Computing Cross-Products and Rotations in 2- and 3-Dimensional Euclidean Spaces

Prof. W. Kahan
Mathematics and Computer Science Depts.
University of California
Berkeley CA 94720

§0: Abstract and Table of Contents

Applications of cross-products to geometrical problems in Euclidean 3-Space lead to formulas that are easier to manipulate with associative matrix multiplications than with conventional non-associative and anti-commutative cross-products. This thesis is supported by derivations of neat formulas for rotations and for solutions to nearest-point problems. However, regardless of their notation, many neat formulas in textbooks can be traduced numerically by roundoff unless they are evaluated extra-precisely. Otherwise, unobvious extra steps must be taken to compute results at least about as accurately as the data deserve. How accurately is that?

Table of Contents

Starting in 1981, these notes were assembled from lecture notes for classes in Sophomore Multi-Variable Calculus and Linear Algebra, and in Junior Numerical Analysis. Students and diligent readers are expected to fill in some omitted but routine algebraic manipulations and proofs.

§1: √(–I) in Euclidean 2-Space (a Summary and Review)

The operator $-I$ reverses vectors. In two dimensions it has a skew-symmetric square root $J := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ determined uniquely but for its sign by the two equations $J^2 = -I$ and $J^T = -J$. This operator J rotates the plane through a quarter turn; whether clockwise or counter-clockwise depends upon the respectively left- or right-handed orientation of the coordinate system. More generally, $\exp(\theta\cdot J) := I\cdot\cos(\theta) + J\cdot\sin(\theta)$ turns the plane through an angle θ. To construct a vector of length $\|\mathbf{u}\| = \sqrt{(\mathbf{u}^T\mathbf{u})}$ perpendicular to any given vector $\mathbf{u}$ in the Euclidean plane, form $J\mathbf{u}$. For any 2-by-2 matrix $B := [\mathbf{u}\ \ \mathbf{v}]$ we find that $B^TJB = J\cdot\mathbf{v}^TJ\mathbf{u} = J\cdot\det(B)$, which implies $\mathrm{Adj}(B) = -JB^TJ$. (Recall $\mathrm{Adj}(B) := \det(B)\cdot B^{-1}$ when $\det(B) \neq 0$.) Our formulas use associative matrix multiplication for the scalar product $\mathbf{v}^T\cdot\mathbf{w} = \mathbf{w}^T\cdot\mathbf{v}$ instead of the non-associative dot product $\mathbf{v}\bullet\mathbf{w}$ for reasons that will become increasingly persuasive in the following pages.

Because J is unchanged by rotations of coordinates, it can produce ostensibly coordinate-free solutions for many geometric problems in the Euclidean plane. For instance, the equation of a line through $\mathbf{v}$ perpendicular to $\mathbf{w}$ is $\mathbf{w}^T(\mathbf{x}-\mathbf{v}) = 0$; the equation of a line through $\mathbf{v}$ parallel to $\mathbf{u}$ is $\mathbf{u}^TJ(\mathbf{x}-\mathbf{v}) = 0$; two lines whose equations are $\mathbf{u}^T\mathbf{x} = \mu$ and $\mathbf{v}^T\mathbf{x} = ß$ intersect at a point $\mathbf{z} := J\cdot(\mathbf{u}\cdot ß - \mathbf{v}\cdot\mu)/(\mathbf{v}^TJ\mathbf{u})$. However, not every orthogonal change of basis (coordinates) leaves J unchanged; a *Reflection* $W = W^{-1} = W^T \neq \pm I$ changes J to $W^{-1}JW = W^TJW = J\cdot\det(W) = -J$, which reminds us that reflection reverses orientation in the plane.

Do you see why such a W must be a reflection? Why it must have the form $W = I - 2\mathbf{w}\mathbf{w}^T/\mathbf{w}^T\mathbf{w}$ for a suitable vector $\mathbf{w}$? Why $\det(W) = -1$? Can you confirm every unobvious assertion in the summary above?

In many ways, but not all, J is to the Euclidean plane what $\imath := \sqrt{(-1)}$ is to the complex plane. J operates upon vectors in the plane but is not a vector in that plane, whereas $\imath$ is simultaneously a multiplicative operator and a vector in the complex plane. The two planes are topologically different, though often confused: Roughly speaking, the complex plane has just one point at infinity best visualized by *Stereographically* projecting the complex plane upon a sphere, whereas the Euclidean plane has a circle (or at least a line) at infinity. We won't pursue this here.

Cross-products of vectors in Euclidean 2-Space appear in restrictions to 2-space of formulas derived originally for vectors in Euclidean 3-Space. Consequently the 2-space interpretation of " $\mathbf{u}\times\mathbf{v}$ " often reduces to a scalar $\mathbf{u}\times\mathbf{v} = \mathbf{v}^TJ\mathbf{u}$. Because cross-products are neither associative nor commutative, triple products like " $\mathbf{u}\bullet\mathbf{v}\times\mathbf{w}$ ", " $\mathbf{u}\times\mathbf{v}\bullet\mathbf{w}$ " and " $\mathbf{u}\times\mathbf{v}\times\mathbf{w}$ " can generate confusion if parsed improperly. When all vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ lie in the same Euclidean 2-Space, …

$\mathbf{u}\bullet(\mathbf{v}\times\mathbf{w})$ and $(\mathbf{u}\times\mathbf{v})\bullet\mathbf{w}$ should both be zero if they appear at all, and

$(\mathbf{u}\times\mathbf{v})\times\mathbf{w} = -\mathbf{w}\times(\mathbf{u}\times\mathbf{v}) = \mathbf{w}\times(\mathbf{v}\times\mathbf{u}) = J\mathbf{w}\cdot(\mathbf{v}^TJ\mathbf{u})$ in 2-space.

These formulas will come from §§4-5, will be used in §9's problem #8, and will make sense after we have found and adopted the matrix notation for cross-products that motivates these notes.

§2: Cross-Products and Rotations in Euclidean 3-Space

Henceforth bold-faced lower-case letters $\mathbf{p}, \mathbf{q}, \mathbf{r}, \ldots, \mathbf{x}, \mathbf{y}, \mathbf{z}$ stand for real 3-dimensional column-vectors. Then row vector $\mathbf{p}^T := [p_1\ \ p_2\ \ p_3]$ is the transpose of column vector $\mathbf{p}$, and $\mathbf{p}^T\mathbf{q} = \mathbf{p}^T\cdot\mathbf{q}$ is the scalar product $\mathbf{p}\bullet\mathbf{q}$ of row $\mathbf{p}^T$ and column $\mathbf{q}$. Euclidean length $\|\mathbf{p}\| := \sqrt{(\mathbf{p}^T\cdot\mathbf{p})}$.

Do not confuse the scalar $\mathbf{p}^T\cdot\mathbf{q} = \mathbf{q}^T\cdot\mathbf{p}$ with the 3–by–3 matrices (“dyads”) $\mathbf{p}\cdot\mathbf{q}^T \neq \mathbf{q}\cdot\mathbf{p}^T$ of rank 1, nor with the vector cross-product $\mathbf{p}\times\mathbf{q} = -\mathbf{q}\times\mathbf{p}$. Cross-products are important enough to justify introducing a notation “ $\mathbf{p}^{¢}$ ”, pronounced “ pee-cross ”, for a 3-by-3 skew-symmetric ($(\mathbf{p}^{¢})^T = -\mathbf{p}^{¢}$) matrix of rank 2 defined by the vector cross-product thus: $\mathbf{p}\times\mathbf{q} = \mathbf{p}^{¢}\cdot\mathbf{q}$. We find the matrix to be this: $\mathbf{p}^{¢} := \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}$. We’ll see whence it comes after we see why we like it.

We like matrix notation for these geometrical entities because matrix multiplication is *associative*:

$$\mathbf{p}^T\cdot\mathbf{q}^{¢}\cdot\mathbf{r} = (\mathbf{p}^T\cdot\mathbf{q}^{¢})\cdot\mathbf{r} = \mathbf{p}^T\cdot(\mathbf{q}^{¢}\cdot\mathbf{r}) = \mathbf{p}\bullet(\mathbf{q}\times\mathbf{r}) \quad \text{and} \quad \mathbf{p}^{¢}\cdot\mathbf{q}^{¢}\cdot\mathbf{r} = (\mathbf{p}^{¢}\cdot\mathbf{q}^{¢})\cdot\mathbf{r} = \mathbf{p}^{¢}\cdot(\mathbf{q}^{¢}\cdot\mathbf{r}) = \mathbf{p}\times(\mathbf{q}\times\mathbf{r})$$

unlike scalar and cross-products; $(\mathbf{p}\bullet\mathbf{q})\cdot\mathbf{r} \neq \mathbf{p}\cdot(\mathbf{q}\bullet\mathbf{r})$ and $(\mathbf{p}\times\mathbf{q})\times\mathbf{r} \neq \mathbf{p}\times(\mathbf{q}\times\mathbf{r})$. Besides legibility, a matrix notation promotes simpler expressions, shorter proofs, and easier operator overloading in programming languages.

Matrix-like cross-products are not new. See precursors cited and an alternative notation offered in work by Götz Trenkler *et al.* published in *Int. J. Math. Educ. Sci. & Technol.* **29** #3 (1998) pp. 455-9, **30** #4 (1999) pp. 549-555, **33** #3 (2002) pp. 431-6, and in *Discussiones Mathematicae — General Algebra & Appl’ns* **21** (2001) pp. 67-82. He explores in directions other than numerical stability. I still prefer my notation $\mathbf{p}^{¢}$ to his $\mathbf{T}_p$ after trying both.

§3: For Readers Reluctant to Abandon • and x Products

(Other readers can skip to §4.)

We’re not abandoning familiar locutions; we’re just writing most of them shorter. Compare the following classical formulas with their matrix equivalents for succinctness and ease of proof:

Triple Cross-Product: $(\mathbf{p}\times\mathbf{q})\times\mathbf{r} = \mathbf{q}\cdot\mathbf{p}\bullet r - \mathbf{p}\cdot\mathbf{q}\bullet r$ vs. $(\mathbf{p}^{¢}\cdot\mathbf{q})^{¢} = \mathbf{q}\cdot\mathbf{p}^T - \mathbf{p}\cdot\mathbf{q}^T$

Jacobi’s Identity: $\mathbf{p}\times(\mathbf{q}\times\mathbf{r}) + \mathbf{q}\times(\mathbf{r}\times\mathbf{p}) = -\mathbf{r}\times(\mathbf{p}\times\mathbf{q})$ vs. $\mathbf{p}^{¢}\cdot\mathbf{q}^{¢} - \mathbf{q}^{¢}\cdot\mathbf{p}^{¢} = (\mathbf{p}^{¢}\cdot\mathbf{q})^{¢}$

Lagrange’s Identity: $(\mathbf{t}\times\mathbf{u})\bullet(\mathbf{v}\times\mathbf{w}) = \mathbf{t}\bullet\mathbf{v}\cdot\mathbf{u}\bullet\mathbf{w} - \mathbf{u}\bullet\mathbf{v}\cdot\mathbf{t}\bullet\mathbf{w}$ vs. $(\mathbf{t}^{¢}\cdot\mathbf{u})^T\cdot(\mathbf{v}^{¢}\cdot\mathbf{w}) = \det([\mathbf{t}\ \ \mathbf{u}]^T\cdot[\mathbf{v}\ \ \mathbf{w}])$

Most things don’t change much; $\mathbf{p}\times\mathbf{q} = -\mathbf{q}\times\mathbf{p}$ becomes $\mathbf{p}^{¢}\cdot\mathbf{q} = -\mathbf{q}^{¢}\cdot\mathbf{p}$, so $\mathbf{p}^{¢}\cdot\mathbf{p} = \mathbf{o}$ (the zero vector), and $\mathbf{p}\bullet(\mathbf{q}\times\mathbf{r}) = \mathbf{p}^T\cdot\mathbf{q}^{¢}\cdot\mathbf{r} = \det([\mathbf{p}\ \ \mathbf{q}\ \ \mathbf{r}])$.

The notations’ difference becomes more pronounced as problems become more complicated. For instance, given a unit vector $\hat{\mathbf{u}}$ (with $||\hat{\mathbf{u}}|| = 1$) and a scalar ψ, what orthogonal matrix $R = (R^T)^{-1}$ rotates Euclidean 3–space through an angle ψ radians around the axis $\hat{\mathbf{u}}$? In other words, $R\cdot\mathbf{x}$ is to transform every vector $\mathbf{x}$ by rotating it through the angle ψ about the axis $\hat{\mathbf{u}}$ fixed through the origin $\mathbf{o}$.

An ostensibly simple formula $R := \exp(\psi\cdot\hat{\mathbf{u}}^{¢})$ uses the skew-symmetric cross-product matrix $\hat{\mathbf{u}}^{¢}$ defined above. Here exp(…) is *not* the *array* exponential that is applied elementwise, but is the *matrix* exponential; think of $R = R(\psi)$ as a matrix-valued function of ψ that solves the differential equation $dR/d\psi = \hat{\mathbf{u}}^{¢}\cdot R = R\cdot\hat{\mathbf{u}}^{¢}$ starting from the identity matrix $R(0) = I$. Given $\hat{\mathbf{u}}$ and ψ, an explicit formula for this R is $R = I + 2\cdot(\,I\cdot\cos(\psi/2) + \hat{\mathbf{u}}^{¢}\cdot\sin(\psi/2)\,)\cdot\hat{\mathbf{u}}^{¢}\cdot\sin(\psi/2)$. Rewriting this expression with solely • and × products doesn’t improve it. Try to do so!

In what follows the formulas above will be first derived and then applied to several examples.

§4: A Derivation of Cross-Products in Euclidean 3-Space

What operators in Euclidean 3-space are analogous to the quarter-turn J in 2-space? Every rotation of 3-space is characterized by its *axis*, a line left unchanged by the rotation, and by its angle of rotation about that axis. Let $\mathbf{v}$ be a nonzero vector parallel to such an axis. Analogous to $-I$ in 2-space is the operator $\mathbf{v}\cdot\mathbf{v}^T/\mathbf{v}^T\mathbf{v} - I$, which projects arbitrary vectors into the plane through $\mathbf{o}$ perpendicular to $\mathbf{v}$ and then reverses the projection through $\mathbf{o}$. That operator's skew-symmetric square root, determined (as we'll see) uniquely but for its sign by $\mathbf{v}$, is analogous to J, but different for every different axis $\mathbf{v}$. Consequently that square root is a discontinuous function of $\mathbf{v}$ at $\mathbf{v} = \mathbf{o}$. Multiplying that square root by $\|\mathbf{v}\|$ renders it continuous everywhere.

Hence we define the operator $\mathbf{v}^{¢}$ to be one of the two solutions $\mathbf{v}^{¢} := \pm S$ of the equations

$$S^2 = \mathbf{v}\cdot\mathbf{v}^T - \mathbf{v}^T\mathbf{v}\cdot I \quad \text{and} \quad S = -S^T .$$

To see why these equations determine S uniquely but for sign, choose an orthonormal basis with $\mathbf{v}/\|\mathbf{v}\|$ as its first basis vector and find a matrix representing S in that coordinate system. Every such matrix S must satisfy $S\mathbf{v} = \mathbf{o}$; here is why: Evidently $S^2\mathbf{v} = \mathbf{o}$, so $\det(S)^2 = \det(S^2) = 0$, and so $S\mathbf{z} = \mathbf{o}$ for some $\mathbf{z} \ne \mathbf{o}$; but then $S^2\mathbf{z} = \mathbf{o}$, and this implies that $\mathbf{z}$ is a scalar multiple of $\mathbf{v}$, whence follows $S\mathbf{v} = \mathbf{o}$ as claimed. Consequently, in the foregoing orthonormal coordinate system, every skew-symmetric solution S is represented by a matrix whose first row and column contain only zeros, whereupon the remaining 2-by-2 principal submatrix must be $\pm J\cdot\|\mathbf{v}\|$ as is explained in the second sentence of §1. Thus, S is determined uniquely but for sign.

Given $\mathbf{v} = \begin{bmatrix}\xi\\ \eta\\ \zeta\end{bmatrix}$, consider the solution $S := \begin{bmatrix}0 & -\zeta & \eta\\ \zeta & 0 & -\xi\\ -\eta & \xi & 0\end{bmatrix}$ of the equations $S\mathbf{v} = \mathbf{o}$ and $S = -S^T$; this $S^2 = \mathbf{v}\mathbf{v}^T - \mathbf{v}^T\mathbf{v}\cdot I$ too, which combines with the previous paragraph to imply $\mathbf{v}^{¢} = \pm S$. Its sign could be chosen arbitrarily but we set $\mathbf{v}^{¢} := +S$, thereby classifying the coordinate system as "right-handed". Note now that $\mathbf{v}^{¢}$ is a continuous function of $\mathbf{v}$. In summary, …

> For every vector $\mathbf{v}$ in Euclidean 3-space, the linear operator $\mathbf{v}^{¢}$ is a continuous linear function of $\mathbf{v}$ determined but for sign by the equations $(\mathbf{v}^{¢})^2 = \mathbf{v}\cdot\mathbf{v}^T - \mathbf{v}^T\mathbf{v}\cdot I$ and $(\mathbf{v}^{¢})^T = -\mathbf{v}^{¢}$. Its sign is determined for every $\mathbf{v}$ by its sign for any one $\mathbf{v} \ne \mathbf{o}$ and by continuity.

The notation for $\mathbf{v}^{¢}$, pronounced "vee-cross", is inspired by the relation $\mathbf{v}^{¢}\cdot\mathbf{w} = \mathbf{v}\times\mathbf{w}$; the latter cross-product coincides with that defined in texts on vector analysis. Six of its properties are …

$\mathbf{v}^{¢}\cdot\mathbf{w} = \mathbf{o}$ just when $\pm\mathbf{v}$ and $\mathbf{w}$ are parallel; this was proved using $\mathbf{z}$ above.

$\mathbf{v}^{¢}\cdot\mathbf{w} \perp \mathbf{v}$ because $\mathbf{v}^T\cdot\mathbf{v}^{¢}\cdot\mathbf{w} = -(\mathbf{v}^{¢}\cdot\mathbf{v})^T\cdot\mathbf{w} = \mathbf{o}^T\cdot\mathbf{w} = 0$.

$\mathbf{v}^{¢}\cdot\mathbf{w} \perp \mathbf{w}$ because $\mathbf{w}^T\cdot\mathbf{v}^{¢}\cdot\mathbf{w} = (\mathbf{w}^T\cdot\mathbf{v}^{¢}\cdot\mathbf{w})^T = \mathbf{w}^T\cdot(\mathbf{v}^{¢})^T\cdot\mathbf{w} = -\mathbf{w}^T\cdot\mathbf{v}^{¢}\cdot\mathbf{w}, \ = 0$.

$\|\mathbf{v}^{¢}\cdot\mathbf{w}\|^2 = \|\mathbf{v}\|^2\,\|\mathbf{w}\|^2 - (\mathbf{v}^T\mathbf{w})^2$ because it is $-\mathbf{w}^T(\mathbf{v}^{¢})^2\mathbf{w}$, *etc*.

Combining the formula $\mathbf{v}^T\mathbf{w} = \|\mathbf{v}\|\cdot\|\mathbf{w}\|\cdot\cos\angle(\mathbf{v}, \mathbf{w})$ with the last equation proves that

$$\|\mathbf{v}^{¢}\cdot\mathbf{w}\| = \pm\|\mathbf{v}\|\cdot\|\mathbf{w}\|\cdot\sin\angle(\mathbf{v}, \mathbf{w})$$

with a sign that depends upon the orientation, if any, assigned to the angle $\angle(\mathbf{v}, \mathbf{w})$ when it has to be distinguished from $\angle(\mathbf{w}, \mathbf{v}) = -\angle(\mathbf{v}, \mathbf{w})$. Anyway,

$$\|\mathbf{v}^{¢}\cdot\mathbf{w}\| = |\text{the area of a parallelogram with adjacent sides } \mathbf{v} \text{ and } \mathbf{w}| .$$

From the foregoing properties we infer by symmetry that $\mathbf{w}^{¢}\cdot\mathbf{v}$ must be one of $\pm\mathbf{v}^{¢}\cdot\mathbf{w}$ whenever they are nonzero because they are vectors with the same length and perpendicular to the same two nonparallel vectors $\mathbf{v}$ and $\mathbf{w}$. Trials with basis vectors for $\mathbf{v}$ and $\mathbf{w}$ imply a seventh property:

$$\mathbf{w}^{¢}\cdot\mathbf{v} = -\,\mathbf{v}^{¢}\cdot\mathbf{w}\ ,$$

and this equation must persist for all $\mathbf{v}$ and $\mathbf{w}$ since both sides are continuous bilinear functions. This *anti-commutative* identity is a good reason to prefer the notation $\mathbf{v}^{¢}\cdot\mathbf{w}$ over $\mathbf{v}\times\mathbf{w}$; and later the preference will intensify when we find the triple cross-product *non-associative*. Besides, we shall need $\mathbf{v}^{¢}$ in isolation later to describe rotations succinctly.

§5: Triple Products

The scalar expression $\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w}$ is linear in each vector separately, and reverses sign when any two vectors are swapped; this follows from anti-commutativity when $\mathbf{v}$ and $\mathbf{w}$ are swapped, from skew-symmetry of $\mathbf{v}^{¢}$ when $\mathbf{u}$ and $\mathbf{w}$ are swapped; and when $\mathbf{u}$ and $\mathbf{v}$ are swapped it follows from $\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w} = -\mathbf{w}^T\mathbf{v}^{¢}\mathbf{u} = \mathbf{w}^T\mathbf{u}^{¢}\mathbf{v} = -\mathbf{v}^T\mathbf{u}^{¢}\mathbf{w}$. Compare this with the characterization of the determinant $\det([\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}])$ as a functional, linear in each column of $[\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}]$ separately, that reverses sign when any two columns are swapped. It follows that $\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w}/\det([\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}])$ must be a constant provided the denominator does not vanish. Setting matrix $[\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}] := \mathrm{I}$ determines that constant to be 1 , whereupon continuity implies an important identity for all $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$:

$$\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w} = \det([\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}])\ .$$

This *Determinantal Identity* can be confirmed by direct but tedious algebraic manipulation, and also by the following geometric argument:

Let parallelogram $\boldsymbol{P}$ have adjacent sides $\mathbf{v}$ and $\mathbf{w}$ so that its area $|\boldsymbol{P}| = \|\mathbf{v}^{¢}\mathbf{w}\| \neq 0$. Next let $\boldsymbol{Q}$ be a parallelepiped whose sides emanating from a vertex are $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$; then its volume is

$$\begin{aligned} |\boldsymbol{Q}| &= |\det([\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}])| \quad \text{and also}\\ |\boldsymbol{Q}| &= |\boldsymbol{P}|\cdot\|\text{ projection of }\mathbf{u}\text{ onto the unit-normal to }\boldsymbol{P}\,\|\\ &= \|\mathbf{v}^{¢}\mathbf{w}\|\cdot\|\text{ projection of }\mathbf{u}\text{ onto }\mathbf{v}^{¢}\mathbf{w}/\|\mathbf{v}^{¢}\mathbf{w}\|\,\| = |\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w}|\ . \end{aligned}$$

Now to confirm that $\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w} = +\det([\mathbf{u}\ \ \mathbf{v}\ \ \mathbf{w}])$ try any three vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, say the basis vectors, and then invoke continuity to cope with the case when $\mathbf{v}$ and $\mathbf{w}$ are (anti)parallel.

Almost as important as that determinantal identity is *Grassmann's triple cross-product formula*

$$\mathbf{u}\times(\mathbf{v}\times\mathbf{w}) = \mathbf{u}^{¢}\mathbf{v}^{¢}\mathbf{w} = \mathbf{v}\cdot\mathbf{u}^T\mathbf{w} - \mathbf{w}\cdot\mathbf{u}^T\mathbf{v} = (\mathbf{v}\mathbf{w}^T - \mathbf{w}\mathbf{v}^T)\cdot\mathbf{u}\ .$$

To prove this, note that it must be perpendicular to a vector $\mathbf{v}^{¢}\mathbf{w}$ perpendicular to both $\mathbf{v}$ and $\mathbf{w}$, and hence must lie in the plane of $\mathbf{v}$ and $\mathbf{w}$. Therefore $\mathbf{u}^{¢}\mathbf{v}^{¢}\mathbf{w} = \mathbf{v}\cdot ß - \mathbf{w}\cdot\mu$ for some scalars $ß$ and μ . Premultiplication by $\mathbf{u}^T$ reveals that $0 = \mathbf{u}^T\mathbf{v}\cdot ß - \mathbf{u}^T\mathbf{w}\cdot\mu$ and therefore some scalar functional $f = \mathbf{u}^T\mathbf{v}/\mu = \mathbf{u}^T\mathbf{w}/ß$ exists satisfying $\mathbf{u}^{¢}\mathbf{v}^{¢}\mathbf{w} = (\,\mathbf{v}\cdot\mathbf{u}^T\mathbf{w} - \mathbf{w}\cdot\mathbf{u}^T\mathbf{v}\,)/f$. Since both sides of this equation are linear in each of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ separately, f can vary with none of them; it must be a constant. Its value $f = 1$ can be found by substituting one basis vector for $\mathbf{u}$ and $\mathbf{w}$ and a second basis vector for $\mathbf{v}$. Alternatively, brute-force manipulation by a computerized algebra system like *Derive*, *Maple*, *Mathematica* or *Macsyma* can be used to confirm the triple cross-product formula. It is easier to remember in §3's form: $(\mathbf{v}^{¢}\cdot\mathbf{w})^{¢} = \mathbf{w}\cdot\mathbf{v}^T - \mathbf{v}\cdot\mathbf{w}^T$.

That formula shows that $(\mathbf{u}\times\mathbf{v})\times\mathbf{w} = (\mathbf{u}^{¢}\mathbf{v})^{¢}\mathbf{w} = -\mathbf{w}^{¢}\mathbf{u}^{¢}\mathbf{v} = (\mathbf{v}\mathbf{u}^T - \mathbf{u}\mathbf{v}^T)\mathbf{w} \neq \mathbf{u}\times(\mathbf{v}\times\mathbf{w})$; the cross-product is *not* associative, though matrix multiplication *is* associative: $(\mathbf{u}^{¢}\cdot\mathbf{v}^{¢})\cdot\mathbf{w} = \mathbf{u}^{¢}\cdot(\mathbf{v}^{¢}\cdot\mathbf{w})$.
That formula also confirms *Jacobi's Identity*:

$$\mathbf{u}^{¢}\mathbf{v}^{¢}\mathbf{w} + \mathbf{v}^{¢}\mathbf{w}^{¢}\mathbf{u} + \mathbf{w}^{¢}\mathbf{u}^{¢}\mathbf{v} = \mathbf{o}\,, \quad \text{or} \quad (\mathbf{u}^{¢}\cdot\mathbf{v})^{¢} = \mathbf{u}^{¢}\cdot\mathbf{v}^{¢} - \mathbf{v}^{¢}\cdot\mathbf{u}^{¢}\,,$$

and helps to confirm *Lagrange's Identity*:

$$(\mathbf{t}^{¢}\cdot\mathbf{u})^T(\mathbf{v}^{¢}\cdot\mathbf{w}) = \mathbf{t}^T\mathbf{v}\cdot\mathbf{u}^T\mathbf{w} - \mathbf{u}^T\mathbf{v}\cdot\mathbf{t}^T\mathbf{w} = \det([\mathbf{t}\ \ \mathbf{u}]^T[\mathbf{v}\ \ \mathbf{w}])\,.$$

Since they are not long, you should work out the confirmations of these identities, which figure in both classical and Quantum mechanics.

§6: Rotations R about an Axis v in Euclidean 3-Space

If skew-symmetric matrix $S = -S^T$ is constant, the unique solution of the initial-value problem

$$R(0) = I \quad \text{and} \quad dR/d\tau = S\cdot R \ \text{ for all } \tau$$

is a matrix $R(\tau)$ that must be orthogonal; $R^T = R^{-1}$ because $d(R^TR)/d\tau = R^TS^TR+R^TSR = O$ and therefore $R^TR = I$ for all τ . This implies $\det(R)^2 = 1$ and then $\det(R) = +1$ because it is continuous for all τ . Thus, $R(\tau)$ is a proper rotation — no reflection. It has a power series too:

$$R(\tau) = \exp(\tau\cdot S) = \textstyle\sum_{k\ge0} \tau^k\cdot S^k/k!\,.$$

Now, every 3-by-3 skew-symmetric matrix S determines a vector $\mathbf{v}$ such that $S = \mathbf{v}^{¢}$; then $S\mathbf{v} = \mathbf{o}$, $S^2 = \mathbf{v}\mathbf{v}^T - \|\mathbf{v}\|^2\cdot I$, $S^3 = -\|\mathbf{v}\|^2\cdot S$, $S^4 = -\|\mathbf{v}\|^2\cdot S^2$, …, $S^{m+2k} = (-\|\mathbf{v}\|^2)^k\cdot S^m$ for $m > 0$.
By taking odd and even terms separately in the series for $\exp(\tau\cdot\mathbf{v}^{¢})$ we condense it to

$$\begin{aligned} R(\tau) = \exp(\tau\cdot\mathbf{v}^{¢}) &= I + \big(1 - \cos(\tau\cdot\|\mathbf{v}\|)\big)\cdot(\mathbf{v}^{¢}/\|\mathbf{v}\|)^2 + \sin(\tau\cdot\|\mathbf{v}\|)\cdot\mathbf{v}^{¢}/\|\mathbf{v}\| \\ &= I + 2\cdot\sin(\tau\cdot\|\mathbf{v}\|/2)\cdot\big(\sin(\tau\cdot\|\mathbf{v}\|/2)\cdot\mathbf{v}^{¢}/\|\mathbf{v}\| + \cos(\tau\cdot\|\mathbf{v}\|/2)\cdot I\,\big)\cdot\mathbf{v}^{¢}/\|\mathbf{v}\|\,, \end{aligned}$$

thus providing a relatively simple and verifiable formula for the operator that rotates Euclidean 3-space through an angle $\tau\cdot\|\mathbf{v}\|$ about a given axis $\mathbf{v} \neq \mathbf{o}$. Its τ-derivative is extremely simple:

$$R'(\tau) = d\exp(\tau\cdot\mathbf{v}^{¢})/d\tau = \mathbf{v}^{¢}\cdot\exp(\tau\cdot\mathbf{v}^{¢}) = \exp(\tau\cdot\mathbf{v}^{¢})\cdot\mathbf{v}^{¢} = R(\tau)\cdot\mathbf{v}^{¢} = \mathbf{v}^{¢}\cdot R(\tau)\,.$$

(The **v**-derivative would require a long expression too complicated to serve the didactic purposes of these notes.)

The converse task is this: Given an orthogonal matrix $R = R^{T-1} \neq I$ that effects a proper rotation because $\det(R) = +1$, how can its axis $\mathbf{v}$ be determined? What seems the simplest way at first extracts $\mathbf{v}$ from $R - R^T = 2\cdot\sin(\tau\cdot\|\mathbf{v}\|)\cdot\mathbf{v}^{¢}/\|\mathbf{v}\|$; this works well unless $\tau\cdot\|\mathbf{v}\|$ is so near $\pm\pi$ that roundoff in R obscures the desired result. A generally more reliable procedure applies Gaussian elimination to solve the equation $(R - I)\cdot\mathbf{v} \approx \mathbf{o}$ for a $\mathbf{v} \neq \mathbf{o}$; this procedure can work well because $R - I$ must be singular (to within roundoff) with rank 2 , whence $\mathrm{Adj}(R - I)$ must be some nonzero scalar multiple of $\mathbf{v}\mathbf{v}^T$. Here is how we know all this to be so:

Consider any eigenvalue μ of R ; this μ may be complex, in which case its eigenvector $\mathbf{z} \neq \mathbf{o}$ is complex too, and we shall write $\mathbf{z}^*$ for its complex-conjugate transpose. Next we find that $|\mu|^2\cdot\mathbf{z}^*\mathbf{z} = (R\mathbf{z})^*(R\mathbf{z}) = \mathbf{z}^*R^TR\mathbf{z} = \mathbf{z}^*\mathbf{z} > 0$, whereupon $|\mu| = 1$. Now, R has three eigenvalues μ , the roots of the characteristic equation $\det(\mu I - R) = 0$. Because its coefficients are real, any of the three eigenvalues that are not real must come in complex-conjugate pairs whose product, their squared magnitude, must equal 1 . The product of all three eigenvalues is $\det(R) = +1$ too.

Two cases can arise:

- If R has a non-real eigenvalue μ then μ^* is another and the third is $1/(\mu^*\cdot\mu) = 1$.
- Otherwise all three eigenvalues are real, namely ±1 , and then $+1$ appears among them an odd number of times (but not thrice) because their product is $+1$ too.

Thus $R–I$ must be singular; axis $\mathbf{v}$ is an eigenvector of R belonging to its eigenvalue $+1$.

Problem (hard): Show that $\mathrm{Adj}(R – I) = (3 – \mathrm{Trace}(R))\cdot\mathbf{v}\mathbf{v}^T/\|\mathbf{v}\|^2$ provided proper orthogonal $R \ne I$.

Must each proper orthogonal $R = \exp(\tau\cdot\hat{\mathbf{u}}^{¢})$ for some real τ and unit vector $\hat{\mathbf{u}}$ (with $\|\hat{\mathbf{u}}\| = 1$) ? Yes, and $\hat{\mathbf{u}} = \mathbf{v}/\|\mathbf{v}\|$ where $\mathbf{v}$ is the axis found above. To see why, change to a new orthonormal coordinate system with $\hat{\mathbf{u}}$ as its first basis vector. The matrix representing R in this new basis has $[1\ \ 0\ \ 0]$ and its transpose as first row and column. (Why?) The matrix's last principal 2-by-2 submatrix must be $\exp(\tau\cdot J)$ because it is proper orthogonal too; thus τ is determined. After changing back to the original basis we find $R = \exp(\pm\tau\cdot\hat{\mathbf{u}}^{¢})$. (We'll explain the $\pm$ sign later.)

§7: Constructing Rotations out of Reflections in a Euclidean Space of any Dimension ≥ 2

For any $\mathbf{w} \ne \mathbf{o}$ in *any* Euclidean space, $W := I – 2\mathbf{w}\mathbf{w}^T/\|\mathbf{w}\|^2$ is an orthogonal reflection.

Problem: Verify that $W = W^T = W^{–1}$, that $\mathbf{w} = –W\mathbf{w}$ is reversed by the reflection, and that it preserves the (hyper)plane of vectors $\mathbf{x}$ orthogonal to $\mathbf{w}$. Thus the reflection's mirror-plane satisfies the equation $\mathbf{w}^T\mathbf{x} = 0$. Verify too that $\det(W) = –1$ by applying the formula $\det(I – \mathbf{u}\mathbf{v}^T) = 1 – \mathbf{v}^T\mathbf{u}$. Can you prove this last formula?

Suppose distinct nonzero vectors $\mathbf{x}, \mathbf{y}, \mathbf{s}$ and $\mathbf{t}$ are given with $\|\mathbf{x}\| = \|\mathbf{y}\|$ and $\|\mathbf{s}\| = \|\mathbf{t}\|$ and $\mathbf{s}^T\mathbf{x} = \mathbf{t}^T\mathbf{y}$. (This last equation says that $|\angle(\mathbf{s}, \mathbf{x})| = |\angle(\mathbf{t}, \mathbf{y})|$.) We wish to construct a proper orthogonal R that rotates $\mathbf{x}$ to $R\mathbf{x} = \mathbf{y}$ and $\mathbf{s}$ to $R\mathbf{s} = \mathbf{t}$. We shall construct this $R := HW$ as a product of two orthogonal reflections: $W := I – 2\mathbf{w}\mathbf{w}^T/\|\mathbf{w}\|^2$ and $H := I – 2\mathbf{h}\mathbf{h}^T/\|\mathbf{h}\|^2$ in which $\mathbf{w} := \mathbf{x}–\mathbf{y}$ and $\mathbf{h} := W\mathbf{s}–\mathbf{t}$, except that if $W\mathbf{s} = \mathbf{t}$ then $\mathbf{h}$ may be any nonzero vector orthogonal to both $\mathbf{y}$ and $\mathbf{t}$ provided such a vector exists. (R might not exist in 2-space; why not?)

Problem: Verify that W swaps $\mathbf{x}$ and $\mathbf{y}$, and that H swaps $W\mathbf{s}$ and $\mathbf{t}$ while preserving $\mathbf{y}$, so that R moves the pair $(\mathbf{s}, \mathbf{x})$ to the pair $(\mathbf{t}, \mathbf{y})$ while preserving their lengths and angle. Verify too that R is proper orthogonal.

Problem (harder): Prove that every rotation in Euclidean 2- or 3-space is a product of two orthogonal reflections. (The proof must ensure that both reflections exist.) How few reflections always suffice in Euclidean N-space?

§8: Changing to an Orthonormal Basis with Opposite Orientation

The vector $\mathbf{v}\times\mathbf{w} = \mathbf{v}^{¢}\mathbf{w}$ is sometimes called a *pseudo-vector* because of how an arbitrary change of orthonormal basis may affect it. For any orthogonal $Q = (Q^T)^{–1}$ we shall find that

$$(Q\mathbf{v})^{¢}(Q\mathbf{w}) = Q\mathbf{v}^{¢}\mathbf{w}\cdot\det(Q) , \quad \text{or equivalently} \quad (Q\mathbf{v})^{¢} = Q\mathbf{v}^{¢}Q^T\cdot\det(Q) .$$

Of course $\det(Q) = \pm1$; its appearance in the formula above is what deserves an explanation.

If $\det(Q) = +1$ then Q is a proper rotation and our geometrical intuition may well persuade us that rotating $\mathbf{v}$ and $\mathbf{w}$ together as a rigid body must rotate $\mathbf{v}^{¢}\mathbf{w}$ the same way, which is what the formula in question says. Otherwise $\det(Q) = –1$, in which case Q combines rotation and reflection; in this case the formula in question, in the form $(Q\mathbf{v})^{¢}Q = Q\mathbf{v}^{¢}\cdot\det(Q)$, will take some work to be confirmed. A comparatively simple proof is provided by …

David Meredith's Identity: $\mathrm{Adj}(L^T)\cdot\mathbf{v}^{¢} = (L\mathbf{v})^{¢}\cdot L$ for any 3-by-3 matrix L and vector $\mathbf{v}$ in Euclidean 3-space. Into this identity substitute $L := Q$ and use $Q^T = Q^{-1}$ and $\mathrm{Adj}(Q^T) = (Q^T)^{-1}\cdot\det(Q^T) = Q\cdot\det(Q)$ to get the formula $(Q\mathbf{v})^{¢}\cdot Q = Q\mathbf{v}^{¢}\cdot\det(Q)$ in question. What remains to be done is to prove Meredith's identity:

Proof: For *all* 3-vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ regarded as columns of a matrix $[\mathbf{v}, \mathbf{w}, \mathbf{u}]$, we find that

$$\det(L)\cdot\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w} = \det(L)\cdot\det([\mathbf{v}, \mathbf{w}, \mathbf{u}]) = \det(L\cdot[\mathbf{v}, \mathbf{w}, \mathbf{u}]) = \det([L\mathbf{v}, L\mathbf{w}, L\mathbf{u}]) = (L\mathbf{u})^T(L\mathbf{v})^{¢}L\mathbf{w} = \mathbf{u}^TL^T(L\mathbf{v})^{¢}L\mathbf{w}\ .$$

Consequently $\det(L)\cdot\mathbf{v}^{¢} = L^T(L\mathbf{v})^{¢}L$. Into this equation substitute $\det(L)\cdot I = \det(L^T)\cdot I = L^T\cdot\mathrm{Adj}(L^T)$ when L^T is nonsingular to get first $L^T\cdot\mathrm{Adj}(L^T)\cdot\mathbf{v}^{¢} = L^T(L\mathbf{v})^{¢}L$, and then the desired identity. It is a polynomial equation in the elements of L and therefore valid also when L is singular. *Q.E.D.*

When $\det(Q) = -1$ the formulas just proved remind us that reflections reverse sense, changing right-handed triad $[\mathbf{v}, \mathbf{w}, \mathbf{v}^{¢}\mathbf{w}]$ into left-handed triad $[Q\mathbf{v}, Q\mathbf{w}, Q\mathbf{v}^{¢}\mathbf{w}]$, whereas $[Q\mathbf{v}, Q\mathbf{w}, (Q\mathbf{v})^{¢}(Q\mathbf{w})]$ is right-handed. (Look in a mirror to see why.) Consequently the last two triads' last elements must be oppositely directed.

Question: Why, when you look in a mirror, do you see *left* and *right* reversed there but not *up* and *down* ?

Answer: That's not what you see. (What you do see is described on the next page.)

§9: Applications of Cross-Products to Geometrical Problems

Cross-products $\mathbf{p}\times\mathbf{q}$, or $\mathbf{p}^{¢}\cdot\mathbf{q}$ in our matrix notation, figure prominently instead of determinants in neat textbook solutions of many commonplace geometrical problems. Our first example is …

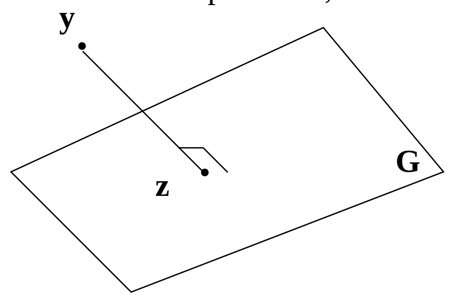
#**0.** Given the equations $\mathbf{p}^T\cdot\mathbf{x} = \pi$, $\mathbf{b}^T\cdot\mathbf{x} = ß$, $\mathbf{q}^T\cdot\mathbf{x} = \theta$ of three planes, they intersect at point

$$\mathbf{z} = (\ \mathbf{b}^{¢}\cdot\mathbf{q}\cdot\pi + \mathbf{q}^{¢}\cdot\mathbf{p}\cdot ß + \mathbf{p}^{¢}\cdot\mathbf{b}\cdot\theta\ )/(\mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q})\ .$$ (This is *Cramer's Rule*.)

Since neat formulas are more memorable they are more likely to appear in textbooks to be copied by programmers than are ugly numerical algorithms like Gaussian Elimination even if the latter are numerically more stable. Gaussian Elimination is also faster than Cramer's Rule but not the rewritten formula $\mathbf{z} = \big((\mathbf{b}^{¢}\cdot\mathbf{q})\cdot\pi + \mathbf{p}^{¢}\cdot(\mathbf{b}\cdot\theta - \mathbf{q}\cdot ß)\big)/\big(\mathbf{p}^T\cdot(\mathbf{b}^{¢}\cdot\mathbf{q})\big)$ if $\mathbf{b}^{¢}\cdot\mathbf{q}$ is reused. Still, this formula is less robust numerically than Gaussian Elimination with pivotal exchanges after row- and column-scaling have curbed any extravagant disparities among magnitudes of coefficients.

Like Beauty, the neatness and speed of a formula lie in the eye of the beholding programmer sooner than does numerical stability. Roundoff doesn't figure in textbooks' formulas. The reader will not easily determine which are numerically unstable among the neat textbook formulas that solve the following eight commonplace geometrical problems each of the *Nearest-Point* kind:

Given a point $\mathbf{y}$ and the definition of a line or plane $\mathbf{G}$, we seek a point $\mathbf{z}$ in $\mathbf{G}$ nearest $\mathbf{y}$.



We expect the line segment joining $\mathbf{y}$ and $\mathbf{z}$ to stick out of $\mathbf{G}$ perpendicularly.

Answer to the ***Question*** on the previous page: What you actually see reversed in a mirror are *forward* and *back* .

If two formulas for $\mathbf{z}$ are offered below they suffer differently from rounding errors; the first formula suffers less than the second whenever $\|\mathbf{z}-\mathbf{y}\| \ll \|\mathbf{y}\|$ and the second less than the first whenever $\|\mathbf{z}\| \ll \|\mathbf{y}\|$. Unless parentheses indicate otherwise, associative products $\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}$ should be evaluated in whichever order, $(\mathbf{A}\cdot\mathbf{B})\cdot\mathbf{C}$ or $\mathbf{A}\cdot(\mathbf{B}\cdot\mathbf{C})$, requires fewer arithmetic operations; doing so below tends to diminish roundoff too. An exercise for the diligent reader is to confirm the mathematical correctness of these formulas, even if roundoff may vitiate their direct application; casual readers will find their confirmations in §12.

"Numerically stable" solutions for these eight "linearly constrained least-squares" problems may be found in some works on numerical linear algebra; but no known stable solution is simply a rational formula like all those below.

#**1.** Given the equation $\mathbf{p}^T\cdot\mathbf{x} = \pi$ of a plane Π, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot(\mathbf{p}^T\cdot\mathbf{y} - \pi)/\|\mathbf{p}\|^2 \;=\; (\,\mathbf{p}\cdot\pi - \mathbf{p}^{\text{¢}}\cdot\mathbf{p}^{\text{¢}}\cdot\mathbf{y}\,)/\|\mathbf{p}\|^2 \,.$$

#**2.** Given three points $\mathbf{u}-\mathbf{v}$, $\mathbf{u}$ and $\mathbf{u}+\mathbf{w}$ through which one plane Π passes, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is $\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2 \;=\; \mathbf{u} - \mathbf{p}^{\text{¢}}\cdot\mathbf{p}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2$ wherein $\mathbf{p} := \mathbf{v}^{\text{¢}}\cdot\mathbf{w}$.

#**3.** Given three points $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ through which one plane Π passes, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2 = \mathbf{u} - \mathbf{p}^{\text{¢}}\cdot\mathbf{p}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2 \quad \text{wherein} \quad \mathbf{p} := (\mathbf{v}-\mathbf{u})^{\text{¢}}\cdot(\mathbf{w}-\mathbf{u}) \,.$$

The order of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ is permutable in each formula separately. To attenuate roundoff in $\mathbf{p}$ choose $\mathbf{u}$ to maximize $\|\mathbf{v}-\mathbf{w}\|$ roughly. (Why? See §11.) For $\mathbf{z}$ choose $\mathbf{u}$ to minimize $\|\mathbf{y}-\mathbf{u}\|$ in the first formula, $\|\mathbf{u}\|$ in the second.

#**4.** Given two points $\mathbf{u}$ and $\mathbf{u}+\mathbf{v}$ through which one line $£$ passes, the point $\mathbf{z}$ in $£$ nearest $\mathbf{y}$ is $\mathbf{z} := \mathbf{y} + \mathbf{v}^{\text{¢}}\cdot\mathbf{v}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{v}\|^2 \;=\; (\,\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} - \mathbf{v}^{\text{¢}}\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{u}\,)/\|\mathbf{v}\|^2 \;=\; \mathbf{u} + \mathbf{v}\cdot\mathbf{v}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{v}\|^2$.

#**5.** Given two points $\mathbf{u}$ and $\mathbf{u}+\mathbf{v}$ through which one line $£$ passes, and two points $\mathbf{y}$ and $\mathbf{y}+\mathbf{w}$ through which another line ¥ passes, the point nearest $£$ in ¥ is $\mathbf{x} := \mathbf{y} + \mathbf{w}\cdot\mathbf{v}^T\cdot\mathbf{p}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2$ wherein $\mathbf{p} = \mathbf{v}^{\text{¢}}\cdot\mathbf{w}$. Nearest ¥ in $£$ is $\mathbf{z} := \mathbf{x} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2 = \mathbf{s} := \mathbf{u} + \mathbf{v}\cdot\mathbf{w}^T\cdot\mathbf{p}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2$.

#**6.** Given two points $\mathbf{u}$ and $\mathbf{w}$ through which a line $£$ passes, the point $\mathbf{z}$ in $£$ nearest $\mathbf{y}$ is $\mathbf{z} := \mathbf{y} + \mathbf{v}^{\text{¢}}\cdot\mathbf{v}^{\text{¢}}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{v}\|^2 \;=\; (\,\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} - \mathbf{v}^{\text{¢}}\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{u}\,)/\|\mathbf{v}\|^2 \;=\; \mathbf{u} + \mathbf{v}\cdot\mathbf{v}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{v}\|^2$ wherein $\mathbf{v} := \mathbf{w} - \mathbf{u}$. Since $\mathbf{u}$ and $\mathbf{w}$ are permutable, choose $\mathbf{u}$ to minimize $\|\mathbf{y}-\mathbf{u}\|$ in the first and last formulas, and to minimize $\|\mathbf{u}\|$ in the middle formula, which is best if $\|\mathbf{z}\| \ll \|\mathbf{u}\|$ too.

#**7.** Given the two equations $\mathbf{p}^T\cdot\mathbf{x} = \pi$ and $\mathbf{b}^T\cdot\mathbf{x} = ß$ of a line $£$, the point $\mathbf{z}$ in $£$ nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(ß-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,)/\|\mathbf{v}\|^2 \;=\; (\,\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\mathbf{p}\cdot ß-\mathbf{b}\cdot\pi)\,)/\|\mathbf{v}\|^2$$

wherein $\mathbf{v} := \mathbf{p}^{\text{¢}}\cdot\mathbf{b}$. Of course we assume $\mathbf{v} \neq \mathbf{o}$ in order that $£$ be determined uniquely.

#**8.** Given three non-collinear points $\mathbf{u}–\mathbf{v}$, $\mathbf{u}$ and $\mathbf{u}+\mathbf{w}$ in Euclidean 2- and 3-space, the point

$$\mathbf{z} := \mathbf{u} + \tfrac{1}{2}(\mathbf{v}^{¢}\mathbf{w})^{¢}\cdot(\mathbf{v}\cdot||\mathbf{w}||^2 + \mathbf{w}\cdot||\mathbf{v}||^2)/||\mathbf{v}^{¢}\mathbf{w}||^2 \;=\; \mathbf{u} + \tfrac{1}{2}(||\mathbf{v}||^2\cdot\mathbf{w}\mathbf{w}^T – ||\mathbf{w}||^2\cdot\mathbf{v}\mathbf{v}^T)\cdot(\mathbf{v} + \mathbf{w})/||\mathbf{v}^{¢}\mathbf{w}||^2$$

is the center of the circle through the three given points, and this circle's radius is

$$||\mathbf{z}–\mathbf{u}|| = \tfrac{1}{2}||\mathbf{v}||\cdot||\mathbf{w}||\cdot||\mathbf{v}+\mathbf{w}||/||\mathbf{v}^{¢}\mathbf{w}|| \,.$$

When the given points are infinitesimally close neighbors on a smooth curve traced by $\mathbf{u} = \mathbf{u}(\tau)$ with non-vanishing velocity $\mathbf{u'} = \mathbf{u'}(\tau) := d\mathbf{u}(\tau)/d\tau$ and acceleration $\mathbf{u''} = \mathbf{u''}(\tau) := d\mathbf{u'}(\tau)/d\tau$, the *Osculating Circle* that matches the curve's tangent and curvature at $\mathbf{u}$ is centered at the curve's *Center of Curvature* $\mathbf{c} := \mathbf{u} + ||\mathbf{u'}||^2\cdot(\mathbf{u'}^{¢}\cdot\mathbf{u''})^{¢}\cdot\mathbf{u'}/||\mathbf{u'}^{¢}\cdot\mathbf{u''}||^2$. The circle's radius is the curve's *Radius of Curvature* $||\mathbf{c}–\mathbf{u}|| = ||\mathbf{u'}||^3/||\mathbf{u'}^{¢}\mathbf{u''}||$. These formulas are derived in §12.

When $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ lie in a Euclidean 2-space, some of the formulas above simplify: The center $\mathbf{z} := \mathbf{u} + \tfrac{1}{2}J\cdot(\mathbf{v}\cdot||\mathbf{w}||^2 + \mathbf{w}\cdot||\mathbf{v}||^2)/(\mathbf{w}^TJ\mathbf{v})$; the radius $||\mathbf{z}–\mathbf{u}|| = \tfrac{1}{2}||\mathbf{v}||\cdot||\mathbf{w}||\cdot||\mathbf{v}+\mathbf{w}||/|\mathbf{w}^TJ\mathbf{v}|$; the center of curvature $\mathbf{c} := \mathbf{u} + ||\mathbf{u'}||^2J\mathbf{u'}/(\mathbf{u''}^TJ\cdot\mathbf{u'})$; and the radius of curvature is $||\mathbf{c}–\mathbf{u}|| = ||\mathbf{u'}||^3/|\mathbf{u''}^TJ\cdot\mathbf{u'}|$. Center $\mathbf{z}$ figures in *Delaunay Triangulations* used to construct well-shaped triangular meshes over plane regions for use in continuum computations of fluid flows and elastic deformations.
Some of the formulas above work in Euclidean n-space too after $||\mathbf{v}^{¢}\mathbf{w}||$ is replaced by $\sqrt{(||\mathbf{v}||^2\cdot||\mathbf{w}||^2 – (\mathbf{v}^T\mathbf{w})^2)}$.

§10: An Unfortunate Numerical Example

It is not at all obvious that formula #7, say, is numerically unstable. In fact all figures carried can be lost if a few too few are carried. Try these data all stored exactly as 4-byte `floats` :

$\mathbf{p}^T = [\,38006,\ 23489,\ 14517\,]$, $\pi = 8972$, $\mathbf{b}^T = [\,23489,\ 14517,\ 8972\,]$, $ß = 5545$, and $\mathbf{y}^T = [\,1,\ –1,\ 1\,]$.

These data define £ as the intersection of two nearly parallel planes, so tiny changes in data can alter £ and $\mathbf{z}$ drastically. More troublesome numerically are the many correlated appearances of the data $\mathbf{p}$ and $\mathbf{b}$ in the formulas for $\mathbf{z}$ in problem #7. Though mathematically crucial, these correlations can be ruined by roundoff. Evaluating both formulas above for $\mathbf{z}$ naively in `float` arithmetic carrying 24 sig. bits (about as precise as 7 sig. dec.) yields $\mathbf{z}_1^T = [\,1,\ 1,\ –1\,]$ and $\mathbf{z}_2^T = [\,1,\ 1,\ –1.5\,]$; but $\mathbf{z}_1$ lies farther from both planes than 0.8, and $\mathbf{z}_2$ lies farther from them than 0.6. These gaps cannot be attributed to end-figure "errors" in the 7th sig. dec. of the given data which could shift £ and $\mathbf{z}$ appreciably but not separate them.

This naïve arithmetic produces geometrically impossible results.

The correct point $\mathbf{z}^T = [\,1/3,\ 2/3,\ –4/3\,]$ is computed correctly rounded when all intermediate results (sub-expressions and local variables) are evaluated in 53 sig. bit `double` before $\mathbf{z}$ is rounded back to `float`. Naively computed $\mathbf{z}_1$ and $\mathbf{z}_2$ are not so far from $\mathbf{z}$ as to be obviously wrong if $\mathbf{z}$ were unknown, but they are too wrong to be acceptable for most purposes.

This unfortunate example exemplifies behavior that occurs surprisingly often: Unless evaluated extra-precisely, too many neat formulas can be intolerably more inaccurate than the data deserve.

WHY ?

§11: Bilinear forms vulnerable to roundoff followed by cancellation occur frequently:

Scalar products: $\mathbf{p}\bullet\mathbf{b} = \mathbf{p}^T\cdot\mathbf{b} = p_1\cdot b_1 + p_2\cdot b_2 + p_3\cdot b_3$.

Linear combinations: $\mathbf{p}\cdot ß - \mathbf{b}\cdot\pi = \begin{bmatrix} p_1\cdot\beta - b_1\cdot\pi \\ p_2\cdot\beta - b_2\cdot\pi \\ p_3\cdot\beta - b_3\cdot\pi \end{bmatrix}$.

Cross products: $\mathbf{p}\times\mathbf{b} = \mathbf{p}^{¢}\cdot\mathbf{b} = \begin{bmatrix} p_2\cdot b_3 - p_3\cdot b_2 \\ p_3\cdot b_1 - p_1\cdot b_3 \\ p_1\cdot b_2 - p_2\cdot b_1 \end{bmatrix}$.

These entities are *geometrically redundant*; they are so correlated that $(\mathbf{p}\cdot ß - \mathbf{b}\cdot\pi)\bullet(\mathbf{p}\times\mathbf{b}) = 0$ for *all* data $\{\mathbf{p}, \pi, \mathbf{b}, ß\}$. Even if data are "accurate" to few sig. digits and computed entities to fewer, their geometrical redundancy must be conserved as accurately as possible. We can tolerate slightly inaccurate results interpretable as realizable geometrical objects slightly different from our original intent, but not geometrically impossible objects like a $\mathbf{p}\times\mathbf{b}$ too far from orthogonal to $\mathbf{p}$ and $\mathbf{b}$ because of roundoff. Suppose ε is the roundoff threshold, meaning that sums, differences and products are computed accurately within a factor $1\pm\varepsilon$. For instance, $\varepsilon = 5/10^{10}$ for arithmetic rounded to 10 sig. dec. Then the angle between the desired cross product $\mathbf{p}\times\mathbf{b}$ and its computed version will be typically about $\pm\varepsilon/\sin(\angle(\mathbf{p}, \mathbf{b}))$ and is proved never to get much bigger in §13. This shows how roundoff degrades $\mathbf{p}\times\mathbf{b}$ as $\mathbf{p}$ and $\mathbf{b}$ approach (anti)parallelism and, in view of the *Sine Law of Triangles*, justifies advice about $(\mathbf{v} - \mathbf{u})^{¢}\cdot(\mathbf{w} - \mathbf{u})$ in problem #3.

Therefore these bilinear forms and other matrix products should be computed carrying somewhat more precision than in the data provided that this extra-precise arithmetic runs at some adequate speed. Doing so preserves geometrical redundancy despite "losses" to cancellation of several digits. At any precision, prolonged chains of computation risk losing geometrical redundancy. The wider is the precision, the longer is that loss postponed and the more often prevented.

And extra precision usually costs less than error-analysis.

This is not said to disparage error-analysis; it is always the right thing to do if you know how and have the time. But to know how you have to take advanced classes in numerical analysis since elementary classes hardly ever teach error-analysis well enough to be useful. To spend enough time you have to believe that the results being (in)validated by error-analysis are worth the time.

For extensive discussions of these and similar computational issues see …

"Marketing versus Mathematics" `<www.cs.berkeley.edu/~wkahan/MktgMath>`
"How Java's Floating-Point Hurts Everybody Everywhere" *ibid.* …`/JAVAhurt.pdf>`
"Miscalculating Area and Angles of a Needle-like Triangle" *ibid.* …`/Triangle.pdf>`
"MATLAB's Loss is Nobody's Gain" *ibid.* …`/MxMulEps.pdf>`
"How Futile are Mindless Assessments of Roundoff in Floating-Point Computation ?" *ibid.* …`/Mindless.pdf>`
Prof. Jonathan Shewchuk's web page, starting at `<www.cs.berkeley.edu/~jrs/meshpapers/robnotes.ps.gz>`
…..

A good book about Error-Analysis is N.J. Higham's *Accuracy and Stability of Numerical Algorithms* 2d ed. (2002, Soc. Indust. & Appl. Math., Philadelphia), though it is about 700 pages long.

§12: Confirmations of the Eight Formulas in § 9 :

#1. Given the equation $\mathbf{p}^T\cdot\mathbf{x} = \pi$ of a plane Π, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot(\mathbf{p}^T\cdot\mathbf{y} - \pi)/\|\mathbf{p}\|^2 = \mathbf{s} := (\mathbf{p}\cdot\pi - \mathbf{p}^{¢}\cdot\mathbf{p}^{¢}\cdot\mathbf{y})/\|\mathbf{p}\|^2 .$$

To confirm these formulas we must verify that $\mathbf{p}^T\cdot\mathbf{z} = \pi$ to put $\mathbf{z}$ in Π, and that $\mathbf{z}-\mathbf{y}$ is (anti)parallel to $\mathbf{p}$ so that it is perpendicular to Π. And then we must verify that $\mathbf{z} = \mathbf{s}$. Only this last equality is unobvious; confirm it by substituting $(\mathbf{p}^{¢})^2 = \mathbf{p}\cdot\mathbf{p}^T - \|\mathbf{p}\|^2\cdot\mathrm{I}$.

#2. Given three points $\mathbf{u}-\mathbf{v}$, $\mathbf{u}$ and $\mathbf{u}+\mathbf{w}$ through which one plane Π passes, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is $\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{p}\|^2 = \mathrm{s} := \mathbf{u} - \mathbf{p}^{¢}\cdot\mathbf{p}^{¢}\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{p}\|^2$ wherein $\mathbf{p} := \mathbf{v}^{¢}\cdot\mathbf{w}$. These formulas follow from problem **1** because the plane's equation is $\mathbf{p}^T\cdot\mathbf{x} = \pi := \mathbf{p}^T\cdot\mathbf{u}$.

#3. Given three points $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ through which one plane Π passes, the point $\mathbf{z}$ in Π nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{p}\|^2 = \mathrm{s} := \mathbf{u} - \mathbf{p}^{¢}\cdot\mathbf{p}^{¢}\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{p}\|^2 \quad \text{wherein} \quad \mathbf{p} := (\mathbf{v} - \mathbf{u})^{¢}\cdot(\mathbf{w} - \mathbf{u}) .$$

These formulas follow from problem #2 after its $\mathbf{v}$ and $\mathbf{w}$ are replaced by $\mathbf{u}-\mathbf{v}$ and $\mathbf{w}-\mathbf{u}$.

#4. Given two points $\mathbf{u}$ and $\mathbf{u}+\mathbf{v}$ through which one line £ passes, the point $\mathbf{z}$ in £ nearest $\mathbf{y}$ is $\mathbf{z} := \mathbf{y} + \mathbf{v}^{¢}\cdot\mathbf{v}^{¢}\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{v}\|^2 = \mathrm{s} := (\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} - \mathbf{v}^{¢}\cdot\mathbf{v}^{¢}\cdot\mathbf{u})/\|\mathbf{v}\|^2 = \mathbf{t} := \mathbf{u} + \mathbf{v}\cdot\mathbf{v}^T\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{v}\|^2$. To confirm these formulas we must verify that $\mathbf{z}-\mathbf{u}$ is a scalar multiple of $\mathbf{v}$, which places $\mathbf{z}$ on £, and that $\mathbf{v}^T(\mathbf{z}-\mathbf{y}) = 0$ so that $\mathbf{z}-\mathbf{y}$ is perpendicular to £. Since $(\mathbf{v}^{¢})^2 = \mathbf{v}\cdot\mathbf{v}^T - \|\mathbf{v}\|^2\cdot\mathrm{I}$, we find that $\mathbf{z}-\mathbf{u} = \mathbf{v}\cdot\mathbf{v}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{v}\|^2 = \mathbf{t}-\mathbf{u}$ *is* a scalar multiple of $\mathbf{v}$ and, incidentally, $\mathbf{z} = \mathbf{t}$. And $\mathbf{v}^T(\mathbf{z}-\mathbf{y}) = 0$ follows from $\mathbf{v}^T\mathbf{v}^{¢} = \mathbf{o}^T$. Finally $\mathbf{s} = \mathbf{t}$ follows from the expansion of $(\mathbf{v}^{¢})^2$.

#5. Given two points $\mathbf{u}$ and $\mathbf{u}+\mathbf{v}$ through which one line £ passes, and two points $\mathbf{y}$ and $\mathbf{y}+\mathbf{w}$ through which another line ¥ passes, the point nearest £ in ¥ is $\mathbf{x} := \mathbf{y} + \mathbf{w}\cdot\mathbf{v}^T\cdot\mathbf{p}^{¢}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2$ wherein $\mathbf{p} = \mathbf{v}^{¢}\cdot\mathbf{w}$. Nearest ¥ in £ is $\mathbf{z} := \mathbf{x} - \mathbf{p}\cdot\mathbf{p}^T\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2 = \mathbf{s} := \mathbf{u} + \mathbf{v}\cdot\mathbf{w}^T\cdot\mathbf{p}^{¢}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2$. To confirm these formulas, we confirm first that $\mathbf{x}$ lies in ¥ because $\mathbf{x}-\mathbf{y} = \mathbf{w}\cdot(\text{scalar})$, and that $\mathbf{s}$ lies in £ because $\mathbf{s}-\mathbf{u} = \mathbf{v}\cdot(\text{scalar})$. Then $\mathbf{z} = \mathbf{s}$ because

$$\begin{aligned}\|\mathbf{p}\|^2\cdot(\mathbf{z}-\mathbf{s}) &= (\|\mathbf{p}\|^2\cdot\mathrm{I} - \mathbf{p}\cdot\mathbf{p}^T + \mathbf{w}\cdot\mathbf{v}^T\cdot\mathbf{p}^{¢} - \mathbf{v}\cdot\mathbf{w}^T\cdot\mathbf{p}^{¢})(\mathbf{y}-\mathbf{u}) \quad \ldots \text{ recall } (\mathbf{p}^{¢})^2 = \mathbf{p}\cdot\mathbf{p}^T - \|\mathbf{p}\|^2\cdot\mathrm{I} \\ &= (-\mathbf{p}^{¢} + \mathbf{w}\cdot\mathbf{v}^T - \mathbf{v}\cdot\mathbf{w}^T)\cdot\mathbf{p}^{¢}\cdot(\mathbf{y}-\mathbf{u}) = \mathbf{O}\cdot\mathbf{p}^{¢}\cdot(\mathbf{y}-\mathbf{u}) = \mathbf{o}\end{aligned}$$

in view of the triple cross-product formula $\mathbf{p}^{¢} = (\mathbf{v}^{¢}\mathbf{w})^{¢} = \mathbf{w}\cdot\mathbf{v}^T - \mathbf{v}\cdot\mathbf{w}^T$ in §3 and §5. Finally, $\mathbf{z}-\mathbf{x} = \mathbf{p}\cdot(\text{scalar})$ is perpendicular to both lines ¥ and £, so $\mathbf{z}$ and $\mathbf{x}$ are nearest each other.

Lest binocular vision's depth perception and range-finding falter when £ and ¥ almost intersect, $(\mathbf{x}+\mathbf{z})/2 = \left(\mathbf{u}+\mathbf{y} + (\mathbf{w}\cdot\mathbf{v}^T + \mathbf{v}\cdot\mathbf{w}^T)\cdot\mathbf{p}^{¢}\cdot(\mathbf{y}-\mathbf{u})/\|\mathbf{p}\|^2\right)/2$ is the point nearest their near-intersection.

#6. Given two points $\mathbf{u}$ and $\mathbf{w}$ through which a line £ passes, the point $\mathbf{z}$ in £ nearest $\mathbf{y}$ is $\mathbf{z} = \mathbf{y} + \mathbf{v}^{¢}\cdot\mathbf{v}^{¢}\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{v}\|^2 = (\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} - \mathbf{v}^{¢}\cdot\mathbf{v}^{¢}\cdot\mathbf{u})/\|\mathbf{v}\|^2 = \mathbf{u} + \mathbf{v}\cdot\mathbf{v}^T\cdot(\mathbf{y} - \mathbf{u})/\|\mathbf{v}\|^2$ wherein $\mathbf{v} = \mathbf{w} - \mathbf{u}$. These formulas follow from problem #4.

#7. Given the two equations $\mathbf{p}^T\cdot\mathbf{x} = \pi$ and $\mathbf{b}^T\cdot\mathbf{x} = \text{ß}$ of a line **£**, the point $\mathbf{z}$ in **£** nearest $\mathbf{y}$ is

$$\mathbf{z} := \mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,)/\|\mathbf{v}\|^2 \;=\; \mathbf{s} := (\,\mathbf{v}\cdot\mathbf{v}^T\cdot\mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\mathbf{p}\cdot\text{ß}-\mathbf{b}\cdot\pi)\,)/\|\mathbf{v}\|^2$$

wherein $\mathbf{v} := \mathbf{p}^{\text{¢}}\cdot\mathbf{b} \ne \mathbf{o}$. This $\mathbf{v}$ is parallel to **£** because it is perpendicular to the normals of both planes that intersect in **£** . To confirm that $\mathbf{z}$ is the point in **£** nearest $\mathbf{y}$ we must verify that $\mathbf{z}$ lies in both those planes and that $\mathbf{v}$ is perpendicular to $\mathbf{z}-\mathbf{y}$. In other words, we must verify that $\mathbf{p}^T\cdot\mathbf{z} = \pi$, $\mathbf{b}^T\cdot\mathbf{z} = \text{ß}$, $\mathbf{v}^T\cdot(\mathbf{z}-\mathbf{y}) = 0$ and $\mathbf{s} = \mathbf{z}$. This was done with DERIVE™, a computerized algebra system, faster than the following dozen lines of a manual verification can be written out:

$$\mathbf{v}^T\cdot(\mathbf{z}-\mathbf{y}) = \mathbf{v}^T\cdot\mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,)/\|\mathbf{v}\|^2 = 0 \;\text{ because }\; \mathbf{v}^T\cdot\mathbf{v}^{\text{¢}} = \mathbf{o}^T .$$

$$\begin{aligned}
\mathbf{p}^T\cdot\mathbf{z} - \pi &= \mathbf{p}^T\cdot\mathbf{y} - \pi + \mathbf{p}^T\cdot\mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,)/\|\mathbf{v}\|^2 \\
&= \mathbf{p}^T\cdot\mathbf{y} - \pi - \mathbf{p}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 \quad \text{because } \mathbf{p}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{p} = 0 , \\
&= \mathbf{p}^T\cdot\mathbf{y} - \pi + \mathbf{v}^T\cdot\mathbf{p}^{\text{¢}}\cdot\mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 \quad \text{because } \mathbf{p}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{b} = -\mathbf{v}^T\cdot\mathbf{p}^{\text{¢}}\cdot\mathbf{b} , \\
&= \mathbf{p}^T\cdot\mathbf{y} - \pi + \mathbf{v}^T\cdot\mathbf{v}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 = 0 .
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}^T\cdot\mathbf{z} - \text{ß} &= \mathbf{b}^T\cdot\mathbf{y} - \text{ß} + \mathbf{b}^T\cdot\mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,)/\|\mathbf{v}\|^2 \\
&= \mathbf{b}^T\cdot\mathbf{y} - \text{ß} + \mathbf{b}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 \quad \text{because } \mathbf{b}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{b} = 0 , \\
&= \mathbf{b}^T\cdot\mathbf{y} - \text{ß} - \mathbf{v}^T\cdot\mathbf{b}^{\text{¢}}\cdot\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 \quad \text{because } \mathbf{b}^T\cdot\mathbf{v}^{\text{¢}}\cdot\mathbf{p} = -\mathbf{v}^T\cdot\mathbf{b}^{\text{¢}}\cdot\mathbf{p} , \\
&= \mathbf{b}^T\cdot\mathbf{y} - \text{ß} + \mathbf{v}^T\cdot\mathbf{v}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y})/\|\mathbf{v}\|^2 = 0 .
\end{aligned}$$

$$\begin{aligned}
(\mathbf{z}-\mathbf{s})\cdot\|\mathbf{v}\|^2 &= (\mathbf{v}^T\cdot\mathbf{v}\cdot\mathrm{I} - \mathbf{v}\cdot\mathbf{v}^T)\cdot\mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y})\,) - \mathbf{v}^{\text{¢}}\cdot(\mathbf{p}\cdot\text{ß}-\mathbf{b}\cdot\pi) \\
&= -(\mathbf{v}^{\text{¢}})^2\cdot\mathbf{y} + \mathbf{v}^{\text{¢}}\cdot(\,\mathbf{p}\cdot(\text{ß}-\mathbf{b}^T\cdot\mathbf{y}) - \mathbf{b}\cdot(\pi-\mathbf{p}^T\cdot\mathbf{y}) - (\mathbf{p}\cdot\text{ß}-\mathbf{b}\cdot\pi)\,) \\
&= \mathbf{v}^{\text{¢}}\cdot(\,-\mathbf{v}^{\text{¢}} - \mathbf{p}\cdot\mathbf{b}^T + \mathbf{b}\cdot\mathbf{p}^T\,)\cdot\mathbf{y} \;=\; \mathbf{v}^{\text{¢}}\cdot(\,-\mathbf{v}^{\text{¢}} + (\mathbf{p}^{\text{¢}}\cdot\mathbf{b})^{\text{¢}}\,)\cdot\mathbf{y} \;=\; \mathbf{o} .
\end{aligned}$$

#8. Given three non-collinear points $\mathbf{u}-\mathbf{v}$, $\mathbf{u}$ and $\mathbf{u}+\mathbf{w}$ in Euclidean 3-space, the center $\mathbf{z}$ of the circle through the given points is $\mathbf{z} := \mathbf{u} + \frac{1}{2}(\|\mathbf{v}\|^2\cdot\mathbf{w}\mathbf{w}^T - \|\mathbf{w}\|^2\cdot\mathbf{v}\mathbf{v}^T)\cdot(\mathbf{v}+\mathbf{w})/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2$, and the circle's radius is $\|\mathbf{z}-\mathbf{u}\| = \frac{1}{2}\|\mathbf{v}\|\cdot\|\mathbf{w}\|\cdot\|\mathbf{v}+\mathbf{w}\|/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|$. To prove these formulas this problem will be reduced to a problem solved previously.

Among points equidistant from the three given points, center $\mathbf{z}$ is nearest them. To simplify its derivation shift the origin to $\mathbf{u}$ temporarily. Now "equidistant" implies $\|\mathbf{v}+\mathbf{z}\|^2 = \|\mathbf{z}\|^2 = \|\mathbf{w}-\mathbf{z}\|^2$ which boils down to two linear equations $\mathbf{v}^T\mathbf{z} = -\|\mathbf{v}\|^2/2$ and $\mathbf{w}^T\mathbf{z} = \|\mathbf{w}\|^2/2$, restricting $\mathbf{z}$ to a straight line. This reduces problem #8 to an instance of problem #7 whose solution becomes $\mathbf{z} = \mathbf{u} + \frac{1}{2}(\mathbf{v}^{\text{¢}}\mathbf{w})^{\text{¢}}\cdot(\mathbf{v}\cdot\|\mathbf{w}\|^2 + \mathbf{w}\cdot\|\mathbf{v}\|^2)/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2 = \mathbf{u} + \frac{1}{2}(\|\mathbf{v}\|^2\cdot\mathbf{w}\mathbf{w}^T - \|\mathbf{w}\|^2\cdot\mathbf{v}\mathbf{v}^T)\cdot(\mathbf{v}+\mathbf{w})/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2$ after some simplification and restoration of the origin. Note that this formula makes $\mathbf{z}$ coplanar with the three given points, as must be expected. Now, $\mathbf{z}$ is the center of a circle whose radius is

$$\begin{aligned}
\|\mathbf{z}-\mathbf{u}\| &= \tfrac{1}{2}\|(\|\mathbf{v}\|^2\cdot\mathbf{w}\mathbf{w}^T - \|\mathbf{w}\|^2\cdot\mathbf{v}\mathbf{v}^T)\cdot(\mathbf{v}+\mathbf{w})\|/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2 \\
&= \tfrac{1}{2}\|\,\mathbf{w}\cdot\|\mathbf{v}\|^2\cdot(\|\mathbf{w}\|^2 + \mathbf{v}^T\mathbf{w}) - \mathbf{v}\cdot\|\mathbf{w}\|^2\cdot(\|\mathbf{v}\|^2 + \mathbf{v}^T\mathbf{w})\,\|/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2 = \dots \\
&= \tfrac{1}{2}\|\mathbf{v}\|\cdot\|\mathbf{w}\|\cdot\|\mathbf{v}+\mathbf{w}\|\cdot\sqrt{(\|\mathbf{v}\|^2\cdot\|\mathbf{w}\|^2 - (\mathbf{v}^T\mathbf{w})^2)}/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|^2 = \tfrac{1}{2}\|\mathbf{v}\|\cdot\|\mathbf{w}\|\cdot\|\mathbf{v}+\mathbf{w}\|/\|\mathbf{v}^{\text{¢}}\mathbf{w}\|
\end{aligned}$$

as claimed.

Next let $\mathbf{u} := \mathbf{u}(\tau)$ trace a smooth curve with non-vanishing velocity $\mathbf{u}' := \mathbf{u}'(\tau) := d\mathbf{u}(\tau)/d\tau$ and acceleration $\mathbf{u}'' := \mathbf{u}''(\tau) := d\mathbf{u}'(\tau)/d\tau$. The Taylor series $\mathbf{u}(\tau+\theta) = \mathbf{u} + \theta\cdot\mathbf{u}' + \theta^2\cdot\mathbf{u}''/2 + \dots$ determines $\mathbf{v} := \mathbf{u}(\tau) - \mathbf{u}(\tau-\phi) = \phi\cdot\mathbf{u}' - \phi^2\cdot\mathbf{u}''/2 + \dots$ and $\mathbf{w} := \mathbf{u}(\tau+\theta) - \mathbf{u}(\tau)$ in the foregoing first formula for $\mathbf{z}$ whose limit, as $\theta \to 0$ and $\phi \to 0$, turns into the *Center of Curvature*

$$\mathbf{c} = \mathbf{u} - \|\mathbf{u}'\|^2\cdot\mathbf{u}'^{\text{¢}}\cdot(\mathbf{u}'^{\text{¢}}\cdot\mathbf{u}'')/\|\mathbf{u}'^{\text{¢}}\cdot\mathbf{u}''\|^2 .$$

And then because $\mathbf{u}'^T\cdot(\mathbf{u}'^{\text{¢}}\cdot\mathbf{u}'') = 0$, the *Radius of Curvature* $\|\mathbf{c}-\mathbf{u}\| = \|\mathbf{u}'\|^3/\|\mathbf{u}'^{\text{¢}}\mathbf{u}''\|$.

§13: Rounding Error-Bounds for Angles

The angle between the desired cross product $\mathbf{p}\times\mathbf{b}$ and its computed version is alleged in §11 to never exceed $\varepsilon\cdot|\csc(\angle(\mathbf{p}, \mathbf{b}))|$ much. Here ε is the roundoff threshold for individual arithmetic operations. This means that executing an assignment statement " $x := y\cdot z$ " actually computes and stores some rounded value $x := (1\pm\varepsilon)\cdot y\cdot z$, which is how an unknown number between $(1-\varepsilon)\cdot y\cdot z$ and $(1+\varepsilon)\cdot y\cdot z$ shall be described. Similarly " $x := y-z$ " actually stores a number $x = (y-z)/(1\pm\varepsilon)$. In special circumstances more than this can be said about x ; for instance, if $1/2 \le y/z \le 2$ then " $x := y-z$ " actually stores $x = y-z$ exactly on almost all today's computers.

Problem: Perhaps aided by a calculator, explore and then confirm the last assertion for arithmetics performed as you might reasonably expect, and then find examples that would violate it if " 1/2 " were diminished or " 2 " increased.

In any event, ε is very tiny. $\varepsilon = 1/2^{24} \approx 5.96/10^8$ for 4-byte wide `floats`. For the 8-byte wide floating-point arithmetic used by MATLAB, $\varepsilon = 1/2^{53} =$ `eps/2` $\approx 1.11/10^{16}$. Thus, ε is so tiny that usually terms of order ε^2 can be disregarded. They have been and will be.

A strict version of §11's allegation is this: Let $\mathbf{w}$ be the column vector computed for $\mathbf{p}\times\mathbf{b} \ne \mathbf{o}$ from $\mathbf{p}$ and $\mathbf{b}$ using floating-point arithmetic whose roundoff threshold is ε ; then $|\sin(\angle(\mathbf{w}, \mathbf{p}\times\mathbf{b}))| \le \varepsilon + (2/\sqrt{3})\cdot\varepsilon\cdot|\csc(\angle(\mathbf{p}, \mathbf{b}))|$. A proof follows: (csc = 1/sin .)

To reduce the strain on aged eyes, subscripts and superscripts will be avoided wherever possible. Set column-vectors $\mathbf{p} := [x, y, z]^T$ and $\mathbf{b} := [e, f, g]^T$, so $\mathbf{p}\times\mathbf{b} = [y\cdot g-z\cdot f,\ \ z\cdot e-x\cdot g,\ \ x\cdot f-y\cdot e]^T$. The first element of the computed cross-product $\mathbf{w}$ is $\big((1\pm\varepsilon)\cdot y\cdot g - (1\pm\varepsilon)\cdot z\cdot f\big)/(1\pm\varepsilon)$; it's typical. Therefore $|\mathbf{w} - \mathbf{p}\times\mathbf{b}| \le \varepsilon\cdot|\mathbf{w}| + \varepsilon\cdot|\mathbf{p}^{\text{¢}}|\cdot|\mathbf{b}|$ *elementwise*; here $\mathbf{p}^{\text{¢}}$ is the skew matrix that produces $\mathbf{p}^{\text{¢}}\cdot\mathbf{b} = \mathbf{p}\times\mathbf{b}$. Consequently the Euclidean norm $\|\mathbf{w} - \mathbf{p}\times\mathbf{b}\| \le \varepsilon\cdot\|\mathbf{w}\| + \varepsilon\cdot\|\,|\mathbf{p}^{\text{¢}}|\cdot|\mathbf{b}|\,\|$. Now, $\|\,|\mathbf{p}^{\text{¢}}|\cdot|\mathbf{b}|\,\|^2 = |\mathbf{b}|^T\cdot|\mathbf{p}^{\text{¢}}|^2\cdot|\mathbf{b}| \le \|\mathbf{b}\|^2\cdot($ the biggest eigenvalue of $|\mathbf{p}^{\text{¢}}|\,)^2$ since $|\mathbf{p}^{\text{¢}}|$ is a real symmetric matrix. Its *Characteristic Polynomial* turns out to be

$$\Phi(\lambda) := \det(\lambda I - |\mathbf{p}^{\text{¢}}|\,) = \lambda^3 - \|\mathbf{p}\|^2\lambda - 2|x\cdot y\cdot z| \quad \text{wherein } \|\mathbf{p}\|^2 = x^2 + y^2 + z^2 > 0 .$$

To locate its zeros, the eigenvalues of $|\mathbf{p}^{\text{¢}}|$, we shall repeatedly use the *Arithmetic-Geometric Means Inequality,* which says that $(x^2 + y^2 + z^2)/3 \ge \sqrt[3]{x^2\cdot y^2\cdot z^2}$; it will be invoked in the equivalent form $|x\cdot y\cdot z| \le \|\mathbf{p}\|^3/\sqrt{27}$. Then substitution of trial arguments reveals that

$$\Phi(-2\|\mathbf{p}\|/\sqrt{3}) < 0 \le \Phi(-\|\mathbf{p}\|/\sqrt{3}) , \quad \text{and} \quad \Phi(0) \le 0 \le \Phi(2\|\mathbf{p}\|/\sqrt{3}) ,$$

so all three zeros of Φ (the eigenvalues of $|\mathbf{p}^{\text{¢}}|$) lie between $\pm2\|\mathbf{p}\|/\sqrt{3}$. From this follows that

$\|\mathbf{w}-\mathbf{p}\times\mathbf{b}\| \le \varepsilon\cdot\|\mathbf{w}\| + (2/\sqrt{3})\cdot\varepsilon\cdot\|\mathbf{p}\|\cdot\|\mathbf{b}\|$. Therefore $\|\mathbf{w}-\mathbf{p}\times\mathbf{b}\| \le \varepsilon\cdot\|\mathbf{p}\times\mathbf{b}\| + (2/\sqrt{3})\cdot\varepsilon\cdot\|\mathbf{p}\|\cdot\|\mathbf{b}\|$ after terms of order ε^2 are ignored. A diagram consisting of a triangle establishes that $\|\mathbf{w}-\mathbf{p}\times\mathbf{b}\| \ge |\sin(\angle(\mathbf{w}, \mathbf{p}\times\mathbf{b}))|\cdot\|\mathbf{p}\times\mathbf{b}\|$; and we know that $\|\mathbf{p}\times\mathbf{b}\| = \|\mathbf{p}\|\cdot\|\mathbf{b}\|\cdot|\sin(\angle(\mathbf{p}, \mathbf{b}))|$. Assemble the last three relations to deduce the desired conclusion, namely that
$|\sin(\angle(\mathbf{w}, \mathbf{p}\times\mathbf{b}))| \le \|\mathbf{w}-\mathbf{p}\times\mathbf{b}\|/\|\mathbf{p}\times\mathbf{b}\| \le \varepsilon + (2/\sqrt{3})\cdot\varepsilon\cdot\|\mathbf{p}\|\cdot\|\mathbf{b}\|/\|\mathbf{p}\times\mathbf{b}\| = \varepsilon + (2/\sqrt{3})\cdot\varepsilon\cdot|\csc(\angle(\mathbf{p}, \mathbf{b}))|$.

Actually " ε + " can be deleted whenever $\sin(\angle(\mathbf{p}, \mathbf{b}))$ is very tiny since then all elements of $\mathbf{p}\times\mathbf{b}$ are tiny because of cancellations during subtractions, which must then be exact, removing the divisors $(1\pm\varepsilon)$ from elements of $\mathbf{w}$.

Attempts to apply the last inequality can fail when the unoriented angle $|\angle(\mathbf{p}, \mathbf{b})|$ differs from 0 or π by so little more than $\sqrt{\varepsilon}$ that its best-known textbook formulas are vitiated by roundoff:

The familiar formula $0 \le |\angle(\mathbf{p}, \mathbf{b})| := \arccos(\,\mathbf{p}^T\mathbf{b}/(\|\mathbf{p}\|\cdot\|\mathbf{b}\|)\,) \le \pi$ can err by as much as $\sqrt{\varepsilon}$ when $|\mathbf{p}^T\mathbf{b}|/(\|\mathbf{p}\|\cdot\|\mathbf{b}\|)$ differs from 1 by less than several rounding errors, thus obliterating $\angle(\mathbf{p}, \mathbf{b})$ when it is tiny. This happens because arccos skips so quickly at such arguments:
$\arccos(1) = 0$; $\arccos(1-\varepsilon) \approx \sqrt{2\varepsilon}$; $\arccos(1-2\varepsilon) \approx 2\sqrt{\varepsilon}$; … ; $\arccos(\varepsilon-1) \approx \pi - \sqrt{2\varepsilon}$.
Consequently the arccos formula should be avoided when $|\angle(\mathbf{p}, \mathbf{b})|$ is near 0 or π .

Another formula almost as familiar is

$$\text{If } \mathbf{p}^T\mathbf{b} \ge 0 \text{ then } |\angle(\mathbf{p}, \mathbf{b})| = \arcsin(\,\|\mathbf{p}\times\mathbf{b}\|/(\|\mathbf{p}\|\cdot\|\mathbf{b}\|)\,)$$
$$\text{else } |\angle(\mathbf{p}, \mathbf{b})| = \pi - \arcsin(\,\|\mathbf{p}\times\mathbf{b}\|/(\|\mathbf{p}\|\cdot\|\mathbf{b}\|)\,) .$$

It can lose to roundoff as many as half the significant digits carried by the arithmetic when $1 - \|\mathbf{p}\times\mathbf{b}\|/(\|\mathbf{p}\|\cdot\|\mathbf{b}\|)$ is not much bigger than ε ; at such arguments arcsin skips by steps of order $\sqrt{\varepsilon}$ through angles near $\pi/2$. Avoid arcsin(…) for such angles.

More uniformly accurate than both familiar formulas is the following unfamiliar formula:

$$|\angle(\mathbf{p}, \mathbf{b})| = 2\cdot\arctan(\,\|\,\mathbf{p}/\|\mathbf{p}\| - \mathbf{b}/\|\mathbf{b}\|\,\| \,/\, \|\,\mathbf{p}/\|\mathbf{p}\| + \mathbf{b}/\|\mathbf{b}\|\,\|\,) .$$

Valid for Euclidean spaces of any dimension, it never errs by more than a modest multiple of ε .

(If the data's magnitudes are not so extreme that exponent over/underflow can invalidate the algebraically equivalent formula $|\angle(\mathbf{p}, \mathbf{b})| = 2\cdot\arctan(\,\|\,\mathbf{p}\cdot\|\mathbf{b}\| - \mathbf{b}\cdot\|\mathbf{p}\|\,\| \,/\, \|\,\mathbf{p}\cdot\|\mathbf{b}\| + \mathbf{b}\cdot\|\mathbf{p}\|\,\|\,)$, use it because it runs slightly faster.)

§14: Preconditioning a Cross-Product

Preconditioning is a process that speeds up a computation or enhances its result's accuracy by altering a problem's data without changing the problem nor its solution. The process may be worth applying to enhance the accuracies of the cross-products that figure in the solutions to the geometrical problems of §9. The geometrical impossibility of §10's solution for problem #7 comes mostly from the cross-product $\mathbf{v} := \mathbf{p}^{¢}\cdot\mathbf{b}$ evaluated without the extra-precise arithmetic recommended in §11 to alleviate inaccuracy when $\mathbf{p}$ and $\mathbf{b}$ are too nearly (anti)parallel.

> Understand that the preconditioning process described hereunder is an *Act of Desperation* justified only when extra-precise arithmetic is impracticable and the computed $\|\mathbf{p}^{¢}\cdot\mathbf{b}\|$ is at least an order of magnitude smaller than $\|\mathbf{p}\|\cdot\|\mathbf{b}\|$.

The process is motivated by the identity $\mathbf{p}^¢\cdot\mathbf{b} = \mathbf{p}^¢\cdot(\mathbf{b}-\mathbf{p}\cdot\rho)$ valid for *all* values of the scalar ρ. When $\mathbf{p}$ and $\mathbf{b}$ are nearly (anti)parallel the choice $\rho := \mathbf{b}^T\mathbf{p}/||\mathbf{p}||^2$ greatly reduces (minimizes) $||\mathbf{b}-\mathbf{p}\cdot\rho||$ below $||\mathbf{b}||$ and thus greatly reduces $|\csc(\angle(\mathbf{p}, \mathbf{b}-\mathbf{p}\cdot\rho))|$. According to the error-analysis in §13, this would enhance greatly the accuracy of the computed $\mathbf{p}^¢\cdot(\mathbf{b}-\mathbf{p}\cdot\rho)$ except for the extra rounding errors inherited from the computation of ρ and $\mathbf{b}-\mathbf{p}\cdot\rho$. These extra rounding errors have to be avoided if the preconditioning process of norm reduction is to work properly.

The process chooses ρ to approximate $\mathbf{b}^T\mathbf{p}/||\mathbf{p}||^2$ rounded to as many sig. digits (at least one) as allow $\mathbf{p}\cdot\rho$ to be computed exactly, after which almost as many sig. digits of a new $\mathbf{b} := \mathbf{b}-\mathbf{p}\cdot\rho$ should cancel during subtraction. Thus $\mathbf{b}$ is replaced by a new $\mathbf{b}$ of smaller norm without any change to $\mathbf{p}^¢\cdot\mathbf{b}$ unless the subtraction incurs a rounding error. If so the process stops. Otherwise the process of norm reduction may be repeated upon the new $\mathbf{b}$ and $\mathbf{p}$, perhaps after swapping them, until either $||\mathbf{p}^¢\cdot\mathbf{b}||$ will no longer be too much tinier than $||\mathbf{p}||\cdot||\mathbf{b}||$, or else it has been reduced so much that the given data deserve to be deemed *Geometrically Degenerate* (parallel).

A preconditioning process like this above will be illustrated by the data for problem #7 from §10: The planes whose coefficients are the two rows of $L := \begin{bmatrix}\mathbf{p}^T & \pi \\ \mathbf{b}^T & \beta\end{bmatrix} = \begin{bmatrix}38006 & 23489 & 14517 & 8972 \\ 23489 & 14517 & 8972 & 5545\end{bmatrix}$ intersect in the same line $\pounds$ as is determined in the same way by the two rows of $R\cdot L$ no matter what 2-by-2 invertible matrix R is chosen so long as the matrix multiplication incurs no rounding error. So long as all elements of R and L are integers, binary `float` arithmetic commits no rounding error provided no intermediate product's nor sum's magnitude exceeds $1/\varepsilon = 2^{24} = 16777216$. This proviso is violated during the naive computation of $\mathbf{p}^¢\cdot\mathbf{b}$ in `float` arithmetic, which then produces $[16,\ 0,\ 0]^T$ instead of the correct $\mathbf{p}^¢\cdot\mathbf{b} = [19,\ -19,\ -19]^T$. The proviso will be satisfied if no element of R exceeds 272 in magnitude. The first row of L is so nearly 1.61803 times the second that consecutive convergents of the continued fraction $1.61803 = 1 + 1/(1 + 1/(1 + \ldots))$, namely $144/89 \approx 1.61798$ and $233/144 \approx 1.61806$, supply suitable choices for the elements of $R := \begin{bmatrix}89 & -144 \\ -144 & 233\end{bmatrix}$. Replacing L by $L_1 := R\cdot L = \begin{bmatrix}118 & 73 & 45 & 28 \\ 73 & 45 & 28 & 17\end{bmatrix}$ replaces *exactly* the data $\begin{bmatrix}\mathbf{p}^T & \pi \\ \mathbf{b}^T & \beta\end{bmatrix}$ for problem #7 by reduced data two or three digits smaller without changing its solution nor $\mathbf{p}_1{}^¢\cdot\mathbf{b}_1 = [19\ \ -19\ \ -19]^T = \mathbf{p}^¢\cdot\mathbf{b}$ (since $\det(R) = 1$). Now the evaluations of §9's formulas solving problem #7 lose far less to roundoff since $\csc(\angle(\mathbf{p}_1, \mathbf{b}_1)) \approx 4.01\cdot10^2$ is so much smaller than $\csc(\angle(\mathbf{p}, \mathbf{b})) \approx 4.14\cdot10^7$. A further reduction by $R_1 := \begin{bmatrix}5 & -8 \\ -8 & 13\end{bmatrix}$ to $L_2 := R_1\cdot L_1 = \begin{bmatrix}6 & 5 & 1 & 4 \\ 5 & 1 & 4 & -3\end{bmatrix}$ has $\csc(\angle(\mathbf{p}_2, \mathbf{b}_2)) \approx 1.55$; now nothing substantial can be lost to roundoff and the process stops.

Of course the foregoing example is artificial. Its small integer data make the preconditioning process easier to justify and to apply than if the data were nonintegers even though, in principle, multiplication by an appropriate power of the arithmetic's *Radix* (10 for decimal, 2 for binary) can convert each floating-point datum to an integer no bigger than $\mathrm{Radix}/(2\varepsilon)$.

§15: Reminiscences and Recommendations

At Cambridge University in 1959 Dr. J.C.P. Miller taught me preconditioning similar to what is exemplified above. For most of his life he had used mechanical desktop calculators which display every digit of every intermediate result, some to be copied to paper for subsequent reuse. He had preconditioned linear equations, polynomial equations, discriminants and recurrences to extract better results than might have been expected from the calculators' eight- or ten-digit keyboard capacities used less deftly. Because of two trends in numerical computations on electronic computers still novel then, Miller had come to doubt whether they could be programmed to perform his tricks. One trend was the emergence of *Backward Error-Analysis* and a tendency to misunderstand and misapply it. A second trend was towards ever more diverse and mysterious floating-point arithmetics being built into the hardware of computers sprouting from a widening field of manufacturers.

A program admits a backward error-analysis just when the program's results are not much worse than if it had performed its computation exactly (with no rounding errors) upon data perturbed only in end-figures. Though such programs can produce grossly inaccurate results from what are deemed *Ill-Conditioned* data, the programs are deemed *Numerically Stable* because their data are usually uncertain in more than their last digits, so such programs' rounding errors usually add relatively little to their results' inherited uncertainties no matter how big these may be. Or so it is widely believed. Preconditioning such programs' data seems pointless if it merely reduces their results' uncertainties a little. In so far as this inference rests upon backward error-analyses it attempts to excuse gross inaccuracy by invoking phenomena that can at best explain it.

Actually, many a program admits no backward error-analysis; an instance is problem #7's solution in §9, as §10's data shows. And some of what a program may treat as data subject to roundoff are actually constants determined exactly by a problem's structure; an example is the coefficient matrix for a boundary-value problem discretized by finite differences. Moreover, data's uncertainties can correlate in ways roundoff cannot conserve. For instance, a structural engineering problem's matrix may have thousands of nonzero coefficients determined by only hundreds of physical and geometrical parameters whose uncertainties propagate into the matrix coefficients but not independently. Another instance is problem #7 whose data consist of eleven scalar parameters most of which appear at least three times in §9's formulas for the solution, but rounding errors in §10 destroy geometric integrity by disrupting these correlations. *Structured Matrices* provide further instances but they are a story for another day.

In the absence of an assessment of inherited uncertainties adequate to take account of their correlations, extra-precise arithmetic, when available, is the simplest way for a numerical program to produce computed results at least about as accurate as data deserve despite roundoff. A not-so-simple way to the same end, when extra-precise arithmetic is unavailable or intolerably slow, is preconditioning, if it is feasible.

To carry out preconditioning on a computer its programmer must know which arithmetic operations it can perform exactly, without roundoff. Few programmers possess this knowledge now, and fewer could depend upon it before about 1985. For example, from 1964 to 1967 IBM's 8-byte-wide floating-point arithmetic did not multiply exactly by 1.0 nor by any other power of 16 (its radix), did not get the same results from $0.5\cdot x$ as from $x/2.0$, and did not get an exact result from subtraction when it mostly cancelled. This anomalous subtraction and some other aberrations afflicted several other manufacturers' floating-point arithmetics for decades. Each radix 2, 3, 4, 8, 10, 16, 100 and 256 had been used by at least one of the computer arithmetics built during those early years.

Since before 1990 the floating-point arithmetic hardware of almost all computers on and under desks has mostly conformed to IEEE Standard 754 (1985) for Binary Floating-Point Arithmetic. They round every rational arithmetic operation and square root in an optimal and therefore predictable way, and with no substantial degradation of speed (that's the tricky bit). Consequently small integer data can be preconditioned relatively easily, as exemplified above.

Noninteger data is harder to precondition. To ease that task IEEE 754 prescribes an *Inexact* flag to be raised whenever a conforming arithmetic operation commits a rounding error. This flag would allow programs to either cease preconditioning after a rounding error is detected or else narrow the preconditioning multipliers for another attempt. Every program on a conforming computer is supposed to be able to sense, save, clear and restore this flag. However, the flag and some other requirements of IEEE 754 have gone unsupported by programming languages other than *C99*, and are threatened by atrophy now. Those other languages' designers and implementors, ignorant of the uses for requirements they chose not to support, have behaved too much like Shakespeare's *Othello*:

“Perplex’d in the extreme … threw a pearl away
Richer than all his tribe.”

The biggest difference between 1959’s computers and today’s is not their millions-fold increases in speed and storage capacity and communications bandwidth; it is their decline in size and price. Now almost anyone can own a computer and many of us own several, most of them idle most of the time. Back then anyone who could afford a computer could afford also to attach to it at least one numerical expert somewhat like a trained Asian elephant kept in place at night by a feeble anklet chain. Numerical experts are not so numerous now, perhaps because floating-point computation has become so cheap as to be hardly ever worth hiring an expert to ascertain its validity. If anyone observes a transient numerical anomaly while playing a computer game or listening to rock “music”, who cares?

Still, we hope some computations do get proved valid; they may figure in medical diagnosis via computerized tomography, or the configurations of pharmaceutical molecules, or financial risk management, or robot(ic)ized manufacture, or the control of speeding vehicles, or the deflections of loaded structures, or predictions of volcanic eruptions or vulnerability to earthquakes, or the propagation and focussing of shock-waves and flame-fronts, or tornado forecasts, *etc*. Not every programmer expert in one of those areas can be expected also to have achieved competency in the error-analysis of roundoff’s effect upon numerical computation. Since (re)educating innumerable roundoff-innumerate programmers is impractical, the burden of accommodating programmers’ predilections falls upon those of us in the computer industry who (re)design computer hardware and programming environments.

Ideally every program that invokes floating-point arithmetic should benefit *by default* (without having to request them explicitly) from extravagantly more precise intermediate variables and arithmetic than might at first be thought necessitated by the precision of data and the accuracy desired in results. And this extra precision should slow the program by little more than the extra time taken to move extra-precise variables through memory. Only when this extra time looms large need it compel applications programmers to consider how much narrower they can afford to declare the precisions of some large arrays of intermediate variables. Otherwise, a programming language that adopts the foregoing ideal policy — extra precision by default — tends to protect the program’s users from the programmer’s naïveté about roundoff.

The ideal policy above influenced the design in the late 1970s of Intel’s floating-point destined to be installed in IBM’s microprocessor-powered PCs in the early 1980s. At that time most of the world’s data fit into the 24 sig. bits of 4-byte-wide `float` formats supported on almost all micro- and minicomputers; and almost all the rest of the world’s data fit into 32 sig. bits and was supported well by the 48 - 56 sig. bits of the mainframes’ and large minis’ 8-byte-wide floating-point called “`double`” on all those machines except CDC *Cyber*s and *CRAY*s. The ideal policy had been tested serendipitously in the early 1970s by the language ***C*** running on Bell Labs’ DEC *PDP-11* minicomputer. A quirk in its floating-point board had induced B.W. Kernighan and D.M. Ritchie to make their ***C*** language evaluate all floating-point expressions in `double` regardless of the width, `float` or `double`, of operands. Consequently matrix and geometrical computations and solutions of differential equations, all starting from `float` data to produce `float` results, lost significantly less accuracy when programmed in ***C*** than when programmed, with the same floating-point expressions, in FORTRAN, the language predominant at the time. But, at the time, almost nobody noticed the difference in accuracy. It was like the dog that did not bark in the night, noticed only by Sherlock Holmes in Sir Arthur Conan-Doyle’s story *Silver Blaze*.

In 1980, with an eye to the future, Intel introduced a third 10-byte-wide 64 sig. bit floating-point format not much slower than the 53 sig. bits of the 8-byte-wide `double` and intended to grow to 16-bytes-wide if and when the market demanded it. Intel’s design was promptly imitated and improved by Motorola whose µ680x0 powered Apple’s early Macintoshes on which *SANE* (Standard Apple Numeric Environment) implemented the ideal policy and supported all the requirements of IEEE 754 before it had become an official standard. Floating-point arithmetic seemed safer now for use by the masses.

But then Othello’s mistake was repeated. Repeatedly.

In 1982 Bill Gates Jr. predicted that almost no IBM PC’s socket for Intel’s 8087 Numeric Coprocessor would ever be filled, so no good reason existed to change Microsoft’s compilers to support all three of the 8087’s floating-point formats. He was quite wrong on both counts, but Microsoft’s compilers still eschew its widest format. In the mid-1980s the ANSI X3J11 committee responsible for standardizing ***C*** acquiesced to demands from CDC and Cray Research to let ***C*** compiler writers choose FORTRANnish expression evaluation instead of Kernighan-Ritchie's. That undid their serendipity and accelerated the migration of scientific and engineering computations from `floats` to `doubles` on all computers but now defunct CDC's and Cray Research's. Early in the 1990s, just as programmers were beginning to appreciate *SANE* and praise it, John Sculley tried to put Apple into bed with IBM and switched Macintoshes to IBM's Power PC microprocessor although it could not support *SANE* on Power Macs. That liaison's *Taligent Inc*. lived only briefly; and recently Apple switched Macintoshes to Intel's microprocessors. These could support *SANE* but Macs don't yet; Apple's efforts focus now upon a far bigger market. In the mid 1990s James Gosling and Bill Joy at Sun Microsystems invented the programming language *Java* partly to cure ***C*** of pointer abuse but mostly to break Microsoft's stranglehold upon the computing industry. They pointedly avoided consultation with Sun's numerical experts when they adopted FORTRANnish expression evaluation instead of Kernighan-Ritchie's, and banned extra-precise arithmetic and any requirement of IEEE 754 they didn't like. *Java*'s floating-point is dangerous; see "How Java's Floating-Point Hurts Everyone Everywhere" posted on my web page at <www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf>. Meanwhile Microsoft's hegemony persists despite judicial intervention in several jurisdictions worldwide. The bigger battalions of lawyers win.

Bereft of support by programming languages, several requirements of IEEE 754 face atrophy. And, on newer microprocessors from both Intel and its clonemaker AMD, the underused extra-precise format is getting slower relative to `float` and `double`, though not yet so slow as the 16-byte-wide hexadecimal, binary and decimal formats in the latest IBM mainframe hardware. On the other hand, processors optimized for computerized games and entertainment have fast vectorized `float` arithmetic and rather slower `double`, neither conforming fully to IEEE 754 but produced in such high volume that they are too cheap to be overlooked by the builders of inexpensive supercomputers designed for massively parallel scientific and engineering computations. Moreover, programmers attempting aggressively to exploit parallelism will succumb to the temptation to use numerical algorithms whose reliability in the face of roundoff on those processors has not been ascertained fully. The users of these programs will be testing them unwittingly, much like the consumers of newly marketed drugs and unregulated herbal remedies..

What can be done to render floating-point computation less hazardous? The people who decide about programming languages, compilers, debuggers and hardware care little for numerical mathematics and will almost surely not read these notes. If you have read this far, you can influence them as a customer more knowledgeable than most who use their products. Your obligations are similar to those of consumers who must demand that cars come with some safety features like tempered glass, seatbelts, airbags, ABS and ESC; that foods and drugs be protected from adulteration and contamination; and so on. Don't wait for some calamity to spur needed action, albeit *The American Way*; it drowns in attempts to cast and shed blame and punishment. Besides, punishment for Sin befalls too much more the innocent concomitants than the deserving sinner, as when drunks drive. The more potent incentive to incorporate safeguards into a product is the belief that sufficiently many customers will know enough to choose safer products.

Think not of your Duty owed to Truth in Computation.
Think about Self-Defense.

§16: How Nearby Singularities Magnify Roundoff

A numerical program's roundoff-induced instability can be cured only after it has been diagnosed. Diagnosis requires scrutiny of every site in the program where a rounding error is committed that subsequent arithmetic operations may amplify intolerably, at least for some data. Since almost every rounding error looks insignificant when committed, identifying one that isn't resembles identifying a pickpocket in the crowd at a carnival; look for characteristic misbehavior.

What characterizes the amplification of rounding errors, ostensibly negligible when committed, into gross departures from the intended result of a computation?

We shall see below that such amplification is caused by *Singularities* too near the computation's data. A function's singularity is a discontinuity or a place where a derivative of the function becomes infinite. We have seen two examples in §13:

$$d\arcsin(x)/dx = 1/\sqrt{(1-x)(1+x)} \quad\text{and}\quad d\arccos(x)/dx = -1/\sqrt{(1-x)(1+x)} \quad\text{on } -1 \le x \le 1 .$$

These derivatives become infinite at $x = \pm1$ where $\arcsin(x) = \pm\pi/2$ or $\arccos(x) = 0$ or π, thus amplifying enormously any error x inherits from roundoff. Those singularities degrade the accuracies of §13's formulas $|\angle(\mathbf{p}, \mathbf{b})| = \arcsin(\ldots)$ and $|\angle(\mathbf{p}, \mathbf{b})| := \arccos(\ldots)$ as if these disliked innocuous data $\mathbf{p}$ and $\mathbf{b}$ for which $\angle(\mathbf{p}, \mathbf{b})$ is too near $\pm\pi/2$ or 0 or π. No such prejudice blights the formula $|\angle(\mathbf{p}, \mathbf{b})| = 2\cdot\arctan(\ldots)$ because $0 < d\arctan(x)/dx = 1/(1+x^2) \le 1$ throughout $-\infty < x < +\infty$, so inherited uncertainties of the order of $\varepsilon\cdot\|\mathbf{p}\|$ in $\mathbf{p}$ and $\varepsilon\cdot\|\mathbf{b}\|$ in $\mathbf{b}$ propagate through the $\arctan(\ldots)$ into an inherited uncertainty of the order of ε in $|\angle(\mathbf{p}, \mathbf{b})|$ exacerbated at most moderately by augmentations of order ε from roundoff during the evaluation of the arctan formula. Consequently this arctan formula deserves to be called "Numerically Stable". On the other hand, unless evaluated in arithmetic at least twice as precise as the data, the arccos and arcsin formulas are numerically unstable, or at least hypersensitive to roundoff, for an extremely narrow range of otherwise unexceptionable data $\mathbf{p}$ and $\mathbf{b}$.

Let's generalize the foregoing examples: Let $F(X)$ represent a program intended to compute a function $f(x)$ in floating-point. Here f and x may be vectors, likewise F and X; but the data X stored in the computer may differ from the intended x because of errors inherited from prior computations. And the text of program F has no names for the rounding errors committed by almost every one of its floating-point arithmetic operations. Let each of these rounding errors be named, and assemble all these names into an array r and rewrite $F(X) = f(X, r)$ to take account of them; then $f(x, o) = f(x)$. For example, take quadratic function $f(x) := 2x^2 - 5x + 3$ and program $F(X) := (2\cdot X - 5)\cdot X + 3$, so $f(x, r) = \big(((2\cdot x\cdot(1+\rho_1) - 5)/(1+\rho_2))\cdot x\cdot(1+\rho_3) + 3\big)/(1+\rho_4)$; $r := [\rho_1, \rho_2, \rho_3, \rho_4]^T$. About each rounding error ρ_j no more may be known than that $\varepsilon \ge |\rho_j|$. More may be known in some cases; here $\rho_1 = 0$ if the arithmetic is binary floating-point. And $\rho_4 = 0$ in binary or decimal whenever $(5 - \sqrt{17})/4 \approx 0.2192236 < X < 2.280776 \approx (5 + \sqrt{17})/4$ because then the last addition "$\ldots + 3$" happens to cancel enough to incur no rounding error.

Normally, in general, the error in X and the rounding errors ρ_j collected in array r are so tiny that their squares may be neglected, and then the error inherited in $F(X)$ from the error in X plus the error due to roundoff is $F(X) - f(x) = f(X, r) - f(x, o) \approx f_x(X, o)\cdot(X-x) + f_r(X, o)\cdot r$ roughly wherein derivatives $f_r(x, r) = \partial f(x, r)/\partial r$ and $f_x(x, r) := \partial f(x, r)/\partial x$ so $f_x(X, o) = f'(X)$. This $F(X) - f(x)$ is the *Absolute Error*; define the *Relative Error* $:= \|F(X) - f(x)\|/\|f(X)\|$ or else the array of *Relative Errors* $:= \big(F(X) - f(x)\big)./f(X)$ elementwise, roughly.

How can $\|F(X) - f(x)\|$ become huge when $\|X-x\|$ and $\|r\|$ are tiny? This can happen only when $\|f'(X)\|$ or $\|f_r(X, o)\|$ is enormous, which can happen only when X is near an argument x_∞ at which $\|f'(x_\infty)\| = \infty$ or $\|f_r(x_\infty, o)\| = \infty$. ($x_\infty$ may be complex.) Such arguments x_∞ are singularities of f or of f. If relative error $\|F(X) - f(x)\|/\|f(X)\|$ matters it may become huge also when X is very near a zero of f, thus adding its zeros to the list of singularities.

Quadratic example F has $ƒ'(X) = 4·X – 5$ and $f_r(X, o) = [2·X^2, 3 – ƒ(X), ƒ(X) – 3, ƒ(X)]$ so its absolute error has no finite singularity $x_∞$. However the relative error has singularities at the zeros $x_o = 1.5$ and $x_o = 1$ of $ƒ$ because $f_r(x_o, o) = [2·x_o^2, 3, –3, 0] ≠ o^T$ and therefore $|F(X) – ƒ(X)|/|ƒ(X)|$ can become far bigger than $ε$ (but not arbitrarily big) when X is very near either x_o , though no roundoff perturbs $F(x_o) = ƒ(x_o) = 0$. A far better program to compute $ƒ(X)$ with fairly high relative accuracy for *all* X is $F(X) := 2·(X – 1.5)·(X – 1)$; can you see why?

In general, the severity of an error in F(X) depends not only upon its numerical value but also upon how it is weighed or measured: By what norm ||…|| ? Absolute or Relative?

Different norms can differ in the weights they assign to the elements of a vector; they can differ by many orders of magnitude. These weights may be buried in the physical units associated with different elements. For instance, a simulation of a system to control the distribution of electricity may involve several different voltage levels at different places in the system:

generators driven by water or steam or gas turbines	1000 - 100000 V.
high-voltage transmission lines astride the countryside	100000 - 1000000 V.
municipal distribution sites (often disguised to look like homes)	10000 - 30000 V.
transformers on each block of a street	2000 - 10000 V.
wall-sockets in homes	110 - 240 V.
power inside computers controlling the system	2 - 12 V.
voltage fluctuations at transistors in the computers	0.001 - 0.01 V.
microwave antennas receiving signals from the system	0.000001 - 0.0001 V.

A one volt error is inconsequential in the first five places, severe in the sixth, and overwhelming in the last two. Consequently the voltages should not be measured in volts everywhere but instead in megavolts, kilovolts, millivolts and microvolts at appropriate places. Choices of appropriate units ensure that an error amounting to 0.01 units will signify the same severity, within an order of magnitude or two, regardless of which variable is afflicted. The choice of appropriate units is an important step towards the choice of a vector norm appropriate to measure a result's error.

The choice of norm affects the meanings of weasel-words like "near" though not the locations of singularities that can amplify roundoff intolerably if data lies too near them. These singularities are determined first by the desired function $ƒ$, and perhaps by its zeros too if relative error is in question, and second by the program $F(X) = f(X, r)$ chosen to compute $ƒ(X)$. The program is usually deemed "Numerically Stable" if every singularity of the second kind is also of the first. But if a singularity $x_∞$ of the second kind (where $f_r(x_∞, o)$ is infinite) is not also of the first, the program is usually deemed "Numerically Unstable" at least for some data X too near $x_∞$.

Weasel-words "near", "some" and "usually" persist because first appearances can deceive. A program's singularities can easily evade casual inspection. For striking examples see my web page's postings "Why is Floating-Point Computation so Hard to Debug when it Goes Wrong ?" `<www.cs.berkeley.edu/~wkahan/WrongR.pdf>`, pp. 36-41 of "Marketing *vs.* Mathematics" `<…/MktgMath.pdf>` and then pp. 14 - 17 of "How Futile are Mindless Assessments of Roundoff in Floating-Point Computation?" `<…/Mindless.pdf>`; in each program's data lurks at least one datum masquerading as a constant, perhaps a loop-count too near a singularity at ∞ . Though a few singularities are harmless exhibitionists, some hide in plain sight. Let's look at instances:

§17: Spurious Singularities Plain and Not so Simple

Problem #0 in §9 concerned the intersection $\mathbf{z}$ of three planes whose equations $\mathbf{p}^T\cdot\mathbf{x} = \pi$, $\mathbf{b}^T\cdot\mathbf{x} = ß$ and $\mathbf{q}^T\cdot\mathbf{x} = \theta$ are given; $\mathbf{z}$ satisfies $M\cdot\mathbf{z} = \mathbf{m}$ wherein matrices $M := [\mathbf{p},\ \mathbf{b},\ \mathbf{q}]^T$ and $\mathbf{m} := [\pi,\ ß,\ \theta]^T$. The formula $\mathbf{z} = M^{-1}\cdot\mathbf{m}$ defines a function $\mathbf{z} = f(M, \mathbf{m})$ we wish to compute without saying how to compute it; §9's formula for $\mathbf{z} = (\ \mathbf{b}^{¢}\cdot\mathbf{q}\cdot\pi + \mathbf{q}^{¢}\cdot\mathbf{p}\cdot ß + \mathbf{p}^{¢}\cdot\mathbf{b}\cdot\theta\ )/(\mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q})$ defines a program equivalent to writing $M^{-1} = [\mathbf{b}^{¢}\cdot\mathbf{q},\ \mathbf{q}^{¢}\cdot\mathbf{p},\ \mathbf{p}^{¢}\cdot\mathbf{b}]/(\mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q})$ in which just one scalar divisor $\mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q} = \det(M)$ appears. It reveals a singularity of the desired f, as does its derivative: $d\mathbf{z} = M^{-1}\cdot(d\mathbf{m} - dM\cdot\mathbf{z})$. At this singularity, $\det(M) = 0$ and M^{-1} is nonexistent or infinite, corresponding to a *degenerate*, *collapsed* or *confluent* (choose whichever pejorative adjective you dislike least) geometrical configuration of three planes whose normals are coplanar.

Often, nearness to the singularity and its amplification of roundoff and infinitesimal perturbations $d\mathbf{m}$ and dM in data are captured by one number, the *Condition Number* $\kappa(M) := ||M^{-1}||\cdot||M||$. Classroom note `<www.cs.berkeley.edu/~wkahan/MathH110/NORMlite.pdf>` explains norms $||\dots||$, and `<.../GIlite.pdf>` explains more of what is summarized tersely hereunder:

•1: $1/||M^{-1}|| = \min ||M - S||$ over all *Singular* ($\det(S) = 0$) matrices S of M's dimensions. In the vector space of all matrices like M the singular matrices S fill out a complicated self-intersecting cone with vertex at the origin O; and $\arcsin(1/\kappa(M))$ is the angle between the cone and the line segment joining O to M.

•2: $d\mathbf{z} = M^{-1}\cdot(d\mathbf{m} - dM\cdot\mathbf{z})$ implies $||d\mathbf{z}||/||\mathbf{z}|| \le \kappa(M)\cdot(||d\mathbf{m}||/||\mathbf{m}|| + ||dM||/||M||)$, and for no constant smaller than $\kappa(M)$ can the inequality hold for all $\mathbf{m}$, $d\mathbf{m}$ and dM.

Were that all there is to singularity, error-analysis would be far simpler than it is. Placing blame for inaccuracy in a computed $M^{-1}\cdot\mathbf{m}$ upon amplification of roundoff by a huge condition number $\kappa(M)$, even if justified, can oversimplify the situation unjustly.

It is unjust because the data $(M, \mathbf{m})$ come from some observations or prior computation which can so correlate M and $\mathbf{m}$ that $\mathbf{z} = M^{-1}\cdot\mathbf{m}$ and its derivatives stay bounded no matter how "near" M comes to singular. This renders $\kappa(M)$ in •2 unrealistic for correlated perturbations in data though uncorrelated rounding errors can get amplified by as much as $\kappa(M)$. An extreme example is provided by the following familiar geometrical problem:

#**9.** Find *Incenter* $\mathbf{c}$ of the circle inscribed in a triangle whose vertices $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are given.

$\mathbf{c}$ is determined by requiring that it lie in the plane containing the triangle, and that a line segment from $\mathbf{c}$ to any vertex make equal angles with the sides adjacent to that vertex. The requirements imply three independent equations to be solved for $\mathbf{c}$; they have been assembled into one linear system $M\cdot\mathbf{c} = \mathbf{m}$ whose 3-by-3 matrix $M := [\mathbf{p},\ \mathbf{b},\ \mathbf{q}]^T$ and column $\mathbf{m} := [\pi,\ ß,\ \theta]^T$ have …

$\mathbf{p} := (\mathbf{v}-\mathbf{u})^{¢}\cdot(\mathbf{w}-\mathbf{u})$,	$\pi := \mathbf{p}^T\mathbf{u}$,	… in the triangle's plane
$\mathbf{b} := ||\mathbf{u}-\mathbf{v}||\cdot(\mathbf{w}-\mathbf{v}) - ||\mathbf{w}-\mathbf{v}||\cdot(\mathbf{u}-\mathbf{v})$,	$\beta := \mathbf{b}^T\mathbf{v}$,	… bisect angle at $\mathbf{v}$
$\mathbf{q} := ||\mathbf{v}-\mathbf{w}||\cdot(\mathbf{u}-\mathbf{w}) - ||\mathbf{u}-\mathbf{w}||\cdot(\mathbf{v}-\mathbf{w})$, and	$\theta := \mathbf{q}^T\mathbf{w}$.	… bisect angle at $\mathbf{w}$

Now $\mathbf{c} = M^{-1}\cdot\mathbf{m}$. Where are its singularities? This linear system's determinant turns out to be

$$\det(M) = \mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q} = \|\mathbf{v}-\mathbf{w}\|\cdot(\|\mathbf{u}-\mathbf{v}\| - \|\mathbf{v}-\mathbf{w}\| + \|\mathbf{w}-\mathbf{u}\|)\cdot\|(\mathbf{v}-\mathbf{u})^{¢}\cdot(\mathbf{w}-\mathbf{u})\|^2 ,$$

which vanishes just when the given triangle is *degenerate* (has area zero), as might reasonably be expected. But this expectation is misleading because the linear system is always *consistent*; the degenerate case is the singularity of the linear system but not of the desired center $\mathbf{c}$. In fact $\mathbf{c}$ is everywhere a continuous function of the vertices, lying always in their convex hull even if they are collinear, in which degenerate case $\mathbf{c}$ falls upon whichever of the three vertices lies between the other two. Explicitly, except for a *Removable Singularity* when $\mathbf{c} = \mathbf{u} = \mathbf{v} = \mathbf{w}$,

$$\mathbf{c} = (\mathbf{u}\cdot\|\mathbf{v}-\mathbf{w}\| + \mathbf{v}\cdot\|\mathbf{w}-\mathbf{u}\| + \mathbf{w}\cdot\|\mathbf{u}-\mathbf{v}\|)/(\|\mathbf{v}-\mathbf{w}\| + \|\mathbf{w}-\mathbf{u}\| + \|\mathbf{u}-\mathbf{v}\|) .$$

In this unobvious formula, roundoff with threshold ε generates less uncertainty in $\mathbf{c}$ than about

$$6\cdot\varepsilon\cdot(|\mathbf{u}|\cdot\|\mathbf{v}-\mathbf{w}\| + |\mathbf{v}|\cdot\|\mathbf{w}-\mathbf{u}\| + |\mathbf{w}|\cdot\|\mathbf{u}-\mathbf{v}\|)/(\|\mathbf{v}-\mathbf{w}\| + \|\mathbf{w}-\mathbf{u}\| + \|\mathbf{u}-\mathbf{v}\|) \text{ elementwise.}$$

It is negligible unless $\|\mathbf{c}\|$ is orders of magnitude smaller than at least two of $\|\mathbf{u}\|$, $\|\mathbf{v}\|$ and $\|\mathbf{w}\|$, whereas the formula $\mathbf{c} = M^{-1}\cdot\mathbf{m}$ further amplifies roundoff by a factor that can be roughly as big as (triangle's length)/(triangle's width) when the triangle is narrow.

The incenter of a triangle has been computed from (the equations of) its edges during navigation aided by signals from beacons, and during attempts to locate the source of radio transmissions from hostile military units and spies. These edges were parts of three lines that would have been concurrent but for small errors in observations and in instruments. Given those lines, the triangle's incenter was deemed the "best" approximation available to the single intended point of concurrence. This incenter is equidistant from the three lines, but so are three other points outside the triangle. They were excluded by computing the desired incenter not directly from the coefficients of the lines' equations but after computing the triangle's vertices. Nowadays a "best" approximation to one intersection point of (perhaps more than three) lines can be computed directly from their coefficients as the solution of a weighted least-squares problem. The weights take signal strength, interfering noise and other uncertainties into account.

Someone ignorant of the unobvious formula who computes $\mathbf{c} = M^{-1}\cdot\mathbf{m}$ will wrongly blame its far worse inaccuracy due to roundoff upon the data's huge $\kappa(M)$ when the triangle is narrow. In fact much of the lost accuracy gets lost during the computation of the rows of $M = [\mathbf{p}, \mathbf{b}, \mathbf{q}]^T$.

Do not confuse ignorance *with* stupidity:

"With Stupidity even the Gods struggle in vain." J.C.F. von Schiller, 1759 - 1805

A programmer who derives a linear system like $M\cdot\mathbf{c} = \mathbf{m}$ to solve for $\mathbf{c}$ is far from stupid; he is unlucky or too impatient or too beset by deadlines to pore through enough old texts on vectors and geometry. How long would you take to find the foregoing accurate but unobvious formula for $\mathbf{c}$?

Who pays the penalty for a programmer's ignorance? Him?
More likely his program's users, perhaps unwitting.

Suppose the user has a program for $\mathbf{c}$ = incenter($\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$) that must be treated as a *Black Box* because it comes precompiled as a library module without readable source-text, or because the source-text is legible but inscrutable (as are too many programs). How may the user of this black box test its accuracy for his data without enough access to the box's design to tamper with it nor to perform an error-analysis? Two ways come to mind:

The first way compares the results $\mathbf{c}$ obtained from incenter($\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$) when its arguments are permuted. In the absence of roundoff $\mathbf{c}$ would not change. Symmetries in every program for incenter($\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$) almost surely keep roundoff from producing more than three different results $\mathbf{c}$

from the six permutations of the vertices. A user lacking foreknowledge may have to try as many as five of them. If any two computed results differ substantially, all are suspect. Alas, substantial agreement among all six results **c** when all are substantially wrong is possible though unlikely.

The second way exploits a little-known and linguistically ill-supported capability mandated by IEEE Standard 754 for Binary Floating-Point Arithmetic: *Directed Roundings*. The default direction of rounding is *To Nearest*, which restricts each rounding error to at most half a unit in the last binary digit retained. The three directed roundings are *To* 0 , *To* +∞ and *To* –∞ ; each such *Rounding Mode* biases all rounding errors in the specified direction while keeping each error smaller than one unit in the last digit retained. Program incenter(**u**, **v**, **w**) should inherit by default whatever rounding mode was in force before incenter was invoked, so its user should be able to obtain four different values of **c** = incenter(**u**, **v**, **w**) , one for each rounding mode. If any one differs substantially from the others, all are suspect. Alas, substantial agreement among all four results **c** when all are substantially wrong is possible though unlikely. And few compilers and fewer debuggers support any way to redirect rounding for the effect desired here. My web page's `<…/Mindless.pdf>` contrasts redirected rounding with other techniques purporting to help debug malignant roundoff.

Versions 6 and later of MATLAB offer ways to redirect roundoff and constrain precision adequate to explore programs like incenter(**u**, **v**, **w**) though their full scopes are difficult to ascertain; see `<…/MxMulEps.pdf>` on my web page.

The time has come for a numerical example. The data **u**, **v**, **w** and the result **c** as well as all intermediate variables are 4-byte-wide `floats` with 24 sig. bits of precision, the same as the arithmetic's, except that one 8-byte `double` value of **c** used for reference is computed from the unobvious but most accurate formula using 53 sig. bit `double` arithmetic. Here is the data:

$$\mathbf{u} := [\ 255429.53125,\ \ -139725.125,\ \ 140508.53125\ ]^T,$$
$$\mathbf{v} := [\ 10487005.,\ \ 8066347.,\ \ -11042884.\ ]^T,$$
$$\mathbf{w} := [\ -5243521.,\ \ -4033150.,\ \ 5521499.\ ]^T\ .$$

These are the vertices of a narrow triangle 80 times as long as wide. Its accurate incenter is

$$\mathbf{c} = [\ 128615.61552\ldots,\ \ -69127.510715\ldots,\ \ 69282.163604\ldots\ ]^T$$
$$\approx [\ 128615.61{\scriptstyle 71785},\quad -69127.50{\scriptstyle 78125},\quad 69282.164{\scriptstyle 0625}\ ]^T \quad \text{rounded to 24 sig. bits.}$$

The naïve formula $\mathbf{c} = M^{-1}\cdot\mathbf{m}$ was tested three ways: One was Gaussian elimination with pivotal row-exchanges. The other two were formulas mentioned for §9's problem #0 , namely $\mathbf{c} = (\ \mathbf{b}^{¢}\cdot\mathbf{q}\cdot\pi + \mathbf{q}^{¢}\cdot\mathbf{p}\cdot ß + \mathbf{p}^{¢}\cdot\mathbf{b}\cdot\theta\ )/(\mathbf{p}^T\cdot\mathbf{b}^{¢}\cdot\mathbf{q})$ and $\mathbf{c} = \big(\ (\mathbf{b}^{¢}\cdot\mathbf{q})\cdot\pi + \mathbf{p}^{¢}\cdot(\mathbf{b}\cdot\theta - \mathbf{q}\cdot ß)\ \big)/\big(\mathbf{p}^T\cdot(\mathbf{b}^{¢}\cdot\mathbf{q})\big)$ with $\mathbf{b}^{¢}\cdot\mathbf{q}$ reused. However, for the data given here, all three ways produced results so much closer to each other than to the correct reference value **c** that they are not distinguished hereunder. The three incenters **c** computed naively carrying 24 sig. bits were in error by about 400.00 in each element. This is over 100 times as big as the error of about 0.30 when **c** is computed from the unobvious formula; its error is expected because ||**v**|| and ||**w**|| are over 50 times as big as ||**c**|| . How does two extra sig. dec. lost to naïveté accord with the condition number $\kappa(M) \approx 100.66$?

The accuracy lost to naïveté depends upon the order in which the vertices are presented. When [**u**, **v**, **w**] is replaced by [**v**, **w**, **u**] the error in **c** drops from 400.00 to near 18.00 ; for [**w**, **u**, **v**] it drops to about 5.00 . $\kappa(M) \approx 35.$ in both latter orderings. These three errors are of the worst kind, too small to be obvious yet too big to ignore. They are not correlated closely with $\kappa(M)$.

Why do the tested evaluations of the naïve formula $\mathbf{c} = \mathrm{M}^{-1}\cdot\mathbf{m}$ yield results so similar and yet so much worse than $\kappa(\mathrm{M})$ might predict? Evidently, for the data given here, roundoff damages the computed elements of array $[\mathrm{M}, \mathbf{m}]$ more than it damages the subsequently computed $\mathrm{M}^{-1}\cdot\mathbf{m}$. Where is the singularity that imperils $[\mathrm{M}, \mathbf{m}]$? If the given triangle were degenerate its collinear vertices would cause one, two or three of the rows of $[\mathrm{M}, \mathbf{m}]$ to vanish, depending upon the vertices' order. This singularity hiding in plain sight wreaks the worst upon naively computed $\mathbf{c}$.

When each of the foregoing computations is rerun three times with redirected roundings, in every case the four computed results $\mathbf{c}$ vary among themselves at least roughly as much as their errors, revealing each computation's (hyper)sensitivity to roundoff. The same kind of revelation occurs when $\mathbf{z}$ in §10 is recomputed with redirected roundings. These are valuable diagnostic tools.

The examples above foreshadow much of what happens in general. First, some singularities are unobvious. Second, among singularities inherent in a computational problem whose solution we seek, some may be malignant when approached by slightly uncertain data, and others benign (removable). Third, roundoff's effects may bloat intolerably when otherwise innocuous data approach spurious singularities concomitant in an ill-chosen numerical algorithm but not inherent in the problem to be solved. Spurious singularities can lurk in "canned" software obtained from the internet or embedded in respectable computational environments like MATLAB. Perplexing examples are posted in my "Do MATLAB's `lu(…)`, `inv(…)`, `/` and `\` have *Failure Modes* ?" `<…/Math128/FailMode.pdf>`. Most such software cannot yet defend its users against perplexity.

How can the producer of numerical software avoid spurious singularities? The simplest way, though neither foolproof nor always economical, is to carry extravagantly more precision in all intermediate variables and arithmetic than the precision of the data and the accuracy desired for the result. The greater the precision carried, the smaller the incidence of embarrassment due to roundoff; usually every extra decimal digit carried reduces that incidence by a factor of one tenth.

The best way to extirpate spurious singularities is to locate them by performing an error-analysis and then suitably revise the program's numerical algorithm. This *Counsel of Perfection*, "more honour'd in the breach", too often costs rather more human effort than the computation is worth.

Iterative Refinement

A good way to cope may be available when the desired result satisfies an equation, say $\mathrm{M}\cdot\mathbf{c} = \mathbf{m}$, that the computation solves. For any approximate solution $\mathbf{x}$, no matter how it was obtained, the *Residual* $\mathbf{r} := \mathbf{m} - \mathrm{M}\cdot\mathbf{x}$ is a computable indication of how far $\mathbf{x}$ dissatisfies that equation. Now repeat the process that produced $\mathbf{x}$ to solve $\mathrm{M}\cdot\Delta\mathbf{x} = \mathbf{r}$ for $\Delta\mathbf{x}$ approximately; often the repeated process reuses a large fraction of the intermediate results generated when $\mathbf{x}$ was computed, so the cost of $\Delta\mathbf{x}$ is relatively small. Then $\mathbf{x} + \Delta\mathbf{x}$ is the *Refined* approximation to $\mathbf{c}$; and the whole process is called "Iterative Refinement" if repeated. It is explored exhaustively in "Error Bounds from Extra-Precise Iterative Refinement" by J.W. Demmel *et al*., pp. 325 - 351 of *ACM Transactions on Mathematical Software (TOMS)* 32 (2006). Here is a rough outline of the facts:

- •i If $\mathbf{r}$ is computed with the same arithmetic precision as was carried to compute $\mathbf{x}$ then $\mathbf{x} + \Delta\mathbf{x}$ will very likely have a residual smaller than $\mathbf{r}$ and possibly an error smaller than $\mathbf{x}$'s. But as $\kappa(\mathrm{M})$ grows so does the likelihood that $\mathbf{x} + \Delta\mathbf{x}$ be less accurate than $\mathbf{x}$.

•ii If $\mathbf{r}$ is computed with arithmetic sufficiently more precise than was used to compute $\mathbf{x}$ then $\mathbf{x}+\Delta\mathbf{x}$ will almost surely have a residual smaller than $\mathbf{r}$ and an error smaller than $\mathbf{x}$'s. And if "sufficiently more precise than" is "at least double the precision that" then repeated refinement can produce a refined solution $\mathbf{x}$ correct in all but its last digit or two unless M differs from singular by less than several units in its last digits at the precision carried to compute $\mathbf{x}$ initially.

•iii Computing $\mathbf{r}$ extra-precisely usually costs a small fraction of the time spent to compute $\mathbf{x}$ initially, even if extra-precise arithmetic has to be simulated slowly in software. And then the refined solution's accuracy is almost always improved, sometimes spectacularly, without anyone having to know whether nor why the initial $\mathbf{x}$ was inaccurate.

At least one refinement should *always* be carried out *by default* whenever the residual can be computed extra-precisely for a tolerable price. But it could not improve $\mathbf{c}$ = incenter($\mathbf{u}$,$\mathbf{v}$,$\mathbf{w}$) for data like the example's above because M and $\mathbf{m}$ were already damaged too much by roundoff.

For similar reasons, iterative refinement cannot help to solve a similar geometrical problem:

#**10.** Find *Incenter* $\mathbf{c}$ of the biggest sphere inscribed in a tetrahedron whose vertices $\mathbf{o}$, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are given. $\mathbf{c}$ must be equidistant from the tetrahedron's four face-planes, and on the same side of each face-plane as its opposite vertex. This requirement translates via problem #3 into a linear system that $\mathbf{c}$ must satisfy: $M\cdot\mathbf{c} = \mathbf{m}$ in which, regardless of $\mathrm{sign}(\,\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w}\,)$,

$$\mathbf{p} := (\mathbf{v}–\mathbf{u})^{¢}(\mathbf{w}–\mathbf{u})/||(\mathbf{v}–\mathbf{u})^{¢}(\mathbf{w}–\mathbf{u})|| ,\quad M := [\mathbf{v}^{¢}\mathbf{w}/||\mathbf{v}^{¢}\mathbf{w}|| + \mathbf{p},\ \ \mathbf{w}^{¢}\mathbf{u}/||\mathbf{w}^{¢}\mathbf{u}|| + \mathbf{p},\ \ \mathbf{u}^{¢}\mathbf{v}/||\mathbf{u}^{¢}\mathbf{v}|| + \mathbf{p}]^T$$

and $\mathbf{m} := [\mathbf{u}, \mathbf{v}, \mathbf{w}]^T\cdot\mathbf{p} = \mathbf{u}^T\cdot\mathbf{p}\cdot[1, 1, 1]^T$. The determinant of the matrix M turns out to be

$$\det(M) = (\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w})^2\cdot(\,||\mathbf{v}^{¢}\mathbf{w}||+||\mathbf{w}^{¢}\mathbf{u}||+||\mathbf{u}^{¢}\mathbf{v}||+||(\mathbf{v}–\mathbf{u})^{¢}(\mathbf{w}–\mathbf{u})||\,)/(\,||\mathbf{v}^{¢}\mathbf{w}||\cdot||\mathbf{w}^{¢}\mathbf{u}||\cdot||\mathbf{u}^{¢}\mathbf{v}||\cdot||(\mathbf{v}–\mathbf{u})^{¢}(\mathbf{w}–\mathbf{u})||\,)$$

which vanishes just when the tetrahedron is degenerate (has no volume). Consequently we expect roundoff to corrupt the equation's solution when the tetrahedron is nearly flat like an axe-blade, and to corrupt the equation's construction when the tetrahedron is narrow like a needle.

Fortunately the linear equation " $M\cdot\mathbf{c} = \mathbf{m}$ " need never be constructed; its solution is simply

$$\mathbf{c} = (\,\mathbf{u}\cdot||\mathbf{v}^{¢}\mathbf{w}|| + \mathbf{v}\cdot||\mathbf{w}^{¢}\mathbf{u}|| + \mathbf{w}\cdot||\mathbf{u}^{¢}\mathbf{v}||\,)/(\,||\mathbf{v}^{¢}\mathbf{w}|| + ||\mathbf{w}^{¢}\mathbf{u}|| + ||\mathbf{u}^{¢}\mathbf{v}|| + ||(\mathbf{v}–\mathbf{u})^{¢}(\mathbf{w}–\mathbf{u})||\,)$$

and is a positively weighted average of the vertices. The weight of each vertex is proportional to the area of its opposite face. Although roundoff corrupts the weights if the tetrahedron is too needle-shaped, its convex hull always very nearly contains the computed $\mathbf{c}$.

Numerical Example: $[\mathbf{u}, \mathbf{v}, \mathbf{w}] := \begin{bmatrix} 4182 & 5168 & 4791 \\ 5168 & 6388 & 5922 \\ 4791 & 5922 & 5490 \end{bmatrix}$ has $\mathbf{u}^T\mathbf{v}^{¢}\mathbf{w} = 36$, so $\det(M) \neq 0$. However, replacing u_1 by 4181 makes $6\mathbf{u} = 3\mathbf{v} + 2\mathbf{w}$, collapsing the tetrahedron. Before replacement, $\mathbf{c} \approx [4789.4057,\ 5920.0275,\ 5488.1688]^T$; afterwards $\mathbf{c} = [4181.,\ 5168.,\ 4791.]^T$ = replaced $\mathbf{u}$. All the data are fairly small integers, so `float` computation rounded to 24 sig.bits produced a fair $\mathbf{c} \approx [\,4789.4204,\ 5920.0454,\ 5488.1851\,]^T$ from the simple formula before replacement; however " $M\cdot¢ = \mathbf{m}$ " produced $¢ \approx [\,342.52,\ 423.32,\ 392.48\,]^T$, far from $\mathbf{c}$ and at least 0.03 outside the tetrahedron. This ¢ errs in the worst way: too far to ignore but not far enough to be obvious. Arithmetic rounded to 53 sig.bits yielded $\mathbf{c}$ well from both formulas.