OctPROJ
Calling PROJ from GNU Octave*

José Luis García Pallero†

May 16, 2020 (version 2.0.1)
May 9, 2020 (version 2.0.0)
April 2, 2019 (version 1.1.6)
June 16, 2015 (version 1.1.5)
February 13, 2015 (version 1.1.4)
June 20, 2013 (version 1.1.3)
October 1, 2012 (version 1.1.1)
April 13, 2012 (version 1.1.0)
May 13, 2011 (version 1.0.2)
November 29, 2010 (version 1.0.1)
February 9, 2010 (version 1.0.0)

Abstract

This is a small introduction to using the OctPROJ package. In this text, you can overview the basic usage of the functions in GNU Octave[1]. If you need a detailed description about the options, available projections and coordinate reference systems, please visit the PROJ website[2]. Starting with OctPROJ 2.0.0 the code was upgraded to PROJ version 6.3.0, so older versions of the library could not work.

Contents

†ETSI en Topografía, Geodesia y Cartografía, Universidad Politécnica de Madrid. jlg.pallero@upm.es, jgpallero@gmail.com.

[1] http://www.octave.org.

[2] https://proj.org/.

1 Overview

`OctPROJ` allows you to perform cartographic projections and coordinate reference systems transformation in GNU Octave using the `PROJ` library. You can take the power of `PROJ` using the facilities that GNU Octave provides, without know the internals of the `PROJ` C API. You can use the conversion programs coming with `PROJ` distribution, but for use them in GNU Octave you must make system calls. With `OctPROJ`, `PROJ` can be integrated in GNU Octave scripts in a natural way.

2 Installation

As several GNU Octave packages, `OctPROJ` installation consists in compiling the C++ kernel sources (see section 3), link them against GNU Octave library to generate `*.oct` functions and copy this `*.oct` executables and other `*.m` functions into a working directory.

The automagic procedure can be easily done by running the command:

```
octave:1> pkg install octproj-x.x.x.tar.gz
```

where `x.x.x` is the version number.

After that, the functions and documentation are installed in your machine and you are ready for use the package.

3 Kernel wrapper

The main functions (the functions which make the actual computations) are written in separate files: `projwrap.h` and `projwrap.c`.

The files contain three functions:

- `proj_fwd`: forward computation of geodetic to projected coordinates.
- `proj_inv`: inverse computation of projected to geodetic coordinates.
- `proj_transform`: general transformations. It is possible to make forward, inverse and other transformations using only one function (see `PROJ` documentation).

3.1 Error handling

Error handling in the kernel wrapper is based on error codes from `PROJ`. Functions in `projwrap` return the `PROJ` error code and the `PROJ` text string error message, which can be catched in order to work in this case.

The functions can stop the execution in presence of errors depending on the nature of the error.

- If exist an error involving a general parameter of the projection, the execution stops.
- If an error is found due to a wrong or out of domain input coordinate, the execution don't stops, but the error code is activated and the output coordinate corresponding to the error position have a special value.

4 GNU Octave functions

Two types of functions are programmed for GNU Octave: `*.oct` functions and `*.m` functions.

4.1 `*.oct` functions

These functions are linked with `projwrap`. You can use it, but it is not recommended because the input arguments are more strict (always column vectors) than `*.m` functions and don't check for errors.

The functions are:

- `_op_geod2geoc`: transformation between geodetic and cartesian tridimensional geocentric coordinates.
- `_op_geoc2geod`: transformation between cartesian tridimensional geocentric and geodetic coordinates.
- `_op_fwd`: forward projection (calls the function `proj_trans_generic` with `PJ_FWD` option).
- `_op_inv`: inverse projection (calls the function `proj_trans_generic` with `PJ_INV` option).
- `_op_transform`: general transformation (calls `proj_trans_generic` with `PJ_FWD` option).

4.2 *.m functions

These functions make the computations by calling the *.oct functions. You must call these functions because you can use various types of input (scalars, vectors or matrices) and checking of input arguments (data type, dimensions) is performed.

The functions are the same as in section 4.1 (without the _ at the beginning of the name):

- op_geod2geoc: calls _op_geod2geoc.
- op_geoc2geod: calls _op_geoc2geod.
- op_fwd: calls _op_fwd.
- op_inv: calls _op_inv.
- op_transform: calls _op_transform.

4.3 Error handling

*.oct and *.m functions can emit errors or warnings, some due to errors in input arguments and other due to errors in functions from projwrap kernel execution (see section 3).

Errors due to wrong input arguments (data types, dimensions, etc.) can be only given for *.m functions and this is the reason because the use of these functions are recommended. In this case the execution is aborted and nothing is stored in output arguments.

Errors due to the execution of projwrap kernel can be emitted for both *.oct and *.m functions. If the error is due to an erroneous projection parameter the execution is aborted and nothing is stored in output arguments; but if the error is due to a wrong or out of domain input coordinate, a warning is emitted and the execution has a normal end.

5 Examples

5.1 Geodetic to geocentric and vice versa

```
lon=-6*pi/180;lat=43*pi/180;h=1000;
[x,y,z]=op_geod2geoc(lon,lat,h,6378388,1/297)
x =  4647300.723262573
y = -488450.9885681378
z =  4328259.364257743

[lon,lat,h]=op_geoc2geod(x,y,z,6378388,1/297);
lon*=180/pi,lat*=180/pi,h
lon = -6
lat =  42.99999999999999
h =  1000
```

5.2 Forward and inverse projection

```
lon=-6*pi/180;lat=43*pi/180;
[x,y]=op_fwd(lon,lat,'+proj=utm +lon_0=3w +ellps=GRS80')
x =  255466.9805506805
y =  4765182.932683380

[lon,lat]=op_inv(x,y,'+proj=utm +lon_0=3w +ellps=GRS80');
lon*=180/pi,lat*=180/pi
lon = -5.999999999999999
lat =  43
```

5.3 Forward and inverse transformation: `op_transform`

This function takes into account some of the new `PROJ` capabilities. The main of them are:

- Projection and coordinate reference systems can be defined not only via the classical `+proj=` format, but also using EPSG[3] codes.
- Time coordinate for dynamical coordinate reference systems can be used.

5.3.1 With altitude and time

```
lon=-6;lat=43;h=1000;t=1;
[x,y,h,t]=op_transform(lon,lat,h,t,...
                       '+proj=latlong +ellps=GRS80',...
                       '+proj=utm +lon_0=3w +ellps=GRS80')
x =  255466.9805506804
y =  4765182.932683380
h =  1000
t =  1

[lon,lat,h,t]=op_transform(x,y,h,t,...
                           '+proj=utm +lon_0=3w +ellps=GRS80',...
                           '+proj=latlong +ellps=GRS80')
lon = -6
lat =  43
h =  1000
t =  1
```

Note that in this case, where the `+proj=latlong` was used, the input geodetic coordinates are provided in degrees instead of radians. This is a feature of `PROJ` when `+proj=latlong` is used in coordinate reference system transformation, but not in cartographic projection, i.e., when it is used in `op_transform`, but not in `op_fwd` nor `op_inv`.

5.3.2 With altitude

```
lon=-6;lat=43;h=1000;
```

[3] http://www.epsg-registry.org/.

```
[x,y,h]=op_transform(lon,lat,h,...
                     '+proj=latlong +ellps=GRS80',...
                     '+proj=utm +lon_0=3w +ellps=GRS80')
x =  255466.9805506804
y =  4765182.932683380
h =  1000

[lon,lat,h]=op_transform(x,y,h,...
                         '+proj=utm +lon_0=3w +ellps=GRS80',...
                         '+proj=latlong +ellps=GRS80')
lon = -6
lat =  43
h =  1000
```

5.3.3 Without altitude

```
lon=-6;lat=43;
[x,y]=op_transform(lon,lat,'+proj=latlong +ellps=GRS80',...
                   '+proj=utm +lon_0=3w +ellps=GRS80')
x =  255466.9805506804
y =  4765182.932683380

[lon,lat]=op_transform(x,y,'+proj=utm +lon_0=3w +ellps=GRS80',...
                       '+proj=latlong +ellps=GRS80')
lon = -6
lat =  43
```

In all the preceding examples the input coordinates can be scalars, vectors or matrices, and height and/or time can be provided as empty matrices.

5.3.4 Using EPSG codes

Supose a transformation from UTM zone 30 coordinates to UTM zone 31 in the ETRS89 coordinate reference system. Using the classical definition the operation is performed as

```
x1=[600000;700000];y1=[4800000;4900000];z1=100;
[x2,y2,z2]=op_transform(x1,y1,z1,...
                        '+proj=utm +lon_0=3w +ellps=GRS80',...
                        '+proj=utm +lon_0=3e +ellps=GRS80')
x2 =

   113677.9843623374
   220776.6200746754

y2 =

   4810303.384473779
   4902896.002979981

z2 =
```

```
   100
   100
```

The same transformation can be carried out using the corresponding EPSG codes, i.e., `EPSG:25830` and `EPSG:25831`[4]:

```
x1=[600000;700000];y1=[4800000;4900000];z1=100;
[x2,y2,z2]=op_transform(x1,y1,z1,'EPSG:25830','EPSG:25831')
x2 =

   113677.9843623374
   220776.6200746754

y2 =

   4810303.384473779
   4902896.002979981

z2 =

   100
   100
```

5.4 Error due to an erroneous parameter

```
lon=-6*pi/180;lat=43*pi/180;
[x,y]=op_fwd(lon,lat,'+proj=utm +lon_0=3w +ellps=GRS8')
proj_create: Error -9: unknown elliptical parameter name
error:
        In function op_fwd:
        In function _op_fwd:
        Error in projection parameters
        unknown elliptical parameter name
        +proj=utm +lon_0=3w +ellps=GRS8
error: called from
    op_fwd at line 96 column 5
```

5.5 Error due to latitude too big

```
lon=[-6*pi/180;-6*pi/180];lat=[43*pi/180;43];
[x,y]=_op_fwd(lon,lat,'+proj=utm +lon_0=3w +ellps=GRS80')
warning: _op_fwd:

warning: Projection error in point 2 (index starts at 1)
x =

   255466.98055
```

[4] See https://spatialreference.org/ref/epsg/25830/, and https://spatialreference.org/ref/epsg/25831/.

```
            Inf

y =

   4765182.93268
             Inf
```

6 Notes

Apart from http://octave.sourceforge.net/octproj/index.html, an up to date version of `OctPROJ` can be downloaded from https://bitbucket.org/jgpallero/octproj/.

References

[1] Eaton, John W.; Bateman, David; Hauberg, Søren; and Wehbring, Rik; GNU Octave. A high-level interactive language for numerical computations; Edition 5 for Octave version 5.2.0, January 2020; https://www.gnu.org/software/octave/octave.pdf; Permanently updated at https://www.gnu.org/software/octave/support.html.

[2] Evenden, Gerald I.; Cartographic Projection Procedures for the UNIX Environment—A User's Manual; USGS Open-File Report 90-284; 2003; ftp://ftp.remotesensing.org/proj/OF90-284.pdf.

[3] Evenden, Gerald I.; Cartographic Projection Procedures Release 4, Interim Report; 2003; ftp://ftp.remotesensing.org/proj/proj.4.3.pdf.

[4] Evenden, Gerald I.; Cartographic Projection Procedures Release 4, Second Interim Report; 2003; ftp://ftp.remotesensing.org/proj/proj.4.3.I2.pdf.

[5] Snyder, John Parr; Map Projections: A Working Manual; USGS series, Professional Paper 1395; Geological Survey (U. S.), 1987; http://pubs.er.usgs.gov/usgspubs/pp/pp1395.