README.md - Grip

Tools for processing, inversion, and interpretation of Airborne ElectroMagnetics (AEM)

This is the public repository for the Airborne Electromagnetic Inversion toolbox "AEMpyX", which was originally developed at DIAS starting with the project "Spatially constrained Bayesian inversion of frequency- and time-domain electromagnetic data from the Tellus projects" (2015-sc-004), followed by "RAFTA: Resolution Analyses for Frequency- and Time-Domain Airborne Electromagnetic Data of the Irish Tellus Programme" (2020-sc-049), both funded by the Geological Survey of Ireland GSI. It is distributed under the GNU GENERAL PUBLIC LICENSE Version 3. Currently it is expanded to include more algorithms (in particular for uncertainty quantification, UQ), platforms, and a branch for making it useful for magnetotelluric (MT) modelling and inversion.

AEMpyX currently works fully under linux operating systems, but mostly also under windows (here short for Windows 10). There are of course changes in the installation procedures, as mentioned below.

Under linux, get your working copy directly via git from the command line. In windows, git functionality is available, once Anaconda is installeded (see below). There, open the powershell terminal and clone the repository with the same line as in linux:

git clone https://github.com/RAFTA-AIRBORNE/AEMpyX.git

The created local repository AEMpyX contains the following subdirectories:

- **environment** Contains conda environment description files, and some useful helper files for working within the conda environment. For linux and windows you can find special versions in the repective directories. The current AEMpyX environment contains some packages which are not strictly necessary for running aempy, but useful for related geoscientific work.

- **aempy/tutorial** Contains python scripts and jupyter notebooks for the most important scripts, explaining typical AEM preprocessing, visualization, one-dimensional inversion, and uncertainty quantification workflows, using the toolbox.

- **aempy/modules** Contains the modules *aesys.py*, *prep.py*, *post.py*, *viz.py*, *inverse.py*, *alg.py*, and *util.py*, which are called from the Python scripts in **aempy/scripts** and **aempy/tutorial**, accomplishing different tasks of AEM inversion. It also contains the *core1d.so* (or *core1d.pyd* whem using windows) module, once compiled from the sources in **aempy/core1d**.

- **aempy/core1d** This directory co Fortran 90 source code for the computational core run by the Python toolbox. The numerics is derived from the AMIRA/CSIRO AirBeo software. Currently it contains working wrappers for the two systems used in Tellus: GTK4, and CGG GENESIS, with TEMPEST and GEOTEM under development. This directory also contains makefiles for both operating systems, named *Makefile_linux* or *Makefile_windows*, respectively.

- **aempy/scripts** Contains a collection of scripts for processing, visualization, and other tasks releted to one-dimensional inversion of AEM data;. Also included is a workflow for static shift correction of MT observations (work in progress). These scripts come from different stages of the RAFTA project, solving specific problems. Thus they may noy be up to date. However, they represent a useful source for cannibalizing.

- **aempy/util** Contains useful helper shell scripts etc.

- **aempy/report** This is the place where publications, reports/manuals are developed.

- **aempy/info** Documentation for the toolbox, python (including the most important extensions, numpy, scipy, and matplotlib), and other tools.

The scripts and jupyter notebooks are available in the subdirectory AEMpyX/aempy.

This version will run under Python 3.9+ (3.11 bring the current development platform). For installing the python environment in any Linux environment (e.g. Ubuntu, SuSE), you need to do the following:

(1) Download the latest Anaconda or Miniconda version (https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html), and install by running the downloaded bash script with:

bash Miniconda3-latest-Linux-x86_64.sh

For windows just execute the downloaded executable, *Miniconda3-latest-Windows-x86_64.exe*. In order to make updates secure and avoid inconsistencies, copy .condarc from AEMpyX/environment to your home directory. As the Miniconda installer is not updated very frequently, it is useful to run the following within Anaconda:

conda update conda

conda update --all

Do this regularly to keep everything consistent!

(2) Create an appropriate conda environment (including the necessary prerequisites) from the files AEMpyX.yml or AEMpyX.txt found in the AEMpyX/environment directories by entering

conda env create -f AEMpyX.yml

or

conda create --name AEMpyX --file AEMpyX.txt

in the command window (powershell under windows).

This will set up a Python 3.11 environment with all dependencies for aempy. Don't forget to update also the used environment regularly, using *conda update --name AEMpyX --all*! There is a replacement for *conda*, called *mamba* (see https://github.com/mamba-org/mamba), which is not only considerably faster, but also better in keeping the environments consistent. It can be installed via *conda* (i. e., *conda install mamba*), and has practically the same syntax as the original package manager.

(3) Activate this environment by:

conda activate AEMpyX

(4) Within this envionment you now need to compile the aempy core, which is written in Fortran 90, and thus needs. For this purpose, *f2py* (part of *numpy*), and the required compilers have been included in the EM environments. To compile and install the numerical core, go to the *core1d* directory, and enter

make -f Makefile_linux or *make -f Makefile_windows*

respectively. If this is succesful, a dynamical library, *core1d.so* (*core1d.pyd* under windows), should be in the *modules* subdirectory.

(5) In order to reproduce identical behavior of matplotlib, you should copy the included *matplotlibrc* file to the appropriate directory. Under Linux (Ubuntu), this should be : *$HOME/.config/matplotlib/matplotlibrc*. Pertinent changes should be made there, or have to be made within the scripts/modules using the *mpl.rcParams[name]=value* mechanism.

(6) For running aempy scripts, we have defined two environmental variable, *AEMPYX_ROOT* and *AEMPYX_DATA*. These point to the place where AEMpyX is installed, and where you keep your AEM data, respectively. Keeping to this scheme makes life much easier if more than one person work on the tools.

In linux you can set them in your *.bashrc* file. Example:

export AEMPYX_ROOT='${HOME}/AEMpyX/'

export AEMPYX_DATA='${HOME}/AEM_Data/Tellus/data/'

Under windows, you should use the system settings dialogue to do so.

(7) Finally, the remaining open source toolboxes you want to use need to be installed, either via the anaconda framework, or the *pip* command.

(8) Once in the activated conda environment *AEMpyX*, there are several ways to start python scripts or jupyter notebooks (https://jupyter.org/).

For getting started with python **scripts** I suggest to use the *spyder* IDE (https://www.spyder-ide.org/), which is already installed within *AEMpyX*. It has been developed for easy development of python software, including visualisation with *matplotlib* and derived packages. Current versions (5.X) also allow to work with other languages as JULIA or R languages, or with jupyter notebooks by installing the appropriate plugin.

However, as we have defined the environmental variables in step (6), the scripts can be run from anywhere in the system, either from the activated AEMpyX environment, or using *conda run* (see, *https://docs.conda.io/projects/conda/en/latest/commands/run.html*) from an initialized conda (i.e., base environment). Running python scripts directly from the command line can be done in different ways:

python3 -u mypythonscript.py > output.log,

if you are in AEMpyX. If the magic first line *#!/usr/bin/env python3* exists in the skript, it can be called as

mypythonscript.py > output.log.

If anywhere in an conda environment, use

conda run -n AEMpyX mypythonscript.py

The usual way to work with **notebooks** is with your favorite web browser, or specialized software as *jupyterlab*. If using your browser (as set in your system as default) you can simply use the classical interface

jupyter notebook mynotebook.ipynb

or the new one

jupyter lab mynotebook.ipynb

Both calls will open a new browser window, in which you can edit and run the notebook. The *jupiterlab* is the future interface for notebooks, and there are many options not available with the classical call (https://jupyterlab.readthedocs.io). Remote clusters often offer a specialized server (JupyterHub) to develop and run notebooks.

Enjoy, read the docs, but please keep in mind that this is an experimental software, and may contain errors. Use at your own risk! However, we will frequently update the repository correcting bugs, and adding additional functionality.

D. Kiyan & V. Rath

October 26, 2023