Available online at www.sciencedirect.com
SCIENCE DIRECT®
Environmental Modelling & Software 21 (2006) 283–309
Environmental Modelling & Software
www.elsevier.com/locate/envsoft

What every agent-based modeller should know about floating point arithmetic

J. Gary Polhill*, Luis R. Izquierdo, Nicholas M. Gotts

Macaulay Institute, Craigiebuckler, Aberdeen AB15 8QH, UK

Received 14 June 2004; received in revised form 10 August 2004; accepted 27 October 2004
Available online 22 January 2005

Abstract

Floating point arithmetic is a subject all too often ignored, yet, for agent-based models in particular, it has the potential to create misleading results, and even to influence emergent outcomes of the model. Using a simple demonstration model, this paper illustrates the problems that accumulated floating point errors can cause, and compares a number of techniques that might be used to address them. We show that inexact representation of parameter values, imprecision in calculation results, and differing implementations of mathematical expressions can significantly influence the behaviour of the model, and create issues for replicating results, though they do not necessarily do so. None of the techniques offer a failsafe approach that can be applied in any situation, though interval arithmetic is the most promising.

Keywords: Agent-based modelling; Emergence; Floating point arithmetic; Interval arithmetic

Software availability

Name: charity-1.1
Developers: Gary Polhill and Luis Izquierdo, Macaulay Institute, Craigiebuckler, Aberdeen, UK
Telephone, fax, and email: As above, plus l.izquierdo@macaulay.ac.uk, n.gotts@macaulay.ac.uk
Year first available: 2004
Hardware required: Intel PC or Sun Sparc, other platforms may work with Linux
Software required: Swarm 2.1.1 (PC, Windows 2K), Swarm 2.2 pretest 10 (PC, Windows XP) or Swarm 2001-12-18 (Sun, Solaris 8), fully implemented IEEE library functions
Program language: Objective-C
Availability: On-line at http://www.macaulay.ac.uk/fearlus/floating-point/charity-world/
Licence: GNU GPL
Cost: Free

1. Introduction

Agent-based modelling is a technique with growing popularity that has been applied to a diverse range of environmental applications. One of the classics is Lansing and Kremer's (1993) work on Balinese water temples, establishing a pedigree for the modelling of various water-related scenarios that has continued to the present day, as exemplified by authors such as Feuillette et al. (2003) and Pahl-Wostl (2005). At a more abstract level, agent-based modelling has also been applied to resource sharing (Rouchier et al., 2001) and common-pool resource dilemmas (e.g. Izquierdo et al., 2004; for a review see Gotts et al., 2003c; and see also CIRAD's CORMAS platform: Bousquet et al., 1998). It has been

* Corresponding author. Tel.: +44 1224 498200; fax: +44 1224 311557.
E-mail address: g.polhill@macaulay.ac.uk (J.G. Polhill).

1364-8152/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2004.10.011

used to investigate trapping strategies for cowbirds (*Molothrus ater*) (Harper et al., 2002) and to study forestry processes in Indiana (Hoffmann et al., 2002). Hare and Deadman (2004) and Bousquet and Le Page (2004) review various agent-based models in environmental and ecosystem management. The issue of numerics in such models has, however, not been covered in any great depth.

Floating point arithmetic is the standard way to represent and work with non-integer numbers in a digital computer. It is designed to create the illusion of working with real numbers in a machine that can only strictly work with a finite set of numbers. Some programming languages (FORTRAN, for example) use the `real` keyword to declare floating point variables. Thinking of floating point numbers as real numbers, however, is incorrect, and we show here that the differences have the potential to create seriously misleading results in research using agent-based models.

Rather than thinking of floating point numbers as reals, it is better to regard them as the product of an integer and an integer power of an integer base – a subset of rational numbers. For example, the normalised IEEE 754 single precision floating point numbers (IEEE, 1985) can be expressed as the product of a 24 bit integer and a power of two in the range −149 to +104 inclusive. Single precision IEEE 754 floating point numbers give between 6 and 9 significant decimal digits of accuracy, and double precision 15–17 significant decimal digits of accuracy (Sun Microsystems, 2001, p. 27).

One might argue that few models require as much as 15 significant figures of accuracy – after all, particularly in the case of agent-based models of social systems, it is unlikely that any measured parameter used as input to the model will have that level of accuracy. However, the agents in such models typically exhibit highly nonlinear behaviour, in that their course of action can depend on comparing variable values with precise thresholds. The behaviour of the model could depend on these values being calculated and compared accurately to ensure that the correct action is always selected.

While such sensitive dependence on correct calculations may be particularly important in agent-based modelling – because real-world decision-makers frequently choose between small sets of discrete alternatives with very different consequences – it may certainly occur in models of other kinds. An obvious example is mathematical models of chaotic systems, such as Lorenz's work with weather forecasting (Gleick, 1988, chapter 1), in which radically different behaviour was observed when starting a run half way through using parameters with three significant figures from a printout instead of the six stored in the computer's memory. This is a good analogy for what is happening in every calculation using floating point arithmetic.

Many programmers are already aware of some of the issues with floating point numbers, though as Fernandez et al. (2003) point out, "floating point representation still remains shrouded in mystery by the average computational science student, and only well understood by experts". Not testing for equality of floating point numbers is a commonly-taught "best practice" heuristic (Knuth, 1998, p. 233), and some compilers feature warning options to check for this (e.g. -Wfloat-equal in GNU's gcc (Stallman, 2001)). Where there are concerns about floating point accuracy, particularly in the use of comparison operators, another heuristic is to use small constants added to one or other of the terms to allow for the possibility of some loss of accuracy in floating point calculations. Neither of these heuristics, however, is sufficient to guarantee that an agent-based model using floating point numbers will be free from unintended effects arising purely from floating point arithmetic.

Edmonds and Hales (2003) have shown the potential for authors to draw the wrong conclusion from their models because of implementation-specific artefacts, and recommend that models be reimplemented to check results. They refer, however, to the complex and potentially ambiguous process of translating a model described in natural language into a computer program. Issues with floating point numbers occur at a different level. Even if a reimplemented model confirms the results of the original, both models could be reporting the wrong results because of accumulated errors arising from computations based on parameters that are not exactly representable using floating point numbers.

As will be shown below, many of the workarounds used in the field to try and cope with or avoid floating point errors are not safe, and thus may create a false sense of security. However, it will also be shown that the IEEE 754 standard stipulates facilities that, if properly used, do permit certainty that a model has not incurred floating point errors that could cause agents to act inappropriately. This places somewhat less of a burden on programmers than infinite accuracy – they need only familiarise themselves with the functions provided by any conforming platform to access these facilities. However, we downloaded 10 publicly available agent-based models[1] written in C or Objective-C, and though all of them included either single or double precision floating point variables, none included the header file

[1] Arborgames (ftp://ftp.swarm.org/pub/swarm/apps/objc/contrib/arborgames-2.1.141-Swarm2.1.141.tar.gz), Biofilm and Colony (http://www.theobio.uni-bonn.de/people/jan_kreft/bacsim.html), The Artificial Stock Market in Swarm (http://prdownloads.sourceforge.net/artstkmkt/), Echo (http://www.santafe.edu/projects/echo/), MAG (http://alife.tuke.sk/projekty/mag_html/mag_intro.html), SCL (http://www.eeng.dcu.ie/~mcmullin/), SLUCE (http://www.cscs.umich.edu/research/projects/sluce/), Tierra (http://www.isd.atr.co.jp/~ray/tierra/) and Village (http://www.santafe.edu/projects/swarm/users/Pages/Village/village.html).