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. 5

2 Definition of Eulerian and semi-Lagrangian schemes. 6
  2.1 Eulerian scheme. .......................................................... 6
  2.2 Semi-Lagrangian scheme. ............................................... 6

3 The 2D equations. 7
  3.1 Denotations for the 2D equations. ................................. 7
  3.2 The 2D shallow-water system of equations in spherical geometry. 7
    3.2.1 Momentum equation. ................................................ 7
    3.2.2 Continuity equation. .............................................. 7

4 The 3D equations in spherical geometry (ARPEGE/IFS). 8
  4.1 Denotations for the 3D equations. .................................. 8
  4.2 The 3D primitive equation model. ................................... 9
    4.2.1 Momentum equation. ............................................... 9
    4.2.2 Thermodynamic equation. ..................................... 9
    4.2.3 Continuity equation. ........................................... 9
    4.2.4 Advecable GFL variables. ..................................... 9
    4.2.5 Non advectable pseudo-historic GFL variables: ............ 9
  4.3 The 3D NHEE fully elastic non-hydrostatic model. .................. 10
    4.3.1 Basics about NHEE equations. ................................ 10
    4.3.2 Momentum equation. ............................................ 10
    4.3.3 Thermodynamic equation. ..................................... 10
    4.3.4 Vertical velocity \( w \) and vertical divergence \( d \) equations. 10
    4.3.5 Pressure departure variable \( Q \) equation. ....................... 11
    4.3.6 Continuity and GFL equations. ................................ 11
  4.4 The 3D NHQE quasi-elastic non-hydrostatic model. ................... 11
    4.4.1 Basics about NHQE equations. ................................ 11
    4.4.2 Momentum equation. ............................................ 12
    4.4.3 Thermodynamic equation. ..................................... 12
    4.4.4 Vertical velocity \( w \) and vertical divergence \( d \) equations. 12
    4.4.5 Continuity and GFL equations. ................................ 12

5 Discretisation of the equations: general aspects. 13
  5.1 Denotations. ............................................................... 13
  5.2 Generalised formulation of extrapolations (SL2TL schemes). .......... 13
  5.3 Uncentering coefficients. .............................................. 14
  5.4 Discretisation for a 3D variable in a 3D model: general case where the RHS has non-zero linear
    and non-linear terms (GMY variables). ................................ 14
    5.4.1 Introduction and preliminary remarks. ......................... 14
    5.4.2 3TL vertical interpolating SL scheme. .......................... 15
    5.4.3 2TL vertical interpolating SL scheme: conventional extrapolation. 15
    5.4.4 2TL vertical interpolating SL scheme: stable extrapolation and mixed extrapolation. 16
  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). .......................... 17
    5.5.1 Introduction and preliminary remarks. ......................... 17
    5.5.2 3TL vertical interpolating SL scheme. .......................... 18
    5.5.3 2TL vertical interpolating SL scheme. .......................... 18
    5.5.4 Non advectable pseudo-historic GFL variables. ................ 18
  5.6 Discretisation for a 2D variable in a 3D model (GMYV variables, for example continuity equation). 18
    5.6.1 Introduction and preliminary remarks. ......................... 18
    5.6.2 3TL vertical interpolating SL scheme. .......................... 19
    5.6.3 2TL vertical interpolating SL scheme: conventional extrapolation. 20
    5.6.4 2TL vertical interpolating SL scheme: stable extrapolation or mixed extrapolation. 20
  5.7 Discretisation for a 2D variable in a 2D model. ..................... 20
  5.8 Additional remarks. ...................................................... 21

6 Computation of medium and origin points. 22
  6.1 Medium point \( M \) and origin point \( O \) for LSLDP_RK=F and LSLDP_CURV=F. ........ 22
    6.1.1 Conventional algorithm. ........................................ 23
    6.1.2 "SETTLTS" stable algorithm for SL2TL scheme. ................ 24
    6.1.3 "LELTRA" alternate stable algorithm for SL2TL scheme. .... 24
    6.1.4 Algorithm used with the iterative centred-implicit schemes. .... 25
    6.1.5 Compute origin point \( O \) from medium point \( M \) ............. 26
  6.2 Refined recomputation of point \( O \). .................................. 26
  6.3 Trajectory going out of atmosphere or underground, and algorithms to prevent that. ................ 26
  6.4 Remarks. ........................................................................ 27
7 The SL discretisation of the 2D shallow-water system of equations (spherical geometry). 29
7.1 Momentum equation. ................................................................. 29
7.2 Continuity equation: conventional formulation (NVLAG>0). .............. 29
7.3 Continuity equation: Lagrangian formulation (NVLAG<0). .............. 29
7.4 Quantities to be interpolated. .................................................. 29

8 The SL discretisation of the 3D primitive equation model. 30
8.1 Formulation of the momentum equation. ..................................... 30
8.2 Thermodynamic equation. ....................................................... 30
8.3 Continuity equation. ............................................................... 30
8.4 Moisture equation. ................................................................. 31
8.5 Other advectable GFL variables. ................................................ 31
8.6 Case of lagged physics ............................................................ 31
8.7 Quantities to be interpolated (computation under subroutine LACDYN). 31
8.8 Case of 3D simulations ........................................................... 31

9 The SL discretisation of the fully elastic non hydrostatic (NHEE) model. 32
9.1 Momentum equation. ............................................................... 32
9.2 Temperature equation. ........................................................... 32
9.3 Vertical divergence variable equation. ........................................ 32
9.4 Pressure departure variable equation. ........................................ 33
9.5 Continuity equation and GFL equations. ...................................... 33
9.6 Case of lagged physics ............................................................ 33
9.7 Quantities to be interpolated (computation under subroutine LACDYN). 33
9.8 Case of 3D simulations ........................................................... 33
9.9 Other advectable GFL variables. ................................................ 33

10 The SL discretisation of the quasi-elastic non hydrostatic (NHQE) model. 34
10.1 Momentum equation. ............................................................. 34
10.2 Temperature equation. .......................................................... 34
10.3 Vertical divergence variable equation. ....................................... 34
10.4 Continuity equation and GFL equations. .................................... 35
10.5 Case of lagged physics ........................................................... 35
10.6 Quantities to be interpolated (computation under subroutine LACDYN). 35
10.7 Case of 3D simulations .......................................................... 35
10.8 Other advectable GFL variables. ................................................ 35

11 $\vec{\gamma}$ operator. ................................................................. 36

12 Computation of longitudes and latitudes on the computational sphere. 37

13 Computation of $\vec{\gamma}$ at full levels. ........................................... 37

14 Interpolations and weights computations. 38
14.1 Interpolation grid and weights (subroutine LASCWA). .................... 38
14.1.1 Horizontal interpolation grid and weights for bi-linear interpolations. 38
14.1.2 Vertical interpolation grid and weights for vertical linear interpolations. 38
14.1.3 Horizontal interpolation grid and weights for 12 points cubic interpolations. 38
14.1.4 Vertical interpolation grid and weights for vertical cubic 4 points interpolations. 40
14.1.5 Vertical interpolation grid and weights for vertical cubic Hermite interpolations. 40
14.1.6 Vertical interpolation grid and weights for vertical cubic spline interpolations. 41
14.1.7 Horizontal interpolation grid and weights for tri-linear interpolations. 41
14.1.8 Interpolation grid and weights for 32 points linear interpolations. 41
14.1.9 Horizontal interpolation grid and weights for 16 points linear least-square fit interpolations. 42
14.1.10 Interpolation grid and weights for 32 points least-square fit interpolations. 42
14.1.11 Interpolation grid and weights for 56 points “WENO” interpolations. 42
14.1.12 Modified weights for COMAD interpolations. 42
14.1.13 Plane geometry (LAM models). ........................................... 43

14.2 Interpolations. ................................................................. 38
14.2.1 Bilinear interpolation (subroutine LAIDLI). ............................ 43
14.2.2 Tri-linear interpolation (subroutine LAITLI). ........................ 43
14.2.3 Horizontal 12 points interpolation (subroutine LAIDDI). .......... 43
14.2.4 Cubic 4 points vertical interpolation. .................................... 44
14.2.5 Cubic Hermite vertical interpolation. .................................... 44
14.2.6 Spline cubic 4 points vertical interpolation. .......................... 44
14.2.7 32 points 3D interpolation with vertical cubic 4 points interpolation (subroutine LAITRI). 44
14.2.8 32 points 3D interpolation with vertical cubic Hermite interpolation (subroutine LAITVSPCQM). 44
14.2.9 48 points 3D interpolation with vertical cubic spline interpolation (subroutine LAITVSPCQM). 45
14.2.10 Horizontal 16 points linear least-square fit interpolation. ........ 45
14.2.11 32 points 3D interpolation with linear least-square fit interpolations and vertical linear interpolations (subroutine LAISMOO). 45
14.2.12 36 points 3D “WENO” interpolations (subroutine LAITRIWENO). 45

14.3 Code structures to store weights. ........................................... 45

3
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:

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

where $\mathbf{U}$ is the 3D wind, $\nabla_3$ 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:

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

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), $|\mathbf{V}|$ 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:

$$M \frac{|\mathbf{V}|}{a} \frac{Dt}{2} (2\pi) \sqrt{\frac{1}{N_m^2 N_n^2 + \frac{L_x^2}{a^2 N_m^2} + \frac{L_y^2}{a^2 N_n^2} < 1}} \quad (3)$$

See above for denotations $M$, $Dt$, $|\mathbf{V}|$, $a$. $N_m$ 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:

$$0.5 \left| \frac{\partial \eta}{\partial t} \right| \Delta \eta < 1 \quad (4)$$

2.2 Semi-Lagrangian scheme.

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

$$\frac{dX}{dt} = \dot{X} \quad (5)$$

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 $\eta$-coordinates, $\dot{\eta} = \frac{d\eta}{dt}$. Lipschitz criterion writes for a three-time level semi-Lagrangian scheme:

$$\left| D + \frac{\partial \dot{\eta}}{\partial \eta} \right| \frac{Dt}{2} < 1 \quad (6)$$

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

$$\left| D + \frac{\partial \dot{\eta}}{\partial \eta} \right| \frac{\Delta t}{2} < 1 \quad (7)$$

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”.

6
3 The 2D equations.

3.1 Denotations for the 2D equations.
- $V$ is the horizontal wind. Zonal and meridian components on computational grid are $U$ and $V$.
- $D$ is the horizontal wind divergence; $\zeta$ is the horizontal wind vorticity.
- $\Phi$ is the equivalent height. $\Phi_s$ is the surface geopotential height (i.e. the orography). $\Phi^*$ is a reference equivalent height which is only used in the semi-implicit scheme and the linear model.
- $\Omega$ is the Earth rotation angular velocity.
- $\nabla$ is the first order horizontal gradient on $\eta$-surfaces.
- $a$ is the Earth radius.
- $\lambda_{\text{base}}, \theta_{\text{base}}$ are the longitude-latitude coordinates on a tilted and not stretched geometry, the tilting being the same as the one of the computational sphere.
- $k$ is the unit vertical vector. One can write:

$$k = \frac{r}{|r|} = \frac{r}{a}$$

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

3.2.1 Momentum equation.
Coriolis force can be treated explicitly ($\delta V=0$) or implicitly ($\delta V=1$) in the Lagrangian equation.

$$\frac{d}{dt}(V + \delta V(2\Omega \wedge r)) = -2(1 - \delta V)(\Omega \wedge V) - \nabla \Phi$$  \hspace{1cm} (8)

3.2.2 Continuity equation.
- Conventional formulation.

$$\frac{d}{dt}(\Phi - (1 - \delta_{\text{TR}})\Phi_s) = -(\Phi - \Phi_s)D + \delta_{\text{TR}}V\nabla(\Phi_s)$$ \hspace{1cm} (9)

- Lagrangian formulation.

$$\frac{d}{dt}((\Phi - \Phi_s)J) = 0$$ \hspace{1cm} (10)

$J$ is a “Jacobian” quantity which matches:

$$\frac{dJ}{dt} = -JD$$ \hspace{1cm} (11)
4 The 3D equations in spherical geometry (ARPEGE/IFS).

4.1 Denotations for the 3D equations.

- $V$ is the horizontal wind. Zonal and meridian components on computational grid are $U$ and $V$.
- $D$ is the horizontal wind divergence; $\zeta$ is the horizontal wind vorticity.
- $T$ is the temperature.
- $q$ is the humidity.
- $q_s$ is the rain.
- $\Pi$ is the hydrostatic pressure; $\Pi_s$ is the hydrostatic surface pressure.
- $\Omega$ is the Earth rotation angular velocity.
- $(\lambda_{\text{hse}}, \theta_{\text{hse}})$ are the longitude-latitude coordinates on a tilted and not stretched geometry, the tilting being the same as the one of the computational sphere.
- $(\lambda, \theta)$ are the geographical longitude-latitude coordinates.
- $(\Lambda, \Theta)$ 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.
- $g$ is the geopotential height.
- $\Phi$ is the total geopotential. $\Phi = gz$ in the set of equations described.
- $\Phi_s = g\zeta_s$ is the surface geopotential (i.e. the orography).
- $r$ is the vector directed upwards, the length of which is the Earth radius.
- $a$ is the average Earth radius near the surface.
- $i$ (resp. $j$) is the unit zonal (resp. meridian) vector on the Gaussian grid.
- $k$ is the unit vertical vector. One can write $k = r/a$.
- $g$ 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.
- $R_s$ is a generic denotation used in definition of vertical divergence, which may be $R_d$ or $R$.
- $c_p$ and $c_v$ are respectively the specific heat at constant pressure for moist air and dry air.
- $c_{p,d}$ are respectively the specific heat at constant volume for moist air and dry air.
- $\kappa_m$ is $R/c_p$; $\kappa_d$ is $R_d/c_{p,d}$.
- $\nabla$ is the first order horizontal gradient on $\eta$-surfaces.
- $\alpha_T$ is a vertical-dependent coefficient used to define a thermodynamic variable $T + \delta_T \frac{\alpha_T \Phi_s}{R_s}$; $\alpha_T$ is less sensitive to orography than temperature $T$. Expression of $\alpha_T$ is:

$$\alpha_T = B \left( \frac{R_s}{g} \frac{dT}{dz} \right) T_s \left( \Pi_{st} \Pi_{sst} \right) \left( \frac{\Pi_{sst}}{\Pi_s} \right) \left( \frac{\Pi_{sst}}{\Pi_s} \right)^{-1}$$

(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 $\eta_s$, subsection (6.1)): $B$ varies from 1 to 0 from bottom to top. Subscript “st” stands for “standard atmosphere”.

- $\rho$ 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.
- $\tau, \gamma, \nu$ are linear operators used in the semi-implicit scheme; $S^*, G^*, N^*$ 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.
- $T_s$ is the reference standard atmosphere surface temperature (288.15 K). $\frac{dT}{dz}_{st}$ is the standard atmosphere tropospheric gradient of temperature (-0.0065 K/m).
- $\Pi^*$ is a reference hydrostatic pressure and $\Pi_s^*$ 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 $\eta$ surfaces) constant. $\Delta \Pi^*$ are layer depths corresponding to a surface hydrostatic pressure equal to $\Pi_s^*$.
- $\Pi_{sst}$ is a reference hydrostatic pressure equal to the surface pressure of the standard atmosphere (101325 Pa, variable VP00). $\Pi_s$ is a reference hydrostatic pressure defined at full levels and half levels corresponding to the surface reference hydrostatic pressure $\Pi_{sst}$ (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 $\eta$ surfaces) constant.

8
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 (\( \delta \nu = 0 \)) or implicitly (\( \delta \nu = 1 \)).

\[
\frac{d}{dt} \left( \mathbf{V} + \delta \nu (2\Omega \wedge \mathbf{r}) \right) = -2(1 - \delta \nu)(\mathbf{\Omega} \wedge \mathbf{V}) - \nabla \Phi - RT \nabla (\log \Pi) + \mathbf{F}_V
\]

\( \mathbf{F}_V \) is the physical contribution on horizontal wind.

4.2.2 Thermodynamic equation.

\[
\frac{dT}{dt} + \delta_{TR} \frac{\alpha_T \Phi_s}{\Pi^{1/3}} = \frac{RT}{c_p} \left( \frac{\omega}{\Pi} + F_T \right)
\]

\( F_T \) is the physical contribution on temperature. When \( \delta_{TR} = 1 \) 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 \( \alpha_T \). Term \( \frac{\delta_{TR} \alpha_T \Phi_s}{\Pi^{1/3}} \) only contains advection terms linked to horizontal variations of orography and vertical variations of the coefficient \( \alpha_T \).

4.2.3 Continuity equation.

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

\[
\frac{\partial}{\partial \eta} \int_{\eta=0}^{\eta=1} d\eta \int_{\eta=0}^{\eta=1} \frac{d}{dt} \mathbf{V} \left( \mathbf{\nabla} \mathbf{V} \right) d\eta + \mathbf{V} \nabla \left( \frac{\log \Pi_s + \delta_{TR} \Phi_s}{\Pi^{1/3}} \right) = \frac{1}{\Pi^{1/3}} \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=1} + \frac{1}{\Pi^{1/3}} \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=0} - \frac{1}{\Pi^{1/3}} F_m \left| F_m \right|_{\eta=1}
\]

Variable \( \delta_{TR} \) is 0 or 1; when \( \delta_{TR} = 1 \) 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 \( NDPSFI = 0 \) in \textsc{namphy} ), \( F_m \) is replaced by zero and \( \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=1} \) 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 \( NDPSFI = 1 \) in \textsc{namphy} ), \( F_m \) (diabatic flux) has to be taken in account and \( \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=1} \) is non-zero (more details are given in documentation (IDEUL)).

\( \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=0} \) is non-zero only when there is an upper radiative boundary condition (\textsc{lrubc}=.\textsc{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).

\[
\frac{dq}{dt} = F_q
\]

\( F_q \) is the physical contribution on moisture.

4.2.5 Non advectable pseudo-historic GFL variables:

Equation is written for rain \( q_r \), and is the same for the other non-advectable GFL variables. Since there is no advection, the SLE equation is identical to the Eulerian equation.

\[
\frac{\partial q_r}{\partial t} = F_{qr}
\]
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 a hydrostatic contribution (hydrostatic pressure $\Pi$) and an anhydrostatic departure ($p - \Pi$). $\Pi$ 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 ($\eta$). 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 $\hat{Q} = \log(\Pi)$), 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: $d_4 = d + \chi$

where:

$$\chi = \frac{p}{mRT} \nabla \Phi \left( \frac{\partial V}{\partial \eta} \right)$$

4.3.2 Momentum equation.

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

$$\frac{d(V + \delta V(2\Omega \wedge r))}{dt} = [-2(1 - \delta V)\Omega \wedge V] - \frac{\partial p}{\partial \Pi} \nabla \Phi - RT \frac{\nabla(p)}{p} + F_V$$

(18)

The pressure gradient term is modified compared to the hydrostatic case. One can see that both non-hydrostatic pressure $p$ and hydrostatic pressure $\Pi$ are used in the RHS (right hand side) of equation (18). Term $\nabla(\log p)$ 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:

$$\frac{dT}{dt} = \frac{RT}{c_p} \left( \frac{dp}{p} \right) + F_T$$

(19)

$$\frac{dT}{dt} = -\frac{RT}{c_v} D_3 + \left[ \frac{c_p}{c_v} F_T \right]$$

(20)

Expression of $D_3$ is:

$$D_3 = \nabla V + \frac{p}{mRT} \nabla \Phi \left( \frac{\partial V}{\partial \eta} \right) - \frac{gp}{mRT} \left( \frac{\partial w}{\partial \eta} \right)$$

(21)

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

$$\frac{dT}{dt} = \left( \frac{s_T}{R_a T \frac{m}{mT}} \right) \frac{\alpha_T}{\frac{\partial \Phi}{\partial \eta}}$$

$$\frac{dT}{dt} = \left( \frac{s_T}{R_a T \frac{m}{mT}} \right) \frac{\alpha_T}{\frac{\partial \Phi}{\partial \eta}} - \frac{RT}{c_v} D_3 + \left[ \frac{c_p}{c_v} F_T \right]$$

(22)

See equation (12) for definition of $\alpha_T$.

4.3.4 Vertical velocity $w$ and vertical divergence $d$ equations.

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

$$d = -\frac{gp}{R_a T \frac{m}{mT}} \left( \frac{\partial w}{\partial \eta} \right)$$

(23)

Equation of $d$ is:

$$\frac{dd}{dt} = -dD_3 + d\nabla V - \frac{gp}{R_a T \frac{m}{mT}} \left[ \frac{\partial w}{\partial \eta} \right]_{ad} + \frac{gp}{R_a T \frac{m}{mT}} (\nabla w) \left( \frac{\partial V}{\partial \eta} \right) + F_d$$

(24)
The physical contribution of $d$ is linked to $F_w$ which is the physical contribution of $w$, and also to $F'_m$ (continuity equation for $\text{NDPSFI=1}$) by the following relationship:

$$F_d = -\left[\frac{d}{dt}\right]F'_m - \left[\frac{gp}{R_a T} \frac{\partial F_w}{\partial \eta}\right]$$

One can notice that the RHS of equation (24) contains the term $\frac{\partial}{\partial \eta}\left[\frac{dp}{d\eta}\right]$ which requires the computation of $\frac{dw}{dt}$ (at least its adiabatic part). Equation of $w$ is:

$$\frac{dw}{dt} = g \frac{\partial (p - \Pi)}{\partial \Pi} + F_w$$

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

$$g \frac{dw_{surf}}{dt} = \frac{dV_{surf}}{dt} \nabla \Phi_s + V_{surf} \left[\nabla V_{surf} \nabla(\nabla \Phi_s)\right] + F_{w_{surf}}$$

Value of term $\frac{dV_{surf}}{dt} \nabla \Phi_s + V_{surf} \left[\nabla V_{surf} \nabla(\nabla \Phi_s)\right]$ 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:

$$\frac{dd_4}{dt} = \frac{dd}{dt} + \frac{dX}{dt}$$

where $\frac{d\Phi}{dt}$ is the Lagrangian total derivative of the quantity $(\Phi = -\frac{p}{\Pi} \nabla \Phi)$. The calculation of $\frac{d\Phi}{dt}$ 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:

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

Equation of $\hat{Q}$ is:

$$\frac{d\hat{Q}}{dt} = -\frac{c_p}{c_v} \frac{D_3}{\Pi} - \frac{\omega}{\Pi} + \frac{c_p}{c_vT} F_T$$

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 $\Pi$. 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 $\tilde{T} = T \exp(-\kappa_m \log(p/\Pi))$.
- Pressure gradient term and RHS of $w$ equation have a different formulation and a different discretisation.
- Expression of $\Phi$ has two contributions $\Phi_D$ and $\Phi_S$: $\Phi = \Phi_D + \Phi_S$.
  
  $$\Phi_D = (1 - \kappa_m)(\omega/\Pi + S D), \text{ where } S[D] = \frac{1}{\Pi} \int_0^\Pi \frac{d\Pi}{d\eta} Dd\eta'$$
  
  $$\Phi_S = \frac{\Pi}{\Pi} \nabla \Phi \frac{d\Phi}{d\Pi}, \text{ where } S \text{ is the non-linear counterpart of linear operator } S^*.$$  

- 3D divergence $D_3$ writes $D_3 = D + d + \Phi_S$ or $D_3 = D + d_4 - \Phi_D$.
- The linear system is different and uses a closure equation.

For the vertical divergence equation, it is still possible to use the prognostic variable $d_4 = d + \Phi$ 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 ($\delta V = 0$) or implicitly ($\delta V = 1$).

$$\frac{d(V + \delta V(2\Omega \wedge r))}{dt} = [-2(1 - \delta V)(\Omega \wedge V)] - \exp(\kappa_m \hat{Q}) \left(1 + \Pi \frac{\partial \hat{Q}}{\partial \Pi}\right) \nabla \Phi - RT \exp(\kappa_m \hat{Q}) \left(\frac{\nabla \Pi}{\Pi} + \nabla \hat{Q}\right) + F_V \tag{31}$$

4.4.3 Thermodynamic equation.

$$\frac{d\tilde{T}}{dt} = \frac{RT}{c_p} \frac{\omega}{\Pi} + \exp(-\kappa_m \hat{Q}) F_T \tag{32}$$

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

$$\frac{d \left(\tilde{T} + \delta_T \frac{\alpha_T \hat{Q}}{\Pi} \right)}{dt} = \frac{d \left(\delta_T \frac{\alpha_T \hat{Q}}{\Pi} \right)}{dt} \frac{RT}{c_p} \frac{\omega}{\Pi} + \exp(-\kappa_m \hat{Q}) F_T \tag{33}$$

See equation (12) for definition of $\alpha_T$.

4.4.4 Vertical velocity $w$ and vertical divergence $d$ equations.

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

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

Equation of $d$ is:

$$\frac{dd}{dt} = -d(d + X_d) - \frac{g \Pi}{RT \frac{\partial \Pi}{\partial \eta}} \left(\frac{\partial \tilde{T}}{\partial \eta}\right) + \frac{g \Pi}{RT \frac{\partial \Pi}{\partial \eta}} (\nabla w) \left(\frac{\partial V}{\partial \eta}\right) + F_d \tag{35}$$

Equation of $w$ is:

$$\frac{dw}{dt} = g \frac{\kappa_m}{\Pi} \left(\frac{\partial \Pi(\exp(\kappa_m \hat{Q}) - 1)}{\partial \Pi} + (\kappa_m - 1)(\exp(\kappa_m \hat{Q}) - 1)\right) + F_w \tag{36}$$

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:

$$\frac{dd_4}{dt} = \frac{dd}{dt} + dX \tag{37}$$

where $\frac{dX}{dt}$ is the Lagrangian total derivative of the quantity $X$. The calculation of $\frac{dX}{dt}$ 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 $\Pi$. 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. $\Delta t$ 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 $A - \beta L$, $V/a$ or $\eta$), 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 $F$), $X$ writes:

$$\tilde{X} = [0.5X^P_F + 0.5X^O_G] + C_\gamma[0.5X^O_G - 0.5X^O_O] + C_\gamma(0.25 - C_\alpha)[(X^C_N - X^F_P) - (X^O_G - X^O_O)]$$

- $C_\gamma=0$ yields the non-extrapolated way to compute $\tilde{X}$ (NESC).
- $C_\gamma=1$, $C_\alpha=0.25$ yields the stable extrapolation (SETTLS).
- $0 < C_\gamma < 1$, $C_\alpha=0.25$ yields the mixed extrapolation (MIXETTLS).
- $C_\gamma=1$, $C_\alpha=0$ 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 $[A^O - \beta \mathcal{L}^O]_O - 0.5[A^- - \beta \mathcal{L}^-]_O$ by $0.5[A^O - \beta \mathcal{L}^O]_O$. 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 $A$ and $\beta L$ evaluated at origin point $O$ by $(1 - \epsilon)$.
  - multiply $A$ and $\beta L$ evaluated at final point $F$ by $(1 + \epsilon)$.
  - no change for $X$ and term containing $F$.

- Pseudo-second order uncentering: it is applied only to linear terms containing $\beta L$. 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 $\beta$ by $(1 + \epsilon_X)\beta$.

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:

- $NW^{LAG}$ for momentum equation.
- $NT^{LAG}$ for temperature equation.
- $NSP^{DLAG}$ for pressure departure variable (NHEE model only).
- $NSV^{DLAG}$ for vertical divergence variable (non-hydrostatic models only).

Generic notation $P(X)^{L0}$, $P(X)^{L9}$, $P(X)^{T1}$ stands for:

- $PUL^{L0}$, $PUL^{L9}$, $PUT^{T1}$ for U-momentum equation.
- $PVL^{L0}$, $PVL^{L9}$, $PVT^{T1}$ for V-momentum equation.
- $PT^{L0}$, $PTL^{L9}$, $PTT^{T1}$ for temperature equation.
- $PSP^{DL0}$, $PSP^{DL9}$, $PSP^{DT1}$ for $\hat{Q}$ equation.
- $PSV^{DL0}$, $PSV^{DL9}$, $PSV^{DT1}$ for $d$ equation.

Generic notation $P(X)^{NLT9}$ stands for:

- $PUN^{LT9}$ for U-momentum equation.
- $PV^{NLT9}$ for V-momentum equation.
- $PT^{NLT9}$ for temperature equation.
- $PSP^{DNLT9}$ for $\hat{Q}$ equation.
- $PSV^{DNLT9}$ for $d$ equation.

Generic notation for total term, linear term, non linear term, physics: $A$ is the total term (sum of dynamical contributions), $L$ is the linear term (treated in the semi-implicit scheme), the difference $A - \beta L$ is the non-linear term. $F$ 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 $LSLINL_{C2}$). 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 \( P(X)\) is used instead of \( P(X)\) 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
\( P(X)\) is used instead of \( P(X)\) 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 \( \Delta tF \) 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 \( R \) has to be applied from interpolation point to final point:

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

5.4.2 3TL vertical interpolating SL scheme.

Equation

\[
\frac{dX}{dt} = A + F
\]

is discretised as follows:

\[
(X - (1 + \epsilon)\Delta t\beta L)_F = \{X - + (1 - \epsilon)\Delta t[A - \beta L]^o + [(1 - \epsilon)\Delta t\beta L + 2\Delta tF]^o\}_O + \{(1 + \epsilon)\Delta t[A - \beta L]^o\}_F
\]

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

- \( P(X)L0 \): \( (1 - \epsilon)\Delta t[A - \beta L]^o + [(1 - \epsilon)\Delta t\beta L]^o \) for trilinear interpolation at \( O \).
- \( P(X)L9 \): \( X^o + [2\Delta tF]^o \) for high-order interpolation at \( O \).
- \( P(X)T1 \): \( (1 + \epsilon)\Delta t[A - \beta L]^o \) then provisional add of quantity \( [(1 + \epsilon)\Delta t\beta L]^o \) before \( t+d\Delta t \) 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 \( A^m - (1 + \epsilon\chi)\beta L^m \) 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 \( A^m - (1 + \epsilon\chi)\beta L^m \) has to be saved in a buffer \( P(X)\) to be available as \( A^m - (1 + \epsilon\chi)\beta L^m \)
for the following timestep.

Equation (38) is discretised as follows:

\[
(X - (1 + \epsilon\chi)\frac{\Delta t}{2}\beta L)^O = \{X^o + \frac{\Delta t}{2}[A - (1 + \epsilon\chi)\beta L]^m + [(1 + \epsilon\chi)\frac{\Delta t}{2}\beta L + \Delta tF]^m\}_O + \{(1 + \epsilon\chi)\frac{\Delta t}{2}[A - (1 + \epsilon\chi)\beta L]^m\}_F
\]

which can be rewritten, once expanded the extrapolation:

\[
(X - (1 + \epsilon\chi)\frac{\Delta t}{2}\beta L)_F = \{X^o + \frac{\Delta t}{2}[A^o + 0.5\Delta tF]^o\}_O + 0.5\frac{\Delta t}{2}[\{A - (1 + \epsilon\chi)\beta L]^o - (A - (1 + \epsilon\chi)\beta L)^o\}_O + \frac{\Delta t}{2}[1.5\{A - (1 + \epsilon\chi)\beta L]^o - 0.5\{A - (1 + \epsilon\chi)\beta L]^o\}_F
\]

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

- \( P(X)L0 \): \( \frac{\Delta t}{2}[A^o + 0.5\Delta tF]^o \) for high-order interpolation at \( O \).
- \( P(X)L9 \): \( X^o + [\Delta tF]^o \) for high-order interpolation at \( O \).
- \( P(X)T1 \): \( \frac{\Delta t}{2}[1.5\{A - (1 + \epsilon\chi)\beta L]^o - 0.5\{A - (1 + \epsilon\chi)\beta L]^o\}_F \) then provisional add of quantity \( [(1 + \epsilon\chi)\frac{\Delta t}{2}\beta L]^o \) before \( t+d\Delta t \) 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 $A^o = (1 + \epsilon_x)\beta L^o$ has to be saved in a buffer $P(X)\text{NLT9}$ to be available as $A^- = (1 + \epsilon_x)\beta L^-$ for the following timestep. Equation (38) is discretised as follows:

\[
(X - (1 + \epsilon_x)\frac{\partial}{\partial x} \beta L)^o_t = [X^o + \frac{\partial}{\partial x} A^o + \Delta t F]^o_O + C, \frac{\partial}{\partial x}[(A - (1 + \epsilon_x)\beta L)^o - (A - (1 + \epsilon_x)\beta L)^-]_O + [\frac{\partial}{\partial x}(A - (1 + \epsilon_x)\beta L)^o]_F
\]  

Buffers content before interpolations for $\text{N}(X)\text{LAG}=3$:

- $P(X)\text{L0}$: $\frac{\partial}{\partial x} A^o + C, \frac{\partial}{\partial x}[(A - (1 + \epsilon_x)\beta L)^o - (A - (1 + \epsilon_x)\beta L)^-]$ for trilinear interpolation at $O$.
- $P(X)\text{L9}$: $X^o + [\Delta t F]^o$ for high-order interpolation at $O$.
- $P(X)\text{T1}$: $\frac{\partial}{\partial x}(A - (1 + \epsilon_x)\beta L)^o$ then proportional add of quantity $[(1 + \epsilon_x)\frac{\partial}{\partial x} \beta L]^o$ 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=$\cdot$,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 ($\text{LVFE}, \text{GW}=\text{F}$) $w$ 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 ($\text{LVFE}, \text{GW}=\text{T}$) $w$ is given at full levels. There is a specific treatment for $w_{\text{surf}}$ (diagnostic condition at $t+\Delta t$).
  - Calculation of the linear terms of the $d$ equation for the semi-simple implicit scheme.
  - Conversion from $w^+$ into $d^+$ or $d_4^+$. If $\text{LVFE}, \text{GW}=\text{T}$, 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 $\frac{\partial}{\partial x}$, and this is not done by a SL treatment of the equation of $\chi$ (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-level semi-Lagrangian scheme for $\text{ND4SYS}=1$: $d\chi/dt = [0.5\chi^o_F + 0.5\chi^o_L - \chi^o_{\text{surf}}]/[\Delta t]$.
  - Two-time level semi-Lagrangian scheme for $\text{ND4SYS}=1$: $d\chi/dt = [\chi^o_N - \chi^o_S]/0.5[\Delta t]$.

$\chi_M$ is a symbolic notation for the extrapolated $t + 0.5\Delta t$ value of $\chi$. The way of extrapolating depends on $\text{LSETTLS}$ and $\text{LINESC}$ and is the same as for the other terms evaluated at $t + 0.5\Delta t$ (there is no interpolation at $M$).
  - Three-level semi-Lagrangian scheme for $\text{ND4SYS}=2$: $d\chi/dt = [\chi^o_F - \chi^o_L]/0.5[\Delta t]$.
  - Two-time level semi-Lagrangian scheme for $\text{ND4SYS}=2$: $d\chi/dt = [\chi^o_F - \chi^o_N]/0.5[\Delta t]$.

More details are given in appendix 1.

* Assumptions at the surface (velocities). Calculation of $d\chi/dt$ for the layer $l = L$ requires the calculation of $d\chi_{\text{surf}}/dt$. There are two options to compute $d\chi_{\text{surf}}/dt$, controlled by variable $\text{LRDBBC}$.
  - For $\text{LRDBBC}=\cdot$,F, one simply discretises equation (27) with a SL treatment. The RHS of this equation contains $d\chi_{\text{surf}}/dt$ and $\chi_{\text{surf}}$, and the assumptions currently done about these quantities are:
    - $\chi_{\text{surf}} = V_{[\text{L}]}$.
    - $d\chi_{\text{surf}}/dt = [d\chi/dt]_{l=L, \text{adiab}}$, with an explicit treatment of the Coriolis term, even if $\text{LADV}=\cdot$. or $\text{LIMP}=\cdot$. .
  - An alternate discretisation (option $\text{LRDBBC}=\cdot$,T.) is to evaluate $d\chi_{\text{surf}}/dt$ by:

16
\[ \frac{\partial w}{\partial t} = (w^+ - w^-)/(2\Delta t) \]

which requires additional interpolations to compute \( w_O \).

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

\[ w^+ = V_{1=L, prov} \nabla \Phi \]

where \( V_{1=L, prov} \) is the provisional \( t + \Delta t \) value of \( V_{1=L} \) computed just after the interpolations (this is this provisional value which is used as input for the lagged physics).

The treatment of \( \frac{\partial}{\partial t} \) for the layer \( l = L \) is done according to the following steps:

1. Calculation of \( \left[ \frac{\partial}{\partial t} \right]_{l=L, lrdbbc=F} \) (as if LRDBBC were .F.), \( \left[ \frac{\partial w_{surf}}{\partial t} \right]_{lrdbbc=F} \) and \( \left[ \frac{\partial w_{surf}}{\partial t} \right]_{lrdbbc=T} \).
2. Calculation of \( \left[ \frac{\partial}{\partial t} \right]_{l=L, lrdbbc=T} \) by the following formula:

\[
\left[ \frac{\partial}{\partial t} \right]_{l=L, lrdbbc=T} = \left[ \frac{\partial}{\partial t} \right]_{l=L, lrdbbc=F} - \left[ \left( \frac{\partial w_{surf}}{\partial t} \right)_{lrdbbc=T} - \left( \frac{\partial w_{surf}}{\partial t} \right)_{lrdbbc=F} \right] \left[ \frac{gp}{R_\alpha T \Delta \Pi} \right]_{l=L, MM}^{o}
\]

where \( MM \) is the decentered middle of \( O \) and \( F \) (that requires a specific interpolation of \( \left[ \frac{\partial w_{surf}}{\partial t} \right]_{l=L} \) at \( O \)).

For more details about these calculations and the additional interpolations required, see documentation (IDVNHH3), 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 (\( \Delta x < 1 \)km) 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 \( \mathbf{P(X)L9} \), \( \mathbf{P(X)T1} \) stands for:

- \( \mathbf{PGFLL9} \), \( \mathbf{PGFLT1} \) for GFL variables.

Generic notation for total term, linear term, non linear term, physics: \( A \) is the total term (sum of dynamical contributions), \( L \) is the linear term (treated in the semi-implicit scheme), the difference \( A - \beta L \) is the non-linear term. \( F \) is the sum of contributions computed in the physical parameterizations. In the present case \( A \) and \( L \) 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 \( \mathbf{P(X)LF9} \) is used instead of \( \mathbf{P(X)L9} \) 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 LTDIABL1N 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:

$$X^+ = \{X^- + [2\Delta t F]^-\}_O$$ (44)

Buffers content before interpolations:

- $P(X)T1$ 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:

$$X^+ = \{X^o + [\Delta t F]^o\}_O$$ (45)

Buffers content before interpolations:

- $P(X)L9$: $X^o + [\Delta t F]^o$ 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):

$$X^+_F = \{X^o + [\Delta t F]^o\}_F$$ (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:

$$[R_{\text{inte}}]_{(\text{top, surf})} \left\langle \frac{W_{\text{vel}}}{\Delta \eta} \frac{dX}{dt} \right\rangle = [R_{\text{inte}}]_{(\text{top, surf})} \left\langle \frac{W_{\text{vel}}}{\Delta \eta} \cdot \Delta \eta \right\rangle + [R_{\text{inte}}]_{(\text{top, surf})} \left\langle \frac{W_{\text{vel}}}{\Delta \eta} F \right\rangle$$ (47)

where:

$$[R_{\text{inte}}]_{(\text{top, surf})} \left\langle \frac{W_{\text{vel}}}{\Delta \eta} \right\rangle = 1$$ (48)

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

In the set of equations described in this documentation, expression of $W_{\text{vel}}$ at full levels is:

$$[W_{\text{vel}}]_B = \Delta B_i$$ (49)
When the finite element vertical discretisation is activated ($\text{LVERTFE} = .T.$), \([R_{\text{inte}}]_{\text{(top, surf)}}\) is a vector tricky to compute, it is precomputed in the setup code under \text{SUVERTFE} and stored in the array \text{RINTE} or \text{RINTBF11}; vertical integrations are done in routine \text{VERINT}.

When the finite difference vertical discretisation is activated ($\text{LVERTFE} = .F.$), \([R_{\text{inte}}]_{\text{(top, surf)}}\) is simply the vector of coordinates \([\Delta \eta_1; \Delta \eta_2; \ldots; \Delta \eta_i; \ldots; \Delta \eta_L]\) so, equation (47) rewrites:

$$\sum_{l=1}^{L} [W_{\text{nov}}]_l \left( \frac{dX}{dt} \right)_l = \sum_{l=1}^{L} [W_{\text{nov}}]_l A_t + \sum_{l=1}^{L} [W_{\text{nov}}]_l F_t$$  \hspace{1cm} (50)

List of equations:
- Continuity equation.

Generic notation $N(X)\text{LAG}$ 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:
- $\text{PX9}$, $\text{PSPT1}$, $\text{PCL9}$ for continuity equation.

Generic notation $P(X)\text{NLT9}$ (3D term) stands for:
- $\text{PSPNLT9}$ for continuity equation.

Generic notation for total term, linear term, non linear term, physics:
- $A$ is the total term (sum of dynamical contributions): it is assumed to be a 3D term (sum of 3D and 2D contributions).
- $L$ 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 $A - \beta L$ is the non-linear term, considered as a 3D term.
- $F$ 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 $N(X)\text{LAG}=2$ is the same one as for $N(X)\text{LAG}=3$ but trilinear interpolations are replaced by high-order ones. Only $N(X)\text{LAG}=3$ 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, $[R_{\text{inte}}]_{\text{(top, surf)}} \left( \left[ \frac{W_{\text{nov}}}{\Delta \eta} \right]_P [\text{surface quantity}]_O \right)$ has no reason to be equal to $[\text{surface quantity}]_{O(y=1)}$, these quantities are generally different: this is $[R_{\text{inte}}]_{\text{(top, surf)}} \left( \left[ \frac{W_{\text{nov}}}{\Delta \eta} \right]_P [\text{surface quantity}]_O \right)$ which has to be computed. $\left( \left[ \frac{W_{\text{nov}}}{\Delta \eta} \right]_P [\text{surface quantity}]_O \right)$ is the vector containing $\left[ \frac{W_{\text{nov}}}{\Delta \eta} \right]_P [\text{surface quantity}]_{O(l)}$, $l = 1$ to $L$. For $\text{LVERTFE} = .F.$ this is $\sum_{l=1}^{L} [W_{\text{nov}}]_P [\text{surface quantity}]_{O(l)}$.

5.6.2 3TL vertical interpolating SL scheme.

Equation (47) is discretised as follows:

$$X - (1 + \epsilon) \Delta t \beta L \right) + \left[ R_{\text{inte}} \right]_{\text{(top, surf)}} \left( \left[ \frac{W_{\text{nov}}}{\Delta \eta} \right]_P \left[ X - (1 - \epsilon) \Delta t \beta L \right] + (1 - \epsilon) \Delta t \beta L + 2 \Delta t \beta F \right)_O \right)$$  \hspace{1cm} (51)

Buffers content before interpolations for $N(X)\text{LAG}=3$:
- $P(X3D)L9$: $\left( 1 - \epsilon \Delta t \beta L \right)^n + \left( 1 - \epsilon \Delta t \beta L \right)^n$ for trilinear interpolation at $O(l)$.
- $P(X2D)9$: $\left[ X - 2 \Delta t F \right]^n$ for horizontal high-order interpolation at $O(l)$.
- $P(X)T1$: $(1 + \epsilon) \Delta t \beta L^n$ then provisional add of quantity \([(1 + \epsilon) \Delta t \beta L^n]\) before t+dt physics; evaluated at the final point $F$. 

19
5.6.3 2TL vertical interpolating SL scheme: conventional extrapolation.

The \( t + 0.5\Delta t \) non-linear term \( \mathcal{A}^n - (1 + \epsilon_X)\beta \mathcal{L}^n \) 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 (McDonald and Haugen, 1992). Quantity \( \mathcal{A}^n - (1 + \epsilon_X)\beta \mathcal{L}^n \) has to be saved in a buffer \( P(X)\text{NLT}9 \) to be available as \( \mathcal{A}^- - (1 + \epsilon_X)\beta \mathcal{L}^- \) for the following timestep.

Equation (47) is discretised as follows:

\[
(X - (1 + \epsilon_X) \Delta t \beta \mathcal{L})^a_p = \{ \Delta t \mathcal{A} - (1 + \epsilon_X)\beta \mathcal{L}^n \}^a_F + [R_{\text{inte}}(\text{top, surf})] \left( \left[ \frac{\text{Num}}{\Delta n} \right]_F \{ X^n + \Delta t A^n + \Delta t F^n \} \right)_O \tag{52}
\]

which can be rewritten, once expanded the extrapolation:

\[
(X - (1 + \epsilon_X) \Delta t \beta \mathcal{L})^a_p = [R_{\text{inte}}(\text{top, surf})] \left( \left[ \frac{\text{Num}}{\Delta n} \right]_F \{ X^n + \Delta t A^n + \Delta t F^n \} \right)_O + 0.5[R_{\text{inte}}(\text{top, surf})] \left( \left[ \frac{\text{Num}}{\Delta n} \right]_F \{ X^n + \Delta t A^n + \Delta t F^n \} \right)_O \tag{53}
\]

Buffers content before interpolations for \( N(X)\text{LAG}=3: \)

- \( P(X3D)\text{L9} \) : \( \Delta t \mathcal{A} - (1 + \epsilon_X)\beta \mathcal{L}^n \) for trilinear interpolation at \( O(l) \).
- \( P(X2D)9 \) : \( X^n + \Delta t F^n \) for horizontal high-order interpolation at \( O(l) \).
- \( P(X)\text{T}1 \) : \( 0.5\Delta t \mathcal{A} - (1 + \epsilon_X)\beta \mathcal{L}^- \) then provisional add of quantity \( (1 + \epsilon_X) \Delta t \mathcal{A}^n \) before \( t + \Delta t \) 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 \( \mathcal{A}^n - (1 + \epsilon_X)\beta \mathcal{L}^n \) has to be saved in a buffer \( P(X)\text{NLT}9 \) to be available as \( \mathcal{A}^- - (1 + \epsilon_X)\beta \mathcal{L}^- \) for the following timestep.

Equation (47) is discretised as follows:

\[
(X - (1 + \epsilon_X) \Delta t \beta \mathcal{L})^a_p = \{ \Delta t \mathcal{A} - (1 + \epsilon_X)\beta \mathcal{L}^n \}^a_F + [\text{R}_{\text{inte}}(\text{top, surf})] \left( \left[ \frac{\text{Num}}{\Delta n} \right]_F \{ X^n + \Delta t A^n + \Delta t F^n \} \right)_O \tag{54}
\]

Buffers content before interpolations for \( N(X)\text{LAG}=3: \)

- \( P(X3D)\text{L9} \) : \( \Delta t \mathcal{A} + C_\gamma \Delta t \mathcal{A} \) \( \{ A - (1 + \epsilon_X)\beta \mathcal{L}^n \} - \{ A - (1 + \epsilon_X)\beta \mathcal{L}^- \} \) for trilinear interpolation at \( O(l) \).
- \( P(X2D)9 \) : \( X^n + \Delta t F^n \) for horizontal high-order interpolation at \( O(l) \).
- \( P(X)\text{T}1 \) : \( \Delta t \mathcal{A} - (1 + \epsilon_X)\beta \mathcal{L}^- \) then provisional add of quantity \( (1 + \epsilon_X) \Delta t \mathcal{A}^n \) before \( t + \Delta t \) 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 \( \text{N}(X)\text{LAG} \) stands for:

- \( \text{NWLAG} \) for momentum equation.
- \( \text{NVLAG} \) for continuity equation. Negative values of \( \text{NVLAG} \) are used for Lagrangian formulation of continuity equation (the following discretisations apply to the absolute value of \( \text{NVLAG} \)), only \( \text{NVLAG}=-2 \) 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: $A$ is the total term (sum of dynamical contributions), $L$ is the linear term (treated in the semi-implicit scheme), the difference $A - \beta L$ is the non-linear term.

There is no diabatic term: $F$ 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 $R$ has to be applied from interpolation point to final point:

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

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

5.8 Additional remarks.

* Additional vertical derivatives: if $\delta_{TR}$ is non-zero, discretisation of temperature equation needs to compute the vertical advection $(\frac{\partial}{\partial \eta} \frac{\partial}{\partial \alpha} T)$ (at full levels) of $\alpha_T$. Layers values of $\alpha_T$ (array RCORDIF) are used to define $T + \delta_{TR} \frac{\partial}{\partial \eta} \frac{\partial}{\partial \alpha} \Phi_s$, but half level values of $\alpha_T$ (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 $\eta$) 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 $\eta$) 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 remarks:
- 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, for cases \texttt{LSLDP\_RK=F and LSLDP\_CURV=F}.
- The subsection (6.4) gives informations about the other cases (for example plane geometry and half level variables).
- Options \texttt{LSLDP\_RK=T and LSLDP\_CURV=T} are not extensively described for the time being; they are available for spherical unstretched untilted geometry.

6.1 Medium point \( M \) and origin point \( O \) for \texttt{LSLDP\_RK=F and LSLDP\_CURV=F}.

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 \( \Omega/a \). 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 \texttt{LPC\_FULL}.

* Denotations:
  - \( R^{MF} \): the rotation operator from medium point to final point (see section 11).
  - \( R^{OF} \): 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).
  - \( \mathbf{r} = \mathbf{ok} \).
  - \( \phi^{MF} \): angle (\( \mathbf{CM},\mathbf{CF} \)).
  - \( \theta^M,\lambda^M \): 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.
  - \( \Delta t \): time-step.
  - \( \delta t \): 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 \( \eta \) levels (\( \Pi = A + B\Pi_s \)).
  - \( \Pi_{sat} \) is the surface pressure of the standard atmosphere (variable \texttt{VP00}).
  - \( \Pi_{at} \) is a reference hydrostatic pressure defined at full levels and half levels corresponding to the surface reference hydrostatic pressure \( \Pi_{sat} \) (\( \Pi_{at} = A + B\Pi_{sat} \)); stored in array \texttt{STPRE}.

* Definition of the vertical coordinate \( \eta \): Research of medium point needs an exact definition of the vertical coordinate \( \eta \). For the half level number \( l \) (\( l \) between 0 and \( L \)), \( \eta_l \) is defined by:
  - Regular spacing (\texttt{LREGETA=T}):
    \( \eta_l = l/L \).
  - Irregular spacing (\texttt{LREGETA=F}):
    \( \eta_l = A_l/\Pi_{sat} + B_T \).

For the layer number \( l \) (\( l \) between 1 and \( L \)), \( \eta_l \) is defined by:

\( \eta_l = 0.5(\eta_{l-1} + \eta_T) \).

A specific definition of \( \eta \) may be required for the VFE operators if \texttt{LVERTFE=T}.: it is controlled by the key \texttt{LVFE\_REGETA}.

22
6.1.1 Conventional algorithm.

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

\[
[r/a]_{k+1}^M = [r/a]^F \cos \phi_k - \frac{R^{MF}([V/a]^M_k)}{|[V/a]^M_k|} \sin \phi_k
\]

where:

\[
\phi_k = \delta t |[V/a]^M_k|
\]

\( \phi \) is a small angle:

\[
\sin \phi \simeq \phi - (1/6)\phi^3
\]

and:

\[
\cos \phi \simeq 1 - 0.5\phi^2
\]

This approximation allows to simplify some calculations. Of course:

\[
\sin \phi \simeq \phi(1-(1/6)\phi^2) \simeq |[V/a]| \delta t(1-(1/6)\phi^2)
\]

thus:

\[
[r/a]_{k+1}^M = [r/a]^F (1 - 0.5\phi^2_k) - R^{MF}([V/a]^M_k)\delta t(1-(1/6)\phi^2_k)
\]

On the vertical, for 3D model:

\[
\eta_{k+1}^M = \eta^F - \delta t \eta_k^M
\]

* First iteration: Let us start with \( M_0 = F \), \( [V/a]_0^M = [V/a]^F \), \( \phi_0 = \delta t |[V/a]| \), \( \eta_0^M = \eta^F \). Horizontal wind \( V \) has components \((u',v')\). Thus

\[
\sin \theta_1^M = \sin \theta^F \cos \phi_0 - \frac{|u'/a|^F}{|[V/a]^F|} \cos \theta^F \sin \phi_0
\]

\[
\cos \theta_1^M \cos(\lambda_1^M - \lambda^F) = \cos \theta^F \cos \phi_0 + \frac{|u'/a|^F}{|[V/a]^F|} \sin \theta^F \sin \phi_0
\]

\[
\cos \theta_1^M \sin(\lambda_1^M - \lambda^F) = -\frac{|u'/a|^F}{|[V/a]^F|} \sin \phi_0
\]

\[
\eta_1^M = \eta^F - \delta t \eta_0^M
\]

This defines the coordinates of \( M_1 \). Then \([u/a], [v/a], \eta\) are interpolated at this point, that gives \([V/a]_1^M\) and \( \eta_1^M \). 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 \( V'(u', v') = R^{MF}(V_k^M) \).

\[
\sin \theta_{k+1}^M = \sin \theta^F \cos \phi_k - \frac{|u'/a|}{|[V/a]|} \cos \theta^F \sin \phi_k
\]

\[
\cos \theta_{k+1}^M \cos(\lambda_{k+1}^M - \lambda^F) = \cos \theta^F \cos \phi_k + \frac{|u'/a|}{|[V/a]|} \sin \theta^F \sin \phi_k
\]

\[
\cos \theta_{k+1}^M \sin(\lambda_{k+1}^M - \lambda^F) = -\frac{|u'/a|}{|[V/a]|} \sin \phi_k
\]

\[
\eta_{k+1}^M = \eta^F - \delta t \eta_k^M
\]

This defines coordinates of \( M_{k+1} \). Then, if it is not the last iteration, \([u/a], [v/a], \eta\) are interpolated at this point, that gives \([V/a]_{k+1}^M\) and \( \eta_{k+1}^M \). 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 \( \delta t \) by \( 2\delta t \) and superscript \( M \) by \( O \) in the above formulae).

- Extrapolation of the wind for SL2TL scheme: the quantity \([V/a]_t\) 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 \([V/a]t\) at \(t + 0.5\Delta t\) is now replaced by a stable (SETTLS) extrapolation.

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

The same type of extrapolation is done for the \(\eta\)-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 \(k + 1\):

\[
[r/a]_{k+1}^O = \left[\frac{r}{a}\right]^F \cos \phi_k - 0.5[V/a]^F(t) + 0.5R_{OF}(2[V/a]^O(t) - [V/a]^O_k(t - \Delta t)) \frac{\sin \phi_k}{0.5[V/a]^F(t) + 0.5R_{OF}(2[V/a]^O(t) - [V/a]^O_k(t - \Delta t))}
\]

where:

\[
\phi_k = 2\delta t \left| 0.5[V/a]^F(t) + 0.5R_{OF}(2[V/a]^O(t) - [V/a]^O_k(t - \Delta t)) \right|
\]

Approximations given by equations (57), (58) and (59) are still valid (change \(\delta t\) by \(2\delta t\) and the superscript \(M\) by \(O\)). This defines the coordinates of \(O_{k+1}\) by the following iterative scheme: for the iteration \(k + 1\):

**First iteration:** One starts with \(M_0 = F, [V/a]_1^F(t)\) as a first guess for the spatio-temporally extrapolated horizontal angular velocity, \(\phi_0 = 2\delta t \left| [V/a]^F(t) \right|, \eta^F(t)\) as a first guess for the spatio-temporally extrapolated \(\eta\)-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 \(\delta t\) by \(2\delta t\) and the superscript \(M\) by \(O\). This defines the coordinates of \(O_{1}\) by the following iterative scheme: for the iteration \(k + 1\):

**Following iterations:** \([V^F]/(a^F)\) (of coordinates \([u^F]/a^F, [v^F]/a^F\)) is a generic notation for \((0.5[V/a]^F(t) + 0.5R_{OF}(2[V/a]^O(t) - [V/a]^O_k(t - \Delta t)))\). For horizontal displacement use equations (66) to (68) replacing \(\delta t\) by \(2\delta t\) and the superscript \(M\) by \(O\). For vertical displacement use equation

\[
\eta_{k+1}^O = \eta^F - 2\delta t(0.5\eta^F(t) + 0.5(2\eta_k^O(t) - \eta^O(t - \Delta t)))
\]

6.1.3 “LETRA” alternate stable algorithm for SL2TL scheme.

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

\[
[V/a]_{ext} = [V/a](t) + 0.5\Delta tRHSAV
\]

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

\[
\eta_{ext} = \eta(t)
\]

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 “\(\delta t\)” by “\(\Delta F\)”.

**First iteration:** Let us start with \(O_0 = F, [V/a]_0^O = [V/a]_{ext}^O, \phi_0 = \Delta t \left| [V/a]_0^F \right|, \eta_0^O = \eta_{ext}^O. Horizontal angular velocity \([V/a]_{ext}\) has components \((u/a, v/a)\). Use equations (62) to (65), replacing “\(M\)” by “\(O\)” and “\(\delta t\)” by “\(\Delta F\)” to define coordinates of \(O_1\). Then \(u/a, v/a, \eta\) are interpolated at this point, that gives \([V/a]_1^O\) and \(\eta_1^O\) (subscript “ext” is omitted). Tri-linear 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 \( u/a, v/a, η \) are interpolated at this point, that gives \( [V/a]_{k+1}^O \) and \( η_{k+1}^O \) (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 ARPECE/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 \( t − Δt \)).

* Extrapolation of the wind if non-extrapolating option: No extrapolation is done. The first iteration of the iterative centred-implicit scheme uses \( [V/a](t) \) and \( η(t) \). The following iterations of the iterative centred-implicit scheme use \( [V/a](t + Δt) \) and \( η(t + Δt) \) 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 \( 1.5[V/a](t) − 0.5[V/a](t − Δt) \) and \( 1.5η(t) − 0.5η(t − Δt) \).

Algorithm: One denotes by index “\( k \)” the numbering of the SL-trajectory research algorithm, 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:

\[ [r/a]_{k+1}^O(i) = [r/a]^F \cos \phi_k(i) − \frac{0.5R^{OP}[V/a]_{k}^O(t) + 0.5[V/a]^F(t + Δt)]_{(i−1)}}{0.5R^{OP}[V/a]_{k}^O(t) + 0.5[V/a]^F(t + Δt)]_{(i−1)}} \sin \phi_k(i) \]

(77)

where:

\[ [\phi_k(i)] = 2\delta t | 0.5R^{OP}[V/a]_{k}^O(t) + 0.5[V/a]^F(t + Δt)]_{(i−1)} \]

(78)

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

\[ [r/a]_{k+1}^O(i) = [r/a]^F(1 − 0.5\phi_k(i)) − \frac{0.5R^{OP}[V/a]_{k}^O(t) + 0.5[V/a]^F(t + Δt)]_{(i−1)}}{2(2\delta t)(1 − (1/6)\phi_k(i))} \]

(79)

On the vertical, for 3D model:

\[ [ρ]_{k+1}^O(l) = \rho^F − 2\delta t \left(0.5η_k^O(t) + 0.5[\dot{η}^F(t + Δt)]_{(i−1)}\right) \]

(80)

For \( i = 0 \):

\[ [V/a](t + Δt)]_{(i=0)} = [V/a](t) \]

and \( [η(t + Δt)]_{(i=0)} = η(t) \) for non-extrapolating option;

\[ [V/a](t + Δt)]_{(i=0)} = 2[V/a](t) − [V/a](t − Δt) \]

and \( [η(t + Δt)]_{(i=0)} = 2η(t) − η(t − Δt) \) for “SETTLS”-stable extrapolating option.

* First iteration of the research of SL trajectory:

One starts with \( M_0 = F \), \([V/a]^F(t + Δt)]_{(i=1)}\) as a first guess for the spatio-temporally extrapolated horizontal wind, \([ϕ_k(i)] = 2\delta t | [V/a]^F(t + Δt)]_{(i−1)} \), \([η^F(t + Δt)]_{(i−1)}\) 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 \( O(\eta_k(i)); [V/a](t) \) and \( η(t) \) 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: \([V/a]\) (of coordinates \([u^\prime/a], [v^\prime/a]\) is a generic notation for \( 0.5R^{OP}[V/a]_{k}^O(t) + 0.5[V/a]^F(t + Δt)]_{(i−1)} \). 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 (66) to (68) replacing \( δt \) by \( 2\delta t \) and the superscript \( M \) by \( O \) otherwise.

\[ [η_k+1(i)] = η^F − 2\delta t \left(0.5η_k^O(t) + 0.5[\dot{η}^F(t + Δt)]_{(i−1)}\right) \]

(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 \( (|r/a|^O, |r/a|^F) \) is small (less than 10°, what is generally matched), one can write for horizontal displacement:

\[
|r/a|^O - |r/a|^F \simeq 2(|r/a|^M - |r/a|^F)
\]  (82)

For vertical displacement on can always write:

\[
\eta^O - \eta^F = 2(\eta^M - \eta^F)
\]  (83)

One denotes by:

- \( \phi = (|r/a|^M, |r/a|^F) \)

- \([V'/a]\) (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:

\[
\sin \theta^O = \sin \theta^F \cos 2\phi - 2\cos \phi \left\{ \frac{|v'/a|}{|V'/a|} \cos \theta^F \sin \phi \right\} \tag{86}
\]
\[
\cos \theta^O \cos(\lambda^O - \lambda^F) = \cos \theta^F \cos 2\phi + 2\cos \phi \left\{ \frac{|v'/a|}{|V'/a|} \sin \theta^F \sin \phi \right\} \tag{87}
\]
\[
\cos \theta^O \sin(\lambda^O - \lambda^F) = -2\cos \phi \left\{ \frac{|u'/a|}{|V'/a|} \sin \phi \right\} \tag{88}
\]

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 \((2\Omega \wedge ak)\) is computed (always analytically) at this improved position of \( O \) (so L2TLFF is active only if \( \text{LADVF}\neq \text{T} \)). 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 \( \text{V}^F(t + \Delta t) \), then \( 0.5(|\text{V}/a|^F + \text{R}^O|\text{V}/a|^O) \) is used to recompute \( O \). A correction \((2\Omega \wedge ak)(O_{\text{improved}}) - (2\Omega \wedge ak)(O)\) is analytically computed and added to wind equation to find the “improved” value of \( \text{V}^F(t + \Delta t) \). Computations are currently done in routine \( \text{LAPINEB} \) and \( \text{LADINE} \).

* 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 \( \text{RW2TLFF}=1 \) on a great circle bow on geographical sphere. For \( \text{RW2TLFF}=0 \). \( ON=O \).

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 \( \eta^O \) between the top of the atmosphere and the layer \( l = 1 \), according to the namelist variable \( \text{VETAON} \) (resp. between the layer \( l = L \) and the ground, according to the namelist variable \( \text{VETAOX} \)). Upper bound of \( O \) is \( \eta^O = \eta^L + (\text{VETAOX} - 1)(\eta^L - \eta^L) \). Lower bound of \( O \) is \( \eta^O = \eta^L + (1 - \text{VETAOX})(\eta^L - \eta^L) \).
* Alternate treatment of vertical displacement (LRLTVDISP=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 $\eta$ and one with $\log((\eta - \eta_N)/\eta_X - \eta))$, and we finally take the shorter displacement computed.

- $\eta_N$ is a lower boundary for $\eta$ which matches $0 \leq \eta_N < \eta_{\text{top}}$ (currently taken equal to VETAON).
- $\eta_X$ is an upper boundary for $\eta$ which matches $\eta_{\text{top}} < \eta_X \leq 1$ (currently taken equal to VETAOX).

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

$$\log \left( \frac{\eta_F - \eta_N}{\eta_X - \eta} \right) = \Delta t \left( \log \left( \frac{\eta - \eta_N}{\eta_X - \eta} \right) \right)$$

This equation can be rewritten:

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

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 descending (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 $\eta$ 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 $L-1$:

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

- 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, NDPSF1=1 and $\Pi_{\text{top}} > 0$ which are not consistent with the constraint $(\dot{\eta}_{\text{top}} = 0; \eta_{\text{art}} = 0)$).
- 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 LARCRINHA.

* Option LSLDP_CURV=T: Trajectories are no longer great circles on the geographical sphere, but use cartesian coordinates on a plane projection. No extensive description is currently given for this option, which is currently available for untilted unstretched spherical geometry, and which is not relevant for plane geometry.
* **Option LSLDP\_RK=T:** Trajectories are still great circles on the geographical sphere, but NITMP-iterative algorithm uses exactly 5 iterations and a rather complicated algorithm involving a 4th order Runge-Kutta method. Routines LARMES\_RK and LARCHE\_INCR are called instead of LARMES and LARCHE.

No extensive description is currently given for this option, which is currently available for untitled unstretched spherical geometry. Extension to a stretched/tilted spherical geometry or a plane geometry has a sense and would require a significant amount of work, to be done later.

* **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. Code is currently limited to options LSLDP\_CURV=F and LSLDP\_RK=F.

* **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 LSETTLSSVF (SL2TL schemes):** When the “SETTLS” stable extrapolation is applied to $\dot{\eta}$, and when LSETTLSSVF=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 $X$, $A$ and $L$.

\[ X = V + \delta V (2\Omega \wedge r) \]  
(91)

\[ A = -2(1 - \delta V)(\Omega \wedge V) - \nabla \Phi \]  
(92)

\[ L = -\nabla \Phi + \beta_{Co}[-2(1 - \delta V)(\Omega \wedge V)] \]  
(93)

* Remarks.
- Coriolis term can be treated explicitly ($\delta V = 0$) or implicitly ($\delta V = 1$). Use key LADVF. LADVF=T matches with ($\delta V = 0$). LADVF=T matches with ($\delta V = 1$). If LADVF=T, term ($2\Omega \wedge r$) is analytically computed.
- For a limited set of options, term ($2\Omega \wedge r$) 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 $\beta_{Co}$ (which has a sense only if LADVF=T). 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 $\beta=1$, the non linear term ($-2(1 - \delta V)(\Omega \wedge V) - \nabla \Phi - \beta(-\nabla \Phi + \beta_{Co}[-2(1 - \delta V)(\Omega \wedge V)]$) is zero.

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

* Definition of $X$, $A$ and $L$.

\[ X = (\Phi - (1 - \delta_{TR})\Phi_s) \]  
(94)

\[ A = - (\Phi - \Phi_s)D + \delta_{TR}\nabla(\Phi_s) \]  
(95)

\[ L = -\Phi^*M^2D' \]  
(96)

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

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

* Definition of $X$, $A$ and $L$.

\[ X = (\Phi - \Phi_s)J \]  
(97)

\[ A = 0 \]  
(98)

\[ L = -\Phi^*M^2D'J \]  
(99)

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

* Calculation of $J$.
- SL3TL: $J^- = (1 - \Delta tD^o)/(1 + \Delta tD^o)$, $J^o = 1/(1 + \Delta tD^o)$ 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 \( L \) (like \( \tau, \gamma, \mu, \nu \)) are given in the documentation (IDSII).

8.1 Formulation of the momentum equation.

* Definition of \( X, A, L \) and \( F \), top and bottom values.

\[
X = V + \delta V (2\Omega \wedge r) \quad (100)
\]

\[
A = -2(1 - \delta V)(\Omega \wedge V) - \nabla \Phi - RT \nabla (\log \Pi) \quad (101)
\]

\[
L = -\nabla [\gamma T + R_d T^* \log (\Pi_0)] + \beta_C [2(1 - \delta V) (\Omega \wedge V)] \quad (102)
\]

\[
F = F_V \quad (103)
\]

Top:

\[
V_{\eta=0} = V_{\eta=1} \quad (104)
\]

Bottom if \( \delta \tau = 0 \):

\[
V_{\eta=1} = V_{\eta=L} \quad (105)
\]

Bottom if \( \delta \tau = 1 \):

\[
V_{\eta=1} = 0 \quad (106)
\]

* Remarks.

- Coriolis term can be treated explicitly (\( \delta V = 0 \)) or implicitly (\( \delta V = 1 \)). Use key LADVF. LADVF=F. matches with (\( \delta V = 0 \)). LADVF=T. matches with (\( \delta V = 1 \)). If LADVF=T., term \( (2\Omega \wedge r) \) is analytically computed.

- For a limited set of options, term \( (2\Omega \wedge r) \) 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 \( \beta_C \) (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 \( X, A, L \) and \( F \), top and bottom values.

\[
X = T + \delta T \frac{\alpha \Phi_s}{R_d T_s} \quad (107)
\]

\[
A = \frac{RT \omega}{c_p \Pi} + \delta T \frac{\alpha \Pi}{R_d T_s} \nabla (\Phi_s) + \delta T \frac{\Phi_s}{R_d T_s} \left( \frac{d \alpha T}{d \eta} \right) \quad (108)
\]

\[
L = -\tau (M^2 \dot{D}) \quad (109)
\]

\[
F = F_T \quad (110)
\]

Top:

\[
T_{\eta=0} = T_{\eta=1} \quad (111)
\]

Bottom if \( \delta \tau = 0 \):

\[
T_{\eta=1} = T_{\eta=L} \quad (112)
\]

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

\[
T_{\eta=1} = T_s \quad (113)
\]

8.3 Continuity equation.

* Definition of \( X, A, L, \) and \( F \).

\[
X = \log \Pi_s + \delta T \frac{\Phi_s}{R_d T_s} \quad (114)
\]

\[
A = -\frac{1}{\Pi_s} \int_{\eta=0}^{\eta=1} \nabla \left( \frac{\partial \Pi}{\partial \eta} \right) d\eta + \nabla \left( \log \Pi_s + \frac{\delta T}{\Pi_s} \frac{\Phi_s}{R_d T_s} \right) - \frac{1}{\Pi_s} \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=1} + \frac{1}{\Pi_s} \left[ \frac{\partial \Pi}{\partial \eta} \right]_{\eta=0} \quad (115)
\]

\[
L = -\frac{M^2}{M^2} \nu D \quad (116)
\]

\[
F = \left( \frac{F_m}{\Pi_s} \right) \quad (117)
\]
8.4 Moisture equation.

Definition of $X$, $A$, $L$ and $F$, top and bottom values.

- $X = q$ (118)
- $A = 0$ (119)
- $L = 0$ (120)
- $F = F_q$ (121)

Top:

- $q_{n=0} = q_{n=1}$ (122)

Bottom if $\delta m = 0$:

- $q_{n=1} = q_{n=L}$ (123)

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

- $q_{n=1} = q_{surf}$ (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 ($\text{LAGPHY}=\text{T.}$, $\text{LSLPHY}=\text{F.}$): the previous discretisations are done without physics, then the provisional

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

or

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

is used as input to the lagged physics.

- For split physics used at ECMWF ($\text{LEPHYS}=\text{T.}$, $\text{LAGPHY}=\text{T.}$, $\text{LSLPHY}=\text{T.}$): 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 $([U/a],[V/a],\eta)$ is needed:

- A 56 point “WENO” high-order interpolation is performed if ($\text{LHOILST,LSLTWENO}=(\text{T,T})$).
- A 32 point high-order interpolation is performed if ($\text{LHOILST,LSLTWENO}=(\text{T,F})$).
- A tri-linear interpolation is performed otherwise.

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 $\text{RW2TLFF}>0$. The two components of the $[V/a]$ 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 L (like $\tau$, $\gamma$, $\mu$, $\nu$) are given in the documentation (IDSI).

9.1 Momentum equation.
* Definition of $X$, $A$, $L$ and $F$, top and bottom values.

\[ X = V + \delta V (2\Omega \wedge r) \]  
\[ A = [-2(1-\delta V)(\Omega \wedge V)] - \frac{\partial p}{\partial \Pi} \nabla \Phi - RT \frac{\nabla (p)}{p} \]  
\[ L = -\nabla [\gamma T - T^* (\gamma \hat{Q}) + R_d T^* \log(\Pi_s) + R_d T^* \hat{Q}] + \beta \psi [\tau - 2(1-\delta V)(\Omega \wedge V)] \]  
\[ F = F_V \]

Top:  
\[ V_{\eta=0} = V_{l=1} \]  
Bottom if $\delta m = 0$:  
\[ V_{\eta=1} = V_{l=L} \]

Bottom if $\delta m = 1$:  
\[ V_{\eta=1} = 0 \]

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

9.2 Temperature equation.
* Definition of $X$, $A$, $L$ and $F$, top and bottom values.

\[ X = T + \delta T \frac{\alpha_T \Phi_s}{R_d T_{st}} \]  
\[ A = \frac{RT}{c_v} D_3 + \delta T \frac{\alpha_T}{R_d T_{st}} V \nabla (\Phi_s) + \delta T \frac{\Phi_s}{R_d T_{st}} \left( \frac{d\alpha_T}{dt} \right) \]  
\[ L = -\frac{R_d T^*}{c_v} \left[ \nabla^2 \tilde{D} + \frac{1}{\partial \eta} \left( \frac{\partial \eta}{\partial \eta} \right) \right] \]  
\[ F = \left[ c_p c_v F_T \right] \]

Top:  
\[ T_{\eta=0} = T_{l=1} \]  
Bottom if $\delta m = 0$:  
\[ T_{\eta=1} = T_{l=L} \]  
Bottom if $\delta m = 1$ (output of physics):  
\[ T_{\eta=1} = T_s \]

9.3 Vertical divergence variable equation.
* Definition of $X$, $A$, $L$ and $F$, top and bottom values, case NVDVAR=3, LGWADV=.F.

\[ X = d \]  
\[ A = -dD_3 + d \nabla V = \frac{gp}{R_d T_{st}} \frac{\partial u}{\partial \eta} + \frac{gp}{R_d T_{st}} (\nabla u) \left( \frac{\partial V}{\partial \eta} \right) \]  
\[ L = -g^2 \frac{1}{R_d T_s} (L^* \hat{Q}) \]  
\[ F = \frac{d}{\partial \eta} F_m' - \left[ \frac{gp}{R_d T_{st}} \frac{\partial F_m}{\partial \eta} \right] \]
Top: \[ d_{\eta=0} = d_{l=1} \] \hspace{1cm} (143)
Bottom if \( \delta m = 0 \): \[ d_{\eta=1} = d_{l=L} \] \hspace{1cm} (144)
Bottom if \( \delta m = 1 \): not yet coded.

* Definition of \( X \), \( A \), \( L \) and \( F \), top and bottom values, case NVDVAR=3, LGWADV=.T.. The linear system uses the \( d \) equation.
\[ X = w \] \hspace{1cm} (145)
\[ A = g \frac{\partial(p - \Pi)}{\partial \Pi} \] \hspace{1cm} (146)
\[ F = F_w \] \hspace{1cm} (147)
Top:
- \textbf{LVERTFE}=.F.: \( w_{\eta=0} \) is computed by the general formula giving \( w \) at half levels.
- \textbf{LVERTFE}=.T.: VFD treatment for \( w_{\eta=0} \) (cf. above).
Bottom if \( \delta m = 0 \): \[ w_{\eta=1} = V_{surf} \nabla z_{surf} \] \hspace{1cm} (148)
Bottom if \( \delta m = 1 \): not yet coded.

9.4 Pressure departure variable equation.
* Definition of \( X \), \( A \), \( L \) and \( F \), top and bottom values, case NPDVAR=2.
\[ X = \hat{Q} \] \hspace{1cm} (149)
\[ A = \frac{c_p}{c_v} D_3 - \frac{\omega}{\Pi} \] \hspace{1cm} (150)
\[ L = - \left[ \frac{c_p d}{c_v d} (\overline{M}^2 D' + d) - \frac{c_p d}{R_d T^*} (\overline{M}^2 D') \right] \] \hspace{1cm} (151)
\[ F = \frac{c_p}{c_v} F_T \] \hspace{1cm} (152)
Top: \[ \hat{Q}_{\eta=0} = 0 \] \hspace{1cm} (153)
Bottom if \( \delta m = 0 \): \[ \hat{Q}_{\eta=1} = \hat{Q}_{l=L} \] \hspace{1cm} (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 $L$ (like $\tau$, $\gamma$, $\mu$, $\nu$) are given in the documentation (IDSI).

10.1 Momentum equation.

Definition of $X$, $A$, $L$ and $F$, top and bottom values.

$$X = V + \delta_V(2\Omega \wedge r)$$  \hspace{1cm} (155)

$$A = [-2(1 - \delta_V)(\Omega \wedge V)] - \exp(\kappa_m \hat{Q}) \left( 1 + \Pi \frac{\partial \hat{Q}}{\partial \Pi} \right) \nabla \Phi - RT \exp(\kappa_m \hat{Q}) \left( \frac{\nabla \Pi}{\Pi} + \nabla \hat{Q} \right)$$  \hspace{1cm} (156)

$$L = -\nabla [\gamma \hat{T} + R_dT^* \log(\Pi_\circ) + R_dT^\circ] + \beta_{\circ}[-2(1 - \delta_V)(\Omega \wedge V)]$$  \hspace{1cm} (157)

$$\mathcal{F} = F_V$$  \hspace{1cm} (158)

Top:

$$V_{\eta=0} = V_{\circ}$$  \hspace{1cm} (159)

Bottom if $\delta m = 0$:

$$V_{\eta=1} = V_{\circ}$$  \hspace{1cm} (160)

Bottom if $\delta m = 1$:

$$V_{\eta=1} = 0$$  \hspace{1cm} (161)

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

10.2 Temperature equation.

Definition of $X$, $A$, $L$ and $F$, top and bottom values.

$$X = T + \delta_{TR} \frac{\alpha_T \Phi_s}{R_d T_{st}}$$  \hspace{1cm} (162)

$$A = \frac{RT^*}{c_p} \frac{\omega}{\Pi} + \delta_{TR} \frac{\alpha_T}{R_d T_{st}} \nabla V(\Phi_s) + \delta_{TR} \frac{\Phi_s}{R_d T_{st}} \left( \eta \frac{d \alpha_T}{d \eta} \right)$$  \hspace{1cm} (163)

$$L = -\tau (\mathcal{M}^2 D')$$  \hspace{1cm} (164)

$$\mathcal{F} = \exp(-\kappa_m \hat{Q}))F_T$$  \hspace{1cm} (165)

Top:

$$T_{\eta=0} = T_{\circ}$$  \hspace{1cm} (166)

Bottom if $\delta m = 0$:

$$T_{\eta=1} = T_{\circ}$$  \hspace{1cm} (167)

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

$$T_{\eta=1} = T_s$$  \hspace{1cm} (168)

10.3 Vertical divergence variable equation.

Definition of $X$, $A$, $L$ and $F$, top and bottom values, case NVDVAR=3, LGWADV=.F.

$$X = d$$  \hspace{1cm} (169)

$$A = -d(d + x_b) - \frac{g \Pi}{RT^* \frac{\partial}{\partial \eta}} \frac{\partial}{\partial \eta} \left( \frac{\partial F}{\partial \eta} \right) + \frac{g \Pi}{RT^* \frac{\partial}{\partial \eta}} (\nabla V) \left( \frac{\partial V}{\partial \eta} \right)$$  \hspace{1cm} (170)

$$L = \frac{g^2}{R_d T_{st}} (\mathcal{L}_s \hat{Q})$$  \hspace{1cm} (171)

$$\mathcal{F} = -d \left( \frac{\partial}{\partial \eta} \right) F_m' - \left[ \frac{g \Pi}{RT^* \frac{\partial}{\partial \eta}} \frac{\partial F_m}{\partial \eta} \right]$$  \hspace{1cm} (172)
Top: \( d_{\eta=0} = d_{l=1} \)  
Bottom if \( \delta m = 0 \): \( d_{\eta=1} = d_{l=L} \)  
Bottom if \( \delta m = 1 \): not yet coded.

* Definition of \( X, A, L \) and \( F \), top and bottom values, case NVDVAR=3, LGWADV=.T. . The linear system uses the \( d \) equation.

\[
\begin{align*}
X &= w \\
A &= g \frac{\partial \Pi(\exp(\kappa_m \hat{Q}) - 1)}{\partial H} + (\kappa_m - 1)(\exp(\kappa_m \hat{Q}) - 1) \\
F &= F_w
\end{align*}
\]

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

Bottom if \( \delta m = 0 \): \( w_{\eta=1} = \nabla z_{\text{surf}} \)  
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 \texttt{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 $R^{OF}$ is defined:

$$V' = R^{OF}(V)$$ (179)

where $V'$ has coordinates $(u',v')$, $V$ has coordinates $(u,v)$, and the relationship between $(u,v)$ and $(u',v')$ is:

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

where:

$$p = \frac{iF^O + jF^O}{1 + k^F k^O} = \frac{\cos \theta^F \cos \theta^O + (1 + \sin \theta^F \sin \theta^O) \cos(\lambda^F - \lambda^O)}{1 + \cos \phi}$$ (181)

$$q = \frac{iF^O + jF^O}{1 + k^F k^O} = \frac{(\sin \theta^F + \sin \theta^O) \sin(\lambda^F - \lambda^O)}{1 + \cos \phi}$$ (182)

(Denotations $\theta^O, \theta^F, \lambda^O, \lambda^F, \phi$: see section 6.).

$p$ and $q$ verify the following identity:

$$p^2 + q^2 = 1$$ (183)

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

* Tilting.

The coordinates of $V'$ and $V$ are linked by the following relationship:

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

where:

$$\cos \alpha = \frac{2c}{A \cos \Theta^O} [\sin \theta_p \cos \Theta^O - \sin \theta^O \cos \theta_p \cos(\lambda^O - \lambda_p)]$$ (185)

$$\sin \alpha = \frac{2c}{A \cos \Theta^O} [\cos \theta_p \sin(\lambda^O - \lambda_p)]$$ (186)

$$A = (1 + c^2) + (1 - c^2) (\sin \theta_p \sin \Theta^O + \cos \theta_p \cos \Theta^O \cos(\lambda^O - \lambda_p))$$ (187)

and where:

- $c$ is the stretching coefficient.
- $\Theta^O$ is the latitude on the computational sphere of the origin point $O$.
- $(\theta_p, \lambda_p)$ 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 $R^{OF}$ 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 \((\Theta^O, \Lambda^O)\), latitude and longitude of the interpolation point \(O\) in the computational sphere. The iterative algorithm allowing to find \(O\) gives \((\theta^O, \lambda^O)\), 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 \((\theta^F, \lambda^F)\) are the coordinates of the final point on the geographical sphere). Transform formulae giving \((\Theta, \Lambda)\) on the computational sphere once knowing \((\theta, \lambda)\) on the geographical sphere are given by equations (188) to (190).

\[
\sin \Theta = \frac{(1 - c^2) + (1 + c^2)(\sin \theta_p \sin \theta + \cos \theta_p \cos \theta \cos(\lambda - \lambda_p))}{A} \tag{188}
\]

\[
\cos \Theta \cos \Lambda = \frac{2c(\cos \theta_p \sin \theta - \sin \theta_p \cos \theta \cos(\lambda - \lambda_p))}{A} \tag{189}
\]

\[
\cos \Theta \sin \Lambda = \frac{2c \cos \theta \sin(\lambda - \lambda_p)}{A} \tag{190}
\]

where:

- • \(A = (1 + c^2) + (1 - c^2)(\sin \theta_p \sin \theta + \cos \theta_p \cos \theta \cos(\lambda - \lambda_p))\)
- • \(c\) is the stretching coefficient.
- • \((\theta_p, \lambda_p)\) are the latitude and longitude on the geographical sphere of the stretching pole.
- • Computation of \(\Theta, \Lambda\) is done in subroutine \textsc{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:

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

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

\[
\dot{\eta}_h = 0.5 \left[ \left( \eta \frac{\partial \Pi}{\partial \eta} \right)_{1} + \left( \eta \frac{\partial \Pi}{\partial \eta} \right)_{-1} \right] \frac{[\Delta \eta]_i}{[\Delta \Pi]_i} \tag{192}
\]

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

\[
\dot{\eta} = \left( \eta \frac{\partial \Pi}{\partial \eta} \right)_{1} \frac{[\Delta \eta]_i}{[\Delta \Pi]_i} \tag{193}
\]
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 \( C_2 \). \( \Lambda \) and \( \Theta \) are the longitudes and latitudes on the computational sphere. Linear weights are defined as follows:

- **Zonal weights for latitudes 1 and 2:**
  \[ ZDLO_1 = (\Lambda_O - \Lambda_{B_1})/(\Lambda_{C_1} - \Lambda_{B_1}) \quad \text{and} \quad ZDLO_2 = (\Lambda_O - \Lambda_{B_2})/(\Lambda_{C_2} - \Lambda_{B_2}). \]
- **Meridian weight:**
  \[ ZDLAT = (\Theta_O - \Theta_{B_1})/(\Theta_{B_2} - \Theta_{B_1}). \]

* Computations:

- The weight \( ZDLO_1 \) and \( ZDLO_2 \) 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 \( T_{i+1} \) and \( T_{i+2} \). The vertical weight is defined by:
  \[ ZDVER = (\eta_O - \eta_{T_{i+1}})/(\eta_{T_{i+2}} - \eta_{T_{i+1}}) \]

* Computations:

- The weight \( ZDVER \) is computed then stored in the array \( PDVER \).
- The level number \( l \) of \( T_i \) is stored in the array \( KLEV \).
- Interpolations use data of points \( T_{i+1} \) and \( T_{i+2} \).

* Remark: The same formulae and computations are valid for half level data, simply replace the layer index \( l \) by the half level index \( l/2 \).

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:**
  \[ ZDLO_0 = (\Lambda_O - \Lambda_{B_0})/(\Lambda_{C_0} - \Lambda_{B_0}) \]
  \[ ZDLO_1 = (\Lambda_O - \Lambda_{B_1})/(\Lambda_{C_1} - \Lambda_{B_1}) \]
  \[ ZDLO_2 = (\Lambda_O - \Lambda_{B_2})/(\Lambda_{C_2} - \Lambda_{B_2}) \]
  \[ ZDLO_3 = (\Lambda_O - \Lambda_{B_3})/(\Lambda_{C_3} - \Lambda_{B_3}) \]

- **Zonal cubic weights for latitude 1:**
  \[ ZCLO11 = f_1(ZDLO1) \]
  \[ ZCLO12 = f_2(ZDLO1) \]
  \[ ZCLO13 = f_3(ZDLO1) \]

where:

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

- **Zonal cubic weights for latitude 2:**
  \[ ZCLO21 = f_1(ZDLO2) \]
  \[ ZCLO22 = f_2(ZDLO2) \]
  \[ ZCLO23 = f_3(ZDLO2) \]

38
• meridian cubic weights:
  \[ ZC\Lambda 1 = [(\Theta_2 - \Theta_B)(\Theta_B - \Theta_1)(\Theta_2 - \Theta_1)] / [(\Theta_B - \Theta_1)(\Theta_B - \Theta_2)] \]
  \[ ZC\Lambda 2 = [(\Theta_3 - \Theta_B)(\Theta_B - \Theta_1)(\Theta_2 - \Theta_1)] / [(\Theta_B - \Theta_1)(\Theta_B - \Theta_2)] \]
  \[ ZC\Lambda 3 = [(\Theta_1 - \Theta_B)(\Theta_B - \Theta_2)(\Theta_2 - \Theta_1)] / [(\Theta_B - \Theta_2)(\Theta_B - \Theta_1)] \]

* 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 PCLLO.
  - The meridian cubic weights ZCCLA1, ZCCLA2 and ZCCLA3 are computed then stored in the array PCLA. One can notice that the denominators of ZCCLA1, ZCCLA2 and ZCCLA3 do not depend on coordinates of O and can be pre-computed in array RIP1.
  - The memory address (in SL arrays) of the data concerning the points A0, A1, A2 and A3 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 B0, C0, A1, B1, C1, D1, A2, B2, C2, D2, B3 and C3.

* 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 (IDS/LP2) and looking in routine LASCW (ELASCW in LAM models). We just give a sum-up of the way to compute the cubic weights:
  - Meridian diffusive cubic weights:
    \[ ZC\Lambda A_{1ahlh} = ZC\Lambda A + ZINCRLM1 \]
    \[ ZC\Lambda A_{2ahlh} = ZC\Lambda A + ZINCRLM2 \]
    \[ ZC\Lambda A_{3ahlh} = ZC\Lambda A + ZINCRLM3 \]
    where each of increments ZINCRLM1, ZINCRLM2 and ZINCRLM3 is a linear combination of cubic Lagrangian weights ZC\Lambda A1, ZC\Lambda A2 and ZC\Lambda A3 and diffusive weights (linear for LSLHD\_OLD=.T., quadratic otherwise).
    Coefficients of these linear combinations depend on:
    - Constants in time Cshift, describing Laplacian smoother.
    - A pre-computed quantity \( \kappa \) (computed in GP\_KAPPA) depending on horizontal flow deformation, ranging from 0 to 1; \( \kappa \) is equal to 1 if LSLHD\_STATIC=.T.. An alternate expression of \( \kappa \) 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:
    \[ ZC\Lambda O1_{1ahlh} = ZC\Lambda O1 + ZINCRL11 \]
    \[ ZC\Lambda O1_{2ahlh} = ZC\Lambda O1 + ZINCRL12 \]
    \[ ZC\Lambda O1_{3ahlh} = ZC\Lambda O1 + ZINCRL13 \]
    where each of increments ZINCRL11, ZINCRL12 and ZINCRL13 is a linear combination of cubic Lagrangian weights ZC\Lambda O11, ZC\Lambda O12 and ZC\Lambda O13 and diffusive weights (linear for LSLHD\_OLD=.T., quadratic otherwise).
    Coefficients of these linear combinations depend on:
    - Constants like C\_slhddepsh.
    - \( \kappa \) (see above).
    - Lower and upper bounds \( \kappa_{min} \) and \( \kappa_{max} \) (see above).
  - Zonal diffusive cubic weights for latitude number 2:
    \[ ZC\Lambda O2_{1ahlh} = ZC\Lambda O2 + ZINCRL21 \]
    \[ ZC\Lambda O2_{2ahlh} = ZC\Lambda O2 + ZINCRL22 \]
    \[ ZC\Lambda O2_{3ahlh} = ZC\Lambda O2 + ZINCRL23 \]
    where each of increments ZINCRL21, ZINCRL22 and ZINCRL23 is a linear combination of cubic Lagrangian weights ZC\Lambda O21, ZC\Lambda O22 and ZC\Lambda O23 and diffusive weights (linear for LSLHD\_OLD=.T., quadratic otherwise).
    Coefficients of these linear combinations depend on C\_slhddepsh, \( \kappa \), \( \kappa_{min} \), \( \kappa_{max} \) 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:

\[
ZCVE_1 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+2}})(\eta_O - \eta_{T_{l+3}})]/[\eta_{T_{l+1}} - \eta_{T_l}][\eta_{T_{l+2}} - \eta_{T_{l+1}}][\eta_{T_{l+3}} - \eta_{T_{l+2}}]
\]

\[
ZCVE_2 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+1}})(\eta_O - \eta_{T_{l+3}})]/[\eta_{T_{l+2}} - \eta_{T_l}][\eta_{T_{l+1}} - \eta_{T_{l+2}}][\eta_{T_{l+3}} - \eta_{T_{l+2}}]
\]

\[
ZCVE_3 = [(\eta_O - \eta_{T_l})(\eta_O - \eta_{T_{l+1}})(\eta_O - \eta_{T_{l+2}})]/[\eta_{T_{l+3}} - \eta_{T_l}][\eta_{T_{l+2}} - \eta_{T_{l+3}}][\eta_{T_{l+3}} - \eta_{T_{l+2}}]
\]

* Computations:

- The vertical weights $ZCVE_1$, $ZCVE_2$ and $ZCVE_3$ are computed then stored in the array PVINTW. One can notice that the denominators of $ZCVE_1$, $ZCVE_2$ and $ZCVE_3$ do not depend on coordinates of $O$ and can be pre-computed in the array VCUICO.

- The level number $l$ of $T_j$ is stored in the array KLEV.

- Interpolations use data of points $T_i$, $T_{i+1}$, $T_{i+2}$ and $T_{i+3}$.

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

\[
ZCVE_{1\text{slhd}} = ZCVE_1 + ZINCRV_1
\]

\[
ZCVE_{2\text{slhd}} = ZCVE_2 + ZINCRV_2
\]

\[
ZCVE_{3\text{slhd}} = ZCVE_3 + ZINCRV_3
\]

Expressions giving vertical increments $ZINCRV_1$ to $ZINCRV_3$ have a shape similar to those of horizontal increments, and still depend on quantities like $\kappa$, $\kappa_{\text{min}}$, $\kappa_{\text{max}}$.

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 $\left(\frac{\dot{\eta}(\frac{\partial X}{\partial \eta})}{\dot{\eta}}\right)$ but with additional approximations allowing to avoid horizontal interpolations for term $\left(\frac{\partial X}{\partial \eta}\right)$.

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

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

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

General formula to evaluate the following weights are:

\[
VDERW11 = 0.5(\eta_{l+2} - \eta_{l+1})(\eta_{l+2} - \eta_l)
\]

\[
VDERW12 = 0.5(\eta_{l+2} - \eta_{l+1})(\eta_{l+2} - \eta_l)
\]

\[
VDERW21 = 0.5(\eta_{l+2} - \eta_{l+1})(\eta_{l+2} - \eta_l)
\]

\[
VDERW22 = 0.5(\eta_{l+2} - \eta_{l+1})(\eta_{l+2} - \eta_l)
\]

with particular treatment when the interpolation point is near the top or the bottom boundary.
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 \( T_i \) and \( T_{i+2} \).

* 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 RVSPSC and pre-computed in the set-up subroutine SUVSPLIP; the original field is stored in the buffer \( P(X)9 \) 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 \( ZCVF0, ZCVF1, ZCVF2 \) and \( ZCVF3 \) necessary for the second part are computed then stored in the array \( PVNTV5 \). Some part of the calculations can be pre-computed in the set-up subroutine SUVSPLIP (array RFVV).
  - The level number \( l \) of \( T_i \) is stored in the array \( KLEV \).
  - Interpolations use data of points \( T_i, T_{i+1}, T_{i+2} \) and \( T_{i+3} \), 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 \( B_{l+4} \), \( C_{l+4} \), \( B_{l+1} \), \( C_{l+1} \), \( B_{l+2} \), \( C_{l+2} \), \( B_{l+3} \), \( C_{l+3} \), \( B_{l+5} \), \( C_{l+5} \), \( B_{l+6} \), \( C_{l+6} \) and \( C_{l+1} \). 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.

* Remark: The same formulae and computations are valid for half level data, simply replace the layer index \( l \) by the half level index \( l' \).

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 \( B_{l+4} \), \( C_{l+4} \), \( B_{l+1} \), \( C_{l+1} \), \( B_{l+2} \), \( C_{l+2} \), \( B_{l+3} \), \( C_{l+3} \), \( B_{l+5} \), \( C_{l+5} \), \( B_{l+6} \), \( C_{l+6} \) and \( C_{l+1} \). For the two levels \( l+1 \) 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.

* Remark: The same formulae and computations are valid for half level data, simply replace the layer index \( l \) by the half level index \( l' \).
Remarks:
- The same formulae and computations are valid for half level data, simply replace the layer index \( l \) by the half level index \( \tilde{l} \).
- 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).

14.1.12 Modified weights for COMAD 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 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 \( B_{1,1}, C_{1,1}, B_{2,1}, C_{2,1}, B_{1,2}, C_{1,2}, B_{2,2} \) and \( C_{2,2} \). 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 \( ZDLAT \) is required; see subsubsection 14.1.2 for vertical linear interpolations.

- The memory address (in SL arrays) of the data concerning the points \( A_{1,1} \) and \( A_{2,1} \) 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 \( A_{0,1}, B_{0,1}, C_{0,1}, D_{0,1}, A_{1,1}, B_{1,1}, C_{1,1}, D_{1,1}, A_{2,1}, B_{2,1}, C_{2,1}, D_{2,1}, A_{3,1}, B_{3,1}, C_{3,1}, D_{3,1}, A_{0,2}, B_{0,2}, C_{0,2}, D_{0,2}, A_{1,2}, B_{1,2}, C_{1,2}, D_{1,2}, A_{2,2}, B_{2,2}, C_{2,2}, D_{2,2}, A_{3,2}, B_{3,2}, C_{3,2}, D_{3,2} \).

Remarks:
- The same formulae and computations are valid for half level data, simply replace the layer index \( l \) by the half level index \( \tilde{l} \).

14.1.11 Interpolation grid and weights for 56 points “WENO” interpolations.

This kind of grid is currently not extensively described (to be done later); compared to the 32 point interpolations grid it uses 6 levels instead of 4.

14.1.12 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:

\[
ZDL_{\text{comad}} = ZDL + q(ZDL) \ast (ZDL - 0.5) \ast (STDDIS - 1)
\]

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 $C\infty$ 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 $q_c$:

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

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

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

14.1.13 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 $l$.

14.2.3 Horizontal 12 points interpolation (subroutine LAIDDII).

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 LQMWM (momentum equation), LQMT (temperature equation), LQMP (continuity equation), LQMSPD (pressure departure variable equation), LQMSVD (vertical divergence equation), $Y[X]_\text{NL}\%LQM$ 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 $ZCVE_1$, $ZCVE_2$ and $ZCVE_3$. The cubic 4 points vertical interpolation gives the final interpolated value:

$$X_{\text{interpolated}} = X_l + ZCVE_1(X_{l+1} - X_l) + ZCVE_2(X_{l+2} - X_{l+1}) + ZCVE_3(X_{l+3} - X_{l+2})$$

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

14.2.5 Cubic Hermite vertical interpolation.

This part is valid for interpolations of full level variables. See figure 14.2 and subsection 14.1.5 for definition of $VDERW_{11}$, $VDERW_{21}$, $VDERW_{12}$ and $VDERW_{22}$. See subsection 14.1.2 for definition of $ZDV$. See subsubsection 14.1.5 for definition of functions $f_{\text{H1}}$ to $f_{\text{H4}}$. The cubic Hermite vertical interpolation gives the final interpolated value:

$$X_{\text{interpolated}} = f_{\text{H1}}(ZDVER)X_{l+1} + f_{\text{H2}}(ZDVER)X_{l+2} + f_{\text{H3}}(ZDVER)(VDERW_{11}(X_{l+1} - X_l) + VDERW_{21}(X_{l+2} - X_{l+1})) + f_{\text{H4}}(ZDVER)(VDERW_{12}(X_{l+2} - X_{l+1}) + VDERW_{22}(X_{l+3} - X_{l+2}))$$

14.2.6 Spline cubic 4 points vertical interpolation.

This part is valid for interpolations of full level variables. See figure 14.2 and subsection 14.1.6 for definition of $ZCVE_0$, $ZCVE_1$, $ZCVE_2$ and $ZCVE_3$. The spline cubic 4 points vertical interpolation gives the final interpolated value:

$$X_{\text{interpolated}} = ZCVE_0 \ast XRP + ZCVE_1 \ast XRP_{l+1} + ZCVE_2 \ast XRP_{l+2} + ZCVE_3 \ast XRP_{l+3}$$

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 subsection 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 subsection 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 subsection 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), $LQM\mbox{P}$ (continuity equation), $LQMSPD$ (pressure departure variable equation), $LQMSVD$ (vertical divergence equation), $Y[X]_{\mbox{NL}}\%LQM$ (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), $LQMHS\mbox{PD}$ (pressure departure variable equation), $LQMHSVD$ (vertical divergence equation), $Y[X]_{\mbox{NL}}\%LQMH$ (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 subsection 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 subsection 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 subsection 14.2.5).

There are shape-preserving options for horizontal or both horizontal and vertical interpolations.

Use of switch $Y[X]_{\mbox{NL}}\%LQM$ (GFL equations), allows to use shape-preserving option for both horizontal and vertical interpolations.

Use of switch $Y[X]_{\mbox{NL}}\%LQMH$ (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 \( Y[X,NL%LVSP] \) 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: \( B_{0,i+1}, C_{0,i+1}, A_{1,i+1}, B_{1,i+1}, C_{1,i+1}, D_{1,i+1}, A_{2,i+1}, B_{2,i+1}, C_{2,i+1}, D_{2,i+1}, B_{3,i+1}, C_{3,i+1} \). 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 \( B_{1,i+1}, C_{1,i+1}, B_{2,i+1}, C_{2,i+1}, B_{1,i+2}, C_{1,i+2}, B_{2,i+2}, C_{2,i+2} \), 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 \( \text{lon} \) for \( \text{lon} = 0, 1, 2, 3 \):
  \[
  X_{\text{lon}} = f_1(ZDLO_{\text{lon}})X_{A_\text{lon}} + f_2(ZDLO_{\text{lon}})X_{B_\text{lon}} + f_3(ZDLO_{\text{lon}})X_{C_\text{lon}} + f_4(ZDLO_{\text{lon}})X_{D_\text{lon}}.
  \]
- 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 \( X_{i+1} \) and \( X_{i+2} \) (see subsubsection 14.2.10). The final interpolated value is then given by the following expression:

\[
X_{\text{interpolated}} = X_{i+1} + ZDVER(X_{i+2} - X_{i+1})
\]

* Remark: In the current usage of this interpolation (for \( j \) in the upper stratosphere) there is an additional smoothing done in routine LAISMOOA before the interpolation by LAISMOO.

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

14.2.12  56 points 3D “WENO” interpolation with vertical 6 points interpolation (subroutine LAITRI_WENO).

This kind of interpolation is currently not extensively described (to be done later); compared to the 32 point interpolations grid it uses 6 levels instead of 4.

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.
\[y = 1\]
\[x_1 = ZDLO1\]
\[y = 0\]
\[x_2 = ZDLO2\]
\[y = 1\]
\[x_1(B_1) = x_2(B_2) = 0; \quad x_1(C_1) = x_2(C_2) = 1.\]

Figure 14.1: Interpolation horizontal grid for bilinear interpolations.
Figure 14.2: Interpolation vertical grid for linear and cubic vertical interpolations.
Figure 14.3: Interpolation horizontal grid for 12 points interpolations.

\[ y = -ZDD \]
\[ y = -ZDA \]
\[ y = -ZDB \]
\[ y = -ZDC \]

\[ x_0(B_0) = x_1(B_1) = x_2(B_2) = x_3(B_3) = 0;\]
\[ x_0(C_0) = x_1(C_1) = x_2(C_2) = x_3(C_3) = 1.\]
Figure 14.4: 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 full level variables at the medium point: The medium point has a vertical coordinate always included between $\eta_{l=0}$ and $\eta_{l=L}$ in case of vertical interpolating scheme. Therefore no extrapolated values are needed.

* Vertical cubic 4 points interpolations for full level 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 full level 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 full level 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.

* Vertical “WENO” 6 points interpolations for full level variables at the origin point: No extensive description currently done, to be done later.
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).
  - LAITRI_WENO: 56 points 3D interpolations, with vertical 6 points interpolations.
  - 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.
  - LAITRIQ3D: 32 points 3D interpolations with 3D quasi-monotonic limiters.

* Other routines:
  - LAISMEO: 32 points interpolation by linear least-square fit.
  - LAISMOA: preliminary smoothing applied on quantities interpolated by the routine LAISMEO.
  - 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.

  - LACDYNSSHW: 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.
  - LARCH2: computes latitude, longitude of the medium/origin point coordinates on the computational sphere, \( p \) and \( q \) of the \( R \) operator matrix.

16.3 Routines used in 3D organigramme.

  - GP_MODEL: interface for grid-point calculations.
  - LACONE: computes analytically \( \left( 2\Omega \wedge \mathbf{a} \right) \).
  - 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 $\kappa_*$ required in the SLHD interpolations. Called only if $\text{LSLHD}=.T.$.
• GP_KAPPAT: alternate way to compute $\kappa_*$ for thermic variables.
• GP_STDDIS: computes the stretching/shrinking deformation $\text{STDDIS}$ required in the COMAD interpolations. Called only if $\text{LCOMAD}=.T.$.
• LACDYN: computation of the SL quantities, and semi-implicit scheme quantities.
• LARCH: computes latitude, longitude of the medium/origin point coordinates on the computational sphere, $p$ and $q$ of the $\mathcal{R}$ operator matrix. \text{ELARCHE} is called instead of \text{LARCH} for plane geometry.
• LARCH_INCR: alternate version of \text{LARCH} called if $\text{LSLDP\_RK}=.T.$.
• LASURE: some initialisations.
• LASSIE: computation of linear terms for hydrostatic equations.
• LANSHE: computation of linear terms for NHEE equations.
• LANHSIB: fills the $\text{P[X]SI}$ arrays in the NH model when $\text{LGWADV}=.T.$.
• LAVENT: computation of quantities to be interpolated (wind) for trajectory research.
• LATTE_NL: for mixed extrapolation, computes coefficient $C_\gamma$.
• LATTEX: computation of quantities to be interpolated for 3D variables equations (GFL and GMV).
• LATTEX_TNT: called by \text{LATTEX} for a SL3TL scheme, part of the code which is the same for all variables.
• LATTEX_DNT: called by \text{LATTEX} for a SL2TL scheme, part of the code which is the same for all variables.
• LATTE_KAPPA: stores $\kappa_*$ in the buffer “SLBUF2”. Called only if $\text{LSLHD}=.T.$.
• LATTE_STDDIS: stores $\text{STDDIS}$ in the buffer “SLBUF2”. Called only if $\text{LCOMAD}=.T.$.
• LATTE_BBC: computes the additional quantities to be stored in the buffers “SLBUF1” and “SLBUF2” for the option $\text{LRDBBC}=.T.$.
• 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.
• LARME: computation of medium point by an iterative algorithm for 3D model. Computes also the origin point. \text{ELARME} is called instead of \text{LARME} for plane geometry.
• LARME_RK: alternate version of \text{LARME} called if $\text{LSLDP\_RK}=.T.$.
• 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 \text{LARCINA} but for half level data.
• LARCINHB: the same as \text{LARCINB} but for half level data.
• GNHW2SVD: manages the conversions from $g\psi$ towards $d$ in the non-hydrostatic models when the key $\text{LGWADV}$ is put to $.T.$.
• MF_PHYS_PREP: preparation of input data for \text{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).
• VDIIFLCZ: 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
* SU0YOMA ->
  - SUGEOMETRY ->
    * geometry setup
    * SUINTERPOLATOR -> SUVSPLIP, SUVSLETA, SUHSLMER
* SU0YOMB ->
  - SUDYN ->
  * SUINTDYNSL
  - SUSC2B ->
    * SUSLB or SUSLB2
    * SLCSET -> SLRSET
  * SUSLAD1
  * SUSLAD2

16.6 Organigramme for 2D model.

STEPO -> SCAN2M -> GP_MODEL_[HEAP,HEAP2,STACK] -> GP_MODEL ->
* CPG2 ->
  - GPTF2
  - LACDYNSHW
  - WRSLTRAJ2
* SLCOMM and SLEXTPOL
* CPG2LAG ->
  - LADINE ->
    * LARMES2 -> LARCM2 ->
      - LARCHE2
      - LASCW (ELASCW in LAM models) -> LASCW_CLA, LASCW_CLD
      - LAIDDI
    * LAINOR2 -> LARCM2 ->
      - LARCHE2
      - LASCW (ELASCW in LAM models) -> LASCW_CLA, LASCW_CLD
      - LAIDDI
    * LACONE
    * LARCHE2
    - GPTF1
16.7 Organigramme for 3D model.

* Organigramme under STEPO.

STEPO -> SCAN2M -> GP_MODEL_[HEAP,HEAP2,STACK] -> GP_MODEL ->
    * CPQ_DRV -> CPQ ->
        - CPQ_GP (see documentation (IDEUL) for details)
        - GPINDDH
        - EC_PHYS_LSLPHY (split ECMWF physics) -> GPADDLSLPHY
        - MF_PHYS_PREP
        - MF_PHYS (unlagged MF physics, organigramme not detailed)
        - CPQ_DIA -> (routines for some diagnostics, organigramme not detailed)
        - CPQ_DSLDIA -> (semi-Lagrangian diagnostics)
    - CPQ_DYN ->
        * LACDYN ->
            - LASURE
            - LASSIE, LANHSI or LANHQESI
            - LAVNTE
            - LATTE_NL
            - LATTEX ->
                * LATEX_TNT or LATEX_DNT
                * VSPLTRANS -> TRIDIA
                - LATTE_KAPPA -> GP_KAPPA and GP_KAPPAT
                - LATTE_STDDIS -> GP_STDDIS
                - LATTE_BEC
                - LATTES -> VERDISINT -> VERINT
                - LAVB0
                - LANHSI or LANHQESI, and LANHSIB for NH and LGWADV=T
            * VDIFLCZ (see documentation (IDEUL) for details)
    - CPQ_END -> (see documentation (IDEUL) for details)
    * CALL_SL_[HEAP,HEAP2,STACK] -> CALL_SL ->
        - SLCOMM and SLEXTPOL (global model) or ESLEXTPOL (LAM model)
        - LAPINEA ->
            * LARMES (ELARMES in LAM models) for case LSLDP_RK=F ->
                - LAISMDA
                - LARCINA ->
                    - LARCH (ELARCH in LAM models)
                    - LASCAX (ELASCAX in LAM models) -> LASCAXCLA, LASCAXCL0, LASCAXVINTW
                    - LAITL, LAITRI or LAITRI_WENO
                - LAISMDG
            * LARMES_RK (expected ELARMES_RK in LAM models) for case LSLDP_RK=T ->
                - LARCH_INCR (expected ELARCH_INCR in LAM models)
                - LARCINA ->
                    - LASCAX (ELASCAX in LAM models) -> LASCAXCLA, LASCAXCL0, LASCAXVINTW
                    - LAITL, LAITRI or LAITRI_WENO
                - LARCH ->
                    - LARCH + LASCAX (ELARCH + ELASCAX in LAM models) -> LASCAXCLA, LASCAXCL0, LASCAXVINTW
                * LARCHHA ->
                    - LARCH + LASCAX (ELARCH + ELASCAX in LAM models) -> LASCAXCLA, LASCAXCL0, LASCAXVINTW
                - SLCOMM2A and SLEXTPOL (global model) or ESLEXTPOL (LAM model) if on-demand SL comm. asked for
        - LAPINEB ->
            * LARCHANB ->
                - LAITRE_GMV -> LAITRI
                - LAITRE_GPL -> LAITRI, LAITHVT, LAITVSPCMQ, LAQMLIMITER, LAITRIQMD
                - LAITL
                - LAIDI
            * LARCHNHB ->
                - LAITRE_GMV -> LAITRI
                - LAITL
            * some GP... and GNH... routines.
            * LARCH
            * LACONE
            * VERDISINT -> VERINT
        * GNHGW2SVD for NH and LGWADV=T
        * CPQLAG (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 (LASC&WTL) 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 \texttt{NTRSLTYPE}. The ADTL-trajectory is stored in the array \texttt{TRAJEC%SLAG}. For \texttt{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.
- \texttt{LATTEXTL} and \texttt{LATESTTL}: quantities \( P(X)T_{15} \) are not computed, excepted in the case \texttt{L2TLFF}=.T. where \texttt{PUT15} and \texttt{PVT15} are computed. PB1...-buffers are directly recovered by a memory transfer under \texttt{CPGTL} if \texttt{NTRSLTYPE}=2.
- \texttt{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 \texttt{LARCHETL} two sets of coordinates (\texttt{PLON,PLAT}) and (\texttt{PLON5,PLAT5}) and two sets of (\texttt{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 LASC&WTL because both sets of weights are needed in the linear tangent versions of the interpolation routines.
- \texttt{LAPINEBTL}: computation of \( P(X)T_1 \) needs both the arrays \( Z(X)95 \) and \( Z(X)9 \).
- \texttt{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 \texttt{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)
        * GPMPPFC5, GP_SPV (for trajectory)
        * some GP.. routines computing intermediate grid-point quantities for trajectories.
    - CPG_GP_TL ->
      * GPTF2
      * GPMPPFC (for model variables)
        * some GP..TL routines computing intermediate grid-point quantities for model variables.
        * GPTINSL
      - MF_PHYSTL (TL of simplified non lagged physics)
      - CPG_DYN_TL ->
        * LACDYNTL ->
          - LASURE
          - LASSIETL
          - LAVENTTL
          - LATTEXTL -> LATTEX_DNT
          - LATTE KAPPA TL -> GP_KAPPA TL and GP_KAPPA TL
          - LATTESTL -> VERINT
          - LAVABOTL
        * VDIFLCZTL (organigramme not detailed)
      - CPG_END_TL ->
        * GPMPPFC (for model variables)
        * GPMPPFC5 (for trajectory)
    - Deallocations
  * CALL_SL_TL ->
    - SLCOMM and SLEXTPOL (DM code only)
      - LAPINEAS ->
        * LARMES5 (ELARMES5 in LAM models) if LSDLDP_RK=F -> LARCINA ->
          - LARCHETL (ELARCHETL in LAM models)
          - LASCAW (ELASCAW in LAM models) -> LASCAW_CLA, LASCAW_CL0, LASCAW_VINTW
          - LAITLI, LAITRI or LAITRI_WENO
        * LARMES_RK5 (expected ELARMES_RK5 in LAM models) if LSDLDP_RK=T ->
          - LARCHETL (ELARCHETL in LAM models)
          - LASCAW (ELASCAW in LAM models) -> LASCAW_CLA, LASCAW_CL0, LASCAW_VINTW
          - LAITLI, LAITRI or LAITRI_WENO
        * LARCINA ->
          - LARCHETL (ELARCHETL in LAM models)
          - LASCAW (ELASCAW in LAM models) -> LASCAW_CLA, LASCAW_CL0, LASCAW_VINTW
        - LAPINEATL ->
          * LARMESTL (ELARMESTL in LAM models) if LSDLDP_RK=F -> LARCINATL ->
            - LARCHETL (ELARCHETL in LAM models)
            - LASCAWTL (ELASCAWTL in LAM models) -> LASCAW_CLA_TL, LASCAW_CL0_TL, LASCAW_VINTW_TL
            - LAITLTL, LAITRITL or LAITRITL_WENO_TL
          * LARMES_RK_TL (ELARMES_RK_TL in LAM models) if LSDLDP_RK=T ->
            - LARCHETL (ELARCHETL in LAM models)
            - LASCAWTL (ELASCAWTL in LAM models) -> LASCAW_CLA_TL, LASCAW_CL0_TL, LASCAW_VINTW_TL
            - LAITLTL, LAITRITL or LAITRITL_WENO_TL
        * LARCINATL ->
          - LARCHETL (ELARCHETL in LAM models)
          - LASCAWTL (ELASCAWTL in LAM models) -> LASCAW_CLA_TL, LASCAW_CL0_TL, LASCAW_VINTW_TL
          - LAITLTL, LAITRITL or LAITRITL_WENO_TL
        * LARCINATL ->
          - LARCHETL (ELARCHETL in LAM models)
          - LASCAWTL (ELASCAWTL in LAM models) -> LASCAW_CLA_TL, LASCAW_CL0_TL, LASCAW_VINTW_TL
          - SLCOMM2A and SLEXTPOL (DM code only and on-demand SL communications asked for)
        - LAPINEBTL ->
          * LARICINATL ->
            - LAITRE_GHY_TL -> LAITRITL
            - LAITRE_GFL_TL -> LAITRITL, LAITRIQMSDTL
            - LAITLTL
            - LADIDITL
          * VERINT
          - LACONETL
          * LARCHETL (ELARCHETL in LAM models)
          * VERINT
          * EC_PHYSTL (organigramme not detailed)
          * CPOLAGTL (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, LARCHÈ 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.
- LARCHÈAD: 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.
Organigramme of AD code under SCAN2MAD.

**SCANN2MAD -> GP_MODULE_AD ->**

* CP5
  + CPGLAD (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)
  + LAPIINEA5 ->
    * LARMES5 (ELARMES in LAM models) if LSLDP_RK=F -> LARCINA ->
      - LARCHE (ELARCHE in LAM models)
      - LASCW (ELASCW in LAM models) -> LASCW_CLA, LASCW_CLO, LASCW_VINTW
      - LAITLI, LAITRI or LAITRI_WENO
    * LARMES_RK5 (expected ELARMES_RK5 in LAM models) if LSLDP_RK=T ->
      - LARCHE_INCR (expected ELARCHE_INCR in LAM models)
      - LARCINA ->
        - LASCW (ELASCW in LAM models) -> LASCW_CLA, LASCW_CLO, LASCW_VINTW
        - LAITLI, LAITRI or LAITRI_WENO
    * LARMES_RK_AD if LSLDP_RK=F -> LARCINAAD ->
      - some LAI...._INIT routines (LSLADREP=T)
      - LAITLIAD
      - LAITRIAD
      - LAITRE_GFL_AD -> LAITRIAD, LAITRIQ3DAD
      - LAIN3DAD
    + LARIHENAD ->
      * LARCIINBAD ->
        - LASCWAD -> LASCW_CLA_AD, LASCW_CLO_AD, LASCW_VINTW_AD
        - LARCHEAD
      + LARMESAD if LSLDP_RK<F -> LARCINAAD ->
        - some LAI...._INIT routines (LSLADREP=T)
        - LAITLIAD, LAITRIAD or LAITRI_WENO_AD
        - LASCWAD -> LASCW_CLA_AD, LASCW_CLO_AD, LASCW_VINTW_AD
        - LARCHEAD
      + LARMES_RK_AD if LSLDP_RK=T ->
        - LARCHE_INCR
        - LARCIINBAD ->
          - some LAI...._INIT routines (LSLADREP=T)
          - LAITLIAD, LAITRIAD or LAITRI_WENO_AD
          - LASCWAD -> LASCW_CLA_AD, LASCW_CLO_AD, LASCW_VINTW_AD
        - LARCHEAD
    - SLEXTPOLAD (case LSLADREP=F)
    - SLCOMM2 (case LSLADREP=F + LSLONDEM=T)
    - SLCOMM (case LSLADREP=F + LSLONDEM=F)
  + CP5_DRV_AD -> CPAD ->
    - Allocations
    - CP5_GP (for trajectory) ->
      * some routines reading trajectory (RDPHTRAJ)
    + GPMPFC5, GP_SPV (for trajectory)
    + some GP.. routines computing intermediate grid-point quantities for trajectories.
    - CP5_END_AD ->
    + GPMPFC
    - CP5_ZERG_AD
    - CP5_DYN_AD ->
      * VDIFLZAD (organigramme not detailed)
    + LACDYNAAD ->
      - LASURE
      - LATESAD -> VERINTAD
      - LATE_KAPPAAD -> GP_KAPPAAD and GP_KAPPATAD
      - LATEKAD -> LATEX_DNT_AD
      - LAVENTAD
      - LASSIEAD
    - MF_PHYSAD (organigramme not detailed)
    - CP5_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.
- LACDYNSHWT: quantities $P(X)T15$ are not computed. One only computes $PB1...$-buffers $P(X)L95$ and $P(X)L05$ if NTRSLETYP is zero or 1; otherwise the $PB1...$-buffers are directly recovered by a call to RDSLTRAJ2 under CPG2TL if NTRSLETYP=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” $Z = X Y_5 + X_5 Y$ and the “ADTL-trajectory code” $Z_5 = X_5 Y_5$). 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 LASCWTL 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 $Z(X)9$.  
- LARCHETL: 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 NTRSLETYP = 1)
    - GPT2
    - LACDYNSHWT
    - RDSLTRAJ2 (if NTRSLETYP = 2)
  * CPG2LAGTL ->
    - LADINETL ->
      * LARMES2TL -> LARCHE2TL -> (see below)
      * LAINOR2TL -> LARCHE2TL -> (see below)
    * LARCHETL
    * LACONETL
    - GPT1

Organigramme below LARCHE2TL:

LARCHE2TL |
  * LARCHETL + LASCWTL -> LASCW_CLA_TL, LASCW_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 $X = X + t MY$ 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_CLG
 * LAIDDI or LAIDL
```

Organigramme below LARCIN2AD:

```
LARCIN2AD ->
 * LARCHE2 + LASCAW -> LASCAW_CLA, LASCAW_CLG
 * LAIDDIAD or LAIDLAD
 * LASCAWAD + LARCHE2AD -> LASCAW_CLA_AD, LASCAW_CLG_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 NFLDSL1: 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.
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, LTDIABL, LHORTURB, LQM, LQM3D, LQML3D, LSLHD, LCOMAD, LHV, LVSPLIP.
- **YOMARG** (0-level control, former command line) and **YOMCT0** (0-level control): 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): some of these variables are in namelist **NAMDYNA**.
- **YOMDYN** (adiabatic dynamics: second part): some of these variables are in namelist **NAMDYNA**.
– LADV, LIMP, L2TLFF, RW2TLFF (treatment of Coriolis term).
– RCMSLP0 (Eulerian treatment of orography).
– All variables, the name starts by LSLHD or SLHD or ZSLHD (SLHD = semi-lagrangian horizontal diffusion): for example SLHDA (κa), SLHDA0 or SLHDA0T (κa0), SLHDB or SLHDBT (κb), SLHDD0 or SLHDD0T, SLHDDIV (Cslhddiv), SLHDRATDDIV (Cslhdratddiv), SLHDOH, ZSLHDP1 (κa), ZSLHDP3 (DEFO), LSLSHEAT, LSLSHPONGE, etc.
– LFINDVSEP, NVSEPL, NVSEPC (optimisations for semi-Lagrangian adjoint code).
– LSETTLSVF, RSETTLF, LSETTFSTAT, RPRES, SETTLLSVF, REPDSYN, NFLEVSF.

Some of these variables are in namelist **NAMdyn**.

- **YOMMpo** and **YOMMP** (distributed memory environment, see documentation (IDDM) for more details).
- **YOMSPLPHY** (interactions between SL scheme and split ECMWF physics).
- **YOMSPLREP** (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**.
- **YOMHSMLME**: for high-order horizontal meridian interpolations.
- **YOMVSPLIP**: for spline vertical interpolations.
- **YOMVSELETA**: for high-order vertical interpolations.
- **YOMACK**.
- **YOMVAR**: LSLADREP, LVECADIN.

- Modules for trajectory stored for TL and AD models, in particular **TRAJ_SEMILAG_MOD**:
- Modules for geometry:
  - **YOMGSGEOM**: (geographic space grid-point geometry)
  - **YOMCSGEO**: (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).
  - **YOMVVF1** (namelist variables for vertical geometry): variables are in namelist **NAMVV1**.
  - **YEMGEO** (LAM model geometry): some variables are in namelist **NEMGEO**.
  - **YEMLBCE** (LAM model geometry for LBC).
  - **TYPE_GEOMETRY**, **TYPE_SPGEOM**.

- Modules for GMV dataflow:
  - **YOMGMV** (grid-point arrays for GMV fields).
  - **TYPE_GMV** (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 from 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.


20.2 Internal notes, workshop proceedings.

• 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.
• 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.
• (IDBAS) Yessad, K., 2019: Basics about ARPEGE/IFS, ALADIN and AROME in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• (IDSI) Yessad, K., 2019: Semi-implicit spectral computations in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• (IDDH) Yessad, K., 2019: Horizontal diffusion in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• (IDTS) Yessad, K., 2019: Spectral transforms in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• (IDDM) Yessad, K., 2019: Distributed memory features in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• (IDCUL) Yessad, K., 2019: Integration of the model equations, and Eulerian dynamics, in the cycle 46t1r1 of ARPEGE/IFS (internal note).
• 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 $\frac{dX}{dt}$ 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 $\frac{dX}{dt}$ uses formula:
    \[ \frac{dX}{dt} = \left[ 0.5X^+_F + 0.5X^-_O - X^- \right] / [\Delta t] \]
  - These calculations are done under LATTEX (and LAPINEB for interpolations).
  - $\frac{dX}{dt}$ 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 $X^o$ and $X^-$ by calling GPXX.

- Three-time level semi-Lagrangian scheme for ND4SYS=2:
  - Calculation of $\frac{dX}{dt}$ uses formula:
    \[ \frac{dX}{dt} = \left[ X^{+-}_{F} - X^-_O \right] / [0.5\Delta t] + \left[ X^o - X^+_O \right] / [\Delta t] \]
  - For advective part $[X^+_F - X^-_O]$, calculations are done under LATTEX (and LAPINEB for interpolations).
  - $X^{+-}_F$ is computed using a mix of $t$ quantities (horizontal derivatives) and provisional $t + \Delta t$ variables in CPGLAG.
  - $\frac{dX}{dt}$ is used to compute $dd_4/dt$ from $dd/dt$.
  - $X^{+-}_F$ is passed to spectral transforms, it provides $X^o$ and $\nabla X^o$ of the following timestep. $X^o$ becomes the $X^-$ of the following timestep.
  - There is no recalculation of $X^o$ 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 $\frac{dX}{dt}$ 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 $\frac{dX}{dt}$ uses formula:
    \[ \frac{dX}{dt} = \left[ X^+_M - X^-_O \right] / [0.5\Delta t] \]
  - $X^+_M$ 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).
  - $\frac{dX}{dt}$ 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 $X^o$ by calling GPXX.

- Two-time level semi-Lagrangian scheme for ND4SYS=2:
  - Calculation of $\frac{dX}{dt}$ uses formula:
    \[ \frac{dX}{dt} = \left[ X^{+-}_F - X^-_O \right] / [0.5\Delta t] + \left[ X^o - X^+_O \right] / [0.5\Delta t] \]
– For advective part \( [X_F^o - X_O^o] \), calculations are done under \texttt{LATTEX} (and \texttt{LAPINEB} for interpolations).

– \( X_F^{+o} \) is computed using a mix of \( t \) quantities (horizontal derivatives) and provisional \( t + \Delta t \) variables in \texttt{CPGLAG}.

– \( dX/dt \) is used to compute \( dd_4/dt \) from \( dd/dt \).

– \( X_F^{+o} \) is passed to spectral transforms, it provides \( X^o \) and \( \nabla X^o \) of the following timestep.

– There is no recalculation of \( X^o \) under \texttt{CPG_GP_NHEE}.

If there is a predictor-corrector scheme activated:

- for \texttt{ND4SYS}=1 all the RHS of \( dX/dt \) is iterated.

- for \texttt{ND4SYS}=2 only the advective part containing \( [X_F^o - X_O^o] \) is iterated; the non advective part containing \( [X_F^{+o} - X_F^o] \) is computed and added at the last corrector step only.

Both options \texttt{ND4SYS}=1 and \texttt{ND4SYS}=2 are coded. \texttt{ND4SYS}=2 is more stable, especially at fine resolutions. \texttt{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.
    - \( X(t) \) 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 \( l-1 \) is stored in index \( jlev=l \) 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 particular 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 \( V(t+\Delta t) \) which is computed in LAPINEB; it does not contain the semi-implicit correction nor horizontal diffusion; surface value of \( V(t+\Delta t) \) is assumed to be equal to full-level \( l=NFLEVG \) value of \( V(t+\Delta t) \).
    - Part 2.3.5 of LAPINEB converts \( w(t+\Delta t) \) into \( d(t+\Delta t) \) by calling GNHGW2SVD. Values of \( T, p, \Pi \) 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 \( \text{NVDVAR}=4 \).
  – 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 \( \text{NVDVAR} \), 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 \( X(t + \Delta t) \) too if \( \text{NVDVAR}=4 \).

• Under (E)TRANSDIRH:
  – \( d(t + \Delta t) \) or \( d_4(t + \Delta t) \) is transformed into spectral space.
  – \( X(t + \Delta t) \) is transformed into spectral space if \( \text{NVDVAR}=4 \).
  – \( w \) 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 \( \text{NVDVAR}=4 \).
  – \( w \) never appears.

• Remark: LGWADV=T combined with LSETTLS=T may generate more complicated code with keys like LSLINL2 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.