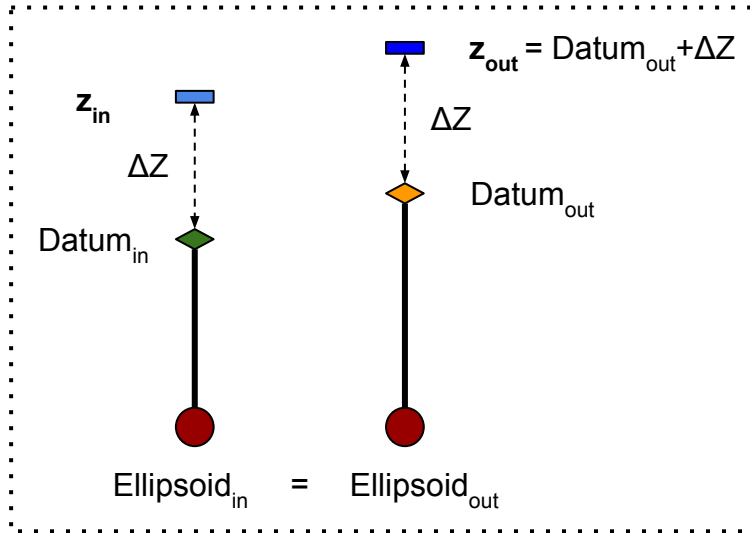


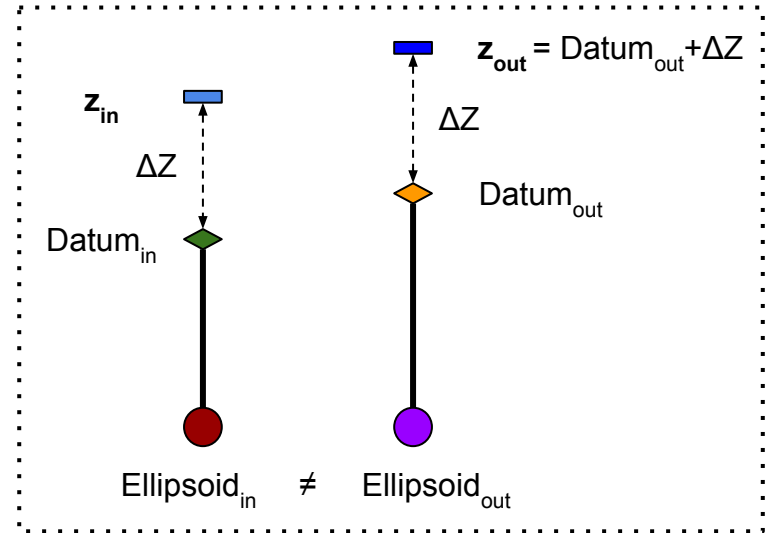
Kinds of Vertical Transformations

Case a: Different Datum, Same Ellipsoid



1. z_{in}
2. $\Delta Z = z_{\text{in}} - \text{Datum}_{\text{in}(\text{.geotiff})}$
3. $z_{\text{out}} = \text{Datum}_{\text{out}(\text{.geotiff})} + \Delta Z$

Case b: Different Datum, Different Ellipsoid



1. z_{in}
2. $\Delta Z = z_{\text{in}} - \text{Datum}_{\text{in}(\text{.geotiff})}$
3. Ellipsoidal Transformation
4. $z_{\text{out}} = \text{Datum}_{\text{out}(\text{.geotiff})} + \Delta Z$

* This includes the tidal datums (MLLW, MSL) and SGEIOD2022 transformations, which are all referenced to the Ellipsoid ITRF2020.

* This includes any conversion including NAVD88 (which is referenced to the NAD83 Ellipsoid) and xGeoid20b (which is referenced to the Ellipsoid ITRF 2014).

CMVD Architecture

1. The Geotiffs (datums) are declared in `_path.py`
2. All Proj-based Ellipsoid transformations are declared in `_geoid_tr.py`
3. The main script `vdatum.py` builds the entire pipeline calling the corresponding datums (geotiffs from `_path.py`) and ellipsoidal transformations (Proj commands in `_geoid_tr.py`). Once the pipeline is build, the entire transformation is carried by `pyproj.Transformer.from_pipeline()`

```
1  #!/usr/bin/env python3
2
3  NAVD88_G2018=f"us_noaa_g2018u0.tif"
4  XGEOID208=f"https://noaa-nos-stofs2d-pds.s3.amazonaws.com/_archive/coastalmodeling-vdatum/xGEOID208.tif"
5  MLLW_ITRF2020_2020=f"https://noaa-nos-stofs2d-pds.s3.amazonaws.com/_archive/coastalmodeling-vdatum/us_noaa_nos_MLLW-I
6  LMSL_ITRF2020_2020=f"https://noaa-nos-stofs2d-pds.s3.amazonaws.com/_archive/coastalmodeling-vdatum/us_noaa_nos_LMSL-I
7  SGEOID2022=f"https://noaa-nos-stofs2d-pds.s3.amazonaws.com/_archive/coastalmodeling-vdatum/us_noaa_sgeoid2022v1a.tif"
```

```
def inputs(vd_from,vd_to):
    """
    Calls the respective (height - geoid), (geoid to geoid), and (gec
    transformations given the vertical datum of origin and target ver
    """
    if vd_from == "xgeoid20b" and vd_to == "mllw":
        h_g = _path.XGEOID208
        g_g = _geoid_tr.ITRF2014_to_ITRF2020
        g_h = _path.MLLW_ITRF2020_2020
    elif vd_from == "mllw" and vd_to == "xgeoid20b":
        h_g = _path.MLLW_ITRF2020_2020
        g_g = _geoid_tr.ITRF2020_to_ITRF2014
        g_h = _path.XGEOID208
    elif vd_from == "xgeoid20b" and vd_to == "lmsl":
        h_g = _path.XGEOID208
        g_g = _geoid_tr.ITRF2014_to_ITRF2020
        g_h = _path.LMSL_ITRF2020_2020
    elif vd_from == "lmsl" and vd_to == "xgeoid20b":
        ...

    return h_g,g_g,g_h
```

```
1  #!/usr/bin/env python3
2
3  ITRF2020_to_ITRF2014 = """\
4      +step +proj=cart +ellps=GRS80
5      +step +inv +proj=helmert +x=0.0014 +y=0.0009 +z=-0.0014 +rx=0 +ry=0 +rz=0 \
6          +s=0.00042 +dx=0 +dy=0.0001 +dz=-0.0002 +drx=0 +dry=0 +drz=0 +ds=0 \
7          +t_epoch=2015 +convention=position_vector \
8      +step +inv +proj=cart +ellps=GRS80"""
9
10 ITRF2014_to_ITRF2020 = """\
11     +step +proj=cart +ellps=GRS80
12     +step +proj=helmert +x=0.0014 +y=0.0009 +z=-0.0014 +rx=0 +ry=0 +rz=0 \
13         +s=0.00042 +dx=0 +dy=0.0001 +dz=-0.0002 +drx=0 +dry=0 +drz=0 +ds=0 \
14         +t_epoch=2015 +convention=position_vector \
15     +step +inv +proj=cart +ellps=GRS80"""
16
17 ITRF2020_2020_to_NAD832011_2010 = """\
18     +step +proj=cart +ellps=GRS80 \
19     +step +proj=set +v_4=2010 \
20     +step +proj=helmert \
21         +dx=0.00037 +dy=0.00035 +dz=0.00074 +drx=0.000045 +dry=-0.000666 +drz=-0
```

```
def build_pipeline(lat, lon ,z ,h_g , g_g, g_h, online, epoch=None):
    """
    Basic pipeline that is common to all transformations.
    h_g: height - geoid
    g_g: geoid to geoid
    g_h: geoid - height
    """

    pipeline=f"""+proj=pipeline
+step +proj=axisswap +order=2,1
+step +proj=unitconvert +xy_in=deg +z_in=m +xy_out=rad +z_out=m
+step +proj=vgridshift +grids={h_g} +multiplier=1
{g_g}
+step +proj=vgridshift +grids={g_h} +multiplier=-1
+step +proj=unitconvert +xy_in=rad +z_in=m +xy_out=deg +z_out=m
+step +proj=axisswap +order=2,1
"""

    if online is True:
        pyproj.network.set_network_enabled(active=True)

    tr = pyproj.Transformer.from_pipeline(pipeline).transform

    if epoch is not None:
        if isinstance(lat,(int,float)):
            t=epoch
        else:
            t=[epoch for l in lat]

        clat,clon,cz,ct = tr(lat,lon,z,t)
    else:
        clat,clon,cz = tr(lat,lon,z)

    return clat,clon,cz
```

Commonly asked questions

- ❖ What is the goal with CMVD?
 - This is a simple package that works as a wrapper for PyProj, intended as a temporary solution for the basic datum conversions needs of the OCS/Storm Surge Modeling Team. This not a comprehensive tool and we don't have plans to develop it beyond the initial scope. This tool will be retired once the Modeling Team finds a better solution (e.g. python version of VDatum that is actively maintained, etc.).
- ❖ Why do I get *inf* values on my conversions?
 - Datum conversions are only possible where the datum is available (see the extent of the geotiff files).
- ❖ But I get a result if I enter the same x,y,z on VDatum.
 - But you shouldn't.
- ❖ Why the z values I get are slightly different than the ones from VDatum?
 - That is hard to tell, the VDatum pipeline is not publicly available, so it is hard to assess why. However, values should be relative close. **Note CMVD is not “making the conversions”, it is just a wrapper for Proj.** Uncertainties (and reasons why the values might be slightly different than VDatum) can be attributed to interpolation errors, slightly different reference frame (ellipsoid) conversion equations, different epochs, etc.
 - **If you find errors in the CMVD pipelines we welcome you to submit a PR!**

How to we know the conversions are correct? How to add new Datums?

1. The approach we use is 100% based on [PyProj](#) and datum files (geotiffs) that are either part of the [cdn.proj.org](#) or experimental reference frames ([xgeoid20b](#), SGeoid2022).
 - a. The experimental geotiffs are either made available online by NOAA, or shared with us by NOAA developers (i.e., they are not part of the Proj database [cdn.proj.org](#) yet)
2. One can create a new Proj pipeline (i.e., datum conversion) by using the `projinfo` command. This command provides the Proj string commands used to create a Proj pipeline.
3. Next we show the step by step process to create a Proj pipeline for a NAVD88 to xgeoid20b conversion. Note you will need to know the EPSG codes for the datums you want to convert. You can search for the EPSG code at [epsg.org/home](#). You also might need to know what reference frame your datums use (although `projinfo` can help you to figure that out). Here we will assume we know that NAVD88 is in NAD83 and xgeoid20b is in ITRF2014.

1. Use projinfo to help find which file to use for NAVD88/NAD83

- Note that you enter the respective EPSG codes as input and output (epsg.org/home)
- Note that the projinfo gives you the proj step needed for the conversion (highlighted line)
- Note that it also gives the link to the .tif file used

```
!projinfo -s EPSG:6318 -t EPSG:5703 --spatial-test intersects --hide-ballpark -o PROJ
```

Candidate operations found: 2

Operation No. 1:

DERIVED_FROM(EPSG):9229, NAD83(2011) to NAVD88 height (3), 0.015 m, United States (USA) - CONUS onshore - Alabama; Arizona; Delaware; Florida; Georgia; Idaho; Illinois; Indiana; Iowa; Kansas; Kentucky; Louisiana; Maine; Maryland; Massachusetts; Montana; Nebraska; Nevada; New Hampshire; New Jersey; New Mexico; New York; North Carolina; North Dakota; Ohio; Oklahoma; olina; South Dakota; Tennessee; Texas; Utah; Vermont; Virginia; Washington; West Virginia; Wisconsin; Wyoming., at least 6

PROJ string:

+proj=pipeline

+step +proj=axisswap +order=2,1

+step +proj=unitconvert +xy_in=deg +xy_out=rad

+step +inv +proj=vgridshift +grids=us_noaa_g2018u0.tif +multiplier=1

+step +proj=unitconvert +xy_in=rad +xy_out=deg

+step +proj=axisswap +order=2,1

Grid us_noaa_g2018u0.tif needed but not found on the system. Can be obtained at https://cdn.proj.org/us_noaa_g2018u0.tif

- Alternatively you could also have manually looked for this file in the CDN website:

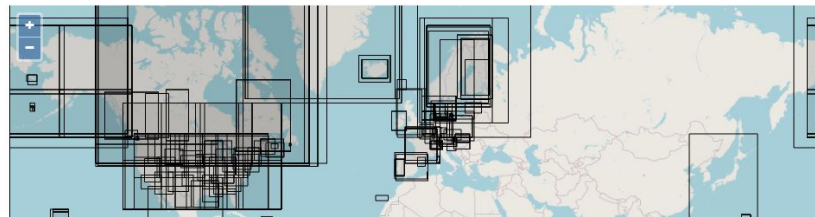
Content

Types: ☒ Horizontal shift grids ☒ Geoid models ☒ Vertical shifts ☒ Velocity grids ☒ Deformation models ☒ Other datasets

Agencies: ☒ ar_ign ☒ at_bev ☒ au_ga ☒ au_icsm ☒ be_ign ☒ br_ibge ☒ ca_nrc ☒ ca_que_mern ☒ ch_swisstopo ☒ cz_cuzk ☒ de_adv ☒ de_bkg ☒ de_geon ☒ de_hvbg ☒ de_lgl_bw ☒ de_lgvl_saarland ☒ eur_nkg ☒ fi_nls ☒ fr_ign ☒ hu_bme ☒ is_lmi ☒ jp_gsi ☒ lv_lgia ☒ mx_inegi ☒ nc_ditt ☒ nl_nsgi ☒ no_kv ☒ nz_linz ☒ pl_gugik ☒ pt_dgt ☒ se_lantmateriet ☒ si_gurs ☒ sk_gku ☒ uk_os ☒ us_nga

Select all

Deselect all



[us_noaa_geoid03_conus.tif](#): USA - Conterminous, US National C (EPSG:4269) to NAVD88 height (EPSG:5703)

geoid_undulation: -50.536 metre

[us_noaa_g2018u0.tif](#): USA - Conterminous, US National Oceano (EPSG:6319) to NAVD88 height (EPSG:5703)

geoid_undulation: -50.206 metre

2. Use projinfo to help find the NAD83 to ITRF ellipsoidal transformation

- Note that you enter the respective EPSG codes as input and output (epsg.org/home)
- Note that the projinfo gives you the proj step needed for the conversion (highlighted)
- Note that in this case it is a mathematical conversion (not geotiffs):

```
!projinfo -s EPSG:6318 -t EPSG:7912 --spatial-test intersects --hide-ballpark -o PROJ
```

Candidate operations found: 1

Operation No. 1:

unknown id, Conversion from NAD83(2011) (geog2D) to NAD83(2011) (geocentric) + Inverse of
14 (geog3D), 0 m, Puerto Rico - onshore and offshore. United States (USA) onshore and of
Delaware; Florida; Georgia; Idaho; Illinois; Indiana; Iowa; Kansas; Kentucky; Louisiana;
ntana; Nebraska; Nevada; New Hampshire; New Jersey; New Mexico; New York; North Carolina
ina; South Dakota; Tennessee; Texas; Utah; Vermont; Virginia; Washington; West Virginia;

PROJ string:

+proj=pipeline

+step +proj=axisswap +order=2,1

+step +proj=unitconvert +xy_in=deg +xy_out=rad

+step +proj=cart +ellps=GRS80

+step +inv +proj=helmert +x=1.0053 +y=-1.9092 +z=-0.5416 +rx=0.0267814

+ry=-0.0004203 +rz=0.0109321 +s=0.00037 +dx=0.0008 +dy=-0.0006

+dz=-0.0014 +drx=6.67e-05 +dry=-0.0007574 +drz=-5.13e-05 +ds=-7e-05

+t_epoch=2010 +convention=coordinate_frame

+step +inv +proj=cart +ellps=GRS80

+step +proj=unitconvert +xy_in=rad +z_in=m +xy_out=deg +z_out=m

+step +proj=axisswap +order=2,1

3. Create a Proj string for the xgeoid20b/ITRF14

- Note that the experimental xgeoid20b is not on CDN, and thus `projinfo` cannot help us.
- However, we know from step 1 that all we need is the datum/ellipsoid geotiff, which in the case (xgeoid20b/ITRF2014) we get directly from NOAA ([xgeoid20b](#)).
- Thus, we can just download and call this file using the same proj command from step1:

```
+step +proj=vgridshift +grids=C:/Users/Felicio.Cassalho/Downloads/xGEOID20B.tif +multiplier=-1
```

4. Concatenate all the Proj strings to build the Proj pipeline:

```
nad832011_navd88geoid18_to_itrf14_2010_xgeoid20b = ""+proj=pipeline
+step +proj=axisswap +order=2,1
+step +proj=unitconvert +xy_in=deg +xy_out=rad
+step +proj=vgridshift +grids=us_noaa_g2018u0.tif +multiplier=1
+step +proj=cart +ellps=GRS80
+step +inv +proj=helmert +x=1.0053 +y=-1.9092 +z=-0.5416 +rx=0.0267814
+ry=-0.0004203 +rz=0.0109321 +s=0.00037 +dx=0.0008 +dy=-0.0006
+dz=-0.0014 +drx=6.67e-05 +dry=-0.0007574 +drz=-5.13e-05 +ds=-7e-05
+t_epoch=2010 +convention=coordinate_frame
+step +inv +proj=cart +ellps=GRS80
+step +proj=vgridshift +grids=C:/Users/Felicio.Cassalho/Downloads/xGEOID20B.tif +multiplier=-1
+step +proj=unitconvert +xy_in=rad +xy_out=deg
+step +proj=axisswap +order=2,1""
```

5. Make the transformation using pyproj:

```
t_nad832011_navd88geoid18_to_itrf14_2010_xgeoid20b = pyproj.Transformer.from_pipeline(nad832011_navd88geoid18_to_itrf14_2010_xgeoid20b).transform
```

```
t_nad832011_navd88geoid18_to_itrf14_2010_xgeoid20b(38.88950026013818, -77.03529973563386, 0, 2010.0)
```

```
(38.889508621337995, -77.035304482066, -0.36200872191263045, 2010.0)
```