Using proj4's `ob_tran` for arbitrary grid rotations

Kevin Stadler

February 10, 2017

proj.4's powerful oblique transform pseudo-projection `ob_tran` allows for arbitrary globe rotations (and more) to be performed on-the-fly, using data in any underlying projection format. To perform a simple grid rotation in which some coordinate `(x,y)` on the globe gets aligned with the origin before being reprojected, one simply has to specify the source data projection as

```
+proj=ob_tran +o_lon_p=x +o_lat_p=(90+y) +to_meter=0.0174532925199432954329 5
```

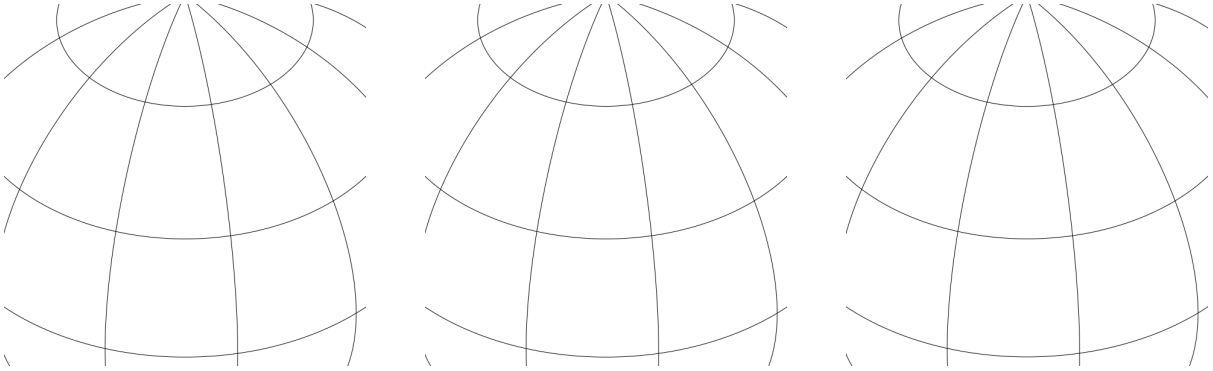
followed by specifying the original projection of the data with `+o_proj=...` as well as any other arguments relevant to the underlying projection. (The `+to_meter` argument is there to convert the units used to specify the rotated pole coordinates from degrees to radians and can also be given with somewhat lower accuracy.)

As shown below, direct reprojection of the original data to an orthogonal projection centered on an arbitrary point on the globe (left) is identical to reprojecting the `ob_tran` shifted data to an orthogonal projection centered on the origin (middle).

Left: projection to `+proj=ortho +ellps=WGS84 +lon_0=-71.86 +lat_0=42.13` from `+proj=longlat +ellps=WGS84`

Middle: projection to `+proj=ortho +ellps=WGS84 +lon_0=0 +lat_0=0` from `+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +to_meter=0.017453292519943295 +o_proj=longlat +ellps=WGS84`

Right: projection to `+proj=ortho +ellps=sphere +lon_0=0 +lat_0=0` from `+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +to_meter=0.017453292519943295 +o_proj=longlat +ellps=sphere`



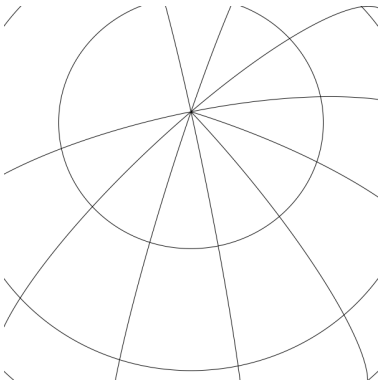
The `ob_tran` pole rotation does not just do a simple datum shift but is sensitive to the shape of the ellipsoid. So while the shifted-origin orthogonal projection of the original data and the origin-centered projection of the `ob_tran` transformed data set are identical, the same shift of the same data set but declared to be from and to a *spherical* rather than WGS84 ellipsoid (shown on the right) yields a slightly different result:

```
diff ob_tran1.png ob_tran2.png
diff ob_tran1.png ob_tran_sphere.png
```

```
## Binary files ob_tran1.png and ob_tran_sphere.png differ
```

Grid shift drawing artefacts

Note that data sets are often formatted so that there are no geometries straddling the date line, which would lead to artefacts (especially in mapnik). Unless one takes care of fixing all geometries accordingly (for example using `ogr2ogr`'s 'clip' function) one might run into problems. Even with our simple graticule data set here, the two of the twelve 30° graticules which are pushed across the date line by our shift are not rendered:



Other source data projections

The `+to_meter` parameter has to be omitted if the data source of the layer is already in a projected (rather than in a geographic) coordinate system, such as is the case for the default import of OSM data via `osm2pgsql` (which is in Web Mercator projection coordinates).

Left: projection to `+proj=ortho +ellps=WGS84 +lon_0=-71.86 +lat_0=42.13` from `+proj=merc +k=1 +a=6378137 +b=6378137 +no_defs`

Right: projection to `+proj=ortho +ellps=WGS84 +lon_0=0 +lat_0=0` from `+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +o_proj=merc +k=1 +a=6378137 +b=6378137 +no_defs`

```
diff ob_tran_merc1.png ob_tran_merc2.png
```

More examples

Reprojection of the rotated Web Mercator data to a Mollweide projection using two different target ellipsoids. Even when only rendering a small window around the equator, the relative oblateness of the two ellipsoids (WGS84 on the left, sphere on the right) cause miniscule differences in the output.

Left: projection to `+proj=moll +ellps=WGS84` from `+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +o_proj=merc +k=1 +a=6378137 +b=6378137`

Right: projection to `+proj=moll +ellps=sphere` from `+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +o_proj=merc +k=1 +a=6378137 +b=6378137`

```
diff ob_tran_mollweide_wgs84.png ob_tran_mollweide_sphere.png
```

```
## Binary files ob_tran_mollweide_wgs84.png and ob_tran_mollweide_sphere.png differ
```

Code

```
# get simple graticule data set
if [ ! -f "ne_110m_graticules_30.zip" ]; then
  wget http://www.naturalearthdata.com/http//www.naturalearthdata.com/download/110m/physical/ne_110m
  unzip ne_110m_graticules_30.zip
fi
```

Producing the figures above using mapnik via its python bindings:

```
from mapnik import *

m = Map(600, 600)
m.background = Color('white')
m.zoom_to_box(Box2d(-4500000, -4500000, 4500000, 4500000))

s = Style()
r = Rule()
ln = LineSymbolizer()
r.symbols.append(ln)
s.rules.append(r)
m.append_style('grid', s)

lr = Layer('grid')
lr.datasource = Shapefile(file='ne_110m_graticules_30.shp')
```

```
lr.styles.append('grid')
m.layers.append(lr)

# way #1: center orthographic on place of interest
m.srs = '+proj=ortho +ellps=WGS84 +lon_0=-71.86 +lat_0=42.13'
render_to_file(m, 'ob_tran1.png')

# way #2: orthographic projection on origin, ob_tran shift both layers
m.srs = '+proj=ortho +ellps=WGS84'
m.layers[0].srs = '+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +to_meter=0.017453292519943295 +o_p
render_to_file(m, 'ob_tran2.png')

# same shift but with different ellipsoid
m.srs = '+proj=ortho +ellps=sphere'
m.layers[0].srs = '+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +to_meter=0.017453292519943295 +o_p
render_to_file(m, 'ob_tran_sphere.png')

# shift data up by another 30 degree so we can see north pole
m.layers[0].srs = '+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=162.13 +to_meter=0.017453292519943295 +o_p
render_to_file(m, 'ob_tran_shift.png')


# osm data from postgis (in web mercator by default): omit +to_meter
m.layers[0].datasource = PostGIS(host="localhost", dbname="gis", table="planet_osm_line")

# close-up of osm data
m.zoom(40000 / m.scale_denominator())

# plain mercator data to orthographic projection (centered on point)
m.srs = '+proj=ortho +ellps=WGS84 +lon_0=-71.86 +lat_0=42.13'
m.layers[0].srs = '+proj=merc +k=1 +a=6378137 +b=6378137 +no_defs'
render_to_file(m, 'ob_tran_merc1.png')

# shifted web mercator data to orthographic projection (centered on origin)
m.srs = '+proj=ortho +ellps=WGS84'
m.layers[0].srs = '+proj=ob_tran +o_lon_p=-71.86 +o_lat_p=132.13 +o_proj=merc +k=1 +a=6378137 +b=637
render_to_file(m, 'ob_tran_merc2.png')

# shifted web mercator data to mollweide, assuming two different ellipsoids
m.srs = '+proj=moll +ellps=sphere'
render_to_file(m, 'ob_tran_mollweide_sphere.png')
m.srs = '+proj=moll +ellps=WGS84'
render_to_file(m, 'ob_tran_mollweide_wgs84.png')
```