Polair 1.3
3D chemistry-transport model

User's guide, version 1

CEREA – ENPC
Jaouad Boutahar, Vivien Mallet, Denis Quélo,
Yelva Roustan and Bruno Sportisse

Contents

Chapter 1

Introduction

This user's guide describes the 3D Eulerian Chemistry-Transport-Model POLAIR, developed by CEREA (Research Center for Atmospheric Environment) at Ecole Nationale des Ponts et Chaussées.

The purpose of POLAIR is to provide a numerical platform for atmospheric dispersion studies. POLAIR is supposed to ensure:

- modularity:

  Several chemical mechanisms may be used on the basis of a uniform standard for the chemical preprocessor used by POLAIR, namely SPACK (Simplified Preprocessor for Atmospheric Chemical Kinetics, [1]).

  Several different modules are or will be available for the chemical production term, depending on the application. They include gas-phase chemistry for tropospheric ozone, multiphase chemistry (aqueous-phase chemistry inside clouds and aerosols), mechanisms for pesticides, heavy metals, mercury, radionuclides, aso.

- multiple functions:

  POLAIR provides the time evolution of a spatial distribution for a set of given chemical species (this is the so-called *direct model*). The adjoint and linear tangent versions of POLAIR may be automatically computed by use of ODYSSEE (an automatic differenciation tool developed at INRIA, [2]): the adjoint version is necessary for variational methods in data assimilation, the linear tangent version for sensitivity analysis.

- multiple scales:

  POLAIR may be used at the regional/urban scale (typically 100 kilometers$\times$100 kilometers) or/and at the regional/continental scale (typically over Europe). One-way nesting is under development.

This report is organized as follows:

- the background for CTM (modeling and numerics) is described in Chapter 2,
- the software of architecture of POLAIR is described in Chapter 3,
- the main recommendations for using POLAIR is described in Chapter 4,
- some additional details are given in Chapter 5 (including multiphase modeling and data assimilation),
- some applications can be found in Chapter 6,

- planned extensions of POLAIR are described in Chapter 7.

Chapter 2

3D chemistry-transport model

Comprehensive 3D CTM are now available. We refer to [3, 4] for a description of the underlying models and to [5] for some numerical issues.

We describe in this chapter the underlying model formulation and numerical algorithms used for POLAIR. The key modeling (resp. numerical) aspects are investigated in Section 1 (resp. Section 2).

2.1 Model

2.1.1 Dispersion equation

We describe the time and space evolution of some trace gases, let say X_i (i is the index labeling chemical species, X is the symbol of chemical species). We have then for the concentration c_i of species X_i:

$$\frac{\partial c_i}{\partial t} = L_{adv}(c_i) + L_{diff}(c_i) + [L_{conv}(c)]_i + [L_{chem}(c)]_i + S_i \tag{2.1}$$

where the take into account the following processes:

- Wind advection: $L_{adv}(c_i) = -div(Vc_i)$,
- Turbulent diffusion: $L_{diff}(c_i) = div(K\nabla c_i)$,
- Convection: $L_{conv}(c)$,
- Chemical production: $L_{chem}(c) = \chi(c, T, L, I)$,
- Volume source emissions: $S_i(x, t)$ (in practice in the first vertical levels).

We will distinguish diffusion (resp. advection) along x, y and z directions:

$$L_{diff} = L_{diffx} + L_{diffy} + L_{diffz} \tag{2.2}$$

where for instance (with obvious notations):

$$L_{diffx}(c) = div(K_x \nabla c_i) \ , \quad L_{advx} = -div(V_x c_i) \tag{2.3}$$

We have used moreover the following notations:

- c_i is the concentration for species X_i in molecules per volume (in practice we will use a mixing ratio $q_i = \frac{c_i}{\rho}$ with ρ the air density),

- $V(x,t)$ is the wind field,
- $K(x,t)$ is the eddy diffusivity matrix (computed with the parameterization of [6]),
- $L_{conv}(c)$ is the parameterization of convective processes ([7]),
- $\chi_i(c,T,L,I)$ is the chemical production for species i. $T(x,t)$ is the temperature field, L is the liquid water content, I is the actinic flux whose knowledge is necessary for computing photolytic rates.

t (resp. x) is the time (resp. spatial) coordinate.

All the meteorological fields are supposed to be known (off-line coupling).

2.1.2 Boundary conditions

For diffusion

At the top of our domain, we have a no-flux boundary condition (free atmosphere):

$$K\frac{\partial c_i}{\partial n} = K\nabla c_i \cdot n = 0 \tag{2.4}$$

where n is the normal vector to the domain (inward oriented).

At ground, we have:

$$-K\frac{\partial c_i}{\partial n} = K\nabla c_i \cdot n = v_i^{dep}(x,t)\, c_i - E_i(x,t) \tag{2.5}$$

where v_i^{dep} is the dry deposition velocity and $E_i(x,t)$ is the emission factor for species i. We refer to [8] for the computation of dry deposition velocities. Emission factors E_i are supposed to be known.

At lateral boundaries, we have:

$$K\nabla c_i \cdot n = 0 \tag{2.6}$$

For advection

For boundaries along which $n \cdot V > 0$ (n is inward oriented), we use:

$$n \cdot Vc = Flux \tag{2.7}$$

where $Flux$ is given (by a global model or by another way such as inverse modeling).

2.1.3 Inputs/Outputs

Inputs

Solving Equation (2.1) requires several inputs.

The *wind speed* V is given by a meteorological solver. For instance, data from ECMWF[†1] are used. From meteorological forecasts, *humidity*, *temperature*, *nebulosity* and *precipitations* are extracted as well.

Initial concentrations are needed too. They may be inferred from observations, or they may be computed by a CTM. Notice that initial conditions are not a key point for a CTM that has run for a while (spin-up).

[†1] European centre for medium-range weather forecasts.

Boundary concentrations are required due to the transport process. A model running at a higher scale may provide those boundary conditions.

Emissions of species appear in Equation (2.1), though E. Emission inventories, notably based on traffic and industrial emissions, are provided to the CTM.

Finally, one should add data (such as *dry-deposition velocities*) related to different processes, depending on the CTM complexity.

Outputs

Outputs are concentrations of species, at given positions and for given dates.

Of course, if the CTM deals with more complex processes, other outputs may be involved, e.g. aerosol distributions.

2.1.4 Chemical kinetics

Kinetic scheme

We suppose that the kinetic scheme is described by a set of n_r reactions among n_s species. For a species X_i, the chemical source term is:

$$\chi_i(c,T,L,I) = \sum_{j=1}^{j=n_r} S_{ij}\omega_j(c,T,L,I) \tag{2.8}$$

with S the stoichiometric matrix ($n_s \times n_r$) and ω_j the reaction rate for reaction j.

A typical example is given by the tropospheric so-called Chapman's cycle (with M the third body):

$$\mathrm{O_2 + O + M \xrightarrow{1} O_3 + M}$$
$$\mathrm{NO_2} + h\nu \xrightarrow{2} \mathrm{NO + O}$$
$$\mathrm{NO + O_3 + M \xrightarrow{3} NO_2 + O_2}$$

The chemical production term for ozone is then $\chi_{O_3} = \omega_1 - \omega_3$.

Kinetic and reaction rates

Due to Mass Action Law, we have for the reaction rate of reaction j:

$$\omega_j = k_j \Pi_{i=1}^{i=n_s} c_i^{S_{ij}} \tag{2.9}$$

where k_j is the kinetic rate to be computed according to the reaction type:

- For thermal reactions, the Arrhenius' law is used:

$$k_j(T) = A_j T^{B_j} exp(-C_j/T) \tag{2.10}$$

  A_j, B_j and C_j are given for each reaction.

- For fall-off (or TROE) reactions:

$$k_j(p,T) = \frac{a_0(T)M}{(1+a_0(T)M)a_1(T)} 0.6^{(1+(log_{10}(a_0(T)a_1(T)M))^2)^{-1}} \tag{2.11}$$

  with $a_n(T) = a_n^{(300)}(T/300)^{-b_n}$ for $n = 0$ and $n = 1$. The parameters a_0^{300}, a_1^{300}, b_0 and b_1 have then to be defined for each reaction. M is the concentration of the third body (function of p and T).

- For photolytic reactions $X_i + h\nu(\lambda_j^{min} \leq \lambda \leq \lambda_j^{max}) \rightarrow \ldots$:

$$\omega_j(c, I) = J_j(I)c_i \tag{2.12}$$

where the photolytic rate J_j is computed through:

$$J_j(I) = \int_{\lambda_j^{min}}^{\lambda_j^{max}} q_\lambda(j) I(\lambda) \sigma_\lambda^a(i) d\lambda \tag{2.13}$$

where $q_\lambda(j)$ is the quantum yield for reaction j and $\sigma_\lambda^a(i)$ is the absorption cross section for species i (defined as the only reactant for reaction j). Photolytic rates used to be computed by Fast-J ([9]). They are now pre-computed by JPROC which provides clear sky photolysis rates (see chapter 14 of CMAQ documentation at `http://www.epa.gov/asmdnerl/models3/doc/science/science.html`). A cloud attenuation factor is applied to correct clear sky rates.

Preprocessing

We use the preprocessor SPACK (see [10] and appendix) for generating in an automatic way the Fortran code giving $L_{chem}(c)$ and the associated Jacobian matrix $A_{chem} = \frac{\partial L_{chem}}{\partial c}$.

Extension to a multiphase model

The next step (POLAIR 2.0) is to extent the previous scheme to multiphase models (aerosols, see Chapter 5).

2.1.5 Wet scavenging

Wet Scavenging by rain is parameterized by:

$$L_{wet}(c_i) = -\Lambda_i(t)c_i \tag{2.14}$$

where Λ_i is a parameterization of wet scavenging and is computed through meteorological data (see [11] for instance):

$$\Lambda_i = \frac{6.10^{-6}\, p_0\, K_i}{3.6\, U_{rain}\, D} exp\left(-\frac{6\,(h-z)\,K_i}{D\, H_i RT\, U_{rain}}\right) \tag{2.15}$$

with p_0 the rain intensity (in mm.hr^{-1}), z the height (in m), h the height of cloud basis (in m), H_i the Henry's coefficient for species i (in mol.l^{-1}.atm^{-1}), R the perfect gas constant (in atm.l.mol^{-1}.K^{-1}), U_{rain} the rain velocity (in m.s^{-1}), D the raindrop diameter (in m) and K_i the mass transfer coefficient for a falling drop (in m.s^{-1}).

Usually, the following parameterizations are used:

$$U_{rain} = 9.58\left(1 - e^{-\left[\frac{D}{0.171}\right]^{1.147}}\right) \tag{2.16}$$

Remark: in equation (2.16) D should be given in cm.

$$K_i = \frac{Dg_i}{D}\, Sh_i \tag{2.17}$$

with Dg_i the gas-phase molecular diffusion for species i (in m^2.s^{-1}) and Sh the Sherwood number:

$$Sh_i = 2 + 0.6\left(\frac{DU_{rain}}{\nu_{air}}\right)^{1/2}\left(\frac{\nu_{air}}{Dg_i}\right)^{1/3} \tag{2.18}$$

with ν_{air} the kinematic viscosity (in m^2.s^{-1}).

2.1.6 Parameterization for K_z

We use the classical Louis parameterization for vertical diffusion due to turbulence ([6]):

$$K_z = l(z)^2 || \frac{\partial V}{\partial z} || F(R_i, z) \tag{2.19}$$

where:

- $l(z)$ is a mixing length:

$$l(z) = K_a \frac{z + z_0}{1 + K_a \frac{z+z_0}{L}} \tag{2.20}$$

  where $K_a \simeq 0.4$ is the Von Karman constant and z_0 and L are scale factors ($z_0 \simeq 1$ m and $L \sim 100$ m).

- R_i is the Richardson number:

$$R_i = g \frac{\partial ln\theta}{\partial z} \frac{1}{||\frac{\partial V}{\partial z}||^2} \tag{2.21}$$

  where θ is the potential temperature and g the gravity constant.

- F is the following function:

$$F(R_i, z) = \frac{1}{1 + 3B\, R_i \sqrt{1 + D\, R_i}} \quad \text{if} \quad R_i \geq 0 \tag{2.22}$$

$$F(R_i, z) = 1 - \frac{3B\, R_i}{1 + 3B\, C \sqrt{\frac{|R_i|}{27}} \left(\frac{l(z)}{z+z_0}\right)^2} \quad \text{if} \quad R_i \leq 0 \tag{2.23}$$

  where usually $B = C = D = 5$.

For horizontal diffusion a constant value is usually advocated. Its magnitude depends on the scale. For urban scale (typically 100km×100 km), we advocate:

$$K_x = K_y \simeq 500 \tag{2.24}$$

while we advocate for continental scale:

$$K_x = K_y \simeq 10^5 \tag{2.25}$$

Parameterizing diffusion makes then necessary the choice of z_0, L and $K_x = K_y$ which remains more or less a kind of tuning.

2.1.7 Convective parameterization

Parameterizing convective processes is necessary for large-scale applications in order to take into account subgridscale transport induced by cumulus clouds. The sugridscale fluxes are defined in the following way:

$$L_{conv}(c) = -\frac{\partial}{\partial z}(F_u(c) + F_d(c) + F_s(c)) \tag{2.26}$$

where F_u (resp. F_d) stands for the updraft (resp. downdraft) flux and F_s is the subsidence flux. These fluxes are computed on the basis of entrainment and detrainment rates into the updraft and into the downdraft (see [7, 12]). After discretization, the scheme proposed by Tiedke ([7]) is the following one:

$$L_{conv}(c) = M_{conv} c \tag{2.27}$$

where M is defined by the cloud model.

2.1.8 Space coordinates

We distinguish different applications:

- for a regional application (photochemical smog), we use cartesian coordinates (x, y, z).
- for a continental application (dispersion of pollutants over Europe), we use spherical and hybrid coordinates as for ECMWF model: (λ, ϕ, η).

  η is defined in the following way:

  $$p(\lambda, \phi, \eta, t) = A(\eta) + B(\eta)p_g(\lambda, \phi, t) \tag{2.28}$$

  where p (resp. p_g) is the pressure (resp. at ground level). A and B are given functions: if $A = 0$ we have pure σ-coordinates. We refer to [13] for the definition of A and B.

  After some tedious calculations the dispersion equation (2.1) reads now:

  $$\frac{\partial c}{\partial t} + \frac{1}{r\cos\phi}(\frac{\partial uc}{\partial \lambda} + \frac{\partial vc\cos\phi}{\partial \phi}) + \frac{1}{h_\eta r^2}\frac{\partial wcr^2}{\partial \eta} = \frac{1}{h_\eta}\frac{\partial}{\partial \eta}(\frac{d_\eta}{h_\eta}\frac{\partial c}{\partial \eta}) \tag{2.29}$$

  where h_η is a scale factor. In the above equation, u, v and w stand for the velocity components in the λ, ϕ and η components while $r \simeq 6378$ km is the distance from earth center (supposed to be constant in practice).

2.2 Numerics

2.2.1 Splitting methods

We use different splitting methods in order to solve (2.1). In the following, the volume source term is supposed to be coupled with chemistry.

S_1: Strang Splitting

The first approach is the so-called Strang splitting with the following sequence (Δt is the splitting timestep):

- Integrate L_{adv} in the sequence $(L_{advx}, L_{advy}, L_{advz})$ on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{chem} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{diffx} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{diffy} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{diffz} on $[t, t + \Delta t]$,
- Integrate L_{diffy} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,
- Integrate L_{diffx} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,
- Integrate L_{chem} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,
- Integrate L_{adv} in the sequence $(L_{advz}, L_{advy}, L_{advx})$ on $[t + \frac{\Delta t}{2}, t + \Delta t]$,

After each sequence, initial conditions are changed.

S_2: Coupling of L_{chem} and L_{diffz}

The second approach is a Strang Splitting with a kind of coupling between chemical production and vertical diffusion since splitting these processes is the main source of errors ([14, 15]):

- Integrate L_{adv} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{diffx} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate L_{diffy} on $[t, t + \frac{\Delta t}{2}]$,
- Integrate $L_{diffz} + L_{chem}$ on $[t, t + \Delta t]$,
- Integrate L_{diffy} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,
- Integrate L_{diffx} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,
- Integrate L_{adv} on $[t + \frac{\Delta t}{2}, t + \Delta t]$,

The way we integrate $L_{diffz} + L_{chem}$ is precised below.

2.2.2 Integration of L_{chem} and L_{diffz}

General formulation

We use a second-order Rosenbrock method (ROS2) for the integration of an Ordinary Differential Equation:

$$\frac{dc}{dt} = f(t, c) \tag{2.30}$$

The scheme reads as:

$$c^{n+1} = c^n + (3k_1 + k_2)\Delta t/2 \tag{2.31}$$

where

$$(1 - \gamma \Delta t\, A)k_1 = f(t_n, c_n)\ , \quad (1 - \gamma \Delta t\, A)k_2 = f(t_{n+1}, c_n + \Delta t\, k_1) - 2k_1 \tag{2.32}$$

$\gamma = 1 + 1/\sqrt{2}$ for stability requirements and A denotes an approximation of the Jacobian matrix $A \simeq \partial f/\partial c$.

In (2.32) $\gamma_i\, \Delta t^2 \partial f/\partial t$ is an nonautonomous term that may be added ([1]). This term is zero in the case of nonphotolytic activity and constant water content in the considered integration period. We refer to [16] for more details concerning the use of Rosenbrock methods for atmospheric chemistry.

Coupling between chemistry and diffusion

In the case of coupling integration between chemistry and diffusion we use a so-called "internal splitting" (or Approximate Matrix Factorization). In this case:

$$f(c) = L_{diff}(c) + L_{chem}(c)\ , \quad A = A_{diff} + A_{chem} \tag{2.33}$$

We approximate:

$$(I - \gamma_i \tau A) \sim (I - \gamma_i \tau A_{chem})(I - \gamma_i \tau A_{diff}) \tag{2.34}$$

which avoids the inversion of systems whose dimension would be the the product of the number of grid cells by the number of species.

2.2.3 Advection scheme

For a monodimensional model we use the third-order Direct Space Time (DST) scheme with the Koren-Sweby flux limiter function as advocated in [13]:

$$c_i^{n+1} = c_i^n + (F_{i-\frac{1}{2}} - F_{i+\frac{1}{2}}) \tag{2.35}$$

where:

- for $u_{i+\frac{1}{2}} \geq 0$:

$$F_{i+\frac{1}{2}} = \nu_{i+\frac{1}{2}}(c_i + \psi(\nu_{i+\frac{1}{2}}, \theta_i)(c_{i+1} - c_i)) \tag{2.36}$$

- for $u_{i+\frac{1}{2}} < 0$:

$$F_{i+\frac{1}{2}} = -\nu_{i+\frac{1}{2}}(c_{i+1} + \psi(\nu_{i+\frac{1}{2}}, \frac{1}{\theta_{i+1}})(c_i - c_{i+1})) \tag{2.37}$$

with the Koren-Sweby flux limiter function:

$$\psi(\nu, \theta) = max(0, min(1, d_0(\nu) + d_1(\nu)\theta, \mu\theta) \tag{2.38}$$

where

$$d_0(\nu) = \frac{1}{6}(2-\nu)(1-\nu) \ , \quad d_1(\nu) = \frac{1}{6}(1-\nu^2) \ , \tag{2.39}$$

$$\nu_{i+\frac{1}{2}} = \frac{\Delta t}{\Delta x}\left|u_{i+\frac{1}{2}}\right| \tag{2.40}$$

and the slopes are:

$$\theta_i = \frac{c_i - c_{i-1}}{c_{i+1} - c_i} \tag{2.41}$$

Let us notice that there is still a free parameter (μ in Eq. (2.38)). As in [5] we take:

$$\mu = \frac{1-\nu}{\nu} \tag{2.42}$$

For this scheme, the Courant-Friedrichs-Lewy (CFL) condition is:

$$\nu_{i+\frac{1}{2}} \leq 1 \tag{2.43}$$

2.2.4 Diffusion

Diffusion is discretized in the classical way with a three point scheme. For instance for a given direction (let say x), at point i (namely x_i):

$$div(K_x \nabla c) \sim \frac{K_x(x_{i+\frac{1}{2}})\frac{c(x_{i+1})-c(x_i)}{x_{i+1}-x_i} - K_x(x_{i-\frac{1}{2}})\frac{c(x_i)-c(x_{i-1})}{x_i - x_{i-1}}}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \tag{2.44}$$

In the above equation we have omitted the dependence with respect to other spatial coordinates for more clarity.

Boundary conditions are taken into account in the classical way:

- Horizontal Boundary Conditions.

  For $i = 1$:

$$K_x(x_{i-\frac{1}{2}})\frac{c(x_i) - c(x_{i-1})}{x_i - x_{i-1}} = 0 \tag{2.45}$$

  For $i = n_x$:

$$K_x(x_{i+\frac{1}{2}})\frac{c(x_{i+1}) - c(x_i)}{x_{i+1} - x_i} = 0 \tag{2.46}$$

  where n_x is the number of grid cells in the x-direction.

- Vertical Boundary Conditions

  At the ground ($i = 1$):

$$K_z(z_{i-\frac{1}{2}})\frac{c(z_i) - c(z_{i-1})}{z_i - z_{i-1}} = E - v^{dep}c(z_i) \tag{2.47}$$

  At the top ($i = n_z$):

$$K_z(z_{i+\frac{1}{2}})\frac{c(z_{i+1}) - c(z_i)}{z_{i+1} - z_i} = 0 \tag{2.48}$$

  where n_z is the number of grid cells in the x-direction.

Chapter 3

Software structure of Polair

3.1 Short description

Polair is a three-dimensional chemistry-transport model based on the models described in Chapter 2. Polair is a research tool and has been built as a numerical platform for several applications. It is released under the GNU General Public License.

It was mainly written by Jaouad Boutahar, Denis Quélo and Bruno Sportisse from CEREA[†1]. It was written in Fortran 77 so that it could be automatically differentiated[†2].

It can handle different chemical schemes and compute concentrations through Equation (2.1). Extensions enable to perform simulations with aerosols (see Chapter 5).

As for inverse modeling, the adjoint model, generated thanks to automatic differentiation, enables to work on sensitivity analysis, on data assimilation, etc.

3.2 Code

3.2.1 Configuration

Polair is neither a library nor a software. There is script called `POLAIR` which compiles and launches Polair. After compilation, Polair is a program that runs on a given case, set up before compilation. In order to set up a case, one has to write (or to modify) configuration files.

There is a particular file which is called the input file, because it is the input file of the script `POLAIR`. Hence, one launches "`POLAIR InputFile.inc`" in order to launch a simulation. The input file contains the directory of configuration files. So, through the input file, configuration files are determined, and, then, the simulation is set up.

The input file and configuration files are described in chapter 4.

3.2.2 A few numerical issues

Equation (2.1) and equations related to other processes are discretized on an 3D-grid (orthogonal mesh) containing $N_x \times N_y \times N_z$ points, N_d being the number of points along direction d. Each point is a node centered within a cell. Space steps Δx and Δy are constant whereas Δz is not.

[†1] Centre d'enseignement et de recherche en environnement atmosphŕique – École nationale des ponts et chaussées `http://www.enpc.fr/cerea/`

[†2] Actually, it is automatically differentiable by O∂yssée.

For instance, for a typical simulation over Paris, we have: $N_x = N_y = 25$ and $\Delta x = \Delta y = 6\,000$ m. In the vertical direction, the first-node height is 15 m, the second-node height is 90 m, the third-node height is 250 m.

Data is provided at nodes or at cell interfaces. Concentrations are computed at nodes, in $\mu g \cdot m^{-3}$.

3.2.3 Directories tree

Home directory

At home directory (let us say `~POLAIR`), one can find the following directories:

- ▷ `bin`: script `POLAIR`.
- ▷ `Code`: code, i.e. Fortran functions "`*.f`".
- ▷ `Run_references`: simulations that work (i.e. configuration files are correct) and that should be used to set up a new case.
- ▷ `UsersGuide`: this guide.

Code

In directory `Code`, one may find the GNU General Public License (in file `LICENSE`) and the version of Polair (file `version`).

The code directory contains the following directories:

- ▷ `ADVECTION`: advection numerical-scheme.
- ▷ `CHEMISTRY`: chemical schemes (`EMEP`, `MOCA`, `OZ16`, `CBM IV`, `MELCHIOR`, `RADM`, `EURORADM`, `RACM`, `MERCURY`, `PASSIVE_TRACERS`).
- ▷ `DIFFUSION`: diffusion numerical-scheme.
- ▷ `DIRECT`: main function `CTMASTER.f`.
- ▷ `DRYDEPOSITION`: dry deposition velocities.
- ▷ `INCLUDE`: files that are included by many subroutines; those files contain common definitions.
- ▷ `MAITRE`: integration function `ctm.f`.
- ▷ `POSTRT`: functions called to save data from computations.
- ▷ `SETUP`: initialization functions.
- ▷ `SPLITTING`: splitting of first and second order.
- ▷ `SCAVENGING`: gas below-cloud scavenging.

`Run_references` directory

`Run_references` contains reliable simulations. Each simulation is defined, in a dedicated directory, by configuration files.

As for a direct simulation, the corresponding directory contains a "`.inc`" file which is the input file (see section 3.2.1, and section 4.1 for details). Moreover, it contains too directories: `CONTROL` and `PARAMETER`.

Most of configuration files are in `CONTROL`. They begin with "`NAM`" and define most of simulation parameters. Additional parameters (especially those related to a specific chemical scheme), which have to be known at compile time because of memory allocation, are in files contained by `PARAMETER`: `PARADOM.inc` and `PARACHEM.inc`.

Further details may be found in section 4.1.

3.2.4 Code overview

Figures (3.1), (3.2) and (3.3) display code structure.

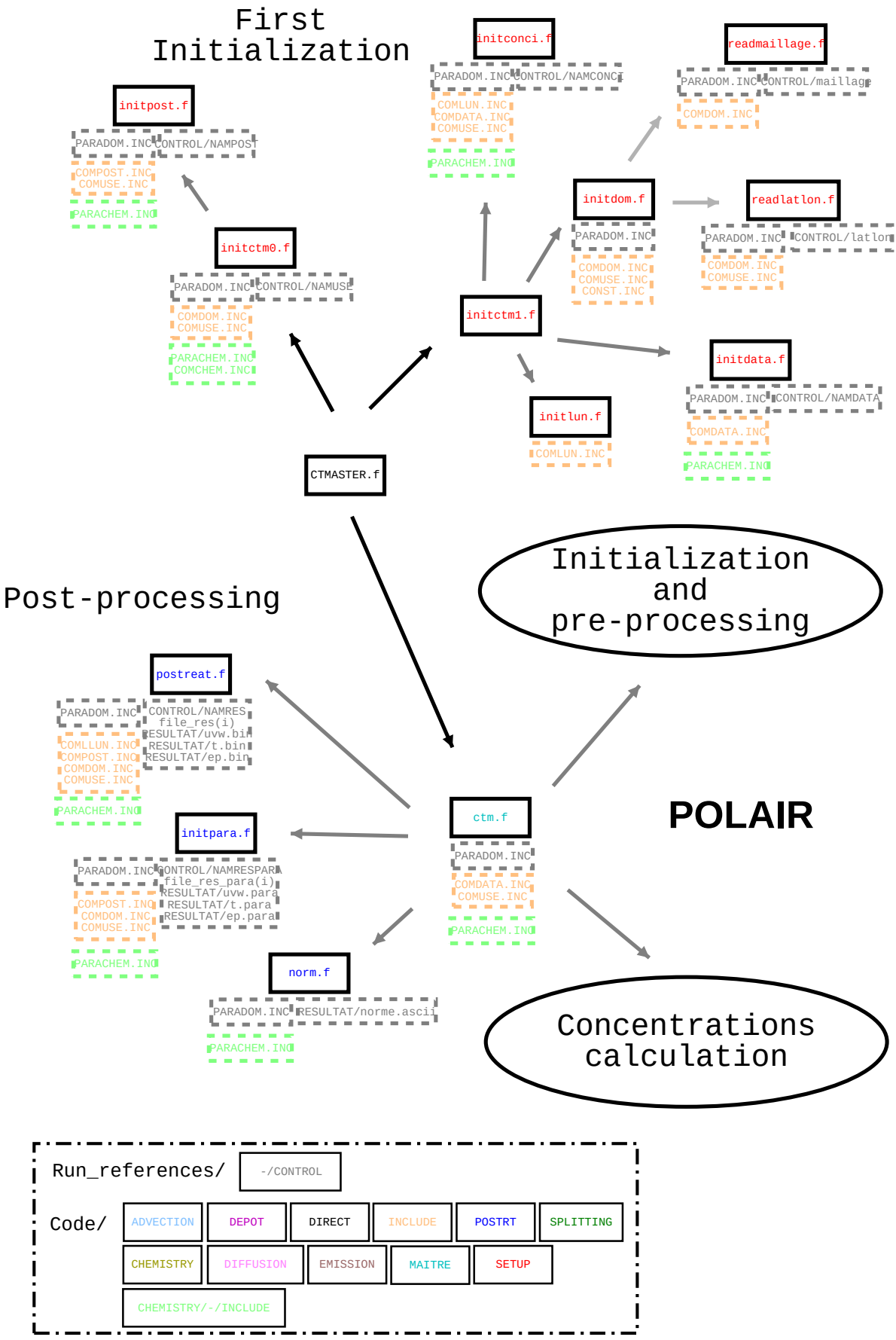

Figure 3.1: Code structure – first initializations and post processing.

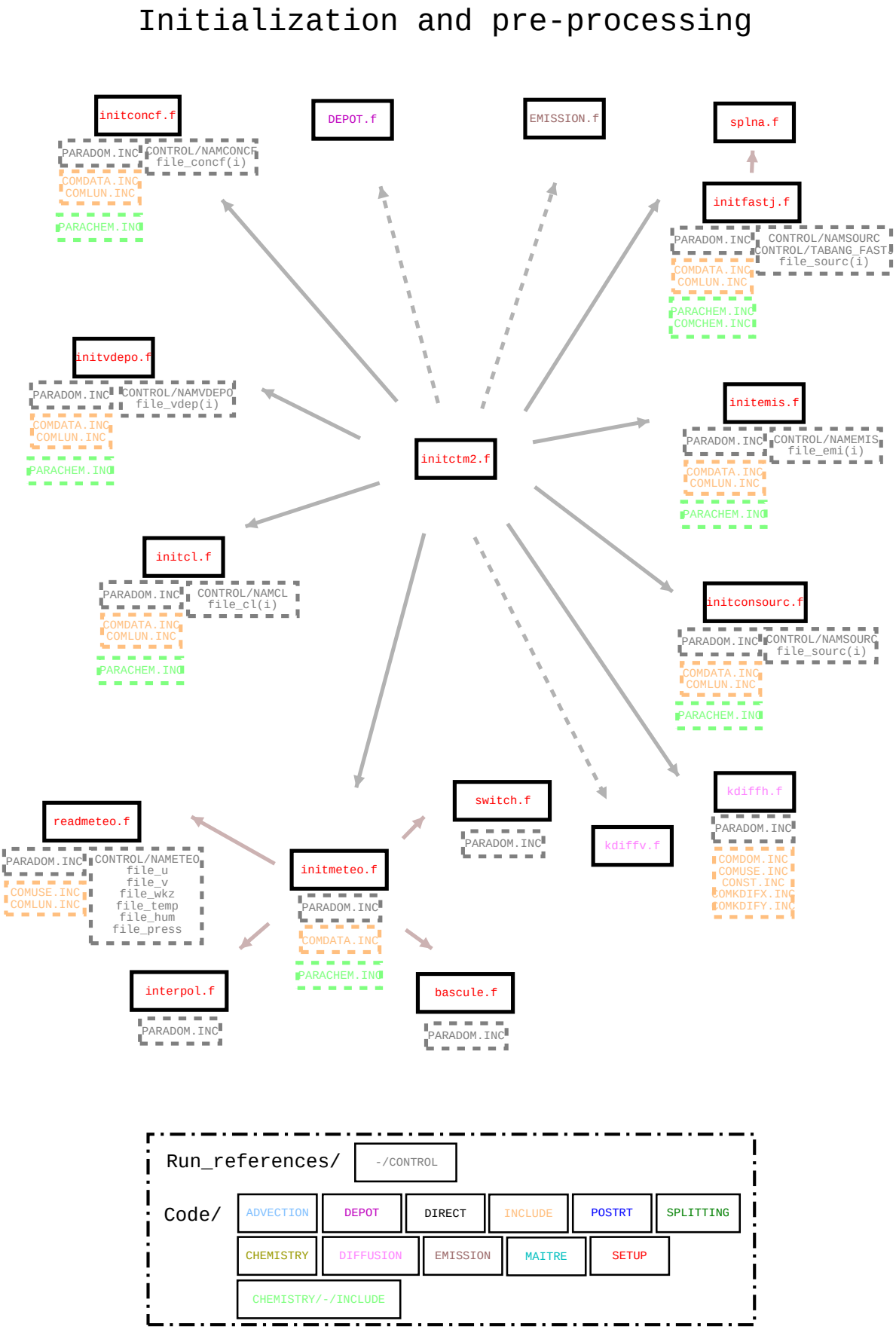

Figure 3.2: Code structure – initializations (at each timestep).
This scheme corresponds to the code version 1.1.

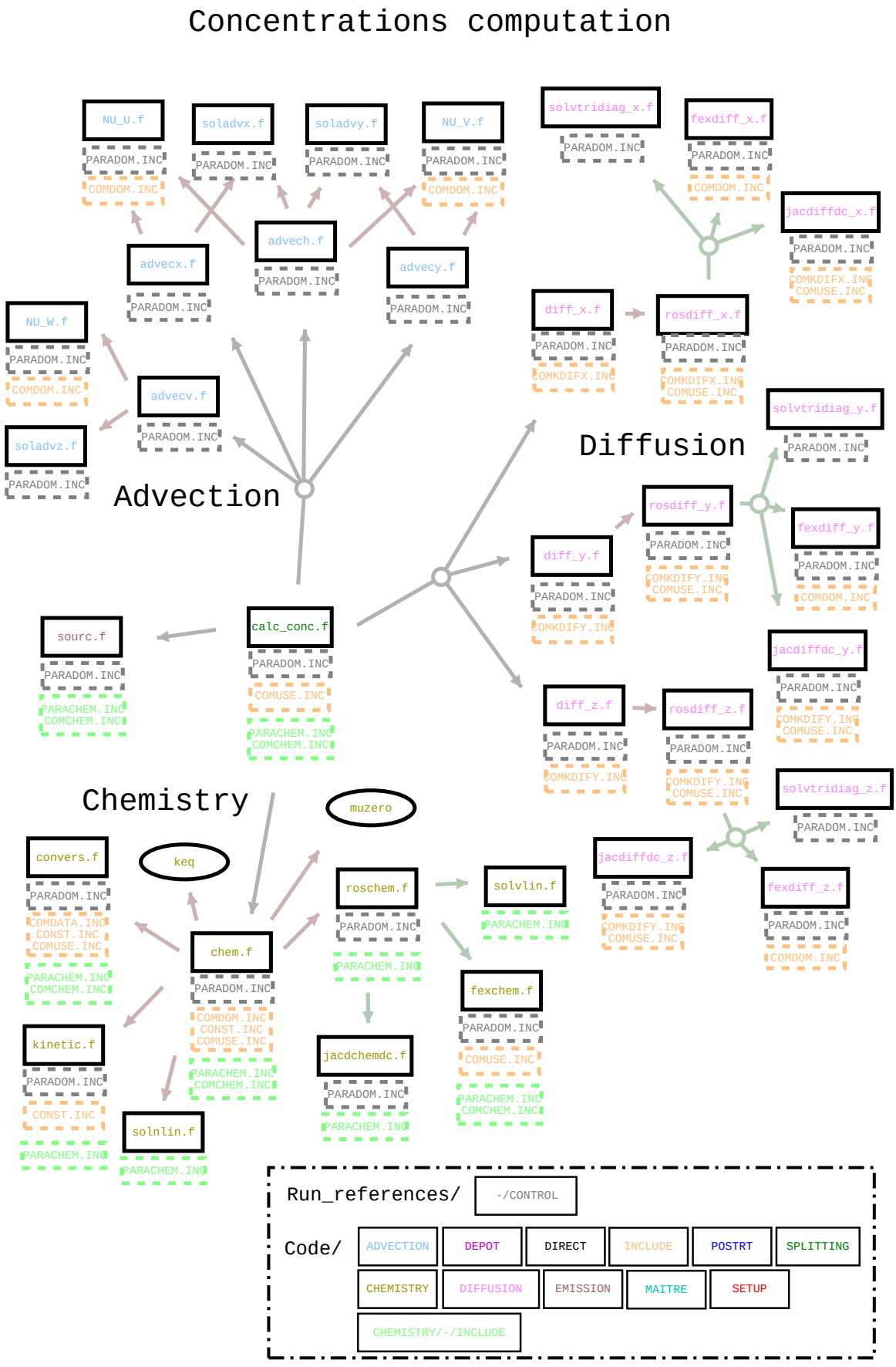

Figure 3.3: Code structure – computations (for chemistry "EMEP").
This scheme corresponds to the code version 1.1.

Chapter 4

How to use

4.1 Configuration files

In this section, we describe all files that are needed in order to set up a simulation, except data files (see section 4.2 for details on input data-files).

Most inputs files have a fixed format which must remain as is. For instance, one should never remove or add a line in `NAM*` files.

4.1.1 Input file

The input file is a parameter of script `POLAIR`. The input file for a simulation with the chemical mechanism RADM could be:

```
#################################################################
# Code path (from home directory)
#################################################################
Polair/Code

#################################################################
# "CONTROL" path (from home directory)
#################################################################
Polair/Run_references/Radm/CONTROL

#################################################################
# File "PARADOM.INC" (path from home directory and name)
#################################################################
Polair/Run_references/Radm/PARAMETER/PARADOM.INC

#################################################################
# File "PARACHEM.INC" (path from home directory and name)
#################################################################
Polair/Run_references/Radm/PARAMETER/PARACHEM.INC

#################################################################
# path (from home directory) to the program
```

```
#################################################################
Polair/Code/DIRECT

#################################################################
# Results directory (from current directory)
# If needed, the directory is made.
#################################################################
RESULTS

#################################################################
# Chemistry scheme: EMEP, OZ16, MOCA, RADM, RACM, MELCHIOR
#################################################################
RADM

#################################################################
# Splitting order: ORDRE1, ORDRE2
#################################################################
ORDRE1

#################################################################
# Compilation line
# Debugging compilation: g77 -g -Wall -W -fbounds-check
# Optimized compilation:
#    ifc -lg2c -cm -w -align -O3 -tpp7
#    g77 -malign-double -mcpu=pentiumpro -march=pentiumpro -mpentiumpro -O3
#################################################################
ifc -lg2c -cm -w -align -O3 -tpp7
```

The input file mainly provides directories[†1]:

- ▷ `POLAIR/Code`: code directory.
- ▷ `POLAIR/Run_references/Radm/CONTROL`: configuration files (mainly "`NAM*`").
- ▷ `POLAIR/Run_references/Radm/PARAMETER/PARADOM.INC`: memory allocation, see description of `PARADOM.INC` in section 4.1.2.
- ▷ `POLAIR/Run_references/Radm/PARAMETER/PARACHEM.INC`: parameters for the chemical mechanism.
- ▷ `POLAIR/Code/DIRECT`: directory containing `CTMASTER.f` (main function).
- ▷ `RESULTS`: directory where output files are stored. This directory is a subdirectory of the directory where the script is launched.
- ▷ `RADM`: chemical scheme.
- ▷ `ORDRE1`: choose "`ORDRE1`" for a first-order splitting and "`ORDRE2`" for a second-order splitting.
- ▷ `ifc -lg2c -cm -w -align -O3 -tpp7`: compilation command, used for compilation of all Fortran functions.

[†1] Except for results, paths are referred from the Unix `HOME` directory.

4.1.2 Configuration files

In configuration-files directory (e.g. `POLAIR/Run_references/Radm/CONTROL`), one may find:

- ▷ `NAMCL`: names of files for boundary conditions.
- ▷ `NAMCONCI`: names of files for initial conditions.
- ▷ `NAMDATA`: initial time, time step and number of stored steps for meteorological data, emissions, deposition velocities, forced concentrations, boundary conditions, photolytic constants and cloud attenuation.
- ▷ `NAMEMIS`: names of files for emissions.
- ▷ `NAMETEO`: names of files for meteorological data.
- ▷ `NAMPHOTOLYSIS`: names of files for clear-sky photolysis rates and indices of those reactions (set in Spack reactions input-file), time steps (i.e. days at which photolysis rates are provided), time angles, latitudes and heights (at which rates are provided).
- ▷ `NAMPHYS`: physical constant for chemical species.
- ▷ `NAMPOST`: dry deposition velocities, scavenging coefficient and which concentrations should be saved. One chooses for which species and at which vertical levels concentrations have to be saved.
- ▷ `NAMRES`: names of files into which computed concentrations are saved.
- ▷ `NAMSOURC`: names of files for sources.
- ▷ `NAMSCAV`: names of species for scavenging.
- ▷ `NAMSPECIES`: names of species. This file is a SPACK input.
- ▷ `NAMUSE`: main configuration file, described below.
- ▷ `NAMVDEPO`: names of files for deposition velocities.
- ▷ `TABANG.FASTJ`: zenithal angles at which photolytic constants are provided.
- ▷ `latlon`: latitude/longitude coordinates of grid points (on the ground).
- ▷ `maillage`: heights of cells interfaces (not nodes).

Each time a file name is provided, it is associated to a given species (except for boundary condition – see in `NAMCL`). In files `NAMCONCI`, `NAMEMIS`, `NAMRES`, `NAMSOURC` and `NAMVDEPO`, files may be provided in any order, since they are associated to a given species (specified before the file name). *To insert blanks (between species names and file names), one must use whitespaces instead of tabular.*

`NAMUSE` gathers main simulation-options:

- ▷ Integration issues: initial time (GTM time, in seconds from the beginning of the current year), time step Δt, and number of iterations.
- ▷ Iterations at which data is saved (if any).

- ▷ The horizontal-grid format: Cartesian coordinates or latitude/longitude coordinates
- ▷ The horizontal-grid definition: southwestern-most point of the grid, and space steps Δx and Δy.
- ▷ Which physical processes have to be included in the simulation: chemical reactions, advection, diffusion, surface emissions, volume emissions, dry deposition, below-cloud scavenging.
- ▷ Which inputs are available: boundary conditions, forced concentrations, initial concentrations, photolytic constants, vertical wind, temperature, pressure, humidity, precipitations, height of cloud basis, horizontal diffusion constant. *If the vertical wind is not available, it is computed by Polair so that the divergence of the wind is zero.*
- ▷ For which species we have: surface emissions, volume emissions, dry deposition velocities and boundary conditions.
- ▷ Whether comments are displayed on screen.
- ▷ Whether post treatment is used.
- ▷ Whether a previous simulation is used to restart, and in this case what is the number of the output to start from.

`PARADOM.INC` is used for memory allocation. Since there is no dynamic memory-allocation, arrays sizes are set. Thus, one sets:

- ▷ The number of grid nodes `Nx`, `Ny` and `Nz`;
- ▷ The number of points `Nxt`, `Nyt` and `Nzt` where temperature is provided;
- ▷ The number of points `Nxp`, `Nyp` and `Nzp` where pressure is provided;
- ▷ The number of points `Nxatt`, `Nyatt` and `Nzatt` where cloud attenuation is provided;
- ▷ The number of points `Nxh`, `Nyh` and `Nzh` where humidity is provided;
- ▷ The number of points `Nxr`, `Nyr` and `Nzr` where precipitations are provided;
- ▷ The number of points `Nxf`, `Nyf` and `Nzf` where forced concentrations are provided;
- ▷ The number of points `Nxl`, `Nyl` and `Nzl` where boundary conditions are provided.
- ▷ The number of points `Nxdd`, `Nydd` and `Nzdd` where data is provided for dry deposition velocities computing and `Nland`, `Nseason`, `Nlayer` additional dimensions for dry deposition parameters.

In fact, `Nx*` (i.e. `Nxt`, `Nxp`, `Nxh`, etc.) equals 1 if the involved field is not available, or equals `Nx` if the involved field is available. This is due to Fortran 77 which doesn't enable dynamic memory allocation. For instance, if temperature is not available, one sets (`Nxt`, `Nyt`, `Nzt`) to (1, 1, 1) to save memory (i.e. to allocate a $1 \times 1 \times 1$ array). Otherwise, temperature is available (at all nodes), and (`Nxt`, `Nyt`, `Nzt`) equals (`Nx`, `Ny`, `Nz`).

If all options are set, the last requirement is to provide input data-files.

4.2 Input data-files

4.2.1 Notations

Recall the following definitions. Cell centers are called nodes. Cells boundaries are called interfaces. Along x, there are N_x nodes and $N_x + 1$ interfaces. For any field, N_t is the number of timesteps at which data for the field is available.

In all data files, values are stored in *single-precision binary files* as below:

* Loop on time t
    * Loop on z
        * Loop on y
            * Loop on x

Let those loops be symbolized by $\{t, z, y, x\}$.

Recall that the set of dates, in which t is running, is defined in `NAMDATA`. Spatial loops depend on the grid and on the kind of data. 3D data may be provided at $N_x \times N_y \times N_z$ nodes[†2] (e.g. concentrations) or at cells interfaces (e.g. wind speeds) in the normal direction[†3] to the interface.

Let x_i, y_i and z_i be nodes coordinates. Let α_i, β_i and γ_i be coordinates of cell corners (which provide "interfaces coordinates") so that:

$$x_{i-1} < \alpha_i < x_i, \quad y_{i-1} < \beta_i < y_i, \text{ and } \quad z_{i-1} < \gamma_i < z_i$$

As for x_i, y_i and z_i, index i runs in $[\![1, N_d]\!]$ where $d \in \{x, y, z\}$. As for α_i, β_i and γ_i, index i runs in $[\![1, N_d + 1]\!]$ where $d \in \{x, y, z\}$ (respectively).

4.2.2 Data provided at cells interfaces

Vertical-diffusion constant K_{zz}

Along z, the vertical-diffusion constant K_{zz} is provided on interfaces (along z). Then, the corresponding loops are $\{t, \gamma, y, x\}$[†4]; $N_x \times N_y \times (N_z + 1)$ values are read.

Notice that values at the domain bottom and at the domain top (i.e. for γ_1 and γ_{N_z+1}) is provided, but it is not used by Polair.

Wind speed

Wind speeds are provided in format $\{t, z, y, \alpha\}$, $\{t, z, \beta, x\}$ and $\{t, \gamma, y, x\}$ for winds speeds along x, y and z respectively.

In case where the domain bottom is at ground level (i.e. $\gamma_1 = 0$), notice that ground-level wind-speeds along z is provided and should be set to zero.

If latitude/longitude coordinates are used, then horizontal winds must be transformed. Let u be the zonal wind and v the meridional wind. Wind fields must be transformed according to the following transformations:

$$U = \frac{u}{\cos\varphi} \qquad \text{and} \qquad V = v\cos\varphi$$

where φ is the latitude.

U and V are zonal and meridional winds for Polair.

[†2] i.e. cells centers.

[†3] The normal is directed along increasing coordinates.

[†4] Previously defined as symbol for: loop on t, loop on γ, loop on y and loop on x.

4.2.3 Data provided at nodes

3D data

Emissions (volume) and initial concentrations are written in format $\{t, z, y, x\}$; $N_x \times N_y \times N_z$ values are read. Notice that initial concentrations are needed for every species. One cannot choose for which species initial concentrations are provided.

Specific humidity (also called water mass fraction), temperature (in K) and pressure (in Pa) are written in the same format, $\{t, z, y, x\}$ values are read. For the EMEP chemistry model relative humidity is needed, and note that dry deposition velocities have to be read and not computed.

For each photolysis reaction, clear-sky photolysis rates are read in files in format $\{d, angle, lat, z\}$ where d is the time, $angle$ the time angle (in $[0, 12]$), lat is the latitude and z is the height. Those coordinates are defined in `NAMPHOTOLYSIS`.

Cloud attenuation coefficients are numbers by which all photolytic constants are multiplied. They are provided in format $\{t, z, y, x\}$.

2D data

Surface emissions are in format $\{t, y, x\}$. If read (i.e. computed outside Polair), deposition velocities are provided in format $\{t, y, x\}$. Otherwise, they are computed according to the following data set. Surface temperature (in K) surface pressure (in Pa), snow depth (in m), rain intensity (in mm.h^{-1}), surface solar radiation (in $\text{W.m}^{-2}\text{.s}$), surface latent heat flux (in $\text{W.m}^{-2}\text{.s}$), soil wetness level (in m) and module of surface wind (in m.s^{-1}) are in format $\{t, y, x\}$. Land Use Category data (in %) are in format $\{LUC, y, x\}$.

For scavenging height of cloud basis (in m) are in format $\{t, y, x\}$.

Boundary conditions are known concentrations outside the domain. They are provided on an extra layer around the domain. For instance, along x direction, one provides concentrations at nodes (x_0, y_j, z_k) and (x_{N_x+1}, y_j, z_k) where $j \in [\![1, N_y]\!]$ and $k \in [\![1, N_z]\!]$. If boundary conditions are provided in x or y direction, they are available at both sides, e.g. for x_0 and x_{N_x+1}. In z direction, boundary conditions are provided only at the top (z_{N_z+1}).

4.2.4 Remarks

Notice that the time step for input data (specified in `NAMDATA`) cannot be strictly smaller than the integration time step (specified in `NAMUSE`).

4.3 Output files

4.3.1 Simulation configuration

In the results directory, the directory `info` stores the configuration of the simulation. `CONTROL` directory, `PARADOM.INC`, `PARACHEM.INC`, and the script input-file are saved. Moreover, the file `config.log` stores `NAMUSE`, `NAMPOST` and the version of Polair which was used. Finally, the file `timing.log` stores: the name of the host which computed results, the date and the CPU time used by the simulation.

4.3.2 Concentrations

Concentrations are stored in binary files. Concentrations of species (indicated in `NAMRES`[†5]) are stored in format $\{t, z, y, x\}$ (z runs in a set specified in `NAMPOST`[†5]). Concentrations may be integrated (in time) or not[†6]. See explanations below.

Integrated concentrations

Measurements may be integrated (over a given period, e.g. one hour) concentrations, not instantaneous concentrations. So, Polair is able to perform a trapezoidal integration over a given period of time.

A number N_{int} of iterations over which concentrations are to be integrated is provided in `NAMPOST`. Let the initial state (initial concentrations) be the step #0 and the state after one step be the step #1 of the simulation. Then the first integration is performed between steps #0 and step $\#N_{int}$. The second integration is performed between steps $\#N_{int}$ and $\#(2 \times N_{int})$.

Instantaneous concentrations

If instantaneous concentrations are chosen, notice that initial concentrations are saved.

4.3.3 Remarks

In the absence of diffusion, there will be no deposition and surface emissions.

4.4 Compiling and running Polair

4.4.1 Requirements

To run Polair, one needs a Fortran 77 compiler, BLAS and LAPACK.

4.4.2 Getting Polair

In order to use Polair 1.1 (for instance), one uses CVS features. Let `$CVSROOT` be the CVS root-directory which stores Polair (for instance, at CEREA, it is `/u/cergrene/0/sportiss/cvsroot`). In your home directory, launch the command: "`cvs -d [$CVSROOT] export -r Polair-1_1 Polair`".

It is even possible to perform this download from a remote server. Please refer to CVS documentation for that purpose. If you want to import the code in another directory (the default directory being `Polair`), type: "`cvs -d [$CVSROOT] export -r Polair-1_1 -d [your Polair directory] Polair`". Nevertheless, simulations in `Run_references` work if Polair is downloaded in directory `Polair` (from the home directory)[†7].

Data should be obtained by other means: it is not included in the CVS root-directory. At CEREA, data (for simulation in `Run_references`) is mainly in `/u/cergrene/0/sportiss/POLAIR/Data`.

[†5] See section 4.1.2.

[†6] The choice is made in `NAMPOST`.

[†7] See the input file of the script, which provides directories.

Then, add the Polair-script directory (`Polair/bin`) in Unix variable `PATH`. It may be done by adding, in the file "`.tcshrc`" (in your home directory), the following line: "`set path = ($path $HOME/Polair/bin)`". Then, open a new terminal so that `PATH` should be changed (when a terminal is launched, the file "`.tcshrc`" is read in order to set up the configuration).

4.4.3 Compiling and running Polair

The Polair script enables to launch Polair easily. Launch: "`POLAIR InputFile.inc`" where `InputFile.inc` is the input file described in section 4.1.1. Then, Polair is compiled and is launched.

One may launch a reference simulation (in `Run_references`). To do so, one may copy one of `Run_references` directories (let's say `Radm`). Then, one must change paths in the main input file (`Radm.inc`, for Radm), and maybe the compilation line. Finally, paths (to input data) in configuration files should be changed. Then, "`POLAIR Radm.inc`" will launch the right simulation.

4.4.4 Working with Polair

One of the best ways to work with Polair is to make a working directory (for instance, `~/runs`) in which all simulations are stored in dedicated directories: `~/runs/Radm_IOP2`, `~/runs/Radm_IOP5`, `~/runs/Mercury_2999`, ... Actually, one may store several simulations in the same directory provided that they are similar (recall that the simulation configuration is stored with results – see section 4.3.1).

Those simulations may share the same code (e.g. `~/codes/Polair-1.1`). If changes in the code are needed for a given simulation, then one could copy the modified code in the working directory. For instance, if one wants to modify the code for the simulation in `~/runs/Radm_IOP5`, one could put the modified code in `~/runs/Radm_IOP5/Polair`.

Chapter 5

Additional abilities

5.1 Modal models of atmospheric aerosols

5.1.1 Modeling

The aerosol distribution may be modeled by the general dynamic equation:

$$\frac{\partial n}{\partial t} = (\frac{\partial n}{\partial t})_{adv} + (\frac{\partial n}{\partial t})_{diff} + (\frac{\partial n}{\partial t})_{coag} + (\frac{\partial n}{\partial t})_{cond} + (\frac{\partial n}{\partial t})_{nucl} + (\frac{\partial n}{\partial t})_{emi}$$

where $n(v,t)$ is the number distribution of aerosols, which volume ranges between volumes v and $v + dv$. A similar equation gives the evolution of chemical species in the aerosols. *Adv*, *Diff*, *Coag*, *Cond*, *Nucl* and *Emi* stand respectively for advection (including gravitational settling), diffusion, coagulation, condensation/evaporation, nucleation and emission.

In *modal models* the number of aerosols is fixed as the sum of two (or three) log-normal distributions: for instance, the first one describes the nuclei mode, the second one the accumulation mode. In practice evolution equations for three moments, M_0, M_3 and M_6, of each mode are solved. To know chemical composition of aerosols the third moment is subdivided in as many parts as there are involved chemical gaseous species. Finally to model the aerosol distributions $N_{aero} = 2 * (2 + N_{gas})$ moments and N_{gas} concentrations need to be computed (N_{gas} = number of gaseous species involved).

5.1.2 Aerosols in POLAIR

To compute aerosols, the first step is to set $LAERO = .TRUE.$ in NAMUSE. It is impossible to compute aerosol if $LCHIM = .FALSE.$ because the chemistry routines chem.f calls the aerosol solver.

As for gaseous concentrations it is possible or not to take into account advection ($LADVEC$) or diffusion($LDIFF$) for example. For aerosols other booleans need to be set to solve the general dynamic equation, $LNUCL$, $LCOND$, $LCOAG$, $LEMISAER$, respectively for nucleation, condensation, coagulation and emission. Some of them need complementary information if activated (see NAMAERO). The last one $LSPLAER$ allows to choose a solving method, aerosol processes are splitted and solved with a second-order midpoint explicit scheme (Explicit Trapezoidal Rule) or not splitted and solved with Rosenbrock.

Initial values must be given in NAMAERO, for each mode number of aerosol, mean diameter and standard deviation. Percentage of sulfate, nitrate and ammonium in the initial distribution.

The output are set in NAMPOST, moments are written in result files . A post-treatment is needed to compute the number of aerosols, the diameter or their composition.

5.1.3 More information

More detailed informations are available in [17]:

1. about modal models,
2. about aerosol programming in POLAIR → sartelet/POLAIR/Code/README.MODIF_AERO.

5.1.4 Future

The next version of POLAIR should contain a size-resolved simulation of aerosols.

5.2 Inverse modeling and data assimilation

5.2.1 Theory

The purpose of inverse modeling and data assimilation is to combine observations (measurements) and model results. The classical approach for variational methods (like $4Dvar$) yields to minimizing a cost function (let say J), defined as a gap between model outputs and observations, with respect to parameters (for example: initial conditions, emission factors,...).

A gradient-like method is used in order to find iteratively the solution of the control problem. This makes necessary to compute several times the gradient of J. Due to RAM limitations a tricky algorithm is used in order to compute it. The key point is to get the adjoint code of `calc_conc`[†1] and its dependencies. It is provided by O∂yssée ([2]), an automatic differentiation tool, on the basis of some of the subroutines of Polair.

5.2.2 The algorithm to compute the gradient

Polair may be described at the algorithmic level as follows:

1. Initialization of time-independent data,
2. Time loop (labelled by $1 \leq i \leq n$):
   - read forced data ϕ_{i-1} (this contains data needed by the solver to compute concentrations at time t_i),
   - compute new state $C_i = F(C_{i-1}, \phi_{i-1})$[†2],
   - compute the cost function at time t_i,
   - update the cost function: $J = J + J_i$.

Computing ∇J may be directly done. This is unfortunately unaffordable due to storage requirements: it would need to store all the forced data $(\phi_0, \ldots, \phi_{n-1})$ and the trajectory $(C_0, \ldots, C_n)$. We use another approach by noticing that $\nabla J = \sum_i \nabla J_i$, that is in the case of data assimilation:

$$\begin{aligned} \nabla_u J &= \sum_{1\leq i\leq n} \left(\frac{\partial C_1}{\partial C_0}\right)^T_{|t_0} \cdots \left(\frac{\partial C_i}{\partial C_{i-1}}\right)^T_{|t_{i-1}} \left(\frac{\partial J_i}{\partial C_i}\right)^T_{|t_i} \\ &= \ldots \times \left[\left(\frac{\partial J_{n-2}}{\partial C_{n-2}}\right)^T + \left(\frac{\partial C_{n-1}}{\partial C_{n-2}}\right)^T \left[\left(\frac{\partial J_{n-1}}{\partial C_{n-1}}\right)^T + \left(\frac{\partial C_n}{\partial C_{n-1}}\right)^T \left(\frac{\partial J_n}{\partial C_n}\right)^T \right] \right] \end{aligned} \tag{5.1}$$

We now need adjoint models in order to compute, for any vector z (with coherent dimension):

$$\left(\frac{\partial C_i}{\partial C_{i-1}}\right)^T_{|t_{i-1}} z \tag{5.2}$$

This requires as well to store the trajectory $(C_0, C_1, \ldots, C_n)$ (practically in files).
The algorithm now reads:

1. $\hat{C}_n = \left(\frac{\partial J_n}{\partial C_n}\right)^T$,

[†1] Polair subroutine computing one timestep.
[†2] Actually, F is subroutine `calc_conc`.

2. backward time loop (labelled by $n \geq i \geq 1$):
   - read forced data ϕ_{i-1},
   - read saved state C_{i-1},
   - compute $\hat{C}_{i-1} = \left(\frac{\partial C_i}{\partial C_{i-1}}\right)^T_{|C_{i-1},\phi_{i-1}} \hat{C}_i$,
   - update $\hat{C}_{i-1} = \hat{C}_{i-1} + \left(\frac{\partial J_{i-1}}{\partial C_{i-1}}\right)^T$.
3. the output is $\nabla_{C_0} J = \hat{C}_0$.

An example of this implemented algorithm can be found in ??.

5.2.3 How to get the adjoint subroutine of *calc_conc*

The purpose is to obtain the derivative code of *calc_conc* with as few manual interventions as possible.

Using O∂yssée

O∂yssée is an automatic differentiation tool for Fortran77 code whose user's guide can be downloaded at `http://www.inria.fr/rrrt/rt-0224.html`.

Let's see through a simple example how to obtain the derivative code of the subroutine `example` contained in file `example.f`. It may be achieved in three steps:

1. Load the subroutine in O∂yssée: `load example.f`
2. Differentiate in adjoint mode (*cl*) with respect to the input variables: `diff -h example -cl -vars input`
3. Get the adjoint code in `adjoint.f`: `getdiffprogram examplecl adjoint.f`

Adjoint of `solvlin`

The adjoint code of the linear solver `solvlin` can not be obtained directly by using O∂yssée. In fact, `solvlin` calls subroutines `dgetrf` and `dgetrs` coming from the Lapack library that is not automatically differentiable by O∂yssée. Moreover it is cheaper (in terms of CPU time) to write the adjoint by hand. The general algorithm of the adjoint of linear solvers may be found in [18].

Differentiation of Polair

The `ody.file` contains the commands to get the adjoint code of `calc_conc`:

```
load CODE/ADVECTION/NU_U.f CODE/ADVECTION/NU_V.f CODE/ADVECTION/NU_W.f
CODE/ADVECTION/advech.f CODE/ADVECTION/advecv.f CODE/ADVECTION/advecx.f
CODE/ADVECTION/advecy.f CODE/ADVECTION/phi.f CODE/ADVECTION/soladvx.f
CODE/ADVECTION/soladvy.f CODE/ADVECTION/soladvz.f

load CHIMIE/chem.f CHIMIE/fexchem.f CHIMIE/jacdchemdc.f CHIMIE/roschem.f
CHIMIE/kinetic.f CHIMIE/convers.f CHIMIE/splnb.f CHIMIE/angzen.edf.f

load CODE/DIFFUSION/diff_x.f CODE/DIFFUSION/diff_y.f CODE/DIFFUSION/diff_z.f
CODE/DIFFUSION/fexdiff_x.f CODE/DIFFUSION/fexdiff_y.f CODE/DIFFUSION/fexdiff_z.f
```

```
CODE/DIFFUSION/jacddiffdc_x.f CODE/DIFFUSION/jacddiffdc_y.f
CODE/DIFFUSION/jacddiffdc_z.f CODE/DIFFUSION/rosdiff_x.f CODE/DIFFUSION/rosdiff_y.f
CODE/DIFFUSION/rosdiff_z.f CODE/DIFFUSION/solvtridiag_x.f CODE/DIFFUSION/solvtridiag_y.f
CODE/DIFFUSION/solvtridiag_z.f

load CODE/EMISSION/sourc.f

load SPLITTING/calc_conc.f

load solvlin.f

libgraph

diff -cl -h calc_conc -vars DLCG
getdiffprogram calc_conccl calc_conccl
```

where `CODE`, `CHIMIE` and `SPLITTING` have to be replaced by the directory where files may be found.

Notice that the routine `solvlin` has to be replaced by a fake subroutine that respects the variable dependencies without any calls to the Lapack library.

One can launch ody.file through the command line:
`echo "loadbatch ody.file" | odyssee`.

The script "nettoie"

The adjoint code provided by O∂yssée can not be directly used:

- The saving lines before each call to `solvlin` has to be suppressed.
- O∂yssée includes systematically an include file `odyparam.inc` in each adjoint subroutines. This file has to be created.
- O∂yssée copies some of the include files (used by the direct code) in each adjoint subroutines. It is not done properly and that can lead to errors when compile them. So, it is more convenient to suppress systematically all the declaration coming from the include files and to put them in the file `odyparam.inc`.

One can apply the script "nettoie" that automatically performs the operations mentioned above.

5.2.4 Validation

The Taylor's test indicates the validity of the resulting adjoint model by computing the ratio of ∇J given by the adjoint model over the value provided by finite differences, with different values of the perturbation. The ratio is plotted in Figure 5.1 and is close to 1 for "appropriate values" of perturbation (if the perturbation is too small, round-off errors appear; if it is too large, linearization is no more valid).

Moreover, our tests indicate that the ratio of the CPU time needed for the adjoint model over the CPU time for the initial model is about $4 \sim 5$, which corresponds to a rather classical ratio.

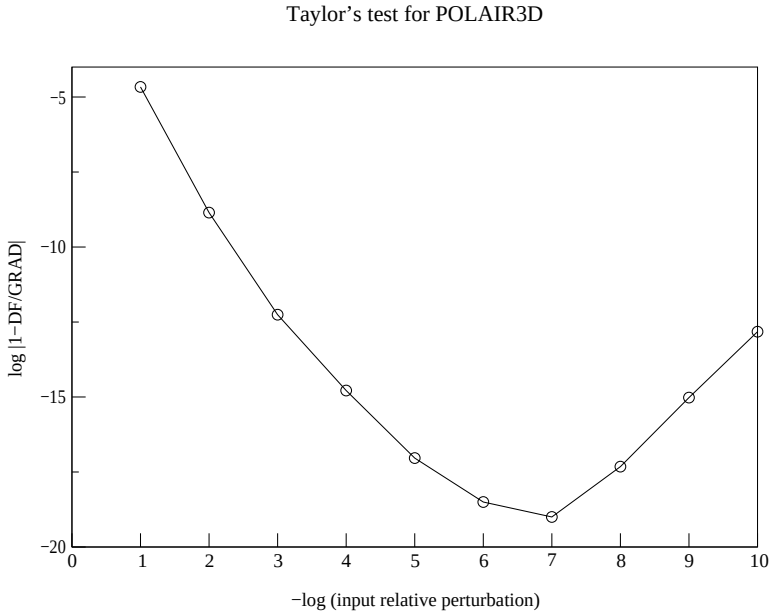

Figure 5.1: Taylor's test for POLAIR. DF is the gradient computed with finite differences, $GRAD$ with the adjoint model.

5.2.5 Description of a future reference run

The required functionalities to perform 4D-var are:

- a cost function `cost`,
- adjoint subroutines of `calc_conc` and `cost`,
- the algorithm that combines the adjoint subroutines in order to compute the gradient of J,
- the optimization driver `BFGS`.

Chapter 6

Some Results obtained with POLAIR

6.1 Results

6.1.1 Continental scale: EMEP Chemistry

POLAIR has been used for simulating EMEP chemistry at continental scale. Benchmarks have been performed with respect to the CTM DIFEUL developed at Electricité de France.

6.1.2 Continental scale: ETEX campaign

POLAIR has been used in order to simulate the campaign ETEX and perform inverse modeling of sources. For direct simulation, POLAIR ranks in the best models.

6.1.3 Regional scale: ESQUIF campaign

The ESQUIF[†1] campaign was a deep study on air quality in Île de France, i.e. Paris region. Twelve intensive observation periods (IOP) brought interesting data to which a CTM is likely to be compared.

On IOP #2, Polair computations were compared to measurements and to Azur computations (a CTM written by LISA[†2]). Results were as good as Azur results. Ozone-prediction accuracy was satisfactory.

Figures (6.1) and (6.2) display some results that were computed by Polair.

[†1] Étude et simulation de la qualité de l'air en Île de France – `http://climserv.lmd.polytechnique.fr/esquif/`.
[†2] Laboratoire inter-universitaire des systèmes atmosphériques – `http://www.lisa.univ-paris12.fr/`.

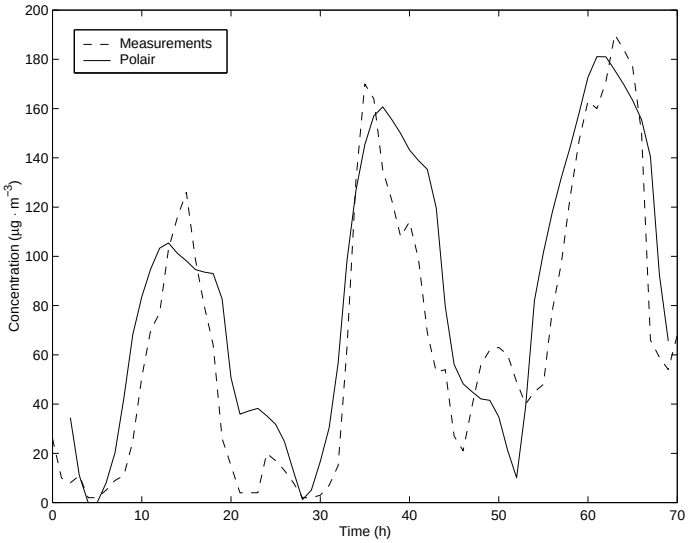

Figure 6.1: Ozone concentration in Paris 18$^{\text{th}}$, from 7$^{\text{th}}$ to 9$^{\text{th}}$ August 1998, compared to measurements.

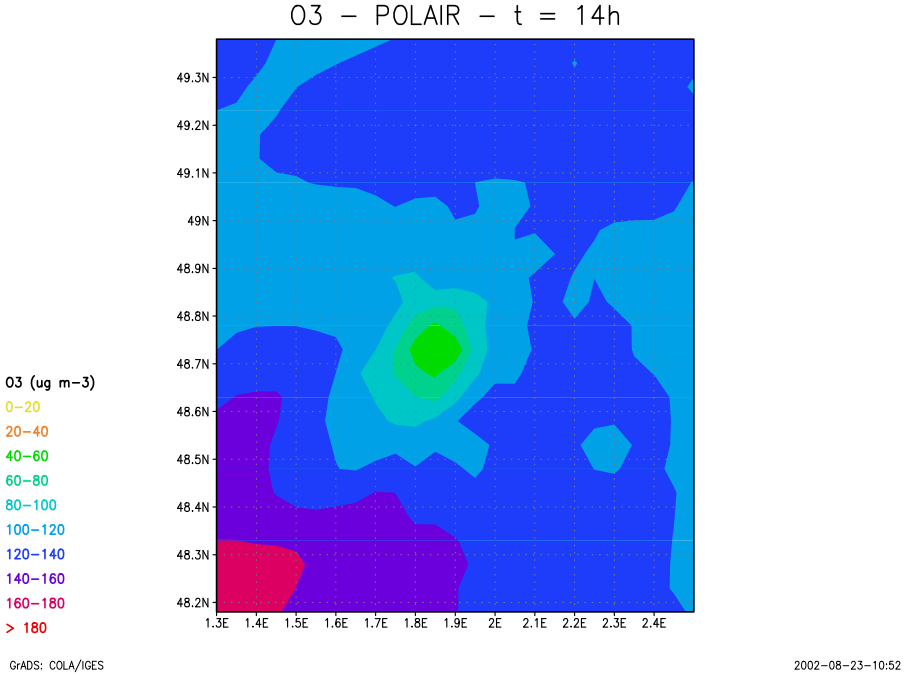

Figure 6.2: Ozone concentations over Île de France, at $t = 14$h and at ground level ($z = 15$ m).

Chapter 7

Future and work in progress

The following features of POLAIR are currently under development or are planned:

1. parameterization of convective processes at continental scale,
2. size-resolved description of aerosols,
3. extension to new "chemical" mechanisms: pesticides, radionuclides, heavy metals and mercury,
4. a Plume-In-Grid model will be included in POLAIR,
5. sequential data assimilation will be provided for POLAIR,
6. the meteorological and chemical preprocessors of POLAIR will be updated.

Bibliography

[1] R. Djouad, B. Sportisse, and N. Audiffren. Numerical simulation of aqueous-phase atmospheric models : use of a non-autonomous rosenbrock method. *Atmos.Environ.*, 36:873–879, 2002.

[2] Y. Papegay C. Faure. Odyssé version 1.6: The language reference manual. Technical Report RT-211, INRIA, 1997.

[3] J.H. Seinfeld. *Atmospheric physics and chemistry of Air Pollution.* Wiley, 1985.

[4] M.Z. Jacobson. *Fundamentals ol Atmospheric modeling.* Cambridge University Press, 1999.

[5] J.G. Verwer, W.H. Hundsdorfer, and J.G. Blom. Numerical time integration for air pollution models. In *Proceedings of the Conference APMS'98.* ENPC-INRIA, October 26-29 1998.

[6] J.F. Louis. A parametric model of vertical eddy fluxes in the atmosphere. *Boundary Layer Met.*, 17:197:202, 1979.

[7] M. Tiedke. A comprehensive mass flux scheme for cumulus parameterization in large scale models. *Monthly Weather Review*, 117:1779–1800, 1989.

[8] M.L. Wesely. Parameterization of surface resistance to gaseous-dry deposition in regional scale numerical models. *Atmos. Environ.*, 23:1293:1304, 1989.

[9] O. Wild, X. Zhu, and M.J. Prather. Fast-j: accurate simulation of in and below cloud photolysis in tropospheric chemical models. *J.Atm.Chem.*, 37:245:282, 2000.

[10] B. Sportisse, R. Djouad, and N. Audiffren. Preprocessing lumped species for multiphase atmospheric models. spack: a simplified preprocessor for atmospheric chemical kinetics. *Submitted to Journal of Atmospheric Chemistry*, 2001.

[11] B. Sportisse and L. Du Bois. Numerical and theoretical investigation of a simplified model for the parameterization of below-cloud scavenging by falling raindrops. *Atmos.Environ.*, 36:5719:5727, 2002.

[12] M. Heimann. The global atmospheric tracer model tm2. Technical report, Max Planck Inst. fur Meteorologie, 1996.

[13] E.J. Spee. *Numerical methods in global transport-chemistry models.* PhD thesis, Univ. Amsterdam, 1998.

[14] J. Graf and N. Moussiopoulos. Intercomparison of two models for the dispersion of chemically reacting pollutants. *Beitr.Phys.Atmosph.*, 64(1):13–25, Febr. 1991.

[15] B. Sportisse. An analysis of operator splitting techniques in the stiff case. *J. Comp. Phys.*, 161:140–168, 2000.

[16] J.H. Verwer, E.J. Spee, J.G. Blom, and W.H. Hundsdorfer. A second order rosenbrock method applied to photochemical dispersion problem. *SIAM J. SCI. COMPUT.*, 20(4):1456–1480, 1999.

[17] K. Sartelet, A. El Aarbaoui, and B. Sportisse. Modal modeling of atmospheric aerosols. Technical Report 2002-12, ENPC-CEREVE, 2002.

[18] J.C. Gilbert F. Eyssette, C. Faure and N. Rostaing-Schmidt. Applicabilité de la différentiation automatique à un système d'équations aux dérivées partielles régissant les phénmènes thermo-hydrauliques dans un tube chauffant. Technical Report 2795, INRIA, 1996.