

Matching Paired Sets of Space and Orientation Data

Andrew J. Hanson

hansona@indiana.edu

School of Informatics and Computing

Indiana University

WTC 2018

Champaign-Urbana, IL

16–19 October 2018

Matching Paired Sets of Space and Orientation Data

Why is this interesting ?? **Because -**

(a) It requires *quaternions* (I wrote a 498 page book [*Visualizing Quaternions*](#)).

(b) For half a century, **all known solution methods** for the 3D and 4D matching problems have involved numerical approximations.

Exact algebraic solutions in 3D and 4D were known to involve 4th order polynomials, and, so far as I know, were considered to be **intractable**.

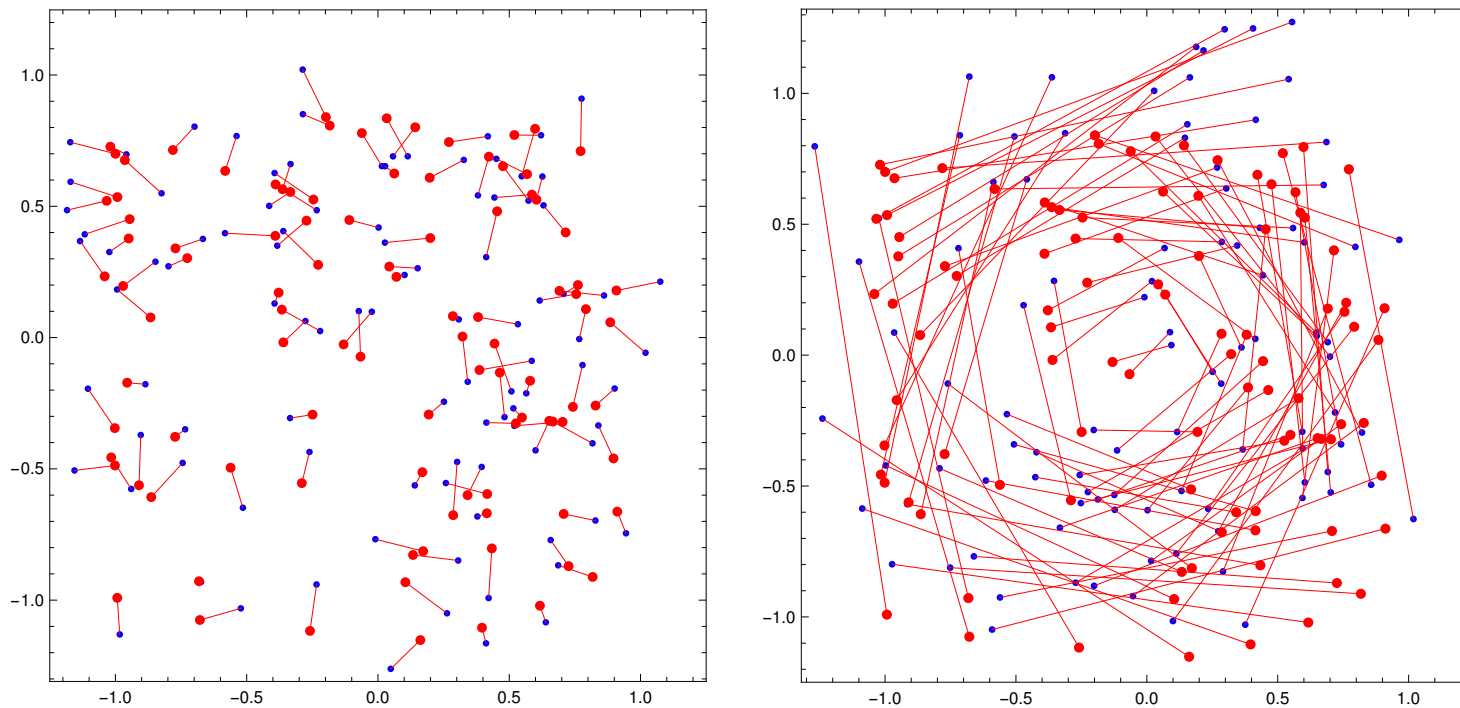
I have used quaternions and Mathematica to produce the first exact algebraic solutions of the full 3D and 4D eigenspectra ever written down, and extended the method to orientation frames.

The RMSD Matching Problem

- Our input data are the reference data set $\{y_k\}$ and one or more item-by-item matched test data sets $\{x_k\}$.
- The task is to **PICK ONE ROTATION MATRIX** R that minimizes the *Root-Mean-Square Deviation* measure

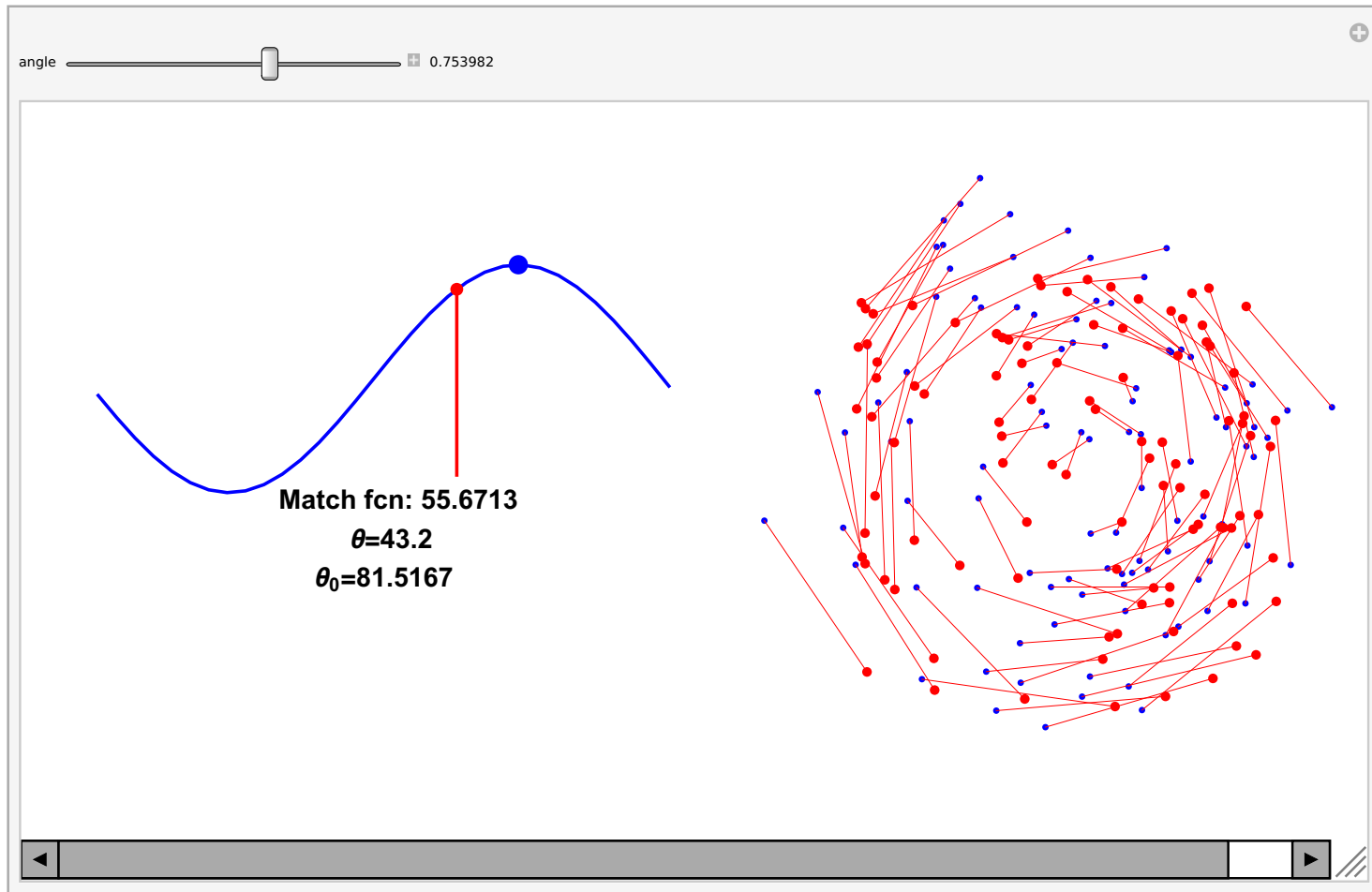
$$\text{RMSD}^2 \rightarrow S^2 = \sum_{k=1}^N \|R \cdot x_k - y_k\|^2$$

DEMO: Collections of $\|x_k - y_k\|^2$



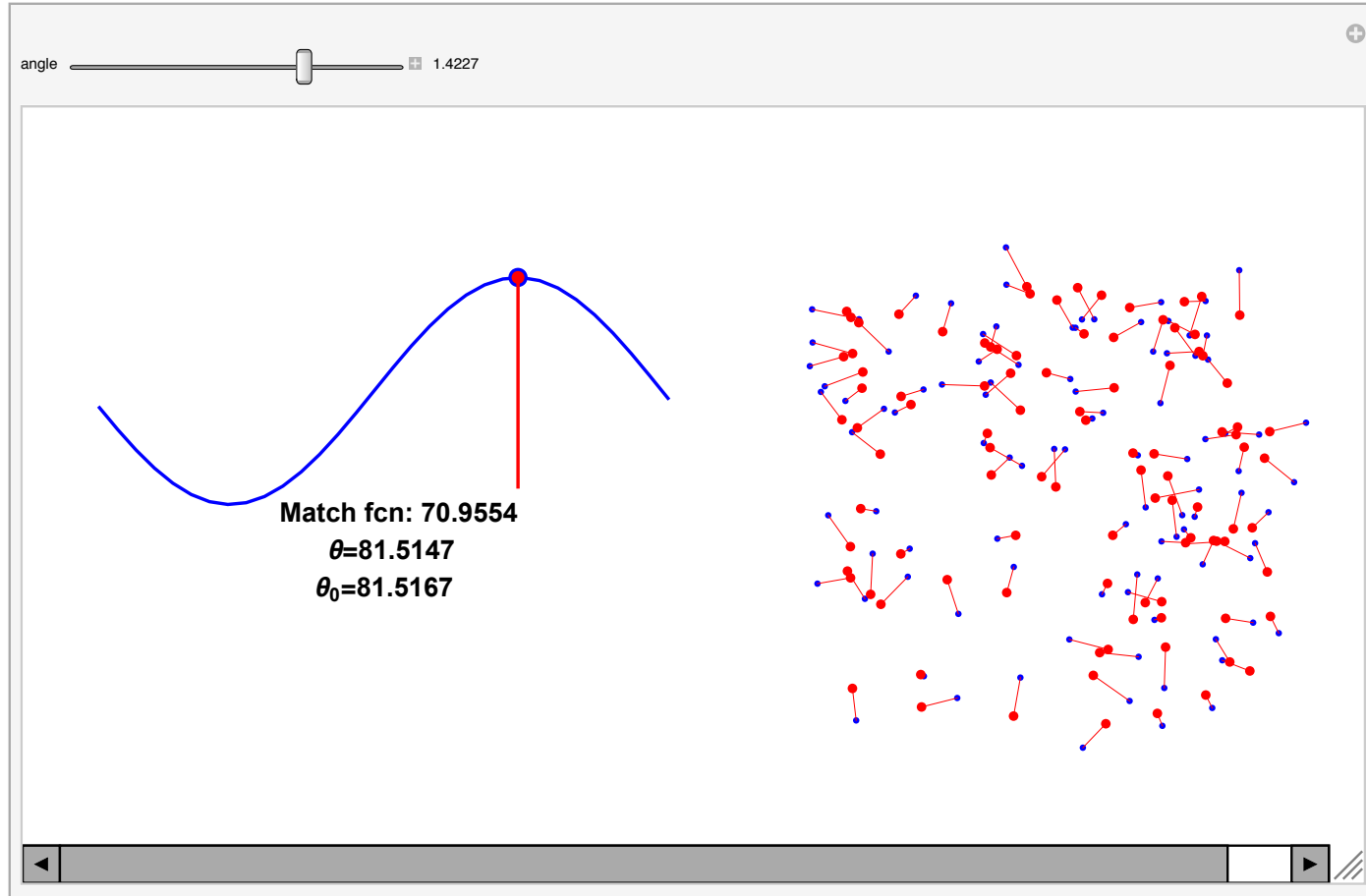
Left: *Reference* data $\{y_k\}$ are in **red**, noisy *test* data $\{x_k\}$ in **blue**, *before* global rotation. **Right:** What happens to the incremental distances after a **global rotation** of the noisy blue test data around the mutual center of mass.

Solving – how the optimization looks!



Partway through the rotation from mismatched state, still lots of space between the blue and red points.

At the Solution!



The matrix R_0 rotates the blue points to optimal alignment with red points.

Refining the Optimization Problem

Method: *Minimizing* the RMSD is equivalent (dropping constants, for any dimension D) to *maximizing* the simpler cross-term measure Δ :

$$\Delta_D = \sum_{k=1}^N (R_D \cdot x_k) \cdot y_k = \sum_{a=1, b=1}^D R_D^{ba} E_{ab} ,$$

where the “data contents” all reduce to the Euclidean component-wise averages,

$$E_{ab} = \sum_{k=1}^N x_k^a y_k^b .$$

The raw form of the RMSD optimization task is thus to find the rotation matrix R that maximizes Δ .

Quaternion Version of 3D Spatial RMSD Problem

- It was discovered independently, in at least three different literatures, and at least four different non-co-citing published papers, that the 3D problem reduces to a 4D **quaternion eigenvalue problem**, and that finding that *numerical solution* is relatively easy.
- **We now show how to exploit quaternions in Mathematica to algebraically solve this 4D problem, in principle a very difficult quartic algebraic equation.**

First, We Need Some Quaternion Properties

- **Quaternions are Unit four-vectors.**

Take $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})$ to obey the constraint $q \cdot q = 1$.

- **They have a Multiplication Rule.** The quaternion product of q and p is

$$q * p = (q_0 p_0 - \vec{q} \cdot \vec{p}, q_0 \vec{p} + p_0 \vec{q} + \vec{q} \times \vec{p}),$$

or, alternatively,

$$\begin{bmatrix} [q * p]_0 \\ [q * p]_1 \\ [q * p]_2 \\ [q * p]_3 \end{bmatrix} = \begin{bmatrix} q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3 \\ q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2 \\ q_0 p_2 + q_2 p_0 + q_3 p_1 - q_1 p_3 \\ q_0 p_3 + q_3 p_0 + q_1 p_2 - q_2 p_1 \end{bmatrix}$$

3D Rotations Have a Quaternion Form

Any 3D rotation matrix has a quaternion quadratic form

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} .$$

$R(q)$ converts $\Delta = \text{tr}(R \cdot E) \rightarrow \Delta(q)$ having this matrix form

$$\Delta(q) = \sum_{i,j=0}^3 q_i \mathbf{M}_{ij} q_j = q \cdot \mathbf{M} \cdot q ,$$

where \mathbf{M} is a novel linear combination of the E_{ab} defined as
...

...the 3D RMSD Profile Matrix $M(E)$

$$\begin{bmatrix} E_{xx} + E_{yy} + E_{zz} & E_{yz} - E_{zy} & E_{zx} - E_{xz} & E_{xy} - E_{yx} \\ E_{yz} - E_{zy} & E_{xx} - E_{yy} - E_{zz} & E_{xy} + E_{yx} & E_{zx} + E_{xz} \\ E_{zx} - E_{xz} & E_{xy} + E_{yx} & -E_{xx} + E_{yy} - E_{zz} & E_{yz} + E_{zy} \\ E_{xy} - E_{yx} & E_{zx} + E_{xz} & E_{yz} + E_{zy} & -E_{xx} - E_{yy} + E_{zz} \end{bmatrix}$$

Observe that this 4D matrix is **traceless and symmetric**, which simplifies some aspects of the 3D problem.

(It turns out we can also solve the **4D** Euclidean RMSD problem analytically, where the matrix M is completely general, with no constraints whatever.)

Eigenvalues of the Profile Matrix \mathbf{M}

The maximum value of

$$\Delta = \text{tr} (R \cdot E) = q \cdot \mathbf{M} \cdot q$$

is given by the *maximal eigenvalue* ϵ_{opt} of \mathbf{M} (corresponding to the *quaternion eigenvector* q_{opt}).

Thus we must solve $\det[\mathbf{M} - eI_4] = 0$, where e denotes a generic eigenvalue and I_4 is the 4D identity matrix. We can write this in two ways:

$$e^4 + e^3 p_1 + e^2 p_2 + e p_3 + p_4 = 0 ,$$

$$(e - \epsilon_1)(e - \epsilon_2)(e - \epsilon_3)(e - \epsilon_4) = 0 .$$

where ϵ_i are the eigenvalues, $p_1 = 0$ is the trace, $p_4(E)$ is the determinant, and $p_2(E)$ and $p_3(E)$ are polynomials in the data averages E_{ab} .

*Eigenvalues of the Profile Matrix **M***

Observe the following:

- The data coefficients $\{p_1(E), p_2(E), p_3(E), p_4(E)\}$ are known: *they are just numbers*.
- The equations above give the *unknown eigenvalues* ϵ_i in terms of the *known data coefficients*:

$$p_1(E) = -\epsilon_1 - \epsilon_2 - \epsilon_3 - \epsilon_4$$

$$p_2(E) = \epsilon_1\epsilon_2 + \epsilon_1\epsilon_3 + \epsilon_2\epsilon_3 + \epsilon_1\epsilon_4 + \epsilon_2\epsilon_4 + \epsilon_3\epsilon_4$$

$$p_3(E) = -\epsilon_1\epsilon_2\epsilon_3 - \epsilon_1\epsilon_2\epsilon_4 - \epsilon_1\epsilon_3\epsilon_4 - \epsilon_2\epsilon_3\epsilon_4$$

$$p_4(E) = \epsilon_1\epsilon_2\epsilon_3\epsilon_4$$

- **Therefore our task is to invert this equation for**
 $\epsilon_i(p_1, p_2, p_3, p_4)$.

Approach to Exact Quaternion Soln

Studying the apparently intractable algebraic expressions for \mathbf{M} 's symbolic eigensystem in Mathematica, I eventually discovered that it was useful to *change the variables* in the above (traceless) equation to express the 4 eigenvalues of \mathbf{M} in the following form:

$$\begin{aligned}\epsilon_1 &= +\sqrt{X} + \sqrt{Y} + \sqrt{Z} \\ \epsilon_2 &= +\sqrt{X} - \sqrt{Y} - \sqrt{Z} \\ \epsilon_3 &= -\sqrt{X} + \sqrt{Y} - \sqrt{Z} \\ \epsilon_4 &= -\sqrt{X} - \sqrt{Y} + \sqrt{Z}\end{aligned}$$

This is a peculiarly advantageous form of the 4D symmetric, traceless matrix \mathbf{M} that is adapted to machine algebra.

Approaching the Quaternion RMSD Solution

Now the equations for the eigenvalues reduce to the following form, where of course *knowing* $(X(p), Y(p), Z(p))$ means *knowing the eigenvalues* ϵ_i :

$$\text{Eqns} = \left\{ \begin{array}{lcl} p_4(E) & = & X^2 - 2XY + Y^2 - 2XZ - 2YZ + Z^2 \\ [p_3(E)]^2 & = & 64XYZ \\ p_2(E) & = & -2(X + Y + Z) \end{array} \right\}$$

Then `Solve[Eqns, {X, Y, Z}]` runs for a minute, giving 6 sets of solutions with this `ByteCount[]` list:

`{{4584, 19544, 21552}, {4584, 19448, 21552}, {5224, 22232, 28640},
{5224, 22400, 28736}, {5224, 22256, 28632}, {5224, 22392, 28728}}`

Exact Quaternion Soln to 3D RMSD, contd

Each of the *short* expressions looks like this, e.g., for the first $X(p)$, while the other expressions are pages of algebra that do not respond to `Simplify[]` :

$$\begin{aligned}
 & -\frac{p_2}{6} \\
 & - \frac{\sqrt[3]{-p_2^3 + 36p_4p_2 + \frac{3}{2} \left(\sqrt{-48p_4p_2^4 + 12p_3^2p_2^3 + 384p_4^2p_2^2 - 432p_3^2p_4p_2 + 81p_3^4 - 768p_4^3 - 9p_3^2} \right)}}{12} \\
 & - \frac{p_2^2 + 12p_4}{12 \sqrt[3]{-p_3^3 + 36p_2p_4 + \frac{3}{2} \left(\sqrt{-48p_4p_2^4 + 12p_3^2p_2^3 + 384p_4^2p_2^2 - 432p_3^2p_4p_2 + 81p_3^4 - 768p_4^3 - 9p_3^2} \right)}}
 \end{aligned}$$

Note: Why you need analytic solns:

Another basic problem is that you can't include any parameters: if m_0 and m_1 are 4×4 profile matrices, you get abstract roots:

$$\text{Eigenvalues}[t_0 m_0 + t_1 m_1] \implies$$

```
{Root[-275457. t0^4 - 1.24918*10^6 t0^3 t1 - 1.98623*10^6 t0^2 t1^2 -
  1.33246*10^6 t0 t1^3 - 319975. t1^4 + (-47234.5 t0^3 - 149423. t0^2
  153487. t0 t1^2 - 51093.6 t1^3) #1 + (-2131.46 t0^2 -
  4172.69 t0 t1 - 2147.62 t1^2) #1^2 - 1.77636*10^-15 t0 #1^3 +
  #1^4 \&, 1],
Root[-275457. t0^4 - 1.24918*10^6 t0^3 t1 - 1.98623*10^6 t0^2 t1^2 -
  1.33246*10^6 t0 t1^3 ... #1^4 \&, 2],
Root[ ... ... #1^4 \&, 3],
Root[ ... ... #1^4 \&, 4] }
```

Exact Quaternion Soln to 3D RMSD

So what happens is that 2/3 of every solution is pages of impenetrable symbols, 20Kb each, but the **FIRST** one is only 5Kb, as on the previous slide.

If you plug in random numbers, you find that each of the short ones matches one of the long ones, so if you assemble them like a puzzle, you have a completely usable algebraic solution.

THAT solution in turn breaks up magically into a sum of **cube roots of unity**, hugely simplifying our task.

The Solution

By comparing the numerical values to the assortment of algebraic expressions using the tricks above, I was ultimately able to reduce the algebra for the **all eigenvalues of M** to the following form for $(X(p), Y(p), Z(p)) = F_{(x,y,z)}$:

$$F_f(p_2, p_3, p_4) = \frac{1}{6} (r(p_2, p_3, p_4) \cos_f(p_2, p_3, p_4) - p_2)$$

where we define

$$\begin{aligned}\cos_x(p_2, p_3, p_4) &= \cos\left(\frac{\arg(a+ib)}{3}\right) \\ \cos_y(p_2, p_3, p_4) &= \cos\left(\frac{\arg(a+ib)}{3} - \frac{2\pi}{3}\right) \\ \cos_z(p_2, p_3, p_4) &= \cos\left(\frac{\arg(a+ib)}{3} + \frac{2\pi}{3}\right)\end{aligned}$$

Exact Quaternion Soln to 3D RMSD

In this expression $\arg(u+iv) = \text{atan2}(v, u) = \text{ArcTan}(u, v)$,
 $F_f(p)$ corresponds to $X(p)$, $Y(p)$, and $Z(p)$ for $f = \{x, y, z\}$,
and the utility functions reduce to

$$a(p_2, p_3, p_4) = p_2^3 + \frac{1}{2} (27p_3^2 - 72p_2p_4)$$

$$n(p_2, p_3, p_4) = p_2^2 + 12p_4$$

$$b(p_2, p_3, p_4) = \sqrt{n^3 - a^2}$$

$$r(p_2, p_3, p_4) = \sqrt[6]{a^2 + b^2} = \sqrt{n} .$$

Exact Quaternion Soln to 3D RMSD

The (all real) 3D eigenvalues in order of descending magnitude are now written in terms of the three phases of $F_f(p)$ for $f = \{x, y, z\}$ corresponding to $\{X, Y, Z\}$:

$$\epsilon_1 = +\sqrt{X} + \sqrt{Y} + \sqrt{Z}$$

$$\epsilon_2 = +\sqrt{X} - \sqrt{Y} - \sqrt{Z}$$

$$\epsilon_3 = -\sqrt{X} + \sqrt{Y} - \sqrt{Z}$$

$$\epsilon_4 = -\sqrt{X} - \sqrt{Y} + \sqrt{Z} .$$

Eigenvectors for 3D RMSD

The eigenvector formulas corresponding to ϵ_k can be generically computed by solving the bottom three rows of

$$[M(E) \cdot \mathbf{v} - e\mathbf{v}] = 0$$

for the elements of $\mathbf{v} = (1, v_1, v_2, v_3)$ as a function of some eigenvalue e , so we just use the exact algebraic solution for $e_{\text{opt}} = \epsilon_1$ and we are done!

Caveat - rearrange \mathbf{v} if any element of M is already diagonal!

DEMO of running the solutions if time permits...

Summary: Quaternion RMSD Solns

- **This solution is unknown to the best of our knowledge.** All previous literature solves *numerically* using, e.g., Newton's method, for the maximal quaternion eigenvalue, giving the quaternion solution for the optimal aligning rotation via the quadratic formula for $R(q)$. In fact, we get **all four** eigenvalues, not just the maximal one.
- **Using similar methods, we have solved the *3D frame alignment problem* and the 6 DOF *combined problem* in closed form as well.**
- Using still further nontrivial quaternion-driven algebraic manipulations, we have also solved the *4D spatial data problem*, the *4D orientation-frame problem*, and the corresponding *combined 10 DOF problem*.

(End) Blank Slide

- .

- .

Outline: Generalizing to 3D Quaternion Frames

The *spatial* matching problem can be generalized to a *frame orientation* problem.

Why is this interesting? Example: Proteins contain hundreds of amino acid residues, and the unique orientations of each residue can be encoded efficiently as **Quaternion Frames**.

We next briefly outline the closed form **quaternion frame matching** aka **QRMSD** problem.

Quaternions as 3D Frames

If we take a quaternion q and write the 3D rotation matrix as

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

the *columns* are the axes of a 3D orthonormal frame triad.

Taking a set of almost-aligned quaternion-valued test frames $\{p_k\}$ and reference frames $\{r_k\}$, we can write the chord-distance measure as

$$\Delta_f = \sum_{k=1}^N (q * p_k) \cdot r_k ,$$

where the quaternion product of q and p is

$$q * p = (q_0p_0 - \vec{q} \cdot \vec{p}, q_0\vec{p} + p_0\vec{q} + \vec{q} \times \vec{p}),$$

... Solution for 3D Frames

Rearrange in Quaternion Form:

We see the quaternion analog of the *profile matrix* emerge as $E_{ab} = \sum_{k=1}^N p_k^{(a)} r_k^{(b)}$, and

$$\Delta_f = q \cdot V$$

where V is the column vector

$$V = \begin{bmatrix} E_{00} + E_{11} + E_{22} + E_{33} \\ -E_{01} + E_{10} - E_{23} + E_{32} \\ -E_{02} + E_{20} - E_{31} + E_{13} \\ -E_{03} + E_{30} - E_{12} + E_{21} \end{bmatrix}$$

exactly analogous to the spatial *profile matrix*.

... Exact Solution for 3D Frames

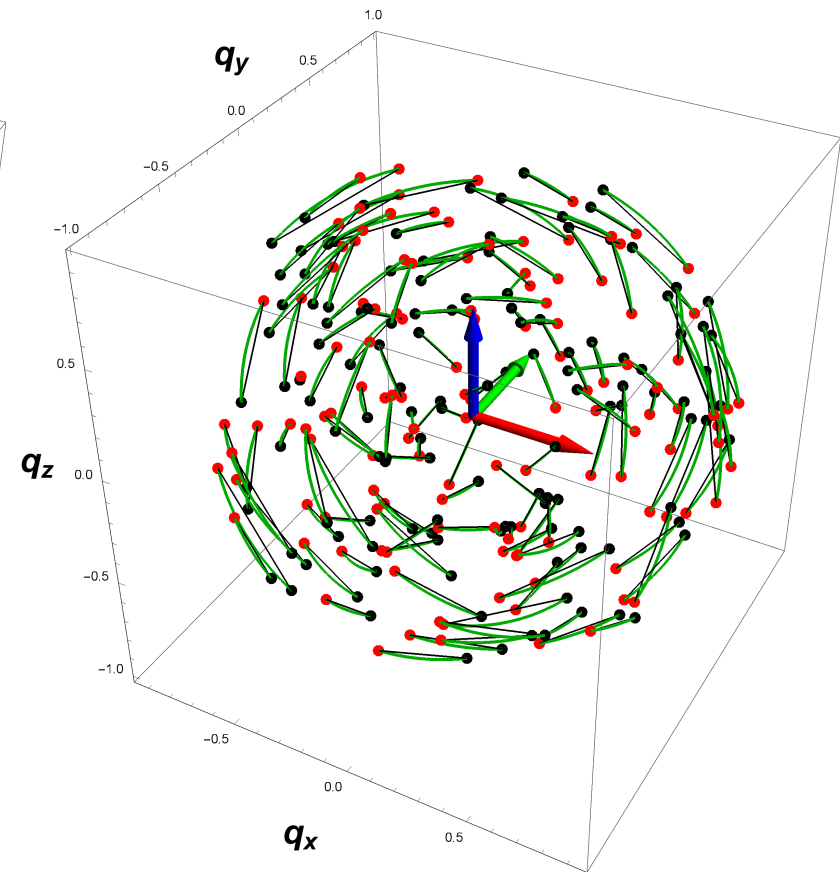
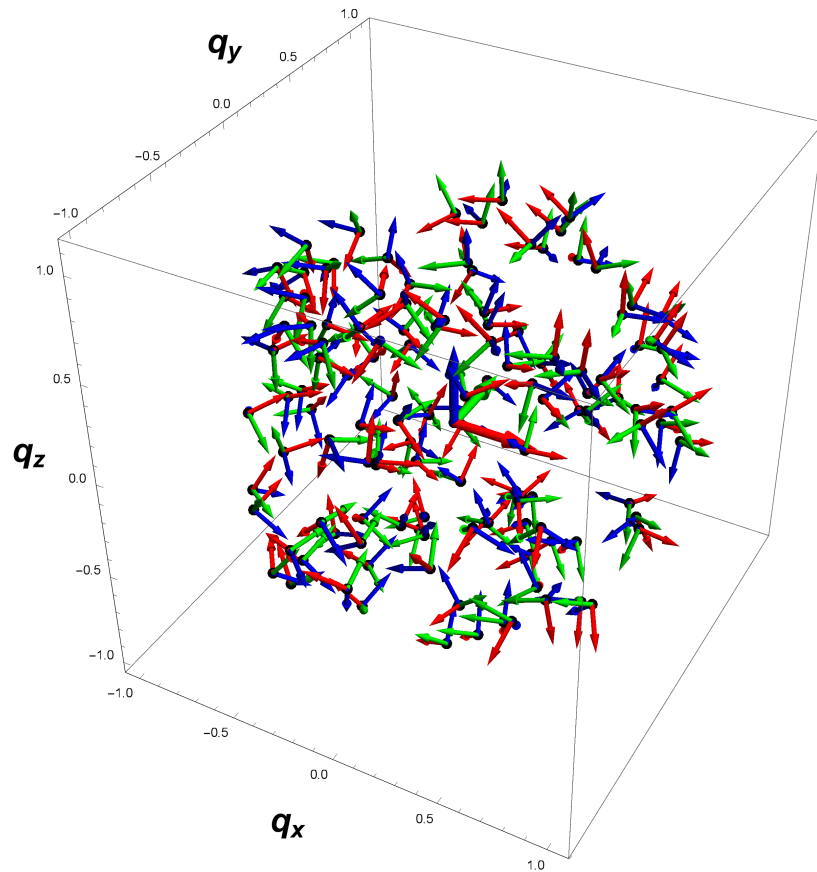
The exact solution (assuming resolved quaternion sign ambiguities in the data) is just

$$q_0 = \frac{V}{\|V\|}$$

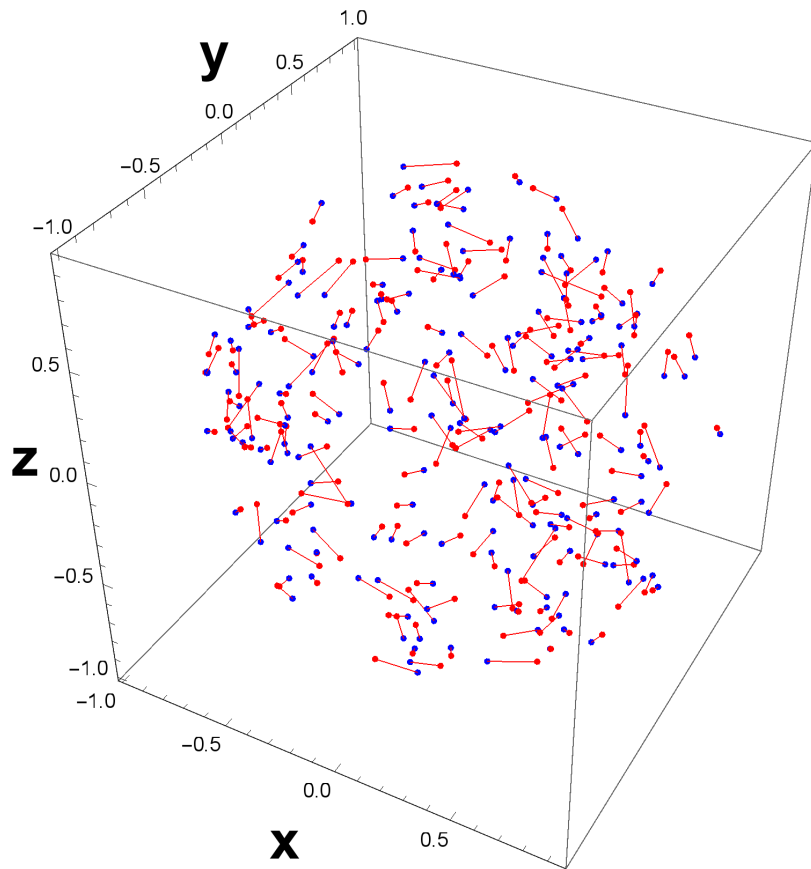
and the “cost” of the alignment at the optimal q_0 is therefore very simple:

$$\Delta_f(q_0) = q_0 \cdot V = \|V\| .$$

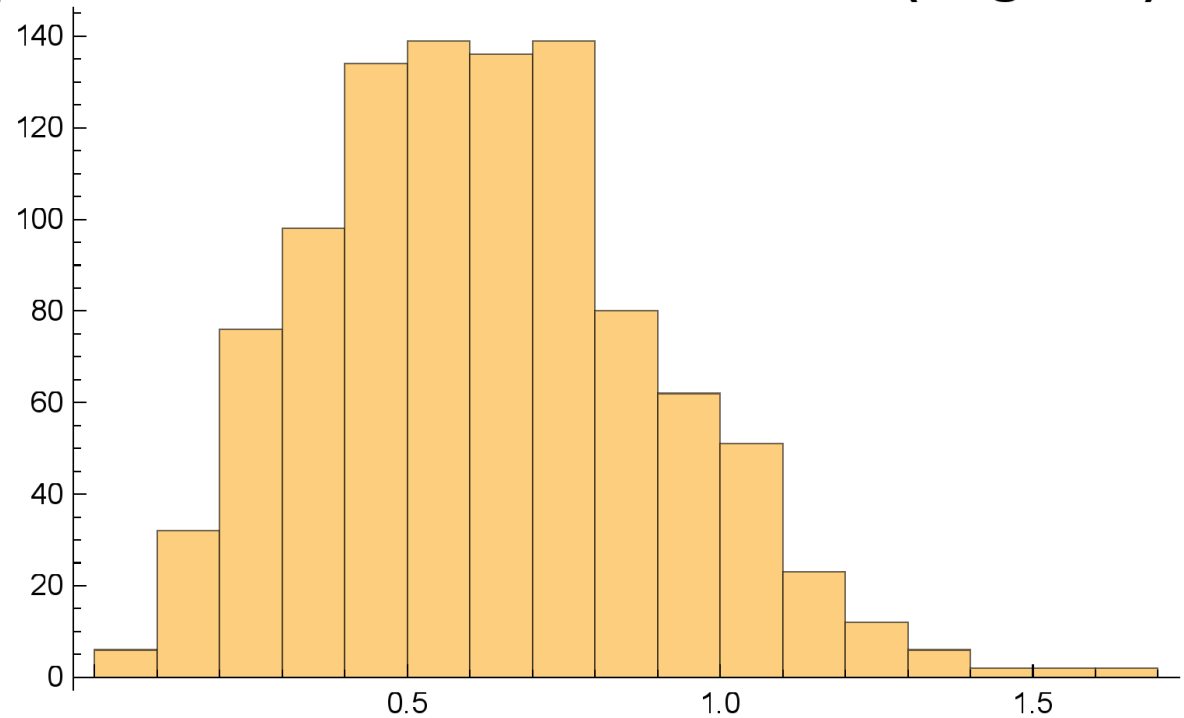
Slides of Quaternion Frame Analysis



Slides of Quaternion Frame Analysis, contd.



Quaternion Optimization Method: Chord vs Arc Rotation Error (degrees)



Remark on the 4D Data Problem

It turns out that while 3D data, with a 3×3 data matrix E_{3ab} , can be optimized with a quaternion-based symmetric, traceless 4×4 profile matrix $M(E_3)$, a slight extension of the 3D quaternion method expresses the 4D Euclidean data problem, with a 4×4 data matrix E_{4ab} , in terms of a quaternion eigenvalue problem with a *non-traceless, non-symmetric* 4×4 profile matrix $M(E_4)$.

We have an extension of the method given here that solves both the 4D spatial data and the 4D quaternion frame data problems *exactly*.

Summary

- **Defined RMSD Spatial Matching Problem.** Find a rotation that best aligns a pair of matched lists of spatial, orientation frame, or combined data. Used the 2D system to build intuition.
- **Presented the exact solutions of both spatial and frame data matching for the 3D problem.** The expression in terms of a triple of cube roots of unity was unknown and unexpected.
- **Also solved the 4D spatial and orientation frame RMSD problems in terms of exact algebraic eigenvalues of completely general 4D real matrices.**

Blank Slide

- .

- .

REPEAT with 2D Orientation Frames

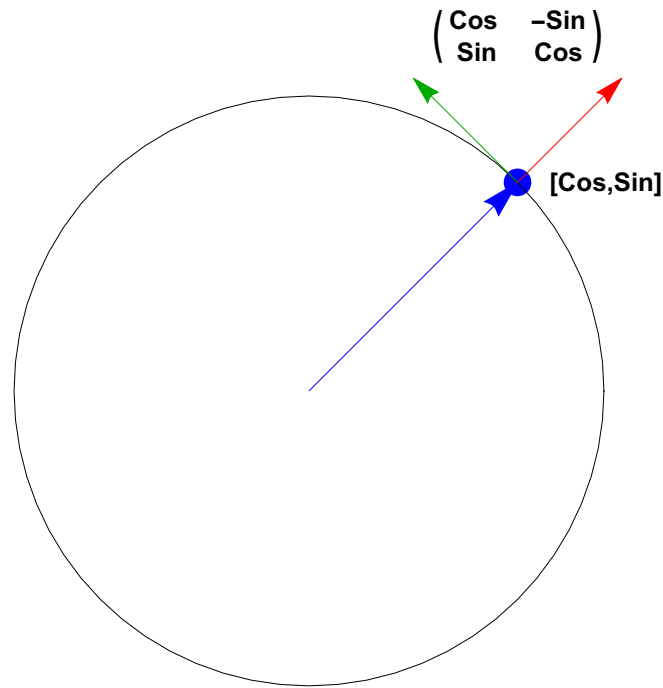
Orientation Frames can be adjoined to each point (this happens for amino acids in proteins, for example). Since they also can be misaligned by a global rotation, we can **repeat** the entire matching procedure for orientation: first in 2D, then extending to **quaternion-based frame representations** in 3D.

A 2D frame can be represented as a single unit-norm complex number $e^{i\theta} = \cos \theta + i \sin \theta$, interpreted as a pair of orthonormal vectors representing the frame:

$$\hat{\mathbf{u}} = (\cos \theta, \sin \theta)$$

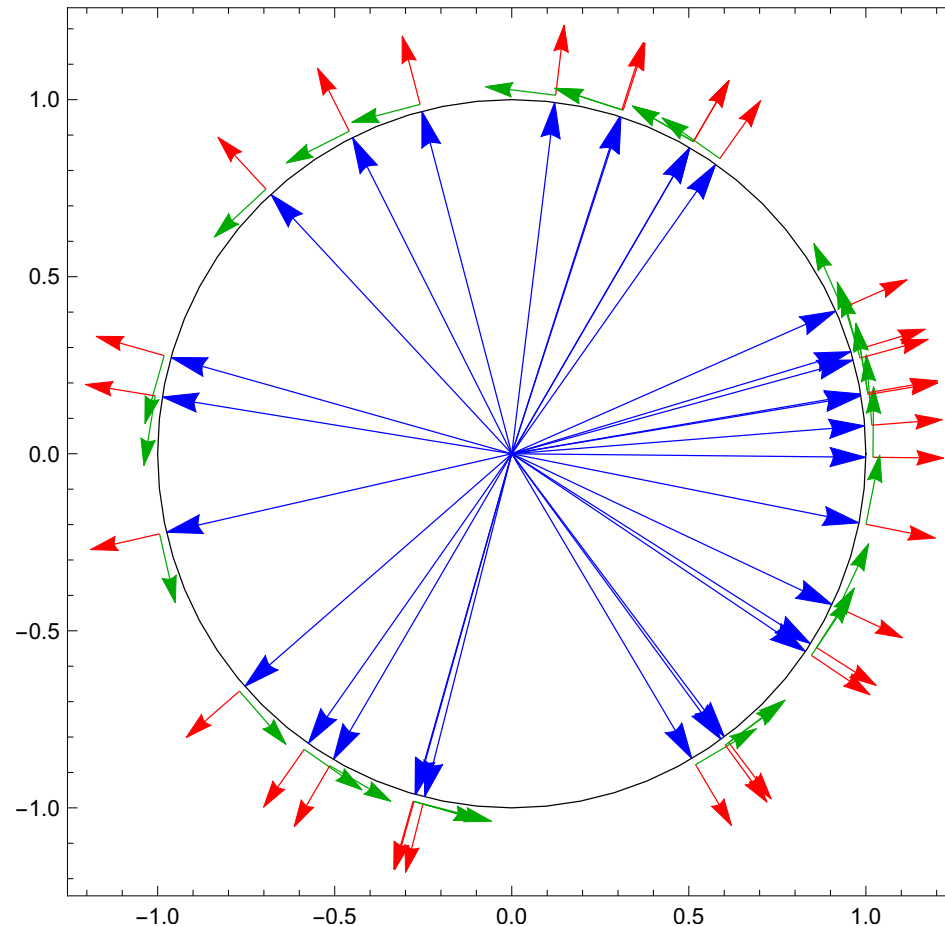
$$\hat{\mathbf{v}} = (-\sin \theta, \cos \theta) .$$

Visualizing a 2D Frame



In 2D, an angle θ represents an orthonormal frame (u, v) .

Visualizing 2D Orientation Collections



Our 2D frame data sets can then be represented as points on the unit circle.

Simplest version: 2D Frame Matching

Suppose we have a data set with a test set of 2D coordinate frames represented by angles $\{\alpha_k\}$ to be compared with a reference set of frame angles $\{\beta_k\}$. A close analog of the translational cumulative measure of the distances between frames is just the sum of squared angular distances

$$S_f^2 = \sum_{k=1}^N (\alpha_k - \beta_k)^2 .$$

2D Frame Minimizer

Since rotating a 2D frame (represented by α_k) by an angle θ is just complex multiplication of the form

$$e^{i\theta} e^{i\alpha_k} = e^{i(\theta + \alpha_k)} ,$$

a rotation of the entire set of test frames by the same angle has a cumulative measure of

$$\begin{aligned} S_f^2(\theta) &= \sum_{k=1}^N (\theta + \alpha_k - \beta_k)^2 \\ &= N\theta^2 + 2\theta \sum_{k=1}^N (\alpha_k - \beta_k) + \text{const} . \end{aligned}$$

... minimizing 2D frame match

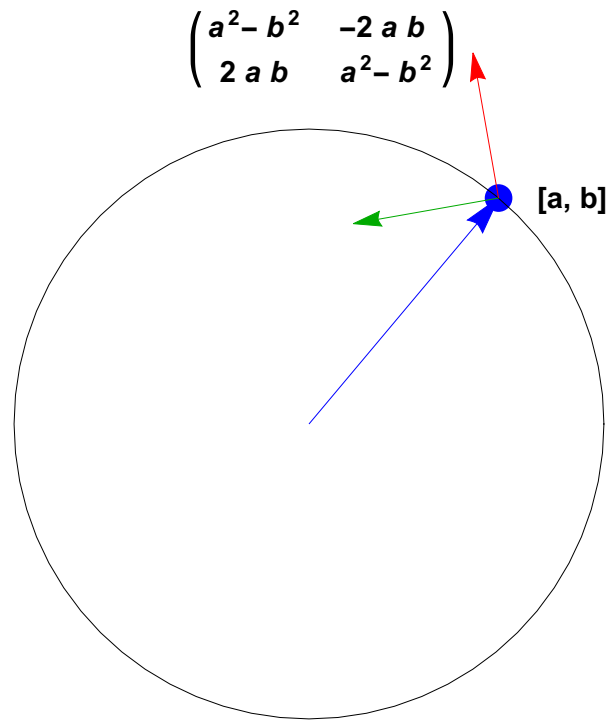
Differentiating with respect to θ , we see the optimal angle is essentially the center of mass of the differences,

$$\theta_0 = -\frac{1}{N} \sum_{k=1}^N (\alpha_k - \beta_k) .$$

This news is too good to be true: only in 2D do the angular variables behave *exactly* like a linear Euclidean space.

Nothing like this appears in higher dimensions.

As before: Need a double-valued 2D Frame



Extendability to 3D requires *half-angles*:

... double-valued 2D Frame

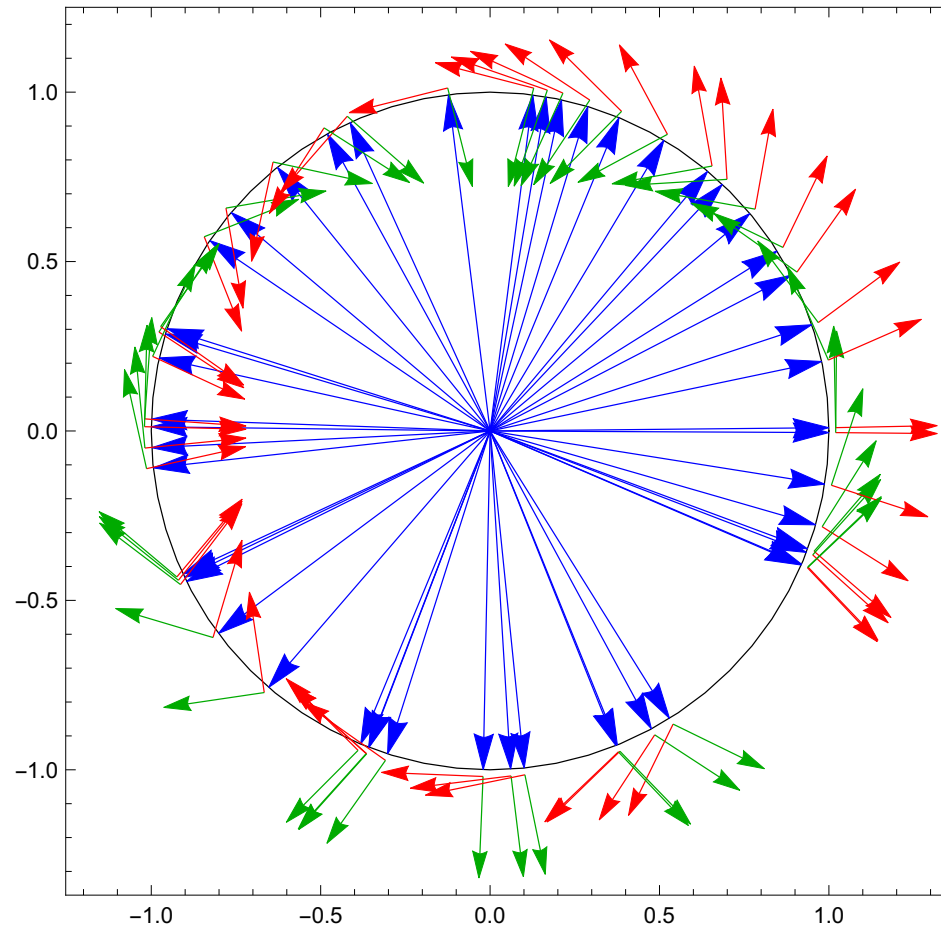
That is

$$(a, b) = (\cos(\theta/2), \sin(\theta/2)) ,$$

so the frame denoted by $(a, b) = e^{i\theta/2}$ is our old friend

$$\begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix} .$$

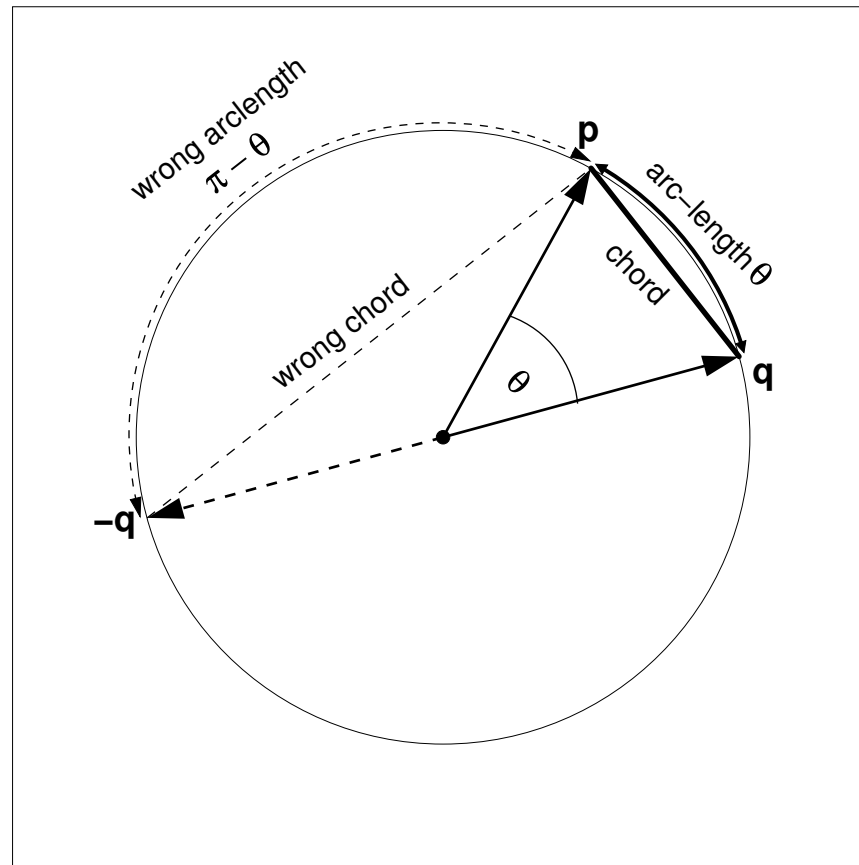
Visualizing 2D Orientation Collections



In 2D quaternions, the points (a, b) and $(-a, -b)$ on the unit circle *now* represent **ONE** orthonormal frame

$$u = (a^2 - b^2, 2ab), v = (-2ab, a^2 - b^2).$$

Tricky Part of Quaternion Distances



Because of sign ambiguity – the vectors q and $-q$ give the same rotation matrix — you need to choose $|\cos \theta|$ or the *minimum* of the chord distances to get the *right* arc-length or chord measure.

Frame distance with 2D chords

The **chord distance** on the previous slide is in fact the key to making the angular distance work like the spatial distance. It is an **approximation**, but it is a very good approximation. If we simply use $\| (R_2(\theta/2) \cdot \mathbf{p}) - \mathbf{r} \|$ as the fundamental minimizing measure, where \mathbf{p} and \mathbf{r} are unit vectors in the 2D complex plane, the optimal maximizing chord distance can be written

$$\Delta_f = \sum_{k=1}^N \cos(\theta/2 + b_k/2 - c_k/2)$$

Here $\mathbf{p} = (\cos(b/2), \sin(b/2))$, etc., to be quaternion-like.

Frame distance with 2D chords

After some manipulation with trigonometric arithmetic, the measure Δ_f can be written in essentially the same form as the Euclidean measure,

$$\Delta_f(a, b) = a(\theta)(E_{00} + E_{11}) + b(\theta)(E_{01} - E_{10}) = a\tilde{M} + b\tilde{N},$$

where $E_{ij} = \sum_{k=1}^N p_k^i r_k^j$. This is a linear equation in (a, b) and can be solved exactly as before.

Remarkably, this situation repeats in exactly the same way for the closed form solution of the 3D quaternion frame problem with the chord-measure approximation.

Space+Frame distance

Note: In fact, if one uses $(\tilde{\Delta}_f)^2$ as a measure, the angular optimization problem can be written **in terms of a quadratic form in (a, b)** , and thus can be folded directly into the **spatial optimization**, with a few caveats (one needs dimensional constants), to solve the **composite translational-rotational** optimization problem as well.