Supplementary material for:
A scripted control system for autonomous hardware-timed experiments

P. T. Starkey, C. J. Billington, S. P. Johnstone, M. Jasperse, K. Helmerson, L. D. Turner, and R. P. Anderson
School of Physics, Monash University, Victoria 3800, Australia.

(Dated: 12 April 2013)

APPENDIX: GENETIC ALGORITHM USED BY THE MISE OPTIMIZER

Every genetic algorithm requires a genome representation, as well as methods of selection, recombination, and mutation[1]. Representation concerns the information stored in each individual. Selection is the process of choosing which individuals will be allowed to reproduce, based on their fitness. Recombination is the reproduction process itself, combining the genomes of parents to produce offspring, and mutation is applying random changes to those offspring. Here we describe these aspects of the genetic algorithm used by `mise`. The algorithm uses a variation on Pointed Directed mutation[2] (PoD), which biases mutations in directions that were previously successful. This allows faster traversal of uninteresting regions of parameter space by decreasing the likelihood of mutations that send the population back the way it came.

A. Genome representation

Each individual represents one point in the parameter space of the globals being traversed for optimization. As such, the *genome* of an individual comprises one floating point value for each global. Each global also has an associated *bias flag* stored along with it in the genome. This flag determines the direction in which mutations of that global are biased (see Sec. D).

B. Selection

`mise` selects individuals for reproduction based on their fitness. A higher fitness increases the probability of that individual being chosen for reproduction. `mise` first gives each individual a rank n, with the least fit individual having $n = 1$ and the most fit having $n = N$, where N is the population size. Individuals with fitness ranking n are selected from the probability distribution:

$$P(n) = \frac{2n}{N(N+1)}, \tag{1}$$

which favours the selection of more fit individuals. Two unique *parent* individuals are selected, which are recombined to produce a single offspring (see Sec. C). `mise` repeats this process (with replacement of the parents) until N offspring have been produced, comprising the next generation. Experiment shots are produced for these individuals and are executed. Only individuals in this new generation participate in the next round of selection; the previous generation is not used again.

C. Recombination

To produce a child, `mise` randomly chooses a single *crossover parameter* $0 \leq \alpha \leq 1$ to construct a weighted average of two parents:

$$\vec{P}_c = \alpha \vec{P}_a + (1-\alpha)\vec{P}_b, \tag{2}$$

where $\vec{P}_c$ is the vector of global values of the child, and $\vec{P}_a$ and $\vec{P}_b$ those of the parents. This method puts the child at a random point on the line joining its two parents in parameter space. The child inherits all of the mutation bias flags from its closest parent (a if $\alpha > 0.5$, otherwise b).

The main difference between this algorithm and PoD is that in PoD, each global and corresponding bias flag is exactly that of one of the parents, chosen at random, with no contribution from the other parent. PoD couples bias flags tightly with their globals, but does not couple the globals to each other. If the steepest fitness gradient is not along an axis of the parameter space, our method finds the optimum more efficiently. Without this modification, shots are wasted with individuals that are not directed towards the steepest ascent in fitness. By inheriting mutation bias flags from the closest parent, we still maintain some coupling between bias flags and global values.

D. Mutation

Another difference between our method and PoD is that we mutate the globals of every individual produced. For each global in a new individual, a value is drawn from a half-Gaussian distribution with standard deviation equal to the *mutation rate* for that global, set by the user in `runmanager` or `mise`. This value is either added or subtracted to the global based on the mutation bias flag. Mutations are prevented from exceeding the boundaries of the parameter space set in `runmanager` or `mise`. The bias flag is then flipped with probability $1/N$, affecting the direction of mutation in the next generation.

[1] T. Bäck and H.-P. Schwefel, Evol. Comput. **1**, 1–23 (1993).
[2] A. Berry and P. Vamplew, in *AISAT 2004: The 2nd International Conference on Artificial Intelligence in Science and Technology* (Hobart, Tasmania, Australia, 2004) pp. 200–205.