Software architecture of an ideal modeling platform in air quality – A first step: Polyphemus

Vivien Mallet*, Denis Quélo, Bruno Sportisse

Cerea: Teaching and Research Center in Atmospheric Environment
Joint laboratory École Nationale des Ponts et Chaussées / Électricité de France R&D
http://www.enpc.fr/cerea/
Clime
Joint team Inria/École Nationale des Ponts et Chaussées
http://www-rocq.inria.fr/clime/
Vivien.Mallet@cerea.enpc.fr, Denis.Quelo@cerea.enpc.fr,
Bruno.Sportisse@cerea.enpc.fr
ENPC – CEREA, 6–8 avenue Blaise Pascal, Cité Descartes, Champs-sur-Marne,
77455 Marne la Vallée Cedex 2

Abstract

This paper investigates the main features of an ideal platform for air-quality modeling. A key issue is to derive a software architecture in which four distinct components are identified: the databases, libraries for data processing and for the physical parameterizations, programs to generate the input data to the chemistry-transport model, and the chemistry-transport model itself which mainly retains the numerical integration. The introduction of high-level applications, such as ensemble forecast or data assimilation, is also addressed. The proposed architecture provides a strong flexibility, and it enables to share developments with other teams in the community. The advocated structure has been partially implemented with the system Polyphemus.

Key words: databases, libraries, data processing, parameterizations, chemistry-transport models.

1 Introduction

The software architecture of the modeling systems based on chemistry-transport models should be carefully addressed due to the diversity of the target applications, the large amount of processed data and the complexity of the currently developed models and methods (notably in data assimilation).

Modeling activities in air quality range from the dispersion of passive tracers at small scales (a few hundred meters) to the simulation of aerosol distributions at continental (or even global) scale. These activities share common models (at least for the transport) but there is still a strong diversity due to the dedicated add-ons such as the chemical mechanisms. Moreover, within a given application, there is some diversity in the fields involved, from chemical to meteorological fields, and in the goals, e.g. forecasts or data assimilation.

The maturity of the models allows to use complex methods such as the data assimilation methods or the ensemble forecasts. A well designed modeling system should allow to apply and to test these methods without endless developments.

One should also stress the large amount of data that is processed. It requires safe and robust software. In addition, efficient software is necessary because of the high computational costs of the time integration (especially for models with aerosols or for data assimilation).

From the previous remarks, the need for well designed software is obvious if one wants to use advanced methods in a safe, efficient and perennial framework. Current comprehensive 3D modeling systems gather four main components:

*Corresponding author.

1. the databases and data processing;
2. the physical parameterizations (turbulence closure, deposition velocities, etc.);
3. the numerical core of the chemistry-transport model;
4. the high level methods, in which the chemistry-transport model is simply viewed as a function, such as the ensemble forecasts or data assimilation.

These four components are usually combined and mixed in an all-in-one system called chemistry-transport model. This historical conception blends components whose natures are completely different. We propose a new architecture that is better suited to carry the modern high-level applications. The system Polyphemus – initiated at Cerea[1] and currently developed at Cerea, Inria[2] and IRSN[3] – is an illustration of what we advocate.

This paper first explains the context, the target applications and, this way, the main constraints and aims of a full modeling system. Then the software architecture is shown, and details about its components are given. An existing system, close to the proposed one, Polyphemus, is described. Finally the next steps to achieve a full modeling system are drawn.

2 Chemistry-transport models

2.1 Brief introduction

A chemistry-transport model computes the concentrations of a set of pollutants over a given domain (usually three-dimensional) and a given period. It takes into account the transport (wind advection and turbulent diffusion), the chemical reactions (which depend on the target application) and the additional fluxes such as the boundary conditions, the emissions and the deposition fluxes.

Basically, a chemistry-transport model solves the following equation:

$$\frac{\partial c_i}{\partial t} + \operatorname{div}\left(V c_i\right) = \operatorname{div}\left(\rho K \cdot \nabla \frac{c_i}{\rho}\right) + \chi_i(c,t) + E_i - D_i \tag{1}$$

where:

- c, the unknown, is the vector of pollutants concentrations and c_i its i-th component;
- V is the wind velocity (three-dimensional vector);
- ρ is the air density;
- K is the 3×3 diffusion matrix, $-\operatorname{div}\left(\rho K \cdot \nabla \frac{c_i}{\rho}\right)$ being a parameterization for the turbulent diffusion;
- $\chi_i(c,t)$ stands for the chemistry (production and loss due to chemical reactions), notice its dependence on c that couples the equations of all individual species;
- E is the vector of emission rates;
- D is the vector of deposition fluxes.

Additional terms may be added, e.g. for convection.

[1] Teaching and Research Center in Atmospheric Environment – Joint laboratory École Nationale des Ponts et Chaussées / Électricité de France R&D – `http://www.enpc.fr/cerea/`.

[2] The French National Institute for Research in Computer Science and Control – Clime team: `http://www-rocq.inria.fr/clime/`.

[3] The Institute for Radiological Protection and Nuclear Safety – `http://www.irsn.fr/`.

2.2 Numerical issues

Building a chemistry-transport model of practical interest implies the use of relevant numerical schemes: there are applications for which the computational costs may be very high (especially if aerosols are taken into account). Most chemistry-transport models use finite differences or finite volumes to solve the transport equation and tailored time integrators for the stiff equations coming from chemistry. For details, one should refer to the dedicated papers. For instance, the numerical strategy for the model Polair3D (part of Polyphemus) is explained in Boutahar et al. (2004). One may refer to Verwer et al. (1998) for a general overview.

To properly understand this paper, one should know that the numerical issues are not prominent in the modeling process due to the satisfactory algorithms that are now available. The attention should be directed to the code architecture instead, in particular to allow the automatic differentiation (Mallet and Sportisse, 2004). One should also notice that most chemistry-transport models use similar spatial discretizations. It is therefore possible to share some code above the numerical schemes themselves.

2.3 Input fields

Relevant numerical schemes have been available for a while. They cover the needs of most applications. Therefore the challenge rather lies in the manner the input fields are computed. Several fields are provided by meteorological models (winds, temperature, etc.) and other fields are computed with physical parameterizations.

For instance, the deposition flux (related to D_i in equation (1)) is approximated by $D_i = v_i c_i$ where v_i is the deposition velocity of the i-th species. To compute deposition velocities, several physical parameterizations have been developed (e.g. Wesely, 1989; Zhang et al., 2003). These parameterizations rely on meteorological data and parameters related to the land cover and the chemical species. Among the existing parameterizations and data sets (e.g. land use data), one has to determine the combinations that lead to the best results or that are the most reliable ones according to physics.

Beyond the deposition velocities, the same issues are encountered for several parameterizations. A major issue to build an efficient modeling system is thus the ability to deal with many parameterizations and to take into account various data sets.

In addition the data sets are usually large and distributed in any format, in any coordinate system, any unit, etc. Consequently, a comprehensive system should include good data-processing facilities.

As a conclusion, the parameterizations to compute the input fields, crucial to solve the chemistry-transport equation (1), require a strong flexibility and powerful tools to deal with the associated data.

2.4 Target applications

There are many different applications that may be supported by a chemistry-transport model. First the involved pollutants range from passive tracers to aerosols involved in complex chemical and physical processes. The most simple applications deal with passive tracers, namely pollutants that do not interact with other species. Radionuclides are an example, even if an extra term is added to equation (1) in order to take into account the radioactive decay. Other applications are related to heavy metals (such as mercury) and involve simplified chemical mechanisms. Complex mechanisms, e.g. photochemical mechanisms, simulate concentrations of pollutants such as ozone or nitrogen oxides. The most complex applications include aerosols that interact with the gaseous species and whose distribution and composition evolve due to physical processes (nucleation, evaporation/condensation, coagulation).

The applications vary from local scales to global scales. At local scales, the equation (1) is not relevant. However it is successfully applied at any higher scale, from regional to global scales.

The applications may differ in their goals. Simulations may be forecasts because they are run in an operational mode to estimate the concentrations for the next days. Other simulations try to assess the long-term impacts of changes in the input data (such as emissions). Several applications are comparisons between simulated concentrations and existing measurements; their aim is to get as close as possible to the measurements thanks to improvements in the input data. There are also academic simulations for process studies that only claim a rough modeling of the real concentrations.

Solving the equation (1) only returns a single time-evolution of the concentrations. New applications now provide a set of possible concentrations of the pollutants. In an operational context, this is called

ensemble forecast: the system generates several forecasts, which roughly returns a density probability function. One option to generate an ensemble forecast is to run several chemistry-transport models (Delle Monache and Stull, 2003), which supposes the use of a multi-models platform. Another option is to take advantage of a well design modeling system that would allow the use of multiple data sets, multiple parameterizations and multiple numerical schemes.

The last major application is data assimilation. Data assimilation is a way to take into account measurements to improve the quality of the computed concentrations (Elbern and Schmidt, 2001; Segers, 2002) or, if one performs inverse modeling, the quality of interesting input fields. A strong constraint is that data assimilation may require a tangent linear model or an adjoint model. A modern system should allow the use of both sequential (optimal interpolation, Kalman filters) and variational data assimilation.

2.5 Conclusion

From the previous subsections, the requirements of a full modeling system appear. First there are many different applications, notably depending on the pollutants and the chemical mechanism. Hence a flexible system is needed. Second a key point is the parameterizations used to generate the input fields and the large amount of involved data. An efficient strategy is required to handle the data and the parameterizations of the modeling system. The last issue is the distinct natures of the applications: academic simulations, forecasts, ensemble forecast, data assimilation. The ability to perform any of these activities is necessary in modern modeling systems, and it introduces strong constraints.

2.6 Existing models

It is hard to give an accurate picture of the numerous existing models among which one may find CMAQ (Byun and Ching, 1999), DEHM (Christensen, 1997), EMEP (Simpson et al., 2003), Eurad (Hass, 1991), Lotos (Builtjes, 1992), Polair3D (Boutahar et al., 2004), etc. – we apologize for the omitted models. A rather complete review may be found in Russell and Dennis (2000). Current models do not satisfy the requirements previously pointed out. A few lack powerful data processing facilities or provide a few simulation options (with respect to the input data or the parameterizations), they are therefore limited to a small set of applications. Others are not associated with complete data assimilation facilities for both sequential and variational methods. Ensemble forecasts are not always a feature of these systems.

The great number of models raises the question of the interoperability of these models and of the shared development. One would want to fairly compare these models and to generate ensemble forecasts, which would require the availability of a common platform. Moreover the same parameterizations and the same data processing facilities have been developed along with these models. A shared development would have saved time for research and should still be considered for the forthcoming applications.

Due to the historical conception of those systems, they are more or less all of a piece with a limited modularity and with components that may not be easily shared independently from the other components. The first steps in the achievement of a well designed modeling system is to identify the components of different nature and to propose relevant implementations for them.

3 Proposed architecture

3.1 Overall structure

A full modeling system relies on:

- its *databases*: the raw data used in the chemistry-transport equation, such as the wind V that is provided by a meteorological model, and the raw data included in the parameterizations, e.g. the land use coverage that is taken into account to compute the deposition velocities. Among the raw data, one may list: the meteorological fields, the emission databases, the land use coverage (and miscellaneous data associated with the land categories), the pollutant concentrations at higher scales (e.g. global concentrations that constitute the boundary conditions for continental simulations), the physical parameters associated with the chemical species.

- its *parameterizations*: the physical parameterizations are necessary to compute many fields, including important fields such as the vertical diffusion coefficients. Up-to-date parameterizations are required to perform the best simulations. They are strongly linked to the raw data since they use these data as input.

- its *chemistry-transport model(s)*: the chemistry-transport model is the last step in the simulation process since it solves the equation (1) and returns the pollutant concentrations. It is described in Section 3.5. Notice that several chemistry-transport models may be available within a single modeling system.

The proposed design is:

1. to build *databases*. The system should rely on databases available for all applications, that is to say not bound to one chemistry-transport model or to one of its applications.

2. to gather all data-processing facilities and all parameterizations in a *library* (or several libraries). The point is to extract from the chemistry-transport models, that are otherwise all of a piece, the facilities that may be used in other programs or by other teams. A library is perfectly suited to this respect as it is shown in Subsection 3.3.

3. to call functions of the libraries in *programs that generate the input data* (with the physical parameterizations and the raw data). The libraries may be called directly by the chemistry-transport model in order to compute the input fields (Subsection 2.3). Nevertheless we advocate to compute these fields in external programs that are executed before the chemistry-transport model, when possible. Details are given in Subsection 3.4.

4. to integrate in time the chemistry-transport equation, maybe along with additional calls to physical parameterizations (that must be called in the integration process). It constitutes the *chemistry-transport model* and it is actually limited to the numerical core of most chemistry-transport models.

Figure 1 shows the calling sequence. An important feature of this design is to separate the independent components in order to increase the flexibility of the system.

The next four subsections deal with the content of the four levels (databases, libraries, programs calling the libraries and the chemistry-transport model).

3.2 Databases

As previously mentioned, a large amount of data is needed to have the simulations running. Databases would ease the management of the data and their use. At least it is necessary to gather the data that come from different sources: meteorological models, transport models, databases for chemistry, etc.

Another useful action would be to define standard file formats. It is barely possible to restrict to a single file format because of the various nature of the involved data. Nonetheless stating that one should use a given format for meteorological fields and pollutant concentrations would be reasonable. For instance a format like NetCDF could be fine. A standard format for files containing measurements would also help. Another format may handle chemical data (mainly constants associated with species or chemical reactions).

However defining a standard data-format requires the approval of all actors (those who share the data and those who use them).

3.3 Libraries for atmospheric chemistry and physics

3.3.1 The need for libraries

In modern computer languages, there are libraries that implement multidimensional arrays, double-chained lists, etc. These libraries provide structures (called *objects* in object-oriented languages). Other libraries are a collection of *functions* to be called from programs; Blas (Lawson et al., 1979) and Lapack (Anderson et al., 1999) are the most famous examples for linear algebra. In the context of atmospheric chemistry and physics, we envision libraries that would provide both objects and functions.

Knowing that an extensive use of multidimensional data is done in atmospheric chemistry and physics, dedicated data structures are welcome. A library is a suited software framework to make data structures available to many programs.

The same is true for the physical parameterizations involved in the simulation process. Once implemented, a physical parameterization can be seen as a function to be called in the modeling system whenever necessary. Gathering the parameterizations in a library is a natural strategy. This is particularly true for the simple parameterizations that may be called many times, e.g. the conversion from specific humidity to relative humidity. A library avoids duplicates.

It has many other advantages. First it may be extended very easily: one just adds functions in it. Indeed all parameterizations implemented in the library are essentially independent from each other. Second the improvements in the library are immediately available to the programs that rely on it. For example, a bug fix in a given function is propagated in the programs, that use the function, as soon as the library is updated. Finally a library may be shared within a team or in the whole community. It may be modified and extended by many people without a strong coordination.

3.3.2 Content of the libraries

As previously explained and as shown in Figure 1 the libraries manage:

1. facilities for data processing thanks to powerful data structures and functions to help dealing with them. An object-oriented library is highly recommended because objects are suited for the implementation of advanced data structures. Among the requested features, there are the ability to deal with common file formats, the availability of interpolation functions and statistical functions, and the manipulation of gridded data.

2. the parameterizations and, generally, physical functions. These functions are supposed to manipulate the data structures previously mentioned. They range from simple physical functions to compute the potential temperature or the Richardson number, to complex parameterizations to evaluate surface fluxes, to diagnose the clouds, etc.

An example of a library for data processing in atmospheric chemistry and for parameterizations is AtmoData (Mallet and Sportisse, 2005a, in which one may also find additional arguments in favor of such a library). It is used by the modeling system Polyphemus, described in Section 4 where a few details about AtmoData are provided as well.

One may also find useful miscellaneous libraries to deal with chemical mechanisms, with dates, with configuration files, with graphical user interfaces, etc. These libraries may be called anywhere, not only in the input-data generation.

3.4 Input-data generation

The third level shown in Figure 1 is a set of programs to generate the input data demanded by the chemistry-transport model. They use the libraries previously described. For example, one of these programs could compute the vertical diffusion coefficients according to a parameterization wrote in a library. This program would take as input raw meteorological data (and maybe additional fields such as land data) from the databases, would make calls to the libraries to compute the vertical diffusion coefficients and would write the results in a file to be read by the chemistry-transport model.

One may wonder why these programs are not included in the chemistry-transport model. Notice that in Figure 1 the chemistry-transport model may also make calls to the libraries in order to compute a few fields (with the relevant parameterizations). The question of whether or not a given field should be computed inside the chemistry-transport model depends on:

- the number of available parameterizations to compute the field and the reliability of the parameterizations: if there are many parameterizations for the field (e.g. for the vertical diffusion coefficient: Louis, 1979; Troen and Mahrt, 1986), it should not be computed inside the chemistry-transport model. Indeed one should avoid the constraints of a full chemistry-transport model to work on the various parameterizations available for a given field. On the other hand, if a parameterization is reliable and is not going to change, it may be put directly in the integration process (i.e. in the chemistry-transport model).

- the computational time needed to compute the field: if the computational time is high, the parameterization should be called only once and therefore outside the chemistry-transport model. A given simulation is often executed several times (especially for ensemble forecast or data assimilation) and saving the computation of several fields is of high interest.
- the size of the data: if the data (output of a parameterization) cannot be stored on disk because of their huge size (e.g. the scavenging coefficients that depend on the time, the 3D position and the species), they must be computed inside the chemistry-transport model.

In practice, it is a lot more convenient to work on external programs to improve the parameterizations or their use. It is also safer to generate the data step by step. Most fields should be computed outside the chemistry-transport model which should mainly integrate the equation (1) in time and save the pollutant concentrations.

3.5 The chemistry-transport model

As previously explained, there are many different applications for a full modeling system. In particular, the equation (1) introduces a chemical mechanism through $\chi(c, t)$. Since one chemical mechanism may strongly differ from another, the chemistry-transport model, responsible for the time integration of equation (1), should be quite flexible. It means that the main part of the chemistry-transport model should be independent from the chemical mechanism. The chemistry is therefore a module that may be changed, independently from the other terms to be integrated (such as the transport). It can be achieved in many ways and it depends on how the whole code is built (computer language, architecture, numerical schemes that may or may not couple the chemistry with other processes, etc.). An example of such a flexible chemistry-transport model is Polair3D (see Boutahar et al., 2004) which is used by Polyphemus (described later in Section 4).

An additional constraint may be the automatic differentiation. It is a technique to differentiate a code automatically. It usually enables the automatic generation of a tangent linear model and an adjoint model. They are necessary to perform data assimilation with methods such as the Kalman filter or the four dimensional assimilation. Notice that automatic differentiation adds several strong constraints because most computer languages cannot be automatically differentiated and because the structure of a code to be automatically differentiated has to be carefully devised. For instance, the parts to be differentiated should not include input/output operations on files.

Furthermore, even without the constraints for automatic differentiation, it is advocated to split the numerical schemes from the input/output operations. The chemistry-transport model should actually be split in two parts: (1) a driver and (2) the initializations and the numerical schemes. The numerical schemes are implemented in a set of functions that perform the time integration over one time step. The driver first calls the functions that initialize the simulation and then it performs the time loop, calling in this loop the numerical schemes. Moreover the driver is responsible for the input/output operations: it reads on files the data generated by the programs (previously described) and saves the requested pollutant concentrations. It also manages the options of the simulation since it has to call the relevant functions in the time loop. Furthermore the driver could make calls to functions of the libraries (Section 3.3) in order to compute online the fields that could not be computed outside the chemistry-transport model (probably because of the size of these fields). Figure 2 shows the proposed structure with the driver.

A promising perspective is the inclusion of several chemistry-transport models in the same framework. The final step would be the design of a generic driver that could handle virtually any chemistry-transport model. All chemistry-transport models have a close structure which includes an initialization process and a time loop in which there are calls to physical parameterizations and numerical schemes. The proposed software-strategy is the use of an object-oriented language and the definition of an interface to define the standard communication with the chemistry-transport model. In an object-oriented language, an interface may define the content of a typical object. Recall that, roughly speaking, an object is a structure that contains variables, called attributes, and functions, called methods, to work on the attributes. An object is used through calls to its methods. For instance, an object `Vector` may contain a data array and its length (attributes), and two functions `GetLength` and `Resize` (methods) to interact with the vector. In our context, the main object would manage the chemistry-transport model and it would be used by the driver. The interface would define the methods that should be

available in the object so that the driver could call the right methods. Assume that `Polair3D` is the object associated with the chemistry-transport model Polair3D (described later in Section 4.3). The interface would impose that, for the advection problem, the following methods are available: `Init`, `Advection`. `Init` initializes Polair3D and `Advection` integrates the concentrations over one time step. The driver would simply call `Polair3D.Init` (i.e. the method `Init` of `Polair3D`) and would perform a time loop in which it would call `Polair3D.Advection`. If another chemistry-transport model has such an interface, the same driver may perform the same simulation, which obviously provides a convenient platform. We believe that a dedicated interface may be written for virtually any chemistry-transport model without deep changes in it.

The interface would also include methods in order that the driver could manage the state of the chemistry-transport model. This way, the driver could perform the output operations. Moreover several drivers could be built. Obviously there would be a driver to perform forward simulations, but there would be a driver for ensemble forecasts, for the Kalman filter, the four dimensional assimilation, etc. For instance, the driver responsible for a four dimensional assimilation would perform a forward simulation, and then a backward simulation with an appropriate loop in time and calls to the adjoint code. An issue to investigate is whether a driver could also allow to plug a meteorological model. At least a driver is the right entry point for extensions such as a radiative transfer model that would compute photo-dissociation rates.

Any chemistry-transport model, for which an interface would be available, may be used as the underlying model of one of the previous drivers. Hence any model could benefit from the whole framework. This way, several teams may efficiently share their developments. There are also advantages with a single model since the overall structure is easier to manage, especially in high-level applications such as ensemble forecast and data assimilation.

4 A practical implementation: Polyphemus

Polyphemus[4] is an example of a system close to what is proposed in this paper. The missing feature is the driver: the chemistry-transport model used by Polyphemus, Polair3D, is not designed with a complete driver as described in the previous section. Polyphemus thus follows the structure shown in Figure 1.

4.1 The library AtmoData

The library AtmoData includes facilities for data processing and the parameterizations needed to perform reliable simulations. The library takes advantage of the abilities of the object-oriented language C++ to provide well designed data structures dedicated to atmospheric chemistry and physics. These data structures are multidimensional arrays stored together with their coordinates (in other words, gridded data) and they may store the meteorological fields, the emissions inventories, etc. They are associated with convenient functions (mainly methods, in the C++ terminology) to compute statistics, to switch dimensions, to perform linear interpolations, etc. The input/output operations in many formats (text files, binary files, NetCDF files, Grib files, etc.) are part of the library. The framework provided by AtmoData for data processing is greatly appreciable for the safety of the programs and to speed-up the development.

The library gathers many parameterizations: for deposition velocities, for emissions, for clouds, etc. The physical parameterizations are implemented in functions whose arguments are data structures as defined in AtmoData. All functions are independent from the type of the input variables (single precision, double precision, ...). AtmoData is therefore easy to use and flexible.

A detailed description of AtmoData is available in Mallet and Sportisse (2005a).

4.2 The input-data generation

Several programs are available to generate complete sets of data for comprehensive simulations. Of course they rely on AtmoData. In the version 0.2 of Polyphemus (lastest stable version), the available programs generate: boundary conditions (based on output fields from Mozart 2 – Horowitz et al.,

[4]Polyphemus is available under the GNU General Public License – i.e. Polyphemus is free software – at `http://www.enpc.fr/cerea/polyphemus/`.

2003), initial conditions (from Mozart 2 output as well), emissions (from EMEP[5] data and biogenic emissions – Middleton et al. (1990); Simpson et al. (1999)), transformation of meteorological fields (from ECMWF[6] and MM5[7]), cloud diagnoses and cloud attenuation, vertical diffusion coefficients (Troen and Mahrt, 1986; Louis, 1979), deposition velocities (Zhang et al., 2003; Wesely, 1989).

The programs are driven by flexible configuration files, which means that no changes are necessary in the code to have the system working for most applications. They are therefore easy to use even in operational mode.

4.3 The chemistry-transport model

The chemistry-transport model is Polair3D (Boutahar et al., 2004). It is written in Fortran 77 and with a careful design so that it can be automatically differentiated (Mallet and Sportisse, 2004). It is flexible enough to handle all the target applications previously mentioned: gaseous simulations (e.g. ozone forecasts), simulations with aerosols, data assimilation and ensemble forecast (with external programs to be officially included in Polyphemus in its next versions).

A complete description of Polair3D is not the purpose of this paper and may be found in Boutahar et al. (2004). Polair3D meets all the recommendations explained in Section 3.5 but the use of a driver. Such a driver has been roughly written to perform data assimilation experiments, but there is still some work to this respect.

4.4 Current applications

Polyphemus 0.2 could be used for most air-quality applications involving gaseous species. If the meteorological fields come from the ECMWF or are output of MM5, essentially no changes in the programs are necessary to have a simulation running: the programs read configuration files in which all options are available. One simply specifies the domains involved, the simulation period, the meteorological time steps, the parameterizations to be used and a few other options.

Polyphemus is also ready for operational use: it currently runs on a daily basis to perform ozone forecasts. A key strength of the system is the ability to perform ensemble forecast thanks to the multiple choices available in the configuration files. One simply associates a set of configuration files to each forecast in the ensemble. An application based on this feature may be found in Mallet and Sportisse (2005b).

4.5 Next steps

Polyphemus is already a satisfactory modeling system but it still needs improvements. It implements the structure shown in Figure 1: the databases, the libraries (actually one library: AtmoData), the programs for the input-data generation and the chemistry-transport model (Polair3D) are four distinct parts as advocated in this paper. As far as the whole structure is concerned, the next development could focus on a driver (see Section 3.5). The final aim is to include several chemistry-transport models in the framework of Figure 2 and Polyphemus is completely open to any team that would want to take part in the project.

As for AtmoData, the next steps are mainly the addition of parameterizations. Its design should not change.

The programs used to generate the input data are satisfactory as well. They will be updated according to the new data and parameterizations. New programs will be added so that the system could handle more cases (e.g. new meteorological models or new emission inventories) and the simulations with aerosols.

A work in progress is the operational use of Polyphemus for forecast of radionuclides dispersion at continental scale, in the Emergency Response Organization Department of IRSN.

Additional improvements, that are currently planned (and under development), deal with the output-data processing: visualization, comparison with measurements and support for ensemble forecasting. They are included in a library called AtmoPy. They are mainly written in Python and use the

[5] Co-operative Programme for Monitoring and Evaluation of the Long-range Transmission of Air pollutants in Europe – `http://www.emep.int/`.

[6] European Centre for Medium-Range Weather Forecasts – `http://www.ecmwf.int/`.

[7] PSU/NCAR mesoscale model – `http://www.mmm.ucar.edu/mm5/`.

free libraries numarray (Greenfield et al., 2003) and Matplotlib (`http://matplotlib.sourceforge.net/`) that clone and sometimes extend Matlab®.

The last improvement may be the inclusion of the material already developed, above the system (usually on top of the chemistry-transport model Polair3D), to perform (1) data assimilation and (2) ensemble forecast. Alternatively both could be implemented again through a dedicated driver (as proposed in Section 3.5).

5 Conclusion

This paper proposes a structure for a full modeling system in air quality. The division into four parts (databases, libraries, input-data generation and the chemistry-transport model itself) enables to build a flexible system that may handle all useful applications from the transport of passive tracers to data assimilation with complex chemical mechanisms. The use of libraries and the ability to plug several chemistry-transport models in the system make it open to the community. Hence this system has to be viewed as a proposition to the air-quality community to share the developments that would otherwise be duplicated and to constitute a comprehensive platform. The structure has proven to be a fruitful strategy through the development of Polyphemus.

Acknowledgement: The first author is partially supported by the Île-de-France region.

References

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D., 1999. LAPACK users' guide, 3rd Edition. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Boutahar, J., Lacour, S., Mallet, V., Quélo, D., Roustan, Y., Sportisse, B., 2004. Development and validation of a fully modular platform for numerical modelling of air pollution: POLAIR. Int. J. Environment and Pollution 22 (1/2), 17–28.

Builtjes, P., 1992. The LOTOS – LOng Term Ozone Simulation – project, summary report. Tech. Rep. R92/240, TNO, Delft, the Netherlands.

Byun, D. W., Ching, J. K. S. (Eds.), 1999. Science algorithms of the EPA models-3 community multiscale air quality (CMAQ) modeling system. EPA.

Christensen, J. H., 1997. The Danish Eulerian hemispheric model – a three-dimensional air pollution model used for the arctic. Atmos. Env. 31, 4,169–4,191.

Delle Monache, L., Stull, R. B., 2003. An ensemble air-quality forecast over western Europe during an ozone episode. Atmos. Env. 37, 3,469–3,474.

Elbern, H., Schmidt, H., 2001. Ozone episode analysis by four-dimensional variational chemistry data assimilation. J. Geophys. Res. 106 (D4), 3,569–3,590.

Greenfield, P., Miller, J. T., Hsu, J.-C., White, R. L., 2003. numarray: a new scientific array package for Python. In: PyCon DC 2003.

Hass, H., 1991. Description of the EURAD chemistry-transport-model version 2 (CTM2). Tech. Rep. 83, Institute of Geophysics and Meteorology, University of Cologne.

Horowitz, L. W., Walters, S., Mauzerall, D. L., Emmons, L. K., Rasch, P. J., Granier, C., Tie, X., Lamarque, J.-F., Schultz, M. G., Tyndall, G. S., Orlando, J. J., Brasseur, G. P., 2003. A global simulation of tropospheric ozone and related tracers: description and evaluation of MOZART, version 2. J. Geophys. Res. 108 (D24).

Lawson, C. L., Hanson, R. J., Kincaid, D. R., Krogh, F. T., 1979. Basic Linear Algebra Subprograms for Fortran usage. ACM Trans. on Math. Soft. 5 (3), 308–323.

Louis, J.-F., 1979. A parametric model of vertical eddy fluxes in the atmosphere. Boundary-Layer Meteor. 17, 187–202.

Mallet, V., Sportisse, B., 2004. 3-D chemistry-transport model Polair: numerical issues, validation and automatic-differentiation strategy. Atmos. Chem. Phys. Discuss. 4, 1,371–1,392.

Mallet, V., Sportisse, B., 2005a. Data processing and parameterizations in atmospheric chemistry and physics: the AtmoData library. Submitted to Environmental Modelling and Software.

Mallet, V., Sportisse, B., 2005b. Uncertainty in a chemistry-transport model due to physical parameterizations and numerical approximations: an ensemble approach applied to ozone modeling. Accepted for publication in J. Geophys. Res.

Middleton, P., Stockwell, W. R., Carter, W. P. L., 1990. Aggregation and analysis of volatile organic compound emissions for regional modeling. Atmos. Env. 24A (5), 1,107–1,133.

Russell, A., Dennis, R., 2000. NARSTO critical review of photochemical models and modeling. Atmos. Env. 34, 2,283–2,234.

Segers, A., 2002. Data assimilation in atmospheric chemistry models using kalman filtering. Ph.D. thesis, Delft University.

Simpson, D., Fagerli, H., Jonson, J. E., Tsyro, S., Wind, P., Tuovinen, J.-P., 2003. Transboundary acidification, eutrophication and ground level ozone in Europe – part I: unified EMEP model description. Tech. rep., EMEP.

Simpson, D., Winiwarter, W., Börjesson, G., Cinderby, S., Ferreiro, A., Guenther, A., Hewitt, C. N., Janson, R., Khalil, M. A. K., Owen, S., Pierce, T. E., Puxbaum, H., Shearer, M., Skiba, U., Steinbrecher, R., Tarrasón, L., Öquist, M. G., 1999. Inventorying emissions from nature in Europe. J. Geophys. Res. 104 (D7), 8,113–8,152.

Troen, I., Mahrt, L., 1986. A simple model of the atmospheric boundary layer; sensitivity to surface evaporation. Boundary-Layer Meteor. 37, 129–148.

Verwer, J. G., Hundsdorfer, W., Blom, J. G., 1998. Numerical time integration for air pollution models. Tech. rep., CWI.

Wesely, M. L., 1989. Parameterization of surface resistances to gaseous dry deposition in regional-scale numerical models. Atmos. Env. 23, 1,293–1,304.

Zhang, L., Brook, J. R., Vet, R., 2003. A revised parameterization for gaseous dry deposition in air-quality models. Atmos. Chem. Phys. 3, 2,067–2,082.

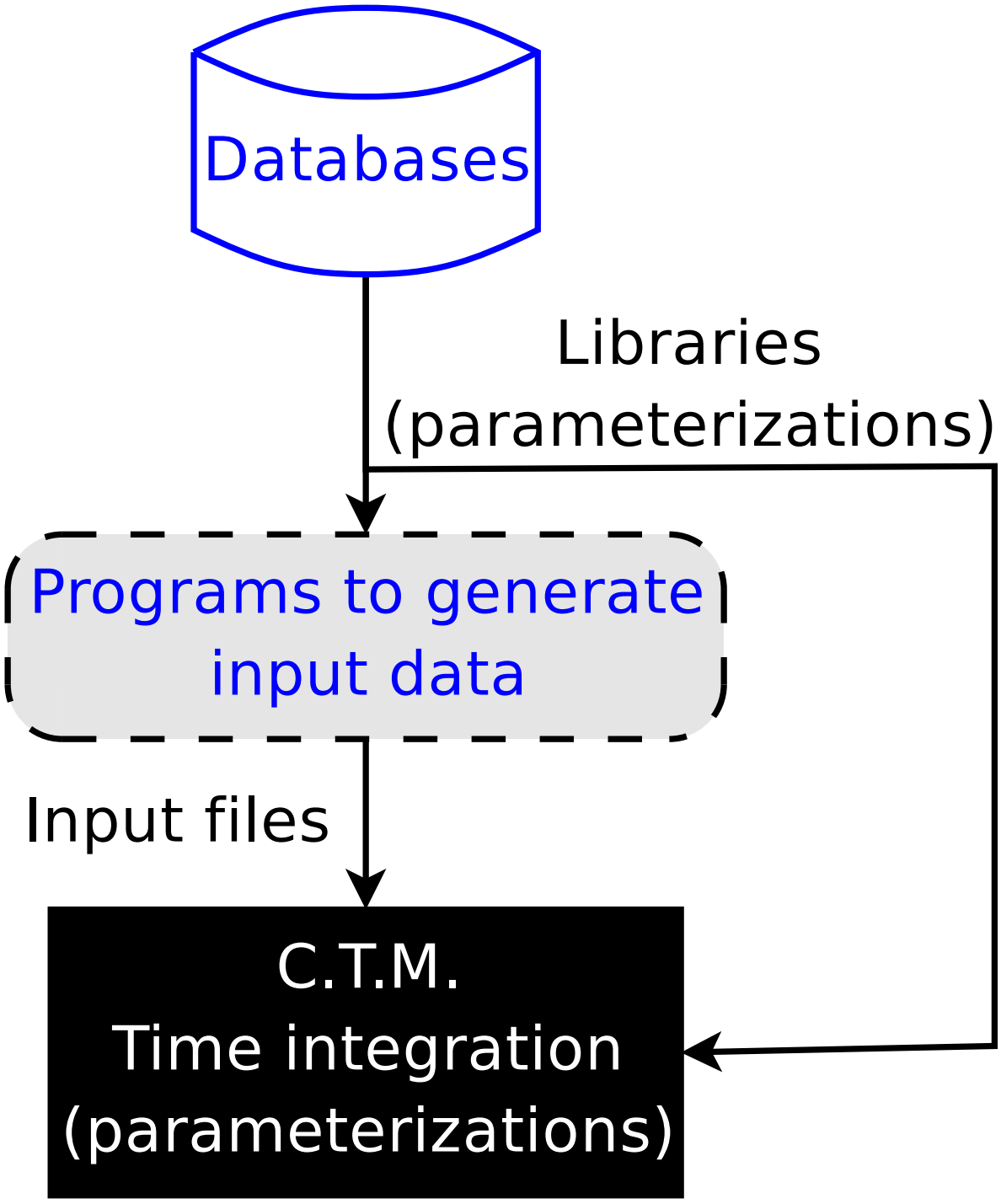

Figure 1: Overall structure of a full modeling system. It includes (1) databases, (2) libraries for data processing and physical parameterizations, and (3) programs to generate input data for (4) a chemistry-transport model (C.T.M.).

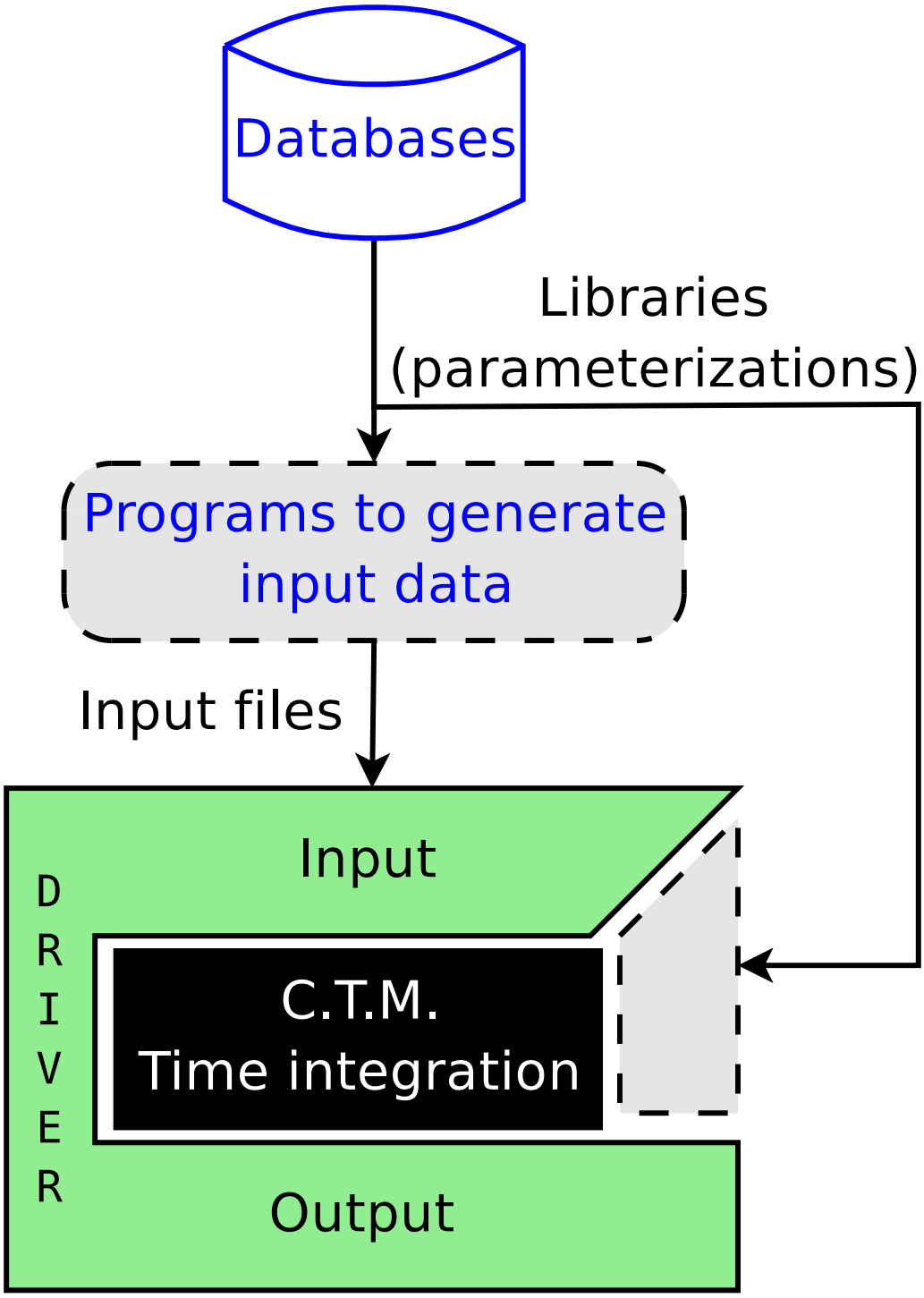

Figure 2: Proposed structure of a full modeling system. It includes (1) databases, (2) libraries for data processing and for physical parameterizations, and (3) programs to generate input data for (4) a chemistry-transport model (C.T.M.) split into a driver and its numerical schemes. The driver is responsible for input/output operations and of the time loop in which it makes the relevant calls for the time integration. The driver may also compute a few fields through the parameterizations available in the libraries.