Invert 4x4 matrix - Numerical most stable solution needed

Asked 16 years, 2 months ago Modified 9 years, 10 months ago

Viewed 13k times

13

I want to invert a 4x4 matrix. My numbers are stored in fixed-point format (1.15.16 to be exact).

With floating-point arithmetic I usually just build the adjoint matrix and divide by the determinant (e.g. brute force the solution). That worked for me so far, but when dealing with fixed point numbers I get an unacceptable precision loss due to all of the multiplications used.

Note: In fixed point arithmetic I always throw away some of the least significant bits of immediate results.

So - What's the most numerical stable way to invert a matrix? I don't mind much about the performance, but simply going to floating-point would be to slow on my target architecture.

language-agnostic matrix linear-algebra fixed-point matrix-inverse

Share

Improve this question

edited Jul 3, 2013 at 15:52

Follow

Amro
125k ● 25 ● 247 ● 461

asked Oct 1, 2008 at 0:08
Nils Pipenbrinck
86.2k ● 33 ● 155 ● 223

Plain old Gaussian elimination would work well. It depends on what libraries/classes/structures you're using. You could take a look at the GSL. – axblount Oct 1, 2008 at 0:18

are the magnitudes of the elements in you matrix close in magnitude? – David Nehme Oct 1, 2008 at 0:19

To minimize truncation errors and other badness, use "pivoting" - see the chapter on inverting matrices in Numerical Recipes. They have the best explanation i've found so far. – DarenW Oct 1, 2008 at 0:22

No - unfortunately they are all over the place.
– Nils Pipenbrinck Oct 1, 2008 at 0:30

Do you have an approximate condition number for the matrix? The paper I cite in my answer has success up to a condition number of few hundred, though this is for 8x8 or 32x32 matrices so you may do better than this.
– Chris Johnson Oct 1, 2008 at 0:49

6 Answers

Sorted by: Highest score (default)

19

Meta-answer: Is it really a general 4x4 matrix? If your matrix has a special form, then there are direct formulas for inverting that would be fast and keep your operation count down.

For example, if it's a standard homogenous coordinate transform from graphics, like:

```
[ux vx wx tx]
[uy vy wy ty]
[uz vz wz tz]
[ 0  0  0  1]
```

(assuming a composition of rotation, scale, translation matrices)

then there's an easily-derivable direct formula, which is

```
[ux uy uz -dot(u,t)]
[vx vy vz -dot(v,t)]
[wx wy wz -dot(w,t)]
[ 0  0  0     1    ]
```

(ASCII matrices stolen from the linked page.)

You probably can't beat that for loss of precision in fixed point.

If your matrix comes from some domain where you know it has more structure, then there's likely to be an easy answer.

Share Improve this answer Follow

edited Feb 1, 2015 at 8:19
Yoav
6,098 ● 5 ● 41 ● 63

answered Oct 1, 2008 at 0:24
Adrian
1,862 ● 13 ● 25

5 I presume this doesn't work if there are scaling factors involved? – Alnitak Mar 29, 2012 at 7:35

6

I think the answer to this depends on the exact form of the matrix. A standard decomposition method (LU, QR, Cholesky etc.) with pivoting (an essential) is fairly good on fixed point, especially for a small 4x4 matrix. See the book 'Numerical Recipes' by Press et al. for a description of these methods.

This paper gives some useful algorithms, but is behind a paywall unfortunately. They recommend a (pivoted) Cholesky decomposition with some additional features too complicated to list here.

Share Improve this answer Follow

edited Jan 28, 2015 at 9:02

answered Oct 1, 2008 at 0:23

Chris Johnson
10.7k ● 4 ● 33 ● 35

5

I'd like to second the question Jason S raised: are you certain that you need to invert your matrix? This is almost never necessary. Not only that, it is often a bad idea. If you need to solve Ax = b, it is more numerically stable to solve the system directly than to multiply b by A inverse.

Even if you have to solve Ax = b over and over for many values of b, it's still not a good idea to invert A. You can *factor* A (say LU factorization or Cholesky factorization) and save the factors so you're not redoing that work every time, but you'd still solve the system each time using the factorization.

Share Improve this answer

Follow

answered Dec 10, 2008 at 15:02

John D. Cook

30k ● 10 ● 69 ● 94

1

You might consider doubling to 1.31 before doing your normal algorithm. It'll double the number of multiplications, but you're doing a matrix invert and anything you do is going to be pretty tied to the multiplier in your processor.

For anyone interested in finding the equations for a 4x4 invert, you can use a symbolic math package to resolve them for you. The TI-89 will do it even, although it'll take several minutes.

If you give us an idea of what the matrix invert does for you, and how it fits in with the rest of your processing we might be able to suggest alternatives.

-Adam

Share Improve this answer

Follow

answered Oct 1, 2008 at 0:36

Adam Davis

93.5k ● 60 ● 271 ● 333

Use a symbolic math program to invert a generic matrix and it should supply formulas that you can compute much more easily. – Karl Dec 10, 2008 at 15:07

1

Let me ask a different question: do you definitely need to invert the matrix (call it M), or do you need to use the matrix inverse to solve other equations? (e.g. Mx = b for known M, b) Often there are other ways to do this w/o explicitly needing to calculate the inverse. Or if the matrix M is a function of time & it changes slowly then you could calculate the full inverse once, & there are iterative ways to update it.

Share Improve this answer Follow

answered Dec 10, 2008 at 14:48

Jason S
189k ● 171 ● 630 ● 995

-3

If the matrix represents an affine transformation (many times this is the case with 4x4 matrices so long as you don't introduce a scaling component) the inverse is simply the transpose of the upper 3x3 rotation part with the last column negated. Obviously if you require a generalized solution then looking into Gaussian elimination is probably the easiest.

Share Improve this answer Follow

answered Oct 1, 2008 at 0:29

Ron Warholic
10.1k ● 33 ● 47

This answer is not true. See Adrian's anwer for a correct one, going in the same direction. – Suma May 2, 2009 at 11:29