SEMI-LAGRANGIAN COMPUTATIONS IN THE CYCLE 46T1 OF ARPEGE/IFS.

YESSAD K. (METEO-FRANCE/CNRM/GMAP/ALGO)

December 19, 2018

Abstract:

The general purpose of this documentation is to describe the set of equations used, and also the way to integrate the dynamics of the model with the semi-Lagrangian method currently implemented in ARPEGE/IFS. The following points will be described: semi-Lagrangian formulation and discretisation for different sets of equations, semi-Lagrangian trajectory research, horizontal and vertical interpolations done in the semi-Lagrangian scheme, specific geometric problems met in this type of discretisation. An organigramme is provided. An introduction to tangent linear and adjoint code is provided.

Résumé:

Le but général de cette documentation est de décrire le jeu d'équations utilisé, et également la façon de discrétiser ces équations avec un schéma d'advection semi-Lagrangien tel qu'il est utilisé dans ARPEGE/IFS. On décrit les points suivants: formulation lagrangienne des équations, leur discrétisation avec un schéma semi-lagrangien, recherche de trajectoire, interpolations horizontales et verticales faites dans le schéma semi-lagrangien, problèmes de géométrie spécifiques. On fournit un organigramme. Une introduction au code tangent linéaire et adjoint est également proposée.

Contents

1 Introduction.

The general purpose of this documentation is to describe the set of equations used, and also the way to integrate the dynamics of the model with the semi-Lagrangian method currently implemented in ARPEGE/IFS. Equations will be written without horizontal diffusion scheme (which is treated in spectral computations), and without Rayleigh friction (which is done in grid-point space) in order to give a clearer presentation of the discretised equations. For additional information about horizontal diffusion scheme, report to documentation (IDDH). Deep atmosphere effects are ignored. Extensions to limited area models (LAM) are not described in detail, but differences are briefly mentioned.

The following sets of equations will be described in this documentation:

- The primitive equations hydrostatic model.
- The fully elastic non-hydrostatic equations model (denoted NHEE): uses $\hat{Q}$ and d (or d_4) as NH prognostic variables.
- The quasi-elastic non-hydrostatic equations model (denoted NHQE).
- The 2D shallow-water equations model.

2 Definition of Eulerian and semi-Lagrangian schemes.

2.1 Eulerian scheme.

In Eulerian form of equations, the time dependency equation of a variable X writes:

$$\frac{\partial X}{\partial t} = -\mathbf{U}.\nabla_3 X + \dot{X} \tag{1}$$

where $\mathbf{U}$ is the 3D wind, ∇_3 is the 3D gradient operator, $\dot{X}$ is the sum of the dynamical and physical contributions. $X(t+\Delta t)$ is computed knowing $X(t-\Delta t)$ at the same grid-point. Eulerian technique obliges to use a time-step that matches the CFL (Courant Friedrich Levy) condition everywhere.

- For global variable-mesh spectral global models, the horizontal CFL condition writes:

$$M\frac{\mid \mathbf{V} \mid}{a}\frac{Dt}{2}\sqrt{N(N+1)} < 1 \tag{2}$$

  where M is the mapping factor, Dt is the time-step at the first integration step and twice the time-step otherwise (leap-frog scheme), $\mid \mathbf{V} \mid$ is the horizontal wind modulus, N is the truncation, a is the mean Earth radius.

- For spectral limited area models (LAM), the horizontal CFL condition writes:

$$M\frac{\mid \mathbf{V} \mid}{a}\frac{Dt}{2}(2\pi)\sqrt{\frac{1}{\frac{L_{\mathrm{x}}^2}{a^2 N_{\mathrm{m}}^2}+\frac{L_{\mathrm{y}}^2}{a^2 N_{\mathrm{n}}^2}}} < 1 \tag{3}$$

  See above for denotations M, Dt, $\mid \mathbf{V} \mid$, a. N_{m} is the zonal truncation, N_{n} is the meridian truncation, L_{x} (resp. L_{y}) is the zonal (resp. meridian) length of the LAM domain taken on a surface iso $r = a$.

The vertical CFL condition writes:

$$0.5 \mid \dot{\eta} \mid Dt\Delta\eta < 1 \tag{4}$$

2.2 Semi-Lagrangian scheme.

In semi-Lagrangian form of equations, the time dependency equation of a variable X writes:

$$\frac{dX}{dt} = \dot{X} \tag{5}$$

In a three-time level semi-Lagrangian scheme $X(t+\Delta t)$ is computed at a grid-point F knowing $X(t-\Delta t)$ at the point O (not necessary a grid-point) where the same particle is at the instant $t-\Delta t$. In a two-time level semi-Lagrangian scheme $X(t+\Delta t)$ is computed at a grid-point F knowing $X(t)$ at the point O (not necessary a grid-point) where the same particle is at the instant t. The semi-Lagrangian technique is more expensive for one time-step than the Eulerian technique because it is necessary to compute the positions of the origin point O along the trajectory and to interpolate some quantities at this point. But it allows to use larger time-steps: the stability condition is now the Lipschitz criterion (trajectories do not cross each other) and is less severe than the CFL condition.
D is the divergence of the horizontal wind on the η-coordinates, $\dot{\eta} = \frac{d\eta}{dt}$. Lipschitz criterion writes for a three-time level semi-Lagrangian scheme:

$$\mid D + \frac{\partial \dot{\eta}}{\partial \eta} \mid \frac{Dt}{2} < 1 \tag{6}$$

Lipschitz criterion writes for a two-time level semi-Lagrangian scheme:

$$\mid D + \frac{\partial \dot{\eta}}{\partial \eta} \mid \frac{\Delta t}{2} < 1 \tag{7}$$

Expressions “semi-Lagrangian scheme”, “three-time level semi-Lagrangian scheme” and “two-time level semi-Lagrangian scheme” will be from now on abbreviated into “SL scheme”, “SL3TL scheme” and “SL2TL scheme”.

3 The 2D equations.

3.1 Denotations for the 2D equations.

- $\mathbf{V}$ is the horizontal wind. Zonal and meridian components on computational grid are U and V.
- D is the horizontal wind divergence; ζ is the horizontal wind vorticity.
- Φ is the equivalent height. $\Phi_{\rm s}$ is the surface geopotential height (i.e. the orography). Φ^* is a reference equivalent height which is only used in the semi-implicit scheme and the linear model.
- $\mathbf{\Omega}$ is the Earth rotation angular velocity.
- ∇ is the first order horizontal gradient on η-surfaces.
- a is the Earth radius.
- $(\lambda_{\rm bne}, \theta_{\rm bne})$ are the longitude-latitude coordinates on a tilted and not stretched geometry, the tilting being the same as the one of the computational sphere.
- $\mathbf{k}$ is the unit vertical vector. One can write:

$$\mathbf{k} = \frac{\mathbf{r}}{\mid r \mid} = \frac{\mathbf{r}}{a}$$

3.2 The 2D shallow-water system of equations in spherical geometry.

3.2.1 Momentum equation.

Coriolis force can be treated explicitly ($\delta_{\mathbf{V}}$=0) or implicitly ($\delta_{\mathbf{V}}$=1) in the Lagrangian equation.

$$\frac{d\left(\mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r})\right)}{dt} = -2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V}) - \nabla\Phi \tag{8}$$

3.2.2 Continuity equation.

- **Conventional formulation.**

$$\frac{d(\Phi - (1-\delta_{\rm TR})\Phi_{\rm s})}{dt} = -(\Phi - \Phi_{\rm s})D + \delta_{\rm TR}\mathbf{V}\nabla(\Phi_{\rm s}) \tag{9}$$

- **Lagrangian formulation.**

$$\frac{d((\Phi - \Phi_{\rm s})J)}{dt} = 0 \tag{10}$$

J is a "Jacobian" quantity which matches:

$$\frac{dJ}{dt} = -JD \tag{11}$$

4 The 3D equations in spherical geometry (ARPEGE/IFS).

4.1 Denotations for the 3D equations.

- $\mathbf{V}$ is the horizontal wind. Zonal and meridian components on computational grid are U and V.
- D is the horizontal wind divergence; ζ is the horizontal wind vorticity.
- T is the temperature.
- q is the humidity.
- q_{r} is the rain.
- Π is the hydrostatic pressure; Π_{s} is the hydrostatic surface pressure.
- $\mathbf{\Omega}$ is the Earth rotation angular velocity.
- $(\lambda_{\mathrm{bne}}, \theta_{\mathrm{bne}})$ are the longitude-latitude coordinates on a tilted and not stretched geometry, the tilting being the same as the one of the computational sphere.
- (λ, θ) are the geographical longitude-latitude coordinates.
- (Λ, Θ) are the computational sphere longitude-latitude coordinates.
- w is the z-coordinate vertical velocity: $w = dz/dt$.
- $\omega = d\Pi/dt$ is the total temporal derivative of the hydrostatic pressure.
- gz is the geopotential height.
- Φ is the total geopotential. $\Phi = gz$ in the set of equations described.
- $\Phi_{\mathrm{s}} = gz_{\mathrm{s}}$ is the surface geopotential (i.e. the orography).
- $\mathbf{r}$ is the vector directed upwards, the length of which is the Earth radius.
- a is the average Earth radius near the surface.
- $\mathbf{i}$ (resp. $\mathbf{j}$) is the unit zonal (resp. meridian) vector on the Gaussian grid.
- $\mathbf{k}$ is the unit vertical vector. One can write $\mathbf{k} = \mathbf{r}/a$.
- g is the gravity acceleration constant, assumed to be vertically constant.
- R is the gas constant for air and R_{d} the gas constant for dry air.
- R_{a} is a generic denotation used in definition of vertical divergence, which may be R_{d} or R.
- c_{p} and $c_{\mathrm{p\,d}}$ are respectively the specific heat at constant pressure for moist air and dry air.
- c_{v} and $c_{\mathrm{v\,d}}$ are respectively the specific heat at constant volume for moist air and dry air.
- κ_{m} is R/c_{p}; κ_{d} is $R_{\mathrm{d}}/c_{\mathrm{p\,d}}$.
- ∇ is the first order horizontal gradient on η-surfaces.
- α_{T} is a vertical-dependent coefficient used to define a thermodynamic variable $T + \delta_{\mathrm{TR}} \frac{\alpha_{\mathrm{T}} \Phi_{\mathrm{s}}}{R_{\mathrm{d}} T_{\mathrm{st}}}$ less sensitive to orography than temperature T. Expression of α_{T} is:

$$\alpha_{\mathrm{T}} = B \left(-\frac{R_{\mathrm{d}}}{g} \left[\frac{dT}{dz} \right]_{\mathrm{st}} \right) T_{\mathrm{st}} \left(\frac{\Pi_{\mathrm{st}}}{\Pi_{\mathrm{sst}}} \right)^{\left(-\frac{R_{\mathrm{d}}}{g} \left[\frac{dT}{dz} \right]_{\mathrm{st}} - 1 \right)} \tag{12}$$

  where B is a vertically dependent and horizontally constant quantity which defines the vertical hybrid coordinate (see later paragraph "Definition of the vertical coordinate η", subsection (6.1)): B varies from 1 to 0 from bottom to top. Subscript "st" stands for "standard atmosphere".
- ρ is the mass per volume unit of air.
- M is the mapping factor; $\overline{M}$ is a reference mapping factor for the semi-implicit scheme.
- τ, γ, ν are linear operators used in the semi-implicit scheme; $\mathbf{S}^*$, $\mathbf{G}^*$, $\mathbf{N}^*$ are their adimensioned versions. For more details, see documentation (IDSI) about semi-implicit scheme.
- T^* is a vertically-constant reference temperature which is used in the semi-implicit scheme.
- T_{st} is the reference standard atmosphere surface temperature (288.15 K). $\left[\frac{dT}{dz} \right]_{\mathrm{st}}$ is the standard atmosphere tropospheric gradient of temperature (-0.0065 K/m).
- Π^* is a reference hydrostatic pressure and ${\Pi_{\mathrm{s}}}^*$ is a reference hydrostatic surface pressure, which are used in the semi-implicit scheme. These reference quantities are vertically dependent and "horizontally" (i.e. on η surfaces) constant. $\Delta\Pi^*$ are layer depths corresponding to a surface hydrostatic pressure equal to ${\Pi_{\mathrm{s}}}^*$.
- Π_{sst} is a reference hydrostatic pressure equal to the surface pressure of the standard atmosphere (101325 Pa, variable **VP00**). Π_{st} is a reference hydrostatic pressure defined at full levels and half levels corresponding to the surface reference hydrostatic pressure Π_{sst} (stored in array **STPRE**).
- $\Delta\Phi^*$ is a reference geopotential depth computed on model layers. $\Delta\Phi^*$ is vertically dependent and "horizontally" (i.e. on η surfaces) constant.

Additional quantities for non-hydrostatic models:

- p is the pressure, $p_{\rm s}$ is the surface pressure.
- $\tilde{T}$ is a modified temperature used in the NHQE model. $\tilde{T} = T\exp(-\kappa_{\rm m}\log(p/\Pi))$.
- D_3 is the 3D divergence used in the NH model. Its expression is given by equation (21).
- $\mathbf{L}^*$ is a linear operator used in the NHEE semi-implicit scheme (see documentation (IDSI)).
- $\mathbf{L}^*_\kappa$ is a linear operator used in the NHQE semi-implicit scheme (see documentation (IDSI)).
- $T^*_{\rm a}$ is a cold vertically-constant reference temperature which is used in the semi-implicit scheme in the NH vertical divergence equation; it is recommended to have $T^*_{\rm a}$ lower than the current temperature.

4.2 The 3D primitive equation model.

4.2.1 Momentum equation.

Vectorial form of momentum equation is used. Coriolis force can be treated explicitly ($\delta_{\mathbf{V}}$=0) or implicitly ($\delta_{\mathbf{V}}$=1).

$$\frac{d\left(\mathbf{V}+\delta_{\mathbf{V}}(2\mathbf{\Omega}\wedge\mathbf{r})\right)}{dt} = -2(1-\delta_{\mathbf{V}})(\mathbf{\Omega}\wedge\mathbf{V}) - \nabla\Phi - RT\nabla(\log\Pi) + \mathbf{F_V} \tag{13}$$

$\mathbf{F_V}$ is the physical contribution on horizontal wind.

4.2.2 Thermodynamic equation.

$$\frac{d\left(T+\delta_{\rm TR}\frac{\alpha_{\rm T}\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right)}{dt} = \frac{d\left(\delta_{\rm TR}\frac{\alpha_{\rm T}\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right)}{dt} + \frac{RT}{c_{\rm p}}\frac{\omega}{\Pi} + F_{\rm T} \tag{14}$$

$F_{\rm T}$ is the physical contribution on temperature. When $\delta_{\rm TR}=1$ the Eulerian treatment of orography is applied and the prognostic variable is replaced by one variable less sensitive to the surface orography. This modification has been proposed by Ritchie and Tanguay (1996). See equation (12) for definition of $\alpha_{\rm T}$. Term $\frac{d\left(\delta_{\rm TR}\frac{\alpha_{\rm T}\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right)}{dt}$ only contains advection terms linked to horizontal variations of orography and vertical variations of the coefficient $\alpha_{\rm T}$.

4.2.3 Continuity equation.

The equation which is discretised is the vertically integrated Lagrangian formulation of continuity equation.

$$\int_{\eta=0}^{\eta=1}\frac{\partial B}{\partial\eta}\frac{d\left[\log\Pi_{\rm s}+\delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right]}{dt}d\eta =$$

$$\int_{\eta=0}^{\eta=1}\frac{\partial B}{\partial\eta}\left[-\frac{1}{\Pi_{\rm s}}\int_{\eta=0}^{\eta=1}\nabla\left(\mathbf{V}\frac{\partial\Pi}{\partial\eta}\right)d\eta + \mathbf{V}\nabla\left(\log\Pi_{\rm s}+\delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right) - \frac{1}{\Pi_{\rm s}}\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=1} + \frac{1}{\Pi_{\rm s}}\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=0} - \frac{1}{\Pi_{\rm s}}g\left[F_{\rm m}\right]_{\eta=1}\right]d\eta \tag{15}$$

Variable $\delta_{\rm TR}$ is 0 or 1; when $\delta_{\rm TR}=1$ the new variable is less sensitive to the orography (new variable proposed by Ritchie and Tanguay (1996) to reduce orographic resonance).
If one assumes that a volume of air occupied by rainfall drops is not replaced by dry air when drops are falling (case $\delta m=0$, variable **NDPSFI** is 0 in **NAMPHY**), $F_{\rm m}$ is replaced by zero and $\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=1}$ is equal to zero.
If one assumes that a volume of air occupied by rainfall drops is replaced by dry air when drops are falling (case $\delta m=1$, variable **NDPSFI** is 1 in **NAMPHY**), $F_{\rm m}$ (diabatic flux) has to be taken in account and $\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=1}$ is non-zero (more details are given in documentation (IDEUL)).
$\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=0}$ is non-zero only when there is an upper radiative boundary condition (**LRUBC**=.TRUE.).

4.2.4 Advectable GFL variables.

Equation is written for moisture q, and is the same for the other advectable GFL variables (for example ozone, liquid water, ice, cloud fraction, TKE, aerosols and extra GFL variables).

$$\frac{dq}{dt} = F_{\rm q} \tag{16}$$

$F_{\rm q}$ is the physical contribution on moisture.

4.2.5 Non advectable pseudo-historic GFL variables:

Equation is written for rain $q_{\rm r}$, and is the same for the other non-advectable GFL variables. Since there is no advection, the SL equation is identical to the Eulerian equation.

$$\frac{\partial q_{\rm r}}{\partial t} = F_{q_{\rm r}} \tag{17}$$

4.3 The 3D NHEE fully elastic non-hydrostatic model.

4.3.1 Basics about NHEE equations.

The system of equations uses an idea of Laprise (1992) to be consistent with the hydrostatic equations: pressure p is split into an hydrostatic contribution (hydrostatic pressure Π) and an anhydrostatic departure ($p-\Pi$). Π is still used as an independent prognostic variable. This way of treating the non-hydrostatic equations is compatible with the use of a terrain-following vertical coordinate (η). For **NPDVAR**=2 and **NVDVAR**=3 the non-hydrostatic system of equations includes two additional prognostic variables: a variable $\hat{Q}$ linked to the pressure departure (its expression is $\hat{Q} = \log \frac{p}{\Pi}$), and a vertical divergence d. Details about these equations (choice of the two additional prognostic variables, discretisations, linearisation for semi-implicit scheme) can be found in (Bubnová et al., 1995), (Geleyn and Bubnová, 1995), (IDNHPB), (IDVNH1), (IDVHN2) and (IDVNH3). For the vertical divergence equation, it is also possible to use the prognostic variable:

$$d_4 = d + \mathtt{X}$$

where:

$$\mathtt{X} = \frac{p}{\frac{\partial \Pi}{\partial \eta} RT} \nabla \Phi \left(\frac{\partial \mathbf{V}}{\partial \eta} \right)$$

4.3.2 Momentum equation.

Vectorial form of momentum equation is used. Coriolis force can be treated explicitly ($\delta_{\mathbf{V}}$=0) or implicitly ($\delta_{\mathbf{V}}$=1).

$$\frac{d\left(\mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r})\right)}{dt} = \left[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})\right] - \frac{\partial p}{\partial \Pi} \nabla \Phi - RT \frac{\nabla(p)}{p} + \mathbf{F}_{\mathbf{V}} \tag{18}$$

The pressure gradient term is modified compared to the hydrostatic case. One can see that both non-hydrostatic pressure p and hydrostatic pressure Π are used in the RHS (right hand side) of equation (18). Term $\frac{\nabla(p)}{p}$ replaces $\nabla(\log p)$, the discretisation of this term is no longer a discretisation of $\nabla(\log p)$ and is slightly more complicated (more details are given in documentation (IDEUL)).

4.3.3 Thermodynamic equation.

The RHS of equation (19) can be written according to the two following forms:

$$\frac{dT}{dt} = \frac{RT}{c_{\mathrm{p}}} \frac{\left(\frac{dp}{dt}\right)}{p} + F_{\mathrm{T}} \tag{19}$$

$$\frac{dT}{dt} = -\frac{RT}{c_{\mathrm{v}}} D_3 + \left[\frac{c_{\mathrm{p}}}{c_{\mathrm{v}}} F_{\mathrm{T}}\right] \tag{20}$$

Expression of D_3 is:

$$D_3 = \nabla \mathbf{V} + \frac{p}{\frac{\partial \Pi}{\partial \eta} RT} \nabla \Phi \left(\frac{\partial \mathbf{V}}{\partial \eta} \right) - \frac{gp}{\frac{\partial \Pi}{\partial \eta} RT} \left(\frac{\partial w}{\partial \eta} \right) \tag{21}$$

Alternate form of equation (20) with Eulerian treatment of orography is:

$$\frac{d\left(T + \delta_{\mathrm{TR}} \frac{\alpha_{\mathrm{T}} \Phi_{\mathrm{s}}}{R_{\mathrm{d}} T_{\mathrm{st}}}\right)}{dt} = \frac{d\left(\delta_{\mathrm{TR}} \frac{\alpha_{\mathrm{T}} \Phi_{\mathrm{s}}}{R_{\mathrm{d}} T_{\mathrm{st}}}\right)}{dt} - \frac{RT}{c_{\mathrm{v}}} D_3 + \left[\frac{c_{\mathrm{p}}}{c_{\mathrm{v}}} F_{\mathrm{T}}\right] \tag{22}$$

See equation (12) for definition of α_{T}.

4.3.4 Vertical velocity w and vertical divergence d equations.

∗ **Using d as prognostic variable:** The relationship between d and w writes:

$$d = -\frac{gp}{R_{\mathrm{a}} T \frac{\partial \Pi}{\partial \eta}} \left(\frac{\partial w}{\partial \eta} \right) \tag{23}$$

Equation of d is:

$$\frac{dd}{dt} = -dD_3 + d\nabla \mathbf{V} - \frac{gp}{R_{\mathrm{a}} T \frac{\partial \Pi}{\partial \eta}} \frac{\partial \left[\frac{dw}{dt}\right]_{\mathrm{ad}}}{\partial \eta} + \frac{gp}{R_{\mathrm{a}} T \frac{\partial \Pi}{\partial \eta}} (\nabla w) \left(\frac{\partial \mathbf{V}}{\partial \eta} \right) + F_{\mathrm{d}} \tag{24}$$

The physical contribution of d is linked to $F_{\rm w}$ which is the physical contribution of w, and also to $F_{\rm m}^{'}$ (continuity equation for **NDPSFI**=1) by the following relationship:

$$F_{\rm d} = -\frac{d}{\left[\frac{\partial \Pi}{\partial \eta}\right]} F_{\rm m}^{'} - \left[\frac{gp}{R_{\rm a} T \frac{\partial \Pi}{\partial \eta}} \frac{\partial F_{\rm w}}{\partial \eta}\right] \tag{25}$$

One can notice that the RHS of equation (24) contains the term $\frac{\partial \left[\frac{dw}{dt}\right]_{\rm ad}}{\partial \eta}$ which requires the computation of $\frac{dw}{dt}$ (at least its adiabatic part).

Equation of w is:

$$\frac{dw}{dt} = g\frac{\partial (p-\Pi)}{\partial \Pi} + F_{\rm w} \tag{26}$$

At the surface we have to use another equation (which requires some assumptions about the surface wind):

$$g\frac{dw_{\rm surf}}{dt} = \frac{d\mathbf{V}_{\rm surf}}{dt}\nabla\Phi_{\rm s} + \mathbf{V}_{\rm surf}\left[\mathbf{V}_{\rm surf}\nabla(\nabla\Phi_{\rm s})\right] + F_{\rm w_{surf}} \tag{27}$$

Value of term $\frac{d\mathbf{V}_{\rm surf}}{dt}\nabla\Phi_{\rm s} + \mathbf{V}_{\rm surf}\left[\mathbf{V}_{\rm surf}\nabla(\nabla\Phi_{\rm s})\right]$ contains the horizontal second derivatives of surface orography and Coriolis term (cf. equation (32) of Geleyn and Bubnová, 1995). Its calculation is rather tricky and is not detailed in this documentation (for more details, see documentation (IDEUL)).

Remark: an alternative formulation is to mix the w equation in the advection scheme and the d equation in the other parts of the code.

∗ **Using d_4 as prognostic variable:** Equation of d_4 is:

$$\frac{dd_4}{dt} = \frac{dd}{dt} + \frac{d\mathtt{X}}{dt} \tag{28}$$

where $\frac{d\mathtt{X}}{dt}$ is the Lagrangian total derivative of the quantity ($\mathtt{X} = \frac{p}{\frac{\partial \Pi}{\partial \eta} RT}\nabla\Phi\left(\frac{\partial \mathbf{V}}{\partial \eta}\right)$). The calculation of $\frac{d\mathtt{X}}{dt}$ is not done by a semi-Lagrangian temporal advection but simply by a diagnostic (see appendix 1).

4.3.5 Pressure departure variable $\hat{Q}$ equation.

Definition of $\hat{Q}$ is:

$$\hat{Q} = \log\frac{p}{\Pi} \tag{29}$$

Equation of $\hat{Q}$ is:

$$\frac{d\hat{Q}}{dt} = -\frac{c_{\rm p}}{c_{\rm v}} D_3 - \frac{\omega}{\Pi} + \frac{c_{\rm p}}{c_{\rm v} T} F_{\rm T} \tag{30}$$

where D_3 is defined by equation (21). The physical contribution of $\hat{Q}$ is linked to the physical contribution of T.

4.3.6 Continuity and GFL equations.

Continuity equation is unchanged compared to the hydrostatic case and only uses the hydrostatic pressure Π. GFL equations are unchanged compared to the hydrostatic case.

4.4 The 3D NHQE quasi-elastic non-hydrostatic model.

4.4.1 Basics about NHQE equations.

Compared to the NHEE system, the main differences are:

- Pressure departure is a diagnostic variable: quantity $\hat{Q} = \log\frac{p}{\Pi}$ is still used.
- Temperature equation uses the modified temperature $\tilde{T} = T\exp(-\kappa_{\rm m}\log(p/\Pi))$.
- Pressure gradient term and RHS of w equation have a different formulation and a different discretisation.
- Expression of $\mathtt{X}$ has two contributions $\mathtt{X}_{\rm D}$ and $\mathtt{X}_{\rm S}$: $\mathtt{X} = \mathtt{X}_{\rm D} + \mathtt{X}_{\rm S}$.

  $\mathtt{X}_{\rm D} = (1-\kappa_{\rm m})(\omega/\Pi + \mathbf{S}D)$, where $[\mathbf{S}D]_\eta = \frac{1}{\Pi_\eta}\int_{top}^{\eta}\frac{\partial \Pi}{\partial \eta} D d\eta'$.

  $\mathtt{X}_{\rm S} = \frac{\Pi}{R\tilde{T}}\nabla\Phi\frac{\partial \mathbf{V}}{\partial \Pi}$, where $\mathbf{S}$ is the non-linear counterpart of linear operator $\mathbf{S}^*$.
- 3D divergence D_3 writes $D_3 = D + d + \mathtt{X}_{\rm S}$ or $D_3 = D + d_4 - \mathtt{X}_{\rm D}$.
- The linear system is different and uses a closure equation.

For the vertical divergence equation, it is still possible to use the prognostic variable $d_4 = d + \mathtt{X}$ and this is actually the only choice which allows to invert the linear system.

4.4.2 Momentum equation.

Vectorial form of momentum equation is used. Coriolis force can be treated explicitly ($\delta_{\mathbf{V}}$=0) or implicitly ($\delta_{\mathbf{V}}$=1).

$$\frac{d\left(\mathbf{V}+\delta_{\mathbf{V}}(2\mathbf{\Omega}\wedge\mathbf{r})\right)}{dt}=[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega}\wedge\mathbf{V})]-\exp(\kappa_{\mathrm{m}}\hat{Q})\left(1+\Pi\frac{\partial\hat{Q}}{\partial\Pi}\right)\nabla\Phi-R\tilde{T}\exp(\kappa_{\mathrm{m}}\hat{Q})\left(\frac{\nabla\Pi}{\Pi}+\nabla\hat{Q}\right)+\mathbf{F}_{\mathbf{V}} \tag{31}$$

4.4.3 Thermodynamic equation.

$$\frac{d\tilde{T}}{dt}=\frac{R\tilde{T}}{c_{\mathrm{p}}}\frac{\omega}{\Pi}+\exp(-\kappa_{\mathrm{m}}\hat{Q}))F_{\mathrm{T}} \tag{32}$$

Alternate form of equation (32) with Eulerian treatment of orography is:

$$\frac{d\left(\tilde{T}+\delta_{\mathrm{TR}}\frac{\alpha_{\mathrm{T}}\Phi_{\mathrm{s}}}{R_{\mathrm{d}}T_{\mathrm{st}}}\right)}{dt}=\frac{d\left(\delta_{\mathrm{TR}}\frac{\alpha_{\mathrm{T}}\Phi_{\mathrm{s}}}{R_{\mathrm{d}}T_{\mathrm{st}}}\right)}{dt}\frac{R\tilde{T}}{c_{\mathrm{p}}}\frac{\omega}{\Pi}+\exp(-\kappa_{\mathrm{m}}\hat{Q}))F_{\mathrm{T}} \tag{33}$$

See equation (12) for definition of α_{T}.

4.4.4 Vertical velocity w and vertical divergence d equations.

∗ **Using d as prognostic variable:** The relationship between d and w writes:

$$d=-\frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}}\left(\frac{\partial w}{\partial\eta}\right) \tag{34}$$

Equation of d is:

$$\frac{dd}{dt}=-d(d+\mathtt{X}_{\mathrm{S}})-\frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}}\frac{\partial\left[\frac{dw}{dt}\right]_{\mathrm{ad}}}{\partial\eta}+\frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}}(\nabla w)\left(\frac{\partial\mathbf{V}}{\partial\eta}\right)+F_{\mathrm{d}} \tag{35}$$

Equation of w is:

$$\frac{dw}{dt}=\frac{g}{\kappa_{\mathrm{m}}}\left[\frac{\partial\Pi(\exp(\kappa_{\mathrm{m}}\hat{Q})-1)}{\partial\Pi}+(\kappa_{\mathrm{m}}-1)(\exp(\kappa_{\mathrm{m}}\hat{Q})-1)\right]+F_{\mathrm{w}} \tag{36}$$

At the surface we can still use equation 27.

Remark: an alternative formulation is to mix the w equation in the advection scheme and the d equation in the other parts of the code.

∗ **Using d_4 as prognostic variable:** Equation of d_4 is:

$$\frac{dd_4}{dt}=\frac{dd}{dt}+\frac{d\mathtt{X}}{dt} \tag{37}$$

where $\frac{d\mathtt{X}}{dt}$ is the Lagrangian total derivative of the quantity $\mathtt{X}$. The calculation of $\frac{d\mathtt{X}}{dt}$ is not done by a semi-Lagrangian temporal advection but simply by a diagnostic (see appendix 1).

4.4.5 Continuity and GFL equations.

Continuity equation is unchanged compared to the hydrostatic case and only uses the hydrostatic pressure Π. GFL equations are unchanged compared to the hydrostatic case.

5 Discretisation of the equations: general aspects.

This section does not describe in detail iterative centred-implicit schemes and describes only non-iterative schemes in detail. For more details about iterative centred-implicit schemes, see documentation (IDSI).

5.1 Denotations.

∗ Upper index:

- First integration step: + (resp. m, o, $-$) for $t+\Delta t$ (resp. $t+0.5\Delta t$, t, t) quantity.
- Following integration steps: + (resp. m, o, $-$) for $t+\Delta t$ (resp. $t+0.5\Delta t$, t, $t-\Delta t$) quantity.

∗ Lower index: F (resp. M and O) for final (resp. medium and origin) point.

∗ Particular case of the first timestep in a SL3TL scheme: Written discretisations are valid from the second integration step. Δt has to be replaced by $0.5\Delta t$ for the first integration step (in this case the $t-\Delta t$ quantities are equal to the t quantities).

∗ The different classes of prognostic variables: Prognostic variables can be split into different classes:

- 3D variables, the equation RHS of which has a non-zero adiabatic contribution and a non-zero semi-implicit correction contribution. They are called "GMV" in the code ("GMV" means "grid-point model variables"). This class of variables includes wind components, temperature (and two additional variables in a non-hydrostatic model). The sub-class of thermodynamic variables includes T, and two additional variables in a non-hydrostatic model. There are **NFTHER** thermodynamic variables.
- 3D advectable "conservative" variables. The equation RHS of these variables has a zero adiabatic contribution, only the diabatic contribution (and the horizontal diffusion contribution) can be non-zero. They are called "GFL" in the code ("GFL" means "grid-point fields"). This class of variables includes for example humidity, liquid water, ice, cloud fraction, ozone, TKE, aerosols, and some extra fields. See documentation (IDEUL) for a comprehensive list of advectable GFL.
- 3D non advectable pseudo-historic variables. The equation RHS of these variables looks like the one of the 3D advectable "conservative" variables, but there is no advection. They are included in the GFL variables. This class of variables includes for example rain, snow, graupels, hail, convective precipitation flux, stratiform precipitation flux. See documentation (IDEUL) for a comprehensive list of non advectable pseudo-historic GFL.
- 2D variables, the equation RHS of which mixes 3D and 2D terms, has a non-zero adiabatic contribution and a non-zero semi-implicit correction contribution. They are called "GMVS" in the code ("GMVS" means "grid-point model variables for surface"). This class of variables includes the logarithm of surface pressure (continuity equation).

5.2 Generalised formulation of extrapolations (SL2TL schemes).

In theory, the RHS in SL2TL schemes would require to evaluate quantities at $t+0.5\Delta t$. This is replaced by an extrapolation using quantities at t and $t-\Delta t$. The simplest way to extrapolate is the non extrapolating formulation (replace $X(t+0.5\Delta t)$ by $X(t)$).

One writes a general formulation, including non extrapolating formulation (NESC), stable extrapolation (SETTLS), mixed "settls-nesc" extrapolation (MIXETTLS), conventional extrapolation, and formulation of (Gospodinov et al., 2001). For a quantity X (X stands for example for $\mathcal{A}-\beta\mathcal{L}$, $\mathbf{V}/a$ or $\dot{\eta}$), and if $\tilde{X}$ is an extrapolated value of X which represents the value of X at $t+0.5\Delta t$ and at location M (half location between O and F), $\tilde{X}$ writes:

$$\tilde{X}=[0.5X_F^o+0.5X_O^o]+C_\gamma[0.5X_O^o-0.5X_O^-]+C_\gamma(0.25-C_\alpha)[(X_F^o-X_F^-)-(X_O^o-X_O^-)]$$

- C_γ=0 yields the non-extrapolated way to compute $\tilde{X}$ (NESC).
- C_γ=1, C_α=0.25 yields the stable extrapolation (SETTLS).
- $0<C_\gamma<1$, C_α=0.25 yields the mixed extrapolation (MIXETTLS).
- C_γ=1, C_α=0 yields the conventional extrapolation.

For more details about stable extrapolation, see (Hortal, 1998), (Hortal, 2002).

Discretisations below will be given for conventional and stable (SETTLS) extrapolations. For non-extrapolating formulation (NESC), start from formulations with stable extrapolations; replace $[\mathcal{A}^o-\beta\mathcal{L}^o]_O-0.5[\mathcal{A}^--\beta\mathcal{L}^-]_O$ by $0.5[\mathcal{A}^o-\beta\mathcal{L}^o]_O$. Non-extrapolating formulation is generally used in the predictor step when an ICI (PC) scheme is switched on. See documentation (IDSI) for more details.

5.3 Uncentering coefficients.

Two kinds of uncentering coefficients have been introduced in the code in the 1990s.

- First order uncentering: it is applied to both non-linear terms and linear terms in the RHS of equations. It is implemented for both SL3TL and SL2TL advections.
  - multiply $\mathcal{A}$ and $\beta\mathcal{L}$ evaluated at origin point O by $(1-\epsilon)$.
  - multiply $\mathcal{A}$ and $\beta\mathcal{L}$ evaluated at final point F by $(1+\epsilon)$.
  - no change for X and term containing $\mathcal{F}$.
- Pseudo-second order uncentering: it is applied only to linear terms containing $\beta\mathcal{L}$. It is implemented for SL2TL advection. For more details see part 5 (equations (37) and (38)) of (Simmons and Temperton, 1996). After calculations not detailed there, one can show that it is equivalent to replace β by $(1+\epsilon_{\mathcal{X}})\beta$.

Uncentering allows to remove the noise due to gravity waves (orographic resonance). SL2TL discretisations below will be given with second order uncentering. SL3TL discretisations below will be given with first order uncentering.

5.4 Discretisation for a 3D variable in a 3D model: general case where the RHS has non-zero linear and non-linear terms (GMV variables).

5.4.1 Introduction and preliminary remarks.

List of equations:

- Momentum equation.
- Temperature equation.
- Pressure departure variable (NHEE model only).
- Vertical divergence variable (non-hydrostatic models only).

Generic notation **N(X)LAG** stands for:

- **NWLAG** for momentum equation.
- **NTLAG** for temperature equation.
- **NSPDLAG** for pressure departure variable (NHEE model only).
- **NSVDLAG** for vertical divergence variable (non-hydrostatic models only).

Generic notation **P(X)L0**, **P(X)L9**, **P(X)T1** stands for:

- **PUL0**, **PUL9**, **PUT1** for U-momentum equation.
- **PVL0**, **PVL9**, **PVT1** for V-momentum equation.
- **PTL0**, **PTL9**, **PTT1** for temperature equation.
- **PSPDL0**, **PSPDL9**, **PSPDT1** for $\hat{Q}$ equation.
- **PSVDL0**, **PSVDL9**, **PSVDT1** for d equation.

Generic notation **P(X)NLT9** stands for:

- **PUNLT9** for U-momentum equation.
- **PVNLT9** for V-momentum equation.
- **PTNLT9** for temperature equation.
- **PSPDNLT9** for $\hat{Q}$ equation.
- **PSVDNLT9** for d equation.

Generic notation for total term, linear term, non linear term, physics: $\mathcal{A}$ is the total term (sum of dynamical contributions), $\mathcal{L}$ is the linear term (treated in the semi-implicit scheme), the difference $\mathcal{A}-\beta\mathcal{L}$ is the non-linear term. $\mathcal{F}$ is the sum of contributions computed in the physical parameterizations.

Description stands for the general case where linear and non-linear terms are gathered in the same buffer, and where no additional splitting is required to do diagnostics or to apply SLHD interpolations to a subset of the terms interpolated by a high-order interpolation. In some particular cases, additional splitting involving separate buffers may be required:

- In the NH models for options where linear terms must be separately interpolated (controlled by variables **LSLINL** and **LSLINLC2**). Linear terms are stored in arrays, the name of them has appendix **_SI**, or in parts of **PB1** with pointers, the name of them has appendix **_SI**.
- Splitting between linear and non-linear terms may have to be done for some DDH diagnostics too (if **LRSIDDH**=T).

- When some diagnostics (for example DDH) impose that evaluation of the dynamics and of the physics at the origin point of the SL trajectory must be done separately in two different buffers (controlled by variable **NSPLTHOI** set to -1). Buffer **P(X)LF9** is used instead of **P(X)L9** to store quantities, the interpolation of which is a non-SLHD high-order one.
- When SLHD interpolations, if switched on, are applied only to a subset of the quantities interpolated by a high order interpolation (applied to $X(t)$ or $X(t-\Delta t)$ but not to the other terms); in particular physics does not use a SLHD interpolation in this case (controlled by variable **NSPLTHOI** set to 1). Buffer **P(X)LF9** is used instead of **P(X)L9** to store quantities, the interpolation of which is a non-SLHD high-order one.

Only case **N(X)LAG**=3 is described in detail, and content of **P(X)L0**, **P(X)L9** and **P(X)T1** given below is valid for **N(X)LAG**=3; code also contains cases **N(X)LAG**=2 and **N(X)LAG**=4 for a subset of options:

- **N(X)LAG**=2: **P(X)L0** is not used; **P(X)L9** contains the sum of what is spread among **P(X)L9** and **P(X)L0** when **N(X)LAG**=3, and is used for high-order interpolation at the origin point O.
- **N(X)LAG**=4: in this case diabatic terms are interpolated by trilinear interpolations and they enter the buffer containing terms interpolated by trilinear interpolations. Start from discretisations given for **N(X)LAG**=3 and move term $\Delta t\mathcal{F}$ from **P(X)L9** to **P(X)L0**.

High-order interpolations: in the following discretisations, "high-order interpolations" means 32 points interpolations for 3D terms (vertical interpolations are cubic), 12 points interpolations for 2D terms.

Vectorial variables: the following discretisations are written for scalar variables. For vectorial variables (for example the horizontal wind) a rotation operator $\mathcal{R}$ has to be applied from interpolation point to final point:

- expression interpolated at O has to be replaced by $\mathcal{R}^{OF}\{\text{this expression}\}_O$.
- expression interpolated at M has to be replaced by $\mathcal{R}^{MF}\{\text{this expression}\}_M$.

5.4.2 3TL vertical interpolating SL scheme.

Equation

$$\frac{dX}{dt} = \mathcal{A} + \mathcal{F} \tag{38}$$

is discretised as follows:

$$(X - (1+\epsilon)\Delta t\beta\mathcal{L})_F^+ = \{X^- + (1-\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o + [(1-\epsilon)\Delta t\beta\mathcal{L} + 2\Delta t\mathcal{F}]^-\}_O + \{(1+\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o\}_F \tag{39}$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X)L0**: $(1-\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o + [(1-\epsilon)\Delta t\beta\mathcal{L}]^-$ for trilinear interpolation at O.
- **P(X)L9**: $X^- + [2\Delta t\mathcal{F}]^-$ for high-order interpolation at O.
- **P(X)T1**: $(1+\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o$ then provisional add of quantity $[(1+\epsilon)\Delta t\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.

5.4.3 2TL vertical interpolating SL scheme: conventional extrapolation.

The $t+0.5\Delta t$ non-linear term $\mathcal{A}^m - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^m$ used in the SL2TL scheme is replaced by a linear temporal extrapolation using the t and $t-\Delta t$ quantities at the same location. At the first time integration step, values at $t+0.5\Delta t$ are set equal to initial values. This discretisation of the SL2TL scheme follows (Mc Donald and Haugen, 1992). Quantity $\mathcal{A}^o - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^o$ has to be saved in a buffer **P(X)NLT9** to be available as $\mathcal{A}^- - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^-$ for the following timestep.
Equation (38) is discretised as follows:

$$(X - (1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L})_F^+ = \{X^o + \frac{\Delta t}{2}[\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^m + [(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L} + \Delta t\mathcal{F}]^o\}_O + \{\frac{\Delta t}{2}[\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^m\}_F \tag{40}$$

which can be rewritten, once expanded the extrapolation:

$$\begin{aligned}(X - (1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L})_F^+ = {} & [X^o + \tfrac{\Delta t}{2}\mathcal{A}^o + \Delta t\mathcal{F}^o]_O + 0.5\tfrac{\Delta t}{2}[(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - (\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]_O \\ & + \tfrac{\Delta t}{2}[1.5(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - 0.5(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]_F\end{aligned} \tag{41}$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X)L0**: $\frac{\Delta t}{2}\mathcal{A}^o + 0.5\frac{\Delta t}{2}[(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - (\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]$ for trilinear interpolation at O.
- **P(X)L9**: $X^o + [\Delta t\mathcal{F}]^o$ for high-order interpolation at O.
- **P(X)T1**: $\frac{\Delta t}{2}[1.5(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - 0.5(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]$ then provisional add of quantity $[(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.

5.4.4 2TL vertical interpolating SL scheme: stable extrapolation and mixed extrapolation.

At the first time integration step, values at $t + 0.5\Delta t$ are set equal to initial values. Quantity $\mathcal{A}^o - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^o$ has to be saved in a buffer **P(X)NLT9** to be available as $\mathcal{A}^- - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^-$ for the following timestep.
Equation (38) is discretised as follows:

$$(X - (1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L})_F^+ = [X^o + \tfrac{\Delta t}{2}\mathcal{A}^o + \Delta t\mathcal{F}^o]_O$$
$$+C_\gamma \tfrac{\Delta t}{2}[(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - (\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]_O + [\tfrac{\Delta t}{2}(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o]_F \quad (42)$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X)L0**: $\frac{\Delta t}{2}\mathcal{A}^o + C_\gamma\frac{\Delta t}{2}[(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o - (\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]$ for trilinear interpolation at O.
- **P(X)L9**: $X^o + [\Delta t\mathcal{F}]^o$ for high-order interpolation at O.
- **P(X)T1**: $\frac{\Delta t}{2}(\mathcal{A} - (1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o$ then provisional add of quantity $[(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.

5.4.5 Specific treatment for some options in the vertical divergence equation.

∗ **Option LGWADV=.T. .** When this option is activated, the SL scheme treats the Lagrangian equation of w instead of the one of d (or d_4). That implies the following steps in the code:

- Change of variable from d or d_4 to w.
- SL explicit treatment of the w equation. For finite difference vertical discretisation (**LVFE_GW**=F) w is given at half levels: that means that the SL trajectory must be computed also for trajectories ending at half levels. For finite element vertical discretisation (**LVFE_GW**=T) w is given at full levels. There is a specific treatment for $w_{\rm surf}$ (diagnostic condition at $t + \Delta t$).
- Calculation of the linear terms of the d equation for the semi-implicit scheme.
- Conversion from w^+ into d^+ or ${d_4}^+$. If **LVFE_GW**=T, this conversion is done by applying the vertical derivation to the temporal increment of w to obtain the temporal increment of d.
- Add the linear terms to d^+ or ${d_4}^+$.

It is necessary to keep the prognostic variable d or d_4 in the linear model: w in the linear model would lead to linear instabilities.

This option allows to remove spurious chimneys above slopes, and also spurious noise in some "bubble" tests.

More details about dataflow are given in appendix 2.

∗ **Option NVDVAR=4.** In this case we must discretise the term $\frac{d\mathtt{X}}{dt}$, and this is not done by a SL treatment of the equation of $\mathtt{X}$ (the RHS of this equation is not easy to compute).
If there is no predictor-corrector scheme activated, this term is diagnosed as follows:

- Three-time level semi-Lagrangian scheme for **ND4SYS**=1: $d\mathtt{X}/dt = [0.5\mathtt{X}_F^o + 0.5\mathtt{X}_O^o - \mathtt{X}_O^-]/[\Delta t]$.
- Two-time level semi-Lagrangian scheme for **ND4SYS**=1: $d\mathtt{X}/dt = [\mathtt{X}_M^m - \mathtt{X}_O^o]/[0.5\Delta t]$.
  $\mathtt{X}_M^m$ is a symbolic denotation for the extrapolated $t + 0.5\Delta t$ value of $\mathtt{X}$. The way of extrapolating depends on **LSETTLS** and **LNESC** and is the same as for the other terms evaluated at $t + 0.5\Delta t$ (there is no interpolation at M).
- Three-time level semi-Lagrangian scheme for **ND4SYS**=2: $d\mathtt{X}/dt = [\mathtt{X}_F^+ - \mathtt{X}_F^o]/[\Delta t] + [\mathtt{X}_F^o - \mathtt{X}_O^o]/[\Delta t]$.
  $\mathtt{X}_F^+$ is computed using a mix of t quantities (horizontal derivatives) and provisional $t + \Delta t$ variables.
- Two-time level semi-Lagrangian scheme for **ND4SYS**=2: $d\mathtt{X}/dt = [\mathtt{X}_F^+ - \mathtt{X}_F^o]/[0.5\Delta t] + [\mathtt{X}_F^o - \mathtt{X}_O^o]/[0.5\Delta t]$.
  $\mathtt{X}_F^+$ is computed using a mix of t quantities (horizontal derivatives) and provisional $t + \Delta t$ variables.

More details are given in appendix 1.

∗ **Assumptions at the surface (velocities).** Calculation of $\frac{dd}{dt}$ for the layer $l = L$ requires the calculation of $\frac{dw_{\rm surf}}{dt}$. There are two options to compute $\frac{dw_{\rm surf}}{dt}$, controlled by variable **LRDBBC**.

- For **LRDBBC**=.F., one simply discretises equation (27) with a SL treatment. The RHS of this equation contains $\frac{d\mathbf{V}_{\rm surf}}{dt}$ and $\mathbf{V}_{\rm surf}$, and the assumptions currently done about these quantities are:
  . $\mathbf{V}_{\rm surf} = \mathbf{V}_{l=L}$
  . $\frac{d\mathbf{V}_{\rm surf}}{dt} = \left[\frac{d\mathbf{V}}{dt}\right]_{l=L,\rm adiab}$, with an explicit treatment of the Coriolis term, even if **LADVF**=.T. or **LIMPF**=.T. .
- An alternate discretisation (option **LRDBBC**=.T.) is to evaluate $dw_{\rm surf}/dt$ by:

– SL3TL: $dw_{\rm surf}/dt = (w_F^+ - w_O^-)/(2\Delta t)$

– SL2TL: $dw_{\rm surf}/dt = (w_F^+ - w_O^o)/(\Delta t)$

which requires additional interpolations to compute w_O.

This option allows to remove spurious chimneys above slopes.
w^+ is computed by:

$$w^+ = \mathbf{V}^+_{l=L,\rm prov} \nabla \Phi_{\rm s}$$

where $\mathbf{V}^+_{l=L,\rm prov}$ is the provisional $t+\Delta t$ value of $\mathbf{V}_{l=L}$ computed just after the interpolations (this is this provisional value which is used as input for the lagged physics).
The treatment of $\frac{dd}{dt}$ for the layer $l=L$ is done according to the following steps:

– Calculation of $\left[\frac{dd}{dt}\right]_{l=L,\rm lrdbbc=F}$ (as if **LRDBBC** were .F.), $\left[\frac{dw_{\rm surf}}{dt}\right]_{\rm lrdbbc=F}$ and $\left[\frac{dw_{\rm surf}}{dt}\right]_{\rm lrdbbc=T}$.

– Calculation of $\left[\frac{dd}{dt}\right]_{l=L,\rm lrdbbc=T}$ by the following formula:

$$\left[\frac{dd}{dt}\right]_{l=L,\rm lrdbbc=T} = \left[\frac{dd}{dt}\right]_{l=L,\rm lrdbbc=F} - \left(\left[\frac{dw_{\rm surf}}{dt}\right]_{\rm lrdbbc=T} - \left[\frac{dw_{\rm surf}}{dt}\right]_{\rm lrdbbc=F}\right)\left[\frac{gp}{R_{\rm a}T\Delta\Pi}\right]^o_{l=L,MM} \quad (43)$$

where MM is the decentered middle of O and F (that requires a specific interpolation of $\left[\frac{gp}{R_{\rm a}T\Delta\Pi}\right]_{l=L}$ at O).

For more details about these calculations and the additional interpolations required, see documentation (IDVNH3), especially the parts 1 to 3.

∗ **Numerical stability.**

- **NVDVAR**=4 is more stable than **NVDVAR**=3.
- **ND4SYS**=2 is more stable than **ND4SYS**=1.
- For refined horizontal resolution (Δx < 1km) we recommend to use **NVDVAR**=4, **LGWADV**=.T., **ND4SYS**=2.

5.5 Discretisation for a 3D variable in a 3D model: particular case where the RHS has zero linear and non-linear terms (advectable GFL variables).

5.5.1 Introduction and preliminary remarks.

List of equations: humidity equation, and for example:

- Ozone equation.
- Liquid water equation.
- Ice equation.
- Cloudiness equation.
- TKE.
- Aerosols equation.
- Extra GFL variables equations.

See documentation (IDEUL) for a comprehensive list of advectable GFL variables.

Generic notation **P(X)L9**, **P(X)T1** stands for:

- **PGFLL9**, **PGFLT1** for GFL variables.

Generic notation for total term, linear term, non linear term, physics: $\mathcal{A}$ is the total term (sum of dynamical contributions), $\mathcal{L}$ is the linear term (treated in the semi-implicit scheme), the difference $\mathcal{A}-\beta\mathcal{L}$ is the non-linear term. $\mathcal{F}$ is the sum of contributions computed in the physical parameterizations. In the present case $\mathcal{A}$ and $\mathcal{L}$ are equal to zero.

Description stands for the general case where diabatic and adiabatic terms are gathered in the same buffer, and where no additional splitting is required to do diagnostics or to apply SLHD interpolations to a subset of the terms interpolated by a high-order interpolation. In some particular cases, additional splitting involving separate buffers may be required:

- When some diagnostics (for example DDH) impose that evaluation of the dynamics and of the physics at the origin point of the SL trajectory must be done separately in two different buffers (controlled by variable **NSPLTHOI** set to -1). Buffer **P(X)LF9** is used instead of **P(X)L9** to store quantities, the interpolation of which is a non-SLHD high-order one.

- When diabatic terms use a different interpolation from the adiabatic one. That can be the case for non-zero values of **NSPLTHOI**, attribute **LTDIABLIN** set to T. Buffer **P(X)LF9** is used instead of **P(X)L9** to store diabatic terms to be interpolated.

High-order interpolations: in the following discretisations, "high-order interpolations" means: 32 points interpolations for 3D terms (vertical interpolations are cubic), 12 points interpolations for 2D terms. For ozone, vertical cubic interpolations can be replaced by vertical Hermite cubic interpolations (switch **YO3_NL%LHV** in **NAMGFL**), or vertical spline cubic interpolations (switch **YO3_NL%LVSPLIP** in **NAMGFL**).

5.5.2 3TL vertical interpolating SL scheme.

Equation (38) is discretised as follows:

$$X_F^+ = \{X^- + [2\Delta t\mathcal{F}]^-\}_O \tag{44}$$

Buffers content before interpolations:

- **P(X)L9**: $X^- + [2\Delta t\mathcal{F}]^-$ for high-order interpolation at O.
- **P(X)T1** contains zero; evaluated at the final point F.

5.5.3 2TL vertical interpolating SL scheme.

At the first time integration step, values at $t + 0.5\Delta t$ are set equal to initial values. This discretisation of the SL2TL scheme follows (Mc Donald and Haugen, 1992).

Equation (38) is discretised as follows:

$$X_F^+ = \{X^o + [\Delta t\mathcal{F}]^o\}_O \tag{45}$$

Buffers content before interpolations:

- **P(X)L9**: $X^o + [\Delta t\mathcal{F}]^o$ for high-order interpolation at O.
- **P(X)T1** contains zero; evaluated at the final point F.

$*$ **Remark for iterative centred-implicit schemes:** For options where this type of scheme involves the momentum equation (this is the case for the option **LPC_FULL**=.T.) X_F^+ has to be recomputed at all iterations of the iterative centred-implicit scheme since the origin point O is recomputed at each iteration.

5.5.4 Non advectable pseudo-historic GFL variables.

For these variables the discretisation always writes (SL3TL and SL2TL):

$$X_F^+ = \{X^o + [\Delta t\mathcal{F}]^o\}_F \tag{46}$$

and there are never interpolations.

5.6 Discretisation for a 2D variable in a 3D model (GMVS variables, for example continuity equation).

5.6.1 Introduction and preliminary remarks.

The equation which is now discretised is:

$$[\mathcal{R}_{\text{inte}}]_{(top,surf)} \left\langle \frac{\mathcal{W}_{\text{vei}}}{\Delta\eta}\frac{dX}{dt} \right\rangle = [\mathcal{R}_{\text{inte}}]_{(top,surf)} \left\langle \frac{\mathcal{W}_{\text{vei}}}{\Delta\eta}\mathcal{A} \right\rangle + [\mathcal{R}_{\text{inte}}]_{(top,surf)} \left\langle \frac{\mathcal{W}_{\text{vei}}}{\Delta\eta}\mathcal{F} \right\rangle \tag{47}$$

where:

$$[\mathcal{R}_{\text{inte}}]_{(top,surf)} \left\langle \frac{\mathcal{W}_{\text{vei}}}{\Delta\eta} \right\rangle = 1 \tag{48}$$

and $[\mathcal{R}_{\text{inte}}]_{(top,surf)}$ is the vertical integral matricial operator (the scalar product $[\mathcal{R}_{\text{inte}}]_{(top,surf)}\langle X\rangle$ is the discretisation of $\int_{\eta=0}^{\eta=1} X d\eta$, $\langle X\rangle$ is the vector containing the layer values of X: $(X_1; X_2; ...; X_l; ...; X_L)$).

In the set ot equations described in this documentation, expression of $\mathcal{W}_{\text{vei}}$ at full levels is:

$$[\mathcal{W}_{\text{vei}}]_l = \Delta B_l \tag{49}$$

When the finite element vertical discretisation is activated (**LVERTFE**=.T.), $[\mathcal{R}_{\rm inte}]_{(top,surf)}$ is a vector tricky to compute, it is precomputed in the setup code under **SUVERTFE** and stored in the array **RINTE** or **RINTBF11**; vertical integrations are done in routine **VERINT**.

When the finite difference vertical discretisation is activated (**LVERTFE**=.F.), $[\mathcal{R}_{\rm inte}]_{(top,surf)}$ is simply the vector of coordinates $([\Delta\eta]_1;[\Delta\eta]_2;...;[\Delta\eta]_l;...;[\Delta\eta]_L)$ so, equation (47) rewrites:

$$\sum_{l=1}^{L}[\mathcal{W}_{\rm vei}]_l\left(\frac{dX}{dt}\right)_l=\sum_{l=1}^{L}[\mathcal{W}_{\rm vei}]_l\mathcal{A}_l+\sum_{l=1}^{L}[\mathcal{W}_{\rm vei}]_l\mathcal{F}_l \tag{50}$$

List of equations:

- Continuity equation.

Generic notation **N(X)LAG** stands for:

- **NVLAG** for continuity equation.

Generic notations **P(X2D)9** (2D term), **P(X)T1** (2D term), **P(X3D)L9** (3D term), stand for:

- **PX9**, **PSPT1**, **PCL9** for continuity equation.

Generic notation **P(X)NLT9** (3D term) stands for:

- **PSPNLT9** for continuity equation.

Generic notation for total term, linear term, non linear term, physics:

- $\mathcal{A}$ is the total term (sum of dynamical contributions): it is assumed to be a 3D term (sum of 3D and 2D contributions).
- $\mathcal{L}$ is the linear term (treated in the semi-implicit scheme): it is assumed to be a 2D term (vertical integral of a 3D term).
- the difference $\mathcal{A}-\beta\mathcal{L}$ is the non-linear term, considered as a 3D term.
- $\mathcal{F}$ is the sum of contributions computed in the physical parameterizations; it is assumed to be a 2D term (vertical integral of a 3D term).

Description stands for the general case where linear and non-linear terms are gathered in the same buffer, and where no additional splitting is required to do diagnostics or to apply SLHD interpolations to a subset of the terms interpolated by a high-order interpolation. In some particular cases, additional splitting involving separate buffers may be required (see in part 5.4).

Buffers content for **N(X)LAG**=2 is the same one as for **N(X)LAG**=3 but trilinear interpolations are replaced by high-order ones. Only **N(X)LAG**=3 is described below.

High-order interpolations: in the following discretisations, "high-order interpolations" means: 32 points interpolations for 3D terms (vertical interpolations are cubic), 12 points interpolations for 2D terms.

Horizontal interpolation of 2D terms: since the horizontal position of the interpolation point is vertical dependent, horizontal interpolations of 2D quantities have to be done for each layer. For example, when interpolating a 2D surface variable at the origin point, $[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\frac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F[\text{surface quantity}]_O\right\rangle$ has no reason to be equal to $[\text{surface quantity}]_{O(\eta=1)}$, these quantities are generally different: this is $[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\frac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F[\text{surface quantity}]_O\right\rangle$ which has to be computed. $\left\langle\left[\frac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F[\text{surface quantity}]_O\right\rangle$ is the vector containing $\left[\frac{[\mathcal{W}_{\rm vei}]_l}{[\Delta\eta]_l}\right]_F[\text{surface quantity}]_{O(l)}$, $l=1$ to L. For **LVERTFE**=.F. this is $\sum_{l=1}^{L}[[\mathcal{W}_{\rm vei}]_l]_F[\text{surface quantity}]_{O(l)}$.

5.6.2 3TL vertical interpolating SL scheme.

Equation (47) is discretised as follows:

$$\begin{aligned}(X-(1+\epsilon)\Delta t\beta\mathcal{L})_F^+&=\{(1+\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o\}_F\\&+[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\frac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F\{X^-+(1-\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o+[(1-\epsilon)\Delta t\beta\mathcal{L}+2\Delta t\mathcal{F}]^-\}_O\right\rangle\end{aligned} \tag{51}$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X3D)L9**: $(1-\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o+[(1-\epsilon)\Delta t\beta\mathcal{L}]^-$ for trilinear interpolation at $O(l)$.
- **P(X2D)9**: $[X^-+2\Delta t\mathcal{F}]^-$ for horizontal high-order interpolation at $O(l)$.
- **P(X)T1**: $(1+\epsilon)\Delta t[\mathcal{A}-\beta\mathcal{L}]^o$ then provisional add of quantity $[(1+\epsilon)\Delta t\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.

5.6.3 2TL vertical interpolating SL scheme: conventional extrapolation.

The $t+0.5\Delta t$ non-linear term $\mathcal{A}^m-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^m$ used in the SL2TL scheme is replaced by a linear temporal extrapolation using the t and $t-\Delta t$ quantities at the same location. At the first time integration step, values at $t+0.5\Delta t$ are set equal to initial values and second-order uncentering is replaced by a first order uncentering. This discretisation of the SL2TL scheme follows (Mc Donald and Haugen, 1992). Quantity $\mathcal{A}^o-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^o$ has to be saved in a buffer **P(X)NLT9** to be available as $\mathcal{A}^--(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^-$ for the following timestep.
Equation (47) is discretised as follows:

$$\begin{aligned}(X-(1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L})^+_F=\{\tfrac{\Delta t}{2}[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^m\}_F\\ +[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\tfrac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F\{X^o+\tfrac{\Delta t}{2}[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^m+[(1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L}+\Delta t\mathcal{F}]^o\}_O\right\rangle\end{aligned}\tag{52}$$

which can be rewritten, once expanded the extrapolation:

$$\begin{aligned}(X-(1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L})^+_F=[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\tfrac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F\{X^o+\tfrac{\Delta t}{2}\mathcal{A}^o+\Delta t\mathcal{F}^o\}_O\right\rangle\\ +0.5[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\tfrac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F\tfrac{\Delta t}{2}\{(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o-(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-\}_O\right\rangle\\ +\tfrac{\Delta t}{2}[1.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o-0.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]_F\end{aligned}\tag{53}$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X3D)L9**: $\frac{\Delta t}{2}[1.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o-0.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]+[(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}]^o$ for trilinear interpolation at $O(l)$.
- **P(X2D)9**: $X^o+[\Delta t\mathcal{F}]^o$ for horizontal high-order interpolation at $O(l)$.
- **P(X)T1**: $\frac{\Delta t}{2}[1.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^o-0.5(\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L})^-]$ then provisional add of quantity $[(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.

5.6.4 2TL vertical interpolating SL scheme: stable extrapolation or mixed extrapolation.

At the first time integration step, values at $t+0.5\Delta t$ are set equal to initial values and second-order uncentering is replaced by a first order uncentering. Quantity $\mathcal{A}^o-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^o$ has to be saved in a buffer **P(X)NLT9** to be available as $\mathcal{A}^--(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}^-$ for the following timestep.
Equation (47) is discretised as follows:

$$\begin{aligned}(X-(1+\epsilon_{\mathcal{X}})\tfrac{\Delta t}{2}\beta\mathcal{L})^+_F=\{\tfrac{\Delta t}{2}[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^o\}_F\\ +[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\tfrac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F\{X^o+\tfrac{\Delta t}{2}\mathcal{A}+\Delta t\mathcal{F}]^o\}_O\right\rangle\\ +[\mathcal{R}_{\rm inte}]_{(top,surf)}\left\langle\left[\tfrac{\mathcal{W}_{\rm vei}}{\Delta\eta}\right]_F C_\gamma\tfrac{\Delta t}{2}\{[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^o-[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^-\}_O\right\rangle\end{aligned}\tag{54}$$

Buffers content before interpolations for **N(X)LAG**=3:

- **P(X3D)L9**: $\frac{\Delta t}{2}\mathcal{A}+C_\gamma\frac{\Delta t}{2}\{[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^o-[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^-\}$ for trilinear interpolation at $O(l)$
- **P(X2D)9**: $X^o+[\Delta t\mathcal{F}]^o$ for horizontal high-order interpolation at $O(l)$.
- **P(X)T1**: $\frac{\Delta t}{2}[\mathcal{A}-(1+\epsilon_{\mathcal{X}})\beta\mathcal{L}]^o$ then provisional add of quantity $[(1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}]^o$ before t+dt physics; evaluated at the final point F.
- **P(X3D)L0** and **P(X2D)0** are not used.

5.7 Discretisation for a 2D variable in a 2D model.

The content of the part (5.4) is generally valid, but there are particular remarks.

List of equations:

- Momentum equation.
- Continuity equation.

Generic notation **N(X)LAG** stands for:

- **NWLAG** for momentum equation.
- **NVLAG** for continuity equation. Negative values of **NVLAG** are used for Lagrangian formulation of continuity equation (the following discretisations apply to the absolute value of **NVLAG**), only **NVLAG**=-2 is available in this case.

Generic notation **P(X)L0**, **P(X)L9**, **P(X)T1** stands for:

- **PUL0**, **PUL9**, **PUT1** for U-momentum equation.
- **PVL0**, **PVL9**, **PVT1** for V-momentum equation.
- **PSPL0**, **PSPL9**, **PSPT1** for continuity equation.

Generic notation **P(X)NLT9** stands for:

- **PUNLT9** for U-momentum equation.
- **PVNLT9** for V-momentum equation.
- **PSPNLT9** for continuity equation.

Generic notation for total term, linear term, non linear term: $\mathcal{A}$ is the total term (sum of dynamical contributions), $\mathcal{L}$ is the linear term (treated in the semi-implicit scheme), the difference $\mathcal{A} - \beta\mathcal{L}$ is the non-linear term.

There is no diabatic term: $\mathcal{F}$ is replaced by zero.

High-order interpolations: "high-order interpolations" means 12 points interpolations.

Uncentering: only first-order uncentering is coded.

Vectorial variables: for vectorial variables (for example the horizontal wind) a rotation operator $\mathcal{R}$ has to be applied from interpolation point to final point:

- expression interpolated at O has to be replaced by $\mathcal{R}^{OF}\{\text{this expression}\}_O$.
- expression interpolated at M has to be replaced by $\mathcal{R}^{MF}\{\text{this expression}\}_M$.

For momentum equation: if $\beta = 1$ the non-linear term $\mathcal{A} - \mathcal{L}$ is zero.

5.8 Additional remarks.

∗ **Additional vertical derivatives:** if δ_{TR} is non-zero, discretisation of temperature equation needs to compute the vertical advection $\left(\dot{\eta}\frac{d\alpha_{\mathrm{T}}}{d\eta}\right)$ (at full levels) of α_{T}. Layers values of α_{T} (array **RCORDIF**) are used to define $T + \delta_{\mathrm{TR}}\frac{\alpha_{\mathrm{T}}\Phi_{\mathrm{s}}}{R_{\mathrm{d}}T_{\mathrm{st}}}$, but half level values of α_{T} (array **RCORDIH**) are used to compute vertical advection.

∗ **Case when some variables are evaluated at half levels:** the prognostic variables and the RHS of equations are generally evaluated at full levels in the discretisation. In the non-hydrostatic models there is an option allowing to advect w (at half levels) instead of d (at full levels). The previous general considerations valid to layer variables also apply to half level variables, but a "half level" trajectory has now to be computed (the origin O now matches a half level final point F).

- Horizontal displacement at half levels: the coordinates of the half level-trajectory interpolation point are computed as the average (with a vertical weight taking account of η) of the coordinates of the adjacent full level-trajectory interpolation points; this average is a lon-lat average on the computational sphere in spherical geometry, and a x-y average on the projection plane in plane geometry (LAM models).
- Vertical displacement at half levels: the vertical coordinate of the half level-trajectory interpolation point is computed as the average (with a vertical weight taking account of η) of the vertical coordinates of the adjacent full level-trajectory interpolation points; no vertical displacement if there is no vertical displacement for the two adjacent full levels.
- No complete iterative recalculation of trajectory is done for half-level trajectories.

∗ **Remarks for spline cubic vertical interpolations:** in this case the vertical interpolation uses all model levels and can be written as the product of two vertical interpolations: the first one uses all model levels and can be done at F in the unlagged grid-point calculations (the intermediate quantity obtained is stored in the array **P(X)SPL9**), the second one is a 4 points interpolation, done in the lagged grid-point calculations in the interpolation routine. Interpolation routine uses both **P(X)SPL9** (for interpolations) and **P(X)L9** to apply a monotonicity constraint.

6 Computation of medium and origin points.

Preliminary remark: the subsections (6.1) and (6.2) are detailed for a spherical geometry and for the trajectory which is used in the advection of full level variables; the subsection (6.4) gives informations about the other cases (for example plane geometry and half level variables).

6.1 Medium point M and origin point O.

Trajectories are great circles on the geographical sphere. The computation of the medium point M and the origin point O locations of the Lagrangian trajectory is performed by an iterative method described by Robert (1981) and adapted to the sphere by M. Rochas. In a SL3TL scheme, the particle is at the point M at the instant t ($t+0.5\Delta t$ for the first integration step) and at the point O at the instant $t-\Delta t$ (t for the first integration step). In a SL2TL scheme, the particle is at the point M at the instant $t+0.5\Delta t$ and at the point O at the instant t. M is at the middle position of the origin point O and the final point F. For convenience equations are written with the angular velocity $\mathbf{V}/a$. Parts (6.1.1), (6.1.2) and (6.1.3) are valid for non implicit iterative schemes. Part (6.1.4) is valid for a class of iterative centred-implicit schemes where the momentum equation is treated in an iterative centred-implicit manner (this is the case of the option **LPC_FULL**).

$*$ Denotations:

- $\mathcal{R}^{MF}$ is the rotation operator from medium point to final point (see section 11).
- $\mathcal{R}^{OF}$ is the rotation operator from origin point to final point (see section 11).
- $\mathbf{r}^F = \mathbf{CF}$ (C Earth centre, F final point); $\mathbf{r}^M = \mathbf{CM}$ (M medium point).
- $\mathbf{r} = a\mathbf{k}$.
- ϕ^{MF}: angle $(\widehat{\mathbf{CM}, \mathbf{CF}})$.
- θ^F,λ^F: latitude, longitude on the geographical sphere of F.
- θ^M,λ^M: latitude, longitude on the geographical sphere of M.
- $\mathbf{V}^M$: interpolated horizontal wind at M (wind at t in SL3TL scheme, $t+0.5\Delta t$ in SL2TL scheme).
- $\mathbf{V}^O$: interpolated horizontal wind at O (wind at t in SL3TL scheme, $t+0.5\Delta t$ in SL2TL scheme).
- a is the average Earth radius near the surface.
- Δt: time-step.
- δt: in a SL3TL scheme, $\delta t = 0.5\Delta t$ at the first integration step, $\delta t = \Delta t$ otherwise; in a SL2TL scheme, $\delta t = 0.5\Delta t$.
- L: number of layers of the model.
- A, B define hydrostatic pressure on the η levels ($\Pi = A + B\Pi_s$).
- Π_{sst} is the surface pressure of the standard atmosphere (variable **VP00**).
- Π_{st} is a reference hydrostatic pressure defined at full levels and half levels corresponding to the surface reference hydrostatic pressure Π_{sst} ($\Pi_{st} = A + B\Pi_{sst}$): stored in array **STPRE**.

$*$ Definition of the vertical coordinate η: Research of medium point needs an exact definition of the vertical coordinate η. For the half level number $\bar{l}$ ($\bar{l}$ between 0 and L), $\eta_{\bar{l}}$ is defined by:

- Regular spacing (**LREGETA**=.T.): $\eta_{\bar{l}} = \bar{l}/L$.
- Irregular spacing (**LREGETA**=.F.): $\eta_{\bar{l}} = A_{\bar{l}}/\Pi_{sst} + B_{\bar{l}}$.

For the layer number l (l between 1 and L), η_l is defined by: $\eta_l = 0.5(\eta_{\bar{l}} + \eta_{\bar{l}-1})$.
A specific definition of η may be required for the VFE operators if **LVERTFE**=.T.: it is controlled by the key **LVFE_REGETA**.

6.1.1 Conventional algorithm.

$*$ Algorithm: The medium point is computed by the following iterative scheme: for the iteration $k+1$:

$$[\mathbf{r}/a]^M_{k+1} = [\mathbf{r}/a]^F \cos\phi_k - \frac{\mathcal{R}^{MF}([\mathbf{V}/a]^M_k)}{\mid [\mathbf{V}/a]^M_k \mid} \sin\phi_k \tag{55}$$

where:

$$\phi_k = \delta t \mid [\mathbf{V}/a]^M_k \mid \tag{56}$$

ϕ is a small angle:

$$\sin\phi \simeq \phi - (1/6)\phi^3 \tag{57}$$

and:

$$\cos\phi \simeq 1 - 0.5\phi^2 \tag{58}$$

This approximation allows to simplify some calculations. Of course:

$$\sin\phi \simeq \phi(1-(1/6)\phi^2) \simeq \mid [\mathbf{V}/a] \mid \delta t(1-(1/6)\phi^2) \tag{59}$$

thus:

$$[\mathbf{r}/a]^M_{k+1} = [\mathbf{r}/a]^F(1-0.5\phi_k^2) - \mathcal{R}^{MF}([\mathbf{V}/a]^M_k)\delta t(1-(1/6)\phi_k^2) \tag{60}$$

On the vertical, for 3D model:

$$\eta^M_{k+1} = \eta^F - \delta t\dot{\eta}^M_k \tag{61}$$

∗ **First iteration:** Let us start with $M_0 = F$, $[\mathbf{V}/a]^M_0 = [\mathbf{V}/a]^F$, $\phi_0 = \delta t \mid [\mathbf{V}/a]^F \mid$, $\dot{\eta}^M_0 = \dot{\eta}^F$. Horizontal wind $\mathbf{V}$ has components (u,v). Thus

$$\sin\theta^M_1 = \sin\theta^F\cos\phi_0 - \frac{[v/a]^F}{\mid [\mathbf{V}/a]^F \mid}\cos\theta^F\sin\phi_0 \tag{62}$$

$$\cos\theta^M_1\cos(\lambda^M_1 - \lambda^F) = \cos\theta^F\cos\phi_0 + \frac{[v/a]^F}{\mid [\mathbf{V}/a]^F \mid}\sin\theta^F\sin\phi_0 \tag{63}$$

$$\cos\theta^M_1\sin(\lambda^M_1 - \lambda^F) = -\frac{[u/a]^F}{\mid [\mathbf{V}/a]^F \mid}\sin\phi_0 \tag{64}$$

$$\eta^M_1 = \eta^F - \delta t\dot{\eta}^F_0 \tag{65}$$

This defines the coordinates of M_1. Then $[u/a]$, $[v/a]$, $\dot{\eta}$ are interpolated at this point, that gives $[\mathbf{V}/a]^M_1$ and $\dot{\eta}^M_1$. Tri-linear interpolations are used in the 3D primitive equation model, horizontal 12 points interpolations are used in the 2D shallow-water model (see section 14).

∗ **Following iterations:** Let us denote by $\mathbf{V}'(u',v') = \mathcal{R}^{MF}(\mathbf{V}^M_k)$.

$$\sin\theta^M_{k+1} = \sin\theta^F\cos\phi_k - \frac{[v'/a]}{\mid [\mathbf{V}'/a] \mid}\cos\theta^F\sin\phi_k \tag{66}$$

$$\cos\theta^M_{k+1}\cos(\lambda^M_{k+1} - \lambda^F) = \cos\theta^F\cos\phi_k + \frac{[v'/a]}{\mid [\mathbf{V}'/a] \mid}\sin\theta^F\sin\phi_k \tag{67}$$

$$\cos\theta^M_{k+1}\sin(\lambda^M_{k+1} - \lambda^F) = -\frac{[u'/a]}{\mid [\mathbf{V}'/a] \mid}\sin\phi_k \tag{68}$$

$$\eta^M_{k+1} = \eta^F - \delta t\dot{\eta}^M_k \tag{69}$$

This defines coordinates of M_{k+1}. Then, if it is not the last iteration, $[u/a]$, $[v/a]$, $\dot{\eta}$ are interpolated at this point, that gives $[\mathbf{V}/a]^M_{k+1}$ and $\dot{\eta}^M_{k+1}$. Tri-linear interpolations are used in the 3D primitive equation model, horizontal 12 points interpolations are used in the 2D shallow-water model (see section 14).

∗ **Remarks:**

- Calculation of M is done for the SL3TL advection. SL2TL advection directly computes the origin point O (replace δt by $2\delta t$ and superscript M by O in the above formulae).
- Extrapolation of the wind for SL2TL scheme: the quantity $[\mathbf{V}/a]$ at $t + 0.5\Delta t$ used in the SL2TL scheme is computed by a conventional linear temporal extrapolation using the t and $t - \Delta t$ winds at the same location.

6.1.2 "SETTLS" stable algorithm for SL2TL scheme.

∗ **Extrapolation of the wind:** The previous algorithm with a conventional extrapolation can sometimes generate instability (especially when applied to the vertical displacement). The quantity $[\mathbf{V}/a]$ at $t + 0.5\Delta t$ is now replaced by a stable (SETTLS) extrapolation.

$$0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O(t) - [\mathbf{V}/a]^O(t-\Delta t))$$

The same type of extrapolation is done for the η-coordinate vertical velocity. The algorithm of research of trajectory directly computes the origin point O.

∗ **Algorithm:** The origin point is computed by the following iterative scheme: for the iteration $k+1$:

$$[\mathbf{r}/a]^O_{k+1} = [\mathbf{r}/a]^F \cos\phi_k - \frac{0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O_k(t) - [\mathbf{V}/a]^O_k(t-\Delta t))}{\mid 0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O_k(t) - [\mathbf{V}/a]^O_k(t-\Delta t)) \mid} \sin\phi_k \tag{70}$$

where:

$$\phi_k = 2\delta t \mid 0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O_k(t) - [\mathbf{V}/a]^O_k(t-\Delta t)) \mid \tag{71}$$

Approximations given by equations (57), (58) and (59) are still valid (change δt by $2\delta t$ in (59)), thus:

$$[\mathbf{r}/a]^O_{k+1} = [\mathbf{r}/a]^F(1-0.5\phi_k^2) - (0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O_k(t) - [\mathbf{V}/a]^O_k(t-\Delta t)))(2\delta t)(1-(1/6)\phi_k^2) \tag{72}$$

On the vertical, for 3D model:

$$\eta^O_{k+1} = \eta^F - 2\delta t(0.5\dot\eta^F(t) + 0.5(2\dot\eta^O_k(t) - \dot\eta^O_k(t-\Delta t))) \tag{73}$$

∗ **First iteration:** One starts with $M_0 = F$, $[\mathbf{V}/a]^F(t)$ as a first guess for the spatio-temporally extrapolated horizontal angular velocity, $\phi_0 = 2\delta t \mid [\mathbf{V}/a]^F(t) \mid$, $\dot\eta^F(t)$ as a first guess for the spatio-temporally extrapolated η-coordinate vertical wind. Remark that quantities at t are taken as a first guess and not quantities at $(t+0.5\Delta t)$, contrary to the case with conventional extrapolations. Use equations (62) to (65) replacing δt by $2\delta t$ and the superscript M by O. This defines the coordinates of O_1; $(2[\mathbf{V}/a](t) - [\mathbf{V}/a](t-\Delta t))$ and $(2\dot\eta(t) - \dot\eta(t-\Delta t))$ are interpolated at this point, that allows to compute the wind components which will be used for the next iteration.

∗ **Following iterations:** $[\mathbf{V}'/a]$ (of coordinates $([u'/a],[v'/a])$) is a generic notation for $(0.5[\mathbf{V}/a]^F(t) + 0.5\mathcal{R}^{OF}(2[\mathbf{V}/a]^O_k(t) - [\mathbf{V}/a]^O_k(t-\Delta t)))$. For horizontal displacement use equations (66) to (68) replacing δt by $2\delta t$ and the superscript M by O. For vertical displacement use equation

$$\eta^O_{k+1} = \eta^F - 2\delta t(0.5\dot\eta^F(t) + 0.5(2\dot\eta^O_k(t) - \dot\eta^O_k(t-\Delta t))) \tag{74}$$

6.1.3 "LELTRA" alternate stable algorithm for SL2TL scheme.

Extrapolation of the horizontal angular velocity: the quantity $[\mathbf{V}/a]$ at $t+0.5\Delta t$ used in the SL2TL scheme is computed using the RHS of the $[\mathbf{V}/a]$ equation with explicit formulation of Coriolis term. We denote this RHS by $RHSAV$ in subsubsection (6.1.3). The extrapolated value of $[\mathbf{V}/a]$ is given by:

$$[\mathbf{V}/a]_{\rm ext} = [\mathbf{V}/a](t) + 0.5\Delta t RHSAV \tag{75}$$

Extrapolation of the vertical velocity: the RHS is assumed to be zero, so:

$$\dot\eta_{\rm ext} = \dot\eta(t) \tag{76}$$

In fact, there is no extrapolation at all.

The algorithm of research of trajectory directly computes the origin point O.

∗ **Algorithm:** The origin point is computed by the following iterative scheme: for the iteration $k+1$, a provisional position of the origin point is computed using equations (55) to (61), replacing "M" by "O" and "δt" by "Δt".

∗ **First iteration:** Let us start with $O_0 = F$, $[\mathbf{V}/a]^O_0 = [\mathbf{V}/a]^F_{\rm ext}$, $\phi_0 = \Delta t \mid [\mathbf{V}/a]^F_{\rm ext} \mid$, $\dot\eta^O_0 = \dot\eta^F_{\rm ext}$. Horizontal angular velocity $[\mathbf{V}/a]_{\rm ext}$ has components $(u/a, v/a)$. Use equations (62) to (65), replacing "M" by "O" and "δt" by "Δt" to define coordinates of O_1. Then u/a,v/a,$\dot\eta$ are interpolated at this point, that gives $[\mathbf{V}/a]^O_1$ and $\dot\eta^O_1$ (subscript "ext" is omitted). Tri-linear interpolations are used in the 3D primitive equation model, horizontal 12 points interpolations are used in the 2D shallow-water model (see section 14).

∗ **Following iterations:** Use equations (66) to (69), replacing "M" by "O" and "δt" by "Δt" to define coordinates of O_{k+1}. Then u/a,v/a,$\dot\eta$ are interpolated at this point, that gives $[\mathbf{V}/a]^O_{k+1}$ and $\dot\eta^O_{k+1}$ (subscript "ext" is omitted). Tri-linear interpolations are used in the 3D primitive equation model, horizontal 12 points interpolations are used in the 2D shallow-water model (see section 14).

6.1.4 Algorithm used with the iterative centred-implicit schemes.

∗ **Preliminary remarks:** Iterative centred-implicit schemes are used to improve stability and it has been shown that this type of scheme has to be used when non-hydrostatic equations are advected by a SL2TL scheme. There are several manners to do iterative centred-implicit schemes. For description of those coded in ARPEGE/IFS, see documentation (IDSI). When the momentum equation is treated by an iterative centred-implicit scheme, the semi-Lagrangian trajectory has to be recomputed at each iteration of the iterative centred-implicit scheme and interpolations have to be done again. In this case the algorithm of research trajectory is modified. The case is currently encountered with a SL2TL scheme and **LPC_FULL**=.T. . The following algorithm will be described for a SL2TL scheme but it can be extended to a SL3TL scheme (replacing instant t by instant $t-\Delta t$).

∗ **Extrapolation of the wind if non-extrapolating option:** No extrapolation is done. The first iteration of the iterative centred-implicit scheme uses $[\mathbf{V}/a](t)$ and $\dot{\eta}(t)$. The following iterations of the iterative centred-implicit scheme use the $[\mathbf{V}/a](t+\Delta t)$ and $\dot{\eta}(t+\Delta t)$ of the previous iteration to start the research of trajectory. As for the "SETTLS"-stable extrapolating option, the algorithm computes the origin point O.

∗ **Extrapolation of the wind if "SETTLS"-stable extrapolating option:** The only difference with the previous case is for the first iteration of the iterative centred-implicit scheme: the wind which is used is now $1.5[\mathbf{V}/a](t) - 0.5[\mathbf{V}/a](t-\Delta t)$ and $1.5\dot{\eta}(t) - 0.5\dot{\eta}(t-\Delta t)$.

∗ **Algorithm:** One denotes by index "k" the numbering of the SL-trajectory research algorithm iteration, and by index "(i)" the number of the iterative centred-implicit scheme iteration (in variable **NCURRENT_ITER**). The origin point is computed by the following iterative scheme: for the iteration $k+1$ of the SL-trajectory research algorithm:

$$\left[[\mathbf{r}/a]^O_{k+1}\right]_{(i)} = [\mathbf{r}/a]^F \left[\cos\phi_k\right]_{(i)} - \frac{0.5\mathcal{R}^{OF}[\mathbf{V}/a]^O_k(t) + 0.5\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)}}{\mid 0.5\mathcal{R}^{OF}[\mathbf{V}/a]^O_k(t) + 0.5\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)} \mid} \left[\sin\phi_k\right]_{(i)} \tag{77}$$

where:

$$\left[\phi_k\right]_{(i)} = 2\delta t \mid 0.5\mathcal{R}^{OF}[\mathbf{V}/a]^O_k(t) + 0.5\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)} \mid \tag{78}$$

Approximations given by equations (57), (58) and (59) are still valid (change δt by $2\delta t$ in (59)), thus:

$$\left[[\mathbf{r}/a]^O_{k+1}\right]_{(i)} = [\mathbf{r}/a]^F(1 - 0.5[\phi^2_k]_{(i)}) - \left(0.5\mathcal{R}^{OF}[\mathbf{V}/a]^O_k(t) + 0.5\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)}\right)(2\delta t)(1 - (1/6)[\phi^2_k]_{(i)}) \tag{79}$$

On the vertical, for 3D model:

$$\left[\eta^O_{k+1}\right]_{(i)} = \eta^F - 2\delta t\left(0.5\dot{\eta}^O_k(t) + 0.5\left[\dot{\eta}^F(t+\Delta t)\right]_{(i-1)}\right) \tag{80}$$

For $i = 0$: $[[\mathbf{V}/a](t+\Delta t)]_{(i=0)} = [\mathbf{V}/a](t)$ and $[\dot{\eta}(t+\Delta t)]_{(i=0)} = \dot{\eta}(t)$ for non-extrapolating option; $[[\mathbf{V}/a](t+\Delta t)]_{(i=0)} = 2[\mathbf{V}/a](t) - [\mathbf{V}/a](t-\Delta t)$ and $[\dot{\eta}(t+\Delta t)]_{(i=0)} = 2\dot{\eta}(t) - \dot{\eta}(t-\Delta t)$ for "SETTLS"-stable extrapolating option.

∗ **First iteration of the research of SL trajectory:**
One starts with $M_0 = F$, $\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)}$ as a first guess for the spatio-temporally extrapolated horizontal wind, $[\phi_0]_{(i)} = 2\delta t \mid \left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)} \mid$, $\left[\dot{\eta}^F(t+\Delta t)\right]_{(i-1)}$ as a first guess for the spatio-temporally extrapolated η-coordinate vertical wind. Use equations (62) to (65) replacing δt by $2\delta t$ and the superscript M by O in the SL-trajectory research. This defines the coordinates of $[O_1]_{(i)}$; $[\mathbf{V}/a](t)$ and $\dot{\eta}(t)$ are interpolated at this point, that allows to compute the wind components which will be used for the next iteration of the research of SL trajectory.

∗ **Following iterations of the research of SL trajectory:** $[\mathbf{V}'/a]$ (of coordinates $([u'/a], [v'/a])$ is a generic notation for $0.5\mathcal{R}^{OF}[\mathbf{V}/a]^O_k(t) + 0.5\left[[\mathbf{V}/a]^F(t+\Delta t)\right]_{(i-1)}$. For horizontal displacement use equations (66) to (68) replacing δt by $2\delta t$ and the superscript M by O otherwise. For vertical displacement use equation

$$\left[\eta^O_{k+1}\right]_{(i)} = \eta^F - 2\delta t\left(0.5\dot{\eta}^O_k(t) + 0.5\left[\dot{\eta}^F(t+\Delta t)\right]_{(i-1)}\right) \tag{81}$$

∗ **"LELTRA" alternate stable algorithm for SL2TL scheme:** For each iteration of the iterative centred-implicit scheme the algorithm is the same as for explicit schemes; at each iteration the RHS of the momentum equation is updated with the "provisional" $t+\Delta t$ information computed at the previous iteration.

6.1.5 Compute origin point O from medium point M.

When the position of M is computed, the position of O can be computed with the following algorithm.

O is on the same great circle arc (on the geographical sphere) as M and F and the length of OF is twice the length of MF. If angle $(\widehat{[\mathbf{r}/a]^O, [\mathbf{r}/a]^F})$ is small (less than 10^o, what is generally matched), one can write for horizontal displacement:

$$[\mathbf{r}/a]^O - [\mathbf{r}/a]^F \simeq 2([\mathbf{r}/a]^M - [\mathbf{r}/a]^F) \tag{82}$$

For vertical displacement on can always write:

$$\eta^O - \eta^F = 2(\eta^M - \eta^F) \tag{83}$$

One denotes by:

- $\phi = (\widehat{[\mathbf{r}/a]^M, [\mathbf{r}/a]^F})$
- $[\mathbf{V}'/a]$ (of coordinates $([u'/a], [v'/a])$) the last interpolated horizontal velocity.

Using the following identities:

$$\cos 2\phi = 2\cos^2\phi - 1 \tag{84}$$

$$\sin 2\phi = 2\sin\phi\cos\phi \tag{85}$$

the origin point horizontal coordinates can be computed by:

$$\sin\theta^O = \sin\theta^F \cos 2\phi - 2\cos\phi \left[\frac{[v'/a]}{|\,[\mathbf{V}'/a]\,|} \cos\theta^F \sin\phi \right] \tag{86}$$

$$\cos\theta^O \cos(\lambda^O - \lambda^F) = \cos\theta^F \cos 2\phi + 2\cos\phi \left[\frac{[v'/a]}{|\,[\mathbf{V}'/a]\,|} \sin\theta^F \sin\phi \right] \tag{87}$$

$$\cos\theta^O \sin(\lambda^O - \lambda^F) = -2\cos\phi \left[\frac{[u'/a]}{|\,[\mathbf{V}'/a]\,|} \sin\phi \right] \tag{88}$$

Terms in brackets are already computed to determine M.

6.2 Refined recomputation of point O.

∗ **Option L2TLFF for RW2TLFF=1:** This option controls recomputation of the origin point using the average between the angular velocity at the origin point and the provisional $t+\Delta t$ angular velocity, according to the algorithm previously described. Only term $(2\mathbf{\Omega} \wedge a\mathbf{k})$ is computed (always analytically) at this improved position of O (so **L2TLFF** is active only if **LADVF**=.T.). Refined recomputation of point O is available only in a limited set of options. In the following sections, discretised equations are written with notation O for all quantities. Equations system is integrated to find a first guess of $\mathbf{V}^F(t+\Delta t)$, then $0.5([\mathbf{V}/a]^F + \mathcal{R}^{OF}[\mathbf{V}/a]^O)$ is used to recompute O. A correction $(2\mathbf{\Omega}\wedge a\mathbf{k})(O_{\text{improved}}) - (2\mathbf{\Omega}\wedge\mathbf{k})(O)$ is analytically computed and added to wind equation to find the "improved" value of $\mathbf{V}^F(t+\Delta t)$. Computations are currently done in routine **LAPINEB** and **LADINE**.

∗ **Options L2TLFF for RW2TLFF between 0 and 1:** In this case the improved position ON of the origin point O is computed as a linear interpolation between O and its position for **RW2TLFF**=1 on a great circle bow on geographical sphere. For **RW2TLFF**=0. ON=O.

6.3 Trajectory going out of atmosphere or underground, and algorithms to prevent that.

∗ **Trajectory going out of atmosphere for trajectories ending at a layer final point:** If the origin point O is found above the top of the atmosphere (resp. under the ground) it is put on the top of the atmosphere (resp. on the ground). Then the position of the medium/origin point is recomputed (when necessary), that gives necessary a point between the bottom and the top of the atmosphere. At last the origin point is bounded by a vertical position η_O between the top of the atmosphere and the layer $l = 1$, according to the namelist variable **VETAON** (resp. between the layer $l = L$ and the ground, according to the namelist variable **VETAOX**). Upper bound of O is $\eta_O = \eta_{l=1} + (\mathbf{VETAON} - 1)(\eta_{l=1} - \eta_{\bar{l}=0})$. Lower bound of O is $\eta_O = \eta_{l=L} + (1 - \mathbf{VETAOX})(\eta_{\bar{l}=L} - \eta_{l=L})$.

∗ **Alternate treatment of vertical displacement (LRALTVDISP=T):** The algorithm proposed is an extension of an algorithm described in (Wood et al., 2009). We propose to compute two vertical displacements, one with η and one with $\log((\eta - \eta_N)/(\eta_X - \eta))$, and we finally take the shorter displacement computed.

- η_N is a lower boundary for η which matches $0 \leq \eta_N < \eta_{l=1}$ (currently taken equal to **VETAON**).
- η_X is an upper boundary for η which matches $\eta_{l=L} < \eta_X \leq 1$ (currently taken equal to **VETAOX**).

The vertical displacement computed with $\log((\eta - \eta_N)/(\eta_X - \eta))$ yields, in a 2TLSL scheme (replace Δt by $2\Delta t$ for a SL3TL scheme):

$$\log\frac{(\eta_O - \eta_N)}{(\eta_X - \eta_O)} = \log\frac{(\eta_F - \eta_N)}{(\eta_X - \eta_F)} - \Delta t \frac{d(\log((\eta - \eta_N)/(\eta_X - \eta)))}{dt} \tag{89}$$

This equation can be rewritten:

$$\log\frac{(\eta_O - \eta_N)}{(\eta_X - \eta_O)} = \log\frac{(\eta_F - \eta_N)}{(\eta_X - \eta_F)} - \Delta t \left(\frac{1}{\eta - \eta_N} + \frac{1}{\eta_X - \eta} \right) \dot{\eta} \tag{90}$$

The solution of this equation ensures that:

$$\eta_N < \eta_O < \eta_F$$

For $\eta < 0.5$ the vertical displacement computed with $\log((\eta - \eta_N)/(\eta_X - \eta))$ computes lower displacements if the trajectory is descending (this is the case where the origin point may be out of the atmosphere), and bigger displacements if the trajectory is ascending.
For $\eta > 0.5$ the vertical displacement computed with $\log((\eta - \eta_N)/(\eta_X - \eta))$ computes lower displacements if the trajectory is ascending (this is the case where the origin point may be under the ground), and bigger displacements if the trajectory is descending.
In both cases the origin point remains in the atmosphere.

This algorithm can be done for non extrapolating or "SETTLS" extrapolation in a 2TLSL scheme.

6.4 Remarks.

∗ **Shallow water model:** Computations remain valid for horizontal coordinates (there is no vertical movement, equations containing η have not to be considered). There are remaining some old features (compute M in **LARMES2** even for options where O is directly computed in the 3D model, then O in **LAINOR2**).

∗ **Case of interpolations applied to half level variables:** That occurs for example in the non-hydrostatic scheme when the half level variable w is advected instead of the full level vertical divergence variable (option **LGWADV**=.T.). In this case one needs to define an origin $O(\bar{l})$ for a half level trajectory (which ends at a half level final point $F(\bar{l})$). The following rules are applied to compute such a kind of trajectory:

- The horizontal displacement from $F(\bar{l})$ is a weighted average of the horizontal displacements from $F(l-1)$ and $F(l)$ (see below for vertical displacement).
- At the top (resp. bottom) the horizontal displacement from $F(\bar{l} = 0)$ (resp. $F(\bar{l} = L)$) is equal to the horizontal displacement from $F(l = 1)$ (resp. $F(l = L)$).
- The rule applied to the horizontal displacement is also applied to the vertical displacement for $\bar{l}$ between 1 and $L-1$. For example, if $\bar{l}$ is between 1 and $L-1$:

$$\eta_{O(\bar{l})} = + \left(1 - \frac{\eta_{O(\bar{l})} - \eta_{O(l)}}{\eta_{O(l+1)} - \eta_{O(l)}}\right) \eta_{O(l)} + \frac{\eta_{O(\bar{l})} - \eta_{O(l)}}{\eta_{O(l+1)} - \eta_{O(l)}} \eta_{O(l+1)}$$

- A particle coming from the top or the bottom has no vertical displacement (that assumes that $\dot{\eta} = 0$ at the top and the bottom, and so that excludes the options **LRUBC**=.T., **NDPSFI**=1 and $\Pi_{\text{top}} > 0$ which are not consistent with the constraint $(\dot{\eta}_{\text{top}} = 0; \dot{\eta}_{\text{surf}} = 0)$).
- If the trajectory goes above the top of the atmosphere, it is bounded at the top of the atmosphere.
- If the trajectory goes below the surface, it is bounded at the surface.
- Computation of the position of $O(\bar{l})$ is done in routine **LARCINHA**.

∗ **Plane geometry (LAM models):** Computation of the SL trajectory is done on the projected plane geometry. **ELARMES** and **ELARMES2** are called instead of **LARMES** and **LARMES2**.

∗ **Treatment of $\dot{\eta}$ in the upper stratosphere:** For some options of the code the horizontal interpolations applied on $\dot{\eta}$ are replaced by "least-square" interpolations; that allows to remove some instabilities. See documentation (IDSVTSM) for more details.

∗ **Option LSETTLSVF (SL2TL schemes):** When the "SETTLS" stable extrapolation is applied to $\dot{\eta}$, and when **LSETTLSVF**=T, the spirit is to replace the stable extrapolation by a non-extrapolating evaluation of $\dot{\eta}$ at locations where $\dot{\eta}$ has large variations; this algorithm has been recently modified in order to have a more continuous one.

∗ **Variable NFOST (SL2TL schemes):** When the "SETTLS" stable extrapolation is applied, by default it is applied to all timesteps. When setting **NFOST** to a positive non-zero value, "NESC" extrapolation is done for timesteps 0 to **NFOST**-1, "SETTLS" extrapolation is done for timesteps from **NFOST**.

7 The SL discretisation of the 2D shallow-water system of equations (spherical geometry).

7.1 Momentum equation.

∗ Definition of X, $\mathcal{A}$ and $\mathcal{L}$.

$$X = \mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r}) \tag{91}$$

$$\mathcal{A} = -2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V}) - \nabla\Phi \tag{92}$$

$$\mathcal{L} = -\nabla\Phi + \beta_{\rm Co}[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] \tag{93}$$

∗ Remarks.

- Coriolis term can be treated explicitly ($\delta_{\mathbf{V}} = 0$) or implicitly ($\delta_{\mathbf{V}} = 1$). Use key **LADVF**. **LADVF**=.F. matches with ($\delta_{\mathbf{V}} = 0$). **LADVF**=.T. matches with ($\delta_{\mathbf{V}} = 1$). If **LADVF**=.T., term ($2\mathbf{\Omega} \wedge \mathbf{r}$) is analytically computed.
- For a limited set of options, term ($2\mathbf{\Omega} \wedge \mathbf{r}$) can be recomputed at an improved position of the origin point (**RW2TLFF**>0).
- Coriolis term can also be put in the semi-implicit scheme by tuning $\beta_{\rm Co}$ (which has a sense only if **LADVF**=.F.). Caution: do not use **LIMPF**=.T. in variable resolution (formulation of spectral computations is not correct in this case for the semi-implicit scheme).
- If β=1, the non linear term $(-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V}) - \nabla\Phi) - \beta(-\nabla\Phi + \beta_{\rm Co}[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})])$ is zero.

7.2 Continuity equation: conventional formulation (NVLAG>0).

∗ Definition of X, $\mathcal{A}$ and $\mathcal{L}$.

$$X = (\Phi - (1-\delta_{\rm TR})\Phi_{\rm s}) \tag{94}$$

$$\mathcal{A} = -(\Phi - \Phi_{\rm s})D + \delta_{\rm TR}\mathbf{V}\nabla(\Phi_{\rm s}) \tag{95}$$

$$\mathcal{L} = -\Phi^{*}\overline{M}^{2}D' \tag{96}$$

$\delta_{\rm TR} = 1$ plays the same role as the "Tanguay-Ritchie" modification for 3D model.

7.3 Continuity equation: Lagrangian formulation (NVLAG<0).

∗ Definition of X, $\mathcal{A}$ and $\mathcal{L}$.

$$X = (\Phi - \Phi_{\rm s})J \tag{97}$$

$$\mathcal{A} = 0 \tag{98}$$

$$\mathcal{L} = -\Phi^{*}\overline{M}^{2}D'J \tag{99}$$

where J is a "Jacobian" quantity defined by its Lagrangian derivative (see equation (11)).

∗ Calculation of J.

- SL3TL: $J^{-} = (1-\Delta t D^{o})/(1+\Delta t D^{o})$, $J^{o} = 1/(1+\Delta t D^{o})$ and $J^{+} = 1$.
- SL2TL: currently not coded.

7.4 Quantities to be interpolated.

When researching the medium/origin point by an iterative algorithm, the interpolation at the medium/origin point of the two components of the horizontal wind is needed: a 12 points interpolation is used. For other quantities to be interpolated, see section 5. For more details about interpolations, see section 14.

8 The SL discretisation of the 3D primitive equation model.

Remark: the detailed discretisation of each part of the RHS is given in the documentation (IDEUL). Some notations used in the expression of linear term $\mathcal{L}$ (like τ, γ, μ, ν) are given in the documentation (IDSI).

8.1 Formulation of the momentum equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = \mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r}) \tag{100}$$

$$\mathcal{A} = -2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V}) - \nabla\Phi - RT\nabla(\log\Pi) \tag{101}$$

$$\mathcal{L} = -\nabla\left[\gamma T + R_{\rm d}T^{*}\log(\Pi_{\rm s})\right] + \beta_{\rm Co}[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] \tag{102}$$

$$\mathcal{F} = \mathbf{F}_{\mathbf{V}} \tag{103}$$

Top:

$$\mathbf{V}_{\eta=0} = \mathbf{V}_{l=1} \tag{104}$$

Bottom if $\delta m = 0$:

$$\mathbf{V}_{\eta=1} = \mathbf{V}_{l=L} \tag{105}$$

Bottom if $\delta m = 1$:

$$\mathbf{V}_{\eta=1} = 0 \tag{106}$$

$*$ Remarks.

- Coriolis term can be treated explicitly ($\delta_{\mathbf{V}} = 0$) or implicitly ($\delta_{\mathbf{V}} = 1$). Use key **LADVF**. **LADVF**=.F. matches with ($\delta_{\mathbf{V}} = 0$). **LADVF**=.T. matches with ($\delta_{\mathbf{V}} = 1$). If **LADVF**=.T., term ($2\mathbf{\Omega} \wedge \mathbf{r}$) is analytically computed.
- For a limited set of options, term ($2\mathbf{\Omega} \wedge \mathbf{r}$) can be recomputed at an improved position of the origin point (**RW2TLFF**>0).
- Coriolis term can also be put in the semi-implicit scheme by tuning $\beta_{\rm Co}$ (which has a sense only if **LADVF**=.F.). Caution: do not use **LIMPF**=.T. in variable resolution (formulation of spectral computations is not correct in this case for the semi-implicit scheme).

8.2 Thermodynamic equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = T + \delta_{\rm TR}\frac{\alpha_{\rm T}\Phi_{\rm s}}{R_{\rm d}T_{\rm st}} \tag{107}$$

$$\mathcal{A} = \frac{RT}{c_{\rm p}}\frac{\omega}{\Pi} + \delta_{\rm TR}\frac{\alpha_{\rm T}}{R_{\rm d}T_{\rm st}}\mathbf{V}\nabla(\Phi_{\rm s}) + \delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\left(\dot{\eta}\frac{d\alpha_{\rm T}}{d\eta}\right) \tag{108}$$

$$\mathcal{L} = -\tau(\overline{M}^2 D^{'}) \tag{109}$$

$$\mathcal{F} = F_{\rm T} \tag{110}$$

Top:

$$T_{\eta=0} = T_{l=1} \tag{111}$$

Bottom if $\delta m = 0$:

$$T_{\eta=1} = T_{l=L} \tag{112}$$

Bottom if $\delta m = 1$ (output of physics):

$$T_{\eta=1} = T_{\rm s} \tag{113}$$

8.3 Continuity equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$, and $\mathcal{F}$.

$$X = \log\Pi_{\rm s} + \delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}} \tag{114}$$

$$\mathcal{A} = -\frac{1}{\Pi_{\rm s}}\int_{\eta=0}^{\eta=1}\nabla\left(\mathbf{V}\frac{\partial\Pi}{\partial\eta}\right)d\eta + \mathbf{V}\nabla\left(\log\Pi_{\rm s} + \delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\right) - \frac{1}{\Pi_{\rm s}}\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=1} + \frac{1}{\Pi_{\rm s}}\left[\dot{\eta}\frac{\partial\Pi}{\partial\eta}\right]_{\eta=0} \tag{115}$$

$$\mathcal{L} = -\frac{\overline{M}^2}{M^2}\nu D \tag{116}$$

$$\mathcal{F} = \left(\frac{F_{\rm m}}{\Pi_{\rm s}}\right) \tag{117}$$

∗ **Remarks:**

- $\mathcal{A}$ is a sum of 3D terms (the advection term) and 2D terms (the other terms).
- $\mathcal{L}$ and $\mathcal{F}$ are 2D terms (vertical integrals).

8.4 Moisture equation.

∗ **Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.**

$$X = q \tag{118}$$

$$\mathcal{A} = 0 \tag{119}$$

$$\mathcal{L} = 0 \tag{120}$$

$$\mathcal{F} = F_{\mathrm{q}} \tag{121}$$

Top:

$$q_{\eta=0} = q_{l=1} \tag{122}$$

Bottom if $\delta m = 0$:

$$q_{\eta=1} = q_{l=L} \tag{123}$$

Bottom if $\delta m = 1$ (see **CPQSOL**, relative humidity is the same for $\eta = \eta_L$ and $\eta = 1$):

$$q_{\eta=1} = q_{\mathrm{surf}} \tag{124}$$

8.5 Other advectable GFL variables.

Equations are discretised as for humidity equation. Vertical boundary conditions: quantities are assumed constant above the middle of the upper layer and below the middle of the lower layer in case $\delta m = 0$; quantities are assumed constant above the middle of the upper layer in case $\delta m = 1$; quantities other than q are assumed to be zero at the surface in case $\delta m = 1$.

8.6 Case of lagged physics.

All the previous discretisations have been written with not lagged physics (interpolated at O).

- For lagged physics (**LAGPHY**=.T., **LSLPHY**=.F.): the previous discretisations are done without physics, then the provisional

$$(X^+ - (1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}^+ + (1+\epsilon_{\mathcal{X}})\frac{\Delta t}{2}\beta\mathcal{L}^o)_F$$

  or

$$(X^+ - (1+\epsilon)\frac{\Delta t}{2}\beta\mathcal{L}^+ + (1+\epsilon)\frac{\Delta t}{2}\beta\mathcal{L}^o)_F$$

  is used as input to the lagged physics.

- For split physics used at ECMWF (**LEPHYS**=.T., **LAGPHY**=.T., **LSLPHY**=.T.): one part of the physics is interpolated at O (t or $t-\Delta t$ physics according to **LTWOTL**), the remainder is evaluated at the final point ($t+\Delta t$ physics). The physical contribution is put in a separate interpolation buffer (name **P(X)P9**) and tri-linearly interpolated. The way to compute the non-lagged contribution is different than the way used at METEO-FRANCE: compute it at the previous timestep as a lagged contribution, then saving it from one timestep to the following one (where it is restored and added to the interpolation buffer by calling the routine **GPADDSLPHY**). Partition between the non-lagged and lagged contribution is done by a linear partition of coefficient **RSLWX**.

8.7 Quantities to be interpolated (computation under subroutine LACDYN).

∗ **Research of trajectory:** When researching the medium/origin point by an iterative algorithm, the interpolation at the medium/origin point of $([U/a], [V/a], \dot{\eta})$ is needed: a tri-linear interpolation is performed. For more details about interpolations, see section 14.

∗ **RHS of equations.** The list of quantities to be interpolated has been described in subsections 5.4, 5.5 and 5.6 for each type of equation.

∗ **Additional quantities to be interpolated at O if RW2TLFF>0.** The two components of the $[\mathbf{V}/a]$ at time t (if SL2TL scheme) or $t-\Delta t$ (if SL3TL scheme) when not available after the other interpolations (for example if not lagged physics). These additional interpolations are useless if lagged physics (or adiabatic run) and **NWLAG**=3.

9 The SL discretisation of the fully elastic non hydrostatic (NHEE) model.

Remarks: Some notations used in the expression of linear term $\mathcal{L}$ (like τ, γ, μ, ν) are given in the documentation (IDSI).

9.1 Momentum equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = \mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r}) \tag{125}$$

$$\mathcal{A} = [-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] - \frac{\partial p}{\partial \Pi}\nabla\Phi - RT\frac{\nabla(p)}{p} \tag{126}$$

$$\mathcal{L} = -\nabla\left[\gamma T - T^*(\gamma\hat{Q}) + R_{\rm d}T^*\log(\Pi_{\rm s}) + R_{\rm d}T^*\hat{Q}\right] + \beta_{\rm Co}[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] \tag{127}$$

$$\mathcal{F} = \mathbf{F}_{\mathbf{V}} \tag{128}$$

Top:

$$\mathbf{V}_{\eta=0} = \mathbf{V}_{l=1} \tag{129}$$

Bottom if $\delta m = 0$:

$$\mathbf{V}_{\eta=1} = \mathbf{V}_{l=L} \tag{130}$$

Bottom if $\delta m = 1$:

$$\mathbf{V}_{\eta=1} = 0 \tag{131}$$

Remarks about implicit formulations of the Coriolis term are generally valid for the non-hydrostatic equations.

9.2 Temperature equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = T + \delta_{\rm TR}\frac{\alpha_{\rm T}\Phi_{\rm s}}{R_{\rm d}T_{\rm st}} \tag{132}$$

$$\mathcal{A} = -\frac{RT}{c_{\rm v}}D_3 + \delta_{\rm TR}\frac{\alpha_{\rm T}}{R_{\rm d}T_{\rm st}}\mathbf{V}\nabla(\Phi_{\rm s}) + \delta_{\rm TR}\frac{\Phi_{\rm s}}{R_{\rm d}T_{\rm st}}\left(\dot{\eta}\frac{d\alpha_{\rm T}}{d\eta}\right) \tag{133}$$

$$\mathcal{L} = -\frac{R_{\rm d}T^*}{c_{\rm vd}}\left[\overline{M}^2 D^{'} + d\right] \tag{134}$$

$$\mathcal{F} = \left[\frac{c_{\rm p}}{c_{\rm v}}F_{\rm T}\right] \tag{135}$$

Top:

$$T_{\eta=0} = T_{l=1} \tag{136}$$

Bottom if $\delta m = 0$:

$$T_{\eta=1} = T_{l=L} \tag{137}$$

Bottom if $\delta m = 1$ (output of physics):

$$T_{\eta=1} = T_{\rm s} \tag{138}$$

9.3 Vertical divergence variable equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values, case NVDVAR=3, LGWADV=.F. .

$$X = d \tag{139}$$

$$\mathcal{A} = -dD_3 + d\nabla\mathbf{V} - \frac{gp}{R_{\rm a}T\frac{\partial\Pi}{\partial\eta}}\frac{\partial\left[\frac{dw}{dt}\right]_{\rm ad}}{\partial\eta} + \frac{gp}{R_{\rm a}T\frac{\partial\Pi}{\partial\eta}}(\nabla w)\left(\frac{\partial\mathbf{V}}{\partial\eta}\right) \tag{140}$$

$$\mathcal{L} = -\frac{g^2}{R_{\rm d}T_{\rm a}^*}(\mathbf{L}^*\hat{Q}) \tag{141}$$

$$\mathcal{F} = -\frac{d}{\left[\frac{\partial\Pi}{\partial\eta}\right]}F_{\rm m}^{'} - \left[\frac{gp}{R_{\rm a}T\frac{\partial\Pi}{\partial\eta}}\frac{\partial F_{\rm w}}{\partial\eta}\right] \tag{142}$$

Top:

$$d_{\eta=0} = d_{l=1} \tag{143}$$

Bottom if $\delta m = 0$:

$$d_{\eta=1} = d_{l=L} \tag{144}$$

Bottom if $\delta m = 1$: not yet coded.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values, case NVDVAR=3, LGWADV=.T. .

The linear system uses the d equation.

$$X = w \tag{145}$$

$$\mathcal{A} = g\frac{\partial(p-\Pi)}{\partial\Pi} \tag{146}$$

$$\mathcal{F} = F_{\rm w} \tag{147}$$

Top:

- **LVERTFE**=.F.: $w_{\eta=0}$ is computed by the general formula giving w at half levels.
- **LVERTFE**=.T.: VFD treatment for $w_{\eta=0}$ (cf. above).

Bottom if $\delta m = 0$:

$$w_{\eta=1} = \mathbf{V}_{\rm surf}\nabla z_{\rm surf} \tag{148}$$

Bottom if $\delta m = 1$: not yet coded.

9.4 Pressure departure variable equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values, case NPDVAR=2.

$$X = \hat{Q} \tag{149}$$

$$\mathcal{A} = -\frac{c_{\rm p}}{c_{\rm v}}D_3 - \frac{\omega}{\Pi} \tag{150}$$

$$\mathcal{L} = -\left[\frac{c_{\rm pd}}{c_{\rm vd}}(\overline{M}^2 D^{'} + d) - \frac{c_{\rm pd}}{R_{\rm d}T^*}\tau(\overline{M}^2 D^{'})\right] \tag{151}$$

$$\mathcal{F} = \frac{c_{\rm p}}{c_{\rm v}T}F_{\rm T} \tag{152}$$

Top:

$$\hat{Q}_{\eta=0} = 0 \tag{153}$$

Bottom if $\delta m = 0$:

$$\hat{Q}_{\eta=1} = \hat{Q}_{l=L} \tag{154}$$

Bottom if $\delta m = 1$: not yet coded.

9.5 Continuity equation and GFL equations.

Same discretisation as in the hydrostatic case, see parts (8.3), (8.4) and (8.5).

9.6 Case of lagged physics.

See part (8.6).

9.7 Quantities to be interpolated (computation under subroutine LACDYN).

See part (8.7).

10 The SL discretisation of the quasi-elastic non hydrostatic (NHQE) model.

Remarks: Some notations used in the expression of linear term $\mathcal{L}$ (like τ, γ, μ, ν) are given in the documentation (IDSI).

10.1 Momentum equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = \mathbf{V} + \delta_{\mathbf{V}}(2\mathbf{\Omega} \wedge \mathbf{r}) \tag{155}$$

$$\mathcal{A} = [-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] - \exp(\kappa_{\mathrm{m}}\hat{Q}) \left(1 + \Pi \frac{\partial \hat{Q}}{\partial \Pi}\right) \nabla\Phi - R\tilde{T}\exp(\kappa_{\mathrm{m}}\hat{Q}) \left(\frac{\nabla\Pi}{\Pi} + \nabla\hat{Q}\right) \tag{156}$$

$$\mathcal{L} = -\nabla[\gamma\tilde{T} + R_{\mathrm{d}}T^{*}\log(\Pi_{\mathrm{s}}) + R_{\mathrm{d}}T^{*}\hat{Q}] + \beta_{\mathrm{Co}}[-2(1-\delta_{\mathbf{V}})(\mathbf{\Omega} \wedge \mathbf{V})] \tag{157}$$

$$\mathcal{F} = \mathbf{F}_{\mathbf{V}} \tag{158}$$

Top:

$$\mathbf{V}_{\eta=0} = \mathbf{V}_{l=1} \tag{159}$$

Bottom if $\delta m = 0$:

$$\mathbf{V}_{\eta=1} = \mathbf{V}_{l=L} \tag{160}$$

Bottom if $\delta m = 1$:

$$\mathbf{V}_{\eta=1} = 0 \tag{161}$$

Remarks about implicit formulations of the Coriolis term are generally valid for the non-hydrostatic equations.

10.2 Temperature equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values.

$$X = T + \delta_{\mathrm{TR}} \frac{\alpha_{\mathrm{T}}\Phi_{\mathrm{s}}}{R_{\mathrm{d}}T_{\mathrm{st}}} \tag{162}$$

$$\mathcal{A} = \frac{R\tilde{T}}{c_{\mathrm{p}}} \frac{\omega}{\Pi} + \delta_{\mathrm{TR}} \frac{\alpha_{\mathrm{T}}}{R_{\mathrm{d}}T_{\mathrm{st}}} \mathbf{V}\nabla(\Phi_{\mathrm{s}}) + \delta_{\mathrm{TR}} \frac{\Phi_{\mathrm{s}}}{R_{\mathrm{d}}T_{\mathrm{st}}} \left(\dot{\eta} \frac{d\alpha_{\mathrm{T}}}{d\eta}\right) \tag{163}$$

$$\mathcal{L} = -\tau(\overline{M}^{2}D^{'}) \tag{164}$$

$$\mathcal{F} = \exp(-\kappa_{\mathrm{m}}\hat{Q}))F_{\mathrm{T}} \tag{165}$$

Top:

$$T_{\eta=0} = T_{l=1} \tag{166}$$

Bottom if $\delta m = 0$:

$$T_{\eta=1} = T_{l=L} \tag{167}$$

Bottom if $\delta m = 1$ (output of physics):

$$T_{\eta=1} = T_{\mathrm{s}} \tag{168}$$

10.3 Vertical divergence variable equation.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values, case NVDVAR=3, LGWADV=.F. .

$$X = d \tag{169}$$

$$\mathcal{A} = -d(d + \mathtt{X}_{\mathrm{S}}) - \frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}} \frac{\partial\left[\frac{dw}{dt}\right]_{\mathrm{ad}}}{\partial\eta} + \frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}} (\nabla w) \left(\frac{\partial\mathbf{V}}{\partial\eta}\right) \tag{170}$$

$$\mathcal{L} = -\frac{g^{2}}{R_{\mathrm{d}}T_{\mathrm{a}}^{*}} (\mathbf{L}_{\kappa}^{*}\hat{Q}) \tag{171}$$

$$\mathcal{F} = -\frac{d}{\left[\frac{\partial\Pi}{\partial\eta}\right]} F_{\mathrm{m}}^{'} - \left[\frac{g\Pi}{R\tilde{T}\frac{\partial\Pi}{\partial\eta}} \frac{\partial F_{\mathrm{w}}}{\partial\eta}\right] \tag{172}$$

Top:

$$d_{\eta=0} = d_{l=1} \tag{173}$$

Bottom if $\delta m = 0$:

$$d_{\eta=1} = d_{l=L} \tag{174}$$

Bottom if $\delta m = 1$: not yet coded.

$*$ Definition of X, $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{F}$, top and bottom values, case NVDVAR=3, LGWADV=.T. .

The linear system uses the d equation.

$$X = w \tag{175}$$

$$\mathcal{A} = \frac{g}{\kappa_{\rm m}} \left[\frac{\partial \Pi (\exp(\kappa_{\rm m} \hat{Q}) - 1)}{\partial \Pi} + (\kappa_{\rm m} - 1)(\exp(\kappa_{\rm m} \hat{Q}) - 1) \right] \tag{176}$$

$$\mathcal{F} = F_{\rm w} \tag{177}$$

Top:

- **LVERTFE**=.F.: $w_{\eta=0}$ is computed by the general formula giving w at half levels.
- **LVERTFE**=.T.: FD treatment for $w_{\eta=0}$ (cf. above).

Bottom if $\delta m = 0$:

$$w_{\eta=1} = \mathbf{V}_{\rm surf} \nabla z_{\rm surf} \tag{178}$$

Bottom if $\delta m = 1$: not yet coded.

10.4 Continuity equation and GFL equations.

Same discretisation as in the hydrostatic case, see parts (8.3), (8.4) and (8.5).

10.5 Case of lagged physics.

See part (8.6).

10.6 Quantities to be interpolated (computation under subroutine LACDYN).

See part (8.7).

11 $\mathcal{R}$ operator.

$*$ No tilting.

To transport a vector along a trajectory (part of a great circle) from an origin point O to a final point F the following operator $\mathcal{R}^{OF}$ is defined:

$$\mathbf{V}^{'} = \mathcal{R}^{OF}(\mathbf{V}) \tag{179}$$

where $\mathbf{V}^{'}$ has coordinates $(u^{'}, v^{'})$, $\mathbf{V}$ has coordinates (u, v), and the relationship between (u, v) and $(u^{'}, v^{'})$ is:

$$\begin{pmatrix} u^{'} \\ v^{'} \end{pmatrix} = \begin{pmatrix} p & q \\ -q & p \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{180}$$

where:

$$p = \frac{\mathbf{i}^F\mathbf{i}^O + \mathbf{j}^F\mathbf{j}^O}{1 + \mathbf{k}^F\mathbf{k}^O} = \frac{\cos\theta^F\cos\theta^O + (1 + \sin\theta^F\sin\theta^O)\cos(\lambda^F - \lambda^O)}{1 + \cos\phi} \tag{181}$$

$$q = \frac{\mathbf{i}^F\mathbf{j}^O + \mathbf{j}^F\mathbf{i}^O}{1 + \mathbf{k}^F\mathbf{k}^O} = \frac{(\sin\theta^F + \sin\theta^O)\sin(\lambda^F - \lambda^O)}{1 + \cos\phi} \tag{182}$$

(Denotations θ^O,θ^F,λ^O,λ^F,ϕ: see section 6.).

p and q verify the following identity:

$$p^2 + q^2 = 1 \tag{183}$$

Computation of p and q is done in subroutine **LARCHE**.

$*$ Tilting.

The coordinates of $\mathbf{V}^{'}$ and $\mathbf{V}$ are linked by the following relationship:

$$\begin{pmatrix} u^{'} \\ v^{'} \end{pmatrix} = \begin{pmatrix} GNORDM & GNORDL \\ -GNORDL & GNORDM \end{pmatrix} \begin{pmatrix} p & q \\ -q & p \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{184}$$

where:

$$\cos\alpha = \frac{2c}{A\cos\Theta^O}[\sin\theta_{\rm p}\cos\theta^O - \sin\theta^O\cos\theta_{\rm p}\cos(\lambda^O - \lambda_{\rm p})] \tag{185}$$

$$\sin\alpha = \frac{2c}{A\cos\Theta^O}[\cos\theta_{\rm p}\sin(\lambda^O - \lambda_{\rm p})] \tag{186}$$

$$A = (1 + c^2) + (1 - c^2)(\sin\theta_{\rm p}\sin\theta^O + \cos\theta_{\rm p}\cos\theta^O\cos(\lambda^O - \lambda_{\rm p})) \tag{187}$$

and where:

- c is the stretching coefficient.
- Θ^O is the latitude on the computational sphere of the origin point O.
- $(\theta_{\rm p}, \lambda_{\rm p})$ are the latitude and longitude on the geographical sphere of the stretching pole.
- p and q are computed like in the not tilted case (in subroutine **LARCHE**).
- $\cos\alpha$ and $\sin\alpha$ are also computed in subroutine **LARCHE**).
- $(GNORDL, GNORDM)$ are the coordinates in the computational sphere of the unit vector directed towards the true north, computed in subroutine **SUGEM2**.

$*$ Plane geometry (LAM models).

The curvature of the Earth is now taken into account in computing an operator $\mathcal{R}^{OF}$ in the routine **ELARCHE** instead of computing curvature terms. Expressions of p and q are different from the global model ones and are not detailed here.

12 Computation of longitudes and latitudes on the computational sphere.

For interpolations it is necessary to compute (Θ^O, Λ^O), latitude and longitude of the interpolation point O in the computational sphere. The iterative algorithm allowing to find O gives (θ^O, λ^O), latitude and longitude in the geographical sphere (more exactly $\sin\theta^O$, $\cos\theta^O\cos\lambda^O - \lambda^F$ and $\cos\theta^O\sin\lambda^O - \lambda^F$ where (θ^F, λ^F) are the coordinates of the final point on the geographical sphere). Transform formulae giving (Θ, Λ) on the computational sphere once knowing (θ, λ) on the geographical sphere are given by equations (188) to (190).

$$\sin\Theta = \frac{(1-c^2)+(1+c^2)(\sin\theta_{\rm p}\sin\theta+\cos\theta_{\rm p}\cos\theta\cos(\lambda-\lambda_{\rm p}))}{A} \tag{188}$$

$$\cos\Theta\cos\Lambda = \frac{2c(\cos\theta_{\rm p}\sin\theta-\sin\theta_{\rm p}\cos\theta\cos(\lambda-\lambda_{\rm p}))}{A} \tag{189}$$

$$\cos\Theta\sin\Lambda = \frac{2c\cos\theta\sin(\lambda-\lambda_{\rm p})}{A} \tag{190}$$

where:

- $A = (1+c^2)+(1-c^2)(\sin\theta_{\rm p}\sin\theta+\cos\theta_{\rm p}\cos\theta\cos(\lambda-\lambda_{\rm p}))$
- c is the stretching coefficient.
- $(\theta_{\rm p}, \lambda_{\rm p})$ are the latitude and longitude on the geographical sphere of the stretching pole.
- Computation of Θ, Λ is done in subroutine **LARCHE**.

∗ **Plane geometry (LAM models):** The SL trajectory is already computed on the computational grid, so equivalent transformation formulae from geographical space to computational space are useless.

13 Computation of $\dot\eta$ at full levels.

∗ **General expression:** $\dot\eta$ is needed to find the height of the medium and origin points.
$\dot\eta$ can be written:

$$\dot\eta = \left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)\frac{\partial\eta}{\partial\Pi} \tag{191}$$

∗ **Discretisation at full levels for LVERTFE=.F.:** $\left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)$ is provided at half levels, and $\Delta\eta$ and $\Delta\Pi$ are provided at full levels. Discretisation of (191) is:

$$\dot\eta_l = 0.5\left[\left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)_{\bar l} + \left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)_{\bar l-1}\right]\frac{[\Delta\eta]_l}{[\Delta\Pi]_l} \tag{192}$$

∗ **Discretisation at full levels for LVERTFE=.T.:** $\left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)$ is provided at full levels, and $\Delta\eta$ and $\Delta\Pi$ are provided at full levels. Discretisation of (191) is:

$$\dot\eta_l = \left(\dot\eta\frac{\partial\Pi}{\partial\eta}\right)_l\frac{[\Delta\eta]_l}{[\Delta\Pi]_l} \tag{193}$$

14 Interpolations and weights computations.

14.1 Interpolation grid and weights (subroutine LASCAW).

14.1.1 Horizontal interpolation grid and weights for bi-linear interpolations.

∗ **Definitions:** A 16 points horizontal grid is defined as it is shown in figure 14.1, but only 4 of these 16 points are used in the interpolations. The interpolation point O (medium or origin point) is between B_1, C_1, B_2 and C_2. Λ and Θ are the longitudes and latitudes on the computational sphere. Linear weights are defined as follows:

- zonal weights for latitudes 1 and 2:
  $ZDLO1 = (\Lambda_O - \Lambda_{B_1})/(\Lambda_{C_1} - \Lambda_{B_1})$ and $ZDLO2 = (\Lambda_O - \Lambda_{B_2})/(\Lambda_{C_2} - \Lambda_{B_2})$.
- meridian weight: $ZDLAT = (\Theta_O - \Theta_{B_1})/(\Theta_{B_2} - \Theta_{B_1})$

∗ **Computations:**

- The weights $ZDLO1$ and $ZDLO2$ are computed then stored in the array **PDLO**.
- The weight $ZDLAT$ is computed then stored in the array **PDLAT**.
- The memory address (in SL arrays) of the data concerning the points A_1 and A_2 are computed then stored in the array **KL0** or **KLH0**. Memory address of the data concerning the points B_1, B_2, C_1 and C_2 can be easily computed in interpolations routines knowing these ones of A_1 and A_2.
- Interpolations use data of points B_1, B_2, C_1 and C_2.

14.1.2 Vertical interpolation grid and weights for vertical linear interpolations.

∗ **Definitions:** A 4 points vertical grid is defined as it is shown in figure 14.2, but only 2 of these 4 points are used in the interpolations. The interpolation point O (medium or origin point) is between T_{l+1} and T_{l+2}. The vertical weight is defined by: $ZDVER = (\eta_O - \eta_{T_{l+1}})/(\eta_{T_{l+2}} - \eta_{T_{l+1}})$

∗ **Computations:**

- The weight $ZDVER$ is computed then stored in the array **PDVER**.
- The level number l of T_l is stored in the array **KLEV**.
- Interpolations use data of points T_{l+1} and T_{l+2}.

∗ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.1.3 Horizontal interpolation grid and weights for 12 points cubic interpolations.

∗ **Definitions:** A 16 points horizontal grid is defined as it is shown in figure 14.3, but only 12 of these 16 points are used in the interpolations. The interpolation point O (medium or origin point) is between B_1, C_1, B_2 and C_2. The following weights are defined as follows:

- zonal linear weights for latitudes 0, 1, 2, 3:

$$ZDLO0 = (\Lambda_O - \Lambda_{B_0})/(\Lambda_{C_0} - \Lambda_{B_0})$$
$$ZDLO1 = (\Lambda_O - \Lambda_{B_1})/(\Lambda_{C_1} - \Lambda_{B_1})$$
$$ZDLO2 = (\Lambda_O - \Lambda_{B_2})/(\Lambda_{C_2} - \Lambda_{B_2})$$
$$ZDLO3 = (\Lambda_O - \Lambda_{B_3})/(\Lambda_{C_3} - \Lambda_{B_3})$$

- zonal cubic weights for latitude 1:

$$ZCLO11 = f_1(ZDLO1)$$
$$ZCLO12 = f_2(ZDLO1)$$
$$ZCLO13 = f_3(ZDLO1)$$

  where:

  * $f_1(\alpha) = (\alpha + 1)(\alpha - 2)(\alpha - 1)/2$
  * $f_2(\alpha) = -(\alpha + 1)(\alpha - 2)\alpha/2$
  * $f_3(\alpha) = \alpha(\alpha - 1)(\alpha + 1)/6$

- zonal cubic weights for latitude 2:

$$ZCLO21 = f_1(ZDLO2)$$
$$ZCLO22 = f_2(ZDLO2)$$
$$ZCLO23 = f_3(ZDLO2)$$

- meridian cubic weights:

$$ZCLA1 = [(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_2})(\Theta_O - \Theta_{B_3})]/[(\Theta_{B_1} - \Theta_{B_0})(\Theta_{B_1} - \Theta_{B_2})(\Theta_{B_1} - \Theta_{B_3})]$$

$$ZCLA2 = [(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_3})]/[(\Theta_{B_2} - \Theta_{B_0})(\Theta_{B_2} - \Theta_{B_1})(\Theta_{B_2} - \Theta_{B_3})]$$

$$ZCLA3 = [(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_2})]/[(\Theta_{B_3} - \Theta_{B_0})(\Theta_{B_3} - \Theta_{B_1})(\Theta_{B_3} - \Theta_{B_2})]$$

∗ Computations:

- The zonal linear weights $ZDLO0$, $ZDLO1$, $ZDLO2$ and $ZDLO3$ are computed then stored in the array **PDLO**.
- The zonal cubic weights $ZCLO11$, $ZCLO12$, $ZCLO13$, $ZCLO21$, $ZCLO22$, $ZCLO23$ are computed then stored in the array **PCLO**.
- The meridian cubic weights $ZCLA1$, $ZCLA2$ and $ZCLA3$ are computed then stored in the array **PCLA**. One can notice that the denominators of $ZCLA1$, $ZCLA2$ and $ZCLA3$ do not depend on coordinates of O and can be pre-computed in array **RIPI**.
- The memory address (in SL arrays) of the data concerning the points A_0, A_1, A_2 and A_3 are stored in the array **KL0** or **KLH0**. Once knowing these addresses one can easily retrieve the other points addresses in the interpolations routines.
- Interpolations use data of the following 12 points B_0, C_0, A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2, B_3 and C_3.

∗ Extension to diffusive interpolations (SLHD): The way to take account of the diffusive properties of interpolations has been redesigned, and now it is completely contained in the calculation of cubic weights: we pass from conventional cubic interpolations to diffusive SLHD cubic interpolations simply by changing the way of computing the cubic weights. The comprehensive way of computing the SLHD cubic weights is not provided in this documentation (because it leads to rather tricky formulae), but details about the calculations can be found in documentation (IDSLIF2) and looking in routine **LASCAW** (**ELASCAW** in LAM models). We just give a sum-up of the way to compute the cubic weights.

- Meridian diffusive cubic weights:

$$ZCLA1_{\text{slhd}} = ZCLA1 + ZINCRM1$$

$$ZCLA2_{\text{slhd}} = ZCLA2 + ZINCRM2$$

$$ZCLA3_{\text{slhd}} = ZCLA3 + ZINCRM3$$

  where each of increments $ZINCRM1$, $ZINCRM2$ and $ZINCRM3$ is a linear combination of cubic Lagrangian weights $ZCLA1$, $ZCLA2$ and $ZCLA3$ and diffusive weights (linear for **LSLHD_OLD**=.T., quadratic otherwise).

  Coefficients of these linear combinations depend on:

  - Constants in time C_{sldw}, describing Laplacian smoother.
  - A pre-computed quantity κ (computed in **GP_KAPPA**) depending on horizontal flow deformation, ranging from 0 to 1; κ is equal to 1 if **LSLHD_STATIC**=T. An alternate expression of κ is computed in **GP_KAPPAT** if different coefficients are asked for thermic variables.
  - Lower and upper bounds ($\kappa_{\min}$ and $\kappa_{\max}$) respectively stored in variables **SLHDKMIN** and **SLHDKMAX**, used to construct limit interpolators (0 - cubic Lagrange, 1 - linear/quadratic; they are not restricted to range 0-1).

- Zonal diffusive cubic weights for latitude number 1:

$$ZCLO11_{\text{slhd}} = ZCLO11 + ZINCRL11$$

$$ZCLO12_{\text{slhd}} = ZCLO12 + ZINCRL12$$

$$ZCLO13_{\text{slhd}} = ZCLO13 + ZINCRL13$$

  where each of increments $ZINCRL11$, $ZINCRL12$ and $ZINCRL13$ is a linear combination of cubic Lagrangian weights $ZCLO11$, $ZCLO12$ and $ZCLO13$ and diffusive weights (linear for **LSLHD_OLD**=.T., quadratic otherwise).

  Coefficients of these linear combinations depend on:

  - Constants like C_{slhdepsh}.
  - κ (see above).
  - Lower and upper bounds $\kappa_{\min}$ and $\kappa_{\max}$ (see above).

- Zonal diffusive cubic weights for latitude number 2:

$$ZCLO21_{\text{slhd}} = ZCLO21 + ZINCRL21$$

$$ZCLO22_{\text{slhd}} = ZCLO22 + ZINCRL22$$
$$ZCLO23_{\text{slhd}} = ZCLO23 + ZINCRL23$$

  where each of increments $ZINCRL21$, $ZINCRL22$ and $ZINCRL23$ is a linear combination of cubic Lagrangian weights $ZCLO21$, $ZCLO22$ and $ZCLO23$ and diffusive weights (linear for **LSLHD_OLD**=.T., quadratic otherwise).
  Coefficients of these linear combinations depend on C_{slddepsh}, κ, $\kappa_{\min}$, $\kappa_{\max}$ like the zonal weights for latitude number 1.

Several options of SLHD smoothing are available, according to values of the keys **LSLHDQUAD** and **LSLHD_OLD**.

In part 14.2, cubic interpolations are written with conventional cubic weights: replace conventional cubic weights by their SLHD counterpart for diffusive cubic interpolations.

∗ **Extension to 3D-turbulence:** not described in detail; additional cubic weights are computed if **L3DTURB**=T.

14.1.4 Vertical interpolation grid and weights for vertical cubic 4 points interpolations.

A 4 points vertical grid is defined as it is shown in figure 14.2. The interpolation point O (medium or origin point) is between T_{l+1} and T_{l+2}. The vertical weights are defined by:

$$ZCVE1 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+2}})(\eta_O - \eta_{T_{l+3}})]/[(\eta_{T_{l+1}} - \eta_{T_l})(\eta_{T_{l+1}} - \eta_{T_{l+2}})(\eta_{T_{l+1}} - \eta_{T_{l+3}})]$$

$$ZCVE2 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+1}})(\eta_O - \eta_{T_{l+3}})]/[(\eta_{T_{l+2}} - \eta_{T_l})(\eta_{T_{l+2}} - \eta_{T_{l+1}})(\eta_{T_{l+2}} - \eta_{T_{l+3}})]$$
$$ZCVE3 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+1}})(\eta_O - \eta_{T_{l+2}})]/[(\eta_{T_{l+3}} - \eta_{T_l})(\eta_{T_{l+3}} - \eta_{T_{l+1}})(\eta_{T_{l+3}} - \eta_{T_{l+2}})]$$

∗ **Computations:**

- The vertical weights $ZCVE1$, $ZCVE2$ and $ZCVE3$ are computed then stored in the array **PVINTW**. One can notice that the denominators of $ZCVE1$, $ZCVE2$ and $ZCVE3$ do not depend on coordinates of O and can be pre-computed in the array **VCUICO**.
- The level number l of T_l is stored in the array **KLEV**.
- Interpolations use data of points T_l, T_{l+1}, T_{l+2} and T_{l+3}.

∗ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$, and use array **VCUICOH**.

∗ **Extension to diffusive interpolations (SLHD):** Like we do for horizontal weights, the semi-Lagrangian diffusion can be taken into account by modifying vertical cubic weights as follow:

$$ZCVE1_{\text{slhd}} = ZCVE1 + ZINCRV1$$

$$ZCVE2_{\text{slhd}} = ZCVE2 + ZINCRV2$$
$$ZCVE3_{\text{slhd}} = ZCVE3 + ZINCRV3$$

Expressions giving vertical increments $ZINCRV1$ to $ZINCRV3$ have a shape similar to those of horizontal increments, and still depend on quantities like κ, $\kappa_{\min}$, $\kappa_{\max}$.

In part 14.2, cubic interpolations are written with conventional cubic weights: replace conventional cubic weights by their SLHD counterpart for diffusive cubic interpolations.

14.1.5 Vertical interpolation grid and weights for vertical cubic Hermite interpolations.

This part is described for interpolations of full level data but can be extended for interpolations of half level data.

A 4 points vertical grid is defined as it is shown in figure 14.2. The interpolation point O (medium or origin point) is between T_{l+1} and T_{l+2}.

First weights to compute vertical derivatives at layers $l+1$ and $l+2$ are computed. For a variable X, $\frac{\partial X}{\partial \eta}$ is computed as close as possible as $\left(\dot{\eta}\frac{\partial X}{\partial \eta}\right)/\dot{\eta}$, but with additional approximations allowing to avoid horizontal interpolations for term $\left(\dot{\eta}\frac{\partial \Pi}{\partial \eta}\right)$.

General formula to evaluate $\frac{\partial X}{\partial \eta}_{l+1}$ is:

$$\left(\frac{\partial X}{\partial \eta}\right)_{l+1} = \frac{(X_{l+2} - X_l)}{\eta_{l+2} - \eta_l} \tag{194}$$

with particuliar treatment for $l = 1$ and $l = L$.

General formula to evaluate the following weights are:

$$VDERW11 = 0.5(\eta_{l+2} - \eta_{l+1})/(\eta_{l+2} - \eta_l)$$
$$VDERW21 = 0.5(\eta_{l+2} - \eta_{l+1})/(\eta_{l+2} - \eta_l)$$
$$VDERW12 = 0.5(\eta_{l+2} - \eta_{l+1})/(\eta_{l+3} - \eta_{l+1})$$
$$VDERW22 = 0.5(\eta_{l+2} - \eta_{l+1})/(\eta_{l+3} - \eta_{l+1})$$

with particuliar treatment when the interpolation point is near the top or the bottom boundary.

∗ Computations:

- The vertical weights $VDERW11$, $VDERW21$, $VDERW12$ and $VDERW22$ are computed then stored in the array **PVDERW**.
- The weight $ZDVER$ is computed then stored in the array **PDVER** (see subsubsection 14.1.2).
- Functions $f_{\mathrm{H1}}(ZDVER)$ to $f_{\mathrm{H4}}(ZDVER)$ (involved in any Hermite cubic interpolation), where:
  - $f_{\mathrm{H1}}(\alpha) = (1-\alpha)^2(1+2\alpha)$
  - $f_{\mathrm{H2}}(\alpha) = \alpha^2(3-2\alpha)$
  - $f_{\mathrm{H3}}(\alpha) = \alpha(1-\alpha)^2$
  - $f_{\mathrm{H4}}(\alpha) = -\alpha^2(1-\alpha)$

  are computed and stored in array **PHVW**.
- The level number l of T_l is stored in the array **KLEV**.
- Interpolations use data of points T_l, T_{l+1}, T_{l+2} and T_{l+3}.

14.1.6 Vertical interpolation grid and weights for vertical cubic spline interpolations.

A 4 points vertical grid is defined as it is shown in figure 14.2. The interpolation point O (medium or origin point) is between T_{l+1} and T_{l+2}.

∗ Computations:

- Vertical interpolation is the product of two operators. The first one uses all the layers and is done in the unlagged part of the grid-point calculations by a routine **VSPLTRANS** and needs to compute top and bottom values and vertical derivatives of the field to be interpolated, and also the inversion of a tridiagonal matrix; for this operation it is necessary to use some coefficients stored in the arrays **RVSPTRI** and **RVSPC** and pre-computed in the set-up subroutine **SUVSPLIP**; the original field is stored in the buffer **P(X)L9** and the intermediate result after this first part is stored in the buffer **P(X)SPL9**. The second part uses 4 points and is done in the interpolation routine itself.
- The vertical weights $ZCVE0$, $ZCVE1$, $ZCVE2$ and $ZCVE3$ necessary for the second part are computed then stored in the array **PVINTWS**. Some part of the calculations can be pre-computed in the set-up subroutine **SUVSPLIP** (array **RFVV**).
- The level number l of T_l is stored in the array **KLEV**.
- Interpolations use data of points T_l, T_{l+1}, T_{l+2} and T_{l+3}, stored in the intermediate array **P(X)SPL9**.

14.1.7 Interpolation grid and weights for tri-linear interpolations.

A 64 points grid is defined as it is shown in figure 14.4, but only 8 of these 64 points are used in the interpolations. The interpolation point O (medium or origin point) is between $B_{1,l+1}$, $C_{1,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $B_{1,l+2}$, $C_{1,l+2}$, $B_{2,l+2}$ and $C_{2,l+2}$. For the two levels $l+1$ and $l+2$ see subsubsection 14.1.1 corresponding to bi-linear horizontal interpolations for weights computations. For weights needed for vertical interpolations ($ZDVER$) see subsubsection 14.1.2 corresponding to linear vertical interpolations.

- The memory address (in SL arrays) of the data concerning the points $A_{0,l}$, $A_{1,l}$, $A_{2,l}$ and $A_{3,l}$ is computed then stored in the array **KL0**. Once knowing these addresses one can easily retrieve the other points addresses in the interpolations routines.
- Interpolations use data of points $B_{1,l+1}$, $C_{1,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $B_{1,l+2}$, $C_{1,l+2}$, $B_{2,l+2}$ and $C_{2,l+2}$.

∗ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.1.8 Interpolation grid and weights for 32 points interpolations.

A 64 points grid is defined as it is shown in figure 14.4, but only 32 (48 if vertical spline cubic interpolations) of these 64 points are used in the interpolations. The interpolation point O (medium or origin point) is between $B_{1,l+1}$, $C_{1,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $B_{1,l+2}$, $C_{1,l+2}$, $B_{2,l+2}$ and $C_{2,l+2}$. For the two levels l and $l+3$ see subsubsection 14.1.1 corresponding to bi-linear horizontal interpolations for weights computations. For the two levels $l+1$ and $l+2$ see subsubsection 14.1.3 corresponding to 12 points horizontal interpolations for weights computations. For weights needed for vertical interpolations see subsubsection 14.1.4 for vertical cubic interpolations, see subsubsection 14.1.5 for vertical Hermite cubic interpolations, see subsubsection 14.1.6 for vertical spline cubic interpolations.

- The memory address (in SL arrays) of the data concerning the points $A_{0,l}$, $A_{1,l}$, $A_{2,l}$ and $A_{3,l}$ is computed then stored in the array **KL0**. Once knowing these addresses one can easily retrieve the other points addresses in the interpolations routines.
- Interpolations use data of the following 32 points $B_{1,l}$, $C_{1,l}$, $B_{2,l}$, $C_{2,l}$, $B_{0,l+1}$, $C_{0,l+1}$, $A_{1,l+1}$, $B_{1,l+1}$, $C_{1,l+1}$, $D_{1,l+1}$, $A_{2,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $D_{2,l+1}$, $B_{3,l+1}$, $C_{3,l+1}$, $B_{0,l+2}$, $C_{0,l+2}$, $A_{1,l+2}$, $B_{1,l+2}$, $C_{1,l+2}$, $D_{1,l+2}$, $A_{2,l+2}$, $B_{2,l+2}$, $C_{2,l+2}$, $D_{2,l+2}$, $B_{3,l+2}$, $C_{3,l+2}$, $B_{1,l+3}$, $C_{1,l+3}$, $B_{2,l+3}$ and $C_{2,l+3}$.

∗ **Remarks:**

- The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.
- For vertical spline cubic interpolations a 12 points grid is used on each level l, $l+1$, $l+2$, $l+3$ (total = 48 points used).

14.1.9 Horizontal interpolation grid and weights for 16 points linear least-square fit interpolations.

A 16 points horizontal grid is defined as it is shown in figure 14.3. The interpolation point O (medium or origin point) is between B_1, C_1, B_2 and C_2. The interpolation is replaced by a linear least-square fit minimisation of a first order polynomial in each direction (first zonal interpolations, then meridian interpolations).
The weights used are the same ones as for the 4 points bilinear horizontal interpolation, but the zonal linear weights are required for the 4 latitudes of the 16 points grid ($ZDLO0$, $ZDLO1$, $ZDLO2$, $ZDLO3$, $ZDLAT$). The actual weights used in one direction (for example the meridian direction) are respectively: $0.4-0.3ZDLAT$, $0.3-0.1ZDLAT$, $0.2+0.1ZDLAT$, $0.1+0.3ZDLAT$.

∗ **Computations:**

- The weights $ZDLO0$, $ZDLO1$, $ZDLO2$ and $ZDLO3$ are computed then stored in the array **PDLO**.
- The meridian weight $ZDLAT$ is computed then stored in the array **PDLAT**.
- The memory address (in SL arrays) of the data concerning the points A_0, A_1, A_2 and A_3 are stored in the array **KL0** or **KLH0**. Once knowing these addresses one can easily retrieve the other points addresses in the interpolations routines.
- Interpolations use data of the following 16 points A_0, B_0, C_0, D_0, A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2, A_3, B_3, C_3, D_3.

14.1.10 Horizontal interpolation grid and weights for 32 points linear least-square fit interpolations.

A 32 points grid is defined as it is shown in figure 14.4 where only the two intermediate layers are retained. The interpolation point O (medium or origin point) is between $B_{1,l+1}$, $C_{1,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $B_{1,l+2}$, $C_{1,l+2}$, $B_{2,l+2}$ and $C_{2,l+2}$. For the two levels $l+1$ and $l+2$ see subsubsection 14.1.9 corresponding to 16 points linear least-square fit horizontal interpolations for weights computations. For weights needed for vertical interpolations, only the linear weight $ZDVER$ is required; see subsubsection 14.1.2 for vertical linear interpolations.

- The memory address (in SL arrays) of the data concerning the points $A_{1,l}$ and $A_{2,l}$ is computed then stored in the array **KL0**. Once knowing these addresses one can easily retrieve the other points addresses in the interpolations routines.
- Interpolations use data of the following 32 points $A_{0,l+1}$, $B_{0,l+1}$, $C_{0,l+1}$, $D_{0,l+1}$, $A_{1,l+1}$, $B_{1,l+1}$, $C_{1,l+1}$, $D_{1,l+1}$, $A_{2,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $D_{2,l+1}$, $A_{3,l+1}$, $B_{3,l+1}$, $C_{3,l+1}$, $D_{3,l+1}$, $A_{0,l+2}$, $B_{0,l+2}$, $C_{0,l+2}$, $D_{0,l+2}$, $A_{1,l+2}$, $B_{1,l+2}$, $C_{1,l+2}$, $D_{1,l+2}$, $A_{2,l+2}$, $B_{2,l+2}$, $C_{2,l+2}$, $D_{2,l+2}$, $A_{3,l+2}$, $B_{3,l+2}$, $C_{3,l+2}$, $D_{3,l+2}$.

∗ **Remarks:**

- The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.1.11 Modified weights for COMAD interpolations.

Name COMAD stands for "continuous mapping around departure points".

Any linear weight ZDL is modified as follows, to take account of convergence:

$$ZDL_{\text{comad}} = ZDL + q(ZDL) * (ZDL - 0.5) * (STDDIS - 1)$$

where $STDDIS$ is the stretching/shrinking deformation along interpolation direction:

$$STDDIS = 1 + (dV/dx)Dt$$

- $STDDIS$ is then bounded by 0.0001 and 1.
- dV/dx is a generic denotation for zonal, meridian or vertical first-order wind derivative.
- $q(ZDL)$ is by default 1 on the whole interval $[0, 1]$. To ensure continuity of interpolations, $q(ZDL)$ can be taken as a compact support C-∞ function, close to 1 on $[0, 1]$, and matching $q(0) = 0$, $q(0.5) = 1$, $q(1) = 0$; For example the following expression could be used for q:

$$q(ZDL) = \exp(0.0000001(1 - 1/(1 - (2ZDL - 1)^2)))$$

Modified high-order weights are computed using COMAD linear weights.

This scheme is described in more details in (Malardel and Ricard, 2015).

14.1.12 Plane geometry (LAM models).

All previous formulae for weight computation can be used for an irregular latitude spacing and a different number of points on each longitude. The LAM model grid has a horizontal regular spacing, so the previous formulae can be simplified and array **RIPI** is no longer necessary. **ELASCAW** is called instead of **LASCAW** and is cheaper in CPU time.

14.2 Interpolations.

14.2.1 Bilinear interpolation (subroutine LAIDLI).

See figure 14.1 and subsubsection 14.1.1 for definition of $ZDLO1$, $ZDLO2$, $ZDLAT$ and points B_1, C_1, B_2 and C_2.

For a quantity X, are computed successively:

- a linear interpolation on the longitude number 1: $X_1 = X_{B_1} + ZDLO1(X_{C_1} - X_{B_1})$.
- a linear interpolation on the longitude number 2: $X_2 = X_{B_2} + ZDLO2(X_{C_2} - X_{B_2})$.
- a meridian linear interpolation: $X_{\text{interpolated}} = X_1 + ZDLAT(X_2 - X_1)$.

14.2.2 Tri-linear interpolation (subroutine LAITLI).

For layers $l + 1$ and $l + 2$ (see figure 14.4) bilinear horizontal interpolations give two interpolated values X_{l+1} and X_{l+2} (see subsubsection 14.2.1). Then the final interpolated value is given by the following expression:

$$X_{\text{interpolated}} = X_{l+1} + ZDVER(X_{l+2} - X_{l+1})$$

$*$ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.2.3 Horizontal 12 points interpolation (subroutine LAIDDI).

See figure 14.3 and subsubsection 14.1.3 for definition of $ZDLO0$, $ZDLO1$, $ZDLO2$, $ZDLO3$, $ZCLA1$, $ZCLA2$ and $ZCLA3$ and points B_0, C_0, A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2, B_3 and C_3.

For a quantity X, are computed successively:

- a linear interpolation on the longitude number 0:
  $X_0 = X_{B_0} + ZDLO0(X_{C_0} - X_{B_0})$.
- a cubic 4 points interpolation on the longitude number 1:
  $X_1 = X_{A_1} + ZCLO11(X_{B_1} - X_{A_1}) + ZCLO12(X_{C_1} - X_{A_1}) + ZCLO13(X_{D_1} - X_{A_1})$.
- a cubic 4 points interpolation on the longitude number 2:
  $X_2 = X_{A_2} + ZCLO21(X_{B_2} - X_{A_2}) + ZCLO22(X_{C_2} - X_{A_2}) + ZCLO23(X_{D_2} - X_{A_2})$.
- a linear interpolation on the longitude number 3:
  $X_3 = X_{B_3} + ZDLO3(X_{C_3} - X_{B_3})$.
- a meridian cubic 4 points interpolation:
  $X_{\text{interpolated}} = X_0 + ZCLA1(X_1 - X_0) + ZCLA2(X_2 - X_0) + ZCLA3(X_3 - X_0)$.

There is a shape-preserving option: after cubic 4 points interpolations on longitudes number 1 and 2, X_1 is bounded between X_{B_1}, and X_{C_1} and X_2 is bounded between X_{B_2} and X_{C_2}; after meridian cubic 4 points interpolation $X_{\text{interpolated}}$ is bounded between X_1 and X_2. Use of switches **LQMW** (momentum equation), **LQMT** (temperature equation), **LQMP** (continuity equation), **LQMSPD** (pressure departure variable equation), **LQMSVD** (vertical divergence equation), **Y[X]_NL%LQM** for GFL variables allow to use shape-preserving option.

14.2.4 Cubic 4 points vertical interpolation.

See figure 14.2 and subsubsection 14.1.4 for definition of $ZCVE1$, $ZCVE2$ and $ZCVE3$. The cubic 4 points vertical interpolation gives the final interpolated value:

$$X_{\text{interpolated}} = X_l + ZCVE1(X_{l+1} - X_l) + ZCVE2(X_{l+2} - X_l) + ZCVE3(X_{l+3} - X_l)$$

∗ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.2.5 Cubic Hermite vertical interpolation.

This part is valid for interpolations of full level variables. See figure 14.2 and subsubsection 14.1.5 for definition of $VDERW11$, $VDERW21$, $VDERW12$ and $VDERW22$. See subsubsection 14.1.2 for definition of $ZDVER$. See subsubsection 14.1.5 for definition of functions f_{H1} to f_{H4}. The cubic Hermite vertical interpolation gives the final interpolated value:

$$\begin{aligned} X_{\text{interpolated}} &= f_{\text{H1}}(ZDVER)X_{l+1} + f_{\text{H2}}(ZDVER)X_{l+2} \\ &+ f_{\text{H3}}(ZDVER)(VDERW11(X_{l+1} - X_l) + VDERW21(X_{l+2} - X_{l+1})) \\ &+ f_{\text{H4}}(ZDVER)(VDERW12(X_{l+2} - X_{l+1}) + VDERW22(X_{l+3} - X_{l+2})) \end{aligned}$$

14.2.6 Spline cubic 4 points vertical interpolation.

This part is valid for interpolations of full level variables. See figure 14.2 and subsubsection 14.1.6 for definition of $ZCVE0$, $ZCVE1$, $ZCVE2$ and $ZCVE3$. The spline cubic 4 points vertical interpolation gives the final interpolated value:

$$X_{\text{interpolated}} = ZCVE0 * XRP_l + ZCVE1 * XRP_{l+1} + ZCVE2 * XRP_{l+2} + ZCVE3 * XRP_{l+3}$$

where XRP is the re-profiled version of X available in **P(X)SPL9**. A monotonic constraint can be added, bounding $X_{\text{interpolated}}$ between X_{l+1} and X_{l+2}.

14.2.7 32 points 3D interpolation with vertical cubic 4 points interpolation (subroutine LAITRI).

For layers l and $l+3$ (see figure 14.4) bilinear horizontal interpolations give two interpolated values X_l and X_{l+3} (see subsubsection 14.2.1). For layers $l+1$ and $l+2$ (see figure 14.4) 12 points horizontal interpolations give two interpolated values X_{l+1} and X_{l+2} (see subsubsection 14.2.3). The final interpolated value $X_{\text{interpolated}}$ is a cubic 4 points vertical interpolation of X_l, X_{l+1}, X_{l+2} and X_{l+3} (see subsubsection 14.2.4).

There are shape-preserving options for horizontal or both horizontal and vertical interpolations.
Use of switches **LQMW** (momentum equation), **LQMT** (temperature equation), **LQMP** (continuity equation), **LQMSPD** (pressure departure variable equation), **LQMSVD** (vertical divergence equation), **Y[X]_NL%LQM** (GFL equations), allows to use shape-preserving option for both horizontal and vertical interpolations.
Use of switches **LQMHW** (momentum equation), **LQMHT** (temperature equation), **LQMHP** (continuity equation), **LQMHSPD** (pressure departure variable equation), **LQMHSVD** (vertical divergence equation), **Y[X]_NL%LQMH** (GFL equations), allows to use shape-preserving option for horizontal interpolations.

∗ **Remark:** The same formulae and computations are valid for half level data, simply replace the layer index l by the half level index $\bar{l}$.

14.2.8 32 points 3D interpolation with vertical cubic Hermite interpolation (subroutine LAIHVT).

This part is valid for interpolations of full level variables. For layers l and $l+3$ (see figure 14.4) bilinear horizontal interpolations give two interpolated values X_l and X_{l+3} (see subsubsection 14.2.1). For layers $l+1$ and $l+2$ (see figure 14.4) 12 points horizontal interpolations give two interpolated values X_{l+1} and X_{l+2} (see subsubsection 14.2.3). The final interpolated value $X_{\text{interpolated}}$ is a cubic Hermite vertical interpolation of X_l, X_{l+1}, X_{l+2} and X_{l+3} (see subsubsection 14.2.5).

There are shape-preserving options for horizontal or both horizontal and vertical interpolations.
Use of switch **Y[X]_NL%LQM** (GFL equations), allows to use shape-preserving option for both horizontal and vertical interpolations.
Use of switch **Y[X]_NL%LQMH** (GFL equations), allows to use shape-preserving option for horizontal interpolations.

14.2.9 48 points 3D interpolation with vertical cubic spline interpolation (subroutine LAITVSPCQM).

This part is valid for interpolations of full level variables. This type of interpolation is activated for GFL variables when **Y[X]_NL%LVSPLIP**=.T. in **NAMGFL** (currently for ozone only). Contrary to what is done for the other 32 points interpolations routines, the vertical interpolations are performed first. 12 vertical interpolations are done on the verticals matching the following points: $B_{0,l+1}$, $C_{0,l+1}$, $A_{1,l+1}$, $B_{1,l+1}$, $C_{1,l+1}$, $D_{1,l+1}$, $A_{2,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $D_{2,l+1}$, $B_{3,l+1}$, $C_{3,l+1}$. A monotonic constraint is added for the lower levels (currently the 9 lower levels). The projection horizontal plane on the level of the interpolation point provides a 12-points grid. A 12 points interpolation is done on this projection (see part 14.2.3). A final monotonic constraint is added: the interpolated value of X is bounded between the value of X at the points $B_{1,l+1}$, $C_{1,l+1}$, $B_{2,l+1}$, $C_{2,l+1}$, $B_{1,l+2}$, $C_{1,l+2}$, $B_{2,l+2}$, $C_{2,l+2}$, and the overshoots/undershoots found are dispatched on upper levels to ensure as possible it can be done conservation properties.

14.2.10 Horizontal 16 points linear least-square fit interpolation.

See figure 14.3 and subsubsection 14.1.9 for definition of $ZDLO0$, $ZDLO1$, $ZDLO2$, $ZDLO3$, $ZDLAT$ and points A_0, B_0, C_0, D_0, A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2, A_3, B_3, C_3, D_3. Let us define:

- $f_1(\alpha) = 0.4 - 0.3\alpha$
- $f_2(\alpha) = 0.3 - 0.1\alpha$
- $f_3(\alpha) = 0.2 + 0.1\alpha$
- $f_4(\alpha) = 0.1 + 0.3\alpha$

For a quantity X, are computed successively:

- a linear least-square fit 4 points interpolation on the longitude number lon for $lon = 0, 1, 2, 3$:
  $X_{lon} = +f_1(ZDLO_{lon})X_{A_{lon}} + f_2(ZDLO_{lon})X_{B_{lon}} + f_3(ZDLO_{lon})X_{C_{lon}} + f_4(ZDLO_{lon})X_{D_{lon}}$.
- a meridian linear least-square fit 4 points interpolation:
  $X_{\text{interpolated}} = +f_1(ZDLAT)X_0 + f_2(ZDLAT)X_1 + f_3(ZDLAT)X_2 + f_4(ZDLAT)X_3$.

14.2.11 32 points 3D interpolation with linear least-square fit horizontal interpolations and vertical linear interpolations (subroutine LAISMOO).

For layers $l+1$ and $l+2$ (see figure 14.4) 16 points linear least-square fit horizontal interpolations give two interpolated values X_{l+1} and X_{l+2} (see subsubsection 14.2.10). The final interpolated value is then given by the following expression:

$$X_{\text{interpolated}} = X_{l+1} + ZDVER(X_{l+2} - X_{l+1})$$

∗ **Remark:** In the current usage of this interpolation (for $\dot{\eta}$ in the upper stratosphere) there is an additional smoothing done in routine **LAISMOA** before the interpolation by **LAISMOO**.

∗ **For more information:** See the internal paper (IDSVTSM).

14.3 Code structures to store weights.

Two structures have been created in module intdynsl_mod.F90:

- structure **TLSCAW** for linear weights.
- structure **TRSCAW** for non-linear weights.

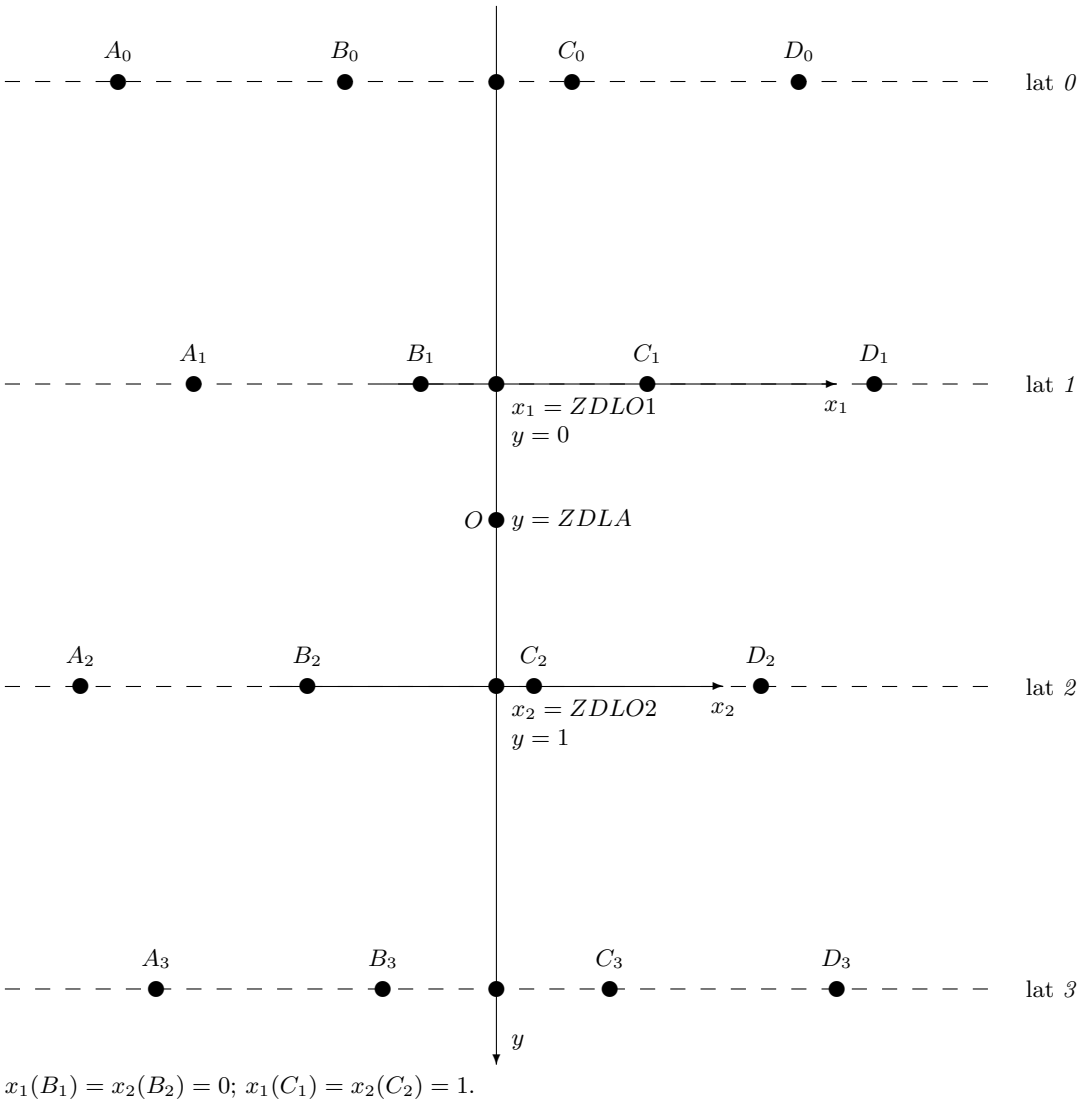


Figure 14.1: Interpolation horizontal grid for bilinear interpolations.

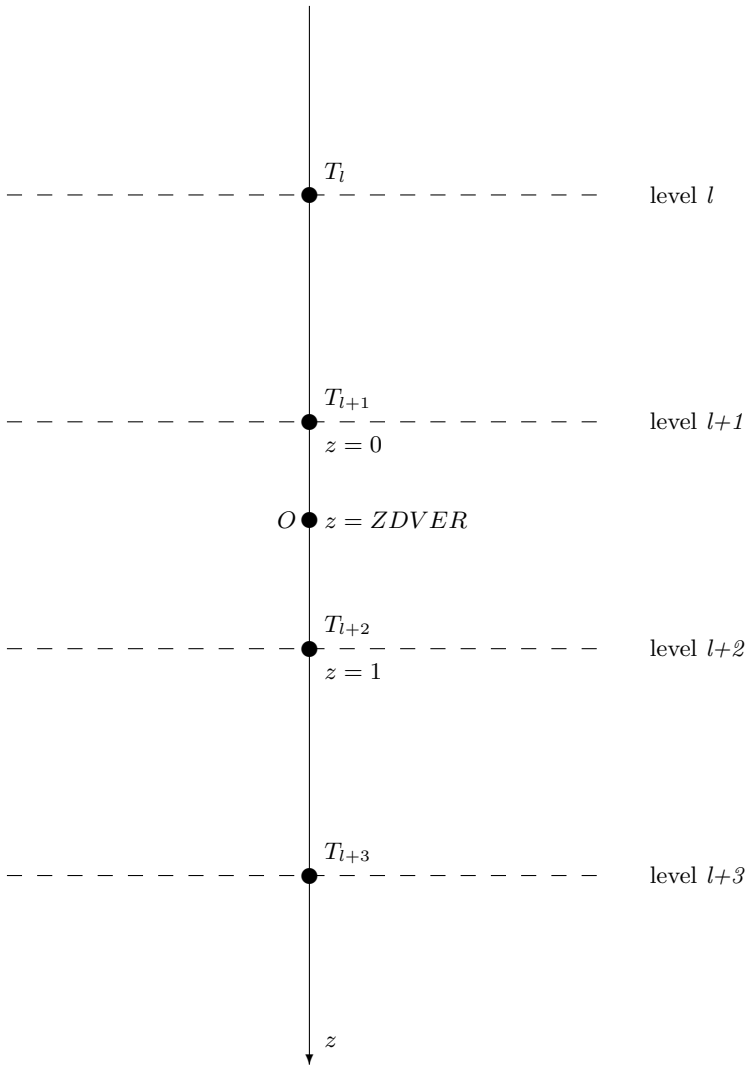
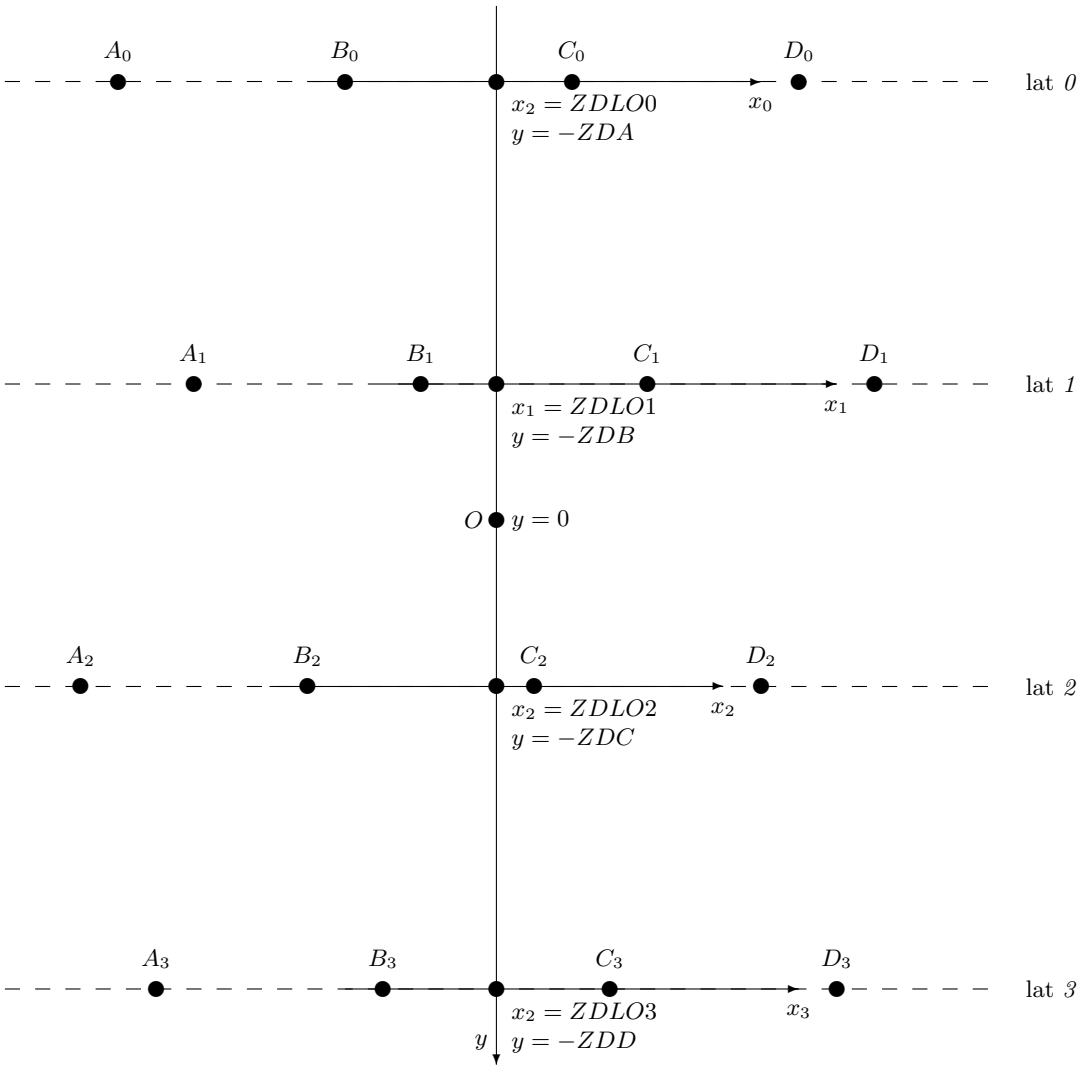


Figure 14.2: Interpolation vertical grid for linear and cubic vertical interpolations.



$x_0(B_0) = x_1(B_1) = x_2(B_2) = x_3(B_3) = 0$; $x_0(C_0) = x_1(C_1) = x_2(C_2) = x_3(C_3) = 1$.

Figure 14.3: Interpolation horizontal grid for 12 points interpolations.

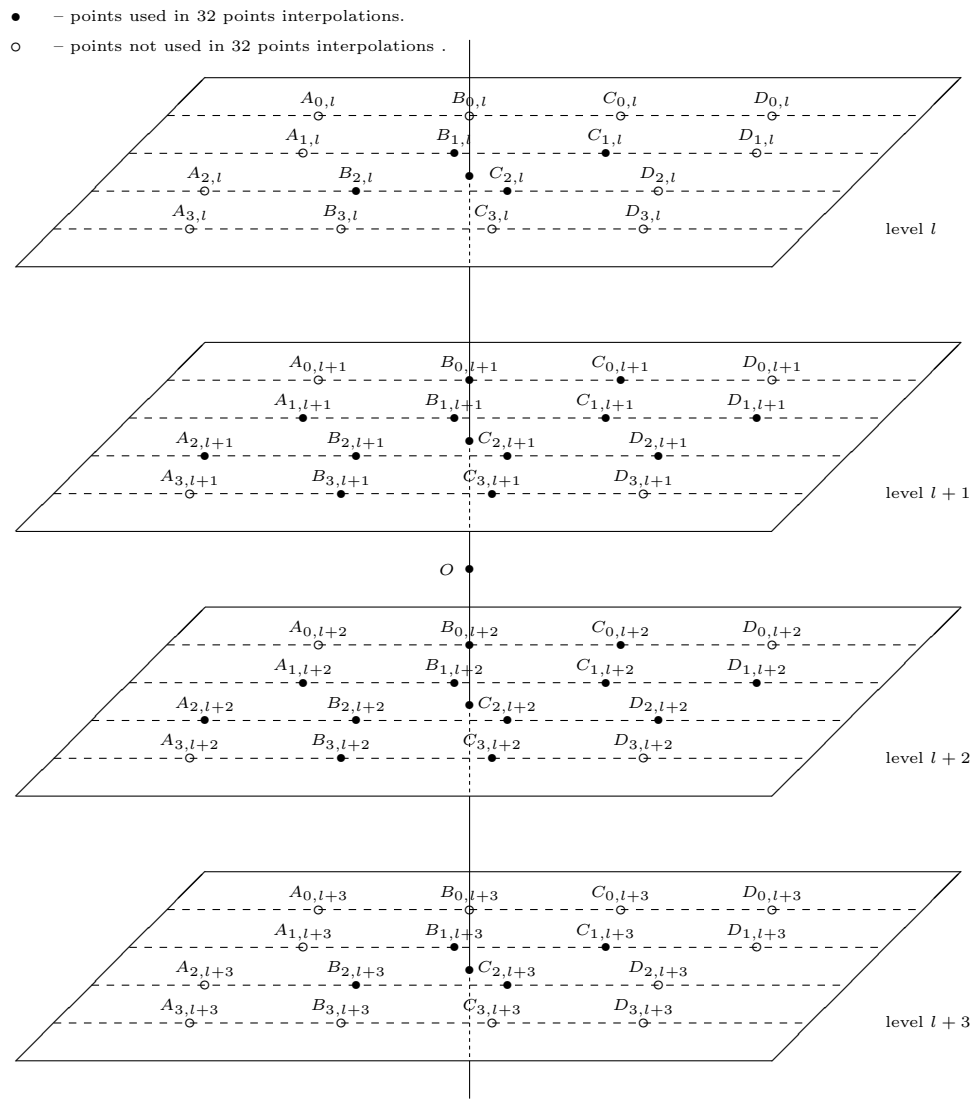


Figure 14.4: Interpolation grid for tri-linear and 32 points interpolations.

15 Lateral boundary conditions.

15.1 Extra longitudes and extra latitudes.

Let us denote by LX the number of longitudes (in the array **NLOENG** for each latitude in the code). For a quantity X, let us define:

X(longitude number 0)=X(longitude number LX).

X(longitude number LX+1)=X(longitude number 1).

X(longitude number LX+2)=X(longitude number 2).

These extra computations are necessary for all interpolated fields. For distributed memory computations are done when making the halo (routine **SLCOMM**+**SLCOMM2A** which exchange data with other processors).

Let us denote by lx the number of latitudes (**NDGLG** in the code): latitudes number -1,0,$lx+1$,$lx+2$ are respectively the symmetric of latitudes number 2,1,lx,$lx-1$. These extra computations are necessary for all interpolated fields. For distributed memory computations are done in **SLEXTPOL**.

15.2 Vertical boundary conditions in the 3D model.

∗ **Vertical linear interpolations for layer variables at the medium point:** The medium point has a vertical coordinate always included between $\eta_{\bar{l}=0}$ and $\eta_{\bar{l}=L}$ in case of vertical interpolating scheme. Therefore no extrapolated values are needed.

∗ **Vertical cubic 4 points interpolations for layer variables at the origin point:** When the origin point is above the layer number 2 (resp. below the layer number $L-1$), the vertical cubic 4 points interpolations using data of the layers number 1, 2, 3 (resp. $L-2$, $L-1$, L) and the extra-layer number 0 (resp. $L+1$) are degenerated into linear interpolations between the layers number 1 and 2 (resp. $L-1$ and L). The extrapolated values at the extra-layer number 0 (resp. $L+1$) are always multiplied by a weight equal to 0 and are set to 0 in subroutine **LAVABO**. This algorithm extends itself to the case where the origin point is between the top (resp. surface) and the layer number 1 (resp. L), but in this case the interpolation using data of the layers number 1 and 2 (resp. $L-1$ and L) becomes an extrapolation.

∗ **Vertical cubic 4 points interpolations for half level variables at the origin point:** When the origin point is above the half level number 1 (resp. $L-1$), the vertical cubic 4 points interpolations using data of the half levels number -1, 0, 1, 2 (resp. $L-2$, $L-1$, L and $L+1$) are degenerated into linear interpolations between the half levels numbers 0 and 1 (resp. $L-1$ and L).

∗ **Vertical cubic Hermite interpolations for layer variables at the origin point:** When the origin point is above the layer number 2 (resp. $L-1$), interpolation is still a vertical cubic Hermite one, computation of vertical derivatives is modified for layer number 1 (resp. L). This algorithm extends itself to the case where the origin point is between the top (resp. ground) and the layer number 1 (resp. L), but in this case the interpolation using data of the layers number 1 and 2 (resp. $L-1$ and L) becomes an extrapolation. For more details see subsubsection 14.1.5.

∗ **Vertical cubic spline interpolations for layer variables at the origin point:** Some top and bottom values are computed (the algorithm of computation will be provided in a later version of this documentation) and the vertical interpolation always uses 4 points.

16 2D shallow water and 3D models organigrammes.

16.1 Interpolation and weight calculation routines.

∗ **Weights:**

- **LASCAW**: computes weights and interpolation grid.
- **LASCAW_CLO**: computes zonal high-order weights (and also meridian high-order weights in LAM models).
- **LASCAW_CLA**: computes meridian high-order weights (spherical geometry).
- **LASCAW_VINTW**: computes vertical high-order interpolation weights.

∗ **Bilinear and trilinear interpolations:**

- **LAIDLI**: bi-linear 3D interpolations.
- **LAITLI**: tri-linear 3D interpolations.

∗ **High-order interpolations:**

- **LAIDDI**: 12 points horizontal interpolation.
- **LAITRI**: 32 points 3D interpolations, with vertical cubic 4 points interpolations. Is also used for SLHD diffusive interpolations (with modified pre-computed weights).
- **LAIHVT**: 32 points 3D interpolations, with vertical cubic Hermite interpolations.
- **LAITVSPCQM**: Shape-preserving routine for 48 points 3D interpolations, with 12 points horizontal interpolations and vertical cubic spline interpolations.
- **LAITRIQM3D**: 32 points 3D interpolations with 3D quasi-monotonic limiters.

∗ **Other routines:**

- **LAISMOO**: 32 points interpolation by linear least-square fit.
- **LAISMOA**: preliminary smoothing applied on quantities interpolated by the routine **LAISMOO**.
- **LAITRE_GFL**: interface routine for interpolations done on GFL variables.
- **LAITRE_GMV**: interface routine for interpolations done on GMV variables.
- **LAQMLIMITER**: a posteriori quasi-monotone limiter (called after **LAITRI**).

16.2 Routines specific to 2D organigramme.

- **LACDYNSHW** computation of the SL quantities, and semi-implicit scheme quantities.
- **CPG2LAG**: lagged grid-point computations; manages interpolations and computation of $t+\Delta t$ quantities.
- **LARMES2**: computation of medium point by an iterative algorithm for 2D model.
- **LAINOR2**: computation of origin point.
- **LARCIN2**: manages computation of medium/origin point coordinates (on the computational sphere), weights for interpolations, and interpolations.
- **LARCHE2**: computes latitude, longitude of the medium/origin point coordinates on the computational sphere, p and q of the $\mathcal{R}$ operator matrix.

16.3 Routines used in 3D organigramme.

- **GP_MODEL**: interface for grid-point calculations.
- **LACONE**: computes analytically $(2\mathbf{\Omega} \wedge a\mathbf{k})$.
- **CPG**: unlagged adiabatic calculations.
- **CPG_DRV**: driver for unlagged adiabatic calculations.
- **CPG_GP**, **CPG_DIA**, **CPG_DYN**, **CPG_END**: different parts of the grid-point calculations (resp. beginning including call to some GP.. routines, diagnostics, unlagged dynamics, end).
- **EC_PHYS**: interface for ECMWF physics.
- **EC_PHYS_LSLPHY**: interface for ECMWF split physics.
- **GPADDSLPHY**: adds the physical tendencies to the interpolation buffers for ECMWF physics if split physics.

- **GP_KAPPA**: computes the deformation and κ_* required in the SLHD interpolations. Called only if **LSLHD**=.T. .
- **GP_KAPPAT**: alternate way to compute κ_* for thermic variables.
- **GP_STDDIS**: computes the stretching/shrinking deformation $STDDIS$ required in the COMAD interpolations. Called only if **LCOMAD**=.T. .
- **LACDYN**: computation of the SL quantities, and semi-implicit scheme quantities.
- **LARCHE**: computes latitude, longitude of the medium/origin point coordinates on the computational sphere, p and q of the $\mathcal{R}$ operator matrix. **ELARCHE** is called instead of **LARCHE** for plane geometry.
- **LASURE**: some initialisations.
- **LASSIE**: computation of linear terms for hydrostatic equations.
- **LANHSI**: computation of linear terms for NHEE equations.
- **LANHQESI**: computation of linear terms for NHQE equations.
- **LANHSIB**: fills the **P[X]SI** arrays in the NH model when **LGWADV**=.T. .
- **LAVENT**: computation of quantities to be interpolated (wind) for trajectory research.
- **LATTE_NL**: for mixed extrapolation, computes coefficient C_γ.
- **LATTEX**: computation of quantities to be interpolated for 3D variables equations (GFL and GMV).
- **LATTEX_TNT**: called by **LATTEX** for a SL3TL scheme, part of the code which is the same for all variables.
- **LATTEX_DNT**: called by **LATTEX** for a SL2TL scheme, part of the code which is the same for all variables.
- **LATTE_KAPPA**: stores κ_* in the buffer "**SLBUF2**". Called only if **LSLHD**=.T. .
- **LATTE_STDDIS**: stores $STTDIS$ in the buffer "**SLBUF2**". Called only if **LCOMAD**=.T. .
- **LATTE_BBC**: computes the additional quantities to be stored in the buffers "**SLBUF1**" and "**SLBUF2**" for the option **LRDBBC**=.T. .
- **LATTES**: computation of quantities to be interpolated for 2D variables equations (currently continuity equation).
- **LAVABO**: vertical extrapolations (boundary conditions).
- **CPGLAG**: lagged adiabatic calculations other than semi-Lagrangian interpolations.
- **CALL_SL**: manages interpolations and computation of $t + \Delta t$ quantities.
- **LAPINEA**: manages the research of SL-trajectory.
- **LAPINEB**: manages the interpolated part of the equations RHS.
- **LARMES**: computation of medium point by an iterative algorithm for 3D model. Computes also the origin point. **ELARMES** is called instead of **LARMES** for plane geometry.
- **LARCINA**: manages computation of medium/origin point coordinates (on the computational sphere), weights for interpolations, and interpolations needed in the SL-trajectory research; apply to interpolations on layer data.
- **LARCINB**: manages interpolations needed at the origin point in the equations RHS; apply to interpolations on layer data.
- **LARCINHA**: the same as **LARCINA** but for half level data.
- **LARCINHB**: the same as **LARCINB** but for half level data.
- **GNHGW2SVD**: manages the conversions from gw towards d in the non-hydrostatic models when the key **LGWADV** is put to .T. .
- **MF_PHYS_PREP**: preparation of input data for **MF_PHYS**.
- **MF_PHYS**: interface for MF physics.
- **VSPLTRANS**: re-profiling of the quantity to be interpolated prior to vertical cubic spline interpolations (calls **TRIDIA** to invert tridiagonal linear systems).
- **VDIFLCZ**: interface for Buizza simplified physics.
- **VERDISINT**: interface routine for VFE integrals or derivatives.
- **VERINT**: does vertical integrals for vertical finite elements discretisation.

16.4 Set-up routines.

- **SUCT0**: 0-level control setup routine.
- **SUDYNA**: sets-up first part of dynamics.
- **SUDYN**: sets-up second part of dynamics.
- **SUINTDYN**: sets-up some structures used in the dynamics.
- **SUINTDYNSL**: sets-up some structures used in the dynamics under **CALL_SL**.
- **SUSLB**: initialise pointers of SL buffers and other quantities used in the SL scheme.
- **SUSLB2**: cf. **SUSLB** for 2D models.
- **SUSC2B**: computes different dimensions for SL buffers.
- **SLCSET+SLRSET**: computes some distributed memory environment for interpolations buffers.
- **SUSLAD1**: initialise data structures for SL adjoint.
- **SUSLAD2**: initialise constant arrays for SL adjoint interpolation.
- **SUSLAD3**: optimise halo for SL adjoint interpolation.
- **SUHSLMER**: compute intermediate quantities used in high order horizontal interpolations.
- **SUVERT**: computes some variables linked to vertical geometry.
- **SUVERTFE**: computes vertical operators used for VFE discretisation.
- **SUVSPLIP**: computes some variables linked to cubic spline vertical interpolations.
- **SUVSLETA**: fills YRVSLETA (intermediate quantities used in high order vertical interpolations).

16.5 Main features of organigramme for setup.

Only the most important features for SL scheme are described.

```
CNT0 ->
* IFS_INIT ->
  - SUCT0
  - SUDYNA
  - SUINTDYN
* SUOYOMA ->
  - SUGEOMETRY ->
    * geometry setup
    * SUINTERPOLATOR -> SUVSPLIP, SUVSLETA, SUHSLMER
* SUOYOMB ->
  - SUDYN ->
    * SUINTDYNSL
  - SUSC2B ->
    * SUSLB or SUSLB2
    * SLCSET -> SLRSET
    * SUSLAD1
    * SUSLAD2
```

16.6 Organigramme for 2D model.

```
STEP0 -> SCAN2M -> GP_MODEL_HEAP or GP_MODEL_STACK -> GP_MODEL ->
  * CPG2 ->
    - GPTF2
    - LACDYNSHW
    - WRSLTRAJ2
  * SLCOMM and SLEXTPOL
  * CPG2LAG ->
    - LADINE ->
      * LARMES2 -> LARCIN2 ->
        - LARCHE2 + LASCAW (ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO
        - LAIDDI
      * LAINOR2 -> LARCIN2 ->
        - LARCHE2 + LASCAW (ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO
        - LAIDDI
      * LACONE
      * LARCHE2
    - GPTF1
```

16.7 Organigramme for 3D model.

* Organigramme under STEPO.

```
STEPO -> SCAN2M -> GP_MODEL_HEAP or GP_MODEL_STACK -> GP_MODEL ->
  * CPG_DRV -> CPG ->
    - CPG_GP (see documentation (IDEUL) for details)
    - GPINIDDH
    - EC_PHYS_LSLPHY (split ECMWF physics) -> GPADDSLPHY
    - MF_PHYS_PREP
    - MF_PHYS (unlagged MF physics, organigramme not detailed)
    - CPG_DIA -> (routines for some diagnostics, organigramme not detailed)
    - CPDYSLDIA -> (semi-Lagrangian diagnostics)
    - CPG_DYN ->
      * LACDYN ->
        - LASURE
        - LASSIE, LANHSI or LANHQESI
        - LAVENT
        - LATTE_NL
        - LATTEX ->
          * LATTEX_TNT or LATTEX_DNT
          * VSPLTRANS -> TRIDIA
        - LATTE_KAPPA -> GP_KAPPA and GP_KAPPAT
        - LATTE_STDDIS -> GP_STDDIS
        - LATTE_BBC
        - LATTES -> VERDISINT -> VERINT
        - LAVABO
        - LANHSI or LANHQESI, and LANHSIB for NH and LGWADV=T
      * VDIFLCZ (see documentation (IDEUL) for details)
    - CPG_END -> (see documentation (IDEUL) for details)
  * CALL_SL_HEAP or CALL_SL_STACK -> CALL_SL ->
    - SLCOMM and SLEXTPOL (global model) or ESLEXTPOL (LAM model)
    - LAPINEA ->
      * LARMES (ELARMES in LAM models) ->
        - LARCINA (see below) + LAISMOA
        - LARCINA ->
          - LARCHE + LASCAW (ELARCHE + ELASCAW in LAM models)
            -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
          - LAITLI
          - LAISMOO
      * LARCINA ->
        - LARCHE + LASCAW (ELARCHE + ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
      * LARCINHA ->
        - LARCHE + LASCAW (ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
    - SLCOMM2A and SLEXTPOL (global model) or ESLEXTPOL (LAM model)
      if on-demand SL communications asked for
    - LAPINEB ->
      * LARCINB ->
        - LAITRE_GMV -> LAITRI
        - LAITRE_GFL -> LAITRI, LAITHVT, LAITVSPCQM, LAQMLIMITER, LAITRIQM3D
        - LAITLI
        - LAIDDI
      * LARCINHB ->
        - LAITRE_GMV -> LAITRI
        - LAITLI
      * some GP... and GNH... routines.
      * LARCHE
      * LACONE
      * VERDISINT -> VERINT
      * GNHGW2SVD for NH and LGWADV=T
  * CPGLAG (organigramme not detailed, see documentation (IDEUL))
  * EC_PHYS_DRV (organigramme not detailed)
```

17 Tangent linear and adjoint codes.

17.1 3D hydrostatic model.

17.1.1 Basics and remarks about TL code.

The coded linearised formulae will not be detailed in this documentation: the tangent linear code is taken line by line, thus it depend on the way the direct code is written. The main features of the tangent linear code are roughly the following ones:

- the tangent code looks like the direct code but with a differentiated shape. For example if the direct code contains an instruction like $Z = XY$ the tangent linear code will contain at the same place the instruction $Z = XY_5 + X_5Y$ using some trajectory variables subscripted by index "5".
- There is additionally a "trajectory" code which is a copy of the direct code, but applied to the trajectory variables subscripted by index "5".
- For the interpolation grid indices that appear in the weight computation routine (**LASCAWTL**) and the interpolation routines, there is only one version and the names of variables are not subscripted by index "5". The "trajectory"-coordinates (subscripted by index "5".) are used to compute the indices of the interpolation grid.
- Concerning the interpolation buffers "**PB1...**" there are several ways to deal with them, according to the value of variable **NTRSLTYPE**. The ADTL-trajectory is stored in the array **TRAJEC%SLAG**. For **NTRSLTYPE**=2 (the default value and the only value coded in the 3D model), some quantities (essentially some quantities of buffer **PB1...**) are stored in the direct model and read in the TL model, so their recomputation is not necessary in the TL model.

Additional remarks:

- Code is available for a subset of options; for example SL2TL.
- For subroutines, the name of which ends by "**5**", see their corresponding direct code routines, the explanation of which is given in the section describing the direct code. These routines compute the trajectory which is required in the TL calculations for some quantities, the trajectory of which is not computed in the direct code.
- For subroutines, the name of which ends by "**TL**", see their corresponding direct code routines, the explanation of which is given in the section describing the direct code.
- **LATTEXTL** and **LATTESTL**: quantities **P(X)T15** are not computed, excepted in the case **L2TLFF**=.T. where **PUT15** and **PVT15** are computed. **PB1...**-buffers are directly recovered by a memory transfer under **CPGTL** if **NTRSLTYPE**=2.
- **LARMESTL**: computation of the semi-Lagrangian displacement yields a duplication of the code into a "differentiated code" and a "ADTL-trajectory code" (for example an instruction like $Z = XY$ yields the "differentiated code" $Z = XY_5 + X_5Y$ and the "ADTL-trajectory code" $Z_5 = X_5Y_5$). This is the reason for which one can find in routine **LARCHETL** two sets of coordinates (**PLON**,**PLAT**) and (**PLON5**,**PLAT5**) and two sets of (p,q) matrices. Only the "5" quantities are used to compute the indices of the interpolation grid. On the contrary both sets of coordinates are used to compute the weights in **LASCAWTL** because both sets of weights are needed in the linear tangent versions of the interpolation routines.
- **LAPINEBTL**: computation of **P(X)T1** needs both the arrays **Z(X)95** and **Z(X)9**.
- **LARCHETL**: has optimisations which are not present in the direct code. First the code for not tilted geometry has been split into a code for stretched geometry (also valid for unstretched geometry) and a code for unstretched geometry. Additionally some functions such as **MAX**, **MIN**, **MOD**, have been replaced by tests with sometimes additional variables. The consequence is that the TL code looks not straightforward to compare with the direct code.

17.1.2 Organigramme of TL code under SCAN2MTL.

```
SCAN2MTL -> GP_MODEL_TL ->
  * CPG_DRV_TL -> CPGTL ->
    - Allocations
    - CPG5_GP (for trajectory) ->
      * some routines reading trajectory (RDPHTRAJM)
      * GPMPFC5, GP_SPV (for trajectory)
      * some GP.. routines computing intermediate grid-point quantities for trajectories.
    - CPG_GP_TL ->
      * GPTF2
      * GPMPFC (for model variables)
      * some GP..TL routines computing intermediate grid-point quantities for model variables.
      * GPINISLB
    - MF_PHYSTL (TL of simplified non lagged physics)
    - CPG_DYN_TL ->
      * LACDYNTL ->
        - LASURE
        - LASSIETL
        - LAVENTTL
        - LATTEXTL -> LATTEX_DNT
        - LATTE_KAPPATL -> GP_KAPPATL and GP_KAPPATTL
        - LATTESTL -> VERINT
        - LAVABOTL
      * VDIFLCZTL (organigramme not detailed)
    - CPG_END_TL ->
      * GPMPFC (for model variables)
    - GPMPFC5 (for trajectory)
    - Deallocations
  * CALL_SL_TL ->
    - SLCOMM and SLEXTPOL (DM code only)
    - LAPINEA5 ->
      * LARMES5 (ELARMES5 in LAM models) -> LARCINA ->
          - LARCHE + LASCAW (ELARCHE + ELASCAW in LAM models)
            -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
          - LAITLI
      * LARCINA ->
        - LARCHE + LASCAW (ELARCHE + ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
    - LAPINEATL ->
      * LARMESTL (ELARMESTL in LAM models) -> LARCINATL ->
        - LARCHETL (ELARCHETL in LAM models)
        - LASCAWTL (ELASCAWTL in LAM models)
          -> LASCAW_CLA_TL, LASCAW_CLO_TL, LASCAW_VINTW_TL
        - LAITLITL
      * LARCINATL ->
        - LARCHETL (ELARCHETL in LAM models)
        - LASCAWTL (ELASCAWTL in LAM models)
          -> LASCAW_CLA_TL, LASCAW_CLO_TL, LASCAW_VINTW_TL
    - SLCOMM2A and SLEXTPOL (DM code only and on-demand SL communications asked for)
    - LAPINEBTL ->
      * LARCINBTL ->
        - LAITRE_GMV_TL -> LAITRITL
        - LAITRE_GFL_TL -> LAITRITL, LAITRIQM3DTL
        - LAITLITL
        - LAIDDITL
      * VERINT
      * LACONETL
      * LARCHETL (ELARCHETL in LAM models)
      * VERINT
  * EC_PHYS_TL (organigramme not detailed)
  * CPGLAGTL -> (organigramme not detailed, see documentation (IDEUL))
```

17.1.3 Basics and remarks about AD code.

One starts from the linear tangent code and "adjoints" the code: for example a matricial expression of the following type $Y = MX$ leads to the expression $X = X + {}^tMY$ in the adjoint code. Some particular features appear in the code of the semi-Lagrangian scheme, and the adjoint code is not completely the mirror of the linear tangent code.

Additional remarks:

- For subroutines, the name of which ends by "**5**", see their corresponding direct code routines, the explanation of which is given in the section describing the direct code. These routines compute the trajectory which is required in the AD calculations for some quantities, the trajectory of which is not computed in the direct code.
- For subroutines, the name of which ends by "**AD**", see their corresponding direct code routines, the explanation of which is given in the section describing the direct code.
- The storage of the ADTL-trajectory variables (indexed by "5") is not fully done in the direct code, because of the huge amount of data to store. So some quantities which have normally to be stored in the direct code, are in this case recomputed in the adjoint code and stored in arrays, the name of which is ending by "5" (sometimes "6" or "7"). This recomputation is done on all data concerning the iterative calculation of the semi-Lagrangian displacement (medium and origin points) and the weights and grid for interpolations. One can notice that, when computing the position of the mid-point, all intermediate positions are stored at each iteration (contrary to the direct code which only retains the last iteration). The consequence is that the adjoint code calls the routines **LARMES5** (slightly modified version of **LARMES**), **LARCINB5**, **LARCHE** and **LACONE**.
- In **LASCAW**, only the computation of interpolation weights has an adjoint code, but not the index-array for the interpolation grid (arrays **KL0** and **KLH0**).
- In interpolation routines (**LAIDDIAD**, **LAIDLIAD**, **LAITRIAD**, **LAITLIAD**), adjoint codes have some optimisations not present in the direct code and the TL code, in order to avoid memory conflicts. The main feature of these optimisation is the use of an intermediate array **PZINC** in the adjoint code of the interpolation routines. Memory transfer in **PSLBUF1** is done only in **LARCINAAD** and **LARCINBAD** once all adjoints of interpolation routines have been called.
- **CPG5** is a very simplified version of **CPG** which currently only reads trajectory.
- **LARCHEAD**: has optimisations which are not present in the direct code and does not follow completely **LARCHETL** (no splitting between stretched and unstretched geometry). Some functions such as **MAX**, **MIN**, **MOD**, have been replaced by tests with sometimes additional real or logical variables. The consequence is that the AD code looks not straightforward to compare with the direct and TL codes.
- The adjoint interpolation routines have options of reproducibility which are not present in the direct and TL codes, and which need to call intermediate **LAI..._INIT** routines.

17.1.4 Organigramme of AD code under SCAN2MAD.

```
SCAN2MAD -> GP_MODEL_AD ->
  * CPG5
  * CPGLAGAD (call tree not detailed, see documentation (IDEUL))
  * EC_PHYS_AD (call tree not detailed)
  * PRE_SLADREP (case LSLADREP=T) ->
    - SLCOMM2A (case LSLONDEM=T)
    - SLCOMM (case LSLONDEM=F)
  * CALL_SL_AD ->
    - SLCOMM and SLEXTPOL (DM code only)
    - LAPINEA5 ->
      * LARMES5 -> LARCINA ->
        - LARCHE (ELARCHE in LAM models)
        - LASCAW (ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
        - LAITLI
      * LARCINA ->
        - LARCHE + LASCAW (ELASCAW in LAM models)
          -> LASCAW_CLA, LASCAW_CLO, LASCAW_VINTW
    - SLCOMM2A and SLEXTPOL (DM code only and on-demand SL communications asked for)
    - LAPINEBAD ->
      * LARCINB5 ->
        - LAITLI
        - LAITRI
      * LACONE
      * LARCHE
      * VERINTAD
      * LACONEAD
      * LARCHEAD
      * LARCINBAD ->
        - some LAI..._INIT routines (LSLADREP=T)
        - LAITLIAD
        - LAITRE_GMV_AD -> LAITRIAD
        - LAITRE_GFL_AD -> LAITRIAD, LAITRIQM3DAD
        - LAIDDIAD
    - LAPINEAD ->
      * LARCINAAD ->
        - LARCHEAD + LASCAWAD
          -> LASCAW_CLA_AD, LASCAW_CLO_AD, LASCAW_VINTW_AD
      * LARMESAD -> LARCINAAD ->
        - some LAI..._INIT routines (LSLADREP=T)
        - LAITLIAD
        - LARCHEAD + LASCAWAD
          -> LASCAW_CLA_AD, LASCAW_CLO_AD, LASCAW_VINTW_AD
    - SLEXTPOLAD (case LSLADREP=F)
    - SLCOMM2 (case LSLADREP=F + LSLONDEM=T)
    - SLCOMM (case LSLADREP=F + LSLONDEM=F)
  * CPG_DRV_AD -> CPGAD ->
    - Allocations
    - CPG5_GP (for trajectory) ->
      * some routines reading trajectory (RDPHTRAJM)
      * GPMPFC5, GP_SPV (for trajectory)
      * some GP.. routines computing intermediate grid-point quantities for trajectories.
    - CPG_END_AD ->
      * GPMPFC
    - CPG_ZERO_AD
    - CPG_DYN_AD ->
      * VDIFLCZAD (organigramme not detailed)
      * LACDYNAD ->
        - LASURE
        - LATTESAD -> VERINTAD
        - LATTE_KAPPAAD -> GP_KAPPAAD and GP_KAPPATAD
        - LATTEXAD -> LATTEX_DNT_AD
        - LAVENTAD
        - LASSIEAD
    - MF_PHYSAD (organigramme not detailed)
    - CPG_GP_AD ->
      * some GP..AD routines computing intermediate grid-point quantities for model variables.
      * GPMPFC (for model variables)
      * GPTF2AD
    - GPMPFC5 (for trajectory)
```

17.2 2D shallow-water model.

17.2.1 Basics and remarks about TL code.

For basics, see part (17.1.1).

Additional remarks:

- Code is available for a subset of options; for example SL2TL, unstretched untilted geometry.
- **RDSLTRAJ2** reads the TL trajectory, which is stored in arrays underscripted by index "5".
- For subroutines, the name of which ends by "**TL**", see their corresponding direct code routines, the explanation of which is given in the section describing the direct code.
- **LACDYNSHWTL**: quantities **P(X)T15** are not computed. One only computes **PB1...**-buffers **P(X)L95** and **P(X)L05** if **NTRSLTYPE** is zero or 1; otherwise the **PB1...**-buffers are directly recovered by a call to **RDSLTRAJ2** under **CPG2TL** if **NTRSLTYPE**=2.
- **LARMES2TL** and **LAINOR2TL**: computation of the semi-Lagrangian displacement yields a duplication of the code into a "differentiated code" and a "ADTL-trajectory code" (for example an instruction like $Z = XY$ yields the "differentiated code" $Z = XY_5 + X_5Y$ and the "ADTL-trajectory code" $Z_5 = X_5Y_5$). This is the reason for which one can find in routine **LARCHE2TL** two sets of coordinates (**PLON**,**PLAT**) and (**PLON5**,**PLAT5**) and two sets of (p,q) matrices. Only the "5" quantities are used to compute the indices of the interpolation grid. On the contrary both sets of coordinates are used to compute the weights in **LASCAWTL** because both sets of weights are needed in the linear tangent versions of the interpolation routines.
- **LADINETL**: computation of **P(X)T1** needs both the arrays **Z(X)95** and **Z(X)9**.
- **LARCHE2TL**: cf. **LARCHETL**.

17.2.2 Organigramme of TL code.

∗ **Direct code modification:** **CPG2** calls **WRSLTRAJ2** to store the trajectory in configurations where the TL code is called (configurations 421 and 521). **WRSLTRAJ2** is a subroutine which writes data on a buffer.

∗ TL code organigramme under SCAN2MTL:

```
SCAN2MTL -> GP_MODEL_TL ->
  * CPG2TL ->
    - RDSLTRAJ2 (if NTRSLTYPE = 1)
    - GPTF2
    - LACDYNSHWTL
    - RDSLTRAJ2 (if NTRSLTYPE = 2)
  * CPG2LAGTL ->
    - LADINETL ->
      * LARMES2TL -> LARCIN2TL -> (see below)
      * LAINOR2TL -> LARCIN2TL -> (see below)
      * LARCHE2TL
      * LACONETL
    - GPTF1
```

Organigramme below **LARCIN2TL**:

```
LARCIN2TL ->
  * LARCHE2TL + LASCAWTL -> LASCAW_CLA_TL, LASCAW_CLO_TL
  * LAIDDITL or LAIDLITL
```

17.2.3 Basics and remarks about AD code.

One starts from the linear tangent code and "adjoints" the code: for example a matricial expression of the following type $Y = MX$ leads to the expression $X = X + {}^tMY$ in the adjoint code.

Additional remarks:

- Code is available for a subset of options; for example SL2TL, unstretched untilted geometry.

- The storage of the ADTL-trajectory variables (indexed by "5") is not fully done in the direct code, because of the huge amount of data to store. So some quantities which have normally to be stored in the direct code, are in this case recomputed in the adjoint code and stored in arrays, the name of which is ending by "5" (sometimes "6" or "7"). This recomputation is done on all data concerning the iterative calculation of the semi-Lagrangian displacement (medium and origin points) and the weights and grid for interpolations. One can notice that, when computing the position of the mid-point, all intermediate positions are stored at each iteration (contrary to the direct code which only retains the last iteration). The consequence is that the adjoint code will call the routines **LARMES25**, **LAINOR2**, **LARCIN2**, **LARCHE2**, **LACONE** and **LASCAW**. Concerning the not lagged dynamics one finds a sequence **CPG25** $->$ **LACDYNSHW** before the call to **CPG2LAGAD** under **SCAN2MAD**. **CPG25** is a simplified version of **CPG2** which calls **LACDYNSHW** and gets some $t - \Delta t$ variables by calling **RDSLTRAJ2**.
- In **LASCAW**, only the computation of interpolation weights has an adjoint code, but not the index-array for the interpolation grid (arrays **KL0** and **KLH0**).
- Some direct code is called again under **SCAN2MAD**, **LADINEAD** and **LARCIN2AD**.
- **LARMES25** is a version of **LARMES2**, the only differences are actually:
  - only the semi-Lagrangian displacement is computed; no call of interpolations at the mid-point for the RHS of the equations.
  - all the intermediate positions of the mid-point are stored in the iterative algorithm.
- **CPG25** is a very simplified version of **CPG2**.
- **LADAD**: interface for interpolations (equivalent of **CALL_SL_AD** in the 3D model).
- **LARCHE2AD**: same remarks as for the hydrostatic 3D model.

17.2.4 Organigramme of AD code.

$*$ AD code organigramme under SCAN2MAD:

```
SCAN2MAD -> GP_MODEL_AD ->
  * CPG25 ->
    - RDSLTRAJ2
    - LACDYNSHW
  * CPG2LAGAD ->
    - GPTF1AD
  * LADAD ->
    - LADINEAD ->
      * LARMES25 -> LARCIN2 -> (see below)
      * LAINOR2 -> LARCIN2 -> (see below)
      * LACONE
      * LARCHE2
      * LACONEAD
      * LARCHE2AD
      * LAINOR2AD -> LARCIN2AD -> (see below)
      * LARMES2AD -> LARCIN2AD -> (see below)
  * CPG2AD ->
    - LACDYNSHWAD
    - GPTF2AD
```

Organigramme below **LARCIN2**:

```
LARCIN2 ->
  * LARCHE2 + LASCAW -> LASCAW_CLA, LASCAW_CLO
  * LAIDDI or LAIDLI
```

Organigramme below **LARCIN2AD**:

```
LARCIN2AD ->
  * LARCHE2 + LASCAW -> LASCAW_CLA, LASCAW_CLO
  * LAIDDIAD or LAIDLIAD
  * LASCAWAD + LARCHE2AD -> LASCAW_CLA_AD, LASCAW_CLO_AD
```

18 Some distributed memory features.

For features common to Eulerian and semi-Lagrangian advections, see documentation (IDEUL). We now focus on specific semi-Lagrangian interpolator features. Description is done for the 3D model. Note that some variables referenced below may be attributes of structures.

∗ **Transmission of data for horizontal interpolations:** First we define, for each interpolation point, what is the processor which treats it. The rule is that the processor which treats it is the processor which treats the corresponding arrival point of the SL trajectory. This associated model grid-point is always between the latitudes 1 and **NDGLG** (it is never a pole or an extra-polar latitude point).

Interpolations use data of points which are not necessary on the same latitude and longitude as the interpolation point and the arrival grid-point of the semi-Lagrangian trajectory. Thus interpolation routines need to have access to a limited number of surrounding latitudes and longitudes which are not necessary treated by the current processor.

The number of surrounding latitudes and longitudes rows necessary for interpolations (but which do not always belong to the current processor) is precomputed in the subroutine **SUSC2B** (variable **NSLWIDE**). This is a sort of "halo" belonging to some other processors. Due to the "halo" this is necessary to split calculations into not lagged ones and lagged ones.

First the not lagged computations are done for the **NGPBLKS** packets of length **NPROMA**.

Quantities to be interpolated are computed in the non-lagged part (but interpolations are performed in the lagged part). In the non-lagged part, only data of the current processor (without any extra-longitudinal data nor polar and extra-polar data) is computed. For all the **NPROMA**-packages treated by the current processor data is stored in the arrays **PB1** (under routine **CPG**). The first dimension of **PB1** is **NASLB1**, which is the total number of points one needs for the interpolations (**NASLB1** is greater than **NGPTOT**). The second dimension of **PB1** is **NFLDSLB1**: this is the number of 2D quantities to be interpolated. Inside **CPG**, an intermediate array **ZSLBUF1AU** is used and data is transferred in **PB1**: this memory transfer uses a precomputed intermediate array **NSLCORE**. Then some communication routines are called in **CALL_SL** to constitute the halo. Routine **SLCOMM** does processor communication to fill the halo (receives and sends data from some other processors).

When all the **NASLB1** dataset is constituted, the lagged part **CALL_SL** is called which does horizontal interpolations for all the data to be interpolated for the current processor. The lagged computations are still divided into **NGPBLKS** packets of length **NPROMA**.

Terms which must be evaluated at F are stored in **GFLT1**, **GMVT1**, **GMVT1S** (for the RHS of equations) and **PB2** for additional intermediate terms.

∗ **Particular case of the "on demand" processor communication:** **SLCOMM** and **SLEXTPOL** are still called with specific options, and do the communications only on the part of the buffer which contains the information to find the semi-Lagrangian displacement. For the RHS of equations, the interpolations need to communicate a subset of points, this subset can be known only when all the interpolations grids have been computed (in **LASCAW**): communications are currently done by **SLCOMM2A** and **SLEXTPOL** called by **CALL_SL** just before the interpolations. **SLCOMM2A** and **SLEXTPOL** have less points to exchange so the model integration is slightly less expensive.

19 Specific SL variables in pointer modules, modules and namelists.

These modules are auto-documented so description of each variable is provided in the code source. We can recall here the most important variables to know for each module:

- **PARDIM** (parameter dimensions).
- **PTRSLB1** (pointers for quantities to be interpolated).
- **PTRSLB15** (pointers for quantities to be interpolated: trajectory for TL and AD models).
- **PTRSLB2** (pointers for arrays to communicate information between the different blocks in the grid-point calculations).
- **YOM_YGFL** and namelist **NAMGFL**: some GFL attributes like LADV, LINTLIN, LTDIABLIN, LHORTURB, LQM, LQMH, LQM3D, LQML3D, LSLHD, LCOMAD, LHV, LVSPLIP.
- **YOMARG** (0-level control, former command line) and **YOMCT0** (0-level control):
  - All NAMARG variables: for ex. NCONF, LELAM, LECMWF, LSLAG, NSUPERSEDE.
  - LR3D, LR2D, LRSHW, LRVEQ.
  - LSPRT.
  - LREGETA, LVFE_REGETA.
  - LNHEE, LNHQE, LNHDYN.
  - LRPLANE.

  Some of these variables are in namelist **NAMCT0**.
- **YOMCVER** (vertical finite element discretisation keys). Some of these variables are in namelist **NAMCVER**.
- **YOMDIM**, **YOMDIMV** and **YOMDIMF** (dimensioning): most of variables. Some of these variables are in namelist **NAMDIM**.
- **YOMDYNA** (adiabatic dynamics: first part):
  - LAPRXPK, NDLNPR, RHYDR0 (vertical discretisation).
  - L3DTURB (3D turbulence).
  - All variables, the name starts by LSLHD or SLHD (SLHD = semi-lagrangian horizontal diffusion).
  - All variables, the name starts by LCOMAD or COMAD (for COMAD interpolations).
  - LSLDIA, LRPRSLTRJ (diagnostics).
  - LRALTVDISP.
  - LVSPLIP (spline cubic vertical interpolations).
  - LPC_FULL, LPC_CHEAP, LPC_CHEAP2 (predictor-corrector scheme).
  - LNESC, LNESCT, LNESCV, LSETTLS, LSETTLST, LSETTLSV, LMIXETTLS, LELTRA (extrapolations).
  - NPDVAR, NVDVAR, ND4SYS, LNHX, LNHXDER, LSI_NHEE, LNHEE_SVDLAPL_FIRST (NH models).
  - LGWADV, NGWADVSI, LRDBBC (treatment of vertical divergence equation in NH model).

  Some of these variables are in namelist **NAMDYNA**.
- **YOMDYN** (adiabatic dynamics: second part):
  - REPS1, REPS2, REPSM1, REPSM2, REPSP1 (Asselin filter).
  - LSIDG, BETADT, RBT, RBTS2, NITERHELM (semi-implicit scheme).
  - REFGEO, SIPR, SITR, SITRA, SITRUB, SIPRUB, SITIME, SIRPRG, SIRPRN (reference values used in the semi-implicit scheme).
  - NSITER, NCURRENT_ITER, LRHDI_LASTITERPC (predictor-corrector scheme).
  - LSVTSM, RPRES_SVTSM (stratospheric vertical trajectory smoothed in SL scheme).
  - NVLAG, NWLAG, NTLAG, NSPDLAG, NSVDLAG (controls interpolations done in equations).
  - NSPLTHOI, LSPLTHOIGFL (controls splitting high order interpolations).
  - NSLDIMK, JPSLDIMK.
  - NITMP, VETAON, VETAOX, VMAX1, VMAX2 (SL trajectory).
  - VESL, XIDT (uncentering factors).
  - LQMHW, LQMHT, LQMHP, LQMHSPD, LQMHSVD (horizontally quasi-monotonic interpolations).
  - LQMW, LQMT, LQMP, LQMSPD, LQMSVD (quasi-monotonic interpolations).
  - LADVF, LIMPF, L2TLFF, RW2TLFF (treatment of Coriolis term).
  - RCMSLP0 (Eulerian treatment of orography).

  - SLHDA ($\kappa_{\rm a}$), SLHDA0 or SLHDA0T ($\kappa_{\rm a0}$), SLHDB or SLHDBT ($\kappa_{\rm b}$), SLHDD0 (DEF_0), SLHDD00 or SLHDD00T, SLHDDIV ($C_{\rm slhddiv}$), SLHDRATDDIV ($C_{\rm slhdratddiv}$), SLHDHOR, ZSLHDP1 ($\kappa_{\rm a}$), ZSLHDP3 (DEF_{00}), LSLHDHEAT for SLHD scheme.
  - LFINDVSEP, NVSEPL, NVSEPC (optimisations for semi-Lagrangian adjoint code).
  - LSETTLSVF, RSETTLF, LSETFSTAT, RPRES_SETTLSVF, REPSDYN, NFLEVSF.

  Some of these variables are in namelist **NAMDYN**.
- **YOMMP0** and **YOMMP** (distributed memory environment, see documentation (IDDM) for more details).
- **YOMSLPHY** (interactions between SL scheme and split ECMWF physics).
- **YOMSLREP** (variables used for bit reproducibility in the AD code of SL). A subset of them must go later in **EINT_MOD**.
- **EINT_MOD** (quantities used in externalisable interpolator and halo management). In particular attributes of variable YRSL.
- **YOMHSLMER**: for high-order horizontal meridian interpolations.
- **YOMVSPLIP**: for spline vertical interpolations.
- **YOMVSLETA**: for high-order vertical interpolations.
- **YOMMASK**.
- **YOMVAR**: LSLADREP, LVECADIN.
- Modules for trajectory stored for TL and AD models, in particuliar **TRAJ_SEMILAG_MOD**:
- Modules for geometry:
  - **YOMGSGEOM** (geographic space grid-point geometry)
  - **YOMCSGEOM** (computational space grid-point geometry)
  - **YOMOROG** (orography)
  - **YOMGEM** (horizontal geometry): some variables are in namelist **NAMGEM**.
  - **SPGEOM_MOD** (spectral geometry).
  - **YOMVERT** (vertical geometry and VFE operators).
  - **YOMVV1** (namelist variables for vertical geometry): variables are in namelist **NAMVV1**.
  - **YEMGEO** (LAM model geometry): some variables are in namelist **NEMGEO**.
  - **YEMLBC_GEO** (LAM model geometry for LBC).
  - **TYPE_GEOMETRY**, **TYPE_SPGEOM**.
- Modules for GMV dataflow:
  - **YOMGMV** (grid-point arrays for GMV fields).
  - **TYPE_GMVS** (derived types for GMV).
  - **GMV_SUBS_MOD** (contains encapsulated routines for GMV management).
- Modules for GFL dataflow:
  - **YOMGFL** (grid-point arrays for GFL fields).
  - **YOM_YGFL** (descriptors of GFL arrays). Some of these variables are in namelist **NAMGFL**.
  - **GFL_SUBS_MOD** (contains encapsulated routines for GFL management).
- Modules for surface dataflow:
  - **SURFACE_FIELDS_MIX** (surface dataflow).

20 Bibliography.

The following list is far to be complete. More detailed references are available in some of the following papers (in particular Staniforth and Côté, 1991), especially for papers published before 1991. Most of the recent papers have been published in *Mon. Wea. Rev.*.

20.1 Publications.

- Bartello, P., and S. J. Thomas, 1996: The cost-effectiveness of semi-Lagrangian advection. *Mon. Wea. Rev.*, **124**, 2883-2897.
- Bates, J. R., S. Moorthi and R. W. Higgins, 1993: A global multilevel atmospheric model using a vector semi-Lagrangian finite-difference scheme: Part I: adiabatic formulation. *Mon. Wea. Rev.*, **121**, 244-263.
- Bates, J. R., Y. Li, A. Brandt, S. F. Mc Cormick, and J. Ruge, 1995: A global shallow-water numerical model based on the semi-Lagrangian advection of potential vorticity. *Q. J. R. Meteorol. Soc.*, **121**, 1981-2005.
- Bates, J. R., 1996: Developments in semi-Lagrangian global atmospheric modelling at NASA/GLA. *Modern Dynamical Meteorology. Proceedings from a symposium in honour of Prof. Aksel Wiin Nielsen*, 3-8.
- Behrens, J., 1996: An adaptative semi-Lagrangian advection scheme and its parallelization. *Mon. Wea. Rev.*, **124**, 2386-2395.
- Bénard, P., and K. Yessad, 2001: On the discretization of the continuity equation in semi-Lagrangian models with a pressure-type vertical coordinate. *Mon. Wea. Rev.*, **129**, 1746-1749.
- Benoît, R., M. Desgagné, P. Pellerin, S. Pellerin, Y. Chartier and S. Desjardins, 1997: The Canadian MC2: a semi-Lagrangian, semi-implicit wideband atmospheric model suited for finescale process studies and simulation. *Mon. Wea. Rev.*, **125**, 2382-2415.
- Bermejo, R., and A. Staniforth, 1992: The conversion of semi-Lagrangian advection schemes to quasi-monotone schemes. *Mon. Wea. Rev.*, **120**, 2622-2632.
- Bermejo, R., and J. Conde, 2002: A conservative quasi-monotone semi-Lagrangian scheme. *Mon. Wea. Rev.*, **130**, 423-430.
- Böttcher, M., 1996: A semi-Lagrangian advection scheme with modified exponential splines. *Mon. Wea. Rev.*, **124**, 716-729.
- Bubnová, R., G. Hello, P. Bénard, and J.F. Geleyn, 1995: Integration of the fully elastic equations cast in the hydrostatic pressure terrain-following coordinate in the framework of the ARPEGE/Aladin NWP system. *Mon. Wea. Rev.*, **123**, 515-535.
- Caya, D., and R. Laprise, 1999: A semi-implicit semi-Lagrangian regional climate model: the Canadian RCM. *Mon. Wea. Rev.*, **127**, 341-362.
- Chen, Minghang, and J. R. Bates, 1996: A comparison of climate simulations from a semi-Lagrangian and an Eulerian GCM. **Journal of Climate**, **9**, 1126-1149.
- Chen, Minghang, and J. R. Bates, 1996: Forecast experiments with a global finite-difference semi-Lagrangian model. *Mon. Wea. Rev.*, **124**, 1992-2007.
- Cordero, E., N. Wood and A. Staniforth, 2005: Impact of semi-Lagrangian trajectories on the discrete normal modes of a non-hydrostatic vertical-column model. *Q. J. R. Meteorol. Soc.*, **131**, 93-108.
- Côté, J., 1988: A Lagrange multiplier approach for the metric terms of semi-Lagrangian models on the sphere. *Q. J. R. Meteorol. Soc.*, **114**, 1347-1352.
- Côté, J., M. Roch, A. Staniforth and L. Fillion, 1993: A variable-resolution semi-Lagrangian finite-element global model of the shallow-water equations. *Mon. Wea. Rev.*, **121**, 231-243.
- Côté, J., S. Gravel, and A. Staniforth, 1995: A generalised family of schemes that eliminate the spurious resonant response of semi-Lagrangian schemes to orographic forcing. *Mon. Wea. Rev.*, **123**, 3605-3613.
- Cotter, C. J., J. Frank and S. Reich, 2007: The remapped particle-mesh semi-lagrangian advection scheme. *Q. J. R. Meteorol. Soc.*, **133**, 251-260.
- Courtier, Ph., and J. F. Geleyn, 1988: A global model with variable resolution. Application to the shallow-water equations. *Q. J. R. Meteorol. Soc.*, **114**, 1321-1346.
- Courtier, Ph., C. Freydier, J. F. Geleyn, F. Rabier and M. Rochas, 1991: The ARPEGE project at METEO-FRANCE. ECMWF Seminar Proceedings 9-13 September 1991, Volume II, 193-231.
- Durran, Dale R., and P. A. Reinecke, 2004: Instability in a class of explicit two-time-level semi-Lagrangian schemes. *Q. J. R. Meteorol. Soc.*, **130**, 365-369.
- Desharnais, F., and A. Robert, 1990: Errors near the poles generated by a semi-Lagrangian integration scheme in a global spectral model. *Atm. Ocean*, Vol **28** No 1, 163-176.
- Diamantakis, M., 2013: The semi-Lagrangian technique in atmospheric modelling: current status and future challenges. *ECMWF Seminar in numerical methods for atmosphere and ocean modelling, 2-5 September 2013*, 183-200.

- Diamantakis, M., 2014: Global mass fixer algorithms for conservative tracer transport in the ECMWF model. *Geosci. Model. Dev.*, **7**, 965-979.
- Diamantakis, M., and L. Magnusson, 2016: Sensitivity of the ECMWF Model to Semi-Lagrangian Departure Point Iterations. *Mon. Wea. Rev.*, **144**, 3233-3250.
- Ekman, A., and E. Källen, 1998: Mass conservation tests with the HIRLAM semi-Lagrangian time integration scheme. *HIRLAM Technical Report*, **39**, 23pp.
- Geleyn, J.F., and R. Bubnová, 1995: The fully elastic equations cast in hydrostatic pressure coordinate: accuracy and stability aspects of the scheme as implemented in ARPEGE/Aladin. *Atm. Ocean, special issue memorial André Robert.*
- Girard, C., and Y. Delage, 1990: Stable schemes for nonlinear vertical diffusion in atmospheric circulation models. *Mon. Wea. Rev.*, **118**, 137-146.
- Giraldo, F. X., 1999: Trajectory calculations for spherical geodesic grids in cartesian space. *Mon. Wea. Rev.*, **127**, 1651-1662.
- Giraldo, F. X., 2006: Hybrid Eulerian-Lagrangian semi-implicit time-integrators. *Computers and Mathematics with Applications*, **52**, 1325-1342.
- Gospodinov, I., V. Spiridonov, and J.-F. Geleyn, 2001: Second order accuracy of two-time-level semi-Lagrangian schemes. *Q. J. R. Meteorol. Soc.*, **127**, 1017-1033.
- Gospodinov, I., V. Spiridonov, P. Bénard, and J.-F. Geleyn, 2002: A refined semi-Lagrangian vertical trajectory scheme applied to a hydrostatic atmospheric model. *Q. J. R. Meteorol. Soc.*, **128**, 323-336.
- Grabowski, W. W., and P. K. Smolarkiewicz, 1996: Two-time-level semi-Lagrangian modelling of precipitating clouds. *Mon. Wea. Rev.*, **124**, 487-497.
- Gravel, S., and A. Staniforth, 1992: Variable resolution and robustness. *Mon. Wea. Rev.*, **120**, 2633-2640.
- Gravel, S., A. Staniforth and J. Côté, 1993: A stability analysis of a family of baroclinic semi-Lagrangian forecast models. *Mon. Wea. Rev.*, **121**, 815-824.
- Gravel, S., and A. Staniforth, 1994: A mass-conserving semi-Lagrangian scheme for the shallow-water equations. *Mon. Wea. Rev.*, **122**, 243-248.
- Gustafsson, N., and A. McDonald, 1996: A comparison of the HIRLAM grid-point and spectral semi-Lagrangian models. *Mon. Wea. Rev.*, **124**, 2008-2022.
- Hansen, A., B., J. Brandt, J. H. Christensen, and E. Kaas, 2011: Semi-Lagrangian methods in air pollution models. *Geosci. Model. Dev.*, **4**, 511-541.
- Héreil, Ph., and R. Laprise, 1996: Sensitivity of internal gravity waves solutions to the time step of a semi-implicit semi-Lagrangian non-hydrostatic model. *Mon. Wea. Rev.*, **124**, 972-999.
- Hortal, M., 1994: Recent studies of semi-Lagrangian advection at ECMWF. *ECMWF Research Department Technical Memorandum* No **204**, 37pp.
- Hortal, M., 1996: Experiments with the linear Gaussian grid at ECMWF. *Research Activities in Atmospheric and Oceanic modelling, WGNE report*, No **23**, 3.17-3.19 .
- Hortal, M., 2002: The development and testing of a new two-time-level semi-Lagrangian scheme (SETTLS) in the ECMWF forecast model. *Q. J. R. Meteorol. Soc.*, **128**, 1671-1687.
- Huang, C. Y., 1994: Semi-Lagrangian advection schemes and Eulerian WKL algorithms. *Mon. Wea. Rev.*, **122**, 1647-1658.
- Kaas, E., A. Guldberg and Ph. Lopez, 1996: A full "particle-in-cell" numerical integration method tested for the shallow water equations. *Modern Dynamical Meteorology. Proceedings from a symposium in honour of Prof. Aksel Wiin Nielsen*, 31-38.
- Kaas, E., A. Guldberg and Ph. Lopez, 1997: A Lagrangian advection scheme using tracer points. *Atm. Ocean, 35:sup1, special issue memorial André Robert*, 171-194.
- Kaas, E., 2008: A simple and efficient locally mass conserving semi-Lagrangian transport scheme. *Tellus*, **60**, 305-320.
- Karpic, S. R., 1994: A spline-based semi-Lagrangian method for advection-dominated transport equations on triangular meshes. *Proceedings, 2nd Annual Conference of the CFD Society of Canada*, Toronto. J. J. Gottlieb and C. R. Ethier (eds.), 91-98.
- Laprise, R., 1992: The Euler equations of motion with hydrostatic pressure as an independent variable. *Mon. Wea. Rev.*, **120**, 197-207.
- Laprise, R., and A. Plante, 1995: A class of semi-Lagrangian integrated-mass (SLIM) numerical transport algorithms. *Mon. Wea. Rev.*, **123**, 553-565.
- Lauritzen, P. H., R. D. Nair, and P. A. Ullrich, 2010: A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. *J. Comput. Phys.*, **229**, 14011424.
- Lee, H. N., 1993: A semi-Lagrangian transport scheme with spectral interpolation. *J. Appl. Meteor.*, **32**, 1908-1918.

- Lee, H. N., 1996: A semi-Lagrangian spectral algorithm for pollutant transport and diffusion studies. *J. Appl. Meteor.*, **35**, 2129-2135.
- Lentine, M., J. T. Grétarsson, and R. Fedkiw, 2011: An unconditionally stable fully conservative semi-Lagrangian method. *J. Comput. Phys.*, **230**, 28572879.
- Le Roux, D. Y., C. A. Lin, and A. Staniforth, 1997: An accurate interpolating scheme for semi-Lagrangian advection on an unstructured mesh for ocean modelling. *Tellus*, **49A**, 119-138.
- Leslie, L. M., and R. J. Purser, 1991: High-order numerics in an unstaggered three-dimensional time-split semi-Lagrangian forecast model. *Mon. Wea. Rev.*, **119**, 1612-1623.
- Leslie, L. M., and R. J. Purser, 1995: Three-dimensional mass-conserving semi-Lagrangian scheme employing forward trajectories. *Mon. Wea. Rev.*, **123**, 2551-2566.
- Li, Dong, and Bin Wang, 2012: Trajectory-tracking scheme in Lagrangian form for solving linear advection problems: preliminary tests. *Mon. Wea. Rev.*, **140**, 650-663.
- Lin, Shian-Jiann, and R. B. Rood, 1996: Multidimensional flux-form semi-Lagrangian transport schemes. *Mon. Wea. Rev.*, **124**, 2046-2070.
- Lindberg, K., and V. A. Alexeev, 2000: A study of the spurious orographic resonance in semi-implicit semi-Lagrangian models. *Mon. Wea. Rev.*, **128**, 1982-1989.
- Mc Donald, A., and J. E. Haugen, 1992: A two-time-level, three-dimensional semi-Lagrangian, semi-implicit, limited-area grid-point model of the primitive equations. *Mon. Wea. Rev.*, **120**, 2603-2621.
- Mc Donald, A., and J. E. Haugen, 1993: A two-time-level, three-dimensional semi-Lagrangian, semi-implicit, limited-area grid-point model of the primitive equations. Part II: extension to hybrid vertical coordinates. *Mon. Wea. Rev.*, **121**, 2077-2087.
- Mc Donald, A., 1998: The origin of noise in semi-Lagrangian integrations. *Irish Meteorological Service Technical Note Nr 55*, 27pp.
- Mc Donald, A., 1999: An examination of alternative extrapolations to find the departure point position in a two-time-level semi-Lagrangian integration. *Mon. Wea. Rev.*, **127**, 1985-1993.
- Mc Donald, A., 2000: Boundary conditions for semi-Lagrangian schemes: testing some alternatives in one-dimensional models. *Mon. Wea. Rev.*, **128**, 4084-4096.
- Mc Gregor, J. L., 1993: Economical determination of departure points for semi-Lagrangian models. *Mon. Wea. Rev.*, **121**, 221-230.
- Mc Gregor, J. L., 1996: Semi-Lagrangian advection on conformal-cubic grids. *Mon. Wea. Rev.*, **124**, 1311-1322.
- Machenhauer, B., and M. Olk, 1996: The implementation of the semi-implicit scheme in cell-integrated semi-Lagrangian models. *Modern Dynamical Meteorology. Proceedings from a symposium in honour of Prof. Aksel Wiin Nielsen*, 103-107.
- Makar, P. A., and S. R. Karpik, 1996: Basis-spline interpolation on the sphere: applications to semi-Lagrangian advection. *Mon. Wea. Rev.*, **124**, 182-199.
- Malardel S., and D. Ricard, 2015: An alternative cell-averaged departure point reconstruction for pointwise semi-Lagrangian transport schemes. *Quart. J. Roy. Meteor. Soc.*, **141**, 2114-2126.
- Malevsky, A. V., 1996: Spline-characteristic methods for simulation of convective turbulence. *Journal of Computational Physics*, **123**, 466-475.
- Malevsky, A. V., and S. J. Thomas, 1997: Parallel algorithms for semi-Lagrangian advection. *International Journal for Numerical Methods in fluids*, **25**, 455-473.
- Mawson, M. H., 1995: Implementation of semi-Lagrangian advection in the next generation U.K. Met Office Unified Model. *Forecasting Research Division Scientific Paper*, **37**, 17pp.
- Moorthi, S., R. W. Higgins and J. R. Bates, 1995: A global multilevel atmospheric model using a vector semi-Lagrangian finite-difference scheme: Part II: version with physics. *Mon. Wea. Rev.*, **123**, 1523-1541.
- Nair, R. D., J. Côté and A. Staniforth, 1999: Monotonic cascade interpolation for semi-Lagrangian advection. *Quart. J. Roy. Meteor. Soc.*, **125**, 197-212.
- Nair, R. D., J. Côté and A. Staniforth, 1999: Cascade interpolation for semi-Lagrangian advection over the sphere. *Quart. J. Roy. Meteor. Soc.*, **125**, 1445-1468.
- Nair, R. D., and B. Machenhauer, 2002: The mass-conservative cell-integrated semi-Lagrangian advection scheme on the sphere. *Mon. Wea. Rev.*, **130**, 649-667.
- Nair, R. D., J. S. Scroggs and F. H. M. Semazzi, 2003: A forward-trajectory global semi-Lagrangian transport scheme. *J. Comput. Phys.*, **190**, 275294.
- Oliveira, A., and A. M. Baptista, 1995: A comparison of integration and interpolation Eulerian-Lagrangian methods. *Int. J. Numer. Methods Fluids*, **21**, 183-204.
- Park, S., and N. Baek, 2013: A new trajectory integration scheme in the semi-Lagrangian framework, 2013: *International Journal of Software Engineering and its Applications*, **7**, 179-188.

- Pinty, J.-P., R. Benoît, E. Richard and R. Laprise, 1995: The performance of a semi-Lagrangian transport scheme for the advection-condensation problem. *Mon. Wea. Rev.*, **123**, 3042-3058.
- Pellerin, P., R. Laprise and I. Zawadski, 1995: The performance of a semi-Lagrangian transport scheme for the advection-condensation problem. *Mon. Wea. Rev.*, **123**, 3318-3330.
- Polavarapu, S., M. Tanguay, R. Ménard and A. Staniforth, 1996: The tangent linear model for semi-Lagrangian schemes: linearizing the process of interpolation. *Tellus*, **48A**, 74-95.
- Priestley, A., 1993: A quasi-conservative version of the semi-Lagrangian advection scheme. *Mon. Wea. Rev.*, **121**, 621-629.
- Pudykiewicz, J. A., and A. Kallaur, 1997: Semi-Lagrangian modelling of tropospheric ozone. *Tellus*, **49**, 231-248.
- Purser, R. J., and L. M. Leslie, 1991: An efficient interpolation procedure for high-order three-dimensional semi-Lagrangian models. *Mon. Wea. Rev.*, **119**, 2492-2498.
- Purser, R. J., and L. M. Leslie, 1994: An efficient semi-Lagrangian scheme using third-order semi-implicit time integration and forward trajectories. *Mon. Wea. Rev.*, **122**, 745-756.
- Purser, R. J., and L. M. Leslie, 1996: Generalized Adams-Bashforth time integration schemes for a semi-Lagrangian model employing the second-derivative form of the horizontal momentum equations. *Q. J. R. Meteorol. Soc.*, **122**, 737-763.
- Purser, R. J., and L. M. Leslie, 1997: High-order generalized Lorenz N-cycle schemes for semi-Lagrangian models employing second derivatives in time. *Mon. Wea. Rev.*, **125**, 1261-1276.
- Qian, J. H., F. H. M. Semazzi, and J. S. Scroggs, 1998: A global nonhydrostatic semi-Lagrangian atmospheric model with orography. *Mon. Wea. Rev.*, **126**, 747-771.
- Ramachandran, D. N., and B. Machenhauer, 2002: The mass-conservative cell-integrated semi-Lagrangian advection scheme on the sphere. *Mon. Wea. Rev.*, **130**, 649-667.
- Rančič, M., 1992: Semi-Lagrangian piecewise biparabolic scheme for two-dimensional horizontal advection of a passive scalar. *Mon. Wea. Rev.*, **120**, 1394-1406.
- Rančič, M., 1995: An efficient, conservative, monotonic remapping for semi-Lagrangian transport algorithms. *Mon. Wea. Rev.*, **123**, 1213-1217.
- Ritchie, H., 1986: Eliminating the interpolation associated with the semi-Lagrangian scheme. *Mon. Wea. Rev.*, **114**, 135-146.
- Ritchie, H., 1988: Application of the semi-Lagrangian method to a spectral model of the shallow water equations. *Mon. Wea. Rev.*, **116**, 1587-1598.
- Ritchie, H., 1991: Application of the semi-Lagrangian method to a multilevel spectral primitive-equations model. *Q. J. R. Meteorol. Soc.*, **117**, 91-106.
- Ritchie, H., and C. Beaudoin, 1994: Approximations and sensitivity experiments with a baroclinic semi-Lagrangian spectral model. *Mon. Wea. Rev.*, **122**, 2391-2399.
- Ritchie, H., C. Temperton, A. Simmons, M. Hortal, T. Davies, D. Dent, and M. Hamrud, 1995: Implementation of the semi-Lagrangian method in a high resolution version of the ECMWF forecast model. *Mon. Wea. Rev.*, **123**, 489-514.
- Ritchie, H., and M. Tanguay, 1996: A comparison of spatially-averaged Eulerian and semi-Lagrangian treatments of mountains. *Mon. Wea. Rev.*, **124**, 167-181.
- Rivest, Ch., and A. Staniforth, 1994: Spurious resonant response of semi-Lagrangian discretizations to orographic forcing: diagnosis and solution. *Mon. Wea. Rev.*, **122**, 366-376.
- Rivest, Ch., and A. Staniforth, 1995: Modifying the conventional three-time level semi-implicit semi-Lagrangian scheme to eliminate orographically-induced spurious resonance. *Atm. Ocean*, **33**, 109-119.
- Robert, A., 1981: A stable numerical integration scheme for the primitive meteorological equations. Atmos. Ocean, **19**, 35-46.
- Scroggs, J. C., F. H. M. Semazzi, and D. T. Miller, 1995: A second-order conservative semi-Lagrangian method for multi-dimensional fluid dynamics applications. *Report CRSC-TR95-29, Center for Research in Scientific Computation, North Carolina State University.*
- Seibert, P., and B. Morariu, 1991: Improvements of upstream, semi-Lagrangian numerical advection schemes. *Mon. Wea. Rev.*, **119**, 117-125.
- Semazzi, F. H. M., and P. Dekker, 1994: Optimal accuracy in semi-Lagrangian models. *Mon. Wea. Rev.*, **122**, 2139-2159.
- Semazzi, F. H. M., J. H. Qian, and J. S. Scroggs, 1995: A global nonhydrostatic semi-Lagrangian atmospheric model without orography. *Mon. Wea. Rev.*, **123**, 2534-2550.
- Semazzi, F. H. M., D. W. Webb, and G. Pouliot, 1996: A study of trajectory uncentering in semi-Lagrangian models. *Journal of the Meteorological Society of Japan*, **74**, 695-707.
- Semazzi, F. H. M., J. S. Scroggs, G. Pouliot, A. L. Mac Kee-Burrows, M. Norman, V. Poojary and Y. M. Tsai, 2005: On the accuracy of semi-Lagrangian numerical simulation of internal gravity wave motion in the atmosphere. *Journal of the Meteorological Society of Japan*, **83/5**, 851-869.

- Simmons, A. J., and C. Temperton, 1997: Stability of a two-time-level semi-implicit integration scheme for gravity-wave motion. *Mon. Wea. Rev.*, **125**, 600-615.
- Smolarkiewicz, P. K., and P. J. Rasch, 1991: Monotone advection on the sphere: an Eulerian versus semi-Lagrangian approach. *Journal of the Atmospheric Sciences*, Vol **48** No 6, 793-810.
- Smolarkiewicz, P. K., 1991: On forward-in-time differencing for fluids. *Mon. Wea. Rev.*, **119**, 2505-2510.
- Smolarkiewicz, P. K., and J. A. Pudykiewicz, 1992: A class of semi-Lagrangian approximations for fluids. *Journal of the Atmospheric Sciences*, **49**, 2082-2096.
- Smolarkiewicz, P. K., and G. Grell, 1992: A class of monotone interpolation schemes. *J. Comput. Phys.*, **101**, 431440.
- Smolarkiewicz, P. K., and L. G. Margolin, 1997: On forward-in-time differencing for fluids: an Eulerian/Semi-Lagrangian non-hydrostatic model for stratified flows. *Atm. Ocean, 35:sup1, special issue memorial André Robert*, 127-152.
- Sorensen, B., E. Kaas, and U. S. Korsholm, M., 2013: A mass-conserving and multi-tracer efficient transport scheme in the online integrated Enviro-HIRLAM model. *Geosci. Model. Dev.*, **6**, 1029-1042.
- Spiridonov, V., 1988: On the problem of time-dependent lateral boundary conditions for limited area atmospheric models. Some advantages of the semi-Lagrangian method. *PSMP report series ECMWF*, **27**, 89-92.
- Staniforth, A., and J. Côté, 1991: Semi-Lagrangian integration schemes for atmospheric models - A review. *Mon. Wea. Rev.*, **119**, 2206-2223.
- Staniforth, A., A. White, and N. Wood, 2003: Analysis of semi-Lagrangian trajectory computations. *Q. J. R. Meteorol. Soc.*, **129**, 2065-2085.
- Staniforth, A., and N. Wood, 2005: An Unsuspected Boundary-Induced Temporal Computational Mode in a Two-Time-Level Discretization. *Mon. Wea. Rev.*, **133**, 712-720.
- Staniforth, A., A. White, and N. Wood, 2010: Treatment of vector equations in deep-atmosphere, semi-Lagrangian models. I: Momentum equation. *Q. J. R. Meteorol. Soc.*, **136**, 497-506.
- Staniforth, A., A. White, and N. Wood, 2010: Treatment of vector equations in deep-atmosphere, semi-Lagrangian models. II: Kinematic equation. *Q. J. R. Meteorol. Soc.*, **136**, 507-516.
- Steppeler, J., and G. Ország, 1995: A non interpolating semi-Lagrangian method based on transformations to an Eulerian grid. *Beitr. Phys. Atmosph.*, **68**, 263-270.
- Steppeler, J., and P. Prohl, 1996: Applications of finite volume methods to atmospheric models. *Beitr. Phys. Atmosph.*, **69**, 297-306.
- Sun, W.-Y., K.-S. Yeh, and R.-Y. Sun, 1996: A simple semi-Lagrangian scheme for advection equations. *Q. J. R. Meteorol. Soc.*, **122**, 1211-1226.
- Sun, W.-Y. and K.-S. Yeh, 1997: A general semi-lagrangian advection scheme employing forward trajectories. *Q. J. R. Meteorol. Soc.*, **123**, 2463-2476.
- Tanguay, M., A. Simard and A. Staniforth, 1989: A three-dimensional semi-Lagrangian scheme for the Canadian regional finite-element forecast model. *Mon. Wea. Rev.*, **117**, 1861-1871.
- Tanguay, M., E. Yakimiw, H. Ritchie, and A. Robert, 1992: Advantages of spatial averaging in semi-implicit semi-Lagrangian schemes. *Mon. Wea. Rev.*, **120**, 113-123.
- Tanguay, M., S. Polavarapu, and P. Gauthier, 1997: Temporal accumulation of first-order linearization error for semi-Lagrangian passive advection. *Mon. Wea. Rev.*, **125**, 1296-1311.
- Temperton, C., 1997: Treatment of the Coriolis terms in semi-Lagrangian spectral models. *Atm. Ocean, 35:sup1, special issue memorial André Robert*, 293-302.
- Thomas, S., and J. Côté, 1995: Massively parallel semi-Lagrangian advection. *J. Simulation, Practice and Theory*, **3**, 223-238.
- Thuburn, J., M. Zerroukat, N. Wood and A. Staniforth, 2010: Coupling a mass-conserving semi-Lagrangian scheme (SLICE) to a semi-implicit discretization of the shallow-water equations: Minimizing the dependence on a reference atmosphere. *Q. J. R. Meteorol. Soc.*, **136**, 146-154.
- Thuburn, J., and A. A. White, 2013: A geometrical view of the shallow-water approximation, with application to the semi-Lagrangian departure point calculation. *Q. J. R. Meteorol. Soc.*, **139**, 261-268.
- Tolstykh, M. A., 1994: Application of Fifth-Order Compact Upwind Differencing to moisture transport equation in atmosphere. *Journal of Computational Physics*, **112** No 2, 394-403.
- Tolstykh, M. A., 1995: The modified two-time level scheme in the global semi-Lagrangian shallow-water model. *Research Activities in Atmospheric and Oceanic modelling, WGNE report*, No **21**, 3.28-3.29 .
- Tolstykh, M. A., 1996: The response of the variable resolution semi-Lagrangian NWP model to the change in horizontal interpolation. *Q. J. R. Meteorol. Soc.*, **122**, 765-778.
- Tolstykh, M. A., 2002: Vorticity-divergence semi-Lagrangian shallow-water model on the sphere based on compact finite differences. *J. Comput. Phys.*, **179**, 180200.

- Tolstykh, M. A., and V. V. Shashkin, 2012: Vorticity-divergence mass-conserving semi-Lagrangian shallow-water model using the reduced grid on the sphere. *J. Comput. Phys.*, **231**, 42054233.
- Váňa F., P. Bénard, J.F. Geleyn, A. Simon, and Y. Seity, 2008: Semi-Lagrangian advection scheme with controlled damping: an alternative to nonlinear horizontal diffusion in a numerical weather prediction model. *Quart. J. Roy. Meteor. Soc.*, **134**, 523-537.
- White, A. A., 2003: Dynamical equivalence and the departure-point equation in semi-Lagrangian numerical models. *Q. J. R. Meteorol. Soc.*, **129**, 1317-1324.
- Wong, M., W. C. Skamarock, P. H. Lauritzen, and R. B. Stull, 2013: A cell-integrated semi-Lagrangian semi-implicit shallow-water model (CSLAM-SW) with conservative and consistent transport. *Mon. Wea. Rev.*, **141**, 2545-2560.
- Wood, N., A. Staniforth, and A. White, 2009: Determining near-boundary departure points in semi-Lagrangian models. *Q. J. R. Meteorol. Soc.*, **135**, 1890-1896.
- Xindong Peng, Feng Xiao, Wataru Ohfuchi and Hiromitsu Fuchigami, 2005: Conservative Semi-Lagrangian Transport on a Sphere and the Impact on Vapor Advection in an Atmospheric General Circulation Model. *Mon. Wea. Rev.*, **133**, 504-520.
- Yabe, T., R. Tanaka, T. Nakamura, and F. Xiao, 2001: An exactly conservative semi-Lagrangian scheme (CIP-CSL) in one dimension. *Mon. Wea. Rev.*, **129**, 332-344.
- Zerroukat, M., N. Wood and A. Staniforth, 2002: SLICE: a semi-Lagrangian inherently conserving and efficient scheme for transport problems. *Q. J. R. Meteorol. Soc.*, **128**, 2801-2820.
- Zerroukat, M., N. Wood and A. Staniforth, 2004: SLICE: a semi-Lagrangian inherently conserving and efficient scheme for transport problems on the sphere. *Q. J. R. Meteorol. Soc.*, **130**, 2649-2664.
- Zerroukat, M., N. Wood and A. Staniforth, 2005: A monotonic and positive-definite filter for a semi-lagrangian inherently conserving and efficient (SLICE) scheme. *Q. J. R. Meteorol. Soc.*, **131**, 2923-2936.
- Zerroukat, M., N. Wood and A. Staniforth, 2009: An improved version of SLICE for conservative monotonic remapping on a C-grid. *Q. J. R. Meteorol. Soc.*, **135**, 541-546.
- Zerroukat, M. and T. Allen, 2012: A three-dimensional monotone and conservative semi-Lagrangian scheme (SLICE-3D) for transport problems. *Q. J. R. Meteorol. Soc.*, **138**, 1640-1651.
- Zhang, Y., and H-M. H. Juang, 2012: A mass-conserving non-iteration-dimensional-split semi-Lagrangian advection scheme for limited-area modelling. *Q. J. R. Meteorol. Soc.*, **138**, 2118-2125.

20.2 Internal notes, workshop proceedings.

- (TDECDYN) 2017: IFS technical documentation (CY43R3). Part III: dynamics and numerical procedures. Available at "https://software.ecmwf.int/wiki/display/IFS/Official+IFS+Documentation".
- (TDECTEC) 2017: IFS technical documentation (CY43R3). Part VI: technical and computational procedures. Available at "https://software.ecmwf.int/wiki/display/IFS/Official+IFS+Documentation".
- (IDNHPB) Bénard, P., 2013: Scientific documentation for ALADIN NH model (version 3.1). Internal note (94pp), available on "http://www.umr-cnrm.fr/gmapdoc/".
- Besse, N., 2013: A non exhsustive review of semi-Lagrangian methods for the Vlasov equation. Slides for presentation, 30pp.
- Bonaventura, L., 2002: An introduction to semi-Lagrangian methods for geophysical flows. Internal note (44pp).
- Craciun, A., 2014: Application of ENO technique to semi-Lagrangian interpolations. *RC LACE stay report*, 14pp.
- De Boor, C., 2001: A practical guide to splines. *Applied mathematical sciences 27, Springer-Verlag: New York.*
- Diamantakis, M., 2014: A modification of the IFS semi-Lagrangian trajectory scheme to improve sudden stratospheric warming forecasts. *ECMWF Research Department Technical Memorandum* No **???**, 13pp.
- Diamantakis, M., 2014: Improving ECMWF forecasts of sudden stratospheric warmings. *ECMWF Newsletter nr* **141**, 30-36.
- Dugault, E., 1995: Evaluation du décentrage variable dans une version semi-lagrangienne à maille variable du modèle ARPEGE. *Rapport de stage universitaire*, 25pp.
- Ekman, A., and E. Källén, 1998: Mass conservation tests with the HIRLAM, semi-Lagrangian, time integration scheme. *HIRLAM technical report nr* **39**, 23pp.
- Gospodinov, I., 2000: Two-time-level semi-Lagrangian method for numerical weather prediction models: enhanced conservation and stability. *Manuscrit de thèse*, 121pp.
- Haugen, J. E., and A. Mc Donald, 1991: Testing the two-time level and the three-time level semi-Lagrangian HIRLAM models. *Report for HIRLAM group*, 26pp.
- Héreil, Ph., 1993: Sensibilité de la solution d'ondes de relief au choix du pas de temps dans un modèle non hydrostatique semi-implicite semi-Lagrangien. *Rapport de stage de DEA*, 63pp.

- Hortal, M., 1998: New two-time-level semi-Lagrangian scheme. *Memorandum research department*, 18pp.
- (IDSVTSM) Hortal, M., and A. Untch, 2003: A new interpolation for the vertical computation of the semi-Lagrangian trajectory. Internal note, 7pp.
- (IDSLIF) Mašek, J., and F. Váňa, 2006: Study of semi-Lagrangian interpolators in idealized framework. Report from RC LACE scientific stay, 27pp.
- (IDSLIF2) Mašek, J., and F. Váňa, 2007: Test implementation of new semi-Lagrangian interpolators in ARPEGE/ALADIN cycle 32t1alr01. Report from RC LACE scientific stay, 15pp.
- Mc Donald, A., 1995: The HIRLAM two time level, three dimensional semi-Lagrangian, semi-implicit, limited area, grid-point model of the primitive equations. *HIRLAM technical report nr* **17**, 25pp (available from SMHI).
- Mc Donald, A., 1996: Controlling noise in two time level semi-Lagrangian semi-implicit integrations; comments on some new developments. *HIRLAM technical report nr* **24**, 4-10.
- Mc Donald, A., 1998: Alternative extrapolations to find the departure point in a two time level semi-Lagrangian integration. *HIRLAM Technical Report*, **34**, 17pp.
- Mc Donald, A., 1999: Well-posed boundary conditions for semi-Lagrangian schemes: the one-dimensional case. *HIRLAM technical report nr* **43**, 28pp.
- Mc Donald, A., 2000: Well-posed boundary conditions for semi-Lagrangian schemes: the one-dimensional case, part II. *HIRLAM technical report nr* **44**, 21pp.
- (IDEQR) Mozdzynski, G., 2006: A new partitioning approach for IFS. Internal note, 6pp.
- Rivals, H., 1993: Validation d'un modèle non hydrostatique semi-implicite semi-Lagrangien par des simulations d'ondes de relief. *Rapport de stage de DEA*, 69pp.
- Simmons, A. J., and C. Temperton, 1996: Stability of a two-time-level semi-implicit integration scheme for gravity-wave motion. *ECMWF Technical Memorandum*, **226**, 33pp.
- (IDVNH1) Smolíková, P., 2001: New strategies for non-hydrostatic temporal scheme. Internal note.
- (IDVNH2) Smolíková, P., 2002: New NH variables: d_4 in three dimensions (in the cycle CY25T1). Internal note.
- (IDVNH3) Smolíková, P., 2003: The use of diagnostic BBC (boundary bottom condition) in semi-Lagrangian schemes. Internal note, 8pp.
- Smolíková, P., and Jozef Vivoda, 2013: Finite elements used in the vertical discretization of the fully compressible forecast model ALADIN-NH. *ALADIN - HIRLAM Newsletter no* **1**, 31-47.
- Spiegelmann, M., and R. F. Katz, 2002: A semi-Lagrangian Crank-Nicholson algorithm for the numerical solution of advection-diffusion problems. Internal note, 10pp.
- Spiridonov, V., 1992: Down-stream semi-Lagrangian method. Internal note, 10pp.
- Váňa, F., P. Bénard, and J.-F. Geleyn, 2001: Semi-Lagrangian advection scheme with controlled damping: an alternative way to nonlinear horizontal diffusion in a semi-Lagrangian numerical weather prediction model. Internal note, 35pp.
- (IDSPL)
  Vaña, F., 2005: Spline interpolation in semi-Lagrangian advection scheme of ALADIN/ARPEGE/IFS. Internal note, 8pp.
- Vivoda, J., 2017: Dynamical PC scheme for NH kernel of AAA models. Part 1: Theoretical considerations and implementation of LSETTLS residual extrapolation into LPC CHEAP scheme. Internal note, RC LACE stay report, 6pp.
- (IDBAS) Yessad, K., 2018: Basics about ARPEGE/IFS, ALADIN and AROME in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDSI) Yessad, K., 2018: Semi-implicit spectral computations in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDDH) Yessad, K., 2018: Horizontal diffusion in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDTS) Yessad, K., 2018: Spectral transforms in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDDM) Yessad, K., 2018: Distributed memory features in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDEUL) Yessad, K., 2018: Integration of the model equations, and Eulerian dynamics, in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDLAM) Zagar, M., and C. Fischer, 2007: The ARPEGE/ALADIN Tech'Book: Implications of LAM aspects on the global model code for CY33/AL33. Internal note, 31pp, available on the internet server "http://www.umr-cnrm.fr/gmapdoc/".
- Workshop proceedings on semi-Lagrangian methods, 6-8 November 1995, 264pp. Available at ECMWF.

Appendix 1: Description of treatment of X for semi-Lagrangian advection.

20.3 3TL Semi-Lagrangian advection.

The calculation of $\frac{d\mathtt{X}}{dt}$ is not done by a temporal advection but simply by a diagnostic evaluation and the management of this term.

If there is no predictor-corrector scheme activated, this term is diagnosed as follows:

- Three-time level semi-Lagrangian scheme for **ND4SYS**=1:
  - Calculation of $d\mathtt{X}/dt$ uses formula:
$$d\mathtt{X}/dt = [0.5\mathtt{X}_F^o + 0.5\mathtt{X}_O^o - \mathtt{X}_O^-]/[\Delta t]$$
  - These calculations are done under **LATTEX** (and **LAPINEB** for interpolations).
  - $d\mathtt{X}/dt$ is used to compute dd_4/dt from dd/dt, not to compute the $t+\Delta t$ value of X.
  - Routine **CPG_GP_NHEE** recomputes $\mathtt{X}^o$ and $\mathtt{X}^-$ by calling **GPXX**.
- Three-time level semi-Lagrangian scheme for **ND4SYS**=2:
  - Calculation of $d\mathtt{X}/dt$ uses formula:
$$d\mathtt{X}/dt = [\mathtt{X}_F^{+o} - \mathtt{X}_F^o]/[\Delta t] + [\mathtt{X}_F^o - \mathtt{X}_O^o]/[\Delta t]$$
  - For advective part $[\mathtt{X}_F^o - \mathtt{X}_O^o]$, calculations are done under **LATTEX** (and **LAPINEB** for interpolations).
  - $\mathtt{X}_F^{+o}$ is computed using a mix of t quantities (horizontal derivatives) and provisional $t+\Delta t$ variables in **CPGLAG**.
  - $d\mathtt{X}/dt$ is used to compute dd_4/dt from dd/dt.
  - $\mathtt{X}_F^{+o}$ is passed to spectral transforms, it provides $\mathtt{X}^o$ and $\nabla\mathtt{X}^o$ of the following timestep. $\mathtt{X}^o$ becomes the $\mathtt{X}^-$ of the following timestep.
  - There is no recalculation of $\mathtt{X}^o$ under **CPG_GP_NHEE**.

Note that PC scheme is not implemented for SL3TL advection.

Both options **ND4SYS**=1 and **ND4SYS**=2 are coded; **ND4SYS**=1 is obsolescent.

20.4 2TL Semi-Lagrangian advection.

The calculation of $\frac{d\mathtt{X}}{dt}$ is not done by a temporal advection but simply by a diagnostic evaluation and the management of this term.

If there is no predictor-corrector scheme activated, this term is diagnosed as follows:

- Two-time level semi-Lagrangian scheme for **ND4SYS**=1:
  - Calculation of $d\mathtt{X}/dt$ uses formula:
$$d\mathtt{X}/dt = [\mathtt{X}_M^m - \mathtt{X}_O^o]/[0.5\Delta t]$$
  - $\mathtt{X}_M^m$ is a symbolic denotation for the extrapolated $t+0.5\Delta t$ value of X. The way of extrapolating depends on **LSETTLS** and **LNESC** and is the same as for the other terms evaluated at $t+0.5\Delta t$ (there is no interpolation at M).
  - These calculations are done under **LATTEX** (and **LAPINEB** for interpolations).
  - $d\mathtt{X}/dt$ is used to compute dd_4/dt from dd/dt, not to compute the $t+\Delta t$ value of X.
  - Routine **CPG_GP_NHEE** recomputes $\mathtt{X}^o$ by calling **GPXX**.
- Two-time level semi-Lagrangian scheme for **ND4SYS**=2:
  - Calculation of $d\mathtt{X}/dt$ uses formula:
$$d\mathtt{X}/dt = [\mathtt{X}_F^{+o} - \mathtt{X}_F^o]/[0.5\Delta t] + [\mathtt{X}_F^o - \mathtt{X}_O^o]/[0.5\Delta t]$$

- For advective part $[\mathtt{X}_F^o - \mathtt{X}_O^o]$, calculations are done under **LATTEX** (and **LAPINEB** for interpolations).
- $\mathtt{X}_F^{+o}$ is computed using a mix of t quantities (horizontal derivatives) and provisional $t + \Delta t$ variables in **CPGLAG**.
- $d\mathtt{X}/dt$ is used to compute dd_4/dt from dd/dt.
- $\mathtt{X}_F^{+o}$ is passed to spectral transforms, it provides $\mathtt{X}^o$ and $\nabla\mathtt{X}^o$ of the following timestep.
- There is no recalculation of $\mathtt{X}^o$ under **CPG_GP_NHEE**.

If there is a predictor-corrector scheme activated:

- for **ND4SYS**=1 all the RHS of $d\mathtt{X}/dt$ is iterated.
- for **ND4SYS**=2 only the advective part containing $[\mathtt{X}_F^o - \mathtt{X}_O^o]$ is iterated; the non advective part containing $[\mathtt{X}_F^{+o} - \mathtt{X}_F^o]$ is computed and added at the last corrector step only.

Both options **ND4SYS**=1 and **ND4SYS**=2 are coded. **ND4SYS**=2 is more stable, especially at fine resolutions. **ND4SYS**=1 is obsolescent.

Appendix 2: Description of dataflow for option LGWADV=T.

Option **LGWADV**=T is currently coded only for the semi-Lagrangian advection. The following description is given for a SL2TL advection and the NHEE model, we limit it to **LVFE_GW**=F.

Main features of **LGWADV**=T option:

- Non-linear terms are computed for w equation, at half-levels.
- w_{surf} requires a specific treatment.
- Linear terms are computed for d or d_4 equation at full-levels.
- The RHS of w equation appears only in the grid-point calculations.
- Spectral calculations involve (and update) d or d_4, not w.

$*$ More details about calculations for SL2TL advection:

- Under **(E)TRANSINVH**:
  - $d(t)$ or $d_4(t)$ is transformed into grid-point space, with horizontal derivatives calculation.
  - $\mathtt{X}(t)$ is transformed into grid-point space, with horizontal derivatives calculation, if **NVDVAR**=4.
  - w never appears.
- Under **CPG**:
  - Calculation of $w(t)$ from $d(t)$ is done in **CPG_GP_NHEE** by calling **GPXX**.
  - The RHS of w equation is computed in **GNH_TNDLAGADIAB_GW** if **LNHEE_SVDLAPL_FIRST**=F (called by **CPG_GP_NHEE**).
  - The RHS of w equation is computed in **GNHEE_TNDLAGADIAB_GW** if **LNHEE_SVDLAPL_FIRST**=T (called by **CPG_GP_NHEE**).
  - Array PATND(.,1:NFLEVG,YYTTND%M_TNDGW) contains the RHS of w equation for upper-air half-levels. Note that half-level number $\bar{l}-1$ is stored in index jlev= $\bar{l}$ in PATND.
  - The RHS of d equation is not computed.
- Under **CPG_DYN**:
  - The first call to **LANHSI** in **LACDYN** provisionally fills linear terms with 0.
  - **LATTEX**, called by **LACDYN**, fills buffers PB1 and PB2 (pointers MSLB1VD.., MSLB2VD..) using the explicit RHS of w equation. Upper-air half-levels are treated, index numbering is from 1 to NFLEVG in **LATTEX**.
  - The second call to **LANHSI** in **LACDYN** computes linear terms for d equation; **LANHSIB** updates PB1 and PB2 with linear terms.
- Under **CALL_SL**:
  - Interpolations for the RHS of w equation are done in **LARCINHB**, called by **LAPINEB**.
  - Part 2.3.1b of **LAPINEB** updates PGMVT1(.,.,YT1%MSVD) with explicit part of the RHS of w equation: terms at the final point, interpolated terms at the origin point. Level numbering in PGMVT1 is 1 to NFLEVG (match with upper-air half-levels 0 to NFLEVG-1).
  - Note that the interpolator (in particuliar **LASCAW**) does not know if we treat full-level or half-level quantities. All quantities entering the interpolator are assumed to be numbered from 1 to NFLEVG, never from 0 to NFLEVG-1.
  - Part 2.3.5 of **LAPINEB** updates $w_{surf}(t+\Delta t)$ using the surface condition. This surface condition uses the provisional $\mathbf{V}(t+\Delta t)$ which is computed in **LAPINEB**; it does not contain the semi-implicit correction nor horizontal diffusion; surface value of $\mathbf{V}(t+\Delta t)$ is assumed to be equal to full-level l=NFLEVG value of $\mathbf{V}(t+\Delta t)$.
  - Part 2.3.5 of **LAPINEB** converts $w(t+\Delta t)$ into $d(t+\Delta t)$ by calling **GNHGW2SVD**. Values of T, p, Π used inside **GNHGW2SVD** are provisional $t+\Delta t$ ones. Note that output of **GNHGW2SVD** is always d (without linear terms), never d_4.

  - From now on we do not use any more the w equation.
- Under **CPGLAG**:
  - Computes $d_4(t+\Delta t)$ from $d(t+\Delta t)$ if **NVDVAR**=4.
  - Adds some linear terms to $d(t+\Delta t)$ or $d_4(t+\Delta t)$.
  - At the end of **CPGLAG**, PGMVT1(.,.,YT1%MSVD,.) contains $d(t+\Delta t)$ or $d_4(t+\Delta t)$ according to **NVDVAR**, this is the value which enters spectral transforms.
- Under **ECOUPL1** and **ECOUPL2** for LAM models:
  - Coupling is done on $d(t+\Delta t)$ or $d_4(t+\Delta t)$.
  - Coupling is done on $\mathtt{X}(t+\Delta t)$ too if **NVDVAR**=4.
- Under **(E)TRANSDIRH**:
  - $d(t+\Delta t)$ or $d_4(t+\Delta t)$ is transformed into spectral space.
  - $\mathtt{X}(t+\Delta t)$ is transformed into spectral space if **NVDVAR**=4.
  - w never appears.
- Under **(E)SPCM**:
  - Inversion of linear system is done on d or d_4 equation.
  - Horizontal diffusion is applied to d or d_4.
  - Horizontal diffusion is applied to `X` too if **NVDVAR**=4.
  - w never appears.
- Remark: **LGWADV**=T combined with **LSETTLS**=T may generate more complicated code with keys like **LSLINLC2** and **LSLINL** set to T; this option coded for testings and possible use of **LGWADV**=T without PC scheme is generally unstable and not used. In this case we need to interpolate some linear terms at the origin point, they must be put in separate buffers.

∗ Remarks for SL3TL advection:

LGWADV=T cannot be used for the time being for the following reasons (missing pieces of code):

- Interpolations at the origin point are required for linear terms.
- For **LGWADV**=T linear terms and non-linear terms must be interpolated separately (like for **LGWADV**=T with **LSETTLS**=T in the SL2TL scheme).
- There are currently missing pieces of code to manage these separate interpolations.