Answers to comments by Anonymous Referee #1

We thank the reviewer for his/her careful reading and his/her comments on our manuscript.
We are very grateful for the comments resulting from thorough testing of the installation process and the usage instructions. This is very beneficial for improving the software and its documentation. We received additional bug reports from users, which we processed and provided as a bugfix release version v7.1.1 already. Feedback about new features or cosmetic changes were collected and kept for future releases.

The point-by-point replies to the comments are provided below:

Major comments:

Comment #1: *Type of manuscript The manuscript was submitted as a 'model description paper'. However, it does not describe a geoscientific model as such, but a pre-processor for the FLEXPART transport model. As such I strongly feel it should rather be in the category 'development and technical papers'.*

Answer: We agree and have already asked the editor to change it.

Changes in manuscript: The manuscript type was changed to the category 'development and technical paper'.

Comment #2: *Overall the manuscript is too long. Considering that this is 'only' a pre-processor and only few scientific methods are described, it should be possible to present the material in a more concise way, focussing more on the application side of things.*

First of all, the manuscript should more clearly distinguish between methods/scientific background and the application/user's manual section in the appendix. Right now there seems to be too much of a mix between the two, especially in section 3. Here are my suggestions for improving the flow of the main manuscript.
Section 1: Start with description of FLEXPART/FLEXTRA and the need for appropriate input data. Then the history of flex_extract. Wherever possible move historical information from sections below into this section.
Section 2: This section is very valuable, especially the provided Tables summarising the ECMWF data availability, which is cumbersome to obtain from ECMWF/MARS itself.
Section 3: Especially convoluted part between methods and application. Large parts of section 3 especially sub-sections 3.4 and 3.5.x are focussing only on application and should be moved to appendix.
Section 4: Very valuable, but could have clear description of settings for recommended best practices.
Section 5: Although, documenting, testing and benchmarking of a code are as important as the code itself, I suggest to significantly shorten this section and put the details into supplementary material. I can see that a lot of work went into this part, but for most readers/users this information is less relevant and of no direct consequence.

Appendix: This part should work as a user's manual. A such I think the structure would work better if those parts that differ between application modes would be organised by application mode and not by different installation/application steps. This would allow a user, who will usually be running the software in only one of the modes, to follow along without having to jump from section to section.

Answer: Although it is "only" a preprocessor, it is still a very complex task. In the past, users often had problems to use the software, for example with respect to choosing suitable combinations of parameter values and setting the parameter values correctly. A focus of this manuscript is therefore on thorough documentation of the software structure, its behaviour and the parameters that the software controls so that users understand what the software can do, where its limits are and which data sets can be extracted. We have kept the description as short as possible. It is necessary to include some hints and suggestions for use; we consider it appropriate to mention them in in the main part of the paper and do not think that they belong into the appendix. The appendix is intended to contain purely technical instructions for installation and use without additional scientific or application-related information.

Due to the complexity, we have created an additional on-line documentation that will provide more application examples in the future and should be continuously updated based on user feedback. Frequently occurring problems and their solutions should be described there as well as future changes.

Changes in manuscript: As for the structure of the manuscript, we have largely implemented the comments: Section 1 has been changed as suggested. We have moved some application-related comments from section 3 to section 4 and have also shortened section 3 somewhat. Section 4 has been supplemented with a few more hands-on details and an additional reference to the on-line documentation, where we already describe many of the parameters and CONTROL files in more detail. We shortened section 5 by moving a lot of material to the supplement. The technical description in the appendix has been optimised to keep the appendix short. We believe that restructuring as the Referee describes it would worsen the overview.

Comment #3: *As stated by the authors themselves, flex_extract is a collection of shell scripts and python scripts and FORTRAN code. As such there seem to be many possibilities to break the code in case external dependencies change. I am wondering if it would not be a much cleaner approach to provide this tool as a single python package. This should make maintenance, documentation and installation a lot easier. Python codes are able to work with external FORTRAN codes as well. So it would still be possible to make use of the arguably faster computations done in the FORTRAN code. But there seems to be little need for mixing in korn shell scripts (other than for job execution on ECMWF servers). Some interesting information in this respect can be found here: https://packaging.python.org/tutorials/packaging-projects/ https://www.numfys.net/howto/F2PY/ This is more a comment about the future of the code and may not be achieved quickly as part of a revision of this submission. It would also be beneficial to host the code on a more commonly used platform like github or gitlab. I was not able to git clone the code from the address given in the manuscript and would not know how to provide code feedbacks in form of merge/feature requests on the platform that is currently used for hosting the code.*

Answer: We are aware of the software's complexity and that there are many possible problems due to external dependencies. From user feedback about previous versions, we know that there were many problems precisely because of this and, among other benefits, the new version is supposed to make the whole installation and usage process a lot easier without changing too much of the

original behaviour of the software. This was achieved mainly by making use of system and Python packages instead of building the libraries from source, in addition to improving the documentation. We agree that there are many additional possibilities to improve this tool, and are aware of several of them; some are already mentioned in the "Future work" -Section 6.3, such as creating a single package for installation. So far, we have not decided on the method to accomplish that.
We would also like to note that there are only korn shell scripts for the job script submission to the ECMWF batch queue and nothing else. The additional bash scripts for the setup and running the software are not mandatory to use; however, we know from experience that many new users to flex_extract appreciated this easy-to-use access point. We also know from feedback that users of v7.1 were more often successful in installing the software in the first approach than with previous versions.
The code is hosted in the FLEXPART community git repository, and we will not host it on another platform to avoid confusion and to minimise maintainance efforts. We are happy if there are users who would like to contribute as additional developers and they can be granted write access to the git repository upon request (ticket or email). So far, only cloning of the git repository is possible for the public, and we are very sorry to admit that the link to the git repository in the manuscript was wrong. It was the link to FLEXPART and not flex_extract, which was the reason why the reviewer was not able to clone it. We corrected this link and added some details about possible contributions to Section 6.2 (Support). Feedback for new features or about bugs is very much welcome; it can be submitted through our ticket system.

Changes in manuscript: corrected link to git repository in the cloning instruction:
`git clone --single-branch --branch master` https://www.flexpart.eu/gitmob/flex_extract
Added a sentence for code contributions to Sect. 6.2: *Future contributions to the code are welcome; for granting permission of write access to the git repository, communication via email or ticket is necessary.*

Minor comments:

Comment #4: *P2,L39: 'non-authorised' is a bit strange. Users still need to be registered and as such are authorised to use the system. I would rather call this non-member state users.*

Answer: We agree.

Changes in manuscript: changed as suggested

Comment #5: *Version number, title and first mention P2,L44: If a totally revised version was created that is fundamentally different from the previous versions and as such not compatible with, for example, previous config files, it may be more appropriate to express this by incrementing the major version number. So why not go for 8.0?*

Answer: We can understand the question regarding the version number, but we want to stick with version number 7.1.x for the current release because of the following reasons:

1. Even though the code as well as the directory structure were changed substantially, the general behaviour did not change. Moreover, previous CONTROL files can still be used. We mostly added new features and eased the usage and understanding of some of the CONTROL parameters. We also had to rearrange the program flow to be able to allow the retrieval of CERA-20C and ERA5 data as well as using the different application modes. From the perspective of users of version 7.0, these changes are not fundamental.

2. Version 7.1 was already advertised through several presentations and posters at conferences, and we don't want to confuse users by changing the version number.
3. Before releasing version 7.1, we had some minor releases as versions 7.0.2, 7.0.3, and 7.0.4; each of these versions already included first quick-and-dirty implementations of some of the new features present in v7.1. They were not yet bug-free, adequately documented, or adequately tested. (We had to prepare them upon request and release them faster than we could prepare v7.1.) Releasing the current version as 7.1 shows this continuity.

As some bug fixes have been released after the manuscript submission as v7.1.1, and as the corrections resulting from user feedback and reviews warrant another increase of the micro-version, the revised manuscript now refers to version 7.1.2.

Changes in manuscript: No changes.

Comment #6: *P3,L70f: New version of what? I guess flex_extract, but could also refer to FLEXPART. There is also no clear mentioning above about how the first version looked like.*

Answer: We rephrased the sentence for clarification.

Changes in manuscript: *When the Comprehensive Nuclear Test Ban Treaty Organization (CTBTO) started to use FLEXPART operationally, it became necessary to adapt the extraction software (consisting of Korn shell scripts and Fortran programmes for the numerically demanding calculation of etadot) such that it could be integrated into ECMWFs automatic data dissemination system. This became the first numbered …*

Comment #7: *P3,L85: Also mention what happened to the FORTRAN part.*

Answer: A sentence was added.

Changes in manuscript: *… The Fortran part underwent some mostly cosmetic changes (source format, file names, messages, etc. and a minor bug fix) and an overhaul of the makefiles. …*

Comment #8: *P4,L100: 'most accurate data source'. That's a bit general. So ECMWF is better than any other atmospheric model? I would not go that far, but if you do you should link this comment to some kind of model evaluation that provides a similar conclusion.*

Answer: We reformulated the sentence to include the arguments why ECMWF provides the most accurate data for FLEXPART.

Changes in manuscript: *… which is considered to be the most accurate data source, ECMWF being one of the leading global weather forecast centres and providing data on model-level and at high time resolution.*

Comment #9: *P4,L103: Last sentence can be removed, should be clear from context.*

Answer: We agree. This comment is in line with Anonymous Referee #2's opinion.

Changes in manuscript: removed

Comment #10: *P4,L108: I would not say that they share the code, since these are two completely different software packages. But one could say that some of the code goes back to the same original routines. Sharing the code would mean that their would be a common specific library of routines that would be used by both models.*

Answer: We agree.

Changes in manuscript: *FLEXPART is based on it and some code goes back to the same original routines from FLEXTRA.*

Comment #11: *P4,L108: 'It ingests ...', It should be referring to FLEXTRA, but previous sentence on FLEXPART. Replace.*

Answer: Ok

Changes in manuscript: "*FLEXTRA ingests …*"

Comment #12: *P4,L131: 'are to be used' better 'are available' or 'can be used'.*

Answer: We would rather say "are utilised" instead.

Changes in manuscript: exchanged to "*are utilised*"

Comment #13: *P4,L131: 'Flex_extract automatically ...': Not clear how flex_extract comes in at this point. So for this section was describing the different access kinds to ecmwf. It needs to be made clear here what are the access possibilities and then in a second step how flex_extract is using them. Maybe don't mention Flex_extrac in this section at all.*

Answer: Since Sec. 2 is supposed to give a general overview specifically for explaining the data and access modes which are available through flex_extract, we think it is proper to mention flex_extract there. Regarding the sentence on "Flex_extract automatically …", we think it is the best place for mentioning this. Otherwise, we would unnecessarily need to repeat the information about the databases.

Changes in manuscript: No changes.

Comment #14: *P5,L141: Later on this dataset/model version is referred to as DET-FC. Why not keep the ECMWF terminology.*

Answer: This is due to the fact that it was originally the "Deterministic Atmospheric Model", thus DET-FC for the deterministic forecast. It was later changed to "Atmospheric High-Resolution Model". Since flex_extract can retrieve data from the whole available period, we decided to call it DET-FC. Nevertheless, it is confusing to start with HRES and then change back to DET-FC without mentioning a reason. We think that it is essential for the user to know both namies and modified the item accordingly.

Changes in manuscript: *The operational deterministic atmospheric forecast model (DET-FC), nowadays called atmospheric high-resolution forecast model (HRES),*

Comment #15: *P6,L157ff and Table 1: While the text nicely refers to the evolution in the IFS operational setup, I am missing a similar warning in Table 1. Seen on its own the Table may be misleading. Please add reference to Tables 2 and 3.*

Answer: We agree.

Changes in manuscript: … *The specifications for the operational data sets are valid for current data at the time of publication (except ENS-CV -- deprecated since 8 August 2016). For details about resolution and other parameters which have changed in the course of time, see Table 2 and Table 3.* ...

Comment #16: *P6,L177: 'member-state' instead of 'member'.*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #17: *P6,L178: 'not good'. Can you more specific what is not good about? Not robust/reliable or simply (too) slow?*

Answer: Ok.

Changes in manuscript: … *this access mode is very slow,* …

Comment #18: *P7,L184ff: The use of access and applications modes is a bit confusing here. Especially when the text refers first to access and application and then the combination of these is once more called application mode. Maybe the primary application modes could be described more clearly as 'execution location'?*

Answer: We agree that this sentence is not clear enough. We will stick to the "application modes" since this is now an established phrase in the manuscript and the online documentation. However, we will exchange "application mode" with "execution location" in this sentence as it is more accurate and precise. We also rename the "user access mode" to "user group", as this is in line with Sect. 2.1. This phrase would then also be in line with the description in Sec 3.1.

Changes in manuscript: *The actions executed by flex_extract (also called "the software'" henceforth) depend on the user group (see Sect. 2.1), the location of execution, and the data to be retrieved. There are three possible locations of execution, namely the ECMWF Member State Linux servers, the Member State Gateway server, or a local host.*

Comment #19: *P7,L213: 'module system': Rather call it 'environment modules framework'.*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #20: *Figure 1: Could this not be drawn much simpler without the emphasized distinction between the two different web APIs? The main point of this schematic should be an illustration of the 4 application modes. To user it does not matter so much that there are two different web APIs.*

Answer: We don't think that there is a reasonably way to significantly simplify Figure 1. It is important for the users to know which API they are going to use for a specific data set, as this has implications for the API which needs to be installed and for which one has to register.

Changes in manuscript: no changes

Comment #21: *P8,L231: Why not mention this python package in Table 7 as well?*

Answer: Because this table contains the software needed for all application modes, and ecmwf-api-client and cdsapi are only needed for the local mode.

Changes in manuscript: no changes

Comment #22: *P9,L254f: In the schematic (Figure 4) the "remote mode" is on top. So why start with explaining the "gateway mode" first?*

Answer: Ok. We rephrased this paragraph accordingly.

Changes in manuscript: *The remote and gateway mode both create a job script using the command-line parameters and the content of the specified CONTROL file and then sent it to an ECMWF batch queue. In remote mode this happens on an EMCWF server while the gateway mode uses the local gateway server for the creation and submission of the job. As the job script is executed from whichever of the two modes, it creates the job environment (in particular, the working directory) and starts "submit.py" to retrieve and post-process the data. Note that this locally started instance of "submit.py" triggers the work flow of the local mode but uses the MARS client to extract the requested fields from the database. The final output files are sent to the local member-state gateway server only if the corresponding option was selected in the "CONTROL" file. When flex_extract is used on a local host and in local mode, fields are extracted from MARS using one of the Web API's (which sends HTTP requests to ECMWF/CDS) and are received by the local host without storage on ECMWF servers.*

Comment #23: *Figure 4: The schematic may be easier to comprehend if ECMWF and local machines would always be on one side. Also the Labels for the different modes should be given rather on top with the sub-panel index. Right now the may be mistaken as only referring to the boxes on the right.*

Answer: Ok.

Changes in manuscript: Figure was changed accordingly.

Comment #24: *P9,L256: Once again the use of he different modes is confusing. While describing the 'gateway mode' it is mentioned that flex_extract is started in 'local mode'? Confusing. If I understand correctly, it is started locally to generate the ECMWF scripts and then sends them through the gateway. 'local mode', however, is not using the gateway but the WebAPIs. Correct? The same confusion is created in the caption to Figure 2, where 'local mode' is mixed into both gateway and remote mode.*

Answer: This comment is connected to #25, #26 and #22. We rephrased to whole paragraph for clarification.

Changes in manuscript: see answers to the other comments

Comment #25: *Figure 2: What is the purpose of the 'do local work' branch of the flow chart?*

Answer: As indicated in the figure caption it points to the branch where the local mode is executed, which constitues of the retrieval and post-processing tasks. The local mode is demonstrated in Figure 3. We rephrased the caption for clarification.

Changes in manuscript: *Flow diagram for the remote and gateway mode. A job script is created and submitted to the batch queue on an ECMWF server. The job script will then be executed on the ECMWF server to start flex_extract again for retrieving and post-processing of the data. The branch indicated by "queue = None" refers to the work flow shown in Figure 3.*

Comment #26: *Figure 3: Again, I am not sure that I understand the purpose of the two side-branches 'submit' and 'PRINTING MARS_REQUEST'.*

Answer: The submit branch indicates that flex_extract was started in remote or gateway mode and a job script has to be submitted. The "PRINTING MARS_REQUEST" branch indicates that flex_extract was started with the parameter REQUEST = 1, which means that only the mars_request.csv file is created and no retrieval or post-processing is done. We rephrased the caption for clarification.

Changes in manuscript: *If queue != None, flex_extract was started in remote or gateway mode and Fig. 2 applies.*

Comment #27: *Section 3.4.1 and Tables 8 and 9: I feel that these tables include the core information of how to setup and run the data extraction correctly. As such I think the description of the individual parameters should be given more space in the main text (as part of the Appendix) and not only in the supplement. This is crucial information for using flex_extract!*

Answer: We rather consider the Appendix as a technical description. Therefore, we don't discuss the parameters from the Tables there, and rather do that in Section 4, where we added further information on the most critical and potentially confusing parameters in combination with references to example CONTROL files.
Additionally, we recommend to use the online documentation as the future up-to-date source of information for the list of parameters and their value range. Additional detailed examples, which we thought would be too much for this paper, are already available there.

Changes in manuscript: See comment #38 for changes to Section 4.

Comment #28: *P14,L340: 'via the gateway server': not in 'remote mode'!*

Answer: Yes, this is a misleading statement, and it has to be made clear that the gateway server is needed in the gateway mode only.

Changes in manuscript: *… submitted to the ECMWF batch system. In case of the gateway mode, this is done via the local gateway server.*

Comment #29: *P14,L351: Do these modules correspond to the according boxes in the flowchart (Fig3)?*

Answer: They do. To make this connection recognisable, we added a reference to the figure. However, this description was moved to Section 4 due to the restructuring from major comment #2.

Changes in manuscript: *There are two more entry points into the software which can be used for debugging; they are described here for the sake of completeness: the Python scripts "getMARSdata.py and "prepare_flexpart.py". In the standard way of running the software, they are both imported as modules (as shown in Fig. 3), but they can also be used as executable programs.*

Comment #30: *P15,L363: Who is Paul James. Is there a reference to this work?*

Answer: Paul James developed and wrote the first disaggregation method, around 2000. There is no publication on that. We have removed the reference to his name, as admittedly it appears a bit as out of context, and rephrased the sentence slightly for clarification.

Changes in manuscript: *A pre-processing scheme is therefore applied to convert the accumulated values to point values valid at the same times as the main input fields while conserving the integral quantity with FLEXPART's linear interpolation.*

Comment #31: *P16,L390ff: At this stage it is not clear how these additional inputs are treated. Will there be separate output files for these times? How does FLEXPART deal with these? From which FLEXPART version onwards is this feature supported?*

Answer: The paragraph starting at p.16, l.390 already describes the process of how theses times are treated in flex_extract. The current FLEXPART version can not distinguish between the three time steps of a single interval. It will read all of them one after each other and keep the last one read from the file. Therefore, it would always work with the data from the second additional grid point and, obviously, we don't recommend using this scheme with current FLEXPART version. The FLEXPART version which supports the additional data will probably be the next one, v10.5. However, its release date is not yet determined. We added this information to the manuscript.

Changes in manuscript: *Current FLEXPART versions can not properly handle this input files generated with the new disaggregation scheme; they would use the third field (second additional sub-grid point in time), which would be worse than using the current method. The next minor version of FLEXPART to be released shall support the new scheme.*

Comment #32: *P16,L394: Maybe it would be useful to explain briefly why other fluxes are treated differently from precipitation.*

Answer: The other fluxes assume both signs and thus don't require conservation of positive definiteness. Furthermore, their impact on the FLEXPART results is smaller than that of precipitation, which is why the new scheme has been developed first for this latter variable. Motivated by the review comment, we have looked more in depth at the historical scheme implemented for the other fluxes and discovered that it does not correspond to FLEXPART, as it assumes cubic (not bicubic as wrongly stated in the manuscript) interpolation. In the future, we therefore want to adapt the scheme for the other fluxes similar to that of precipitation. However, we simplified the current formula.

Changes in manuscript: *The accumulated values for the other fluxes are first divided by the number of hours, and then interpolated to the times of the major fields. The algorithm was designed to conserve the integrals of the fluxes within each time interval when reconstructed with a cubic polynomial. It uses the integrated values F during four adjacent time intervals (F_0,F_1,F_2,F_3) to generate a new, disaggregated point value F which is output at the central point of the four adjacent time intervals:*

$$F = -\frac{1}{12}F_0 + \frac{7}{12}F_1 + \frac{7}{12}F_2 -\frac{1}{12}F_3$$

Note that a cubic interpolation was never implemented in FLEXPART. We therefore plan to replace this scheme by an adaption of the scheme used for precipitation, adapted to the situation where both positive and negative values are possible.

Comment #33: *P18,L453: grib_api seems to removed from all the makefiles and is also not mentioned in the dependencies (Table 7).*

Answer: Yes, the makefiles now expect eccodes instead of grib_api. However, as both provide the same functionality with respect to GRIB, one may still use grib_api. We rephrased the respective sentence to make that more clear.

Changes in manuscript: *The code is provided with a set of makefiles. The standard version assumes a typical GNU/Linux environment with the “gfortran” compiler and the required libraries: “OpenMP” for parallelisation which is included in the ”gcc” compiler package (“libgomp”), “ecCodes” for handling GRIB files, “EMOSLIB” for transformation between the various representations of fields. Note that the latter two typically require also so-called developer packages containing the Fortran module files. One may substitute “ecCodes” by its predecessor “GRIB_API”, if “ecCodes” is not available.*

Comment #34: *P18,L456: Usually one would refer to these versions as optimised and debug versions, not fast and debug. It would also be better to have a single makefile and only use a switch in the makefile to create either of the desired executables.*

Answer: “fast” is shorter, so we use it in the file name. A makefile switch for the selection of debugging vs optimisation is certainly an option, but we do not consider it a big difference. In that case, instead of the makefile name, the switch value to be used when calling make would have to be provided in the installation script. There might also be other options for makefiles, for example using a different compiler, therefore the flexibility provided by the current solution appears desirable.

Changes in manuscript: no changes

Comment #35: *P19,L483: Does the comment in braces mean, that no two instances of the submit.py script should ever be running at the same time? Where do you expect clashes if done anyway.*

Answer: There can be multiple instances at the same time if each instance uses a unique input directory. This is done automatically in remote and gateway mode. In local mode, the user has to take care of setting the input directory. The retrieved data from MARS are stored in *.grb files which have process ids in their filenames for identification. This feature is used in the post-processing part of flex_extract to adress them. It also allows users to start only the post-processing part again if the program, for example, terminated earlier due to time limit.
However, mixing of different retrievals can happen in the post-processing where several files with the same names are created. They can be overwritten without knowing, mixing the data sets. Additionally, files containing a time stamp (e.g. flux files) can also be mixed if the retrieval time period is overlapping.
Therefore we rephrase the sentence and add a note in the Section “Execution” (3.5.2). A proper solution may be implemented in a future version of flex_extract.

Changes in manuscript: *The process IDs are incorporated so that the GRIB files can be addressed properly in the post-processing.*
In Sect. 3.5.2: *Please note that when flex_extract is started in local mode, the parameter INPUTPATH in the run_local.sh script must be set, so that each retrieval uses a unique directory to avoid mixing of data files.*

Comment #36: *P20,L530: What about forecasts longer than 99 hours. How will they be written into a 2 digit number?*

Answer: The forecast always starts from an analysis at a specific hour which is called forecast time. From this time, each hour into the forecast is called a forecast step, and the forecast could indeed be longer than 99 hours. Therefore, the forecast step is already designated by a 3-digit number as noted in this Section 3.9.2. The forecast time, marked by HH, is always a two-digit number. To get the full date of any forecast step, the date, forecast time and forecast step have to be used to calculate the real date and time.

Changes in manuscript: No changes.

Comment #37: *P22,L577: This is comment that cannot be fixed in the manuscript, but should be addressed in the future. I assume you are referring to the Global Forecast System of NCEP here? If the availability of model level data is a shortcoming to the community. NCEP should be approached to make model levels available as well. Internally GFS operates on a similar hybrid sigma-pressure coordinate system as ECMWF IFS.*

Answer: Yes, we refer to GFS. However, we do not see it in our responsibility to approach NCEP for model level data. Of course, users are free to pursue this and we assume that the FLEXPART developers would support a corresponding update of the FLEXPART code (which would be necessary in that case).

Changes in manuscript: no changes

Comment #38: *Section 4: The section gives many valuable hints for best practices. However, I would find it very useful if these could be manifested in more concrete instructions, maybe in form of clear example CONTROL files. For example it is mentioned that only a subset of levels can be extracted, but how is this done concretely and does it work for all etadot modes (didn't in the past)?*

Answer: Where appropriate, we added references to example CONTROL files, or described how the parameters have to be set. We also moved some of the usage instructions from section 3 to section 4 and extended the instructions to describe new and/or confusing parameter settings.

Changes in manuscript: Section 4 was further divided into subsections "4.1 Example CONTROL files", "4.2 Changes in CONTROL file parameters in comparison to previous versions" and "4.3 Scientific considerations". Section 4.1 was previously Section 5.5 with some additional information about file names. Section 4.2 lists the most important changes to the CONTROL files in version 7.1 and Section 4.3 is the previous Section 3 with some additional references to example CONTROL files.

Comment #39: *P28,L746f: Actually one does not need any code for this task. It can be performed using standard bash cat command. That is one of the few beauties of grib format.*

Answer: We are aware that standard bash cat commands could be used. However, as the Python ecCodes package contains a specific command to do the merging, and as the workflow is implented in Python, we deem it more useful to use the ecCodes command.

Changes in manuscript: No changes

Comment #40: *P28,L752ff: See also previous comment on package structure. The possibility of calculating etadot should not just be discarded, but there certainly are better ways of integrating the FORTRAN code into the python package.*

Answer: Calculation of etadot will not be discarded, especially because of compatibility reasons and older data sets, such as ERA-Interim. We rather anticipate a solution where the software calls calc_etadot only if etadot needs to be calculated. We reformulated the comment in the "Future work" Section (6.3). Please see also comment # 3 (major section) for the answer (repetition).

Changes in manuscript: *In future versions of flex_extract, calc_etadot will probably only be called if etadot really needs to be calculated, not just for multiplying it with \partial p / \partial eta as this can be done with sufficient efficiency in Python.*

Comment #41: *P28,L755f: Again as mentioned above, instead of just providing a tar ball, a public git repository that would allow for swift user feedbacks and contributions would be beneficial.*

Answer: Please see comment # 3 (major section) for the answer (repetition).

Changes in manuscript: No changes

Comment #42: *P29,L790: Better refer to this as dependencies or prerequisites.*

Answer: Ok

Changes in manuscript: We changed the heading of appendix sub-section A2 to *System prerequisites*

Comment #43: *Code sections in Appendix (e.g., P31,L850ff): It would be easier if such sections would be directly given as example config files or scripts. Copy and paste from a manuscript is rather cumbersome and error-prone.*

Answer: We prepared the shell scripts and Python code snippets as supporting scripts for the users. However, we refrained from including the installation commands for the additional software and library packages into a script since the installation commands are distribution-dependent.

Changes in manuscript: *Shell scripts and Python code snippets mentioned in the Appendix can be found in the directory Testing/Installation/ after unpacking the tarball.*

Comment #44: *P34,L994: Does testing CERA-20C refer to CONTROL_CERA or is there another prepared example to test it. Please mention.*

Answer: The details of the data set are described in the following subparagraphs, where we provide step-by-step instructions for this first test with CERA-20C data, depending on the application mode. Since it is thus mentioned twice, we decided to move the details up, out of the subparagraphs, and to add the name of the CONTROL file.

Changes in manuscript: *In the following, we will provide step-by-step instructions for all application modes to retrieve a single day (08 September 2000) from the CERA-20C dataset with 3hourly temporal resolution and a small domain over Europe with 1° resolution, using CONTROL_CERA[.public].*
We removed the details from the subparagraphs ”Remote and gateway modes” and ”Local mode”.

Comment #45: *P35,L103: This unload was not mentioned above for the installation process. Is it necessary? Because once you unload python on ecgate the python command is linked to the system pyhton version 2.6! So I don’t see the gain of the unload.*

Answer: This is indeed not necessary and might confuse readers. Therefore, we removed this line.

Changes in manuscript: Line 1003 (… *module unload python* ...) deleted

Comment #46: *Table 7: It is mentioned in the text that compilers other than GNU compilers were tested as well. This should be reflected in the table. emoslib should just be emos. The information for fortan/pyhton is not required since this is given by the header line.*

Answer: A makefile is provided for the ECWMF HPC environment tested with the Fortran compiler included in CrayPE 2.5.9. Originally, also an Intel Fortran makefile was included, but this has been removed for the sake of simplicity, as gfortran is sufficient nowadays. As ECMWF itself calls the EMOS library “emoslib” in its Wiki system, we think it is better to keep this designation. We accept the proposal to simplify the caption.

Changes in manuscript: Caption has been simplified, information on HPC ftn compiler added.

Comment #47: *Table 11: With files job.temp and job.template it is difficult to distinguish their function just from file name. The second one should maybe be renamed to install.job.template Why does the first template not have the template file name extension. Is it treated in a different way than the other templates? Same for convert.nl. Even if the output of the template processing is a namelist shouldn't it have the same template ending.*

Answer: We agree and have renamed the template files as follows.

Changes in manuscript:
convert.nl -> calc_etadot_nml.template
compilejob.template -> installscript.template
job.template -> jobscript.template
job.temp -> submitscript.template

Comment #48: *Table 20: Is this table really needed? It is only a list of file names without further description. I can obtain this also by looking into the corresponding folder! Either remove or add a meaningful description of the listed files. The caption alone does not explain the details.*

Answer: We agree with the comment and delete this table; instead some more specific references for examples are added in Section 4. Additionally, the on-line documentation will be expanded in the future and will include more detailed descriptions.

Changes in manuscript: See comment #38 for changes to Section 4.
We removed Table 20 and its reference.

Technical comments:

Comment #49: *P3,L83/84: Past tense in previous sentence. Why not here?*

Answer: Agreed.

Changes in manuscript: *Version 7.0.4 enabled the retrieval of multiple ensemble members at a time and included bug fixes for the retrieval of ERA5 and CERA-20C data.*

Comment #50: *P4,L106: Style. Twice representing in same sentence.*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #51: *P5,L146: Citation style for Laloyaux et al. (2018)*

Answer: Ok

Changes in manuscript: changed to … *CERA-20C (Laloyaux et al., 2018) reanalysis.*

Comment #52: *P6,L181: ’necessary’ not ’ecessary’*

Answer: Typo

Changes in manuscript: corrected

Comment #53: *P16,L387: Citation format of ’(Hittmeir et al., 2018, p. 2513)’. No pages in journal reference.*

Answer: We thought that it might be useful for the readers if they don't have to search within the cited article. If it is contrary to the GMD style, the technical editor will remove it.

Changes in manuscript: no changes

Comment #54: *P17,L414f: So in the end FLEXPART converts etadot (or etapoint) back to a Cartesian vertical velocity? So why the painful calculation of etadot in the first place? Only because of the historic development from FLEXTRA?*

Answer: The multiplication of etadot with deta/dp (partial derivative!) does not create a Cartesian vertical velocity, as can be easily seen by looking at the situation close to sloping ground, where etadot is close to zero while w and omega won't be.

Changes in manuscript: no changes

Comment #55: *P17,L416: ’pressure’ not ’perssure’*

Answer: Typo

Changes in manuscript: corrected

Code comments:

Comment #56: *The FORTRAN code works with default FORTAN unit names for I/O (see table 17). This seems a bit outdated and difficult to follow/debug/etc. Please change to meaningful filenames wherever possible. This would allow removing Table 17 from the manuscript.*

Answer: We plan to do this, but not for the current release. It will require some adaptations in the regression testing, which goes too far for the manuscript revision. As users don't normally interact with these file names, we don't see it as a priority.

Changes in manuscript: no changes

Comment #57: *Besides the documentation in the main routine the comments in the FORTRAN code are mostly in German. This should be changed in future submissions.*

Answer: We are aware of this, but because of the uncertain future of the code, we haven't yet put much work into it. However, the main description at the start of each program unit (which appear also in the FORD-generated on-line documentation) are now all in English.

Changes in manuscript: no changes

Comment #58: *Section 3.8.3: And why would the FORTRAN program need to save these variables to a file instead of storing them internally? Makes little sense if they are not used again, for example, by the python scripts.!*

Answer: As stated in a comment line of the Fortran programme, orginally this file was used by a simulation model called POP model. We have commented out the write statement in the code now.

Changes in manuscript: The section was removed.

Comment #59: *Installation process: I only tested the remote mode on ecgate. My experience was not as straightforward as the description may suggest and as one would expect on a system the code should have been tested on. Since the remote installation is the one installation mode that is running on a defined system it should be easier to get this done without having to change various parameters in the setup script. First of all, I tried the git clone command on page 28. However, this does not clone flex_extract, but FLEXPART!? In the following I could not figure out the correct git address for flex_extract and resigned to to downloading the tar ball instead. Once unpacked, I was able to proceed. However, I don't see any reason why the installation process is submitted to the queueing system on ecgate. We are talking about a very small compile task here. It would be more simply done on the login nodes so that one has a more direct feedback when things go wrong. Anyway, first email I get back from the submitted install job was a very general compile error ("No rule to make target 'phgrreal.f', needed by 'phgrreal.o'. Stop"). So I checked if I selected the correct makefile. Yes, one would think: makefile_ecgate (the one system where this should work. Right?). Taking a look into makefile_ecgate shows that file extensions seem to have changed from f to f90, but this was not adjusted in the makefile. So I modified the makefile. Second try. Got an email which may indicate success of the FORTRAN compile, but does not tell me anything else about the rest of the installation. So let's see if it is there. Hm. Not in the directory I had specified in $INSTALLDIR. But there is a new folder flext_extract_v7.1 in my home. OK. Fine. Let's test it. So I follow the instructions in Appendix A4. However, the executable mentioned there is called calc_etadot_fast.out, while in my installation it is simply called calc_etadot. No problem. But shouldn't the exe name be harmonised in the makefiles? I did not change it. Running the executable seems to work but in my case only produced 2 lines of output in contrast to 7 mentioned in Appendix A4. Once again, did I do something wrong. Did it really work? There are 2 output files in the directory now and one can assume that that's sufficient. grib_ls on fort.15 (the obvious output file) at least shows some etadot fields. However, I was still worried about the missing lines of output. So I quickly searched the FORTRAN code for where these messages should be produced. I found them but they are commented out. So no wonder that I did not get these messages! But why are they given in the instructions.*

Answer: Unfortunately, the git clone command in the installation instructions incorrectly pointed to FLEXPART. Testing of flex_extract was done regularly. During the whole development process, and right before the submission of the manuscript, the code has been tested. However, because the existing flex_extract directory had not been removed from the ECWMF server before we started the new installation, we did not notice the problems with the makefiles and outdated source files. This

was corrected right after we received the first bug report in a new flex_extract version 7.1.1. We also harmonised the name of the Fortran executable by linking it to “calc_etadot”, regardless of the makefile selected. There might be additional names which are used for the regression tests and not important for the user. We are very sorry that you had this rough start with our software. Regarding the easier usage of the setup.sh script, please note that flex_extract users choosing gateway or remote mode always need to change user-specific parameters in the setup.sh script, therefore it is not easily possible to further simplify the process. The script already includes default settings for gateway and remote mode.
Concerning the installation process in remote mode, we would like to state that flex_extract distinguishes in its behaviour between ECMWF servers and local hosts. Even though, technically, the gateway and remote modes work differently, they share the same code. Therefore, we will stick to this procedure for now and keep the idea of a local installation for the remote mode as an option for a new feature. Additionally, the remote and gateway mode have both a fixed installation target, namely the $HOME directory, as described in the paper. Currently, this is not changeable.
The differences in the output from the Fortran code compared to what was given in the installation instructions are due to modifications made to the code in a late stage of preparation, unfortunatly having overlooked to update the manuscript.
We went through the steps in the installation instructions again to make sure that everything is now up-to-date.

Changes in manuscript: Corrected the git clone command and the Fortran test output.

Comment #60: *Test retrieval: A test retrieval using the CONTROL_CERA example worked without much trouble. I would only suggest that the loading of the python3 module should be added to the run.sh script.*

Answer: Thank you for the suggestion. We have added automatic loading of Python3 module when the setup or run script detects an ECMWF server.

Changes in manuscript: Removed instruction to load Python3 module in the installation and usage sections.

Comment #61: *EcFlexpart.py: As far as I can see this is where the ectrans command is used to transfer the data. The ectrans command is used in a way that assumes that it would crash if an invalid transfer was submitted. But this is not how ectrans works. Much like a queuing system ectrans submits the requested file transfer into a queue returning a request ID. transfer request will then be repeated if not successful at the first try. So if the settings for source and target file are not correct ectrans will still return a request ID and try the transfer until it reaches its repeat limit. Hence, it is not possible to simply capture the return code from ectrans to detect an error.*

Answer: We know how ectrans works. However, the ectrans command is executed, like all other command-line calls, from Python scripts, making use of an extra function. Therefore it undergoes error handling, even if an error will never occur. We don't think that changes in the code are necessary. As one might perceive a need for an explanation, which is lacking in the manuscript so far, we added a note in the usage section for remote and gateway modes.

Changes in manuscript: *Please note that success of the submission of the ectrans command does not guarantee that the file transfer will succeed. It means only that the output file has been successfully submitted to the ectrans queueing system. One still has to check manually in the local directories or with ECaccess tools whether the files reached their final destination.*

Comment #62: *Work folders on $SCRATCH are called python<PID>. Why not use something that would clearly identify these as flex_extract folders?*

Answer: This has historical reasons; it comes from version 6 where Python was first introduced. However, we agree that the prefix could be more telling, and changed it to “extract” instead of “python”.

Changes in manuscript: We changed the code and text accordingly.
Manuscript p. 36, l.1048: … *temporary retrieval directories (usually extractXXXXX, where XXXXX is the process id). Under extractXXXXX a copy of the CONTROL file ...*

Answers to comments by Anonymous Referee #2

We thank the reviewer for his/her comments on our manuscript. Since the other referee (#1) suggested to shorten the manuscript, we have shortened Section 5 on quality assurance, transferring a major part from this section to the supplement.

The point-by-point replies to the comments are provided below:

Minor comments:

Comment #1: *line 58, 1990ies (and later), change to 1990s or ’90s: MLA Handbook for Writers of Research Papers, 7th edition. This, from p. 84. "Decades are usually written out without capitalization (the nineties), but it is acceptable to express them in figures (the 1990s, the ’60s). Whichever form you use, be consistent."*

Answer: Ok

Changes in manuscript: changed to 1990s

Comment #2: *line 62, interSect –> intersect*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #3: *line 67, and later of –> and later on*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #4: *line 86, ecCodes (earlier referred to as eccodes) be consistent*

Answer: Thank you for this hint. We agree.

Changes in manuscript: changed to ecCodes throughout the manuscript

Comment #5: *line 86, would change ’was implemented’ to ’was utilized’*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #6: *line 98, ECMWF and IFS are already defined... and throughout, drop repeat acronym defintions*

Answer: We agree, except for the definition in the introduction section, although it was already defined in the abstract (abstract may not be read by everybody).

Changes in manuscript: We eliminated repeated acronym definitions throughout the manuscript for ECMWF/IFS/MARS and others.

Comment #7: *line 103, "All details can be found in the literature mentioned" is an empty sentence, suggest dropping it.*

Answer: Ok

Changes in manuscript: eliminated as suggested

Comment #8: *line 113, use ECMWF as it has been defined...*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #9: *line 116, MARS has been defined...*

Answer: Ok

Changes in manuscript: changes as suggested

Comment #10: *line 150, choose meta data or metadata and use consistently*

Answer: We chose meta data.

Changes in manuscript: changed throughout the manuscript to meta data

Comment #11: *line 155, "The horizontal grid type refers to the way how it fields are archived in MARS" not clear...*

Answer: We slightly rephrased the sentence, refering to the spatial representation which was already defined on page 4 lines 118-120.

Changes in manuscript: The horizontal grid type refers to the spatial representation.

Comment #12: *line 169, inplied –> implied*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #13: *line 181, ecessary –> necessarily*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #14: *line 720, Lag rangian –> Lagrangian*

Answer: Ok

Changes in manuscript: changed as suggested

Answers to comments by Anonymous Referee #3

We thank the reviewer for his/her comments on our manuscript.

The point-by-point replies to the comments are provided below:

Minor comments:

Comment #1: *l. 5 I suggest replacing 'therefore' with 'consequently' to avoid a repetition with the previous sentence*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #2: *l. 22 should be 'Integrated Forecast System'*

Answer: Yes, that's an error.

Changes in manuscript: corrected

Comment #3: *l. 41 I believe you want ECMWF here*

Answer: Yes

Changes in manuscript: corrected

Comment #4: *l. 58 should read '1990s'*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #5: *l. 62 You probably want 'intersect' and not 'interSect.'*

Answer: Yes

Changes in manuscript: corrected

Comment #6: *l. 99 Should read US National Centers*

Answer: Ok

Changes in manuscript: corected

Comment #7: *l. 107/108 – the two reference should be in brackets*

Answer: Agreed.

Changes in manuscript: corrected

Comment #8: *l. 116 reference should be in brackets*

Answer: Agreed

Changes in manuscript: corrected

Comment #9: *l. 146 reference should be in brackets*

Answer: Agreed

Changes in manuscript: corrected

Comment #10: *l. 155 suggest skipping 'it' or better replacing it with 'the'*

Answer: Ok

Changes in manuscript: We rephrased the sentence. (See comment # 11 of Anonymous Referee #2)

Comment #11: *l. 169 'implied; instead of 'inplied', I believe*

Answer: Yes

Changes in manuscript: corrected

Comment #12: *l. 181 missing 'n' in 'ecessary'*

Answer: Ok

Changes in manuscript: corrected

Comment #13: *l. 184 - 187 and then also l. 206 – 209: are 'access mode' and 'application mode' two different things? Are 'ECMWF Member States Linux servers', 'Member State Gateway server' and 'local host' rather access modes than application modes? And based on those three access modes, four application modes are derived? Maybe it is then clearer in conjunction with the header*

of section 3.1, where 'Remote', 'Gateway' and 'Local' are referred to as application modes. And then in conjunction with user mode it results in four user application modes.

Answer: This comment is somehow related to Anonymous Referee #1's comment 18. Yes, access modes and application modes are different. However, in this context the term "access mode" is better replaced by "user group" and the "application mode" in this first sentence should be replaced by the "location of execution". The description was not clear and we renamed the respective terms.

Changes in manuscript: *The actions executed by flex_extract (also called "the software'" henceforth) depend on the user group (see Sect. 2.1), the location of execution, and the data to be retrieved. There are three possible execution locations, namely the ECMWF Member State Linux servers, the Member State Gateway server, or a local host.*

Comment #14: *l. 187 results*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #15: *l. 227 – depends which spelling you follow; if it is BE then 'licence'; but I noticed indeed that Copernicus Publications follow AmE spelling of this word, namely 'license'*

Answer: Yes. We use British English. Only in the case of the filename LICENSE.md AE is used following the dominant usage in the software community. In any case, the Copernicus editors will have the final word.

Changes in manuscript: no changes.

Comment #16: *l. 240 – I may be missing something important in the software structure but I do not understand why sending a script to the ECMWF batch queue in step 1 contrasts in steps to and 3; in particular do not understand 'or' here; I would imagine that that job sent to the batch queue also retrieves data from MARS and post-processes them to obtain FLEXPART input fields. From l. 249-252 I understand that flex_extract proceeds with steps 2 and 3 locally; is it correct? It is a small thing but it would be good to clarify*

Answer: Flex_extract sends a korn shell script as job script to the batch queue and when this script will be started from the queue eventually, it starts flex_extract again to finally retrieve (task2) and post-process (task3) the data on ECMWF servers. Other Reviewers had the same problems in understanding this logic and we therefore rephrased it. Please see comments #22, #24, #25 and #26 from Anonymous Reviewer 1 for our answer.

Changes in manuscript: Please see comments #22, #24, #25 and #26 from Anonymous Reviewer 1 for the changes.

Comment #17: *l. 268 – you want to use 'which' once*

Answer: Yes

Changes in manuscript: corrected

Comment #18: *l. 283 – should read 'for a correct setting'*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #19: *l. 284/285 – you probably want to use the word 'combination' only once*

Answer: Yes

Changes in manuscript: corrected

Comment #20: *l. 355/356 – these papers should be referenced in brackets*

Answer: Yes

Changes in manuscript: corrected

Comment #21: *l. 363 – does it stem from private communication with Paul James?*

Answer: This comment is in line with Anonymous Referee #1's comment #30. Please see there for our answer.

Changes in manuscript: Please, see comment #30 of Anonymous Referee #1 for the applied changes.

Comment #22: *l. 373/374 – what is the position of 1 and 2 with respect to a,b,c,d*

Answer: This would be 'b' and 'c' respectively. Nevertheless, we rephrased the sentence for clarification.

Changes in manuscript: *... which is output at the central point of the four adjacent time intervals:*

Comment #23: *l. 416 should read 'pressure'*

Answer: Yes

Changes in manuscript: corrected

Comment #24: *l. 504 – what is a pure forecast? Do you mean deterministic forecast? 'Pure forecast'also later appears in l.527*

Answer: In our context, a pure forecast means that we extract only forecast fields and the retrieval period is longer than one day without analysis fields in between. We added this information in Sect. 3.9.2. and added a reference to this section from the first appearance in 3.8.5. We also changed the naming in “long forecast”.

Changes in manuscript: *For a long forecast, where only forecast fields are retrieved for more than 23 hours,…*

Comment #25: *l. 513/514 – not sure what this sentence mean? Is there just ‘is’ missing in this sentence? Or is there more to add?*

Answer: There is just an ‘is’ missing.

Changes in manuscript: corrected.

Comment #26: *l. 524 - is YYMMDDHH in this file name the analysis hour?*

Answer: No, it is the valid time.

Changes in manuscript: *There is one file per time step and YYMMDDHH indicate the date and hour for which the fields are contained in the file.*

Comment #27: *l. 529 would ‘base time’ be better than ‘starting forecast time’*

Answer: We added the term ‘base time’ since ECMWF uses it.

Changes in manuscript: *The HH represents the starting time (base time) of the forecast.*

Comment #28: *l. 535 – I may be missing something here but why the file names selected for the ensemble members do not account for FORECAST_STEP?*

Answer: If we retrieve a combination of ensemble members and pure forecast, the naming scheme would be a combination of Sect. 3.9.2 and 2.9.3.

Changes in manuscript: no changes

Comment #29: *l. 548 – should read ‘makes’*

Answer: OK

Changes in manuscript: corrected

Comment #30: *l. 581 – I would skip the coma after e.g.*

Answer: OK

Changes in manuscript: changed as suggested

Comment #31: *l. 615 – do you mean ‘paths’?*

Answer: Yes.

Changes in manuscript: corrected

Comment #32: *l. 624 – should read ’except’*

Answer: Yes

Changes in manuscript: corrected

Comment #33: *l. 687 – better ‘indicates a lower complexity’*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #34: *l. 720 ‘Lagrangian’ should be one word*

Answer: Yes

Changes in manuscript: corrected

Comment #35: *l. 721 – 1990s*

Answer: Ok

Changes in manuscript: corrected

Comment #36: *l. 940 – skip ‘an’*

Answer: Ok

Changes in manuscript: corrected

Comment #37: *l. 942 and 943 are too tightly formatted in vertical (also true for l. 848 and 849 and l. 888 and 889 then for lines 966, 967 and 968; subsequently lines 982, 983 and 984 have different vertical formatting than the rest of the manuscript; l. 1062 and 1063 are too close in vertical as well)*

Answer: Yes, we noticed that this comes from the scriptsize environment where we missed to introduce a new paragraph before the following section.

Changes in manuscript: corrected

Comment #38: *l. 954 you may want to insert a blank space after the bracket*

Answer: Ok

Changes in manuscript: changed as suggested

Comment #39: *In the caption of Tab.2 'resolution' instead of 'resoltion'*

Answer: Ok

Changes in manuscript: corrected

Comment #40: *In the body of Tab.6 – should be 'Public users have to register for obtaining an account'*

Answer: Ok

Changes in manuscript: corrected

Comment #41: *In the caption of Tab. 18 you need to insert a blank space after etadot*

Answer: OK

Changes in manuscript: corrected

Comment #42: *In the caption of Tab. 24 it is difficult to understand the first sentence starting from: 'Python code'; could you, please, re-phrase. I would also suggest putting sections v7.0.4 and v7.1 of this table side by side and not one on top of the other, if possible (I am aware there is no strict correspondence between theses two sections)*

Answer: We agree to rephrasing the first sentence. But, putting the two versions next to each other does not make sense since the names of the classes and methods are not the same and do not correspond to each other. We think it would be misleading for the reader. Anyway, since we shortened the manuscript, this table is now moved to the supplement.

Changes in manuscript: *Python code blocks and their cyclomatic complexity (CC) which ranks between C and F and their corresponding CC score.*

Flex_extract v7.1 – A software package to retrieve and prepare ECMWF data for use in FLEXPART

Anne Philipp[1,2], Leopold Haimberger[2], and Petra Seibert[3]

[1]Aerosol Physics & Environmental Physics, University of Vienna, Vienna, Austria
[2]Department of Meteorology and Geophysics, University of Vienna, Vienna, Austria
[3]Institute of Meteorology, University of Natural Resources and Life Sciences, Vienna, Austria

Correspondence: Anne Philipp (anne.philipp@univie.ac.at)

Abstract. Flex_extract is an open-source software package to efficiently retrieve and prepare meteorological data from the European Centre for Medium-Range Weather Forecasts (ECMWF) as input for the widely-used Lagrangian particle dispersion model FLEXPART and the related trajectory model FLEXTRA. ECMWF provides a variety of data sets which differ in a number of parameters (available fields, spatial and temporal resolution, forecast start times, level types etc.). Therefore, the selection of the right data for a specific application and the settings needed to obtain them are not trivial. ~~Therefore~~Consequently, the data sets which can be retrieved through flex_extract by both authorised ~~member state~~ member-state users and public users ~~and~~ as well as their properties are explained. Flex_extract 7.1 is a substantially revised version with completely restructured ~~software~~code, mainly written in Python3, which is introduced with all its input and output files and ~~for the four different~~ an explanation of the four application modes. Software dependencies and the methods for calculating the native vertical velocity $\dot{\eta}$, the handling of flux data and the preparation of the final FLEXPART input files are documented. Considerations for applications give guidance with respect to the selection of data sets, caveats related to the land-sea mask and orography, etc. Formal software ~~quality assurance~~ quality-assurance methods have been applied to flex_extract. ~~It comes with a~~ A set of unit and regression tests as well as code metric data are also supplied. A short description of the installation and usage of flex_extract ~~as well as information about available detailed~~ is provided in the appendix. Information about an additional on-line documentation for future up-to-date documentation is also ~~provided.~~ available.

1 Introduction

The widely used off-line Lagrangian particle dispersion model (LPDM) FLEXPART (Stohl et al., 1998; Stohl et al., 2005; Pisso et al., 2019) and its companion, the trajectory model FLEXTRA (Stohl et al., 1995; Stohl and Seibert, 1998), require meteorological data in GRIB format as input. A software ~~tool~~package, flex_extract, is provided to retrieve and prepare these data from the Meteorological Archival and Retrieval System (MARS) of the European Centre for Medium-Range Weather Forecasts (ECMWF) to run FLEXPART. Because of specific requirements of FLEXPART and FLEXTRA and the variations between the various ECMWF products, this is a complex task.

After the retrieval of the meteorological fields, flex_extract calculates, if necessary, the vertical velocity in the native coordinate system of ECMWF's Integrated Forecast ~~Model~~ System (IFS), the so-called hybrid coordinate (Simmons and Burridge,

1981); furthermore, it calculates approximate instantaneous fluxes from the accumulated flux data provided by the IFS (precipitation and surface fluxes of momentum and energy). It also takes care of packaging and naming the fields as expected by FLEXPART and FLEXTRA. The retrieval software is an integral part of the FLEXPART / FLEXTRA ~~modeling~~ modelling system which is needed by users who apply the main branch based on the ECMWF meteorological fields (Pisso et al., 2019).

Flex_extract is an open-source software package with a history starting in 2003 which has undergone adaptations and extensions ever since. After the release of version 7.0.2, which was very specific as it could retrieve data only from a subset of ECMWF's products, the demand for additional data sources and to adapt to new versions of ECMWF's software packages used arose. Unfortunately, the existing code was not very flexible and thus difficult to maintain and expand. User friendliness was insufficient, as knowledge about flex_extract's driving parameters, the various ECMWF data sets, and their interaction was expected from users; with the increasing popularity of the FLEXPART model, improvements were necessary also in this respect. One of the priorities was to enable the extraction of fields from the reanalysis data sets ERA5 and CERA-20C. Additionally, the need for retrieving ensemble members in combination with forecast products arose. A recently developed new algorithm for ~~disaggregating~~ disaggregation of the precipitation fields (Hittmeir et al., 2018) to improve the wet deposition calculation in FLEXPART should also be considered. With respect to ECMWF software packages on which flex_extract depends ~~, the Meteorological Interpolation and Regridding library MIR superseded the previous EMOSLIB,~~ a package called ~~eccodes~~ecCodes replaced `GRIB_API` `GRIB-API` for decoding and encoding GRIB messages.

Recently, ECMWF opened the access to selected reanalysis data sets for ~~non-authorised~~non-member-state users, so-called public users from anywhere in the world, while ~~before~~ previously only users with a member-state account could access the data. Along with this change, two new web interfaces (~~ECMF's WebAPI~~ ECMWF's Web API and the Copernicus Data Service [CDS] ~~WebAPI~~Web API) were introduced, which allow to download data without direct access to ECMWF servers. This required a further adaptation, so that flex_extract can now be used also on a local host in combination with these APIs for both ~~authorised~~ member-state and public users.

All these developments led to the new and totally revised version 7.1 of flex_extract introduced in this software description paper. It constitutes a more significant change of the code base than one might expect from the version number increment. The code was modularised ~~for implementing~~ in order to implement software quality standards and as a ~~prerequsite~~ prerequisite of the extension of the functionality. A more comprehensive set of test cases was developed, the documentation was significantly enhanced with more details. A big step forward was thus achieved in terms of user friendliness.

This paper contains the first ~~published software~~ documentation of flex_extract ~~. The current~~ published in open literature and is valid for the – currently latest – version 7.1~~replaces all previous versions which will no longer be supported.~~ .2.

1.1 ~~Structure of the paper~~FLEXPART and FLEXTRA

~~Section 2 gives an overview of available ECMWF data sets and their accessibility for member-state and public user , respectively. The diversity of available data sets, possible combinations, and accessibility is a key piece of information for users. The code of~~

The FLEXible PARTicle model (FLEXPART) is one of the most widely used Lagrangian particle dispersion models (LPDM) for multi-scale atmospheric transport studies (Stohl et al., 1998; Stohl et al., 2005; Pisso et al., 2019) with a world-wide user base. It is an open-source model under the GNU General Public Licence (GPL) Version 3. As an off-line model, it requires meteorological fields (analysed or forecast) as input. Such data are available from numerical weather prediction (NWP) models and thus several model branches have been created for input from different models (Pisso et al., 2019). The main branch of the FLEXPART model is able to use data from the ECMWF's IFS and the US National Centers for Environmental Prediction's (NCEP) Global Forecast System (GFS). The software package flex_extract ~~is described in Section 3. This is followed by considerations for application in Section 4, and the methods applied for the quality assurance in Section 5. The final remarks in Section 6 include information support options for users and plans for future development. The instructions for the installation and usage of the software are outlined in the Appendix~~supports the extraction of ECMWF/IFS data, considered to be the most accurate data source, as ECMWF is one of the leading global weather forecast centres and provides data on model-level and at high time resolution. As an LPDM, FLEXPART solves a Langevin equation for the trajectories of computational particles under the influence of turbulence (stochastic component) and quantifies changes to the trace substance mass or mixing ratio represent by these particles due to various processes.

Applications include a wide range of topics, such as air pollution, natural and man-made atmospheric radioactivity, stratosphere-troposphe exchange, and atmospheric water cycle studies and airflow patterns. With the domain-filling mode the entire atmosphere can be represented by particles representing an equal share of mass.

FLEXTRA is a model that calculates simple trajectories as a function of fields of the mean 3D wind (Stohl et al., 1995; Stohl and Seibert, . FLEXPART is based on it and some code goes back to the same original routines from FLEXTRA. FLEXTRA ingests the same input fields in `GRIB` format as FLEXPART, thus it may be considered as a companion model. It is also free software and can be downloaded as well from the FLEXPART community web site.

Both FLEXTRA (v5.0) and FLEXPART (v9.02) can be used from within ECMWF's Metview software (ECMWF, 2019m).

1.2 The history of flex_extract

When the FLEXTRA model was developed in the ~~1990ies~~1990s, one aim was to optimise its accuracy by avoiding unnecessary vertical interpolation. Therefore, it was implemented to directly use the three-dimensional wind fields on the IFS model levels rather than fields interpolated to pressure levels as most other off-line trajectory and particle dispersion models do (Stohl et al., 1995; Stohl and Seibert, 1998). This also solves the issue of the lower boundary conditions over topography (trajectories should not ~~interSect.~~ intersect the surface) in an optimum way. The IFS model uses a hybrid coordinate system, terrain-following near ground and approaching a pressure (p) based coordinate towards the model top; the vertical coordinate is called η and thus the corresponding native vertical velocity is $\dot{\eta}$.

At that time, most ECMWF/IFS model fields were available on η-levels, however, $\dot{\eta}$ was not routinely stored in the MARS archive. Thus, a ~~preprocessing software~~ pre-processing tool was needed to calculate accurate $\dot{\eta}$ values from available fields. A second motivation was the need of a chemical transport model (POP model, see Wotawa et al. (1998)) coupled with FLEXTRA

and later ~~of~~ on FLEXPART for instantaneous surface fluxes (latent and sensible heat, surface stresses, precipitation) instead of accumulated values of these fluxes as stored in MARS.

When the Comprehensive Nuclear Test Ban Treaty Organization (CTBTO) started to use FLEXPART operationally, ~~a new version was created. It still consisted~~ it became necessary to adapt the extraction software (consisting of Korn shell scripts and Fortran programmes for the numerically demanding calculation of $\dot{\eta}$~~. This was~~) such that it could be incorporated into ECMWF's automatic data dissemination system. This became the first numbered version of flex_extract, v1, released in 2003. In version 2 (2006), it became possible to extract ~~subregions~~ sub-regions of the globe and the Fortran code was parallelised with OpenMP. In Version 3, the option to use $\dot{\eta}$ from MARS, which became available for some forecast products from 2008 on, was introduced. Version 4 was ~~neededed~~ needed to adapt the ~~software~~ package to the then new GRIB2 standard for meteorological fields. Versions 5 and 6 (2013) where adaptations to allow for higher horizontal resolutions and additional data sources, e. g. global reanalysis data. At this time, the Korn shell scripts had become quite complicated and difficult to maintain.

In 2015, the demand was raised to retrieve fields from long-term forecasts, not only analyses and short-term forecasts. At this stage, it was decided to rewrite flex_extract in Python2. The Python part controls the program flow by preparing ~~and~~ korn shell scripts which are then submitted to ECMWF ~~servers~~ batch queue to start flex_extract in batch mode. The Fortran program for the calculation of the vertical velocity, `calc_etadot.f90` `calc_etadot` (previously also called `CONVERT2` ~~or~~ `preconvert` `CONVERT2` or `preconvert`), was still used and called from the ~~python~~ Python code. Version 7.0.3 allowed to retrieve CERA-20C and ERA5 data, and introduced local retrieval of MARS data through the ECMWF Web API. Version 7.0.4 enabled the retrieval of multiple ensemble members at a time ~~. It includes also~~ and included bug fixes for the retrieval of ERA5 and CERA-20C data.

For the current version 7.1, the Python part was completely revised by refactoring and modularisation, and it was ported to Python3. Instead of ECMWF's `grib_api` `GRIB-API` for decoding and encoding GRIB messages, its successor `ecCodes` ~~was implemented~~ `ecCodes` was utilised. The installation process has been simplified. In addition to the ECMWF Web API, also the new CDS API is supported. The disaggregation of precipitation data offers to alternatively use the new algorithm of Hittmeir et al. (2018) which maintains non-negativity and preserves the integral precipitation in each time interval. The Fortran part underwent some mostly cosmetic changes (source format, file names, messages, etc. and a minor bug fix) and an overhaul of the makefiles. The code quality of flex_extract was improved by adding a first set of unit tests and the introduction of regression tests. A new, detailed on-line documentation was created with Sphinx / FORD, hosted on the FLEXPART community ~~website~~ web site http://flexpart.eu/flex_extract.

1.3 ~~FLEXPART and FLEXTRA~~Structure of the paper

~~The FLEXible PARTicle model (FLEXPART) is one of the most widely used Lagrangian particle dispersion models (LPDM) for multi-scale atmospheric transport studies (Stohl et al., 1998; Stohl et al., 2005; Pisso et al., 2019) with a world-wide user base. It is an open source model under the GNU General Public License (GPL) Version 3. As an off-line model, it requires meteorological fields (analysed or forecast) as input. Such data are available from numerical weather prediction (NWP) models and thus several model branches have been created for input from different models (Pisso et al., 2019). The main branch of the~~

FLEXPART model is able to use data from the European Centre for Medium-Range Weather Forecasts' (ECMWF) Integrated Forecast System (IFS) and the US National Centre for Environmental Prediction's Global Forecast System (GFS). The data extraction software Section 2 gives an overview of available ECMWF data sets and their accessibility for member-state and public user, respectively. The diversity of available data sets, possible combinations of parameter settings, and accessibility is a key piece of information for users. The code of flex_extract supports the ECMWF/IFS data extraction, which is generally considered the most accurate data source. As an LPDM, FLEXPART solves a Langevin equation for the trajectories of computational particles under the influence of turbulence (stochastic component) and quantifies changes to the trace substance mass or mixing ratio represent by these particles due to various processes. All details can be found in the literature mentioned.

Applications include a wide range of topics, such as air pollution, natural and man-made atmospheric radioactivity, stratosphere-troposphe exchange, and atmospheric water cycle studies and airflow patterns. With the domain-filling mode the entire atmosphere can be represented by particles representing an equal share of mass.

FLEXTRA is a model that calculates simple trajectories as a function of fields of the mean 3D wind Stohl et al. (1995); Stohl and Seibert . FLEXPART is based on it and shares some parts of the code. It ingests the same input fields in GRIB format as FLEXPART, thus it may be considered as a companion model. It is also free software and can be downloaded as well from the FLEXPART community website.

Both FLEXTRA and FLEXPART can be used from within ECMWF's Metview software ECMWF (2019m)is described in Sect. 3. This is followed by considerations for application in Sect. 4, and the methods applied for the quality assurance in Sect. 5. The final remarks in Sect. 6 include information support options for users and plans for future development. The technical instructions for the installation and usage of the software package are outlined in the Appendix.

2 ECMWF data

The European Centre for Medium-Range Weather Forecasts ECMWF produces reanalysis data sets and global numerical weather predictions in operational service to its supporting Member States. All data are available to the national meteorological services in the Member States and the Co-operating States. Some data sets are also publicly available (ECMWF, 2019a). The data are stored in GRIB or BUFR format in the Meteorological Archival and Retrieval System (MARS) ECMWF (2019b) MARS (ECMWF, 2019b). The smallest addressable object is a meteorological field or an observation, grouped into logical entities such as "a forecast". These entities can be addressed through metadata meta data organised in a tree-like manner. The meteorological fields are archived in one of three spatial representations: spherical harmonics (mainly model level fields), Gaussian grid (mainly surface fields, but also some model level fields), or a regular latitude / longitude grid (ECMWF, 2019b).

2.1 Access to ECMWF

For the access to its MARS archive, ECMWF distinguishes two users user groups: member-state and public users.

Member-state users have the possibility to work directly on the ECMWF Member State Linux servers as well as via a Web Access Toolkit (ECaccessECaccess) through a Member State gateway server. This mode provides full access to the MARS

archive. Nevertheless, there might be some limitations in user rights, particularly regarding current forecasts and ensemble forecasts. Member-state user accounts are granted by the Computing Representative of the corresponding Member State.

Public users access the ECMWF public data sets directly from their local facilities, anywhere in the world. The main differences to the ~~member state~~ member-state users are the method of access – through a Web API – and the limited availability of data. Public users have to explicitly accept the ~~license~~ licence for the data set to be retrieved.

Member-state users may also access data via a Web API, without a gateway server, in the same way as public users. The only difference is that different MARS databases are ~~to be used~~utilised. Flex_extract automatically chooses the correct ones.

Users can explore the availability of data in MARS via a web interface where they are guided through a stepwise selection of ~~metadata~~meta data. With this method, it is also possible to estimate the download size of a data set before actually retrieving it through flex_extract. There is a web interface "MARS Catalogue" for member-state users[1] with the full content and an interface "Public data sets" for public users[2] with the subset of public data. The availability of data can also be checked by MARS commands on ECMWF servers. MARS commands[3] are used by flex_extract to retrieve the data on ECMWF servers.

2.2 Data sets available through flex_extract

ECMWF has a large variety of data sets varying in model physics, temporal and spatial resolution as well as forecast times. Only the subset of data which are most commonly used with FLEXPART can be retrieved through flex_extract. The accessible data sets are:

1. The operational deterministic atmospheric forecast model (DET-FC), nowadays called atmospheric high-resolution forecast model (HRES),
2. the operational atmospheric ensemble forecast (ENS),
3. the ERA-Interim reanalysis,
4. the CERA-20C reanalysis, and
5. the ERA5 reanalysis.

Public users have access to the public version of ERA-Interim ~~and~~ (Berrisford et al., 2011), CERA-20C ~~Laloyaux et al. (2018) reanalysis. Even though ERA5 is in principle a public data set, currently only a subset can be accessed by public users. Unfortunately, it does not include the model-level fields which are essential for FLEXPART. Flex_extract is already prepared (requires only some code activation) to retrieve public~~ (Laloyaux et al., 2018) and ERA5 ~~data as soon as model-level fields will be available.~~ (Hersbach et al., 2020) reanalysis.

The retrievable data sets are identified by the key meta data listed in the "Identification" section of Table 1. The relevant data period for each data set is also listed. Furthermore, the table presents the available temporal and spatial resolution as well as

[1]https://apps.ecmwf.int/mars-catalogue/; Last accessed: 17.08.2019
[2]https://apps.ecmwf.int/datasets/; Last accessed: 17.08.2019
[3]https://confluence.ecmwf.int/display/UDOC/MARS+command+and+request+syntax; Last accessed: 17.08.2019

the number of ensemble members (may change in the future for the operational data). The availability of $\dot{\eta}$ is important for the mode of preparing the vertical velocity fields (see Sect. 3.7) and is therefore marked for accessibility as well. With the current operational data, a temporal resolution of 1 h can be established with a well-selected mix of analysis and forecast fields (see Sect. 4). The horizontal grid type refers to the ~~way how it fields are archived in MARS~~spatial representation. Table 4 provides the relationship between corresponding spectral, Gaussian and latitude / longitude grid resolutions.

In this paper, we collect the essential changes in forecast steps and spatial resolution since the first IFS release, as they need to be known for using flex_extract. Table 2 lists the evolution of horizontal and vertical resolutions for all operational data sets. The evolution of the forecast steps and the introduction of additional forecast times in "DET-FC" and "ENS-CF" are summarised in Table 3.

The reanalysis data sets are naturally more homogeneous. Nevertheless, they all have their individual characteristics, making the selection process with flex_extract complex. Table 1 provides an overview of the main differences in the reanalysis meta data. ERA-Interim has a 3-hourly resolution with an analysis and forecast field mix in the full access mode but only a 6-hourly resolution for public users. It lacks the $\dot{\eta}$ fields which makes the retrieval of ERA-Interim computationally demanding (Sect. 3.7). The ERA5 and CERA-20C reanalyses can be retrieved with 1 h resolution and include ensembles; however, ERA5 ensemble members are not yet retrievable with flex_extract and therefore omitted in the tables. Even though the availability of 1-hourly analysis fields means that forecast fields are not required for most of the variables, accumulated fluxes are only available as forecasts. One should also pay attention to ~~differen~~ different forecast start times in both data sets and the complication ~~inplied~~ implied by forecasts starting from 18 UTC as the date will change until the subsequent start time; see also Sect. 3.6.

With the establishment of the Copernicus Climate Change Service (C3S) in March 2019, a new channel for accessing ECMWF reanalysis data, most prominently ERA5 (Hersbach et al., 2020), has been opened. At the same time, access to this data set via the ECMWF Web API was cancelled. While access directly from ECMWF servers is not affected, in local retrieval modes now one has to submit requests to the Copernicus Climate Data Store (CDS), which uses another Web API called CDS API; in the background, this API retrieves the public data from dedicated web servers for faster and easier access. Unfortunately, only surface and pressure level data are available in CDS at the moment; this might change in the future.

~~In the case of member-state users, it~~ It is possible to pass the request for model levels to the MARS archive even through the CDS interface. ~~Flex~~This is done automatically since flex_extract is ~~already modified to use thisAPI so a member user can already retrieve ERA5 data~~configured to do this. However, experience shows that ~~the performance of~~ this access mode is ~~not good~~very slow[4], thus currently it ~~cannot be recommended~~is not recommend for member-state users.

3 Software description and methods

The flex_extract software package allows to retrieve and prepare the meteorological input files from ECMWF for FLEXPART (and FLEXTRA) easily and in an automated fashion. The ~~ecessary~~ necessary meteorological parameters to be retrieved are

[4]https://confluence.ecmwf.int/display/CKB/How+to+download+ERA5; Last accessed: 22.06.2020

predefined according to the requirements of FLEXPART and the characteristics of various data sets. The post-processing after retrieval for the calculations of the flux fields (Sect. 3.6) and the vertical velocity (Sect. 3.7) is also ~~incluced~~included.

The actions executed by flex_extract (also called "the software package" henceforth) depend on the user ~~access mode~~ group (see Sect. 2.1), the ~~application mode~~location of execution, and the data to be retrieved. There are three possible ~~application modes, using~~ locations of execution, namely the ECMWF Member State Linux servers, the Member State Gateway server, or a local host. As not all combinations are possible, a total of four different application modes ~~result~~ results which are described in Sect. 3.1. Because of the ~~depencies~~ dependencies of flex_extract, the respective application environments need to be prepared in different ways as described in ~~Section~~ Sect. 3.2. The software package comprises a Python part for the overall control of the processing, including the data extraction, a Fortran part for the calculation of the vertical velocity, korn-shell scripts for batch jobs to run on ECMWF servers, and bash shell scripts as a user-friendly interface to the Python code. Available settings and input files are described in Sect. 3.4. The output files are divided into temporary files (Sect. 3.8) which are usually deleted at the end and the final output files (Sect. 3.9) which serve as FLEXPART input. An overview of the program structure and the work flow together with an example is given in Sect. 3.3.

~~A general overview of the~~ The structure of the flex_extract root directory is ~~provided~~ presented in Table 5; it is completely different than in previous versions. The installation script `setup.sh` is directly stored under the root directory together with basic information files. `Source` contains all Python and Fortran source files, each in a separate directory. Flex_extract works with template files, stored in `Templates`. The on-line documentation is included in `Documentation` so that it can also be read off-line. The actual work by users takes place in the `Run` directory. There are the `CONTROL_*` files in the `Control` directory, the ~~korn~~ korn shell job scripts in `Jobscripts` and, in the case of applying the ~~*local mode*~~local mode, also a `Workspace` directory where the retrieved GRIB files and final FLEXPART ~~output~~ input files will be stored. The `ECMWF_ENV` file is only created for the ~~*remote* and *gateway mode*~~remote and gateway mode; it contains the user credentials for ECMWF servers. The `run.sh` and `run_local.sh` scripts are the top-level scripts to start flex_extract. Like in the previous versions, users can also directly call the `submit.py` script. There is also a directory `For_developers` which contains the source files of the ~~online~~ on-line documentation, source files for figures, and sheets for parameter definitions.

3.1 Application modes

Arising from the two user groups described in ~~section~~ Sect. 2.1 and the three possible locations of application, three different user application modes are defined, namely ~~`Remote, Gateway` and `Local mode`~~"remote", "gateway" and "local" mode. However, the ~~`Local mode`~~ local mode is further split in the ~~`Local member` and the `Local public`~~ "local member" and the "local public" mode. A summary of the necessary registration method per mode and user group is outlined in Table 6. An overview of locations and modes is sketched in Figure 1 and a definition is given in the following list:

Remote (member) Users work directly on ECMWF Linux Member State servers, such as `ecgate` or `cca/ccb`. The software will be installed and run in the users `$HOME` directory. Users do not need to install any of the additional library pack-

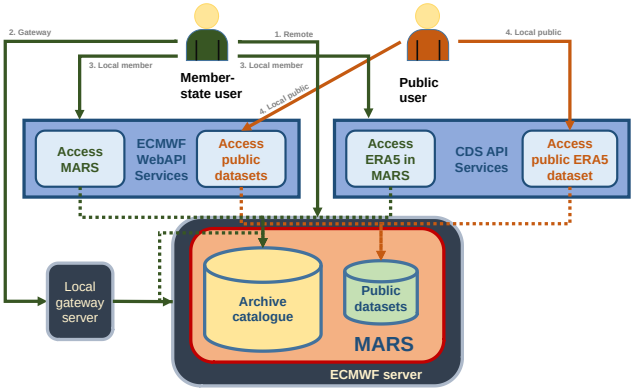

Figure 1. Schematic overview of access methods to the ECMWF MARS archive implemented in flex_extract.

ages mentioned in ~~Section~~ Sect. 3.2 since ECMWF provides everything with ~~a module system~~an environment modules framework. Flex_extract takes care of loading the necessary modules.

Gateway (member) This mode is recommended in the case a local Member State Gateway server is in place (ECMWF, 2019j) and the user has a member-state account. Job scripts would then be prepared locally (on the gateway server) and submitted to the ECMWF Linux Member State server via the ECMWF web access toolkit `ECaccess`. The actual data extraction and post-processing is then done at the ECMWF servers and the final data are, if selected, transferred back to the local gateway server. The installation script of flex_extract must be executed at the local gateway server. However, this will install flex_extract in the users `$HOME` directory on the ECMWF server and some extra setup is done in the local gateway version. For instructions about establishing a gateway server, please consult ECMWF (2019j) directly. The necessary software environment has to be established before installing flex_extract.

Local member Member-state users work on their local machines which require a similar software environment as the one on ECMWF servers plus the provided Web API's as the interface for the MARS archive.

Local public Public users can work on their local machines~~by not only preparing the general software dependencies but also adding~~ , having fulfilled the software dependencies and having added the ECMWF Web API ~~as the interface~~ and the CDS API as the interfaces to the public MARS archive. In this case, a direct registration at ~~the ECMWF~~ ECMWF and CDS is necessary, and all users have to accept a specific ~~license~~ licence agreement for each data set which is intended to be retrieved.

3.2 Software dependencies

~~The software~~ Software required to run flex_extract depends on the application mode. Basic requirements for all application modes are listed in Table ~~??. The *local mode*~~ 7. The local mode requires in addition Python packages `ecmwf-api-client`

and / or `cdsapi`, depending on the data set to be retrieved, to connect to the MARS archive as Table 6 shows. Users should make sure that all dependencies are satisfied before starting the installation. ~~The software~~ Flex_extract is tested only in a GNU / Linux environment, although it might be possible to use it also under other operating systems.

3.3 Program structure

The work of flex_extract ~~can~~ can be decomposed into the following three separate tasks:

1. Setting the parameters controlling the retrieval and the data set:
   Reading of the `CONTROL` file, ~~command line~~ command-line arguments, and ECMWF user credential file (in the case of ~~*remote* or *gateway*~~ remote or gateway mode). Depending on the application mode, flex_extract prepares a job script which is sent to the ECMWF batch queue, or proceeds with the tasks 2 and 3.

2. Retrieve data from MARS:
   MARS requests are created in an optimised way (jobs split with respect to time and parameters) and submitted. Retrieved data are arranged in separate GRIB files. If the parameter `REQUEST` was set to 1, the request is not submitted and only a file `mars_requests.csv` is created. If it is set to 2, this file is created in addition to retrieving the data.

3. Post-process retrieved data to create final FLEXPART input files:
   After all data are retrieved, flux fields are disaggregated, and vertical velocity fields are calculated by the Fortran program `calc_etadot`. Finally, the GRIB fields are merged into a single GRIB file per time step with all the fields FLEXPART expects. Depending on the parameter settings, file transfers are initiated and temporary files deleted.

In task 1, the ~~software differentiates depending~~ execution of the code depends on the application mode. In the case of ~~*remote* or *gateway*~~ remote or gateway mode (see also Fig. 2), the job script for the ECMWF batch system is prepared and submitted to the batch queue. The program finishes with a message to standard output. In the case of the *local* application mode, the work continues locally with tasks 2 and 3, as illustrated in Figure 3.

~~Each application mode has its unique process steps and its own connection to the MARS archive.~~ Figure 4 demonstrates the involved input files, execution scripts and connection methods as well as the locations where each step takes place. ~~In the *gateway mode*, the setup task will be done on the gateway server and the created jobscript is sent via the `ECaccess` command to the batch queue before flex_extract terminates. As soon as~~

The remote and gateway mode both create a job script using the command-line parameters and the content of the specified `CONTROL` file and then sent it to an ECMWF batch queue. In remote mode this happens on an EMCWF server while the gateway mode uses the local gateway server for the creation and submission of the job. As the job script is ~~processed,~~ executed from whichever of the two modes, it creates the job environment ~~is created and flex_extract is started in *local mode*, reading the prepared `CONTROL` file, extraction of data, and post-processing tasks. In this mode, the extraction is done with a MARS command. If it was selected, the~~ (in particular, the working directory) and starts `submit.py` to retrieve and post-process the data. Note that this locally started instance of `submit.py` triggers the work flow of the local mode but uses the MARS client

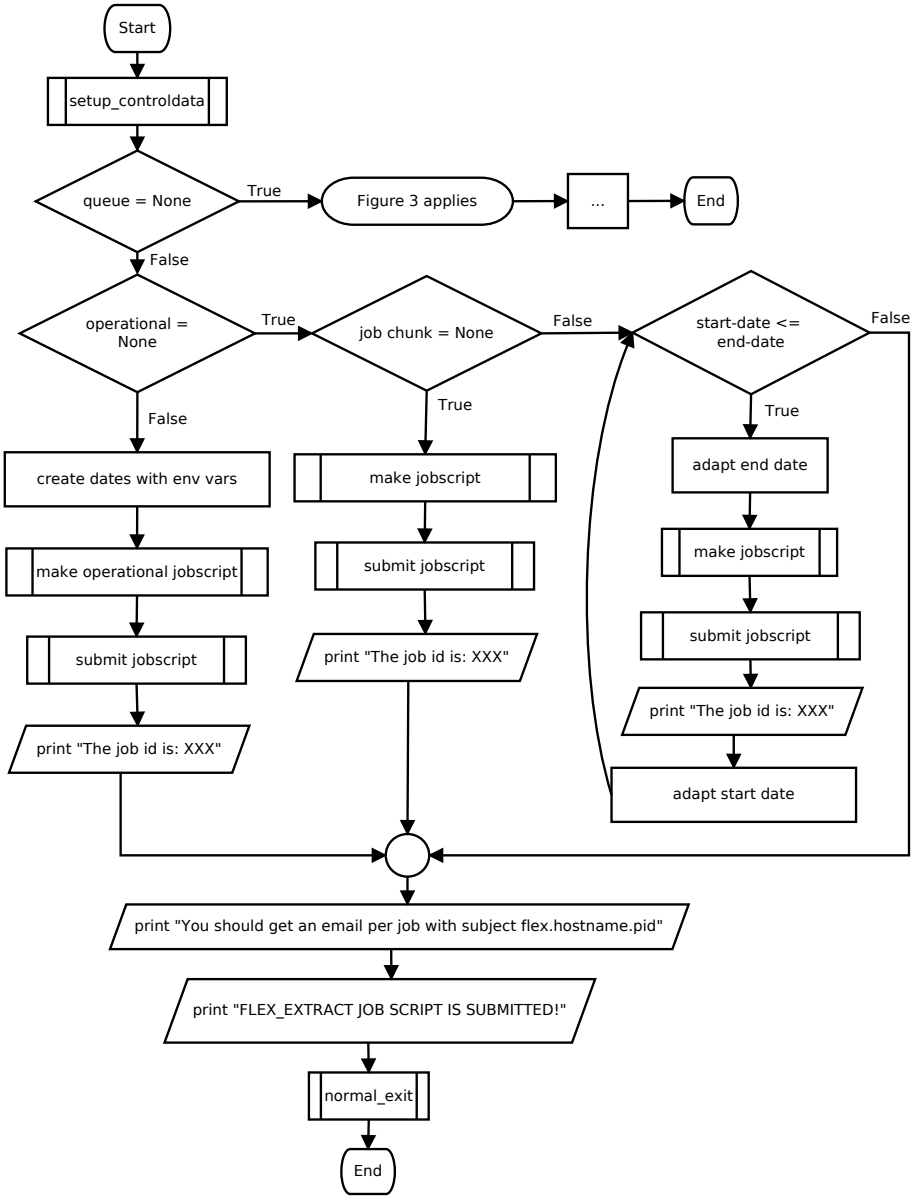

Figure 2. Flow diagram for the ~~*remote*~~ remote and ~~*gateway*~~ gateway mode. ~~The~~ A job script is created and submitted to the batch queue on ~~the~~ an ECMWF server. The job script will then ~~start~~ be executed on the ECMWF server ~~and apply~~ to start flex_extract ~~in *local* mode. In the case~~ again for retrieving and post-processing of the data. The branch indicated by `queue = None` ~~,~~ refers to the ~~local mode is selected and work flow shown in~~ Figure 3~~applies as is indicated by the branch *do local work*~~. Trapezoidal boxes mark standard output, simple rectangles mark the execution of sequential instructions, and the rectangles with a side border mark the execution of subroutines. The boxes in diamond form indicate decisions.

to extract the requested fields from the database. The final output files are sent to the local member-state gateway server ~~. The *remote mode* works completely on ECMWF servers but has the same process sequence as the *gateway mode*. In the *local mode*, all work is done on the local host , except the data extraction which is done by an HTTP request via the Web APIs. The data are sent back to~~ only if the corresponding option was selected in the `CONTROL` file. When flex_extract is used on a local host and in local mode, fields are extracted from MARS using one of the Web API's (which sends HTTP requests to ECMWF/CDS) and are received by the local host ~~instantly~~without storage on ECMWF servers.

3.4 Input files

3.4.1 The `CONTROL` file

Flex_extract needs a number of controlling parameters. They are initialised by flex_extract with their default values and will be overwritten by the settings in the `CONTROL` file. Only those parameters which deviate from the default values have to be provided. It is necessary to understand these parameters and to set them to proper and consistent values. They are listed in ~~Tables 8 and 8~~ Table 8 with their default values and a short description. More detailed information, hints about the conditions of settings, and possible value ranges are available in the supplemental material~~and partially in Section 4. The files are read during task 1. Only those parameters which which deviate from the default values have to be provided. The file `CONTROL.documentation` provides a collection of the available parameters in grouped sections together with their default values. Users can start from this file to define their setup or use one of the sample application `CONTROL` files as a template (in `flex_extract_v7.1/Run/Control/`). These samples correspond to the data sets described in Section 2.2. One example for each data set is provided with some variations in resolution, type of field, or method for the calculation of the vertical verlocity. The naming convention is `CONTROL_<dataset>.optionalIndications`, where the `optionalIndications` is an optional string to provide further characteristics about the retrieval method or the data set. For the operational data sets (OD) this string contains information of the stream, the field type of forecasts, the method for extracting the vertical velocity and other aspects such as time or horizontal resolution~~, partially in Sect. 4 and the on-line documentation.

Regarding the file content, the first string in each line is the parameter name, the following string(s) (separated by spaces) are the parameter values. The parameters may appear in any order, with one parameter per line. Comments can be added as lines beginning with #-sign, or after the parameter value. Some of these parameters can be overruled by ~~command line~~ command-line parameters provided at program call. ~~In earlier versions, each parameter name contained the leading string `M_`; this was removed for version 7.1 but is still possible for compatibility. The grid resolution had to be provided in 1/1000 of a degree before, while now it can be also provided as a decimal number with unit degree. Flex_extract is able to check for correct setting of the `GRID` parameter with the domain-specific settings.~~

~~It is now also possible to reduce the number of data values for the combination of~~ The naming convention is ~~`TYPE`~~`CONTROL_<dataset` ~~`TIME` and `STEP` parameter combination to the actual temporal resolution. Previous versions expected to have 24 values per~~

parameter, one for each hour of the day, even though a 3-hourly temporal resolution was selected as shown in the following example:

```
DTIME 3
TYPE AN AN AN AN ... AN AN AN AN
TIME 00 01 02 03 ... 20 21 22 23
STEP 00 00 00 00 ... 00 00 00 00
```

The more intuitive solution of providing just the data for the time steps to be retrieved leads, for example, to eight data values per parameter for a 3-hourly retrieval as shown in the next example:

```
DTIME 3
TYPE AN AN AN AN AN AN AN AN
TIME 00 03 06 09 12 15 18 21
STEP 00 00 00 00 00 00 00 00
```

or four values for a 6-hourly retrieval.

```
DTIME 6
TYPE AN AN AN AN
TIME 00 06 12 18
STEP 00 00 00 00
```

The only necessity is the consistent setting of the `DTIME` parameter to define the temporal resolution. For backward compatibility, this means that `DTIME` can be coarser than the number of temporal points provided in `TYPE`, `TIME` and `STEP`, but not finer where the `optionalIndications` is an optional string to provide further characteristics about the retrieval method or the data set. See Sect. 4 for more details and examples.

3.4.2 User credential file `ECMWF_ENV`

In the *remote* and *gateway* mode, the software remote and gateway mode, flex_extract sends job scripts to the batch system of an ECMWF server, thus it is necessary to provide the user and group name which are given in file `ECMWF_ENV`. Additionally, this file provides the name of the local member-state gateway server and the destination so that unattended file transfer[5] (`ectrans`) between ECMWF and member gateway servers can be used. The destination is the name of the so-called `ectrans` association; it has to exist on the local gateway server.

3.4.3 Template files

Some files are highly variable depending on the setting in the other input files. They are created during run time by using template files. The templates are listed in Table 10. Flex_extract uses the Python package `genshi` to read the templates and

[5] https://confluence.ecmwf.int/display/ECAC/Unattended+file+transfer+-+ectrans; Last accessed: 09.09.2019

substitute placeholder by values. These placeholders are marked by a leading $ sign. In the case of the korn shell job scripts, where (environment) variables are used, the $ sign needs to be escaped by an additional $ sign. Usually, users do not have to change these files.

3.5 Executable scripts

3.5.1 Installation

The ~~installation of flex_extract is done by the~~ shell script `setup.sh`, which is located in the root directory of flex_extract~~.~~ ~~This script sets all available command line arguments for the installation , does~~ , installs flex_extract. It defines the installation parameters which are defined in Table 9 and Table 11 and applies some plausibility checks ~~and finally~~ before it calls the Python script `install.py`. ~~Users are supposed to provide these arguments according to their environment and application mode. Parameters which must be set for all application modes are defined in Table 11.~~ The Python script does ~~all necessary operations~~ the installation depending on the selected application mode. In the case of ~~*remote* and *gateway*~~ remote and gateway mode, the `ECMWF_ENV` file is created~~(settings of related parameters to be done in `setup.sh`, see Table 9)~~, the job script template ~~is prepared,~~ `submitjob.template` is prepared and stored in the `Templates` directory and the korn shell script for compiling the Fortran source code `compilejob.ksh` is created. After these preparations, a tar ball with the core content is created and copied to the target location (ECMWF server or local installation path). Next, the `compilejob.ksh` is submitted to the batch system of ECMWF servers via `ECaccess` commands, or just untar-ed at the local target location. It compiles the Fortran code, prepares the work environment at ECMWF servers and, in the case of ~~*remote / gateway*~~ remote/gateway mode, a log file is sent to the user's email address.

3.5.2 Execution

~~In earlier versions of flex_extract, an extraction was initiated~~ The shell script `run.sh` or `run_local.sh` starts the whole procedure by calling the Python script `submit.py` with ~~suitable command line arguments~~ predefined command-line arguments (see Table 12) from a user section. The Python script constitutes as the main entry point and controls the program flow including the call of the Fortran program. ~~Now, it is still possible to work in that way, but a wrapper shell script `run.sh` is provided in addition. This shell script contains a user section where the Python command line arguments are to be set. This mode of operation is simpler for beginners, and it is useful for repetitive tasks, as no arguments can be forgotten. The available command line arguments are listed in Table 12.~~ Some of the ~~arguments occur only with~~ parameters in `run.sh` are only needed at the time of the program call, while others are also defined in the `CONTROL` file. In this case, the values in ~~`submit`~~`run.sh` take precedence over those from the `CONTROL` file.

The `submit.py` script interprets the ~~command line arguments (overview available through `./submit.py --help` from the Python source directory)~~ command-line arguments and, based on the input parameter `QUEUE`, it decides which application mode is active. In *local mode*, data are fully extracted and post-processed, while in the *remote* and *gateway mode*, a

korn shell script called `job.ksh` is created from the template `submitjob.template` and submitted to the ECMWF batch system~~via the gateway server.~~

~~The underlying template is `job.temp`, stored in the `Templates` directory. This template was generated in the installation process from `job.template` where some basic settings were done , which are the same for each flex_extract execution.~~ In case of the gateway mode, this is done via the local gateway server. The job script sets necessary directives for the batch system, creates the run directory and the CONTROL file, sets some environment variables (such as the `CONTROL` file name) and executes flex_extract. The standard output is collected in a log file which will be sent to the users' email address in the end. The batch system settings are fixed and they differentiate between the `ecgate` and the `cca`/`ccb` server systems to load the necessary modules for the environment when submitted to the batch queue. The `ecgate` server has directives marked with `SBATCH`[6] for the SLURM workload manager, the high performance computers `cca` and `ccb` have `PBS`[7] comments for PBSpro. The software environment dependencies mentioned in ~~Section~~ Sect. 3.2 are fulfilled by loading the corresponding modules. It should not be changed without further testing.

~~Just for completeness, there are two more entry points in the software which are provided for debugging. The Python scripts `getMARSdata.py` and `prepare_flexpart.py` are normally used as modules, but can also be used as an executable program. The `getMARSdata.py` script controls the complete extraction of ECMWF data, while the `prepare_flexpart.py` controls the complete post-processing.~~

3.6 Disaggregation of aggregated flux data

FLEXPART interpolates meteorological input data linearly to the position of computational particles in time and space ~~Stohl et al. (1998); S~~ (Stohl et al., 1998; Stohl et al., 2005). This method requires point values in the discrete input fields. However, flux data from ECMWF (as listed in Table 13) ~~from ECMWF represent cell averages or~~ represent cell integrals and are accumulated over a time interval ~~,~~ which depends on the data set. ~~Hence, to conserve~~ A pre-processing scheme is therefore applied to convert the accumulated values to point values valid at the same times as the main input fields while conserving the integral quantity with FLEXPART's linear interpolation~~, a pre-processing scheme has to be applied~~.

The first step is to de-accumulate the fields in time so that each value represents an integral in (x, y, t)-space. Afterwards, a disaggregation scheme is applied. While the horizontal cell values are simply ascribed to the cell centre, with respect to time, a more complex procedure is needed because the final values should correspond to the same time as the other variables. In order to be able to carry out the disaggregation~~procedure of Paul James~~, additional flux data are retrieved automatically for one day before and one day after the period specified. Note that these additional data are temporary and used only for disaggregation within flex_extract. They are not contained in the final FLEXPART input files. The flux disaggregation produces files named `fluxYYYYMMDDHH`, where `YYYYMMDDHH` is the date. Note that the first and last two flux files do not contain any data. Note that for operational retrievals which use the `BASETIME` parameter, forecast fluxes are only available until `BASETIME`, so that

[6]https://confluence.ecmwf.int/display/UDOC/Writing+SLURM+jobs; Last accessed: 10.09.2019
[7]https://confluence.ecmwf.int/display/UDOC/Batch+environment%3A++PBS; Last accessed: 10.09.2019

interpolation is not possible in the last two time intervals. This is the reason why setting BASETIME is not recommended for regular on-demand retrievals.

3.6.1 Disaggregation of precipitation in older versions

In versions 7.0.x and before, a relatively simple method was applied to process the precipitation fields, consistent with the linear temporal interpolation applied in FLEXPART for all variables. At first, the accumulated values are divided by the number of hours (i.e., 3 or 6). For the disaggregation, precipitation sums of four adjacent time intervals (p_a, p_b, p_c, p_d) are used to generate the new instantaneous precipitation (disaggregated) value p which is ~~valid at the boundary between time intervals1 and 2 as follows:~~ output at the central point of the four adjacent time intervals:

$$p_{ac} = \begin{cases} 0.5\ p_b \text{ for } p_a + p_c = 0 \\ \frac{p_b\ p_c}{p_a + p_c} \text{ for } p_a + p_c > 0 \end{cases} \tag{1}$$

$$p_{bd} = \begin{cases} 0.5\ p_c \text{ for } p_b + p_d = 0 \\ \frac{p_b\ p_c}{p_b + p_d} \text{ for } p_b + p_d > 0 \end{cases} \tag{2}$$

$$p = p_{ac} + p_{bd} \tag{3}$$

The values p_{ac} and p_{bd} are temporary variables. The new precipitation ~~values~~ value p constitute the ~~deaccumulated~~ de-accumulated time series used later in the linear interpolation scheme of FLEXPART. If one of the four original time intervals has a negative value, it is set to 0 prior to the calculation. Unfortunately, this algorithm does not conserve the precipitation within the interval under consideration, negatively impacting FLEXPART results as discussed by Hittmeir et al. (2018) and illustrated in Figure 5. Horizontally, precipitation is given as cell averages. The cell midpoints coincide with the grid points at which other variables are given, which is an important difference to the temporal dimension. FLEXPART uses bilinear interpolation horizontally.

3.6.2 Disaggregation for precipitation in version 7.1

Due to the shortcomings described above, a new algorithm was developed by Hittmeir et al. (2018). In order to achieve the desired properties (Hittmeir et al., 2018, p. 2513), a linear formulation with two additional supporting points within each interval is used. The current version of flex_extract implements this algorithm for the temporal dimension. Figure 6 shows how these requirements are fulfilled in the new algorithm for the simple case presented in Figure 5.

Flex_extract allows to choose between the old and the new disaggregation method for precipitation. In the latter case, the two additional sub-grid points are added in the output files. They are identified by the parameter ~~"step"~~ STEP which is 0 for the original time at the left boundary of the interval, and, respectively, 1 ~~or~~ and 2 for the two new sub-grid points. Filenames do not change. Current FLEXPART versions can not properly handle this input files generated with the new disaggregation scheme; they would use the third field (second additional sub-grid point in time), which would be worse than using the current method. The next minor version of FLEXPART to be released shall support the new scheme.

3.6.3 Disaggregation for the other flux fields

The accumulated values for the other ~~variables~~ fluxes are first divided by the number of hours, and then interpolated to the ~~exact times using a bicubic interpolation which conserves~~ times of the major fields. The algorithm was designed to conserve the integrals of the fluxes within each ~~timespan. Disaggregation uses~~ time interval when reconstructed with a cubic polynomial. It uses the integrated values F during four adjacent ~~time-spans~~ time intervals (F_0, F_1, F_2, F_3) to generate a new, disaggregated point value F ~~valid at the boundary between intervals1 and 2 as follows:~~ which is output at the central point of the four adjacent time intervals:

$$F_a = \frac{F_3 - F_0 + 3\,(F_1 - F_2)}{6} \qquad F_b = \frac{F_2 + F_0}{2} - F_1 - 9\,\frac{F_a}{2} \qquad F_c = F_1 - \frac{1}{12}F_0 - 7\,\frac{F_a}{2} - 2\,F_b \qquad F_d = F_0 - \frac{F_a}{4} - \frac{F_b}{3} - \frac{F_c}{2} \qquad F = 8\,F_a + 4\,\frac{7}{12}F_{b1} + \tag{4}$$

Note that a cubic interpolation was never implemented in FLEXPART. We therefore plan to replace this scheme by an adaption of the scheme used for precipitation, adapted to the situation where both positive and negative values are possible.

3.7 Preparation of vertical velocity

An accurate representation of the vertical velocity is a key component for atmospheric transport models. One of the considerations for the design of FLEXTRA was to work entirely in the native coordinate system of ECMWF's IFS model to minimise interpolation errors. This meant that the same hybrid η coordinate (terrain-following near ground, approaching pressure levels towards the model top) would be used, which implied to use the corresponding native vertical velocity ("*etadot*")

$$\dot{\eta} = \frac{\mathrm{d}\eta}{\mathrm{d}t} \tag{5}$$

rather than the more commonly used ordinary vertical velocity in a simple z-sytem (units of $\mathrm{m\,s^{-1}}$) or the vertical motion ω of pressure-based systems (unit $\mathrm{Pa\,s^{-1}}$). For reasons that we can't reconstruct, however, FLEXTRA did not use $\dot{\eta}$ strictly, but rather a quantity

$$\dot{\eta}_p = \frac{\mathrm{d}\eta}{\mathrm{d}t}\frac{\partial p}{\partial \eta} \tag{6}$$

which obviously has units of $\mathrm{Pa\,s^{-1}}$. The code calls this quanitity *etapoint*, not to be confused with *etadot*. Even though in FLEXPART this concept had to be abandoned in favour of a terrain-following z-system to allow a correct implementation of the Langevin equation for turbulent motion, FLEXTRA and FLEXPART share the same requirement for the vertical motion with respect to their input. Over many years, ECMWF would store only the post-processed ~~perssure~~ pressure vertical velocity ω=dp/dt. Transforming this back to $\dot{\eta}$, with approximations and interpolations involved in both operations, leads to vertical velocities that do not fulfill continuity. Therefore, $\dot{\eta}$ was reconstructed from the fields of divergence using the continuity equation, integrated from the model top downward as described in Simmons and Burridge (1981). In the IFS model, dynamical variables are horizontally discretised by spherical harmonics. It is best to do this on the reduced Gaussian grid that is used in IFS when a grid-point representation is required.

In September 2008, ECMWF started to archive the model's native vertical velocity fields ($\dot{\eta}$) for the operational analyses and forecasts. This allowed flex_extract to skip the cumbersome reconstruction and directly use this parameter. The amount of data that needs to be extracted from MARS, the CPU time and the memory requirements are all reduced substantially. The ERA5 and CERA-20C reanalyses also provide $\dot{\eta}$. Thus, even though it is possible to use the old method on new data sets, there is no reason to do so and it would be a waste of resources. It is, however, still kept in flex_extract to allow extraction of data from the older data sets, in particular ERA-Interim. In the following, the two methods are briefly characterised.

3.7.1 Reconstruction of the vertical velocity using the continuity equation

The most accurate algorithm for the reconstruction of the native vertical velocity requires the extraction of the horizontal divergence fields and the logarithm of the surface pressure in spectral representation (and thus always global, regardless of the final domain), their transformation to the reduced Gaussian grid (introduced by Ritchie et al. (1995)), on which the continuity equation is solved, a transformation back to the spectral space, and finally the evaluation on the latitude-longitude grid desired by users. Especially for high spectral resolution, this is a compute- and memory-intensive process that also takes time, even when making use of OpenMP parallelisation. Larger data sets can only be treated on the supercomputer (`cca`/`ccb`) but not on `ecgate`. The code for these calculations is written in Fortran90.

Alternatively, data can be extracted from MARS immediately on the latitude-longitude grid for the domain desired, and the continuity equation is then solved on this grid, but this method is not as accurate as the calculations on the Gaussian grid, particularly for higher spatial resolutions.

3.7.2 Preparation of the vertical velocity using archived $\dot{\eta}$

If the vertical velocity is available in MARS, it only needs to be multiplied with $\partial p/\partial \eta$. In the flex_extract version discussed here, this is done by the Fortran program whose functionality is described below.

3.7.3 Short description of the functionality of the `calc_etadot` code

A dedicated working directory is used where all input and output files are kept. Currently, the files have names of the form `fort.xx` where `xx` is some number.

The control file steering the code is `fort.4` and has the form of a Fortran namelist. An overview of the options set by this namelist is contained in Table 14. The control file is prepared automatically by the Python code, but some of these parameters appear also as input to the Python part. Note that the selection of the method for obtaining $\dot{\eta}$ follows the logic laid out in Table 15.

All other input files are data in GRIB format that were retrieved from MARS. The code is using dynamic memory allocation and thus does not need to be recompiled for different data sets.

The code is provided with a set of makefiles. The standard version assumes a typical GNU/Linux environment with ~~a~~ the `gfortran` compiler and the required libraries: ~~*openmp* for parallelisation – it comes with~~ `OpenMP` for parallelisation which is

included in the gcc compiler package ~~,~~(libgomp), ~~libgrib_api~~ecCodes ~~or *libeccodes*~~ for handling GRIB files, ~~*libemos*~~ EMOSLIB for transformation between the various representations of fields. Note that the latter two typically require also so-called developer packages containing the Fortran module files. One may substitute ecCodes by its predecessor GRIB_API, if ecCodes is not available. It is assumed that these libraries have been installed as a package from the distribution and thus are at their standard locations and compatible with the gfortran compiler (if not, the makefile library and include paths need to be adapted). There is one makefile called makefile_fast with optimisation that is used for production. In addition, there is makefile_debug which is optimised for debugging. There are also makefiles for ~~cca~~ ~~and~~ the ECMWF servers cca/ccb and ecgate~~as well as a makefile for the Intel Fortran compiler to be used locally. The latter one may require adaptations by users with respect to the library and include paths~~.

If the program finishes successfully, the last line written to ~~standard output~~ ~~is~~ standard output is SUCCESSFULLY FINISHED calc_etadot: CONGRATULATIONS which is useful for automated checking the success of the run. The output file into which the fields of $\dot{\eta}_p$ and the other three-dimensional variables (temperature, specific humidity, u and v components of the wind – not the recently introduced cloud water variable) are combined is fort.15; it is a GRIB file.

The code also foresees options for certain checks where different forms of the vertical velocity are obtained, statistically compared, and also written out (see Table 14). These options were used for quality control in the development process and should not normally be activated by users.

Currently, the code also unifies the ~~three-dimensional~~ three-dimensional fields extracted from MARS and stored in separate GRIB files with the calculated vertical velocity by writing out all fields into a single GRIB file; later this is unified with the 2D fields and the new 3D parameters such as cloud water and written out into a final single GRIB file as required by FLEXTRA and FLEXPART.

3.8 Temporary output files

These temporary output files are usually deleted after a successful data extraction. They are only kept in debugging mode, which is the case if the DEBUG parameter is set to true.

3.8.1 MARS GRIB files

All extracted meteorological fields from MARS are in GRIB format and stored in files ending with .grb. MARS requests are split in an optimised way to reduce idle times and considering the limit of data transfer per request. The output from each request is stored in one GRIB file whose name is defined as <field_type><grid_type><temporal_property><level_type>.<date>.<ppid>.<pid>.grb. The field type can be analysis (AN), forecast (FC), 4d variational analysis (4V), validation forecast (CV), control forecast (CF) and perturbed forecast (PF). The grid type can be spherical harmonics (SH), Gaussian grid (GG), output grid (OG) (typically lat/lon) or orography (_OROLSM) while the temporal property distinguishes between an instantaneous field (__) or an accumulated field (_acc). Level types can be model (ML) or surface level (SL) and the date is specified in the format YYYYMMDDHH. The last two placeholders are the process number of the parent process of submitted script (ppid) and the process number of the submitted script (pid). The process IDs are incorporated ~~to~~

~~avoid mixing of fields if several flex_extract jobs are performed in parallel (which is, however, not recommended)~~so that the GRIB files can be addressed properly in the post-processing.

3.8.2 MARS request file

This file contains a list of the MARS requests from one flex_extract run, with one request per line. This is an optional file users are able to create in addition to full extraction; it can also be created without actually extracting the data which is useful for test purposes. Each request consist of the following parameters whose meaning is explained in ~~Tables 8and 8~~Table 8, and in more detail in the supplemental material or are self-explanatory: request number, accuracy, area, dataset, date, expver, Gaussian, grid, levelist, levtype, marsclass (alias class), number, param, repres, resol, step, stream, target, time and type. The parameters Gaussian (defines whether the field is regular or a reduced Gaussian grid), levtype (distinguishes between model levels and surface level) and repres (defines the grid type – SH, GG, OG) are internal parameters not defined as any available input parameter.

3.8.3 ~~Vertical discretization constants~~

~~The file `VERTICAL.EC` is created by the Fortran program as a temporary storage for internal usage and contains the A and B constants to calculate the model level heights in .~~

3.8.3 Index file

The index file is called `date_time_stepRange.idx`. It contains indices pointing to specific GRIB messages from one or more GRIB files so Python can easily loop over these messages. The messages are selected with a predefined composition of GRIB keywords.

3.8.4 Files with forecast vertical flux data

The flux files, in the format `flux<date>[.N<xxx>][.<xxx>]`, contain the de-accumulated and disaggregated flux fields which are listed in Table 13. The files are created per time step with the date being in the format `YYYYMMDDHH`. The optional block `[.N<xxx>]` marks the ensemble forecast, where `<xxx>` is the ensemble member number. The second optional block `[.<xxx>]` marks a ~~pure forecast~~ long forecast (see Sec. 3.9.2) with `<xxx>` being the forecast step.

Note that, in the case of the new ~~dis-aggregation~~ disaggregation method for precipitation, two new sub-intervals are added in between each original time interval. They are identified by the forecast step parameter `STEP` which is 0 for the original time interval and 1 or 2 for the two new intervals respectively.

3.8.5 `fort.*` files

There are a number of input files for the `calc_etadot` Fortran program named `fort.xx`, where `xx` is the number which defines the meteorological fields stored in these files. They are generated by the Python part of flex_extract by just splitting

the meteorological fields for a unique time step from the `*.grb` files. Table 16 explains the numbers and the corresponding content. Some of the fields are optional and are retrieved only with specific settings, for example the divergence is retrieved only if $\dot{\eta}$ is not available in MARS, and the total cloud water content is an optional field for FLEXPART v10 and newer. The output of `calc_etadot` is file `fort.15`.

3.9 Final output – FLEXPART input files

The final output files are the FLEXPART input files containing the meteorological information. FLEXPART expects one file with all relevant fields per time step. Table 17 and 18 list all of the meteorological fields that flex_extract retrieves and FLEXPART expects. The naming of these files depends on the extracted data. In the following sections we describe the differences and how the filenames are built.

3.9.1 Standard output files

The standard file names have the format `<prefix>YYMMDDHH`, where the `<prefix>` is by default defined as `EN` and can be re-defined in the `CONTROL` file. Each file contains all ~~meteorological~~ fields on all selected ~~model~~ levels on a latitude-longitude grid as needed by FLEXPART~~for the specified time step in the filename~~. There is one file per time step and `YYMMDDHH` indicate the date and hour for which the fields are contained in the file. Analysis and forecast times with their corresponding forecast steps are summarized to the actual times. If not otherwise stated, ~~model level~~ model-level fields are in GRIB2 format and surface ~~levels~~ fields in GRIB1. ~~Forecast times and steps are summed up to the corresponding analysis hour~~When CERA-20C data are retrieved, the date format is changed to `YYYYMMDDHH`.

3.9.2 Output files for ~~pure~~ long forecasts

For ~~the selection of forecasts longer~~ a long forecast, where only forecast fields are retrieved for more than 23 ~~h~~hours, a different naming scheme has to be applied to avoid collisions of time steps for forecasts of more than one day. This case is defined as ~~pure~~ long forecast mode and file names are defined as `<prefix>YYMMDD.HH.<FORECAST_STEP>`. The `<prefix>` is, as in the standard output files, `EN` by default and can be re-defined in the `CONTROL` file. In this case, the date format `YYMMDD` does not include the hour~~directly and the hour~~ . The `HH` represents the starting ~~forecast time and is placed separately by .HH~~time (base time) of the forecast. The `FORECAST_STEP` is a 3-digit number which represents the forecast step in hours.

3.9.3 Output files for ensemble predictions

If flex_extract retrieves ~~ensembles, we obtain multiple fields per~~ ensemble members, multiple fields result for each meteorological variable (the ensemble members) ~~for~~ at a single time step~~which~~ . They are distinguished by the GRIB parameter `NUMBER`. ~~For each ensemble member , all fields~~ All fields of one ensemble member are collected together in ~~one file~~ a single file per time step. The standard filenames are supplemented by the letter `N` for ~~"number"~~ "number" and the ensemble member number in a ~~3-digit~~ 3-digit format such as `<prefix>YYMMDDHH.N<ENSEMBLE_MEMBER>`.

3.9.4 Additional fields with new precipitation ~~dis-aggregation~~disaggregation

The new ~~dis-aggregation~~ disaggregation method for precipitation fields produces two additional fields for each time step and precipitation type. They contain the sub-grid points in the ~~correspoinding~~ corresponding original time intervals as described above in Sect. 3.6.2. The two additional fields are marked with the ~~step~~ `STEP` parameter in the GRIB messages, are set to "1" and "2", respectively. The output filenames do not change in this case.

4 Considerations for application

As in earlier versions of the software package, it is still possible to directly start flex_extract with the Python script `submit.py`. An overview of its current command-line arguments is available through `./submit.py --help`. Please note that when flex_extract is started in local mode, the parameter `INPUTPATH` in the `run_local.sh` script must be set, so that each retrieval uses a unique directory to avoid mixing of data files.

There are two more entry points into flex_extract which can be used for debugging; they are described here for the sake of completeness: the Python scripts `getMARSdata.py` and `prepare_flexpart.py`. In the standard way of running flex_extract, they are both imported as modules (as shown in Fig. 3), but they can also be used as executable programs. The script `getMARSdata.py` controls the extraction of ECMWF data, while `prepare_flexpart.py` controls the post-processing. It may happen that the procedure terminates unexpectedly during the post-processing due to time limits on ECMWF servers. In this case, the `prepare_flexpart.py` script can be used to redo the complete post-processing, bypassing the need to retrieve the data from MARS again.

4.1 Example CONTROL files

The file `CONTROL.documentation` provides a collection of the available parameters grouped in sections together with their default values. Users can start from this file to define their own setup, or use one of the example `CONTROL` files as a template (in `flex_extract_v7.1/Run/Control/`). For each data set (see Sect. 2.2), a basic example `CONTROL` file is provided with some additional variations in, for example, horizontal and temporal resolution, field type, method for vertical velocity or duration of forecasts.The variations are specified at the end of the file name (`CONTROL_<dataset>[.optionalIndications]`) as an optional string.

The usage section in the on-line documentation provides more details on how to set the `CONTROL` file parameters for specific applications. For example, `CONTROL` filenames which end with `.public` are for public users. They have the specific parameter `DATASET` for CERA-20C and ERA-Interim data sets to identify the public version in MARS. For ERA5, this parameter is not needed, and thus public users may use any ERA5 file for extraction. For the atmospheric high resolution data sets, indicated by `OD`, the optional string contains information of the stream (`OPER`, `ENFO`, `ELDA`), the field type of forecasts (`FC`, `CF`, `CV`, `PF`, `4V`), the method for extracting the vertical velocity (`eta` or `gauss`), and other aspects such as long forecasts

(36hours), operational half-day retrievals (basetime or twicedaily), temporal resolution (1hourly or 3hourly) or different horizontal resolutions with global vs. limited-area domains (highres).

4.2 Changes in CONTROL file parameters in comparison to previous versions

With version 7.1, all CONTROL file parameters are initialised with default values. Thus, only those which need to be changed to identify the data set to be retrieved have to be set in the CONTROL file. In earlier versions, each parameter name contained the leading string M_; this was removed for version 7.1, but is still accepted for compatibility. The grid resolution had to be provided in 1/1000 of a degree before, while now it can be provided also as a decimal number. Flex_extract is able to identify the correct setting of the GRID parameter in combination with the domain-specific settings.

It is now also possible to reduce the number of data values for the combination of TYPE, TIME and STEP parameters to the actual temporal resolution. Previous versions expected to have 24 values per parameter, one for each hour of the day, even if only 3-hourly data would be requested as shown in the following example:

```
DTIME 3
TYPE AN AN AN AN ... AN AN AN AN
TIME 00 01 02 03 ... 20 21 22 23
STEP 00 00 00 00 ... 00 00 00 00
```

The more intuitive solution of providing the data for the time steps to be retrieved leads, in this example, to eight data values per parameter for a 3-hourly retrieval:

```
DTIME 3
TYPE AN AN AN AN AN AN AN AN
TIME 00 03 06 09 12 15 18 21
STEP 00 00 00 00 00 00 00 00
```

or four values for a 6-hourly retrieval

```
DTIME 6
TYPE AN AN AN AN
TIME 00 06 12 18
STEP 00 00 00 00
```

The only necessity is a consistent setting of the DTIME parameter which defines the temporal resolution. For backward compatibility, DTIME may be coarser than the number of temporal points provided in TYPE, TIME and STEP, but not finer.

With this version of flex_extract, it is possible to retrieve data sets with analysis fields at every hour (such as ERA5 and CERA-20C); therefore, it was necessary to introduce new parameters related to flux fields defining the forecast type (ACCTYPE), time (ACCTIME) and step (ACCMAXSTEP) specifically for the flux fields (accumulated quantities). For daily ERA5 retrievals, which need up to 12 h forecasts twice a day for the flux fields, these parameters would be:

ACCTYPE FC

ACCTIME 06/18

ACCMAXSTEP 12

Several new parameters were introduced which work as switches. Among the more important ones are REQUEST in order to write the settings in the MARS requests to an output file mars_requests.csv, and CWC to trigger the additional retrieval of cloud liquid and ice water content. DOUBLEELDA can be used to double the number of ensemble members if only 25 members are available from the ELDA stream. These additional members are calculated by subtracting from each existing ensemble member twice the amount of the difference between the ensemble member and the control run. To distinguish between the old and new precipitation disaggregation scheme, the switch parameter RRINT was introduced. Setting it to 1 indicates that the new scheme is used; 0 selects the old scheme

4.3 Scientific considerations

First of all, users should be aware of the different ~~natures~~ nature of operational and reanalysis data sets (see Table 1). Operational data are available since the start of ECMWF's operational forecasts, and are ~~influenceend~~ influenced by frequent changes in ~~IFS model~~ the IFS model, for example with respect to model physics and resolution. Reanalysis data sets were created using a single IFS model version throughout the whole period covered. More precisely, the CERA-20C data set (with 91 vertical levels, 1.25° horizontal and 3 h temporal resolution) has a lower resolution but covers a very long period (from 1901 to 2010) and will thus be suitable for certain climate applications. The ERA-Interim data set (with 60 vertical levels, a medium resolution of 0.75° horizontally and 3 h temporally) was the standard ECMWF reanalysis until recently, but ~~has no~~ without $\dot{\eta}$ having been stored in the MARS archive~~which make~~ , making retrievals computationally demanding as it needs to be reconstructed from the horizontal winds through the continuity equation. The new ERA5 data set has the highest resolution (0.25° horizontally and 1 h temporally, 137 vertical model levels) and includes $\dot{\eta}$. Users are encouraged to use ERA5 data rather than the ERA-Interim data set (production ended in August 2019). In addition to its better resolution, ERA5 covers a longer period than ERA-Interim, provides uncertainty estimates with a 10-member ensemble ~~of data assimilation~~data assimilation, and uses a newer IFS model version (ECMWF, 2019l).

With respect to the relation between temporal and spatial resolution, it is important to consider the use in FLEXPART and their influence on numerical errors. It is not useful to apply high horizontal resolution in combination with, for example, 6-hourly temporal resolution as in such a case, small fast-moving structures are resolved in space, but their movement will not be properly represented. Interpolation will not let the structures move, but rather jump from their position at time t to that at time $t + 6$ h if the displacement between two subsequent times where fields are available is comparable to or larger than their characteristic width ~~normal~~ along to the phase speed. Users can orient themselves ~~with~~ looking at the spatial and temporal resolutions at which ECMWF provides reanalysis data, and the sample CONTROL files.

On the other hand, one has to keep in mind the requirements of the FLEXPART application. For a climatological study on global scales, a horizontal resolution of 0.5° or 1° could be a reasonable choice, whereas tracking point releases in complex terrain would call for the best available resolution.

Attention should also be paid to the model topography and the land-sea mask. Due to limited resolution, a coastal site with a given geographical coordinate could be over water in the model. Then it might be better to shift the coordinates of a release or receptor point in FLEXPART slightly. Another aspect is that the smoothed representation of the topography could mean that the model topography is above or below the real height of a site. It is therefore important to select the proper kind of z coordinate in the FLEXPART RELEASES file. As a compromise, one can place a release location at a ~~hight~~ height between real and model topography (for mountain sites which are usually lower in the model topography than in reality). In such cases, it is strongly recommended to retrieve the model topography and land-sea mask manually and investigate them carefully before deciding on the FLEXPART set-up details, or even before retrieving the full meteorological data set, as one might come to the conclusion that one with better resolution should be used.

The vertical levels used in FLEXPART follow a hybrid η coordinate system. This is much more efficient than pure pressure levels since hybrid η coordinates ~~are terrain following near the~~ follow the terrain near ground and approach pressure levels ~~close to~~ towards the model top. This ~~has the advantage of better resolution of strong gradients in the boundary layer irrespective of the terrain height, and~~ allows to easily ~~fulfill~~ fulfil the lower boundary condition of a flow parallel to the surface whereas pressure levels do not follow the terrain (Stohl et al., 2001), while still at higher levels of the atmosphere, where the flows are close to horizontal, strong vertical motions derived from coordinate transformation are avoided. It also allows better to assign a higher vertical resolution to the lowest part of the atmosphere. ECMWF data sets either directly provide the $\dot{\eta}$ variable ~~or~~ (set `ETA` and `DPDETA` to 1, see `CONTROL` files with `eta` in their names), or include the data needed to reconstruct it (set `GAUSS` to 1, see `CONTROL` files with `gauss` in their names) accurately. This is a big advantage of ECMWF data compared to other data sources, most notably the NCEP model data, which are publicly available only on pressure levels.

Attention should be paid to the number of vertical model levels to be extracted and used in FLEXPART, as the computational cost of the FLEXPART `verttransform` subroutine (reading and preparing meteorological input) increases with the third power of the number of vertical levels. Thus, only data that are really needed for the application (e.g. ~~,~~ troposphere, or troposphere and lower stratosphere) should extracted. File `CONTROL_OD.OPER.FC.eta.highres.reducedlev`, for example, retrieves a limited domain with high horizontal (0.2°) and 1-hourly temporal resolution with $\dot{\eta}$ levels up to, approximately, 100 hPa by setting `LEVELIST` to `60/TO/137`.

Operational data sets and ERA-Interim have analysis fields ~~on a~~ at 6- or ~~12-hourly basis (0, (6~~12-hour intervals (00, (06), 12 and (18) UTC) only. The gaps inbetween can be filled with forecast fields. Mixing analysis and forecast fields should be done by considering at which time steps the differences between two IFS run segments will be the ~~lowest~~smallest. For example, using all four analysis fields ~~, but the forecasts starting from~~ together with forecasts starting at 00 and 12 UTC ~~only~~ would lead to ~~unnecessary rate of~~ undesired changes between 05, 06, and 07 UTC and 17, 18, and 19 UTC. This should be avoided by using only 00 and 12 UTC analysis fields and the forecast fields for +1 to +11 hours for the forecasts starting at times 00 and 12 UTC, respectively ~~.~~

(note that forecasts from the intermediate analyses at 06 and 18 UTC are not archived). See file `CONTROL_OD.OPER.FC.eta.globa` for an example.

5 Quality assurance

~~Nowadays software development is mainly dominated by adding new features as well as maintaining and adjusting specifications rather than developing from scratch (Beizer, 1990).~~ To assure a certain quality of ~~the software, the testing part~~ a piece of software, testing is at least as important as developing the ~~software~~ code itself. Adding new functionalities requires to develop new tests to ~~find bugsor~~ identify possible bugs, or to show that the ~~software~~ code works under specified conditions. As a consequence, ~~tests from the previous software~~ output from the tests conducted with the preceding version can be used to ~~show~~ verify that there are no ~~undesired changes in the unchanged part of the software~~unexpected side effects. This is called regression testing (Beizer, 1990; Spillner, 2012). ~~Also~~As the functionality of the software changes, tests need ~~constantly be updated to follow up the changes~~to be updated or expanded as well. For this flex_extract version, ~~a huge part was about code refactoring which necessitated the development of~~ code refactoring was at the core of the development, and a number of regression tests ~~. First of all~~were developed for that. In addition, a first set of ~~*unit tests*~~ unit tests (Sect. 5.1), which ~~are~~ also serve as a kind of regression test, have been developed within the refactoring process as they are the established ~~best practice~~ best-practice in software engineering to investigate small code blocks. Furthermore, we defined test cases to compare the outcome of two ~~software~~ flex_extract versions after three different stages of the ~~software, first the prepared MARS requests~~ retrieval process: (1) the MARS requests prepared (Sect. 5.2), ~~second the obtained vertical velocity for the different possibilities~~ (2) the vertical velocity obtained with the different options of calculation (Sect. 5.4)~~and third~~ , and (3) the final output files in GRIB format (Sect. 5.3). In addition, generic tests were performed by applying ~~the software~~ flex_extract with predefined `CONTROL` files (Sect. 5.5) which are distributed with the software package to serve as examples for the typical applications. Finally, ~~on top of these tests,~~ some code metrics were determined to track ~~the quality~~ quantitative quality aspects of the code. ~~Combining~~ The combination of all of these tests establishes a sustainable testing environment~~to improve~~ , which will benefit the future development process. ~~They are not important for the normal user~~ The testing environment is not directly relevant for users of flex_extract.

5.1 Unit tests

Unit tests are used to test the smallest pieces of code (single code blocks) independently to identify a potential lack of functional specification (Beizer, 1990). Applying unit tests does not guarantee error-free software~~rather than limiting the chance of occurrence~~, rather it limits the likelihood of errors. Once the tests ~~were writtenthey are also~~ are written, they serve also as a kind of documentation and ~~serve as a protection to not altering a functional behavior after applying~~ to protect against alteration of the functional behaviour by future code changes (Wolff, 2014). ~~Hence~~In this sense, they are also ~~kind of a~~ a kind of regression test.

As a first step, we launched For the current version of flex_extract, we prepared a first set of unit tests for functions which were designed or partly refactored to be testable code blocks. Since unit tests are for the verification of small and independent code blocks, functions which are too complex or too long are badly testable most of the times. In the future, our Our intention is to increase the number of unit tests and refactor the still too in the future, and to further refactor some still rather complex functions into smaller ones (see also Sect. 5.6 or the supplemental material for identifying complex functions).

We used the `pytest` package which is a part of standard Python as well as the `mock` package which simulates external dependencies or results for the tests solely. This gives the opportunity to test the good and bad pathes in a function and usually a function holds as many unit tests as there are different branches. It is a matter of defining all possible results depending on the input states and verify the expected results. The first set of unit tests were applied for functions from the `install` and `tools` modules as well as for the `UIOFiles` and `EcFlexpart` class. The details for each test are not described here; their functionality is obvious from the code.

5.2 Regression testing for MARS requests

The parameters in the MARS requests produced by flex_extract are a key component of the extraction process. Flex_extract v7.1 contains a test to compare the content of MARS requests as produced by two versions. It checks whether the number of columns (parameters) in the request files (see Sect. 3.8.2) is unchanged, whether the number of requests is equal, and whether the content of the request is identical (exceuept except for the desired differences and the environment-dependent data such as paths).

The MARS request files for the current version in use are generated automatically at runtime without actually retrieving the data, while the files for the reference version have to be in place alreadybefore. Since the MARS request files are grouped by version and are kept saved, the number of reference data sets will grow with each new version.

The release comes with a predefined set of `CONTROL` files explicitly for this test as well as with a set of MARS request reference files from the previous version 7.0.4. The test can compare any number of MARS request files emerging from a set of `CONTROL` files. However, one has to make sure that the reference version contains the request files from the same `CONTROL` files. Results are saved in log files. Instructions on how the test can be conducted are given in a `README.md` file.

The comparison between version 7.0.4 and 7.1 only showed expected differences related to a bug fix in the determination of the time period.

5.3 Regression testing for GRIB files

The final product of flex_extract, the FLEXPART input files in GRIB format (see Sect. 3.9), should be identical equal between the previous and the current version, apart from the new or modified featuresfeatures. Since there is always a possibility to have tiny (insignificant) deviations in the actual field values when retrieving at different points in time (changing changes in the environment, library versions, interpolation computational uncertainties, etc.), the focus of this test lies on the files themselves and the GRIB message headers which should not be different. Future improvments improvements may also test for value differences considering a significance threshold.

A regression test was created which compares the GRIB files produced by two versions with respect to the number of files produced, the file names, the number of GRIB messages per file, the content of the GRIB messages header, and statistical parameters for the data themselves. If differences are reported, the developer has to judge whether they are expected or ~~an indication of problems. The current release version 7.1 includes a minimal set of reference data from version 7.0.4, one for each type of data set (see Sect. 2.2). There will be more test data in the future which can then be downloaded from the community website to limit the size of the distributed release tarball. The corresponding reference `CONTROL` files are also distributed with the tarball to enable the retrieval of the data with the new version. This has to be done manually followed by placing the resulting GRIB files in a specific path as described in the *README.md* file.~~ indicate a problem.

5.4 Functionality and performance tests for the Fortran code

Regression tests were set up to reflect the three possibilities for obtaining the vertical velocity $\dot{\eta}$ listed in Table 15. In addition to a basic test for each, enriched tests are implemented where all checks and additional outputs are activated (names with appended `all`). These tests use a pre-specified small domain ($10° \times 10°$, 11 levels) and low spectral resolution (T159) and thus run quite fast. As high spectral resolution and a large domain may pose specific problems, and as it will be relevant to watch the run times, additional high-resolution tests have been created for the `gauss` and `etadot` cases with a domain covering the northern hemisphere and all 137 vertical levels. The `gausshigh` test uses a grid spacing of 0.25° and the corresponding spectral resolution of T799; the `etadot` case uses 0.1° and T1279.

~~The code package contains a set of reference outputs, and scripts to create the reference output and to run the actual regression tests. It checks for bitwise identity of the output files (data files and `standard output` written to a log file). A quantitative comparison of the resulting $\dot{\eta}_p$ which would be useful for modifications that affect the results is not yet implemented. The scripts run each test with both the fast and the debug version of the executable. The script for creating the reference also ensures that both yield identical results. In addition, the runtimes are saved to a `csv` file.~~

5.5 Generic test using predefined `CONTROL` files

Flex_extract comes with a set of `CONTROL` files~~listed in Table **??**; executing it~~ , which can be found in the `flex_extract_v7.1/Run/C` directory; executing flex_extract with each of them constitutes a generic test ~~ensuring~~ which ensures that the data extraction works ~~without problems~~ for all typical applications. ~~This has been verified for version 7.1 by manually executing the software with all these files, and inspecting the results produced. Note that public users can only use files ending with `.public`; they were tested in local public mode. All other cases were tested both in local and in gateway mode. Since the remote mode does not differ much from the gateway mode, only a subset of files were also tested in this mode. Results were evaluated by inspecting the log files for "success" messages and, where possible, with the regression test for GRIB file comparison (Sect. 5.3). Regarding new features, the files were inspected manually for the expected result.~~

~~List of generic `CONTROL` files coming with the flex_extract version 7.1 release. Each file name contains information about some key aspects of the data set to be retrieved. The first name component is an abbreviation of the data set name (`OD`,~~

EA5, CERA, EI) and whether the domain is global (if not, no name component). Reanalysis data sets are divided into *public* and non-public retrievals. File names for operational data sets also contain information about the stream, the forecast type and the method of deriving the vertical velocity. Further information is optional and mostly indicates time resolution or period to be retrieved or whether a specific CONTROL parameter was used. CONTROL_CERA CONTROL_CERA.global CONTROL_CERA.public CONTROL_EA5 CONTROL_EA5.global CONTROL_EI CONTROL_EI.global CONTROL_EI.public CONTROL_OD.ELDA.FC.eta.ens.double CONTROL_OD.ENFO.CF.36hours CONTROL_OD.ENFO.CV.36hours CONTROL_OD.ENF CONTROL_OD.ENFO.PF.ens CONTROL_OD.OPER.4V.eta.global CONTROL_OD.OPER.FC.36hours CONTROL_OD.OPER.FC.eta.b CONTROL_OD.OPER.FC.eta.global CONTROL_OD.OPER.FC.eta.highres CONTROL_OD.OPER.FC.gauss.global CONTROL_OD.OI CONTROL_OD.OPER.FC.operational CONTROL_OD.OPER.FC.twiceaday.1hourly CONTROL_OD.OPER.FC.twiceaday.3hourly CONTROL_OD.temporary

5.6 Code Metrics

Metrics for the maintainability and complexity of code as well for the documentation are a useful tool for developers who should aim at maintaing maintaining or reaching good scores in these metrics. For the Python code of flex_extract, a number of metrics were calculated for the previous version 7.0.4 and the current version 7.1. This section summarizes the metrics and their main findings. More details can be found in the supplemental material.

Basic metrics, taken from Lacchia (2019) and calculated through with the Python package `radon` (Lacchia, 2019), are

- the total number of lines of code (`LOC`),
- the number of logical lines of code (`LLOC`),
- the number of source lines of code (`SLOC`),
- the number of (single) comment lines (`comments`),
- the number of lines in multi-line comment strings (`multi`), and
- the number of blank lines (`blank`),

with the following relation between these numbers:

$$\texttt{LOC} = \texttt{SLOC} + \texttt{multi} + \texttt{comment} + \texttt{blank}. \tag{7}$$

The comparison shown in Table 19 indicates a significant increase not only in the logical lines of code, but even more in `comment` and `multi`, mostly representing an improvement of in-line documentation by splitting large code blocks into smaller ones, each with a new `docstring`. A so-called `docstring` is a specific multi-line comment for the documentation of functions, methods and classes, describing their input and return values, which can be read by tools for automatic generation of a separate documentation. The re-factorization re-factorisation of code blocks, additional code for new features, and

compliance with certain code style rules (e.g., maximum length of lines), about 1000 lines of pure code were added. The ratio of comment lines (`multi + comment`) to source-code lines (`SLOC`) grew from 20 % to 117 %.

A further metric for code quality is the so-called *cyclomatic complexity* (CC), also called the McCabe metric ~~since it was developed by Thomas J. McCabe Sneed et al. (2010).~~ (Sneed et al., 2010). It is equal to the number of linearly independent paths through the control flow graph of the code or the number of decisions plus one. A lower CC score indicates ~~lower the~~ a lower complexity which is deemed an advantage. ~~CC is calculated as~~

$$CC = E - N + 2P,$$

~~where E is the number of edges (or also called links) of the graph, N is the number of nodes of the graph and P is the number of connected components (which are sub-graphs from functions independent of the supergraph) (Lacchia, 2019; Beizer, 1990; Sneed et al., 20~~ . ~~The nodes represent the conditional branch instructions and program junctions, and edges are the segments between such points.~~ Table 20 gives an overview of the rank definitions. Regarding code testing, CC provides a lower-bound number of how many test cases (unit tests) are necessary to provide complete path coverage (Beizer, 1990). ~~This metric was also calculated with the `radon` package (Lacchia, 2019). It provides the CC rank for each function, class and class method. Table 20 gives an overview of the interpretation of these ranks. In generalits~~

In general, it is said that the score should ~~not be above~~ be less or equal 10, corresponding to rank ~~C to F~~A and B. From the statistical point of view, ~~Table ?? shows that~~ only 10.3 % of flex_extract version 7.1 code blocks have higher complexity, while in version 7.0.4 this was the case for 30.8 %.

~~Number of code blocks (classes, methods, functions) with a specific rank of cyclomatic complexity and the percentage of the total blocks for version 7.1 (116 in total) and 7.0.4 (45 in total). Determined with the Python package `radon` (Lacchia, 2019). Rank A 21 44.6 %7665.52 %B 10 22.2 %2824.14 %C 3 6.6 %97.76 %D 4 8.8 %10.86 %E 3 6.6 %10.86 %F 4 8.8 %10.86 %~~

The mean cyclomatic complexity of all code blocks in the new Python code is 5.74 (B); for those blocks with C to F ~~(Table ??)~~, it is 21 (C). In version 7.0.4, the corresponding numbers are 13 (C) and 31.86 (E), indicating a substantial improvement. ~~Table ?? lists all the code blocks with their ranks and scores.~~ For example, the class `ControlFile` was improved significantly, as well as the class renamed from `EIFlexpart` to `EcFlexpart`. On the other hand, the class method `deacc_fluxes` became more complex in version 7.1. This is mainly due to two new features, ~~of~~ ensemble retrieval and the new disaggregation. Nevertheless, the overall code complexity was reduced.

~~Python code blocks with CC ranks C-F, with rank class and CC score. The block types are classes (C), class methods (M) and functions (F) Block Block Rank CC type score GribTools.setkeys M C 11 MARSretrieval.dataRetrieve M C 15 Control C D 23 EIFlexpart.process_output M D 26 EIFlexpart C E 31 EIFlexpart.deacc_fluxes M E 34 EIFlexpart.retrieve M F 43 EIFlexpart.__init__ M F 49 Control.__init__ M F 56 EIFlexpart.create M F 57 install_args_and_control F C 12 getMARSdata F D 25 install_via_gateway F D 30 interpret_args_and_controlF E 34 Block Block Rank CC type score EcFlexpart C C 13 EcFlexpart._create_params M C 13 EcFlexpart._prep_new_rrint M C 14 EcFlexpart._create_field_typesM C 15 MarsRetrieval.data_retrieve M C 16 ControlFile._read_controlfile M C 17 EcFlexpart.retrieve M D 25 EcFlexpart.create M E 36 EcFlexpart.deacc_fluxes M F 57 install.py::check_install_conditions F C 11 install.py::mk_tarball F C 17 disaggregation.py::IA3 F C 18~~

Another software metric is the maintainability index (MI), ranking between where values 0 – 9 indicate low maintainability, 10 – 19 medium, and 20 – 100 ; it is a function of SLOC, CC and the Halstead volume (V) (Lacchia, 2019):

$$\mathrm{MI} = \max\left[0, \frac{100}{171}\left(171 - 5.2\ln \mathrm{V} - 0.23\,\mathrm{CC} - 16.2\ln \mathrm{SLOC} + 50\sin\sqrt{2.4\mathrm{C}}\right)\right]$$

where C is the fraction of comment lines (converted to radians) (Lacchia, 2019). The Halstead volume is defined as

$$\mathrm{V} = (N_1 + N_2)\log_2(\eta_1 + \eta_2)$$

with η_1 being the number of distinct operators, η_2 being the number of distinct operands, N_1 the total number of operators and N_2 the total number of operands. Table ?? defines the classification ranks. high maintainability. The index is calculated for a complete Python file and Table ?? shows the ranks of . Both Python versions have in general highly maintainable Python files except `FlexpartTools.py` in version 7.0.4 with an MI score of 0.0 and `EcFlexpart.py` in version 7.1 respectively. with an MI score of 10.79.

Definition of maintainability ranks. This classification was taken from the documentation of the Python package `radon` (Lacchia, 2019). MI score Rank Maintainability 20 – 100 A Very high 10 – 19 B Medium 0 – 9 C Extremely low

Maintainability index in increasing order for the Python files of both versions. This was determined with the Python package `radon` (Lacchia, 2019). File Rank MI score Classes/EcFlexpart.py B 10.79 Classes/MarsRetrieval.py A 26.92 Mods/checks.py A 26.15 Classes/MarsRetrieval. py A 26.92 Mods/disaggregation.py A 28.55 Mods/profiling.py A 38.10 Mods/tools.py A 38.32 Mods/get_mars_data.py A 44.77 install.py A 47.33 Mods/prepare_flexpart.py A 47.47 Classes/GribUtil.py A 57.07 submit.py A 58.90 _config.py A 77.35 Classes/UioFiles.py A 100.00 File Rank MI score FlexpartTools.py C 0.00 opposite.py A 45.25 install.py A 48.96 getMARSdata.py A 56.28 GribTools.py A 59.10 submit.py A 67.72 prepareFLEXPART.py A 71.40 UIOTools.py A 85.18

Additionally, we used a source code quality checker program called `pylint` (Thénault, 2001) to support following which indicates how well the Python style guide PEP8 (van Rossum et al., 2001) is followed. This tool provides an overall rating index with a maximum number value of 10. Applying According to this tool, flex_extract version 7.0.4 has a rating of -8.77 and version 7.1 a rating of 9.09. This shows a massive improvement in following the official style guidesguide.

6 Final remarks and outlook

6.1 Conclusions

This paper describes the software package flex_extract v7.1, which retrieves meteorological fields from the ECMWF IFS model and prepares them for the use in the Lag rangian Lagrangian particle dispersion model FLEXPART. The software package was initially developed in the 1990ies 1990s and underwent various developments to adapt to changes in the ECMWF environment

and the data set characteristics. In the past two years, ~~the~~ ECMWF introduced considerable changes in its software environment ~~to retrieve, read and access data as well as the preparation of~~ for retrieval, reading and accessing data and also released new data sets. This necessitated a substantial upgrade of flex_extract to adapt to these changes. ~~Additionally~~Moreover, the user community had new requirements for data retrievals which were considered in this version. In the development process, substantial refactoring was carried out, the number of retrievable data sets was increased, user-friendliness was improved, current ECMWF software packages considered, an ~~online~~ on-line documentation was built, and a first set of test cases for future regression testing was created. Furthermore, a recently developed and improved disaggregation method for precipitation fields was implemented as an option.

The number of groups using FLEXPART grew substantially over the past decade and with the new opportunity of publicly available reanalysis data sets there will likely be even more users ~~who will try out and apply~~ interested in trying out and applying FLEXPART for their research. Alongside with this reference paper, the newly established git repository on the FLEXPART community ~~website~~ web site https://flexpart.eu and the ~~online~~ on-line documentation should assist all these users with up-to-date information about changes, releases of new versions, installation and usage including a documentation useful for future developers.

6.2 Support

FLEXPART has a community ~~website~~ web site http://flexpart.eu, where flex_extract as a pre-processor has its own sub page[8]. The ~~website~~ web site features a ticket system to report issues or submit feature requests. The tickets can be viewed by anyone~~,~~ ; to create a ticket, registration[9] is necessary. There is also a mailing list for discussion among FLEXPART / FLEXTRA users and with developers, where questions may be asked or experiences be shared, including pre- and post-processing issues. Announcements for all FLEXPART users, such as new releases, are distributed through the list as well. Future contributions to the code are welcome; for granting permission of write access to the git repository, communication via email or ticket is necessary.

6.3 Future work

In its current status, the on-line documentation provides a basic reference. In the future, more examples should be provided, including answers to typical user questions and workarounds for known problems. Information about updates and new releases will also be implemented in this documentation.

It is also intended to ~~optionally retrieve meteorological fields from ECMWF~~ provide for the optional retrieval of meteorological fields needed as input (initial and boundary conditions) for the WRF model to support the FLEXPART-WRF community.

The unification of the various three-dimensional fields into a single file shall be removed from the Fortran code as this is a simple task that can be fulfilled more efficiently and transparently with ~~ECcodes~~ecCodes command-line tools.

[8]https://www.flexpart.eu/wiki/FpInputMetEcmwf; Last accessed: 17.08.2019
[9]https://www.flexpart.eu/wiki/RegisteredUser; Last accessed: 17.08.2019

The ERA5 reanalysis has ensemble members stored in the *enda* stream, but the flux data have a different accumulation period and therefore are not yet retrievable. It is planned to allow the retrieval of these ensemble members in the future. Up to now, it is possible to set flex_extract to retrieve ~~reduced Gaussian fields~~fields on the reduced Gaussian grid. This should be extended to include the octahedral reduced Gaussian grid.

The hybrid vertical velocity $\dot{\eta}$ is now stored not only for the operational forecasts but also for the new reanalyses, thus the need to calculate it is diminishing. ~~Therefore, the option to calculate it on the native Gaussian grid might be removed in the future. This would allow to do all the remaining calculations in Python3 without resorting to Fortran code.~~ In future versions of flex_extract, `calc_etadot` will probably only be called if $\dot{\eta}$ really needs to be calculated, not just for multiplying it with $\partial p/\partial \eta$ as this can be done with sufficient efficiency in Python.

The flex_extract software package is currently provided as a compressed tar file. In the future, a package shall be made available to be installed as a system ~~software for all users. Then~~package for certain GNU/Linux distributions. In this case, only user-specific data will need to reside in ~~a~~ the user directories.

Code and data availability. Flex_extract is a code package consisting of Python scripts, shell scripts and a Fortran program; it is open software distributed under a Creative Commons (CC-BY-4.0) licence. The latest version of the code (currently 7.1.2) is available through the flex_extract project webpage (https://www.flexpart.eu/wiki/FpInputMetEcmwf; Last accessed: 09.07.2020) which is part of the FLEXPART community web site and contains links to the tarball and the git repository; the on-line documentation is also hosted there (https://www.flexpart.eu/flex_extract/; Last accessed: 09.07.2020).

The exact version at the time of manuscript submission is archived on Phaidra (https://phaidra.univie.ac.at/view/o:1070149, DOI:10.25365/phaidra.130), the permanent secure storage of the University of Vienna.

The software package comes with a number of test cases which should be used by developers in the future. Some tests need additional reference data which have to be downloaded separately from the project web site. The following open-source libraries have to be available in addition to the libraries mentioned in the installation section in order to run the flex_extract test cases: `numpy/scipy` (Walt et al., 2011), `pandas` (McKinney, 2010), `xarray` (Hoyer and Hamman, 2017), `pytest` (Krekel, 2019), `mock` (Foord and the mock team, 2019). For the generation of the on-line documentation, the Python package `sphinx` (Brandl, 2019) is required, and for the documentation of the Fortran code, FORD[10].

The current version 7.1 of flex_extract was developed under GNU/Linux and was tested only on this platform. Application under other operating systems may be possible, but without supported by the developers.

Appendix A: Installation instructions

First of all, download the release version from the FLEXPART community ~~website~~web site. Alternatively, if git is installed on the target machine, you may clone the latest version ~~from our git repository master branch.~~ of the master branch from the git repository on the community web site.

[10]http://fortranwiki.org/fortran/show/FORD, accessed 20 Dec 2019

```
git clone --single-branch --branch master
~~~~~~~~~ https://www.flexpart.eu/gitmob/flexpart
          https://www.flexpart.eu/gitmob/flex_extract
```

Currently, ~~the software~~ flex_extract was only tested for a GNU/Linux environment. The installation process depends on the user group (see Sect. 2.1) and the application mode (see Sect. 3.1). One should first decide for the modes and then follow the compact instructions in the corresponding subsections. Shell scripts and Python code snippets mentioned in the Appendix can be found in the directory `Testing/Installation/` after unpacking the tarball. For more details see the instructions in the ~~online~~ on-line documentation.

A1 Registration and ~~licenses~~licences

Table 6 summarizes which registration is required. Follow the given links from the literature to the registration ~~websites~~ web sites (or footnotes).

A separate ~~license~~ licence has to be accepted for each ECMWF public data set, regardless of the user group. For the ERA-Interim and CERA-20C datasets this can be done at the ~~website~~ web site for "Available ECMWF Public Datasets"[11]. Log in and follow the ~~license~~ licence links on the right side for each data set and accept it. For the ERA5 data set this has to be done at the "Climate Data Store (CDS) ~~website~~web site"[12]. Log in and select, on the left panel, product type "Reanalysis" for finding ERA5 data sets. Then follow any link with ERA5 to the full data set record, click on tab "Download data" and scroll down. There is a section "Terms of use" where the "Accept terms" button has to be clicked. The ~~licenses for member state~~ licences for member-state users are accepted by the user when receiving a so-called "Token", which generates new passwords for each log in.

A2 ~~Preparing the system~~System prerequisites

Remote mode

ECMWF servers provide all required libraries (see Table ~~??) via a module system~~7) via an environment modules framework. Flex_extract takes care of loading the right modules at runtime~~except the Python3 module which needs to be loaded prior to its execution by `module load python3`. This is due to the fact that flex_extract is first started to prepare the job script with the correct settings before submitting the job to the batch queue~~.

Gateway mode

In this mode, access to the ECMWF computing and archiving facilities is enabled through an ~~*ECaccess*~~ `ECaccess` gateway server on a local member state server. The ~~*ECaccess*~~ `ECaccess` framework is necessary to interactively submit jobs to the ECMWF batch system and to transfer files between ECMWF and local gateway server. As a consequence, a member state

[11] https://confluence.ecmwf.int/display/WEBAPI/Available+ECMWF+Public+Datasets; Last accessed: 11.11.2019
[12] https://cds.climate.copernicus.eu/cdsapp#!/search?type=dataset; Last accessed: 11.11.2019

gateway server has to be established[13] and a so-called association[14] has to be created to use the ~~*ECaccess*~~ ECaccess file transfer service ectrans.

The next step is to create an ~~*ECaccess* certificate to authorize~~ ECaccess certificate to authorise the file transfers and job submissions. This certificate has to be renewed periodically (every 7 days). The certificate is created by executing the command ecaccess-certificate-create on the ~~command line~~ command-line of the local gateway server and the user is prompted for the ECMWF ~~member state~~ member-state user name and a password (generated by a token).

```
$ ecaccess-certificate-create
Please enter your user-id: example_username
Your passcode: ***
```

Additional dependencies on the local gateway server are Python3 and the Python packages NumPy and genshi. Use the package management system of your Linux distribution which ~~required~~ requires admin rights. The installation was tested under GNU/Linux Debian buster and Ubuntu 18.04 Bionic Beaver. The following installation instructions refer to a Debian-based system and use apt-get as package manager; of course, other package managers (e. g. aptitude), or other GNU/Linux distributions can be used as well.

```
apt-get install python3
apt-get install python3-genshi
apt-get install python3-numpy
```

Local mode

For the local mode, all software dependencies listed in ~~Section~~ Sect. 3.2 have to be provided. The installation process is the same for the ~~*member* and *public*~~ member-state and public access modes. Use the package management system of your Linux distribution (requires admin rights) to establish the dependencies if not already available. Note that for the Python version of ecCodes, a version 2.13.0 or higher is necessary.

```
apt-get install python3
apt-get install python3-eccodes
apt-get install python3-genshi
apt-get install python3-numpy
apt-get install gfortran
apt-get install fftw3-dev
apt-get install libeccodes-dev
apt-get install libemos-dev
```

As currently the CDS and ECMWF API packages are not available as Debian packages, they need to be installed outside the Debian (Ubuntu etc.) package management system. The CDS API (cdsapi) is required for ERA5 data and the ECMWF Web API (ecmwf-api-client) for all other public datasets. ~~Since public users currently do not have access to the full ERA5 data set, they can skip the installation of the CDS API.~~ The recommended way is to use the Python package management system pip:

[13] https://confluence.ecmwf.int/display/ECAC/ECaccess+Home; Last accessed: 31.10.2019

[14] https://confluence.ecmwf.int/download/attachments/45759146/ECaccess.pdf see page 17 ff. for instructions; Last accessed: 28.10.2019

```
apt-get install pip
pip install cdsapi
pip install ecmwf-api-client
```

Note that if you would like to use Anaconda Python we recommend you follow the installation instructions of Anaconda Python Installation for Linux and then install the ~~eccodes~~ ecCodes package from conda with:

```
conda install conda-forge::python-eccodes
```

Both user groups have to provide keys with their credentials for the Web APIs in their home directory. Therefore, follow these instructions:

ECMWF Web API Go to MARS access ~~website~~web site[15] and log in with your credentials. Afterwards, on this site in section "Install ECMWF KEY" the key for the ECMWF Web API should be listed. Please follow the instructions in this section under 1 (save the key in a file `.ecmwfapirc` in your home directory).

CDS API Go to CDS API registration[16] and register there too. Log in at the cdsapi ~~website~~ web site and follow the instructions at section "Install the CDS API key" to save your credentials in a `.cdsapirc` file in your home directory.

Since a single retrieval run of flex_extract can take a while, it is recommended to do some basic tests for the local access method to identify problems with the Web APIs early on. A very simple test retrieval for both Web APIs are enough to be sure that everything works.

For the ECMWF Web API and as a ~~*member*~~ member-state user please use this piece of Python code:

```
from ecmwfapi import ECMWFService

server = ECMWFService('mars')

server.retrieve({
'stream'    : "oper",
'levtype'   : "sfc",
'param'     : "165.128/166.128/167.128",
'dataset'   : "interim",
'step'      : "0",
'grid'      : "0.75/0.75",
'time'      : "00/06/12/18",
'date'      : "2014-07-01/to/2014-07-31",
'type'      : "an",
'class'     : "ei",
'target'    : "download_erainterim_ecmwfapi.grib"
})
```

For the ECMWF Web API and as a ~~*public*~~ public user please use that piece of Python code:

[15] https://confluence.ecmwf.int//display/WEBAPI/Access+MARS; Last accessed: 20.10.2019
[16] https://cds.climate.copernicus.eu/api-how-to; Last accessed: 25.10.2019

```
from ecmwfapi import ECMWFDataServer

server = ECMWFDataServer()

server.retrieve({
'stream'    : "enda",
'levtype'   : "sfc",
'param'     : "165.128/166.128/167.128",
'dataset'   : "cera20c",
'step'      : "0",
'grid'      : "1./1.",
'time'      : "00/06/12/18",
'date'      : "2000-07-01/to/2000-07-31",
'type'      : "an",
'class'     : "ep",
'target'    : "download_cera20c_ecmwfapi.grib"
})
```

Extraction of ERA5 data via CDS API ~~(currently only for *member* users)~~ might take time as currently, at the time of publication, there is a high demand for ERA5 data. Therefore, as a simple test for the API, just retrieve pressure-level data (even if that is NOT what we need for FLEXPART), as they are stored on disk and don't need to be retrieved from MARS (which is the time-consuming action):

Please use this piece of Python code (both user groups) to retrieve a small sample of ERA5 pressure levels:

```
import cdsapi

c = cdsapi.Client()

c.retrieve("reanalysis-era5-pressure-levels",
{
"variable": "temperature",
"pressure_level": "1000",
"product_type": "reanalysis",
"year": "2008",
"month": "01",
"day": "01",
"time": "12:00",
"format": "grib"
},
"download_cdsapi.grib")
```

An example for retrieving ERA5 data from MARS is shown below and can be tested if the code from above worked.

```
import cdsapi

c = cdsapi.Client()

c.retrieve('reanalysis-era5-complete',
{
'class'   : 'ea',
```

```
'expver'   : '1',
'stream'   : 'oper',
'type'     : 'fc',
'step'     : '3/to/12/by/3',
'param'    : '130.128',
'levtype'  : 'ml',
'levelist': '135/to/137',
'date'     : '2013-01-01',
'time'     : '06/18',
'area'     : '50/-5/40/5',
'grid'     : '1.0/1.0',
'format'   : 'grib',
}, 'download_era5_cdsapi.grib')
```

A3 Building flex_extract

Remote mode

First, log in on one of the ECMWF servers, such as ~~*ecgate* or *cca/ccb*~~ecgate or cca/ccb. Second, copy the tarfile to the server, untar the flex_extract release tarball and change into the flex_extract root directory.

```
~~scp <localuser>@<localmachine.tld>:</path/to/tarfile/>~~
~~$HOME/~~
scp <localuser>@<localmachine.tld>:</path/to/tarfile> $HOME
cd $HOME
tar xvf flex_extract_vX.X.tar.gz
cd flex_extract_vX.X
```

Substitute the <localuser> and <localmachine.tld> placeholders with your local user name and the IP name or address of your local machine. ~~Untar the flex_extract release file and change into the flex_extract root directory.~~

Eventually, adapt the parameters (described in Table 11 and 9) in the setup.sh script and execute it.

Flex_extract uses the email address connected to the user account to notify the user about successful or failed installation.

Gateway mode

The actual execution of flex_extract with retrieval and preparation of the data will be run on ~~an~~ ECMWF servers. The only difference is the preparation of the job script, which is done on the local gateway server and sent to ECMWF servers by the ~~*ECaccess* services.~~

ECaccess services. Unpack the release tarball and change into its directory. Substitute X.X with the actual release version number.

```
tar xvf flex_extract_vX.X.tar.gz
cd flex_extract_vX.X
```

Afterwards, prepare the setup.sh script by configuring its parameters (described in Table 11 and 9) and execute it. The makefile has to be selected according to the selection of the target, e.g. ~~*ecgate* or *cca/ccb*~~ ecgate or cca/ccb servers. In this

mode the DESTINATION and GATEWAY parameters have to be set to be able to use the ectrans service. A confguration configuration job script is then sent to the ECMWF batch queue and flex_extract uses the email address connected to the user account to notify the user about successful or failed installation.

Local mode

Since flex_extract compiles the Fortran program preconvertcalc_etadot during the installation process, a corresponding makefile has to be provided. Flex_extract comes with two makefiles for the local mode prepared a prepared makefile for the gfortran (https://gcc.gnu.org/fortran/) and the ifort (https://software.intel.com/en-us/fortran-compilers)compiler. The gfortran version assumes that eccodes and emoslib makefile assumes that ecCodes and EMOSLIB are installed as distribution packages. It is necessary to adapt the two parameters ECCODES_INCLUDE_DIR and ECCODES_LIB in these makefiles if other than standard paths are used.

HenceTherefore, if needed, prepare the Fortran makefile for your environment by starting from one of the two provided makefiles the makefile .local.gfortranmakefile_fast provided and edit the paths to point to the ecCodes or makefile.local.ifc They library on your local machine. It can be found at flex_extract_vX.X/Source/Fortran, where vX.X vX.X should be substituted with the current version number. Edit the paths to the eccodes library on your local machine.

Eventually, adapt the command line command-line parameters (described in Table 11 and 9 9 and 11) in the setup.sh script in the root directory of flex_extract and execute it.

A4 Installation test

The most common errors in applying flex_extract arise from wrong installation and settings regarding the libraries for the Fortran program. Therefore it is useful to do a simple test with a prepared minimal data set. The following instructions have to be executed on the local system for the local mode and on the ECMWF servers in the remote and gateway mode.

From the flex_extract root directory change into the Testing/Installation/ConvertCalc_etadot/ directory and execute the Fortran program by

```
cd Testing/Installation/Convert
cd Testing/Installation/Calc_etadot
# execute the Fortran progam
../../../Source/Fortran/calc_etadot_fast.out
../../../Source/Fortran/calc_etadot
```

The installation was successful if you obtain on standard output:

```
readspectral:            1  records read
readlatlon:              8  records read
STATISTICS:          98842.4598   98709.7359    5120.5385
readlatlon:              4  records read
readlatlon:              4  records read
readlatlon:              4  records read
SUCCESSFULLY FINISHED CONVERT_PRE: CONGRATULATIONS
STOP SUCCESSFULLY FINISHED calc_etadot: CONGRATULATIONS
```

Note that on ECMWF servers the flex_extract root directory is placed in the `$HOME` directory.

Appendix B: ~~How to use flex_extract~~Usage instructions

Flex_extract is a command-line tool which can be started by executing the `submit.py` script in the Python source directory or more preferably with an upstream shell script `run.sh` which calls the `submit.py` script with its corresponding command-line arguments. Therefore, the user should navigate to the `Run` directory, where the shell script is located.

~~cd <path-to-flex_extract_vX.X>/Run~~
cd <path-to-flex_extract_vX.X>/Run

with `X.X` as the placeholder for the version number.

This directory contains all information necessary to run flex_extract. The only files which might need modifications by the user are the `run.sh` script and the selected `CONTROL` file within the `Control` directory. This directory contains a sample set of the current range of possible data set retrievals.

This section describes the basic steps to start a flex_extract retrieval within the different modes based on an example. More details about the usage can be found in ~~Section~~ Sect. 4 and in the ~~online~~ on-line documentation, especially specifics of different data sets and `CONTROL` file parameters.

For the first data retrieval it is recommended to use one of the example `CONTROL` files stored in the `Control` directory to avoid unnecessary problems. We recommend to extract CERA-20C data since they are usually not highly demanded and guarantee quick processing for the best testing experience. In the following, we will provide step-by-step instructions for all application modes to retrieve a single day (08 September 2000) from the CERA-20C dataset with 3-hourly temporal resolution and a small domain over Europe with 1° resolution, using `CONTROL_CERA[.public]`.

Remote and gateway modes

For ~~member state~~ member-state users it is recommended to use the remote or gateway mode, especially for more demanding tasks, to retrieve and ~~convert~~ post-process data on ECMWF machines and to transfer only the final output files to the local host.

The only difference between both modes is the location where flex_extract will be started from. In the remote mode we work directly on the ECMWF server, therefore login to the ECMWF server of your choice and change to the `Run` directory as shown above. Remember, at ECMWF servers flex_extract is always installed in the `$HOME` directory. ~~To be able to start the program, please load the Python3 environment with the module system first.~~

~~module unload python~~
~~module load python3~~

Within the gateway mode, only a change into the ~~Run~~ `Run` directory of flex_extract on the gateway server is necessary.

Otherwise, the rest of the working steps are the same in both modes. Now, open the `run.sh` script and modify the parameter block marked in the file as shown below. The parameters are described in Table 12.

```
# ----------------------------------------
# AVAILABLE COMMANDLINE ARGUMENTS TO SET
#
# THE USER HAS TO SPECIFY THESE PARAMETERS:

QUEUE='ecgate'
START_DATE=None
END_DATE=None
DATE_CHUNK=None
JOB_CHUNK=3
BASETIME=None
STEP=None
LEVELIST=None
AREA=None
INPUTDIR=None
OUTPUTDIR=None
PP_ID=None
~~JOB_TEMPLATE='job.temp'~~
JOB_TEMPLATE='submitscript.template'
CONTROLFILE='CONTROL_CERA'
DEBUG=0
REQUEST=2
PUBLIC=0
```

This would retrieve ~~a one day (08.09.2000)~~ CERA-20C ~~dataset with 3 hourly temporal resolution and a small 1° domain over Europe~~ data on the ECMWF server ~~*ecgate*~~ecgate. For the ECMWF ~~*cca/ccb*~~ cca/ccb servers, the parameter QUEUE has to be adapted. Since the ectrans parameter in the CONTROL_CERA file is set to 1 the resulting output files will be transferred to the local gateway into the path stored in the destination, provided that the destination was correctly ~~setup~~set-up.

Please note that success of the submission of the ectrans command does not guarantee that the file transfer will succeed. It means only that the output file has been successfully submitted to the ectrans queueing system. One still has to check manually in the local directories or with ECaccess tools whether the files reached their final destination. The parameters listed in the run.sh script would overwrite existing settings from the CONTROL file.

Starting the retrieval process will be done by executing the script by ./run.sh.

Flex_extract will print some information about the job on standard output. If there is no error in the submission to the ECMWF server a message like this will be shown:

```
---- On-demand mode! ----
The job id is: 10627807
You should get an email per job with
    subject flex.hostname.pid
FLEX_EXTRACT JOB SCRIPT IS SUBMITTED!
```

Once submitted, the job status can be checked by using the command ecaccess-job-list. At the end of the job, the user should receive an email with a detailed protocol of what was done and if the job was successful.

In case the job failed, the subject will contain the keyword *ERROR!* and the job name. Then, the user can check the email or on ECMWF servers in the $SCRATCH directory for debugging information.

In the $SCRATCH directory on ~~*ecgate*~~ ecgate it is recommended to list the content with ls -rthl to list the most recent logs and temporary retrieval directories (usually ~~pythonXXXXX~~extractXXXXX, where XXXXX is the process id). Under ~~pythonXXXXX~~extractXXXXX a copy of the CONTROL file is stored under the name CONTROL, the protocol is stored in the file prot and the temporary files as well as the resulting files are stored in a directory work. The original name of the CONTROL file can be found within this new file under parameter controlfile.

If the job was submitted to the High Performance Computer (HPC) (QUEUE is cca or ccb) you may login to the HPC and look into the directory /scratch/ms/ECGID/ECUID/.ecaccess_do_not_remove for job logs. The working directories are deleted after job failure and thus normally cannot be accessed.

If the resulting files can not be found in the destination path of the local gateway server, it can be checked if the files are still to be transferred to the local gateway server by using the command ecaccess-ectrans-list.

After this test retrieval was successful, feel free to try changing the CONTROL file parameters described in ~~Tables 8 and 8 and~~ Table 8 and by selecting other CONTROL files. Please mind the considerations of application in ~~Section~~ Sect. 4.

Local mode

Since this mode can be used by ~~member~~ member-state and public users, we show an example for both user groups. Open the run_local.sh file and adapt the parameter block marked in the file as shown for the corresponding user group. The parameters are described in Table 12.

Take this setting as a member-state user:

```
# ----------------------------------------
# AVAILABLE COMMANDLINE ARGUMENTS TO SET
#
# THE USER HAS TO SPECIFY THESE PARAMETERs:
#

QUEUE=''
START_DATE=None
END_DATE=None
DATE_CHUNK=None
JOB_CHUNK=None
BASETIME=None
STEP=None
LEVELIST=None
AREA=None
INPUTDIR='./Workspace/CERA'
OUTPUTDIR=None
PP_ID=None
JOB_TEMPLATE=''
CONTROLFILE='CONTROL_CERA'
DEBUG=0
REQUEST=0
PUBLIC=0
```

and take this setting as a public user:

```
# ----------------------------------------
# AVAILABLE COMMANDLINE ARGUMENTS TO SET
#
# THE USER HAS TO SPECIFY THESE PARAMETERs:
#

QUEUE=''
START_DATE=None
END_DATE=None
DATE_CHUNK=None
JOB_CHUNK=None
BASETIME=None
STEP=None
LEVELIST=None
AREA=None
INPUTDIR='./Workspace/CERApublic'
OUTPUTDIR=None
PP_ID=None
JOB_TEMPLATE=''
CONTROLFILE='CONTROL_CERA.public'
DEBUG=0
REQUEST=0
PUBLIC=1
```

~~This would retrieve a one day (08.09.2000) CERA-20C dataset with 3 hourly temporal resolution and a small 1° domain over Europe. The destination~~ The working location for this retrieval is set by the `INPUTDIR` parameter and will be the `Workspace/CERA*` directory within the current `Run` directory. It is also the output directory since `OUTPUTDIR` was not set. This can be changed to whatever path is preferred. The parameters listed in `run_local.sh` would overwrite existing settings in the `CONTROL` file.

Starting the retrieval process will be done by executing the script by `./run\_local.sh`./run_local.sh.

While a job submission on the local host is convenient and easy to monitor (on standard output), there are a few caveats with this option.

There is a maximum size of 20 GB for single retrievals via ECMWF Web API. Normally this is not a problem but for global fields with T1279 resolution and hourly time steps the limit may already apply. If the retrieved MARS files are large but the resulting files are ~~relativly~~ relatively small (small local domain, but large time period) then the retrieval to the local host may be inefficient since all data must be transferred via the Internet. This scenario applies most notably if `ETADOT` has to be calculated via the continuity equation as this requires global fields even if the domain is local and small. In this case, job submission via ~~*ecgate*~~ `ecgate` might be a better choice. It really depends on the patterns used and also on the speed of the internet.

After this test retrieval was successful, feel free to try changing the `CONTROL` file parameters described in ~~Tables 8 and 8 and~~ Table 8 and by selecting other `CONTROL` files. Please mind the considerations of application in ~~Section~~ Sect. 4.

Author contributions. A. Philipp revised the complete software package (except for the Fortran part) and applied the necessary changes to keep it up-to-date with the ECMWF software environment. She coordinated and added new implementations and guided the evaluation. She wrote the on-line documentation as well as most of the manuscript.

L. Haimberger is the original author of the software package and provided the first implementation for the use of the ECMWF Web API and the retrieval of ensemble members. He participated in writing introductory and history parts as well as giving feedback on all other parts.

P. Seibert revised the Fortran code and provided the Fortran code documentation and test cases, and wrote the respective section of the manuscript. She also gave feedback on all other parts, and contributed to editing the final manuscript version.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. Over the years, the development of flex_extract was partly funded by the CTBTO. We thank the ZAMG for providing access to the ECMWF MARS archive and the hosting of the community ~~website~~web site. We would also like to thank the ECMWF user support for their assistance in converging the software environment to the current state and for their many publicly available code examples for working with GRIB files. Additionally, we thank Anne Fouilloux for an initial version of the Python routines. Moreover, we thank the users for their feedback and questions which made it possible to make progress in user friendliness, eliminate bugs and react on requirements.

References

Beizer, B.: Software Testing Techniques, Van Nostrand Reinhold, 2nd edn., 1990.

Berrisford, P., Dee, D., Poli, P., Brugge, R., Fielding, M., Fuentes, M., Kållberg, P., Kobayashi, S., Uppala, S., and Simmons, A.: The ERA-Interim archive Version 2.0, ERA Report Series, Vol. 1, https://www.ecmwf.int/node/8174, 2011.

Brandl, G.: Sphinx - Python Documentation Generator, http://www.sphinx-doc.org/en/master/, 2019.

Buizza, R., Richardson, D. S., and Palmer, T. N.: Benefits of increased resolution in the ECMWF ensemble system and comparison with poor-man's ensembles, Quarterly Journal of the Royal Meteorological Society, 129, 1269–1288, https://doi.org/10.1256/qj.02.92, https://rmets.onlinelibrary.wiley.com/doi/abs/10.1256/qj.02.92, 2003.

Copernicus: How to use the CDS API, https://cds.climate.copernicus.eu/api-how-to, 2019.

ECMWF: ERA-Interim: What is the spatial reference, https://confluence.ecmwf.int/display/CKB/ERA-Interim%3A+What+is+the+spatial+reference, 2016a.

ECMWF: ERA5: What is the spatial reference, https://confluence.ecmwf.int/display/CKB/ERA5%3A+What+is+the+spatial+reference, 2016b.

ECMWF: OpenIFS: Horizontal Resolution Configurations, https://confluence.ecmwf.int/display/OIFS/4.2+OpenIFS%3A+Horizontal+Resolution+Configurations, 2017.

ECMWF: ECMWF, https://www.ecmwf.int, [Online; 05.08.2019], 2019a.

ECMWF: MARS User Documentation, https://software.ecmwf.int/wiki/display/UDOC/MARS+user+documentation, [Online; 05.08.2019], 2019b.

ECMWF: Forecasting system upgrade set to improve global weather forecasts, https://www.ecmwf.int/en/about/media-centre/news/2019/forecasting-system-upgrade-set-improve-global-weather-forecasts, 2019c.

ECMWF: Data spatial coordinate systems, https://www.ecmwf.int/en/forecasts/documentation-and-support/data-spatial-coordinate-systems, accessed: August 26, 2019, 2019d.

ECMWF: MARS Catalogue, https://apps.ecmwf.int/mars-catalogue/, 2019e.

ECMWF: Changes in ECMWF model, https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model, 2019f.

ECMWF: IFS Documentation, https://www.ecmwf.int/en/publications/ifs-documentation, 2019g.

ECMWF: ERA5 data documentation, https://confluence.ecmwf.int/display/CKB/ERA5+data+documentation, 2019h.

ECMWF: Parameter database, https://apps.ecmwf.int/codes/grib/param-db, 2019i.

ECMWF: Ecaccess concepts, https://confluence.ecmwf.int/display/ECAC/Ecaccess+concepts, 2019j.

ECMWF: ECMWF Web API Home, https://confluence.ecmwf.int/display/WEBAPI/ECMWF+Web+API+Home, 2019k.

ECMWF: What are the changes from ERA-Interim to ERA5?, 2019l.

ECMWF: Tutorials, https://confluence.ecmwf.int/display/METV/Tutorials, [Online; 18.12.2019], 2019m.

Foord, M. and the mock team: Mock object library, https://mock.readthedocs.io/en/latest/, 2019.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Munoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., Chiara, G. D., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N.: The ERA5 Global Reanalysis, Quart. J. Roy. Met. Soc., submitted, 2020.

Hittmeir, S., Philipp, A., and Seibert, P.: A conservative reconstruction scheme for the interpolation of extensive quantities in the Lagrangian particle dispersion model FLEXPART, Geoscientific Model Development, 11, 2503–2523, https://doi.org/10.5194/gmd-11-2503-2018, https://www.geosci-model-dev.net/11/2503/2018/, 2018.

Hoyer, S. and Hamman, J.: xarray: N-D labeled arrays and datasets in Python, Journal of Open Research Software, 5, https://doi.org/10.5334/jors.148, http://doi.org/10.5334/jors.148, 2017.

Krekel, H.: pytest Documentation, Online, https://buildmedia.readthedocs.org/media/pdf/pytest/latest/pytest.pdf, 2019.

Lacchia, M.: radon 4.0.0 - Project description, https://pypi.org/project/radon/, 2019.

Laloyaux, P., Balmaseda, M., Dee, D., Mogensen, K., and Janssen, P.: A coupled data assimilation system for climate reanalysis, Quarterly Journal of the Royal Meteorological Society, 142, 65–78, 2016.

Laloyaux, P., de Boisseson, E., Balmaseda, M., Bidlot, J.-R., Broennimann, S., Buizza, R., Dalhgren, P., Dee, D., Haimberger, L., Hersbach, H., Kosaka, Y., Martin, M., Poli, P., Rayner, N., Rustemeier, E., and Schepers, D.: CERA-20C: A Coupled Reanalysis of the Twentieth Century, Journal of Advances in Modeling Earth Systems, 10, 1172–1195, https://doi.org/10.1029/2018MS001273, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001273, 2018.

McKinney, W.: Data Structures for Statistical Computing in Python, in: Proceedings of the 9th Python in Science Conference, edited by van der Walt, S. and Millman, J., pp. 51 – 56, 2010.

Palmer, T. N., Barkmeijer, J., Buizza, R., and Petroliagis, T.: The ECMWF Ensemble Prediction System, Meteorological Applications, 4, 301–304, https://doi.org/10.1017/S1350482797000649, 1997.

Pisso, I., Sollum, E., Grythe, H., Kristiansen, N., Cassiani, M., Eckhardt, S., Arnold, D., Morton, D., Thompson, R. L., Groot Zwaaftink, C. D., Evangeliou, N., Sodemann, H., Haimberger, L., Henne, S., Brunner, D., Burkhart, J. F., Fouilloux, A., Brioude, J., Philipp, A., Seibert, P., and Stohl, A.: The Lagrangian particle dispersion model FLEXPART version 10.3, Geoscientific Model Development Discussions, 2019, 1–67, https://doi.org/10.5194/gmd-2018-333, https://www.geosci-model-dev-discuss.net/gmd-2018-333/, 2019.

Ritchie, H., Temperton, C., Simmons, A., Hortal, M., Davies, T., Dent, D., and Hamrud, M.: Implementation of the Semi–Lagrangian Method in a High–Resolution Version of the ECMWF Forecast Model, Mon. Weather Rev., 123, 489–514, 1995.

Simmons, A. J. and Burridge, D. M.: An Energy and Angular–Momentum Conserving Vertical Finite-Difference Scheme and Hybrid Vertical Coordinates, Mon. Weather Rev., 109, 758–766, 1981.

Sneed, H. M., Seidl, R., and Baumgartner, M.: Software in Zahlen, Hanser, 2010.

Spillner, Andreas; Linz, T.: Basiswissen Softwaretest, dpunkt, 5th edn., 2012.

Stohl, A. and Seibert, P.: Accuracy of trajectories as determined from the conservation of meteorological tracers, Q. J. Roy. Meteor. Soc., 125, 1465–1584, 1998.

Stohl, A., Wotawa, G., Seibert, P., and Kromp-Kolb, H.: Interpolation errors in wind fields as a function of spatial and temporal resolution and their impact on different types of kinematic trajectories, J. Appl. Meteorol., 34, 2149–2165, 1995.

Stohl, A., Hittenberger, M., and Wotawa, G.: Validation of the Lagrangian particle dispersion model FLEXPART against large-scale tracer experiment data, Atmospheric Environment, 32, 4245–4264, https://doi.org/10.1016/S1352-2310(98)00184-8, 1998.

Stohl, A., Haimberger, L., Scheele, M., and Wernli, H.: An intercomparison of results from three trajectory models, Meteorol. Applications, 8, 127–135, 2001.

Stohl, A., Forster, C., Frank, A., Seibert, P., and Wotawa, G.: Technical note: The Lagrangian particle dispersion model FLEXPART version 6.2, Atmos. Chem. Phys., 5, 2461–2474, https://doi.org/10.5194/acp-5-2461-2005, 2005.

Thénault, S.: Pylint, https://www.pylint.org/, 2001.

van Rossum, G., Warsaw, B., and Coghlan, N.: PEP 8 – Style Guide for Python Code, Online, https://www.python.org/dev/peps/pep-0008/, 2001.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G.: The NumPy array: A structure for efficient numerical computation, Computing in Science and Engineering, 13, 22–30, https://doi.org/10.1109/MCSE.2011.37, 2011.

Wolff, E.: Continuous Delivery. Der pragmatische Einstieg, Dpunkt Verlag, Heidelberg, 282 pp., 2014.

Wotawa, G., Stohl, A., and Neininger, B.: The urban plume of Vienna: comparisons between aircraft measurements and photochemical model results, Atmospheric Environment, 32, 2479 – 2489, https://doi.org/https://doi.org/10.1016/S1352-2310(98)00021-1, http://www.sciencedirect.com/science/article/pii/S1352231098000211, 1998.

Table 1. Overview of ECWMF data sets with associated parameters required in MARS requests (Berrisford et al., 2011; Laloyaux et al., 2016; ECMWF, 2019e, h). DET-FC stands for "Deterministic forecast", ENS-DA for "Ensemble data assimilation", ENS-CF for "Ensemble control forecast", ENS-CV for "Ensemble validation forecast" and ENS-PF for "Ensemble perturbed forecast". All times are in UTC, all steps in hours. Dates are written as DD/MM/YYYY (day optional). Steps and members are written in the format of Start/End/Step. The ~~information~~ specifications for the operational data sets ~~is~~ are valid for current data at the time of publication (except ENS-CV – deprecated since 8 August ~~8th,~~ 2016)~~.~~. For details about resolution and ~~may change~~ other parameters which have changed in the ~~future~~course of time, see Table 2 and Table 3. The grid type ~~(Oxxx)~~ for the operational data ($T_{CO}xxx$) refers to the spectral cubic octahedral ~~reduced Gaussian~~ grid and for the reanalysis data ($T_L xxx$) refers to linear spherical harmonics. The identification parameter "~~Data set~~Dataset" is to be used by public users only. Note that there is also the ERA40 reanalysis; however, as it has been superseded by ERA-Interim and ERA5 and thus rarely used nowadays, it is not included here (but flex_extract should still be applicable).

	Operational data					**Reanalyses**		
	DET-FC	ENS-DA	ENS-CF	ENS-CV	ENS-PF	ERA-Interim	ERA5	C
Period of data set availability								
	12/1985 – ongoing	22/06/2010 – ongoing	01/05/1994[2] – ongoing	12/09/2006 – 08/03/2016	12/09/2006[3] – ongoing	01/1979 – 12/2018	01/1979 – ongoing[4]	-
Identification (MARS keywords)								
Class	od	od	od	od	od	ei	ea	
Stream	oper	enda/elda[1]	enfo	enfo	enfo	oper	oper	
Field type	fc/an	fc/an	cf	cv	pf	fc/an	fc/an	
~~Data set~~ Dataset	–	–	–	–	–	interim	–	
Time (where forecast starts or analysis is valid)								
Forecast	00/12	06/18	00/12	00/12	00/12	00/12	06/18	
Analysis	0/6/12/18	0/6/12/18	–	–	–	0/6/12/18	0/1/.../23	0
Step (available forecast steps)								
Forecast	0/125/1	1/12/1 3,6,12[7]	0/90/1 93/144/3 150/360/6	0/144/3 150/360/6	336	3/240/3[5]	0/18/1	
Horizontal grid type and resolution, number of vertical levels								
Grid[8]	~~O1280~~ $T_{CO}1279$ (0.07°)	~~O640~~ $T_{CO}639$ (0.141°)	~~O640~~ $T_{CO}639$ (0.141°)	~~O640~~ $T_{CO}639$ (0.141°)	~~O640~~ $T_{CO}639$ (0.141°)	~~T255~~ $T_L 255$ (0.75°)	~~T639~~ $T_L 639$ (0.25°)	~~T~~
Levels	137	137	91	91	91	60	137	
Ensemble members	–	0/50/1	–	–	1/50/1	–	–	
Availability of $\dot{\eta}$	yes[6]	yes	no	no	no	no	yes	

[1] From 22/06/2010 to 18/11/2013, ENS-DA was stored in stream ENDA, afterwards in stream ELDA.

[2] Exists since 11/1992, but the available dates were unregular in the beginning before 01/05/1994.

[3] The data set exists from 11/1992, but model level data are available only from 12/09/2006 on.

[4] Available with a delay of ca. 3 months. Fast track data with shorter delay are now also available, but subject to possible revisions

[5] For public users, the forecast model level fields are not available.

[6] Available as MARS parameter since 04/06/2008.

[7] On 11/06/2019, the steps changed from 1/12/1 to the single steps 3,6,12.

[8] See Table 4 for correspondence of grid types.

Table 2. List of the evolution of the spatial resolution of the IFS operational forecasts. Changes are marked in bold. The ensemble data are usually provided with higher ~~resoltion~~ resolution for Lag A (1–10 d) than for Lag B (10–15 d). The first part of each entry is the horizontal resolution marked with a "T" for spectral ~~or~~ representation; with "~~O~~T" ~~for~~ representing the linear and "T_{CO}" the cubic octahedral representation. The second part, marked with "L", is the number of vertical model levels. In the case of ensembles, the number N of members is written in front of the resolution as N*. Source: Palmer et al. (1997); Buizza et al. (2003); ECMWF (2019c, e, f, g).

	DET-FC	ENS-DA	ENS-CF	ENS-CV	ENS-PF
20/04/1983	**T106L16**				
13/05/1986	T106**L19**				
17/09/1991	**T213**L19				
17/19/1991	T213**L31**				
01/05/1994	T213L31		**T63L19**		
10/12/1996	T213L31		**T159L31**		
01/04/1998	**T319**L31		T159L31		
09/03/1999	T319**L50**		T159L31		
12/10/1999	T319**L60**		T159**L40**		
21/11/2000	**T511**L60		**T255**L40		
01/02/2006	**T799L91**		**Lag A T399L62**		
			Lag B T255L62		
12/09/2006	T799L91			**Lag A 2*T399L62**	**Lag A 50*T399L6**
				Lag B 2*T255L62	**Lag B 50*T255L6**
26/01/2010	**T1279**L91		**Lag A T639**L62	Lag A 2***T639**L62	Lag A 50***T639**L6
			Lag B T319L62	Lag B 2***T319**L62	Lag B 50***T319**L6
22/06/2010	T1279L91	**25*T399L91**	Lag A T639L62	Lag A 2*T639L62	Lag A 50*T639L6
			Lag B T319L62	Lag B 2*T319L62	Lag B 50*T319L6
01/11/2011	T1279L91	25*T399L91	Lag A T639L62	Lag A 2*T639L62	Lag A 50*T639L6
			Lag B T319L62	Lag B 2*T319L62	Lag B 50*T319L6
25/06/2013	T1279**L137**	25*T399**L137**	Lag A T639L62	Lag A 2*T639L62	Lag A 50*T639L6
			Lag B T319L62	Lag B 2*T319L62	Lag B 50*T319L6
19/11/2013	T1279L137	25*T399L137	Lag A T639L62	Lag A 2*T639**L91**	Lag A 50*T639L6
			Lag B T319L62	Lag B 2*T319**L91**	Lag B 50*T319L6
20/11/2013	T1279L137	25*T399L137	Lag A T639L91	Lag A 2*T639L91	Lag A 50*T639**L9**
			Lag B T319L62	Lag B 2*T319L91	Lag B 50*T319**L9**
08/03/2016	~~**O1280**~~**T_{CO}1279**L137	25*~~**O640**~~**T_{CO}639**L137	~~**O640**~~**T_{CO}639**L91	deprecated	50*~~**O640**~~**T_{CO}639**L9
11/06/2019	~~O1280L137~~ T_{CO}1279L137	**50***~~O640L137~~*T_{CO}639L137	~~O640L91~~ T_{CO}639L91	deprecated	~~50*O640L91~~ 50*T_{CO}63

Table 3. List of the evolution of forecast steps and forecast start times for data sets DET-FC and ENS-CF. "Lag s" denotes different temporal resolution for forecast ranges s; "#steps" is the total number of steps. Source: (ECMWF, 2019e)

	DET-FC					ENS-CF			
	#steps	Lag 1	Lag 2	Lag 3		#steps	Lag 1	Lag 2	Lag 3
01/04/1985	20[1]	12/240/12			01/04/1994	33	0/12/3	18/120/6	132/240/12
	forecast start time 12 UTC					forecast start time 12 UTC			
01/07/1985	30	6/144/6	156/240/12		31/07/1997	55	0/12/3	18/120/6	132/504/12
15/11/1990	32	3/12/3	18/144/6	150/240/12	09/06/1999	65	0/12/3	18/240/6	252/504/12
20/01/1999	42	3/12/3	18/240/6		25/03/2003	two forecast start times per day: 0/12 UTC			
12/09/2000	two forecast start times per day: 0/12 UTC				29/09/2004	63	0/240/6	252/504/6	
24/10/2000	52	3/72/3	78/240/6		13/09/2006	85	0/132/3	138/240/6	252/504/12
29/06/2005	85	0/132/3	138/240/6	252/504/12	22/06/2015	four forecast times per day: 0/6/12/18 UTC			
05/10/2005	87	0/144/3	150/240/6	252/504/12	"	49 (6/18 UTC)	0/144/3		
14/03/2006	57	0/96/3	102/240/6		23/11/2016	145 (0/12UTC)	0/90/1	93/144/3	150/260/6
13/09/2006	65	0/144/3	150/240/6		"	109 (6/18UTC)	0/90/1	93/144/3	
16/11/2011	125	0/90/1	93/144/3	150/240/6	–	–			

[1] Only surface fields.

Table 4. Approximate correspondences between spectral, Gaussian, and latitude / longitude grid resolutions. Source: ~~ECMWF (2019d, e); Berrisford et al. (2011); Laloyaux et al. (2016)~~ECMWF (2017, 2019d, e); Berrisford et al. (2011); Laloyaux et al. (2016) . For the spectral grid the truncation number is denoted by "T" ~~or "T_L"~~ where the ~~latter~~ subscript "q" means quadratic grid and "l" means linear ~~spectral truncation~~grid. The quadratic grid can not be selected with flex_extract. The corresponding reduced Gaussian grids are denoted by "N" followed by the number of lines between the pole and the equator. ~~Only linearly truncated grids can be selected with flex_extract.~~ The new octahedral grid is denoted by "T_{CO}", meaning "spectral cubic octahedral"; they correspond to a octahedral reduced Gaussian grid, denotes with an "O".

Spectral	Gaussian Grid		Lat / Lon
~~T63~~ T_q63	N48	~~200~~ 209 km	1.875°
T_L95	N48	209 km	1.875°
~~T106~~ T_q106	N80	~~120~~ 125 km	1.125°
T_L159	N80	~~120~~ 125 km	1.125°
~~T213~~ T_q213	N160	63 km	0.5625°
T_L255	N128	~~80~~ 78 km	0.75°(*)
T_L319	N160	~~60~~ 63 km	0.5625°
T_L399	N200	50 km	0.45°
T_L511	N256	~~40~~ 39 km	~~0.351~~0.352°
T_L639	N320	31 km	0.25°(*)
T_L799	N400	25 km	0.225°
T_L1023	N512	20 km	0.1758°
T_{CO}639	O640	18 km	0.141°
T_L1279	N640	16 km	~~0.141~~0.1406°
T_{CO}1279	O1280	9 km	0.07°

(*) As `GRIB1` only supports three decimals, ECMWF recommends to round the resolutions to $0.75°$ in the case of ERA-Interim (exact value: $0.703125°$) and to $0.25°$ for ERA5 (exact value: 0.28125) (ECMWF, 2016a, b). See also Table 1.

Table 5. Directory structure of the flex_extract v7.1 root directory.

File / subdirectory	Content	Description
Documentation/	html/	offline version of documentation
For_developers/	Flowcharts	source and PNG files of flow diagrams
	FORD	source files for Fortran code documentation
	Sphinx	source files for documentation
	*.xls, *.sh, *	documentation files, scripts and ~~infos~~ info for developers
Run/	Control/	contains all example CONTROL files
	Jobscripts/	empty after distribution download; later contains korn shell job
	Workspace/	not present before first local retrieval; contains downloaded data
	ECMWF_ENV	contains ~~infos~~ info about user credentials
	run.sh	top-level script to start flex_extract
	run_local.sh	top-level script to start flex_extract in *local mode*
Source/	Fortran	complete Fortran program incl. makefiles
	Python	Python source files
	Pythontest	Python unit tests
Templates/	~~compilejob~~installscript.template	template for the installation on ECMWF server
	~~convert.nl~~ calc_etadot_nml.template	namelist template for the calc_etadot program
	ECMWF_ENV.template	template for the ECMWF user credentials
	ecmwf_grib1_table_128	table for the assigment of parameter names and ids
	~~job~~jobscript.template	~~ob~~ job script template for ECMWF batch mode before the insta
	~~job.temp~~ submitscript.template	job script template after installation (now includes settings such
Testing/	Installation	data for an installation check
	Regression	regression test cases
CODE_OF_CONDUCT.md		rules for contribution to flex_extract
LICENSE.md		full ~~license~~ licence text
README.md		short introduction to ~~the software~~flex_extract
setup.sh		installation script

Table 6. Necessary account registrations per user and application mode for each data set. The registration procedure is indicated by numbers 1–3 and explained below.

Data sets	Member-state user			Public user
	Remote	Gateway	Local	Local
Operational	1	1	1, 2	-
ERA-Interim	1	1	1, 2	2
CERA-20C	1	1	1, 2	2
ERA5	1	1	3	~~-~~3

No.	Registration procedure
1	Access as a member-state user. Account granted by the Computing Representative. Credentials have to be provided during installation.
2	Access through the ECMWF Web API. One needs to sign in at the ECMWF Web API and to configure the ECMWF key as described (ECMWF, 2019k). ~~Member state~~ Member-state users can sign in with their credentials. Public users have to register for obtaining an account.
3	Access through the CDS API (Copernicus, 2019). Registration at CDS and configuration of the CDS key needed.

Table 7. ~~General software and library~~ Software dependencies for flex_extract ~~Python and Fortran parts~~ in all application modes.

Python	Fortran
Python3	gfortran[1] / CrayPE ftn[2]
numpy	fftw3
genshi	emoslib
ecCodes ~~for Python~~ for Python (>= v2.13.0)	ecCodes ~~for~~ for Fortran

[1]Remote mode / gateway mode on ecgate, and local mode. [2]Remote mode gateway mode on cca / ccb.

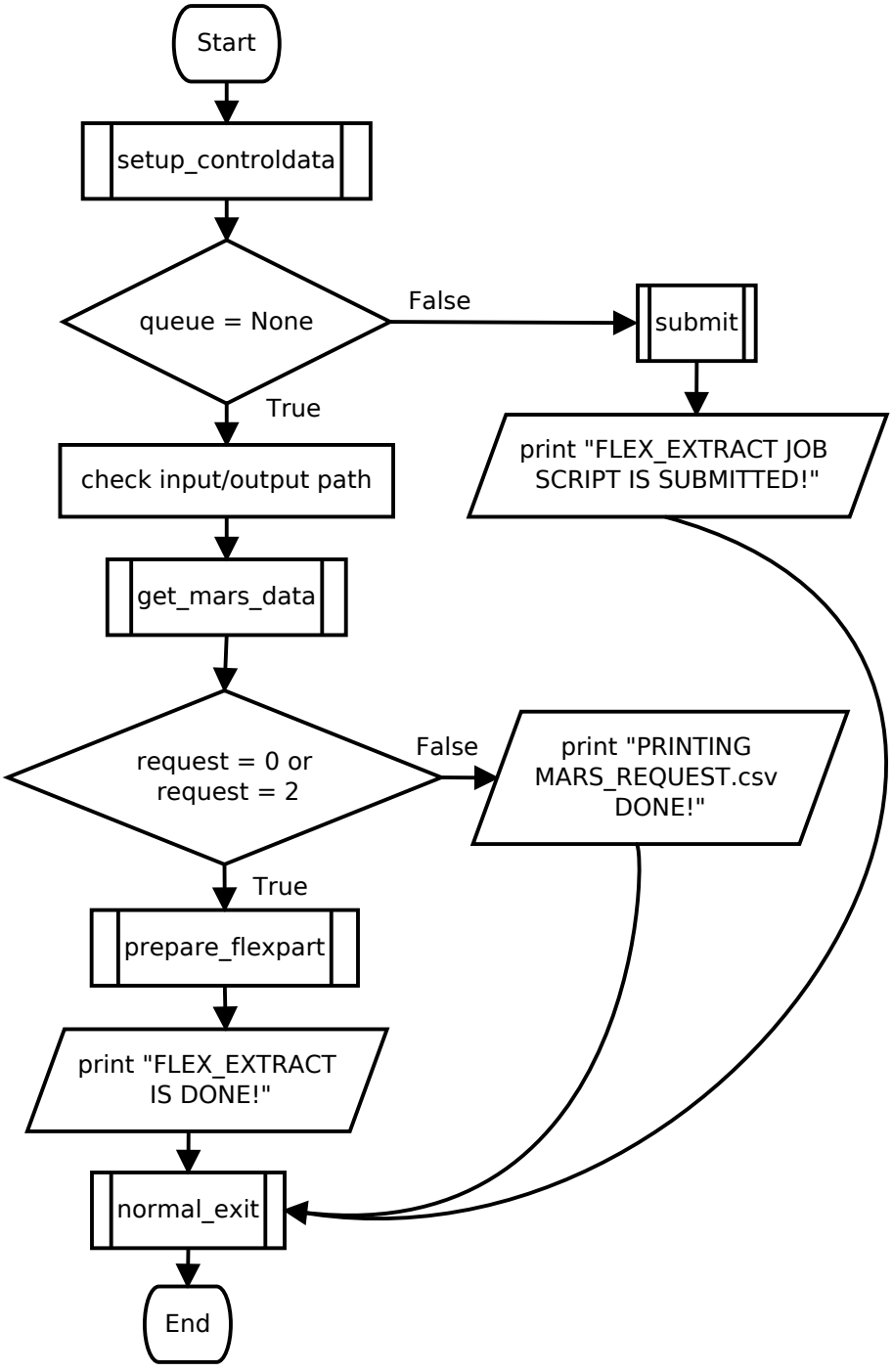

Figure 3. Flow diagram for the ~~*local*~~ local application mode. If `queue` $\neq$ `None`, flex_extract ~~assumes that a job script has to be sent to an ECMWF server and selects another branch shown~~ was started in remote or gateway mode and Fig. ~~–~~2 applies. This is marked by the *submit* block. In the case of `request == 1`, flex_extract ~~is forced to skip~~ skips the retrieval and post-processing steps and just writes the ~~`mars_request`~~ `mars_request.csv` file. Within the ~~pure~~ local mode, the retrieval (`get_mars_data`) and post-processing (`prepare_flexpart`) parts are ~~conducted~~executed. Symbols as in Fig. 2.

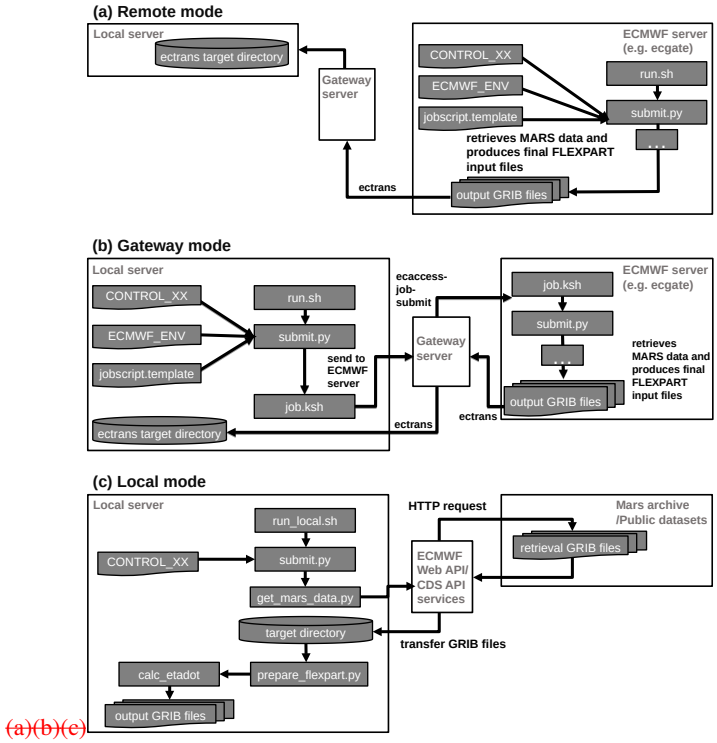

Figure 4. General overview of the work flows and work locations in the different application modes: (a) remote, (b) gateway, and (c) local mode. The files and scripts used in each mode are outlined.

Table 8. Part 1 of the overview of CONTROL file parameters. A more detailed description on parameter handling, setting and value ranges is given in the supplemental material.

Parameter	Default value	Format	Description
Time section			
START_DATE	None	String [YYYYMMDD]	first day of retrieval period
END_DATE	None	String [YYYYMMDD]	last day of retrieval period
DATE_CHUNK	3	Integer	number of days within one MARS request
DTIME	None	Integer	time step
BASETIME	None	Integer	end time for half-day retrievals
Data section			
CLASS	None	String [xx]	data set class identifier in MARS archive
DATASET	None	String	public data set identifier
STREAM	None	String [xxxx]	identifier for forecasting stream
NUMBER	'OFF'	String [i/to/i]	ensemble member numbers
EXPVER	1	Integer	experiment number
FORMAT	'GRIB1'	String	output format of ~~GRIB~~ GRIB fields
Data fields section			
TYPE	None	list of strings [xx xx ... xx]	list of field type per TIME
TIME	None	list of strings [xx xx ... xx]	list of times
STEP	None	list of strings [xx xx ... xx]	list of forecast steps corresponding to TIME
MAXSTEP	None	Integer	maximum forecast step
Flux data fields section			
ACCTYPE	None	String	type of the flux forecast fields
ACCTIME	None	String [i/i]	forecast times of flux fields
ACCMAXSTEP	None	Integer	maximum forecast step of flux fields
RRINT	0	Integer	switch to select method for precipitation disaggregation
Domain section			
GRID	None	String [i/i]	horizontal resolution on longitude/latitude grid
RESOL	None	String	horizontal resolution of spectral grid
SMOOTH	0	Integer	spectral truncation of $\dot{\eta}$ on Gaussian grid
LEFT	None	String	longitude of lower left domain corner
LOWER	None	String	latitude of lower left domain corner
UPPER	None	String	latitude of upper right domain corner
RIGHT	None	String	longitude of upper right domain corner
LEVEL	None	Integer	maximum number of vertical levels
LEVELIST	None	String [start/to/end]	definition of vertical levels
Vertical velocity section			
GAUSS	0	Integer	switch to calculate $\dot{\eta}$
ACCURACY	24	Integer	number of bits per value in ~~GRIB~~ GRIB coded fields
OMEGA	0	Integer	switch to retrieve ω from MARS
OMEGADIFF	0	Integer	switch to calculate ω and ~~Dp_s/Dt~~ $\partial p_s/\partial t$ from continuity equation
ETA	0	Integer	switch to read $\dot{\eta}$ from MARS
ETADIFF	0	Integer	switch to calculate $\dot{\eta}$ and Dp_s/Dt from continuity equation
DPDETA	1	Integer	switch to select multiplication of $\dot{\eta}$ by ~~$dp/d\eta$~~ $\partial p/\partial \eta$
ETAPAR	77	Integer	~~GRIB~~ GRIB parameter id for $\dot{\eta}$~~$/dp/d\eta$~~$/\partial p/\partial \eta$

Table 8. Part 2 of the overview of CONTROL file parameters.

Parameter	Default value	Format	Description
General section			
DEBUG	0	Integer	switch to save the temporary files
REQUEST	0	Integer	switch to create the file mars_requests.csv
PUBLIC	0	Integer	switch to select public ~~WebAPI~~ Web API access
OPER	0	Integer	switch to prepare operation job script
ECSTORAGE	0	Integer	switch to store results in ECFS file system
ECTRANS	0	Integer	switch to transfer final files to local system
PREFIX	'EN'	String	front string in file names before the date string
ECFSDIR	'ectmp:/$USER/ econdemand/'	String	destination directory on ECFS file system
MAILFAIL	['$USER']	List of strings	list of emails to send log files to
MAILOPS	['$USER']	List of strings	list of emails to send log files to
Additional data section			
CWC	0	Integer	switch to retrieve total cloud water content
DOUBLEELDA	0	Integer	switch to manually double ensemble member number
ADDPAR	None	String [p1/p2/. . . /pn]	additional surface fields to retrieve

Table 9. Description of the parameters stored in file ECMWF_ENV.

Parameter	Default value	Format	Description
ECUID	None	String	ECMWF user id
ECGID	None	String	ECMWF group id
DESTINATION	None	String	ectrans association
GATEWAY	None	String	name or ip address of member gateway server

Table 10. Overview of templates used in flex_extract. They are stored in the `Templates` directory.

Template	Description
~~convert~~calc_etadot_nml.~~nl~~template	Used to create a Fortran namelist file called `fort.4`. It will be created in the Python part and contains controlling options for `calc_etadot`. (See Table 14)
`ecmwf_env.template`	Used to create the `ECMWF_ENV` file within application modes ~~*gateway* and *remote*~~gateway and remote.
~~compilejob~~installscript.template	Used to create the file `compilejob.ksh` during the installation process for the application modes ~~*remote* and *gateway*. `job.temp` Used to create the actual job script file called `job.ksh` for the execution of flex_extract in the application modes *remote* and *gateway*~~remote and gateway.
~~job~~jobscript.template	Used to create the template ~~job~~submitjob.~~temp~~template in the installation process. A couple of parameters are set, such as the user credentials and the flex_extract version number.
submitjob.template	Used to create the actual job script file called `job.ksh` for the execution of flex_extract in the application modes remote and gateway.

Table 11. Overview of parameters to be set in the `setup.sh` script for installation. In ~~case of *remote*~~ remote and ~~*local*~~ local mode for ~~member state~~ member-state users, the file `ECMWF_ENV` will be created, hence the parameters from Table 9 must also be set in the `setup.sh` script.

Parameter	Default value	Format	Description
`TARGET`	None	String	defines location and therefore the application mode
`MAKEFILE`	~~Makefile.gfortran~~ None	String	Makefile for compiling `calc_etadot`
`JOB_TEMPLATE`	~~job.template~~ jobscript.template	String	batch job template for ~~*gateway* and *remote*~~ gateway and remote
`INSTALLDIR`	`$HOME` on ECMWF servers; `pwd` in ~~*local*~~ local mode	String	root path for flex_extract working directory
`CONTROLFILE`	`CONTROL_ERA5`	String	input file with parameter settings

Table 12. Overview of the parameter to be set in the `run*.sh` script. In order to provide a complete list, some already defined parameters from Tables 8 ~~, 8~~ and 11 are repeated here. In the case of a special format, a sample format is given in parentheses; f denotes a floating-point number.

Parameter	Default value	Format	Description
START_DATE	None	String (YYYYMMDD)	first day of retrieval period
END_DATE	None	String (YYYYMMDD)	last day of retrieval period
DATE_CHUNK	3	Integer	number of days within one ~~mars~~ MARS request
BASETIME	None	Integer	end time for half-day retrievals
STEP	None	blank ~~seperated~~ separated list of numbers	list of forecast steps of corresponding retrieval times
LEVELIST	None	String (start/to/end)	defines list of vertical levels
JOB_CHUNK	None	Integer	number of days to be retrieved within a single job
AREA	-	String (f/f/f/f)	domain defined as north/west/south/east
PUBLIC	0	Integer	set to 1 for using public access mode
INPUTDIR	None	String	path to temporary working directory
OUTPUTDIR	None	String	path where final output files are stored
PPID	None	Integer	parent process id of the job (only for debugging)
JOB_TEMPLATE	~~job~~submitjob.~~String~~template	String	job template file for ECMWF batch queue
QUEUE	None	String	in case of non-local mode, the ECMWF server name
CONTROLFILE	CONTROL_ERA5	String	input file with parameter settings
RRINT	0	Integer	set to 1 to select new method for precipitation ~~dis-aggregation~~ disa
REQUEST	0	Integer	set to 1 to create the file `mars_requests.csv`
OPER	0	Integer	set to 1 for operational mode (job script)
DEBUG	0	Integer	set to 1 to save the temporary files

Table 13. List of flux fields retrieved by flex_extract and the disaggregation ~~scheme~~ schemes (precip: Eq. 1 or Sect. 3.6.2, flux: Eq. 4) applied.

Short name	Name	Unit	~~Interpolation~~ Disaggregation
LSP	large-scale precipitation	m	~~linear~~ precip
CP	convective precipitation	m	~~linear~~ precip
SSHF	surface sensible heat flux	$\mathrm{Jm^{-2}}$	~~bicubic~~ flux
EWSS	eastward turbulent surface stress	$\mathrm{Nm^{-2}s}$	~~bicubic~~ flux
NSSS	northward turbulent surface stress	$\mathrm{Nm^{-2}s}$	~~bicubic~~ flux
SSR	surface net solar radiation	$\mathrm{Jm^{-2}}$	~~bicubic~~ flux

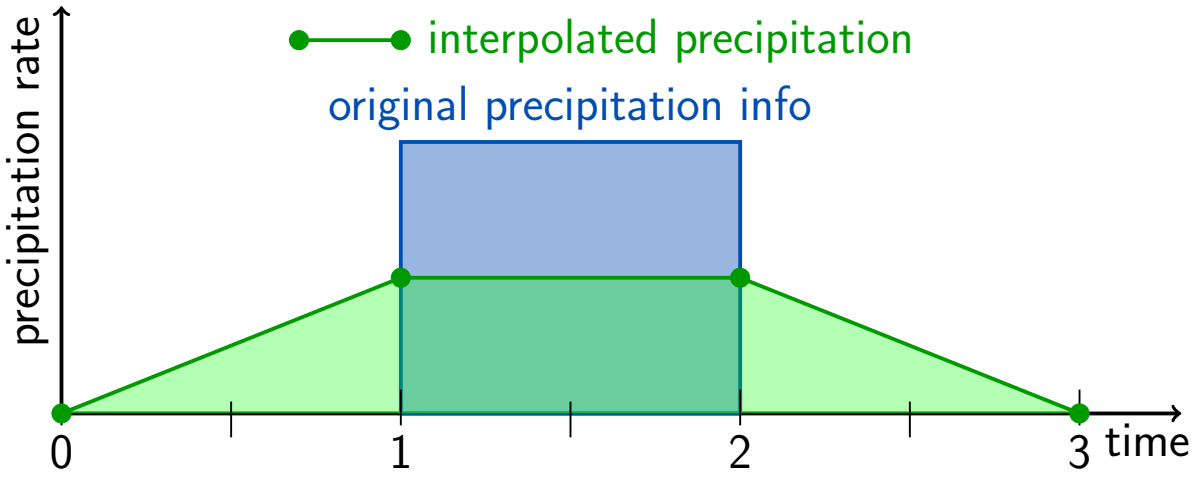

Figure 5. Example of disaggregation scheme as implemented in older versions of flex_extract for an isolated precipitation event lasting one time interval (thick blue line). The amount of original precipitation after de-accumulation is given by the blue-shaded area. The green circles represent the discrete grid points after disaggregation. FLEXPART interpolates linearly between them as indicated by the green line and the green-shaded area. Note that supporting points for the interpolation are shifted by half a time interval compared to the other meteorological fields. From Hittmeir et al. (2018).

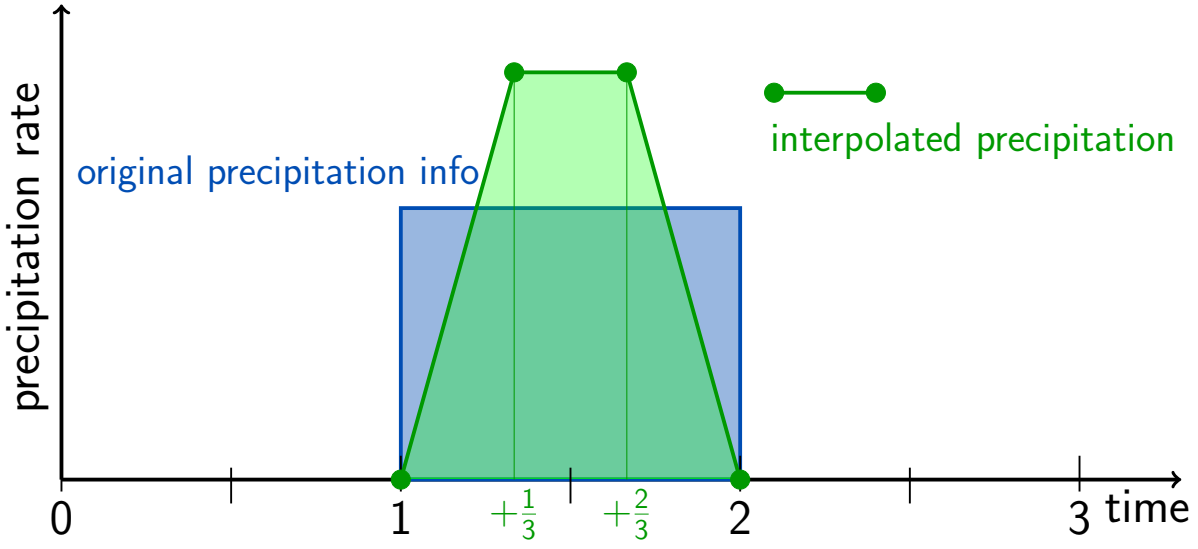

Figure 6. As Figure 5, but with the new interpolation scheme using additional sub-grid points. From Hittmeir et al. (2018).

Table 14. Overview of options controlling `calc_etadot`. Note that the resolution of the latitude-longitude grid is given implicitly by the grid dimensions and extent.

Parameter	Description	Remarks
`maxl`	grid dimension – longitudes	
`maxb`	grid dimension – latitudes	
`mlevel`	grid dimension – number of levels	
`mlevelist`	list of levels	to be given in MARS request notation like `1/to/91`
`mnauf`	number of spectral coefficients in input data	
`metapar`	GRIB ID of vertical velocity in output	standard FLEXPART expects `=77`
`rlo0`	Western border of domain in degree	
`rlo1`	Eastern border of domain in degree	
`rla0`	Southern border of domain in degree	
`rla1`	Northern border of domain in degree	
`momega`	if 1, ω is calculated from $\dot{\eta}$ and output	for testing the accuracy of calculated $\dot{\eta}$ if no $\dot{\eta}$ from MARS is available
`momegadiff`	if 1, calculated ω is compared with ω from MARS	
`mgauss`	if 1, evaluate continuity equation on GG	
`msmooth`	if $\neq 0$, apply spectral smoothing by clipping at given truncation	
`meta`	if 1, use $\dot{\eta}$ from input.	
`metadiff`	if 1 and `meta=0`, $\dot{\eta}$ needs to be available from MARS and this is compared with calculated $\dot{\eta}$	for testing the accuracy of $\dot{\eta}$ calculation
`mdpdeta`	if 1, give $\dot{\eta}_p$ as output	with the current version of FLEXPART, only `=1` is useful; future versions might used $\dot{\eta}$.

Table 15. Determination of the method for obtaining $\dot{\eta}$ in `calc_etadot` as a funtion of control parameters (see also Table 14). GG stands for Gaussian grid. The names of the corresponding regression tests (see ~~Section~~ Sect. 5.4) are also given

Method	`mgauss`	`meta`	Test name
Continuity eq. on lat-lon grid	0	0	`latlon`
Continuity eq. on GG	1	0	`gauss`
Use $\dot{\eta}$ from input	0	1	`etadot`
(Program will stop with ERROR)	1	1	–

Table 16. List of `fort` files generated by the Python part to serve as input for the Fortran program, and the output file of `calc_etadot`. If the optional fields were not extracted, the corresponding files are empty.

Number	Content
Input to the Fortran program `calc_etadot`	
10	U and V wind components
11	temperature
12	logarithm of surface pressure
13	divergence (optional)
16	surface fields
17	specific humidity
18	surface specific humidity (reduced Gaussian)
19	vertical velocity (pressure) (optional)
21	eta-coordinate vertical velocity (optional)
22	total cloud water content (optional)
Output from Fortran program `calc_etadot`	
15	U and V wind components, $\dot{\eta}$, temperature, surface pressure, specific humidity

Table 17. List of model level parameters FLEXPART requires to run and the availability in the different data sets (ECMWF, 2019e, i). The cloud-water content fields are optional. The divergence and logarithm or surface pressure fields are only necessary for the calculation of the vertical velocity when $\dot{\eta}$ is not available directly. These fields are not transferred to the FLEXPART input files. FC stands for "forecast" and AN for "analysis".

Variables	Short name	Parameter ID	Unit	Operational		ERA-Interim		ERA5		CERA-20C	
				FC	AN	FC	AN	FC	AN	FC	AN
Temperature	T	130	K	x	x	x	x	x	x	x	x
Specific humidity	Q	133	$kgkg^{-1}$	x	x	x	x	x	x	x	x
U – wind component	U	131	ms^{-1}	x	x	x	x	x	x	x	x
V – wind component	V	132	ms^{-1}	x	x	x	x	x	x	x	x
Eta-coordinate vertical velocity	etadot	77	s^{-1}	x[2]	x[2]	-	-	x	x	x	x
Divergence	D	155	kgm^{-2}	x	x	x	x	x	x	x	x
Specific cloud liquid water content	clwc	246	$kgkg^{-1}$	x	x	x	x	x	x	x	x
Specific cloud ice water content	ciwc	247	$kgkg^{-1}$	x	x	x	x	x	x	x	x
Logarithm of surface pressure[1]	lnsp	152	-	(x)	(x)	(x)	(x)	(x)	(x)	(x)	(x)

[1]Only available on model level 1.

[2]Available from 4 June 2008 onward.

Table 18. List of surface level parameters FLEXPART requires to run and ~~the~~ their availability ~~in the~~ from different data sets (ECMWF, 2019e, i). FC stands for "forecast" and AN for "analysis". ~~Deviating~~ Special or future versions of FLEXPART or pre/post-processing software may require additional surface level fields which are not listed here.

Variables	Short name	Parameter ID	Unit	Operational		ERA-Interim		ERA5		CERA-20C	
				FC	AN	FC	AN	FC	AN	FC	AN
2 metre temperature	2t	167	K	x	x	x	x	x	x	x	x
2 metre dewpoint temperature	2d	168	K	x	x	x	x	x	x	x	x
10 metre U wind component	10u	165	ms^{-1}	x	x	x	x	x	x	x	x
10 metre V wind component	10v	166	ms^{-1}	x	x	x	x	x	x	x	x
Geopotential	z	129	$m^{-2}s^{-2}$	x	x	x	x	x	x	x	x
Land-Sea Mask	lsm	172	$0-1$	x	x	x	x	x	x	x	x
Mean sea level pressure	msl	151	Pa	x	x	x	x	x	x	x	x
Snow depth	sd	141	m of w. eq.	x	x	x	x	x	x	x	x
Standard deviation of orography	sdor	160	-	-	x	-	x	-	x	-	x
Surface pressure	sp	134	Pa	-	x	x	x	-	x	x	x
Total cloud cover	tcc	164	$0-1$	x	x	x	x	x	x	x	x
Convetice precipitation	cp	143	m	x	-	x	-	x	-	x	-
Large-scale precipitation	lsp	142	m	x	-	x	-	x	-	x	-
Surface sensible heat flux	sshf	146	Jm^{-2}	x	-	x	-	x	-	x	-
Eastward turbulent surface stress	ewss	180	$Nm^{-2}s$	x	-	x	-	x	-	x	-
Northward turbulent surface stress	nsss	181	$Nm^{-2}s$	x	-	x	-	x	-	x	-
Surface net solar radiation	ssr	176	Jm^{-2}	x	-	x	-	x	-	x	-
Forecast surface roughness[1]	fsr	244	m	x	-	-	-	x	x	x	x

[1]Necessary in CERA-20C due to missing surface roughness parameter

Table 19. Basic metrics.

Version	LOC	SLOC	Comments	Multi	Blank
7.0.4	2538	1820	346	13	374
7.1	7543	2842	1072	2265	1397

Table 20. Ranks of cyclomatic complexity (CC) taken from the manual of the Python package `radon` (Lacchia, 2019).

CC score	Rank	Risk
1 - 5	A	low – simple block
6 - 10	B	low – well structured and stable block
11 - 20	C	moderate – slightly complex block
21 - 30	D	more than moderate – more complex block
31 - 40	E	high – complex block, alarming
41+	F	very high – error-prone, unstable block