How to use *map2loop* with your own data

In order to deconstruct a map with *map2loop*, we need to undertake the following steps:

1. Ensure that we have sufficient correctly formatted data, with sufficient information to allow the calculations to work.
2. Once we have these data, we then need to define a set of parameters that inform *map2loop* about this data, and the details of the calculations to undertaken. Use the notebook **Utility 1 - Config file generator.ipynb** to make this slightly less painful.
3. Finally, once we have the data and *map2loop* control parameters sorted, we are ready to create a small python script to test the system.

1) Minimum *map2loop* data requirements:

1.1 Vector Geospatial File Data Formats:

'DXF': 'raw', 'CSV': 'raw', 'OpenFileGDB': 'r', 'ESRIJSON': 'r', 'ESRI Shapefile': 'raw', 'GeoJSON': 'rw', 'GeoJSONSeq': 'rw', 'GPKG': 'rw', 'GML': 'raw', 'MapInfo File': 'raw'

r=read, a=append, w=write

- geology polygons with stratigraphic code and rock type info (required)
- fault polylines (required)
- bed dips as points in dip, dip direction (required)
- mineral deposit layer (optional)
- fold axial trace layer (optional)

1.2 Geology Polygons (or Multipolygons):

- no gaps or overlaps between polygons, nodes from neighbouring polygons coincide (we have code to fix errors when the mismatch is smaller than the minimum node spacing).
- Stratigraphic Coherency: - Ideally the map should consist of polygons which all have the same stratigraphic heirarchical level, e.g. all formations, all groups, all members etc. This is often not the case, and it makes unravelling the stratigraphy more complex, as the parent and child may appear in the same map, so the age sorting needed to build the model becomes ambiguous. National or state-level stratigraphies, and even stratigraphies described on map sheet legends are by definition simplifications of the the local system extracted by *map2loop*. *map2loop* uses the local stratigraphy first and then uses the regional stratigraphy (if available) as a guide to reduce the uncertainty.
- **Attributes with (data type) and {code} (see section 2.2):**
    - Fine Scale Stratigraphic coding, e.g. formation name (str) {'c'}
    - Coarser Scale stratigraphic coding, usch as its parent, e.g. group (str) {'g'}
    - Alternate Coarser Scale stratigraphic coding, e.g. group (str) {'g2'}
    - Relative or absolute maximum age of Fine Scale Stratigraphic Code (float) {'max'}
    - Relative or absolute minimum age of Fine Scale Stratigraphic Code (float) {'min'}
    - Litho code to help determine if rock is intrusive (str) {'ds'}
    - Litho Code to help detemrine if this is a sill-like body (str) {'sill'}
    - Unique ID of polygon (str) {'o'}

1.3 Fault Polylines:

- single polyline per fault (no multipolylines)
- nodes on faulted boundaries coincident with geology polygon nodes
- **Attributes:**
    - Text that identifies polyline as a fault (str) {'fault'}
    - Dip of fault (str or float) {'fdip'}
    - Dip Direction of Fault (str or float) {'fdipdir'}

1.4 Bedding Points:

- single points (no multipoints)

Attributes:

- Dip (float) {'d'}
- Dip Direction or Strike (float) {'dd'}
- Code to show what convention is used: Strike RHR, or DD at the moment (str) {'otype'}
- Code to show what type of foliation: Bedding, S1 etc. (str) {'sf'}
- Code to say this unit is overturned (str) {'bo'}

1.5 Mineral Deposits: (Optional)

- single points (no multipoints)

Attributes:

- Site Code (str) {'msc'}
- Name of deposit (str) {'msn'}
- Type of feature: open pit, occurence, abandoned etc., (str) {'mst'}
- Code to show what main commodity: Fe, Iron, etc. (str) {'mtc'}
- Code to show what main commodity class: Industrial, Metal, etc. (str) {'mcom'}

1.6 Fold Axial Trace Polylines: (Optional)

- single polyline per trace (no multipolylines)

Attributes:

- Text that identifies polyline as a fold axial trace (str) {'feature'}
- Code that defines fold as syncline (str) {'syn'}

2) Defining *map2loop* parameters:

There are four types of parameters we need to define in order to use *map2loop*:

- **2.1 The paths or URLs** that tell *map2loop* where the information layers (GIS files or online sources) are stored. These are passed to the *map2loop* update_config() method.
- **2.2 The names of fields and some text flags** that tell *map2loop* where specific information can be retrieved from these layers. These are stored in an hjson format text file the path of which is passed to the *map2loop* Project() method.
- **2.3 A data output path, bounding box and Coordinate Reference System** information to define the extent of the model and where to put it. This is passed to the *map2loop* update_config() method

- **2.4 Parameters that control the specific functioning** of *map2loop*: what to calculate, what decimation factors to apply to the augmented data outputs, what resolution interpolations to use etc. These is passed to the map2looop project.run() method.

2.1 paths

These is passed to the *map2loop* Project() method.

Examples:

local GIS layers: see Example 3 notebook

```
proj=Project(geology_file="source/geology_polygons.shp",
             fault_file="source/fault_polylines.shp",
             structure_file="source/bedding_points.shp",
             mindep_file="source/mindep_points.shp",
             metadata="source/meta.hjson",
             remote=False)
```

where remote=False signifies that local GIS files will be accessed. Paths can be relative or absolute, or even a URL, however for URLs, the components of the shapefile or TAB file have to be zipped up.

remote WFS layers: See Example 1 Notebook

```
proj=Project(geology_file = 'http://geo.loop-gis.org/geoserver/loop/wfs?service=WFS&version=1.0.0&request=GetFeature&typeName=loop:geol_500k&bbox={}&srs=EPSG:28350',
           fault_file='http:// etc.',
           structure_file='http:// etc.',
           mindep_file='http:// etc.,
           metadata="http://anyurl.org/mydata.hjson',
           remote=True)
```

where remote=True signifies that WFS-served data will be accessed.

2.2 Layer field codes:

You will need to create or modify an *hjson* format file that provides the names of fields and some text flags that tell *map2loop* where and what specific information can be retrieved from these layers. These are stored in an hjson format text file the path of which is passed to the *map2loop* Project() method. The easiest way to get started is to use a jupyter notebook allows you to reduce errors by providing a primitive GUI for creating an *hjson* config file and associated python script, named: **Utility 1 - Config file generator.ipynb**. Alternatively if you are brave you can edit the values to the right of the colon in each row of an existing *hjson* file. For example to specify that the field in the geospatial layer that contains bedding dip information is called **MYDIP**, replace the appropriate code in the *hjson* file below with:

"d":"MYDIP",

Some verification is carried out by *map2loop* to ensure the required parameters have been defined. In the following section *field* refers to a field name in a geospatial layer; *text* refers to some text in the contents of a field for a specific geometric object. You shouldn't use the same field for different codes as this may cause problems.

```
{
    # Orientations----------------------------
    "d": "DIP",  # field that contains dip information
    "dd": "DIP_DIR",  # field that contains dip direction information
    "sf": "FEATURE",  # field that contains information on type of structure
    # text to search for in field defined by sf code to show that this is a bedding measurement
    "bedding": "Bed",
    # flag to determine measurement convention (currently "strike" or "dip direction")
```

```
    "otype": "dip direction",
    "bo": "TYPE",  # field that contains type of foliation
    # text to search for in field defined by bo code to show that this is an overturned bedding measurement
```

```
    "btype": "overturned",
    # Stratigraphy----------------------------
    "g": "GROUP_",  # field that contains coarser stratigraphic coding
    # field that contains alternate coarser stratigraphic coding if "g" is blank
    "g2": "SUPERSUITE",
    "c": "UNITNAME",  # field that contains finer stratigraphic coding
    "ds": "DESCRIPTN",  # field that contains information about lithology
    # field that contains alternate stratigraphic coding (not used??)
    "u": "CODE",
    "r1": "ROCKTYPE1",  # field that contains  extra lithology information
    "r2": "ROCKTYPE2",  # field that contains even more lithology information
    "sill": "sill",  # text to search for in field defined by ds code to show that this is a sill
    # text to search for in field defined by r1 code to show that this is an intrusion
    "intrusive": "intrusive", # text to search for in field defined by ds code to show that this is an volcanic (not intrusion) "volcanic": "volcanic",
    # Mineral Deposits----------------------------
    "msc": "SITE_CODE",  # field that contains site code of deposit
    "msn": "SHORT_NAME",  # field that contains short name of deposit
    "mst": "SITE_TYPE_",  # field that contains site type of deposit
    "mtc": "TARGET_COM",  # field that contains target commodity of deposit
    "mscm": "SITE_COMMO",  # field that contains site commodity of deposit
    "mcom": "COMMODITY_",  # field that contains commodity group of deposit
    # text to search for in field defined by mst code that shows site to ignore
    "minf": "Infrastructure",
    # Timing----------------------------
    "min": "MIN_AGE_MA",  # field that contains minimum age of unit defined by ccode
    "max": "MAX_AGE_MA",  # field that contains maximum age of unit defined by ccode
    #faults and folds----------------------------
    "f": "FEATURE",  # field that contains information on type of structure
    # text to search for in field defined by f code to show that this is a fault
    "fault": "Fault",
    "ff": "FEATURE",  # field that contains information on type of structure
    # text to search for in field defined by f code to show that this is a fold axial trace
    "fold": "Fold axial trace",
    "fdip": "DIP",               # field for numeric fault dip value
    # text to search for in field defined by fdip to show that this has no known dip
    "fdipnull": "0",
    "fdipdir": "DIP_DIR",        # field for text fault dip direction value
    # flag for text fault dip direction type num e.g. 045 or alpha e.g. southeast
    "fdipdir_flag": "alpha",
    "fdipest": "DIP_EST",        # field for text fault dip estimate value
    # text to search for in field defined by fdipest to give fault dip estimate in increasing steepness
    "fdipest_vals": "gentle,moderate,steep",
    # field that contains information on name of fault (not used??)
    "n": "NAME",
    "t": "TYPE",  # field that contains information on type of fold
    # text to search for in field defined by t to show that this is a syncline
    "syn": "syncline",
    # ids----------------------------
    "o": "OBJECTID",  # field that contains unique id of geometry object
    "gi": "GEOPNT_ID",  # field that contains unique id of structure point
    "deposit_dist": 500
}
```

2.3 ROI, Projection, output paths

A data output path which points to a new or existing directory (an ew directory will be created if needed), bounding box and Coordinate Reference System information to define the extent of the model. This is be passed to the *map2loop* update_config() method

```
proj.update_config(
                    out_dir='./model-test',
                    overwrite='overwrite',
                    bbox_3d={
                         "minx": 500000,
                         "miny": 7490000,
                         "maxx": 545000,
                         "maxy": 7520000,
                         "base": -4800,
                         "top": 1200,
                     },
                    proj_crs={'init': 'EPSG:28350'},
                    quiet='none'
                  )
```

- where bbox coordinates are in CRs defined by proj_crs
- where overwite can be 'overwrite', 'true'

- where quiet controls whether we allow or block print statements and matplotlib figures. Use 'none' to quiet nothing, 'all' to quiet everything, 'no-figures' to disable plots and allow text output. Defaults to 'none'

2.4 Calculation control parameters

These control the specific functionality of *map2loop*: what to calculate, what decimation factors to apply to the augmented data outputs, what resolution interpolations to use etc. These are passed to the *map2looop* project run() method:

proj.run()

This method performs the data processing steps of the *map2loop* workflow, and can be modified by including the following parameters [defaults](data type):

- **aus**: Indicates if area is in Australia for using ASUD. [True] (bool)
- **close_dip**: Dip to assign to all new orientations. [-999] (int)
- **contact_decimate**: Save every nth contact data point. [5] (int)
- **contact_dip**: [-999] (int)
- **contact_orientation_decimate**: Save every nth contact orientation point. [5] (int)
- **deposits**: Mineral deposit names for focused topology extraction. ["Fe,Cu,Au,NONE"] (str)
- **dist_buffer**: Buffer for processing plutons. [10] (int)
- **dtb**: Path to depth to basement grid. [''] (str)
- **fat_step**: How much to step out normal to the fold axial trace. [750] (int)
- **fault_decimate**: Save every nth fault data point. [5] (int)
- **fault_dip**: default fault dip [90] (int)
- **fold_decimate**: Save every nth fold data point. [5] (int)
- **interpolation_scheme**: What interpolation method to use of scipy_rbf (radial basis) or scipy_idw (inverse distance weighted). ['scipy_rbf'] (str)
- **interpolation_spacing**: Interpolation grid spacing in meters. [500] (int)
- **intrusion_mode**: 1 to exclude all intrusions from basal contacts, [0] to only exclude sills. [0] (int)
- **max_thickness_allowed**: when estimating local formation thickness [10000] (int)
- **min_fault_length**: Min fault length to be considered. [5000] (int)
- **misorientation**: [30] Maximum misorientation in pole to great circle of bedding between groups to be considered part of same supergroup (int)
- **null_scheme**: How null values present in the dtb. ['null'] (str)
- **orientation_decimate**: Save every nth orientation data point. [0] type int
- **pluton_dip**: default pluton dip [45] (int)
- **pluton_form**: Possible forms from domes, saucers or pendant. ['domes'] (str)
- **thickness_buffer**: How far away to look for next highest unit when calculating formation thickness [5000] (int)
- **use_fat**: Use fold axial trace info to add near-axis bedding info [True] (bool)
- **use_interpolations**: Use all interpolated dips for modelling [True] (bool)

3) Example minimum code:

An example minimum code to run *map2loop* with mostly default settings might look like this (and see the notebook **Example 3 - Local Source Data.ipynb**):

```
from map2loop.project import Project

proj=Project(geology_file="source/geology_polygons.shp",
             fault_file="source/fault_polylines.shp",
             structure_file="source/bedding_points.shp",
             mindep_file="source/mindep_points.shp",
             metadata="source/meta.hjson"
             )

proj.update_config(
                    out_dir='./model-test',
                    bbox_3d={
                        "minx": mbbox.total_bounds[0], #500000,
                        "miny": mbbox.total_bounds[1], #7490000,
                        "maxx": mbbox.total_bounds[2], #545000,
                        "maxy": mbbox.total_bounds[3], #7520000,
                        "base": -4800,
                        "top": 1200,
                        "local': True
                    },
                    proj_crs={'init': 'EPSG:28350'}
                    )

proj.run()
```