Testing 4x4 matrix inversion precision [2]

It is extremely rare that a hobby software project of mine gets completed, but now it has happened. Behold! Fourbyfour!

Have you ever had to implement a mathematical algorithm, say, matrix inversion? You want it to be fast and measuring the speed is fairly simple, right. But what about correctness? Or precision? Behavior around inputs that are on the edge? You can hand-pick a few example inputs, put those into your test suite, and verify the result is what you expect. If you do not pick only trivial inputs, this is usually enough to guarantee your algorithm does not have fundamental mistakes. But what about those almost invalid inputs, can you trust your algorithm to not go haywire on them? How close to invalid can your inputs be before things break down? Does your algorithm know when it stops working and tell you?

Inverting a square matrix requires that the inverse matrix exists to begin with. Matrices that do not mathematically have an inverse matrix are called singular. Can your matrix inversion algorithm tell you when you are trying to invert a matrix that cannot be inverted, or does it just give you a bad result pretending it is ok?

Working with computers often means working with floating-point numbers. With floating-point, the usual mathematics is not enough, it can actually break down. You calculate something and the result a computer gives you is total nonsense, like 1+2=2 in spirit. In the case of matrix inversion, it's not enough that the input matrix is not singular mathematically, it needs to be "nice enough" numerically as well. How do you test your matrix inversion algorithm with this in mind?

These questions I tried to answer with Fourbyfour. The README has the links to the subpages discussing how I solved this, so I will not repeat it here. However, as the TL;DR, if there is one thing you should remember, it is this:
Getting started with COBOL development on Fedora Linux 33

Though its popularity has waned, COBOL is still powering business critical operations within many major organizations. As the need to update, upgrade and troubleshoot these applications grows, so may the demand for anyone with COBOL development knowledge.

Fedora 33 represents an excellent platform for COBOL development. This article will detail how to install and configure tools, as well as compile and run a COBOL program.

3 Excellent Free Books to Learn about ClojureScript

ClojureScript is a compiler for Clojure that targets JavaScript. It emits JavaScript code which is compatible with the advanced compilation mode of the Google Closure optimizing compiler.

Clojure is a dialect of the Lisp programming language. It’s a well-rounded language. It offers broad library support and runs on multiple operating systems. Clojure is a dynamic functional general purpose programming language that runs on the Java platform, combining the approachability and interactive development of a scripting language with an efficient and robust infrastructure for multi-threaded programming. Clojure features a rich set of immutable, persistent data structures, first-class functions and dynamic typing. Clojure programs are composed of expressions and written in terms of abstractions.

ISO 8601: the better date format

If you haven’t been living under a rock, you’ve probably heard that there are different date formats in the world such as the American one (mm/dd/yyyy) and the European one (dd.mm.yyyy). If you’re smart enough, you’ve probably also noticed that the American one makes no sense and is just awful. A simple conclusion that many people draw out of this is that the European format is the best one, however I don’t think this is true. If you’re one of these people who think so, I’m here to (hopefully) change your mind by introducing you to a lesser-known date format called ISO 8601.

Development

Source URL: http://www.tuxmachines.org/node/148219

Links: