all individuals in the

population.
**See also:** `MakeNewGeneration`

### GetFitness

**Library:** RunFunctions
**Interface:** `f = GetFitness(simulation);`
**Description:** This function finalizes the fitness value of an individual. The detailed procedure will vary from case to case.

### MakeNewGeneration

**Library:** EAFunctions
**Interface:** `pop = MakeNewGeneration(population);`
**Description:** This function generates a new population, using elitism, (tournament) selection, crossover, and mutation.
**See also:** `Crossover, GetBestIndividual, Mutate, TournamentSelect`

### Mutate

**Library:** EAFunctions
**Interface:** `chr = Mutate(population,i);`
**Description:** `Mutate` mutates an individual, assuming that all genes take values in the range $[0, 1]$.
**See also:** `MakeNewGeneration`

### RunSimulation

**Library:** RunFunctions
**Interface:** `sim = RunSimulation(simulation);`
**Description:** `RunSimulation` runs a simulation, of a given robot, in a given arena. It is possible (but not always suitable) to update the fitness value in each step of the simulation.

### TournamentSelect

**Library:** EAFunctions
**Interface:** `i = TournamentSelect(population);`
**Description:** This function carries out tournament selection. Note that the tournament size can be set to an arbitrary value.
**See also:** `MakeNewGeneration`