
Data Extension and Integration Documentation

Kirk Thoning

Sep 02, 2021

CONTENTS

1	Introduction	1
1.1	Method	1
1.2	Creating an Extended Record	1
1.3	Example	2
2	Software	7
2.1	Requirements:	7
2.2	Dependencies	7
2.3	Installation	7
2.4	Programs	7
2.5	Changelog	8
3	Quick Start	11
3.1	Using the driver program	11
3.2	Using bootstrap runs with driver program	12
3.3	Adding bootstrap runs to existing mbl results	12
3.4	Other options	12
3.5	Summary bootstrap results	13
3.6	Offline Mode	13
4	Initialization file	15
4.1	Parameters	15
4.2	Curve fit parameters	16
4.3	MBL site designation	16
5	DEI Driver program usage	17
6	DEI Bootstrap summary driver program usage	19
7	Uncertainty calculations using a bootstrap method	21
7.1	Network Uncertainty	21
7.2	Atmospheric Uncertainty	22
7.3	Measurement Bias Uncertainty	23
7.4	Analysis Uncertainty	24
8	Results	27
8.1	Directory Structure	27
8.2	Output Files	27
8.3	Output files from bootstrap summary program (dei_bs_driver.py)	28
8.4	Output files from total bootstrap uncertainty program (dei_effective_unc.py)	29
8.5	Output files from zonal bootstrap uncertainty program (dei_unc.py)	29

9	Latitude Zone Definitions	31
10	Latitude Gradients	33
11	dei class	35
11.1	Methodology:	35
11.2	dei Objects	35
11.3	Methods	36
12	Differences from IDL code:	37
13	DEI Graphical Interface	39
13.1	Saved Configurations	39
13.2	Initialization Section	39
13.3	Data Section	41
13.4	Bootstrap Section	41
13.5	Message Section	41
13.6	Start Button	41
14	Indices and tables	43
	Python Module Index	45
	Index	47

INTRODUCTION

This documentation describes the python version of the data extension software. It does the same steps that the IDL version of the software does, but with some changes to the directory names for the output.

The software consists of a python class 'dei', several helper modules, and a python driver program which can be used to simplify setting options and running the software.

A powerpoint file of Ken Masarie's presentation of the IDL dei software from [November 2015](#) is available.

A pdf file of Kirk Thoning's presentation of the python dei software from [November 2017](#) is available.

1.1 Method

Data extension attempts to address the issue of temporal discontinuities in and among measurement records by characterizing what we have learned from actual observations and extending this knowledge in time beyond the observations themselves. Thus far, "knowledge" has been defined as average behavior, i.e., average seasonal cycle and trends, and how, on average, a site differs from other sites that are nearby in latitude. It is this average behavior that is extended beyond the measurements.

The data extension methodology is described in the paper by [Masarie and Tans \(1995\)](#).

More details of the method are described at the [Marine Boundary Layer Reference web page](#) web page.

The method of fitting curves to the data is [described here](#) using the python `ccgfit` module.

1.2 Creating an Extended Record

The creation of an extended record and the final marine boundary layer surface grid is composed of these steps:

- Fit smooth curves to the data for each sampling site and identify gaps in the data.
- Using only sites designated as mbl sites, calculate latitude gradients at each timestep.
- Create a 'reference' time series at the latitude for each site using the values from the latitude gradients at each time step.
- Compute the difference between the reference time series and the smooth time series data.
- Fit a 2 harmonic function to the difference data. Determine the values of this function at the times where there are gaps in the data.
- Create extended data by filling gaps in the smoothed time series with values from the combined reference + function time series, using a smooth transition from non-gap to gap periods.

- Again using only mbl sites, calculate the latitude gradients at each timestep with the extended data. This is the final mbl reference surface.

1.3 Example

As an example of the data extension technique, we will use the CO₂ measurements from Azores (AZR) to show the steps involved. We will create an extended record for AZR spanning the period January 1, 1979 to January 1, 2016. In the figure below, a smooth curve (orange line) is fitted to the actual observations from weekly air samples (blue circles). Because we are trying to create an uninterrupted synchronized time series, we try to extract values from the smooth curve at 48 equal time steps within a year (7.6 day interval) from 1979 to 2016. If there were no gaps in the observational record, we would be done. However, in the AZR CO₂ record, measurement gaps do exist, with the most significant occurring in 1989-1990, 1992-1994 and 2010-2011.

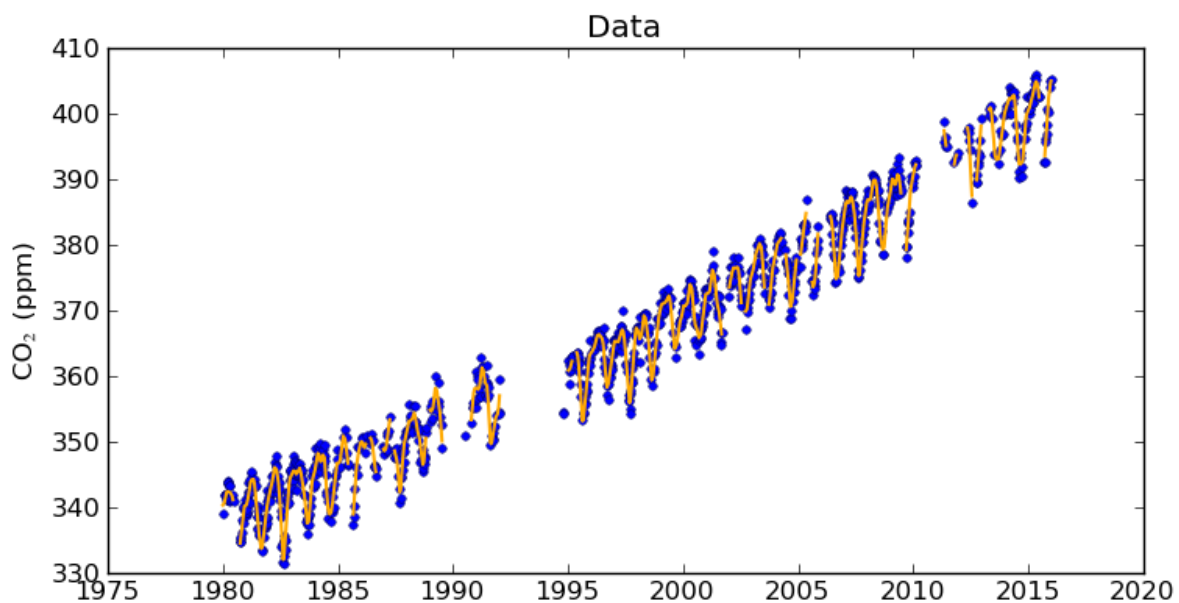


Figure 1. Time series of the CO₂ data from Azores (blue circles), and a smooth curve applied to the data (red line).

Next we construct a matrix of CO₂ mixing ratios as function of time and sine latitude. The matrix is derived from all available observations from locations sampling large well-mixed marine boundary layer (MBL) air. We fit a smooth curve to each site designated as an MBL site. Then we extract values from each curve at “weekly” intervals for the period 1979 to 2016 but only where measurements exist. We step through each week to construct a weekly distribution of CO₂ mixing ratios as a function of sine latitude using all available MBL values. We *fit a curve to this north-south distribution* and extract values from this curve at 0.05 sine latitude intervals. Lastly, we “glue” these weekly north-south values together to construct a preliminary MBL reference matrix.

We can now extract a reference time series from the preliminary MBL matrix at the latitude of AZR (Figure 2 below).

Figure 2. Reference time series from the preliminary MBL matrix at the same latitude as AZR.

To see how observations at AZR differ from this reference, we subtract the smooth values from Figure 1 (for periods where AZR observations actually exist) from the MBL reference shown in Figure 2. This difference distribution is shown below in Figure 2. We then apply a curve fit to this difference distribution, which tells us how, on average, measurements at AZR differ from the MBL reference. We extract values from this curve to fill gaps in the difference distribution. Remember, these gaps in the difference distribution exist because of actual gaps in the AZR observations. These filled in values are shown in red in Figure 3.

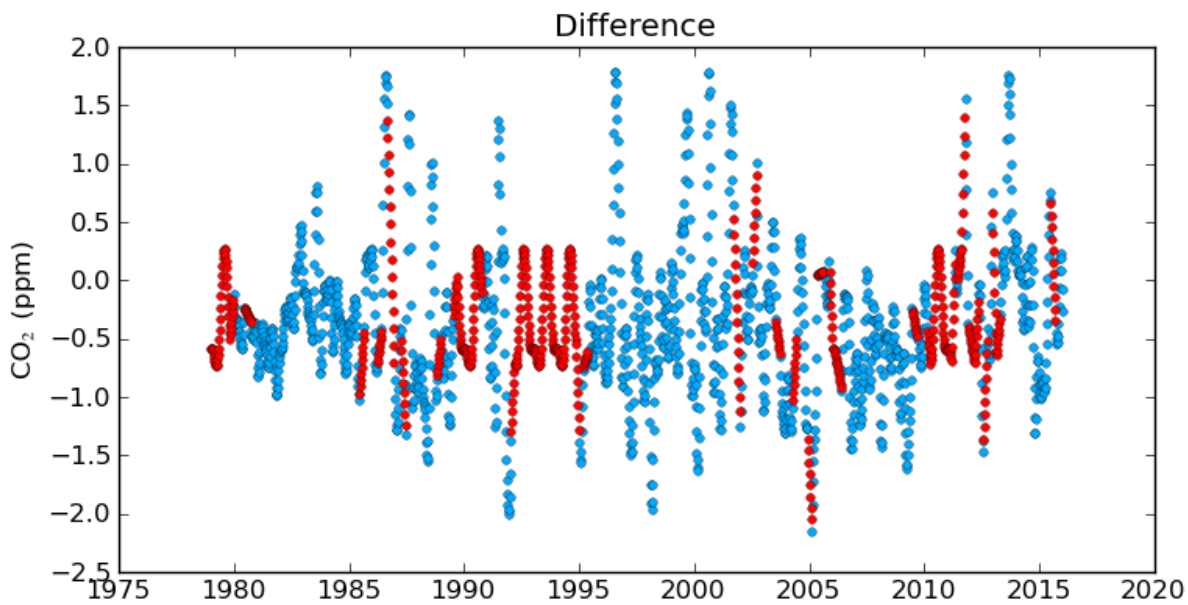
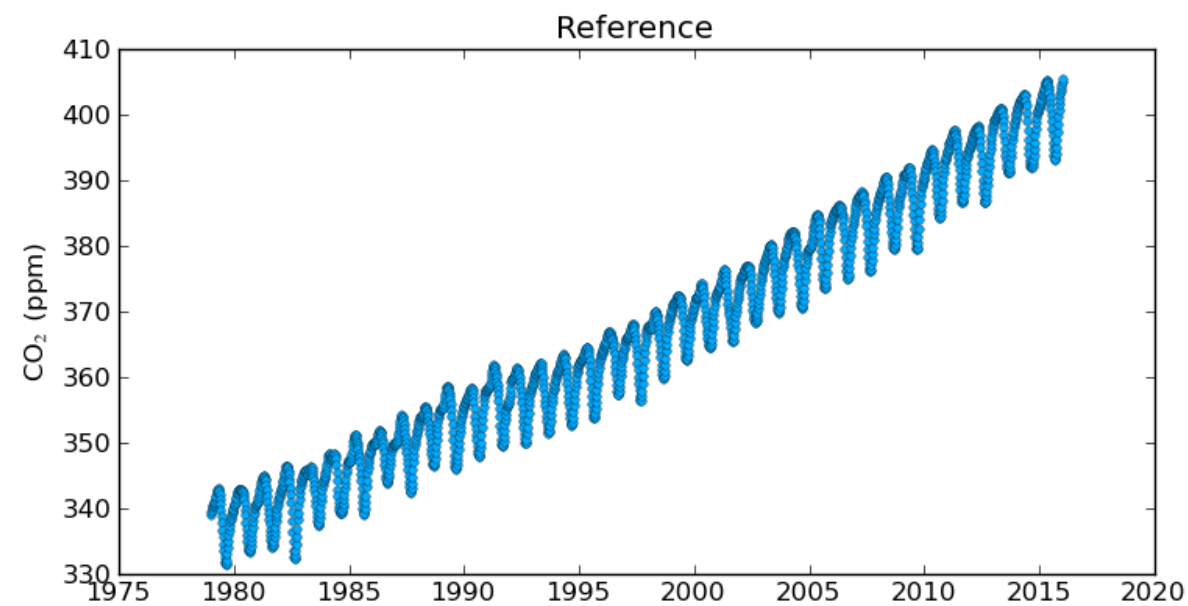


Figure 3. Differences between the smooth curve in Figure 1 and the Reference time series in Figure 2. The red points are data from the reference time series which will be used to fill gaps in the AZR smooth curve.

Finally, we construct the extended record at AZR by adding the reference (Figure 2) to the difference distribution (Figure 3) at the time where there are gaps in the AZR record (Figure 1). The resulting extended record is shown in Figure 4. Note that the blue symbols in Figure 4 are exactly the smooth values extracted at weekly time steps from the AZR record where AZR observations exist (Figure 1). The red values are derived interpolated and extrapolated values for periods where observations at AZR do not exist.

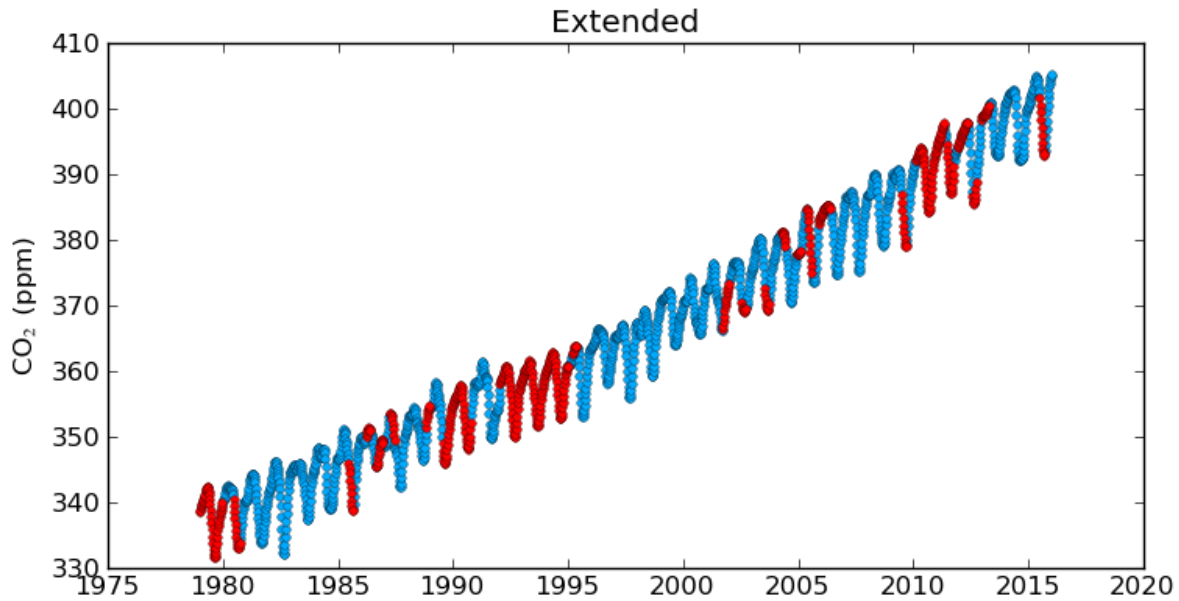
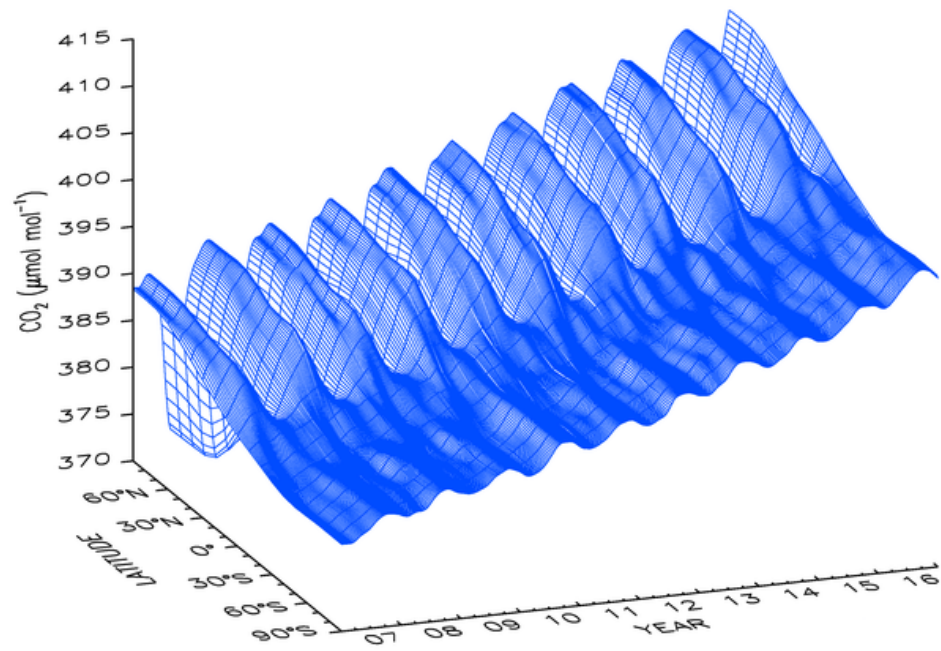


Figure 4. The extended record at AZR. The blue points are the smoothed values from Figure 1, and the red points are the interpolated and extrapolated data where there are gaps in the AZR record.

Once an extended record has been calculated for every mbl site, we then construct a final MBL reference surface by fitting curves to the north-south distribution of mbl sites at every time step, and extracting values at 0.05 sine latitude intervals. Each of these north-south distributions are pieced together at each time step to create a final MBL reference matrix, show below in Figure 5.

Figure 5. CO2 MBL Reference Matrix. Only the last 10 years are shown.



SOFTWARE

2.1 Requirements:

Python 2.6, 2.7, 3.5 (probably will run under lower versions of python 3, but has not been tested)

Should run on any GML Linux server or desktop computer. Should also run on Mac's if all the python dependencies are installed. Has not been tried under Windows.

2.2 Dependencies

The following add-on python packages are required:

- `numpy`
- `scipy`
- `MySQL-python` (for python 2) or `mysqlclient` or `pymysql` (for python 3)
- `dateutil`

2.3 Installation

The python software for making the dei calculations is located in `/nfs/ccg/dei/ext/src/python`. If you run the software from that directory, you do not need to do any additional installation.

If you want to run a standalone installation without a network connection to the `/nfs/ccg` file share, then download the latest zip package of the software:

https://omi.cmdl.noaa.gov/doc/dei/_static/dei_0.35.zip (Sep 1, 2021)

2.4 Programs

There are several programs available for running a dei calculation.

- `dei_driver.py`
- `dei_bs_driver.py`
- `dei_unc.py`
- `dei_effective_unc.py`

- `dei_gui.py`

and a couple of python classes:

- `dei.py`
- `dei_bs.py`

2.5 Changelog

Version 0.35 (September 2021)

`dei_driver.py`, `dei.py`

Added an option to `dei_driver.py` called `--autosync`. This will set the `sync2` date to be equal to the current date - 4 months, with the day set to the end of that month. This is to avoid end affects, especially for `ch4`, when the `--anchor` option is used and data is extrapolated to the current date. The `--anchor` option overrides the `--autosync` option, so don't use both options at the same time.

Version 0.34 (January 2021)

`ccg_db.py`

Modified `ccg_db.py` to use `pymysql` package for python 3.

`dei.py`

Included a check for a special case to skip `drp` data if there was no data after 2005.

`ccg_filter.py`

Included checks in `ccg_filter.py` for arrays that were only 1 element long, and handle accordingly.

Version 0.33 (July 2019)

`dei.py`

Modified the routine `_make_zone_ann_ave()` in `dei.py` again because it wasn't correctly removing non-complete years for some circumstances. Also needed to modify `dei_bs.py` in `zonal_unc()` to make corresponding changes to make sure only complete years are used for annual averages and annual increase. The Changes made in v0.31 and v0.32 did not work correctly.

Version 0.32 (November 2018)

`dei_bs.py`

Modified `zonal_unc()` to correctly handle annual average files. Before, the last year was dropped because it was incomplete. Now, with the changes for v0.31, only complete years are available, so no need to drop the last year. This should have been included with v0.31 changes.

Version 0.31 (October 2018)

`dei.py`

Modified the routine `_make_zone_ann_ave` in `dei.py` to use values only for complete years for annual averages and year to year increase of the trend. Previously 'Nan' values would appear in the output files for incomplete years.

Version 0.3 (August 2018)

dei.py

Added zonal data values for additional ccgvu curve components ('ftn', 'tr', 'gr', 'fsc', 'ssc', 'residf', 'residsc') instead of just ('gr', 'fsc'). This required changes to some of the methods.

dei_unc.py

New script to compute combined uncertainties for zonal files from all bootstrap types available.
Creates new zonal uncertainty files in main results directory.

Version 0.21 (March 2018)

ccg_readinit.py, ccg_siteinfo.py

- Minor formatting changes

ccg_data.py

- Added a 'startdate' parameter to ccg_data()

ccg_filter.py

- Improved calculation of monthly and annual means

Version 0.2 (November 2017)

- Formatting changes and fixes to get software to run under python 3
- Add --siteinfo option to **dei_driver.py**

QUICK START

The easiest way to run the data extension software is to use the **dei_driver.py** program, and calling it directly from the command line, or setting up a bash shell script to run the program. The driver program is located in the `/ccg/dei/ext/src/python` directory.

To do a full data extension that includes bootstrap runs, you need to run three programs:

- `dei_driver.py` - Calculate the mbl surface and perform any bootstrap runs.
- `dei_bs_driver.py` - Summarize the bootstrap runs and create summary files.
- `dei_effective_unc.py` - Create an 'effective' surface uncertainty, which is the sum in quadrature of the uncertainties of all the bootstrap runs.

Warning: Any existing files and sub directories in the results directory are removed before making the dei calculations. This is consistent with the behavior of the IDL version of the code. Be sure to save any results before reusing the same output directory.

3.1 Using the driver program

The simplest way to run the driver program is to just call it from python using the built-in default values

```
$ python /ccg/dei/ext/src/python/dei_driver.py
```

But it is very unlikely the the defaults are what you want to use. **dei_driver.py** has a number of *options* that can be specified. The most important are `--gas`, `--initfile`, `--bsdir`. A more useful way to run the driver program is to use a shell script with the settings, for example, if we have a file called *deiflask.sh* with the following lines

```
#!/bin/bash

gas="ch4"
thisyear=`date +%Y`

initfile="/ccg/dei/ext/${gas}/work/init.${gas}.flask.master.txt"
bsdir="/ccg/dei/ext/${gas}/flask/results.flask.${thisyear}"
python /ccg/dei/ext/src/python/dei_driver.py --initfile=$initfile --bsdir=$bsdir --
↪ gas=$gas
```

We can then make the mbl calculations with the command

```
$ bash deiflask.sh
```

3.2 Using bootstrap runs with driver program

Some of the options for the `dei_driver.py` program are for including bootstrap runs using a built-in uncertainty method to the mbl calculations. To do bootstrap runs, include the `--unctype` and the `--numbs` options. `--unctype` specifies the type of bootstrap run to do, and `--numbs` specifies how many bootstrap runs to perform. For example

```
#!/bin/bash

gas="ch4"
thisyear=`date +%Y`

initfile="/ccg/dei/ext/${gas}/work/init.${gas}.flask.master.txt"
bsdir="/ccg/dei/ext/${gas}/flask/results.flask.${thisyear}"
unctype="atmospheric"
numbs=100
python /ccg/dei/ext/src/python/dei_driver.py --initfile=$initfile --bsdir=$bsdir --
↪ gas=$gas --unctype=$unctype --numbs=$numbs
```

The `--unctype` option must be one of `'atmospheric'`, `'network'`, `'bias'`, or `'analysis'`. If `'bias'` or `'analysis'` is used, then the option `--bias_config` must also be included to specify the file with bias configuration values.

3.3 Adding bootstrap runs to existing mbl results

If an mbl calculation has already been completed, and you decide you want to add bootstrap runs without recalculating the mbl results, or if you want to add additional types of bootstrap runs, use the `--bootstrap_only` option. Including this option will bypass the mbl calculations, and perform only the bootstrap runs.

3.4 Other options

There are a few additional options available when running the `dei_driver` program:

- anchor** Ignore the `sync2` value in the `initfile`, and produce results up to today's date instead.
- quickfilter** Apply quick filtering of outliers of **preliminary data** that are $> \pm 3$ sigma of residuals from smooth curve fits. This option is ignored if the `--inputdir` option is used.
- inputdir=INPUTDIR** Directory containing input data text files. Use these instead of getting data from the database.
- stopdate=STOPDATE** Specify the last date (as decimal date, i.e 2016.25) of input data to use.
- siteinfo=SITEINFO** Specify a file with site information. Use this instead of getting information from the gmd database.

3.5 Summary bootstrap results

The bootstrap runs produce a large number of files. To summarize these files, a second program `dei_bs_driver.py` can be run. This program will go through all of the boot strap runs, and create files that summarize the values produced over all the bootstrap runs. It works best to add another line in your shell script to run this program. It uses the same options that the `dei_driver.py` program uses

```
#!/bin/bash

gas="ch4"
thisyear=`date +%Y`

initfile="/ccg/dei/ext/${gas}/work/init.${gas}.flask.master.txt"
bsdir="/ccg/dei/ext/${gas}/flask/results.flask.${thisyear}"
unctype="atmospheric"
numbs=100
python /ccg/dei/ext/src/python/dei_driver.py --initfile=$initfile --bsdir=$bsdir --
↳gas=$gas --unctype=$unctype --numbs=$numbs
python /ccg/dei/ext/src/python/dei_bs_driver.py --initfile=$initfile --bsdir=$bsdir --
↳gas=$gas --unctype=$unctype --numbs=$numbs
```

The `dei_bs_driver.py` program has 2 additional options

- outdir** Specify a sub directory in the bootstrap results directory in which to put results. If not used, results are placed in the bootstrap results directory.
- npoly** Specify the number of polynomial values to use in the curve fits that are done in the program. Default is 3.

3.6 Offline Mode

Normally the dei software access the input data and site information data from the gmd database. However, you can run the software without needing access to the database by doing the following:

- Grab a *copy of the software* and unzip it.
- Create input data text files containing two columns of data, a decimal date and value. Place all of these in the same directory
- Create an *init file* that will match up with the input data files you created.
- Create a file with site information, which contains 8 columns of data separated by a 'l' character: station code, name, country, latitude, longitude, elevation, intake height, lst to utc offset, comment
- Use the `--inputdir` and `--siteinfo` options to `dei_driver.py`.

INITIALIZATION FILE

The ‘init’ file specifies which sampling sites to use and the curve fitting parameters to use for the mbl calculations. Below is an example of an init file for CO₂. The columns ‘ext’, ‘ftp_event’ and ‘ftp_month’ are optional, are used only in the IDL dei software, and do not need to be in your init file.

Example:

sync1	sync2	rlm	fillgap	mblgap						
1979	2017	2.0	8	48						
site		poly	harm	int	short	long	mbl	ext	ftp_event	ftp_
→month										
abp_01D0		3	4	7	80	667	1	1	1	1
alt_01D0		3	4	7	80	667	1	1	1	1
ams_01D0		3	4	7	80	667	1	1	1	1
amt_01D0		3	4	7	80	667	0	0	0	0
amy_01D0		3	4	7	80	667	0	1	1	1
aoc_01D1		3	4	10	80	667	0	0	1	0
.										
.										
.										

Master initfiles are located in `/ccg/dei/ext/_gas_/work` where `_gas_` is the gas species name, e.g. ‘co2’, ‘ch4’ etc.

4.1 Parameters

- **sync1** - First year to calculate mbl results. Input data can go before this year, but no results will be generated for those years.
- **sync2** - Last year to calculate mbl results. Input data can go after this year, but no results will be generated for those years.
- **rlm** - Minimum length of data record in years required for a site to be included in the mbl calculations.
- **fillgap** - A value in weeks, where gaps in the data larger than this are filled with interpolated values derived from data extension.
- **mblgap** - A value in weeks, where gaps in the data larger than this are filled with a trend and average seasonal cycle from a curve fit to data.

4.2 Curve fit parameters

The columns labeled ‘poly, harm, int, short, and long’ are the standard parameters used in the `ccgcrv smoothing`, and are:

- **poly** - Number of polynomial terms to use in function fit to data.
- **harm** - Number of yearly harmonic terms to use in function fit to data.
- **int** - Sampling interval in days
- **short** - Short term filtering cutoff value in days
- **long** - Long term filtering cutoff value in days

4.3 MBL site designation

The column labeled ‘mbl’ is used to specify if a site is a marine boundary layer site (value of 1) or not (value of 0). Only mbl sites are used when calculating the latitude gradient at each timestep. If a site is not designated as an mbl site, it is not used in determining the latitude gradient.

DEI DRIVER PROGRAM USAGE

Usage: `dei_driver.py` [options]

Create MBL surface data extension files, and optional bootstrap uncertainty runs.

Options:

- | | |
|----------------------------|--|
| -h, --help | show this help message and exit |
| --initfile=INITFILE | Specify initialization file to use. |
| --numbs=NUMBS | Specify number of bootstrap runs. Default is 0. |
| --gas=GAS | Specify gas to use. Default is 'co2'. |
| --anchor | Ignore sync2 value in initfile. Use today's date instead, and extrapolate data for anchor sites if necessary. |
| --autosync | Ignore sync2 value in initfile. Use instead 4 months prior to the current date, and the day set to the end of the month. |
| --quickfilter | Apply quick filtering of outliers > +/- 3 sigma of residuals. |
| --bsdir=BSDIR | Top level directory where to store results. Default is 'dei_results' in current directory. |
| --inputdir=INPUTDIR | Directory containing input data files. Use these instead of data from database. |
| --unctype=UNCTYPE | Specify the type of uncertainty to apply to data when doing bootstrap runs. Must be one of 'atmospheric', 'bias' or 'network'. |
| --stopdate=STOPDATE | Specify the last date (as decimal date, i.e 2016.25) of input data to use. |
| --bootstrap_only | Run bootstrap for unctype only. Do not perform normal mbl calculation. Use this to add a bootstrap run to an existing mbl result. |
| --config=CONFIG | Specify initialization file to use for bias, and analysis bootstrap values. REQUIRED if <code>--unctype=bias</code> or <code>--unctype=analysis</code> |
| --siteinfo=SITEINFO | Specify file with site information. Use this instead of getting information from database. |

DEI BOOTSTRAP SUMMARY DRIVER PROGRAM USAGE

Usage: dei_bs_driver.py [options]

Process MBL bootstrap files.

Options:

- h, --help** show this help message and exit
- initfile=INITFILE** Specify initialization file to use.
- numbs=NUMBS** Specify number of bootstrap runs. Default is 0.
- gas=GAS** Specify gas to use. Default is 'co2'.
- bsdir=BSDIR** Top level directory of bootstrap results. Default is 'dei_results'.
- unctype=UNCTYPE** Specify the type of uncertainty to apply to data when doing bootstrap runs. Must be one of 'atmospheric' or 'bias'.
- npoly=NPOLY** Number of polynomial terms to use in function fit of curve fitting.
- outdir=OUTDIR** Specify output directory to place files. Default is directory specified by --bsdir
- siteinfo=SITEINFO** Specify file with site information. Use this instead of getting information from database.

UNCERTAINTY CALCULATIONS USING A BOOTSTRAP METHOD

There are four types of uncertainty calculation that can be done:

Network Vary the network sites

Atmospheric Add noise to the data based on variability of the data around a smooth curve.

Measurement bias Add random bias offsets to the data

Analysis Add random offsets to the data based on analysis dates.

Note: The python dei software does not include the ability to calculate ‘measurement’ uncertainty that was available in the IDL code. The measurement bias uncertainty method provides a better way of estimating uncertainty due to measurement errors and bias.

7.1 Network Uncertainty

7.1.1 Description

The NOAA MBL reference is constructed from measurements of ~weekly samples collected at a subset of sites from the NOAA network. While the NOAA network is of global extent, it is currently too sparse to construct a reference that includes a longitudinal component. Thus, we construct a reference that averages in latitude any observed variability in longitude. This may introduce bias in the reference surface. We can estimate the uncertainty in the reference surface due to the distribution of MBL sites using a bootstrap analysis [Conway et al., 1994].

The MBL reference is constructed using data from a number of MBL designated sites. For each bootstrap run, we create a different network of sites by randomly choosing the sites from the original list of sites, and keeping the total number of sites the same. Each randomly chosen network has the requirement that a small subset of sites (which is gas dependent) must be included so that the meridional curves fitted to the north/south distribution are adequately constrained by the data. In each bootstrap run, some sites will be missing and some will be present two or more times. We then apply the data extension methodology to each bootstrap run.

Using the reference surface from each bootstrap run, we compute the mean and standard deviation at each time step and 0.05 sine of latitude. The standard deviation is our estimate of the uncertainty in the actual MBL reference.

7.1.2 Options

Use the option `--unctype=network` and `--numbs=100` for the **dei_driver.py** program to perform a network uncertainty bootstrap run. A typical value for `--numbs` is 100. For example:

```
python /ccg/dei/ext/src/python/dei_driver.py --unctype=network --numbs=100 --gas=co2 -  
↪-initfile=myinitfile.txt
```

7.1.3 Results

Results from the network bootstrap runs will be in a subdirectory of the main results directory called 'bs_network'.

7.2 Atmospheric Uncertainty

7.2.1 Description

The smooth curve fit to each MBL record reduces noise in the determination of the MBL reference by filtering out synoptic-scale atmospheric variability. The residual distribution (data minus smooth curve) includes variability related to synoptic scale weather patterns. We estimate uncertainty in the MBL reference due to atmospheric variability at each MBL site using a Monte Carlo analysis.

For each MBL record, we aggregate the residuals by month to compute an average monthly residual standard deviation (RSD). We then construct a new realization of the MBL record by adding a random error to the smooth curve value at each measurement time step. The random error is selected from a normal distribution with standard deviation equal to 1 and scaled by the average monthly RSD. A smooth curve fitted to this new record will differ from the original curve but will give the same residual statistics. We construct a reference using new realizations for each of the MBL sites. We repeat this process multiple times. As with the network uncertainty, we compute the mean and standard deviation at each time step and sine of latitude using all of the bootstrap runs. Again, we use the standard deviation as our estimate of the uncertainty in the actual reference due to atmospheric variability.

7.2.2 Options

Use the option `--unctype=atmospheric` and `--numbs=100` for the **dei_driver.py** program to perform an atmospheric uncertainty bootstrap run. A typical value for `--numbs` is 100. For example:

```
python /ccg/dei/ext/src/python/dei_driver.py --unctype=atmospheric --numbs=100 --  
↪gas=co2 --initfile=myinitfile.txt
```

7.2.3 Results

Results from the atmospheric bootstrap runs will be in a subdirectory of the main results directory called 'bs_atmospheric'.

7.3 Measurement Bias Uncertainty

7.3.1 Description

Over the full record of a gas measured by NOAA at a site, there may be times when measurement results show a small bias for differing periods of time due to unforeseen changes or contamination in the analysis equipment.

The measurement bias uncertainty is for a way of determining uncertainties in the data due to known and unknown instrumental bias in the analytical equipment. For example, there could be a known offset or bias in the results due to a contaminated valve that lasted a few months.

For each bootstrap run:

1. Determine the min and max ANALYSIS dates for the data,
2. Break up this analysis time span into sections, where
3. Each section is given a random length in days between user supplied minimum and maximum values,
4. Data within each section has a random bias value added, which comes from a normal distribution with a mean of 0 and a standard deviation given by the user.

So all data that were analyzed from the earliest analysis date to a randomly chosen time span, have a randomly chosen value added to them. We then go to the next section, pick another random time length and bias, and apply this bias to the data within that time section. This continues until the entire dataset has had a bias value applied. We then run a bootstrap data extension run with this data, and store the results. This is repeated for all bootstrap runs desired.

By looking at the spread of results over the bootstrap runs, you can get an idea of what uncertainty value you can apply to the data, due to this instrumental bias.

7.3.2 Options

Use the option `--unctype=bias` and `--numbs=100` for the `dei_driver.py` program to perform a bias uncertainty bootstrap run. A typical value for `--numbs` is 100. The additional option `--config=CONFIG_FILE_NAME` is also required. The file 'CONFIG_FILE_NAME' (not the actual name, you can call it anything you like), contains the data necessary for determining the time span and bias values to use.:

```
python /ccg/dei/ext/src/python/dei_driver.py --unctype=bias --numbs=100 --gas=co2 --
→initfile=myinitfile.txt --config=myconfig.txt
```

The config file must look like:

```
# gas, min period in days, max period in days, bias value
n2o    90      720    0.15
co2    90      720    0.05
co     90      720    2.00
sf6    90      720    0.02
ch4    90      720    1.00
```

7.3.3 Results

Results from the bias bootstrap runs will be in a subdirectory of the main results directory called 'bs_bias'.

7.4 Analysis Uncertainty

7.4.1 Description

Similar to the bias uncertainty, the analysis uncertainty applies a random value to the data, but over a specified time interval. This may be due to known changes in the analysis system, such as analyzer changes, where the noise of a sample measurement could change over different time periods.

For each bootstrap run:

1. User specifies one or more analysis time periods.
2. For each time period, a random value with user specified mean and standard deviation is added to data.

So all data that were analyzed during specied time spans have a randomly chosen value added to them. The random value can be from either a normal or uniform distribution, and can be applied to either the entire time period, or a new value applied to each sample.

7.4.2 Options

Use the option `--unctype=analysis` and `--numbs=100` for the `dei_driver.py` program to perform an analysis uncertainty bootstrap run. A typical value for `--numbs` is 100. The additional option `--config=CONFIG_FILE_NAME` is also required. The file 'CONFIG_FILE_NAME' (not the actual name, you can call it anything you like), contains the data necessary for determining the time span and bias values to use.:

```
python /ccg/dei/ext/src/python/dei_driver.py --unctype=analysis --numbs=100 --gas=co2
--initfile=myinitfile.txt --config=myconfig.txt
```

The config file must look like:

#	gas,	start_date	stop_date,	mean,	SD,	distrib,	random=1/fixed=0
n2o		1997-01-01	2017-08-31	0	0.2	normal	1
n2o		2017-09-01	2030-08-01	0	0.4	normal	1
co2		1979-01-01	1981-12-31	0	0.2	normal	1
co2		1982-01-01	2030-08-31	0	0.1	normal	1
sf6		1997-01-01	2030-08-31	0	0.04	normal	1
ch4		1983-01-01	1996-12-31	0	3.2	normal	1
ch4		1997-01-01	2009-12-31	0	1.5	normal	1
ch4		2010-01-01	2030-12-31	0	0.8	normal	1
co		1988-01-01	2030-12-31	0	1.8	normal	1

The last column should be set to 1 to apply a unique random value to each sample, or 0 to apply the same random value to all samples over the time period.

7.4.3 Results

Results from the analysis bootstrap runs will be in a subdirectory of the main results directory called 'bs_analysis'.

RESULTS

8.1 Directory Structure

All results from the dei calculations are placed in the directory specified by the `--bsdir` option to the `dei_driver.py` program. If this option is not specified, then the results are placed in a directory called `'dei_results'`. If any bootstrap calculations are done, the results from the bootstrap are placed in a subdirectory of the results directory. The name of the subdirectory depends on the bootstrap type:

- *network* - `'bs_network'`
- *atmospheric* - `'bs_atmospheric'`
- *bias* - `'bs_bias'`
- *analysis* - `'bs_analysis'`

In each bootstrap subdirectory, another directory for each bootstrap run is created, named `'1'`, `'2'`, `'3'` ... up to the number of bootstrap runs requested. Each of these numbered directories will have a set of the *output files* described below.

8.2 Output Files

A large number of files are placed in the results directory. In the example file names below, we'll use `'co2'` as the gas name, so for other gases `'co2'` will be replaced with the correct gas name, e.g. `'ch4'`. The output files are:

- **A copy of the initialization file.**
 - `init.co2.master.txt`
- **5 files for each site in the initfile. If the site code used in the initfile was `'alt_01D0'` then the files are:**
 - `alt_01D0_dat.co2` - The data input values
 - `alt_01D0_ext.co2` - The extended data values
 - `alt_01D0_fits.co2` - Values from the smooth curve fit to the data.
 - `alt_01D0_sum.co2` - Summary data from the curve fit to the site data.
 - `alt_01D0_wts.co2` - The weights used in the latitude gradient fits. One value per year.
- **3 files for each *latitude zone*. For example for the low southern hemispher (lsh) zone:**
 - `zone_lsh.mbl.co2` - Time series of values for the zone at each timestep.
 - `zone_lsh.mbl.ann.ave.sc.co2` - Annual averages for the zone.
 - `zone_lsh.mbl.ann.inc.tr.co2` - Annual increase for the zone.

- **3 files with information on the *latitude gradient* fits:**
 - *merid.data.srfc.mbl.log* - Data used in each latitude gradient fit at each timestep for final mbl surface.
 - *merid.smdata.ext.log* - Data used in each latitude gradient fit for the preliminary mbl surface.
 - *merid.smfits.ext.log* - Surface grid data from the preliminary mbl surface.
- **3 files with mbl surface grid information:**
 - *'surface.mbl.co2'* - surface mbl grid values (a value for each latitude and timestep)
 - *'surface.mbl.gr.co2'* - surface mbl growth rate grid values (a growth rate value for each latitude and timestep)
 - *'surface.mbl.fsc.co2'* - surface with function seasonal cycle grid values (harmonic values only, no polynomial trend)
- A file called *'syncsteps'* which has the date values for the time steps used in the calculations.
- A file called *'site_info.txt'* which has three columns of data: site code, latitude, sine(latitude), for the sites listed in the initfile.

8.3 Output files from bootstrap summary program (*dei_bs_driver.py*)

- *surface.mbl.network.co2* - The standard deviation for each time step/lat bin across all bootstrap surface mbl files.
- **15 files are created containing mean and std. dev. across all the bootstrap runs for each time step and each *latitude zone*:**
 - *zonal.ann.ave.sc.co2.bs* - annual averages from the smooth curve to each zone.
 - *zonal.ave.co2.bs* - annual averages for the zones
 - *zonal.fsc.co2.bs* - harmonic function values
 - *zonal.gr.co2.bs* - growth rate values
 - *zonal.iphd.co2.bs* - polar interhemispheric differences
 - *zonal.mm.tr.co2.bs* - monthly means of the trend
 - *zonal.residsc.co2.bs* - residuals from the smooth curve
 - *zonal.tr.co2.bs* - trend values
 - *zonal.ann.inc.tr.co2.bs* - jan 1 to jan 1 increase
 - *zonal.coef.co2.bs* - coefficients of the function fits
 - *zonal.ftn.co2.bs* - function values
 - *zonal.ihd.co2.bs* - interhemispheric differences
 - *zonal.mm.sc.co2.bs* - monthly means of smooth curve
 - *zonal.residf.co2.bs* - residuals from the function
 - *zonal.ssc.co2.bs* - smoothed seasonal cycle
- **10 files for each *latitude zone* containing uncertainties merged with values, e.g.:**
 - *zone_tnh.mbl.ann.ave.sc.unc.co2*
 - *zone_tnh.mbl.ann.inc.tr.unc.co2*
 - *zone_tnh.mbl.fsc.unc.co2*

- zone_tnh.mbl.ftn.unc.co2
- zone_tnh.mbl.gr.unc.co2
- zone_tnh.mbl.residf.unc.co2
- zone_tnh.mbl.residsc.unc.co2
- zone_tnh.mbl.ssc.unc.co2
- zone_tnh.mbl.tr.unc.co2
- zone_tnh.mbl.unc.co2

8.4 Output files from total bootstrap uncertainty program (dei_effective_unc.py)

Just one file is created, which is located in the main dei results directory:

- surface.mbl.effective_unc.co2 - Contains the total uncertainty (summed in quadrature) of all the bootstrap methods.

8.5 Output files from zonal bootstrap uncertainty program (dei_unc.py)

If more than one bootstrap method has been run, you can combine the uncertainties from each bootstrap method using the 'dei_unc.py' program. This will combine uncertainties in quadrature for each zonal file, for each curve type. For example, the arctic zone would have these files:

```
zone_arctic.mbl.ann.ave.sc.co2
zone_arctic.mbl.ann.inc.tr.co2
zone_arctic.mbl.co2
zone_arctic.mbl.fsc.co2
zone_arctic.mbl.ftn.co2
zone_arctic.mbl.gr.co2
zone_arctic.mbl.residf.co2
zone_arctic.mbl.residsc.co2
zone_arctic.mbl.ssc.co2
zone_arctic.mbl.tr.co2
```

Using 'dei_unc.py' will create corresponding files with uncertainties calculated from multiple bootstrap methods:

```
zone_arctic.mbl.ann.ave.sc.unc.ch4
zone_arctic.mbl.ann.inc.tr.unc.ch4
zone_arctic.mbl.fsc.unc.ch4
zone_arctic.mbl.ftn.unc.ch4
zone_arctic.mbl.gr.unc.ch4
zone_arctic.mbl.residf.unc.ch4
zone_arctic.mbl.residsc.unc.ch4
zone_arctic.mbl.ssc.unc.ch4
zone_arctic.mbl.tr.unc.ch4
zone_arctic.mbl.unc.ch4
```


LATITUDE ZONE DEFINITIONS

Zone Abbr.	Zone Name	Latitude range
HSH	high southern hemisphere	-90 to -30
LSH	low southern hemisphere	-30 to 0
LNH	low northern hemisphere	0 to +30
HNH	high northern hemisphere	+30 to +90
SH	southern hemisphere	-90 to 0
NH	northern hemisphere	0 to +90
GL	global	-90 to +90
EQU	equatorial	-14.5 to +14.5
PSH	polar southern hemisphere	-90 to -53.1
TSH	temperate southern hemisphere	-53.1 to -17.5
TROPICS	tropics	-17.5 to +17.5
TNH	temperate northern hemisphere	+17.5 to +53.1
PNH	polar northern hemisphere	+53.1 to +90
ARCTIC	arctic	+58.2 to +90

LATITUDE GRADIENTS

For each week in the synchronization period, we construct a latitude distribution (CO₂ (ppm) versus sine of latitude) using smoothed and interpolated values from the set of MBL extended records. The number of values in the distribution can vary weekly as MBL sites are added to the NOAA network or terminated. Confidence in values extracted from the smooth curve, SSTA(t), depends on the density of the data, the “scatter” in the data and the length of the measurement period. We use a relative weighting scheme, which assigns greater significance to sites with high signal-to-noise and consistent sampling. Relative weights range from 1 to 10 where the minimum weight of 1 is reserved for all interpolated values. A curve is then fitted to each weekly weighted latitudinal distribution. Details of the latitudinal fitting process are described by Tans et al. [1989].

The python dei software uses a pure python port of the pptfit FORTRAN code for calculating the latitude gradient curves. The python code is set up as a pptfit class and is contained in the file ‘ccg_pptfit.py’. It can be used in custom python programs like this

```
#!/usr/bin/python

import ccg_pptfit

x = [ -1.00,  -.905717, -.500000,  -.258819,  -.138598, .029666, .227756, .334136, .
↪ 433115, .533831, .795592, .894153, .981315]
y = [ 375.16, 375.15, 376.37, 378.06, 376.29, 379.47, 380.79, 381.75, 382.18, 382.37,
↪ 384.41, 382.40, 383.96]
wt = [ 10.0, 10.0, .930, .487, 10.0, 10.0, .990, 10.0, 2.63, 2.00, 3.67, 5.41, 4.94]

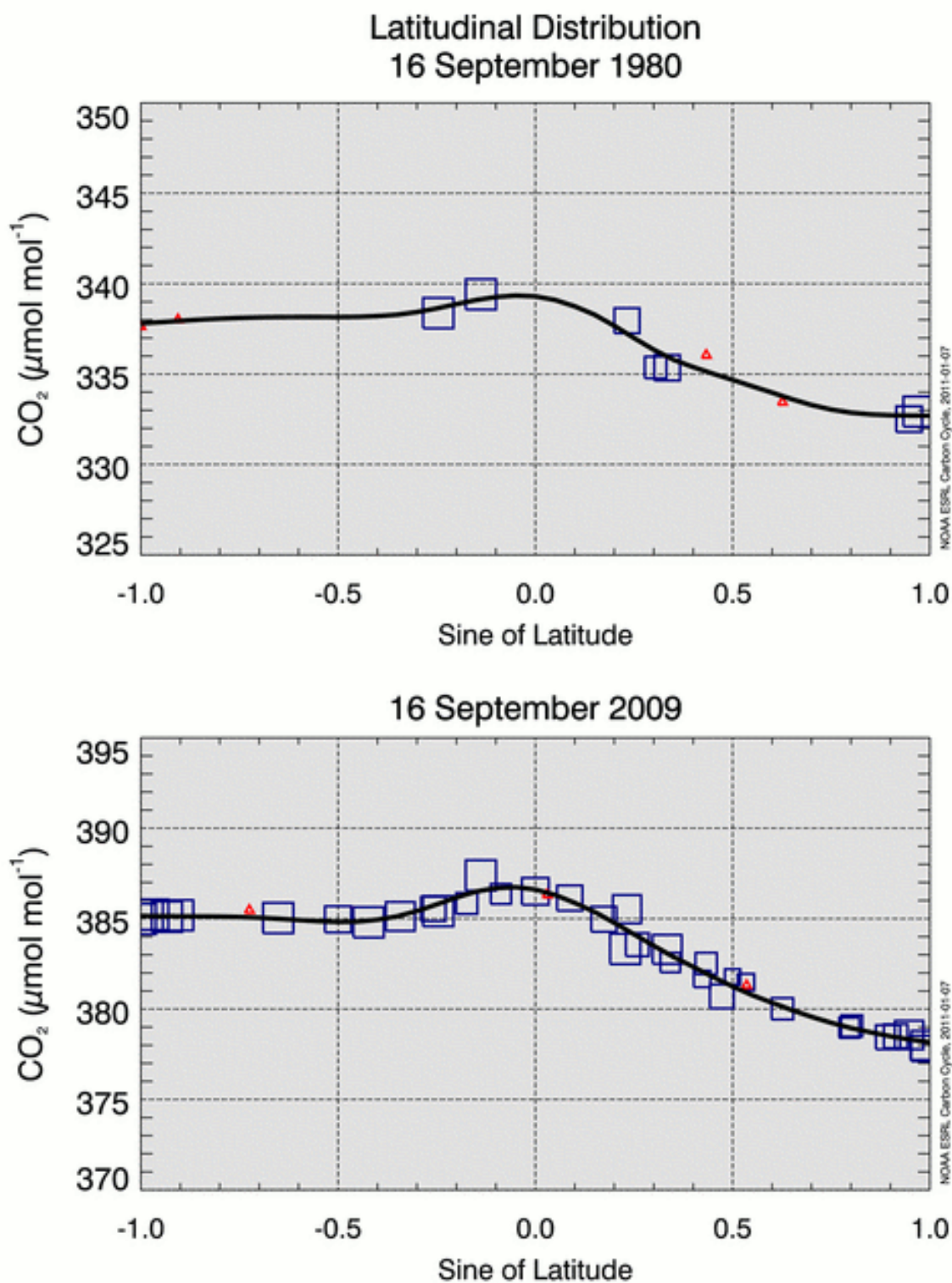
# set where to get predicted values
latbins = [n*0.05 -1 for n in range(41)]

# create pptfit
pptf = ccg_pptfit.ccg_pptfit(x, y, wt)

# get predicted values
ypred = pptf.predict(latbins)

# print lat, value
for xp, yp in zip(latbins, ypred):
    print "%8.2f %10.5f" % (xp, yp)
```

Figure 4: (top) Latitude distribution from MBL sites for 16 September 1980. Open blue squares represent values from the smooth curve. Open red triangles are interpolated values derived using the data extension methodology. The linear size of the symbol indicates the relative weight of the value. (bottom) Same but for 16 September 2009.



11.1 Methodology:

Read an init file with list of sites to use, and the time period. Fit smooth curves to each site. At each time step, fit a latitude gradient for sites designated as ‘mb1’ sites, and compute the value at 0.05 sin latitude steps. This gives a grid of time vs latitude. Compute differences between the smooth curves and the grid, and then recalculate the latitude gradients using these values. This gives final grid of time vs latitude. Combine various values from this grid to get zonal data. Gaps in the data are handled by substituting various values from the curve fits, or from function fits to the smooth-grid data.

Typical usage is something like:

```
ext = dei.dei(
    options.initfile,
    options.gas,
    resultsdir=options.bsdire,
    create_subdirs=True,
    anchor=options.anchor)

ext.make_input_files(options.bsdire) # make text input files we can reuse for each_
↳bootstrap run
ext.set_source_type(ext.FILE)        # tell dei to get input data from files
ext.modify_source(ext.ATMOS_UNC)     # modify the input data with atmospheric_
↳uncertainty
ext.run(options.numbs)               # run the data extension and save results
```

11.2 dei Objects

dei(initfile, gas, resultsdir="dei_results/", create_subdirs=False, anchor=False, quickfilt

Parameters

- **initfile** (*string*) – path and name of initialization file to use
- **gas** (*string*) – gas species to use, e.g. ‘co2’
- **resultsdir** (*string*) – directory name for where to put result files
- **create_subdirs** (*boolean*) – Whether to create a subdirectory under the resultsdir for the results output. Usually False on first initial run, then True for subsequent bootstrap runs

- **anchor** (*boolean*) – Set to True to ignore the sync2 value from the initialization file, and use current date instead. Also creates ‘estimated’ data for brw, kum, smo, spo beyond the last data point up to current date.
- **quickfilter** (*boolean*) – Perform a +/- 3 sigma outlier rejection from the smooth curve to the input data. This is done only when the input source is from the database (die.DB). Default is false.
- **stopdate** (*float*) – Specify the ending date of input data to use. Any available data after this date is not used in the calculations.
- **verbose** (*boolean*) – Set to true to print out messages during calculations.

initfile and gas are required, all others are optional.

11.3 Methods

`dei.set_data_source(inputdir)`

Set the directory where input data files are located.

`dei.make_input_files(bsdir)`

Get flask data from database and create text files with the data. These can then be reused for multiple bootstrap runs, instead of getting from database each time.

`dei.set_source_type(ext.FILE)`

Set where input data is coming from, either database or file. Can be either dei.FILE or dei.DB

`dei.modify_source(ext.ATMOS_UNC)`

Set modifications to input data before calculations. Must be one of ATMOS_UNC, BIAS, NETWORK or CUSTOM

`dei.set_data_callback(funcname)`

Set a function callback that will be called if the source type is CUSTOM. The function will be called with the input data lists, and it should return 2 modified lists,

e.g. `x, y = self.data_callback(x0, y0)`

`dei.run(numbs)`

Start the data extension calculations. Input value sets the number of bootstrap runs to perform. Default is 1

DIFFERENCES FROM IDL CODE:

- IDL version used three separate programs for calculations (ext_step1, ext_stp2, ext_srfc). Python version combines them into one step.
- No scale adjustments.
- mbl is based on smooth curve to data. idl code was based on trend + detrended curves to data.
- No 'exclusion' file for determining which sites to use for mbl surface.
- quickfilter rejects only 'preliminary' data, specified in database. idl code rejected data at any time.
- A 'bias' bootstrap method. A random 'bias' or offset is applied to a random time period of the data based on analysis date.
- The IDL 'measurement' bias was not implemented. Use the 'bias' bootstrap method instead.
- All bootstrap runs are contained in subdirectories of the main results directory.
- Results from each individual bootstrap run are in their own directory, instead of having all in one directory with bootstrap run number in the file name.
- Python version runs many times faster than IDL version, mainly due to eliminating file operations and spawning pptfit processes.
- File formats may be slightly different (number of significant digits e.g.).
- File names may be slightly different.

DEI GRAPHICAL INTERFACE

Usage: deigui

A graphical interface for setting options and running a data extension.

Instead of calling the python programs for running a dei job directly, you can use this graphical interface to set up the options and run the progrms. The gui will call *dei_driver.py*, *dei_bs_driver.py* and *dei_unc.py*, for a complete dei run, including bootstrap summary and uncertainty calculations.

13.1 Saved Configurations

The ‘File’ menu of the interface has two items, ‘Load Settings’ to load in previously saved settings from a file into the interface, and ‘Save Settings’, to save the current settings to file. Both items will bring up a dialog to allow you to enter or browse for the file name with the settings.

Example of a configuration file:

```
species = 'CO2'
init_file = '/home/ccg/kirk/dei/init.co2.txt'
output_dir = '/home/ccg/kirk/dei/co2/test'
use_text_files = True
text_files_dir = 'data'
anchor = True
quick_filter = True
stop_date = '2018-01-01'
bootstrap_methods = ['atmospheric', 'network', 'bias', 'analysis']
bootstrap_runs = 90
bias_config_file = 'bias.conf'
analysis_config_file = 'analysis.conf'
bs_outdir = '3and4'
```

13.2 Initialization Section

Species: Set the gas species you will work with. Equivalent to the *dei_driver.py* `-gas` option.

Init File: Set the location and name of the *initialization file*. Equivalent to the *dei_driver.py* `-initfile` option. Click on the first button to browse and select an init file. Click on the ‘View’ button to view and edit the init file.


Output Directory: Set the directory name where *results* will be written. Equivalent to the *dei_driver.py* `-bsdir` option. Click on the button to browse and select an output directory.


Data Extension

File

Initialization

Species: CO2


Init File: (No...  View

Output Directory: (None) 

Data

☒ Data From Database

☐ Data From Text Files

Location of Text Files: (None) 

☐ Anchor

☐ Quick Filter

☐ Use Stop Date for Data: 17 December 2018


Bootstrap


Type of bootstrap to run: ☐ Atmospheric ☐ Network ☐ Bias ☐ Analysis

☐ Only add bootstrap results to existing mbl run. Do not calculate mbl.

Optional Output Directory for bootstrap summary files:

Number of bootstrap runs: 100

Configuration file for Bias bootstrap: (No...  View

Configuration file for Analysis bootstrap: (No...  View

13.3 Data Section

Data from Database: If you want to use the data from the GML CCGG database, click on this button.

Data from Text Files: If you have prepared a set of text files to use for input data, check this button and fill in the box with the directory name where the files are located.

Anchor: Check this if you want to use the *dei_driver.py -anchor* option.

Quickfilter: Check this if you want to use the *dei_driver.py -quickfilter* option.

Use Stop Date for Data: Check this if you want to use the *dei_driver.py -stopdate* option, and enter the stop date in the boxes.

13.4 Bootstrap Section

Type of bootstrap to run: Select the bootstrap types that you want to run. You can choose more than one.

Only add bootstrap results to existing mbl run. Do not calculate mbl: Check this if you only want to calculate a bootstrap method and add the results to an existing mbl run. Equivalent to the *dei_driver.py -bsonly* option.

Number of bootstrap runs: Enter the number of bootstrap runs to do.

Configuration file for Bias bootstrap: If the 'Bias' bootstrap check button was selected, enter the configuration file for this bias bootstrap.

Configuration file for Analysis bootstrap: If the 'Analysis' bootstrap check button was selected, enter the configuration file for this analysis bootstrap.

13.5 Message Section

This sections will show all the messages that are printed by the programs as they are running.

13.6 Start Button

Click on the 'Start' button to begin the dei calculations. You will be prompted to confirm starting the run. Once the run has started, the 'Stop' button will become available, and you can click on it to stop the run before it completes on its own.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

d

[dei](#), 35

INDEX

D

`dei`
 module, 35

M

`make_input_files()` (*in module dei*), 36
`modify_source()` (*in module dei*), 36
module
 dei, 35

R

`run()` (*in module dei*), 36

S

`set_data_callback()` (*in module dei*), 36
`set_data_source()` (*in module dei*), 36
`set_source_type()` (*in module dei*), 36