ASTR415: Problem Set #2

Curran D. Muhlberger
University of Maryland
(Dated: March 8, 2007)

In this problem set, the effects of roundoff error in various algorithms were explored. In particular, it was confirmed that the subtraction of two floating-point numbers of comparable magnitude resulted in significant roundoff error. In single-precision, this caused a 22% inaccuracy in results obtained using the classic quadratic formula and a 168% inaccuracy after 20 iterations of a recurrence relation to find powers of the silver ratio. Double-precision greatly lessened the magnitude of these errors.

Techniques in numerical linear algebra were also explored in finding the period of a rigid body defined by nearly 2000 discrete points of equal mass. Using Cholesky decomposition to solve a linear system relating the angular momentum, the inertia tensor, and the spin vector, the spin period of the object was found to be 4.3 hours. The object was also animated in 3D to aid in visualization.

I. STABILITY OF THE QUADRATIC FORMULA

A general quadratic equation can be expressed can be expressed as

$$ax^2 + bx + c = 0 \tag{1}$$

where $a, b, c \in \mathbb{R}$. The roots of this equation can be found explicitly using the quadratic formula, which is usually expressed as

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

The polynomial expressed in equation 1 can also be written as

$$a(x - x_1)(x - x_2) = 0 \tag{3}$$

Expanding equation 3 and matching coefficients with equation 1, one obtains expressions relating the elementary symmetric polynomials in the roots to the coefficients a, b, and c. In particular, $x_1 x_2 = c$. Therefore, if the larger root x_1 is known, the smaller root can be calculated as $x_2 = c/x_1$.

Consider the quadratic polynomial

$$x^2 - 4999x + 1 = 0 \tag{4}$$

Since $b^2 \gg 4ac$, the numerator in equation 2 will involve the subtraction of two nearly equal numbers when solving for the smaller root x_2. This greatly magnifies roundoff error since the difference will have far less precision than the two operands. Thus, the smaller root will suffer in accuracy. An alternate means of finding x_2 would be $x_2 = 1/x_1$. Since this does not involve the subtraction of two nearly equal numbers, the result will be more accurate than that obtained via equation 2.

To compare these two accuracies, a program `quadraticsolver` was written to calculate the roots of equation 4. The larger root was calculated using equation 2, while the smaller root was calculated both using equation 2 and the fact that $x_1 x_2 = 1$. This was done in both single- and double-precision. The absolute and relative errors between the two calculations of x_2 were displayed. The results are shown in table I.

In single-precision, the roundoff errors introduced using the quadratic formula are quite significant (22%). Finding the smaller root via $1/x_1$ is clearly the preferred method. In double-precision, far less precision is lost in the subtraction,

TABLE I: Comparison of techniques for finding the smaller root of a quadratic equation. x_{2A} was found using equation 2, while x_{2B} was found via $1/x_1$.

Precision	x_1	x_{2A}	x_{2B}	$x_{2A} - x_{2B}$	Relative Error
Single	4999.000000	0.000244	0.000200	4.41×10^{-5}	2.20×10^{-1}
Double	4998.999800	0.000200	0.000200	-1.09×10^{-13}	-5.43×10^{-10}

TABLE II: Single-precision powers of the silver ratio.

n	Φ^n (Recurrence)	Φ^n (Direct)	Relative error
0	1.00000000	1.00000000	0.000000E+00
1	0.61803401	0.61803401	0.000000E+00
2	0.38196599	0.38196602	-7.802349E-08
3	0.23606801	0.23606800	6.312233E-08
4	0.14589798	0.14589804	-4.085363E-07
5	0.09017003	0.09016995	8.262820E-07
6	0.05572796	0.05572810	-2.473362E-06
7	0.03444207	0.03444186	6.057056E-06
8	0.02128589	0.02128624	-1.636337E-05
9	0.01315618	0.01315562	4.226328E-05
10	0.00812972	0.00813062	-1.112233E-04
11	0.00502646	0.00502500	2.905171E-04
12	0.00310326	0.00310562	-7.614027E-04
13	0.00192320	0.00191938	1.992377E-03
14	0.00118005	0.00118624	-5.217307E-03
15	0.00074315	0.00073314	1.365773E-02
16	0.00043690	0.00045310	-3.575776E-02
17	0.00030625	0.00028003	9.361357E-02
18	0.00013065	0.00017307	-2.450851E-01
19	0.00017560	0.00010696	6.416395E-01
20	-0.00004494	0.00006611	-1.679836E+00

and the relative error is much lower. Still, it is six orders of magnitude greater than $e_m \sim 10^{-16}$. Finding the root via $1/x_1$ is always more accurate, and furthermore it is computationally less intensive.

In *Numerical Recipes in FORTRAN 77* [1], an alternate form of the quadratic formula is given that minimizes roundoff error. This technique was implemented in the C version of `quadraticsolver`, and the results were identical (to machine precision) to those obtained by $x_2 = 1/x_1$.

II. STABILITY OF RECURRENCE RELATIONS

The "golden ratio conjugate" Φ (also called the "silver ratio" [2]) is defined as

$$\Phi = \frac{1}{\phi} = \frac{\sqrt{5}-1}{2} \approx 0.6180339887$$

where ϕ is the more common "golden ratio" equal to $(\sqrt{5}+1)/2$. It obeys the 3-term linear recurrence relation

$$\Phi^{n+1} + \Phi^n - \Phi^{n-1} = 0 \tag{5}$$

This recurrance relation has two linearly independent solutions, namely Φ and $-\phi$ [1]. In this case, for increasing n, the first solution is "minimal" and the second solution is "dominant," meaning that roundoff errors introduced when evaluating the recurrence will cause the recurrence to tend exponentially towards the second solution.

Given the initial conditions $\Phi^0 = 1$ and $\Phi^1 = (\sqrt{5}-1)/2$, the program `goldenpowers` will evaluate Φ^i for $i \leq 20$ using both the recurrence relation given by equation 5 and using successive multiplication (that is, $\Phi^{n+1} = \Phi\Phi^n$). This is done in both single- and double-precision. The results are shown in tables II & III.

The roundoff error is not random. While the sign of the error alternates, the magnitude of the error grows exponentially by a factor of 10 every two iterations, regardless of the precision. After 20 rounds, the results using double-precision are of comparable accuracy to those of the quadratic formula in problem 1, but this accuracy is still 6 orders of magnitude less than e_m, and for large n the results would quickly become worthless. In single-precision, the relative error reaches 10% after 17 rounds, and for $n = 20$ the error is 168%, producing a negative number for a power of a positive number. This is clearly ridiculous.

TABLE III: Double-precision powers of the silver ratio.

n	Φ^n (Recurrence)	Φ^n (Direct)	Relative error
0	1.00000000	1.00000000	0.000000E+00
1	0.61803399	0.61803399	0.000000E+00
2	0.38196601	0.38196601	-2.906602E-16
3	0.23606798	0.23606798	2.351490E-16
4	0.14589803	0.14589803	-1.521916E-15
5	0.09016994	0.09016994	2.462512E-15
6	0.05572809	0.05572809	-8.466911E-15
7	0.03444185	0.03444185	1.974376E-14
8	0.02128624	0.02128624	-5.443871E-14
9	0.01315562	0.01315562	1.395098E-13
10	0.00813062	0.00813062	-3.684673E-13
11	0.00502500	0.00502500	9.610888E-13
12	0.00310562	0.00310562	-2.519874E-12
13	0.00191938	0.00191938	6.592953E-12
14	0.00118624	0.00118624	-1.726529E-11
15	0.00073314	0.00073314	4.519580E-11
16	0.00045310	0.00045310	-1.183299E-10
17	0.00028003	0.00028003	3.097856E-10
18	0.00017307	0.00017307	-8.110358E-10
19	0.00010696	0.00010696	2.123312E-09
20	0.00006611	0.00006611	-5.558911E-09

Even with a slow floating-point unit, direct multiplication is the preferred method of calculating powers of the silver ratio, as the accuracy achieved using the recurrence relation is unacceptable. However, as shown in Problem Set #1, most modern computers can perform floating-point multiplication just as quickly as addition, and in some cases faster if SIMD instructions are used. Therefore, calculating powers of the silver ratio using direct multiplication is preferable both in terms of accuracy and in terms of performance.

III. SPIN PERIOD OF A RIGID BODY

Rigid bodies can be modeled as a collection of discrete point masses whose positions relative to one another are fixed. This allows the bodies to be described using 3-dimensional vectors and matricies such as the angular momentum vector $\mathbf{L}$, the spin vector ω, and the inertia tensor $\mathbf{\{I\}}$.

The inertia tensor $\mathbf{\{I\}}$ can be represented as a 3×3 symmetric positive-definite matrix. Concisely, its elements are given by

$$\mathbf{I} = \sum_i m_i (r_i^2 \mathbf{1} - \mathbf{r}_i \mathbf{r}_i) \tag{6}$$

Since there are only 6 unique elements, it is more convenient to calculate them separately than to perform the above matrix and vector operations for each data point. Written explicitly, the elements of the inertia tensor are (from [3]):

$$\mathbf{\{I\}} = \begin{Bmatrix} \sum_i m_i (x_{i,2}^2 + x_{i,3}^2) & -\sum_i m_i x_{i,1} x_{i,2} & -\sum_i m_i x_{i,1} x_{i,3} \\ -\sum_i m_i x_{i,1} x_{i,2} & \sum_i m_i (x_{i,1}^2 + x_{i,3}^2) & -\sum_i m_i x_{i,2} x_{i,3} \\ -\sum_i m_i x_{i,1} x_{i,3} & -\sum_i m_i x_{i,2} x_{i,3} & \sum_i m_i (x_{i,1}^2 + x_{i,2}^2) \end{Bmatrix} \tag{7}$$

The relationship between angular momentum, the inertia tensor, and the spin vector is

$$\mathbf{L} = \sum_i m_i (\mathbf{r}_i \times \mathbf{v}_i) = \mathbf{I}\omega \tag{8}$$

This is a linear system in 3 unknowns and can easily be solved using Gauss-Jordan elimination or LU decomposition. However, because the inertia tensor is symmetric and positive-definite, Cholesky decomposition can be used to solve the system using half the operations. Furthermore, the algorithm requires no pivoting and is thus straightforward to implement if no platform-optimized linear algebra libraries are available.

The program `spinperiod` was originally written in C and linked against the AMD Core Math Library (ACML). Due to the legacy of FORTRAN, this required that the 3×3 inertia tensor be stored as 9-element vector in which the matrix elements were stored in column-major order. The LAPACK routine `DPOSV` was then used to solve equation 8 for ω. Once found, the period of the rotating object could be calculated as $T = 2\pi/|\omega|$, where the norm of the spin vector was found using the BLAS routine `DNRM2`. The program was then expanded to include its own implementations of Cholesky decomposition and vector norms in the case that ACML was not available. The routine for the Cholesky decomposition is a translation of the FORTRAN routines `choldc` and `cholsl` found in [1].

The program was then re-written to use the GNU Scientific Library (GSL), and it was also translated into Java, where the Colt linear algebra library from CERN was used.

All versions of the program take one argument: the filename of the data file to read in. This file is expected to contain 6 columns of decimal numbers corresponding to x, y, z, vx, vy, vz. The output contains the momentum vector, the inertia tensor, the spin vector, and the period of the object. For the given rigid body defined in `ps2.dat`, these values are:

$$\begin{aligned}
\mathbf{L} &= \begin{pmatrix} -3089.879672 \\ -240.005076 \\ -923896.671436 \end{pmatrix} \\
\mathbf{I} &= \begin{pmatrix} 1058140191.14 & 738005013.47 & -12628549.76 \\ 738005013.47 & 1739743312.28 & -1295027.85 \\ -12628549.76 & -1295027.85 & 2280122047.62 \end{pmatrix} \\
\omega &= \begin{pmatrix} -1.058039 \times 10^{-05} \\ 4.048620 \times 10^{-06} \\ -4.052524 \times 10^{-04} \end{pmatrix} \\
T &= 4.305089 \text{ hr}
\end{aligned}$$

Thus, the spin period of this object is approximately 4.3 hours.

The position coordinates of the discrete particles making up the body were then loaded into the 3D modeling tool Blender and rendered as halos to create a rotating view of the body. This animation is attached as `rigidbody.mp4`.

---

[1] Press, William H., Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing (Volume 1 of Fortran Numerical Recipes)*. Cambridge University Press, 2001.

[2] Weisstein, Eric W. "Golden Ratio Conjugate." From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/GoldenRatioConjugate.html`.

[3] Thornton, Stephen T., Jerry B. Marion. *Classical Dynamics of Particles and Systems*. Fifth Edition. Thomson, 2004.