Fig. 1

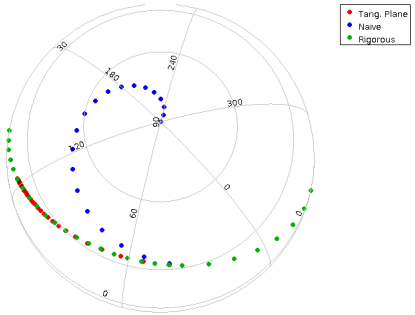

Fig. 2

1. Astrometry in ADQL: An Update

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

- Previously on astrometry in ADQL
- Adopting a subset of ESAC's functions (with a twist)
- Transforms... don't let me down!

(cf. Fig. 1)

2. At the Spring Interop

(cf. Fig. 2)

In spring, I had discussed the various ways to apply proper motions and pointed out that, while it does not make much of a difference now, we really should go for the "rigoruous treatment", which, absent information on radial velocities, will have objects move on great circles.

There are still a few open questions, though.

3. Two Functions

Following ESAC, we have two functions distinguished by their return type:

```
ivo_epoch_prop(
  ra DOUBLE, dec DOUBLE, parallax DOUBLE,
  pmra DOUBLE, pmdec DOUBLE, radial_velocity DOUBLE,
  ref_epoch DOUBLE, out_epoch DOUBLE) -> DOUBLE[6]
ivo_epoch_prop_pos(
  ra DOUBLE, dec DOUBLE, parallax DOUBLE,
  pmra DOUBLE, pmdec DOUBLE, radial_velocity DOUBLE,
  ref_epoch DOUBLE, out_epoch DOUBLE) -> POINT
```

- Only ra, dec, epochs mandatory, elsewhere NULL=0.
- I'd prefer a single delta_t argument, but if you agree, move fast: It's not that much of a big deal to me, and in the end I feel following existing practices overrides practicality.
- Units: deg, mas, mas/yr, km/s, julian years. Yes, that's somewhat eclectic, but breaking widespread practices here would, I think, hurt quite a few people.

Another sore point is that we are returning the transformed 6d vector in an array. Arrays are supposed to be homogeneous, and mixing positions and their derivatives in all kinds of different units in a single array is close to a mortal sin. Just imagine writing a FIELD element for that thing in VOTable.

It would be a lot cleaner to return some sort of record type. But ADQL does not have that. Perhaps some day we will define a SPHER_POS type that would do this cleanly. Until then, I don't think we have a lot of choice *if* we want people to get propagated motions (and distances). Do we, really?

4. The Fine Print

These are already implemented on the GAVO DC TAP service.

If you try them and compare with astropy, you will notice differences in the μas range.

This is because neither ESAC nor I apply

- relativistic corrections – these become important when stars don't have $\beta \approx 0$. We really don't have that in our galaxy. Applying them, however, requires numerical approximation. Greatly uglifying the code for a precision (or, more likely, mis-calculation due to incorrect distances) probably irrelevant in our use cases seems a bad deal to me.
- secular aberration – that's a consequence of the fact that the solar system barycenter does not move in a straight line. It seems wrong to correct for the non-linear motion of the sun when we assume linear motion of the stars.

I'd say we should document this and create new UDFs if these ever become desirable.

5. In pgsphere

To support these functions, I have added code for Lindegren's rigorous treatment (ESA SP-1200, Hipparcos) in pgsphere:

https://github.com/postgrespro/pgsphere/pull/8

Please review – it would be great if we could get that merged and then perhaps have a new pgsphere release. Then again, ESAC has published a pure-SQL (mildly postgres-specific) solution for these, so you do not absolutely *need* the updated pgsphere to have the two functions.

6. Frame Transforms

I'd still need a second implemenation for

```
gavo_transform(
  from_sys TEXT, to_sys TEXT,
  geo GEOMETRY) -> GEOMETRY
```

as in

```
1=CONTAINS(
  POINT(l, b),
  gavo_transform('ICRS', 'GALACTIC', s_region))
```

Do me and your users a favour and implement it – as long as the transforms are rotations, this is really simple on top of pgsphere. In case you would appreciate inspiration, see gavo/adql/ufunctions.py in the DaCHS source code.