The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Dealing with Rounding Error Problems in Evolutionary Physical Simulation

Marcin L. Pilat[1], Reiji Suzuki[1], Takaya Arita[1]

[1]Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
(pilat@alife.cs.is.nagoya-u.ac.jp)

Abstract: This paper introduces the problem of floating-point rounding errors in physical simulation. A simple virtual creature is simulated in a physical environment for a specified number of time steps. The effect of rounding errors is illustrated by varying the initial position of the creature which causes a change in the fitness value computed by a simple distance-based fitness function. With a large evaluation time, these rounding errors can produce significantly large differences in fitness. A discussion is provided on the importance of this finding for evolutionary simulations, including suggestions to alleviate the problem.

Keywords: physical simulation, floating point, rounding errors, evolutionary computation, artificial life

1 INTRODUCTION

Physical simulation is gaining popularity in the artificial life community. It enables experimentation with physical models which reflect the physical nature of biological organisms. Such experiments are vital in real-world applied robotics research. Furthermore, physical simulation allows for a complex simulation environment, important for the study of theoretical problems.

Physics engines are made for the video gaming market and are not meant to be used for accurate scientific simulation. However, they are valuable as they offer an approximation to more accurate physical simulation but at real-time speeds. We present and discuss problems that occur when using video game grade physics engines for scientific simulation, focusing on the problem of floating-point rounding errors. We simulate a simple virtual creature in a simple physical environment and demonstrate the rounding error problem on fitness evaluation by varying the creature's initial position.

We show that the fitness performance is highly sensitive to small variations in the creature's initial position producing significant variation in fitness values. These rounding errors occur after only a few simulation steps and increase drastically over simulation time. While not generally destructive, these errors can impact the performance of evolutionary algorithms that use fitness-proportional selection. Having identified the problem and presented evidence of its occurrence, we offer solutions that can alleviate it.

2 TEST MODEL AND ENVIRONMENT

Floating-point is the most common method to represent real numbers in computers. However, it is often poorly understood by software developers and researchers alike [1]. The most common floating-point representation currently used is defined by the IEEE 754 standard. There are mainly two sources of error when using floating-point representation: not all real numbers can be exactly represented and rounding error is inherent in floating-point computation.

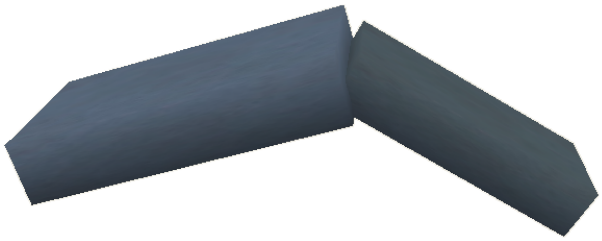


Fig. 1: Hand-crafted virtual creature used in the presented experiments composed of two bodies connected with a hinge joint. Neural network controller not shown.

The virtual creature model provides a simple yet powerful model for the study of evolution of morphologies and controllers for tasks such as locomotion [2] and light following [3]. We use a simple hand-crafted virtual creature, using the virtual creature model described in [2], composed of two blocks of different sizes joined with a hinge joint (shown in Fig. 1). The neural network consists of a sine wave generator neuron feeding into an effector allowing the smaller body part to rotate around the hinge joint.

The testing environment is composed of a simple plane simulation surface with friction. Virtual creatures are initially positioned above the surface and are allowed to drop due to the gravitational force. Once a creature rests on the ground surface and is stable from movement, the neural network of the creature is turned on which makes the creature move along the surface. We use a simple fitness function, measuring fitness as a simple displacement from the original resting position to the final resting position, to demonstrate the rounding error problem.

Physical simulation is performed using the Morphid

Academy software [2]. Morphid Academy is a virtual laboratory for the evolution of functional forms called Morphids. The software features physical simulation using the Open Dynamics Engine[1] (ODE) and NVIDIA PhysX[2] engines, graphical visualization through the OGRE graphics engine, and an ability to perform structured evolutionary experiments with a built-in genetic algorithm.

3 EXPERIMENTS AND RESULTS

We executed a suite of creature evaluation experiments to showcase the problem with physical simulation due to floating-point rounding errors. These experiments only performed creature simulation and fitness evaluation and did not have an evolutionary component. The position and evaluation time experiments were run using the NVIDIA PhysX engine while the precision experiments used ODE. The evaluated creature was recreated for each evaluation to prevent accumulation of internal state errors. For each evaluation, the creature was located at a certain (x,y) coordinate, along the simulation plane. The z-coordinate was kept constant.

3.1 Position

The initial creature position experimental results demonstrate the main problem of rounding errors in the floating-point representation producing unexpected behavior in physical simulations. Physics engines generally produce deterministic behavior (at least on the same hardware). When a fixed initial position is used (as in most virtual creature evolution systems), the behavior of a virtual creature is the same over a number of evaluations. However, modifying the initial position by a small amount can produce chaotic behavior with drastically different fitness values.

Fig. 2c illustrates this phenomenon showing the fitness values of the evaluated creature for 10000 initial positions centered around the origin and offset by multiples of a fixed position delta $\delta_x = 0.001$ along the x-axis. It is expected that the fitness will be independent of the initial x position value, producing a plot with a straight line. However, the fitness values vary greatly with a somewhat random distribution.

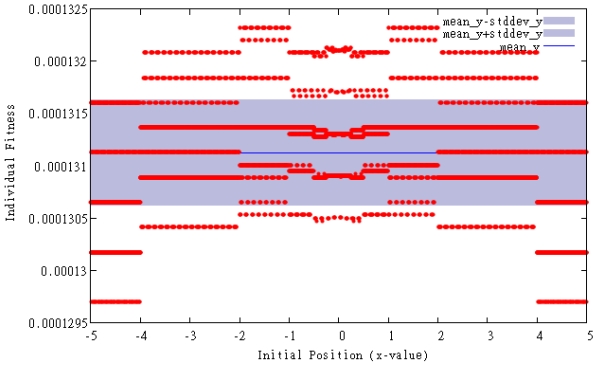
Additional experiments were performed varying the value of δ_x (not shown). Experiments with smaller values of δ_x have similar fitness behavior. With larger δ_x, somewhat periodic patterns of fitness values can be seen, which is likely due to the larger position values (in 100s). Similar results are observed varying δ_y along the y-axis.

3.2 Evaluation Time

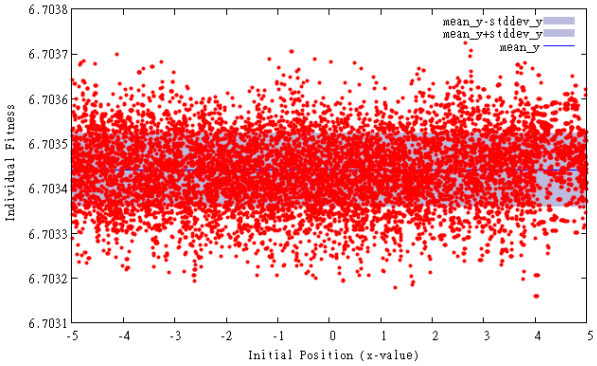
Evaluation time is important for physical simulation since adequate evaluation time is required for the occurrence of certain behaviors. Evaluation time is expressed in terms of evaluation steps where each step is equal to one step of the physics engine. Fig. 2 shows the dependence of the evaluation time of $10, 1000$, and 10000 on the error propagation in fitness values using $\delta_x = 0.001$.

These results indicate that the problem of rounding errors occurs quite early with only a few number of steps (10 evaluation steps in Fig. 2a). The rounding errors propagate over

(a) $t = 10$; mean = 1.311×10^{-4}, std dev = 5.033×10^{-7}



(b) $t = 1000$; mean = 6.70344, std dev = 0.00008



(c) $t = 10000$; mean = 54.25, std dev = 34.89

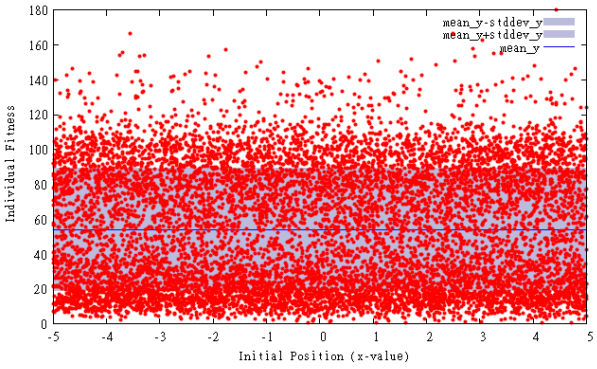


Fig. 2: Results of the evaluation time experiments with evaluation time t of 10 (a), 1000 (b), and 10000 (c). The x-coordinate of the initial creature position is plotted against the computed fitness value. Mean and standard deviation are displayed for each plot.

[1]ODE is available at `http://www.ode.org` under an open source license.

[2]The PhysX SDK is available for free from NVIDIA Corporation at `http://developer.nvidia.com/physx`.

evaluation time to a point where they produce a large range of results (as seen in Figs. 2b and 2c).

3.3 Precision

We run similar fitness evaluation experiments using the single and double precision versions of ODE with various evaluation time settings. ODE was used for these experiments since the NVIDIA PhysX SDK is only available in single precision. Selected results are shown in Fig. 3.

The results for smaller values of evaluation time (100 and 1000) show a larger dispersion of values using single precision. On the other hand, double precision produces more irregular fitness behavior within range $[-2, 2]$ (see Fig. 3a vs. Fig. 3b). With evaluation time of 10000, the single and double precision results are very similar with significantly large standard deviation values as demonstrated in Fig. 3c.

From these results, it is difficult to conclude whether the double precision fitness results are, in general, better than the ones with single precision, especially for a higher number of evaluation steps. It is interesting to note that while the mean fitness values for single and double precision are very similar with 100 and 1000 time steps, they are quite different with 10000 time steps (e.g., means of 96.434 vs. 69.447).

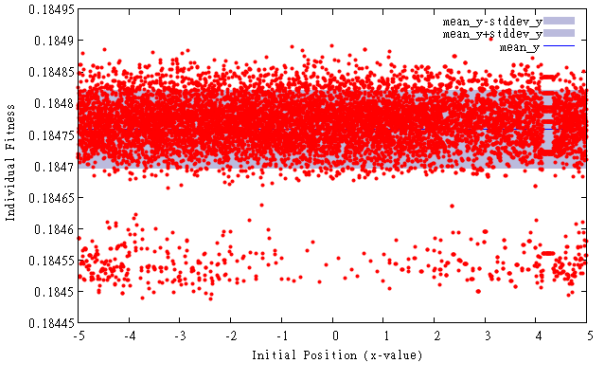
4 DISCUSSION

The floating-point rounding errors, presented by the examples in the previous section, are not necessarily a disadvantage of physics engines. The real world is inherently chaotic and capturing this behavior in simulations allows for results that appear more realistic and can be more suited to the real world. Artificial life simulations, especially involving a large number of interacting entities, are often criticized to be too orderly and thus not able to evolve novelty. Chaotic systems, such as physics engines, can be of benefit in this domain since they introduce inherent perturbations. Physically simulated ecosystems of evolving entities are a promising future for artificial life research.

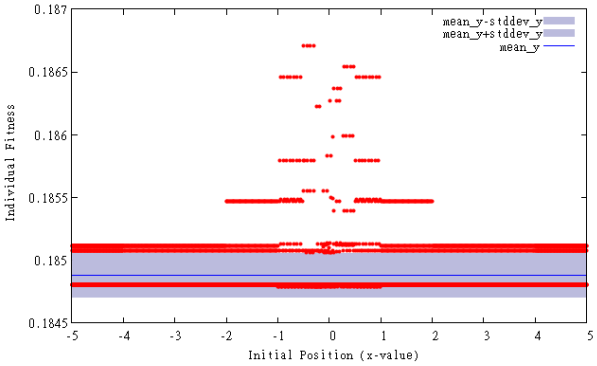
The effect of this chaotic behavior has to be considered for each simulation since it can have a strong effect on the performance of the system and on the acquired results. In an evolutionary system where the fitness value decides on the reproductive strength of an individual, it needs to be calculated accurately for the algorithm to be effective. If the fitness of an individual significantly varies between different evaluations, the evolutionary algorithm might not select the individual for reproduction even though the individual is generally fit.

Fig. 4 illustrates the effect of the position-based rounding error problem on the performance of a genetic algorithm using the simple displacement fitness function. The variation in fitness values can be seen on the best of population line (in green). Furthermore, the evolutionary algorithm performance is decreased since good individuals can be eliminated

(a) single, $t = 1000$; mean 0.18475, std dev = 0.00006



(b) double, $t = 1000$; mean 0.18487, std dev = 0.00017



(c) single, $t = 10000$; mean 96.434, std dev = 56.544

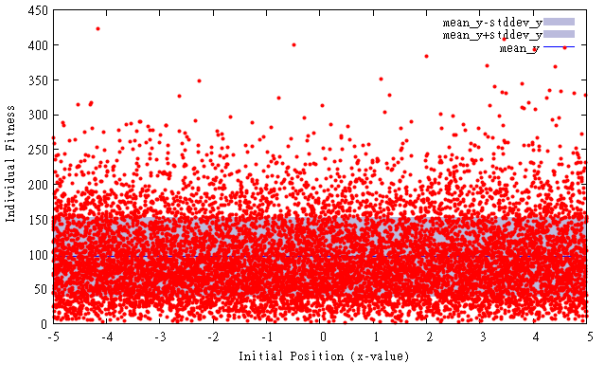


Fig. 3: Results of the precision experiments with single (a,c) and double (b) precision and evaluation time t of 1000 (a,b) and 10000 (c). The x-coordinate of the initial creature position is plotted against the computed fitness value. Mean and standard deviation are displayed for each plot.

from the population by worse performing individuals due to a significant variation in fitness values over different evaluations.

The fitness values and their spread are greatly dependent on the fitness function used. Thus, care must be taken to craft a fitness function where the position-dependent fitness variation of a fit individual is still able to distinguish it from an unfit individual. The simple distance-based locomotion fitness

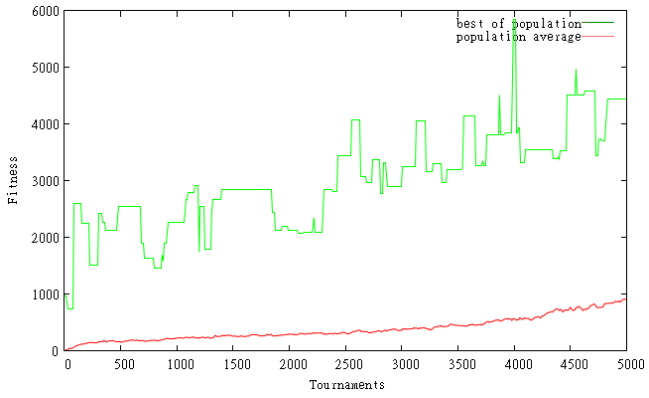


Fig. 4: Fitness plot of an evolutionary run with virtual creatures using the locomotion distance fitness function. The initial position of each evaluated creature is randomized. The best of population (in green) and population average (in red) are shown.

function used in the presented experiments was specifically chosen to demonstrate the rounding error and is not a good choice of a fitness function to alleviate the problem.

A simple solution to the initial position-based rounding error problem is to evaluate each individual at the same initial position. This idea works well with a simple evolutionary scenario where the evaluation of each virtual creature is separate. However, it is not well suited for co-evolutionary scenarios where multiple virtual creatures need to be evaluated in the same environment. Furthermore, a slight variation in initial position can prevent problems with the evolutionary system exploiting the position and can generally be beneficial if the effect of rounding errors can be minimized.

The floating-point precision setting can improve the results as we found the double precision performance in ODE more stable. However, with a large evaluation time, the problem is still significant. For closed source engines, such as NVIDIA PhysX, the precision is fixed and cannot be modified by the user. To alleviate the problem further, the floating-point representation can be replaced with a fixed-point representation if the range of required values does not need to be large. Fixed-point representation can provide more stable behavior but we are currently not aware of any free 3D physics engines that use a fixed-point representation.

A more important problem with floating-point arithmetic is that it is not guaranteed to produce the same results on different computer architectures and across different compilers or optimization settings as detailed in [4]. This can severely affect the reproducibility of results over different machines. The problem can be solved to an extent but solving it is considered to be very difficult [5].

5 CONCLUSIONS

We presented experimental results showing the effect of floating-point rounding errors on a sample fitness calculation of a virtual creature simulated using a physics engine. The variation of fitness values due to the difference in initial position of evaluation is evident and continues to increase with the evaluation time. The variation is significant on the time scale that is typically used for evolutionary experiments such as the virtual creature experiments presented.

This chaotic behavior can add beneficial randomness to the simulations but needs to be considered, especially for evolutionary experiments where the survival of an individual is directly related to its fitness value as compared with other individuals. In these evolutionary scenarios, a wrong choice of fitness function can have a negative effect on the evolutionary performance. We identified several possible methods of alleviating the floating-point rounding error problem.

The presented issue raises some important directions for future research. Can and should floating-point rounding errors be eliminated in physical simulations? Can evolutionary systems using well-crafted fitness functions harness the chaotic power of physics engines? How does this problem affect an evolutionary ecosystem simulated with a physics engine? The answers to these questions can situate physical simulation as a basis for future artificial life simulation systems studying open-ended evolution.

REFERENCES

[1] Goldberg, D., What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Computing Surveys*, 23 (1), pp. 5–48, March 1991.

[2] Pilat, M. L. and Jacob, C., Creature Academy: A System for Virtual Creature Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008)*, pp. 3289–3297, IEEE, 2008.

[3] Pilat, M. L. and Jacob C., Evolution of Vision Capabilities in Embodied Virtual Creatures. In *Proceedings of the 12th annual Conference on Genetic and Evolutionary Computation Conference (GECCO 2010)*, pp. 95–102, ACM, 2010.

[4] Corden, M. J. and Kreitzer D., Consistency of Floating-Point Results using the Intel Compiler or Why doesnt my application always give the same answer? Published online: `http://software.intel.com/file/32018/`. Retrieved: September 2011.

[5] Fiedler, G., Floating Point Determinism. Published online: `http://gafferongames.com/networking-for-game-programmers/floating-point-determinism/`. Retrieved: September 2011.