SPECTRAL TRANSFORMS IN THE CYCLE 46T1 OF ARPEGE/IFS.

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

December 19, 2018

Abstract:
ARPEGE/IFS, ALADIN and AROME are spectral models, so this documentation has for purpose to describe the spectral transforms done. The present note has for aim to give a brief summary of the spectral method (how to compute Legendre polynomials, spectral transforms) and to describe parts of the code performing spectral transforms (organigramme).

Résumé:
ARPEGE/IFS, ALADIN et AROME sont des modèles spectraux, en conséquence cette documentation a pour but de décrire les transformées spectrales nécessaires pour passer de l’espace spectral vers l’espace point de grille. On y fait un bref rappel de la méthode spectrale (comment calculer les polynômes de Legendre et les transformées spectrales). On y décrit les parties de code faisant des transformées spectrales, avec fourniture de quelques organigrammes.
## Contents

1 Introduction. 3

2 Theoretical aspects. 4
   2.1 Notations. 4
   2.2 Spectral representation of a scalar field. 4
   2.3 Truncation. 4
   2.4 Horizontal derivatives. 5
   2.5 Spectral relationship between (divergence, vorticity) and (velocity potential, stream function). 5
   2.6 Spectral relationship giving wind components once known (velocity potential, stream function). 6
   2.7 Spectral relationship giving (divergence, vorticity) once known wind components. 6
   2.8 Steps involved in a direct and in an inverse spectral transformation for scalar fields. 6
   2.9 Relationship between dimension of spectral space and grid point space for Gaussian grid. 8

3 Set-up routines. 10
   3.1 Simplified call-tree. 10
   3.2 Computations done by these routines and remarks. 10

4 Spectral transforms routines: the general routines (E)DIR_TRANS and (E)INV_TRANS. 12
   4.1 Range of use. 12
   4.2 Call tree for direct code. 12
   4.3 Call tree for adjoint code. 14

5 Spectral transforms routines for general application. 15
   5.1 Spectral transforms routines for scalar variables: SPERE and REESPE. 15
   5.2 Wind components: routines SPEUV and UVSPE. 15

6 Spectral transforms used in the model under routine STEPO and for specific applications. 16
   6.1 Call tree under STEPO. 16
   6.2 Spectral transforms necessary for model integration. 16
      6.2.1 3D hydrostatic and non-hydrostatic models. 17
      6.2.2 Shallow-water 2D model. 17
   6.3 Spectral transforms necessary for particular applications. 17

7 Some distributed memory features. 18

8 Specific variables in modules and namelists. 19
   8.1 Specific variables of the “trans” and “etrans” libraries. 19
   8.2 Specific variables of the “arpifs” library. 19
   8.3 Additional remarks. 20

9 Bibliography. 21
   9.1 Publications. 21
   9.2 Some internal notes and other ARPEGE notes. 21
1 Introduction.

ARPEGE/IFS, ALADIN and AROME are spectral models. One part of computations is done in spectral space (semi-implicit scheme, horizontal diffusion scheme for model integration), the other part in grid-point space on a grid defined by a Gaussian quadrature if global model, by the plane projection if LAM model. So it is necessary to perform spectral transforms from spectral space to grid-point space or vice-versa. The present note has for aim to give a brief summary of the spectral method and to describe parts of the code performing spectral transforms. Spectral method is not described with too many details: for more details one can report to (Rochas and Courtier, 1992, note ARPEGE nr 30, in French) or to (Temperton, 1991).

For a global spectral model, spectral transforms are a combination of a Legendre transform and a Fourier transform. A spectral limited-area model like ALADIN uses a double Fourier representation for spectral fields.

* Fast Legendre transforms (FLT): They are now implemented and use specific routines. They are not extensively documented there; one can refer to section 2 of (Wedi et al., 2013).
2 Theoretical aspects.

2.1 Notations.

- $a$ is the Earth mean radius. Equations are written for models with thin layer approximation and deep-layer effects are ignored.
- $\Theta$ is the latitude on the Gaussian collocation grid.
- $\mu = \sin \Theta$.
- $\Lambda$ is the longitude on the Gaussian collocation grid.
- $n$ is the total wave number, $m$ is the zonal wave number.
- $M$ is the mapping factor.
- $V$ is the horizontal geographical wind. Its zonal component is $U$. Reduced components $(U', V')$ are linked to unreduced components $(U, V)$ by relationships $U = M * U'$, $V = M * V'$.
- $D$ is the unreduced divergence of horizontal wind, $D'$ is the reduced divergence: $D = M^2 * D'$.
- $\zeta$ is the unreduced vorticity of horizontal wind, $\zeta'$ is the reduced vorticity: $\zeta = M^2 * \zeta'$.
- $\nabla^2$ is the horizontal unreduced Laplacian operator and $\nabla'^2$ its reduced counterpart: $\nabla'^2 = M^2 * \nabla^2$.
- $x$ and $y$ are coordinates on the plane projection (LAM models).

2.2 Spectral representation of a scalar field.

* Spherical geometry.
A field $f$ can be decomposed into its spectral components according to the following expression:

$$f(\Lambda, \Theta) = \sum_{m=-\infty}^{m=\infty} \sum_{n=0}^{n=\infty} P_{n,m}(\mu) \exp(i m \Lambda)$$

(1)

where $P_{n,m}(\mu)$ are spectral coefficients. $P_{n,m}(\mu)$ are the first kind normalised polynomials of Legendre and are normalised by the following relationship:

$$\frac{1}{2} \int_{-1}^{1} (P_{n,m}(\mu))^2 d\mu = 1$$

(2)

That yields the following expression for $P_{n,m}(\mu)$, if $m$ is a positive integer:

$$P_{n,m}(\mu) = \frac{1}{2\pi n!} \left[ \frac{(n-m)!}{(n+m)!} (1-\mu^2)^{m/2} \frac{d^{n+m}}{d\mu^{n+m}} (\mu^2-1)^n \right]$$

(3)

For a real scalar field, $P_{n,-m}(\mu)$ is set equal to $P_{n,m}(\mu)$, $f_{n,-m} = \overline{f}_{n,m}$, so only the real and imaginary parts of $f_{n,m}$ for $m \geq 0$ are computed and stored. The algorithm of computation of $P_{n,m}(\mu)$ is described in appendix 3.

* LAM models with plane projection.
Coordinates $x$ and $y$ are assumed to vary between 0 and 1 in the limited-area domain. A bi-periodic field $f$ (assumed to match $f(1,0) = f(0,0)$ and $f(0,1) = f(0,0)$) can be decomposed into its spectral components according to the following expression:

$$f(x,y) = \sum_{m=-\infty}^{m=\infty} \sum_{n=0}^{n=\infty} f_{n,m}(x) \exp(i 2\pi m x) \exp(i 2\pi n y)$$

(4)

where $f_{n,m}$ are spectral coefficients.

2.3 Truncation.

* Spherical geometry.
In practical the expression of $f$ is limited to a finite set of harmonics corresponding to $0 \leq n \leq N_s$ and $-N_s \leq m \leq n$. This defines a triangular truncation $N_s$. Fourier representation of the transmission coefficients for simplified physics is a very low truncation $N_s$ always lower than the model truncation $N_s$. From now on formulae and algorithms will be written with truncation $N_s$. So equation (1) becomes:

$$f(\Lambda, \Theta) = \sum_{m=-N_s}^{m=N_s} \sum_{n=0}^{n=N_s} f_{n,m}(\mu) \exp(i m \Lambda)$$

(5)
Due to the aforementioned properties of $P_{(n,m)}$, expression of $f$ becomes for a real scalar field:

$$f(\Lambda, \Theta) = \sum_{m=0}^{N_n} \sum_{n=|m|}^{N_m} f_{(n,m)} P_{(n,m)}(\mu) \exp(i m \Lambda)$$  \hspace{1cm} (6)

Some other types of truncations exist (for example rhomboidal truncation). Such truncations will not be detailed there.

* **LAM models with plane projection.** Elliptic truncation: in practical the expression of $f$ is limited to a finite set of harmonics corresponding to: \((n/N_s)^2 + (m/N_{ms})^2 \leq 1\).

It is also possible (not coded) to use a rectangular truncation: \(n \leq N_s\) and \(m \leq N_{ms}\).

**2.4 Horizontal derivatives.**

* **Meridian derivative relative to Gaussian latitude \(\Theta\) in spherical geometry:** for a variable $f$, meridian derivative is discretised in spectral space by the following formula:

$$\left( \cos \Theta \frac{\partial f}{\partial \Theta} \right)_{(n,m)} = -(n-1)e_{(n,m)} f_{(n-1,m)} + (n+1)e_{(n+1,m)} f_{(n+1,m)}$$  \hspace{1cm} (7)

where $e_{(0,0)} = 0$ and:

$$e_{(n,m)} = \sqrt{\frac{n^2 - m^2}{4n^2 - 1}}$$  \hspace{1cm} (8)

* **Zonal derivative relative to Gaussian longitude \(\Lambda\) in spherical geometry:** for a variable $f$, zonal derivative is discretised in spectral space by the following formula:

$$\left( \frac{\partial f}{\partial \Lambda} \right)_{(n,m)} = im f_{(n,m)}$$  \hspace{1cm} (9)

Such a derivation can be done on Fourier coefficients by a multiplication by $im$.

* **Derivatives for LAM models:** example is given for zonal derivative; for a variable $f$, zonal derivative is discretised in spectral space by the following formula:

$$\left( \frac{\partial f}{\partial x} \right)_{(n,m)} = i2\pi mf_{(n,m)}$$  \hspace{1cm} (10)

* **Horizontal laplacian in the computational sphere for spherical geometry:** for a variable $f$, expression of the laplacian is:

$$\left( \nabla' \nabla' f \right)_{(n,m)} = \left( \frac{1}{a^2 \cos \Theta} \frac{\partial^2 f}{\partial \Lambda^2} \right)_{(n,m)} + \left( \frac{1}{a^2 \cos \Theta} \frac{\partial}{\partial \Theta} \left( \frac{\partial f}{\partial \Theta} \right) \right)_{(n,m)}$$  \hspace{1cm} (11)

One can show (for more details see appendix 1) that its discretisation writes:

$$\left( \nabla' \nabla' f \right)_{(n,m)} = -\frac{n(n+1)}{a^2} f_{(n,m)}$$  \hspace{1cm} (12)

* **Horizontal laplacian in the computational sphere for LAM models:** for a variable $f$, expression of the laplacian is:

$$\left( \nabla' \nabla' f \right)_{(n,m)} = \left( \frac{\partial^2 f}{\partial x^2} \right)_{(n,m)} + \left( \frac{\partial^2 f}{\partial y^2} \right)_{(n,m)}$$  \hspace{1cm} (13)

**2.5 Spectral relationship between (divergence, vorticity) and (velocity potential, stream function).**

- Relationships between: reduced divergence and velocity potential $\chi$; reduced vorticity and stream function $\psi$:

$$D' = \nabla' \chi ; \quad \zeta' = \nabla' \psi$$

- These relationships can be rewritten, for each complex spectral component (spherical geometry):

$$D_{(n,m)} = -\frac{n(n+1)}{a^2} \chi_{(n,m)} ; \quad \zeta_{(n,m)} = -\frac{n(n+1)}{a^2} \psi_{(n,m)}$$
2.6 Spectral relationship giving wind components once known (velocity potential, stream function).

- Global models: relationship between $U'$, $V'$, $\psi$ and $\chi$:

$$
(U' a \cos \Theta) = \frac{\partial \chi}{\partial \Lambda} - \cos \Theta \frac{\partial \psi}{\partial \Theta} ;
(V' a \cos \Theta) = \frac{\partial \psi}{\partial \Lambda} + \cos \Theta \frac{\partial \chi}{\partial \Theta}
$$

the spectral discretisation is:

$$
(U' a \cos \Theta)_{(n,m)} = im\chi_{(n,m)} + (n-1)e_{(n,m)}\psi_{(n-1,m)} - (n+2)e_{(n+1,m)}\psi_{(n+1,m)}
$$

$$
(V' a \cos \Theta)_{(n,m)} = im\psi_{(n,m)} - (n-1)e_{(n,m)}\chi_{(n-1,m)} + (n+2)e_{(n+1,m)}\chi_{(n+1,m)}
$$

- LAM models: relationship between $U'$, $V'$, $\psi$ and $\chi$:

$$
U' = U'_{moy} + \frac{\partial \chi}{\partial x} - \frac{\partial \psi}{\partial y} ;
V' = V'_{moy} + \frac{\partial \psi}{\partial x} + \frac{\partial \chi}{\partial y}
$$

We need a separate storage of $U'_{moy}$ and $V'_{moy}$ in spectral space.

2.7 Spectral relationship giving (divergence, vorticity) once known wind components.

* Global models:

- Relationship between $D'$, $U'$ and $V'$:

$$
D' = \frac{1}{a \cos \Theta} \left( \frac{\partial U'}{\partial \Lambda} + \cos \Theta \frac{\partial (V' \cos \Theta)}{\partial \Theta} \right)
$$

which can be rewritten:

$$
D' = \frac{1}{a^2 \cos^2 \Theta} \left( \frac{\partial (U' a \cos \Theta)}{\partial \Lambda} + \cos \Theta \frac{\partial (V' a \cos \Theta)}{\partial \Theta} \right)
$$

- Relationship between $\zeta'$, $U'$ and $V'$:

$$
\zeta' = \frac{1}{a \cos \Theta} \left( \frac{\partial V'}{\partial \Lambda} - \frac{\partial (U' \cos \Theta)}{\partial \Theta} \right)
$$

which can be rewritten:

$$
\zeta' = \frac{1}{a^2 \cos^2 \Theta} \left( \frac{\partial (V' a \cos \Theta)}{\partial \Lambda} - \cos \Theta \frac{\partial (U' a \cos \Theta)}{\partial \Theta} \right)
$$

- Spectral discretisations of (17) and (19) allow to retrieve easily spectral components of $D' a^2 \cos^2 \Theta$ and $\zeta' a^2 \cos^2 \Theta$, but not directly spectral components of $D'$ and $\zeta'$ (requiring inversion of a pentadiagonal matrix). In practical algorithm involved to retrieve spectral coefficients of $D'$ and $\zeta'$ once knowing values of wind components is slightly different (requiring a division by $a \cos \Theta$ in Fourier space), and is described in detail in (Temperton, 1991) (see equations (2.16), (2.17), (2.21) to (2.27) of Temperton’s paper).

* LAM models: relationship between $D'$, $\zeta'$, $U'$ and $V'$ writes:

$$
D' = \frac{\partial U'}{\partial x} + \frac{\partial V'}{\partial y} ;
\zeta' = \frac{\partial V'}{\partial x} - \frac{\partial U'}{\partial y}
$$

2.8 Steps involved in a direct and in an inverse spectral transformation for scalar fields.

* Transformation from spectral to grid point space for a scalar field or a set of scalar fields in a global model. Transformation from spectral to grid point space needs:
• An inverse Legendre transform, for each zonal wave number \(m\). The following quantity \(F_m(\mu)\) is computed for each latitude \(\Theta\):

\[
F_m(\mu) = \sum_{n=-N_s}^{n=N_s} f(n,m)(\mu) P(n,m)(\mu)
\]  

(20)

One transforms real fields, so \(F_{-m}(\mu) = F_m(\mu)\) and only \(F_m(\mu)\) for \(m \geq 0\) is computed.

• An inverse Fourier transform, for each latitude \(\Theta\):

\[
f(\Lambda, \Theta) = \sum_{m=-N(\Theta)}^{m=N(\Theta)} F_m(\mu) \exp (im\Lambda)
\]

(21)

One transforms real fields, so formula (21) becomes:

\[
f(\Lambda, \Theta) = F_0(\mu) + 2 \sum_{m=1}^{m=N(\Theta)} \left( R(F_m(\mu)) \cos (m\Lambda) - I(F_m(\mu)) \sin (m\Lambda) \right)
\]

(22)

\(F_0(\mu)\) is the average of \(f\) on latitude \(\Theta\) and is real; \(N(\Theta)\) is not always equal to \(N_s\) (due to the use of a reduced Gaussian grid). For latitude \(\Theta\) one obtains \(nlon(\Theta)\) grid-point values.

* Transformation from spectral to grid point space for a scalar field or a set of scalar fields in a LAM model. Transformation from spectral to grid point space needs:

• An inverse meridian Fourier transform.

• An inverse zonal Fourier transform.

* Transformation from grid point to spectral space for global models. Transformation from grid point to spectral space needs:

• A direct Fourier transform, for each latitude \(\Theta\):

\[
F_m(\mu) = \frac{1}{nlon(\Theta)} \sum_{jlon=1}^{jlon=nlon(\Theta)} f(\Lambda(jlon), \Theta) \exp (-im\Lambda(jlon))
\]

(23)

One transforms real fields, so formula (23) becomes for non-zero \(m\):

\[
R(F_m(\mu)) = \frac{1}{nlon(\Theta)} \sum_{jlon=1}^{jlon=nlon(\Theta)} f(\Lambda(jlon), \Theta) \cos (m\Lambda(jlon))
\]

(24)

\[
I(F_m(\mu)) = - \frac{1}{nlon(\Theta)} \sum_{jlon=1}^{jlon=nlon(\Theta)} f(\Lambda(jlon), \Theta) \sin (m\Lambda(jlon))
\]

(25)

and for \(m=0\):

\[
R(F_0(\mu)) = \frac{1}{nlon(\Theta)} \sum_{jlon=1}^{jlon=nlon(\Theta)} f(\Lambda(jlon), \Theta)
\]

(26)

\[
I(F_0(\mu)) = 0
\]

(27)

\(nlon(\Theta)\) is the number of grid-points for latitude \(\Theta\). Only \(N(\Theta)\) Fourier coefficients are kept. \(N(\Theta)\) is not always equal to \(N_s\).

• A direct Legendre transform, for each zonal wave number \(m\).

\[
f(n,m) = \sum_{jgl=lat1(m)}^{jgl=lat2(m)} F_m(\mu(jgl)) P(n,m)(\mu(jgl))
\]

(28)

\(lat1\) and \(lat2\) satisfy to \(\Theta(lat2) = -\Theta(lat1)\). \(lat1(m)\) is the first northern latitude where the wave number \(m\) is represented and is not always equal to 1.

* Transformation from grid point to spectral space for LAM models. Transformation from grid point to spectral space needs:

• A direct zonal Fourier transform.

• A direct meridian Fourier transform.
* Relationship between $N$ and $\text{lat}1$ (global models):

$$N(\text{lat}1(m) - 1) < m \leq N(\text{lat}(m))$$  \hspace{1cm} (29)

* Use of symmetry properties of the polynomials of Legendre (global models): Due to relation (30) scalar fields to be transformed are split into symmetric and antisymmetric parts $f_{\text{sym}}$ and $f_{\text{ant}}$ matching $f_{\text{sym}}(\Lambda, -\Theta) = f_{\text{sym}}(\Lambda, \Theta)$ and $f_{\text{ant}}(\Lambda, -\Theta) = -f_{\text{ant}}(\Lambda, \Theta)$. Spectral expression of $f_{\text{sym}}$ (resp. $f_{\text{ant}}$) include all coefficients with even (resp. odd) value of $m + n$. Legendre transforms are done on $f_{\text{sym}}$ and $f_{\text{ant}}$ and scalar products defined by equations (20) and (28) can reduce to latitudes $\text{lat}1$ to $\text{ndynh}$.

2.9 Relationship between dimension of spectral space and grid point space for Gaussian grid.

* Quadratic grid, linear grid and cubic grids in global models:

  - Spectral space is defined by a triangular truncation $N_s$. Grid point space has $\text{ndglg}$ latitudes and a maximum number of longitudes equal to $\text{ndlon}$. $\text{ndlon}$ and $\text{ndglg}$ are always even integers: if $\text{ndlon}$ is a multiple of 4, $\text{ndglg} = \text{ndlon}/2$; if $\text{ndlon}$ is not a multiple of 4, $\text{ndglg} = \text{ndlon}/2 + 1$. The latitudes of the Gaussian grid are computed by an algorithm detailed in appendix 2.
  - For a quadratic Gaussian grid, there is a relationship between these parameters to avoid aliasing on quadratic terms. If the stretching coefficient $c$ is equal to 1 (resp. > 1) $N_s$ is the biggest integer which matches the relationship $3* N_s \leq (\text{ndlon} - 1)$ (resp. $3* N_s \leq \min(2* \text{ndglg} - 3, \text{ndlon} - 1)$).
  - In a semi-Lagrangian scheme the advective quadratic terms disappear, so it is possible to use a smaller grid-point space. The most frequently grid used is the “linear grid”: if the stretching coefficient $c$ is equal to 1 (resp. > 1) $N_s$ is the biggest integer which matches the relationship $2* N_s \leq (\text{ndlon} - 1)$ (resp. $2* N_s \leq \min(2* \text{ndglg} - 3, \text{ndlon} - 1)$).
  - Recently, the concept of cubic grid has been introduced. If the stretching coefficient $c$ is equal to 1 (resp. > 1) $N_s$ is the biggest integer which matches the relationship $4* N_s \leq (\text{ndlon} - 1)$ (resp. $4* N_s \leq \min(2* \text{ndglg} - 3, \text{ndlon} - 1)$).
  - We generally use a reduced Gaussian grid; the number of grid-points is progressively reduced as we go towards computational poles.

* Quadratic grid, linear grid and cubic grids in LAM models: similar notions exist in LAM models. For example:

For a quadratic grid:

- $N_s$ is the biggest integer which matches the relationship $3* N_s \leq (\text{ndglg} - 1)$.
- $N_{\text{mas}}$ is the biggest integer which matches the relationship $3* N_{\text{mas}} \leq (\text{ndlon} - 1)$.

For a linear grid:

- $N_s$ is the biggest integer which matches the relationship $2* N_s \leq (\text{ndglg} - 1)$.
- $N_{\text{mas}}$ is the biggest integer which matches the relationship $2* N_{\text{mas}} \leq (\text{ndlon} - 1)$.

For a cubic grid:

- $N_s$ is the biggest integer which matches the relationship $4* N_s \leq (\text{ndglg} - 1)$.
- $N_{\text{mas}}$ is the biggest integer which matches the relationship $4* N_{\text{mas}} \leq (\text{ndlon} - 1)$.

* Admissible dimensions for spherical geometry: If FFT992 is used, the codes of FFT (fast Fourier transforms) used allows to use integers $\text{ndlon}$ which can write as $2^{1+p_2} \times 3^{p_3} \times 5^{p_5}$. That limits the possibility of choosing the dimensions in a discontinuous subset of truncations and dimensions for Gaussian grid. In the following table one can find the admissible values for $(\text{ndlon}, \text{ndglg})$ for $\text{ndglg}$ between 32 and 8000, and the corresponding truncations for a quadratic grid and a linear grid. This constraint does not exist with FFTW.

* Admissible dimensions for LAM models: If FFT992 is used, the codes of FFT (fast Fourier transforms) used allows to use integers $\text{ndlon}$ and $\text{ndglg}$ which can write as $2^{1+p_2} \times 3^{p_3} \times 5^{p_5}$. This constraint does not exist with FFTW.
3  Set-up routines.

3.1  Simplified call-tree.

The call-tree given below is not an extensive one but a simplified one to show the main routines. Mention “lctnd” means “lower call-tree not described”. Abbreviation FLT means “Fast Legendre transforms”. Left part is for global model, right part is for LAM model.

SU0YOMA -> | SU0YOMA ->
- SUTRANS0 -> | - SUTRANS0 ->
  * SETUP_TRANS0 -> SUMP_TRANS0 -> (lctnd) | * SETUP_TRANS0 -> SUMP_TRANS0 -> (lctnd)
- SUGEOMETRY -> | - SUGEOMETRY ->
  * SUTRANS -> | * SUTRANS ->
    - SETUP_TRANS -> | - ESETUP_TRANS ->
      * SETUP_Resol | * ESET_RESOL
      * SETUP_DIMS | * ESETUP_DIMS
    * SUMP_TRANS_PRELEG -> (lctnd) | * SUEMP_TRANS_PRELEG
  * SULEG -> | * ESETUP_GEOM
    - PRE_SULEG | |
    - INI_POL | |
    - SUGAW -> (lctnd) | |
    - SUPOL | |
    - SUTRLE -> (lctnd) | |
    - SETUP_GEOM | |
    - SULPOLF, CONSTRUCT_BUTTERFLY (FLT) |
    * SUMP_TRANS -> | * SUEMP_TRANS ->
      - SUMPLAT -> (lctnd) | - SUMPLAT -> (lctnd)
      - SUMPLATF -> (lctnd) | - SUMPLATF -> (lctnd)
      - SUSTAONL -> (lctnd) | - SUSTAONL -> (lctnd)
    * SUFFT -> | * SUFFT ->
      - SET99 (use of FFT992) | - SET99 (use of FFT992)
      - INIT_PLANS_FFTW (use of FFTW) | - INIT_PLANS_FFTW (use of FFTW)
  * SULEGA | * SUELEGA

3.2  Computations done by these routines and remarks.

Upper levels of call-tree:

- **SU(E)TRANS0**: computes not geometry-dependent variables useful for the transform package. There are for example some dimension variables, some distributed memory variables.
- **SUGEOMETRY**: geometry set-up; it includes in particular some variables computed by the transform package (for example Gaussian latitudes).
- **SU(E)TRANS**: computes geometry-dependent variables useful for the transform package.
- **SULEGA**: gets some variables from transform package (these variables depend on Gaussian latitudes), copy them in “arpifs” structures.
- **SUELEGA**: is just for security but computed variables inside should not be used in LAM models.

Lower levels of call-tree for Legendre transforms (global models only): the most important features are the Gaussian grid geometry (for example Gaussian latitudes, Gaussian weights) and the Legendre polynomials.

- The main routine is **SULEG**.
- **SUGAW** computes the Gaussian latitudes (Θ, in attribute RLATIG) and the Gaussian weights (in attribute RW), and some additional variables like cos Θ (in attribute RSQM2) and sin Θ (in attribute RMU).
- **SUPOL** computes the polynomials of Legendre (attribute RPNM). For each Gaussian latitude, RPNM is an array with a structure identical to spectral arrays. RPNM is stored only for the Northern Hemisphere (of computational sphere) due to the following property:
  \[ P_{(n,m)}(-\mu) = (-1)^{m+n} P_{(n,m)}(\mu) \]  
  \[ P_{(n,m)}(\mu) \]  
- **SUPOLF** is an alternate version of **SUPOL** (for fast Legendre transforms).

10
Lower levels of call-tree for Fourier transforms:

- The main routine is SUFFT.
- For “FFT992” Fourier transforms, one needs to compute quantities TRIGS and NFAX necessary to use routine FFT992. These two arrays are computed in routines SUFFT and SUEFFT (note that a call to FFT992 reinitialises them).
- For “FFTW” Fourier transforms, the set-up is done in INIT_PLANS_FFTW.

Lower levels of call-tree for transpositions and distributed memory features:

- SUTRLE: does transpositions for Legendre polynomials, because the distribution required for Legendre polynomials calculation is not the same as the distribution required for Legendre polynomials use. Calculation of Legendre polynomials requires a distribution on latitudes (for a given latitude all the spectral coefficients are treated by the same processor). Use of Legendre polynomials requires a distribution on zonal wavenumbers (for a given zonal wavenumber all the latitudes are treated by the same processor).
- SU(E)MP_TRANS: sets-up geometry-dependent distributed environment for the transform package.

Duplicate version of variables between “arpifs” and “trans”/“etrans”:

- Most of the variables computed under SUTRANS have a “duplicate version” in the “arpifs” library, because they are used in other parts of the ARPEGE/IFS code. The “trans” version (in modules TPM...) is used in the spectral transforms, the “arpifs” version is used elsewhere.
- Most of the variables computed under SUETRANS have a “duplicate version” in the “arpifs” library, because they are used in other parts of the ARPEGE/ALADIN code. The “etrans” version (in modules TPMALD...) is used in the spectral transforms, the “arpifs” version is used elsewhere.
4 Spectral transforms routines: the general routines (E)DIR_TRANS and (E)INV_TRANS.

4.1 Range of use.

They can be used for a set of fields (model fields), but they use optional arguments, so they can also be used for only one separate field. DIR_TRANS (EDIR_TRANS for LAM models) does direct transforms (from grid-point space to spectral space), INV_TRANS (EINV_TRANS for LAM models) does inverse transforms (from spectral space to grid-point space). These routines have adjoint versions (E)DIR_TRANSAD and (E)INV_TRANSAD. The tangent linear code is identical to the direct code. These routines and the routines called under them are in the “trans” library for spherical geometry, “etrans” library for LAM models.

4.2 Call tree for direct code.
* Call tree of INV_TRANS and EINV_TRANS:

INV_TRANS ->
  * SET_RESOL
  * INV_TRANS_CTL ->
    - SHUFFLE (case NPRTRV>1)
    - FIELD_SPLIT
    - LTINV_CTL ->
      * LTINV ->
        - PREPSNM
        - VDFUV
        - SPNSDE
        - LEINV -> DGMEM
          - MULT_BUTM (FLT)
          - ASREIB
          - FSPGL_INT
          * TRMTOL -> (MPL routines)
          - FTINV_CTL ->
            - EFTINV_CTL ->
              - FOURIER_IN
              - EFSC
              - FTINV ->
                - FFT992 or EXEC_FFTW
                - TRLTG0 -> (MPL routines)

EINV_TRANS ->
  * ESET_RESOL
  * EINV_TRANS_CTL ->
    - SHUFFLE (case NPRTRV>1)
    - FIELD_SPLIT
    - ELTINV_CTL ->
      * ELTINV ->
  * ESET_RESOL
  * EINV_TRANS_CTL ->
    - SHUFFLE (case NPRTRV>1)
    - FIELD_SPLIT
    - FTDIR_CTL ->
      * TRGTOL -> (MPL routines)
      * EXTPER
      - AUX_PROC
      - FTDIR ->
        - FFT992 or EXEC_FFTW
        - EFTDIR_CTL ->
          - TRLTOM -> (MPL routines)
        - LTDIR_CTL ->
          * TRLTOM -> (MPL routines)
        - PREFI2 -> PRFI2B
        - LFDOU2
        - LEDIR -> DGMEM
          - MULT_BUTM (FLT)
          - PREPSNM
          - UVTVD
          - UPDSP -> UPDSPB

* Call tree of DIR_TRANS and EDIR_TRANS:

DIR_TRANS ->
  * SET_RESOL
  * DIR_TRANS_CTL ->
    - SHUFFLE (case NPRTRV>1)
    - FIELD_SPLIT
    - FTDIR_CTL ->
      * TRLTG0 -> (MPL routines)
  * FTDIR ->
    - FFT992 or EXEC_FFTW
  * FOURIER_OUT
  - LTDIR_CTL ->
    * TRLTOM -> (MPL routines)
  * LTDIR ->
    - PREFI2 -> PRFI2B
    - LFDOU2
    - LEDIR -> DGMEM
      - MULT_BUTM (FLT)
      - PREPSNM
      - UVTVD
      - UPDSP -> UPDSPB

EDIR_TRANS ->
  * ESET_RESOL
  * EDIR_TRANS_CTL ->
    - SHUFFLE (case NPRTRV>1)
    - FIELD_SPLIT
    - EFTDIR_CTL ->
      - TRLTG0 -> (MPL routines)
      - AUX_PROC
      * ELTDIR ->
        - EPRFI2B -> EPRFI2B
        - EXTPER
        - EEDIR -> FFT992 or EXEC_EFFTW
        - EUVTVD_COMM
        - EUPDSPB -> EUPDSPB

* Work done by routines called under INV_TRANS_CTL and DIR_TRANS_CTL:
  - Legendre transforms:
    - LTINV_CTL: upper control routine for inverse Legendre transforms.
- **LTDIR_CTL**: upper control routine for direct Legendre transforms.
- **LTINV**: lower control routine for inverse Legendre transforms.
- **LTDIR**: lower control routine for direct Legendre transforms.
- **PREPSNM**: stores useful values of $e_{(n,m)}$ (in array REPSNM) for a given zonal wave number $m$.
- **PRF11B**: stores useful spectral coefficients of the fields to be transformed for a given zonal wave number $m$ and latitudes lat$(m)$ to nghost in a work array.
- **VDTUV**: computation of $(\cos \Theta U', \cos \Theta V')$ in spectral space knowing $(D', \zeta')$ when required.
- **SPNSDE**: computation of meridian derivatives when required.
- **LEINV** performs two inverse Legendre transforms, the first one for $f_{\text{sym}}$, the second one for $f_{\text{ant}}$, that gives the Fourier coefficients for symmetrical and antisymmetrical parts.
- **DGEMM**: product matrix by matrix.
- **MULT_BUTM**: product matrix by matrix with the butterfly method.
- **ASRE1B** recombines symmetrical and antisymmetrical parts to compute Fourier coefficients $F_m(\mu)$.

$$F_m(\mu) = F_{\text{sym}} m(\mu) + F_{\text{ant}} m(\mu) ; \quad F_m(-\mu) = F_{\text{sym}} m(\mu) - F_{\text{ant}} m(\mu)$$

- **FSPGL_INT**: interface for calculations done in Fourier space.
- **PRF12B**: splits Fourier coefficients into symmetrical and antisymmetrical parts.
- **PRF12**: interface for **PRF12B** (one call to **PRF12B** for scalars, two for vectors).
- **LDFOU2**: division by $\cos \Theta$ in Fourier space for wind.
- **LEDIR** performs two direct Legendre transforms, the first one for $F_{\text{sym}} m(\mu)$, the second one for $F_{\text{ant}} m(\mu)$.
- **UVTVD**: computation of $(D', \zeta')$ in spectral space knowing $(\frac{\mu'}{\mu - \mu'}, \frac{\mu'}{\mu + \mu'})$.
- **UPDSPB**: final memory transfer in spectral array containing all coefficients.
- **UPDSP**: interface routine for **UPDSPB**.

* Transpositions between Legendre transforms space and Fourier space:
  - **TRLTOM**: transposition routine for distributed memory. Transpose Fourier buffer data from partitioning over latitudes (necessary for direct Fourier transforms) to partitioning over wave numbers (necessary for direct Legendre transforms).
  - **TRMTOL**: does the inverse transposition of **TRLTOM**.

* Fourier transforms:
  - **FTINV_CTL**: upper control routine for inverse Fourier transforms.
  - **FTDIR_CTL**: upper control routine for direct Fourier transforms.
  - **FTINV**: lower control routine for inverse Fourier transforms.
  - **FTDIR**: lower control routine for direct Fourier transforms.
  - **FOURIER_IN**: separates the real and imaginary parts of the Fourier coefficients.
  - **FOURIER_OUT**: recombines the real and imaginary parts of the Fourier coefficients.
  - **FSC**: division by $\cos \Theta$ in Fourier space for wind, and first order derivatives.
  - **FFT992** performs Fourier transforms for “FFT992” package.
  - **EXEC_FFTW** is called instead of **FFT992** if “FFTW” package is used.

* Transpositions between Fourier transforms space and grid-point space:
  - **TRLGTO**: transposition routine for distributed memory. Transpose grid-point data from partitioning over sets of NGPTOT grid-points containing complete columns (all the NFLEVG model layers) and incomplete latitudes to partitioning over complete latitudes (a processor contains a subset of complete latitudes) and layers if NPRTRV$>1$ (a processor contains NFLEVL layers).
  - **TRLTOG**: does the inverse transposition of **TRLGTO**.

* Other routines:
  - **SHUFFLE**: re-shuffle fields for load balancing if NPRTRV$>1$. Note that the relative order of the local spectral fields has to be maintained.
  - **FIELD_SPLIT**: split fields for case NPRTRV$>1$; simple memory transfer if NPRTRV$=1$.

* Work done by routines called under **EINV_TRANS_CTL** and **EDIR_TRANS_CTL**:

We can notice that we generally have the same routines as for “trans” routines, with an additional “E” at the beginning of their names when a specific “etrans” routine is required: for example **ELEDIR** instead of **LEDIR**.

Note the following specifics:
- **ELEDIR** and **LEINV** do meridian Fourier transforms and call **FFT992**.
- **AUX_PROC** is called to do periodicisation of auxiliary fields.
- **EXTPER** is called to do periodisation when there is no extension zone.
4.3 Call tree for adjoint code.

Call tree is given for INV_TRANSAD and DIR_TRANSAD; their LAM counterparts EINV_TRANSAD and EDIR_TRANSAD also exist (organigramme not detailed).

* Call tree of INV_TRANSAD:

INV_TRANSAD ->
* SET_RESOL
* INV_TRANS_CTLAD ->
  - SHUFFLE (case NPRTRV>1)
  - FIELD_SPLIT
  - FTINV_CTLAD ->
    * TRGTOI -> (MPL routines)
    * FTINVAD ->
      - FFT992 or EXEC_FFTW
    * FSCAD
    * FOURIER_INAD
  - LTINV_CTLAD ->
    * TRLTOM -> (MPL routines)
    * LTINVD ->
      - PREPSNM
      - ASRE1BAD
      - LEINVD -> DGEMM
        MULT_BUTM (FLT)
      - VDTUVAD
      - SPNSDEAD
      - PRFI1BAD

* Call tree of DIR_TRANSAD:

DIR_TRANSAD ->
* SET_RESOL
* DIR_TRANS_CTLAD ->
  - SHUFFLE (case NPRTRV>1)
  - FIELD_SPLIT
  - LTDIR_CTLAD ->
    * LTDIRAD ->
      - UPDSPAD -> UPDSPBAD
      - PREPSNM
      - UVTVDAD
      - LEDIRAD -> DGEMM
        MULT_BUTM (FLT)
      - LDFOU2AD
      - PRF12AD -> PRF12BAD
    * TRMTOL -> (MPL routines)
  - FTDIR_CTLAD ->
    * FOURIER_OUTAD
    * FTDIRAD ->
      - FFT992 or EXEC_FFTW
    * TRLTDG -> (MPL routines)
5 Spectral transforms routines for general application.

5.1 Spectral transforms routines for scalar variables: SPEREE and REESPE.

REESPE does direct transforms (from grid-point space to spectral space). SPEREE does inverse transforms (from spectral space to grid-point space). SPEREE calls INV_TRANS; REESPE calls DIR_TRANS; the adjoint SPERAD of SPEREE calls REESPE. These routines perform transforms for scalar fields according to the algorithm described in subsection (2.8).

These routines have LAM counterparts: EREESPE and ESPEREE (direct code and TL code), ESPERAD (adjoint code).

5.2 Wind components: routines SPEUV and UVSPE.

* Transformation from spectral to grid point space. $D'$ is the reduced divergence, $\zeta'$ is the reduced vorticity. $\chi$ is the velocity potential, $\psi$ is the stream function. $U'$ and $V'$ are the reduced wind components. $D'$ and $\zeta'$ are available in spectral space. Transformation from spectral to grid point space needs:
  - A transformation $(\chi,\psi) \rightarrow (D',\zeta')$ if required.
  - Computation of $(a \cos \Theta U', a \cos \Theta V')$ in spectral space knowing $(D',\zeta')$, following formulae (14) and (15).
  - Inverse Legendre transform on $a \cos \Theta U'$ and $a \cos \Theta V'$.
  - In Fourier space, division by $a \cos \Theta$ for each latitude: yields $U'$ and $V'$ in this space.
  - Inverse Fourier transform on $U'$ and $V'$ for each latitude.

* Transformation from grid point to spectral space. Transformation from grid point to spectral space needs:
  - Direct Fourier transform on $U'$ and $V'$ for each latitude.
  - Computation of $\left( \frac{U'}{a \cos \Theta}, \frac{V'}{a \cos \Theta} \right)$ in Fourier space for each latitude.
  - Direct Legendre transform on $\frac{U'}{a \cos \Theta}$ and $\frac{V'}{a \cos \Theta}$. Spectral coefficients obtained after this transform correspond to the $U_{(n,m)}$ and $V_{(n,m)}$ defined in equations (2.24) and (2.25) of (Temperton, 1991).
  - Computation of $(D',\zeta')$ in spectral space knowing $\left( \frac{U'}{a \cos \Theta}, \frac{V'}{a \cos \Theta} \right)$, following formulae (2.26) and (2.27) of (Temperton, 1991).
  - A transformation $(D',\zeta') \rightarrow (\chi,\psi)$ if required.

* Routines performing transformations: SPEUV performs a transformation from spectral space (input variables are $(D',\zeta')$ or $(\chi,\psi)$) to grid-point space (output variables are $U'$ and $V'$). UVSPE performs a transformation from grid-point space (input variables are $U'$ and $V'$) to spectral space (output variables are $(D',\zeta')$ if KOUTP=1 or $(\chi,\psi)$ if KOUTP=2). SPEUV calls INV_TRANS; UVSPE calls DIR_TRANS.

* LAM model versions:
  - ESPEUV is the LAM counterpart of SPEUV.
  - There are also EUVGEVOVD (retrieve divergence and vorticity from wind components in spectral space) and EVDUVGEOD (retrieve wind components from divergence and vorticity in spectral space).
6 Spectral transforms used in the model under routine STEPO and for specific applications.

6.1 Call tree under STEPO.

In model integration, routines TRANSINVH and TRANSDIRH play the same part as SPERE/SPEUV, REESPE/UVSPE, but they deal with a pre-determined set of variables. Horizontal derivatives of these variables have generally to be computed during the inverse transform. Meridian derivatives are computed after converting \((D', \zeta')\) into \((a \cos \Theta U', a \cos \Theta V')\) and before inverse Legendre transforms in spectral space (call to routine SPNSDE). These spectral transforms routines are also used for the spectral fit of FULL-POS.

* General organigramme under STEPO: one starts from routine STEPO which controls one time-step.
  * control/STEPO →
    * (E)TRANSINVH (see below the detailed organigramme)
    * SCAN2M (grid point computations)
    * OBSV (for assimilation part computing \(J_o\))
    * Diagnostics
    * (E)TRANSDIRH (see below the detailed organigramme)
    * Spectral computations

FULLPOS has its own version of STEPO (STEPO_FPOS) calling TRANSINV_FPOS and TRANSDIR_FPOS.

* Call tree under (E)TRANSINVH:
  
  (E)TRANSINVH → (E)TRANSINV_MDL → (E)INV_TRANS (model variables)

* Call tree under (E)TRANSDIRH:

  (E)TRANSDIRH → (E)TRANSDIR_MDL → (E)DIR_TRANS (model variables)

* Tangent linear code: The tangent linear code of (E)TRANSINVH (called under STEPOTL) is (E)TRANSINVH itself. The tangent linear code of (E)TRANSDIRH (called under STEPOTL) is (E)TRANSDIRH itself.

* Adjoint code: The adjoint code of (E)TRANSINVH (called under STEPOAD) is (E)TRANSINVHAD. The adjoint code of (E)TRANSDIRH (called under STEPOAD) is (E)TRANSDIRHAD. Call trees are:

  (E)TRANSINVHAD → (E)TRANSINV_MDLAD → (E)INV_TRANSAD

  and:

  (E)TRANSDIRHAD → (E)TRANSDIR_MDLAD → (E)DIR_TRANSAD

6.2 Spectral transforms necessary for model integration.

Only model integration is here presented. For other applications where TRANSDIRH or TRANSINVH are used (assimilation, initialisation by DFI), see corresponding documentations. Variables are divided in three classes:

* GMV (3D) variables: the adiabatic part of the RHS of their equation is non-zero and these variables have always a spectral representation (example: wind components, temperature).
* GMVS (2D) variables: the adiabatic part of the RHS of their equation is non-zero and these variables have always a spectral representation (example: logarithm of hydrostatic surface pressure).
* GFL variables: the adiabatic part of the RHS of their equation is zero and these variables can have a spectral representation or not (example: specific humidity, ozone).

For more details about this classification, see documentation (IDEUL).
6.2.1 3D hydrostatic and non-hydrostatic models.

* **Input variables in spectral space to be transformed into grid-point space:** For GMV variables: reduced divergence $D'$, reduced vorticity $\zeta'$, temperature $T$ if $\text{LSPRT}=\text{.F.}$ or $\text{RT}$ if $\text{LSPRT}=\text{.T.}$, additional NH variables for non-hydrostatic models. For GMVS variables: pressure surface variable $\log \Pi_s$ ($\Pi_s$ is the surface pressure). All the GFL variables which have a spectral representation.

* **Output variables to be obtained in grid-point space:** Variable $X$, meridian derivative $\frac{1}{a \cos \Theta} \frac{\partial X}{\partial \Theta}$ and zonal derivative $\frac{1}{a} \frac{\partial X}{\partial \Lambda}$ for the following variables: reduced divergence $D'$, reduced vorticity $\zeta'$, reduced wind components $U'$ and $V'$, temperature $T$ if $\text{LSPRT}=\text{.F.}$ or $\text{RT}$ if $\text{LSPRT}=\text{.T.}$, pressure surface variable $\log \Pi_s$, additional NH variables for non-hydrostatic models, all the GFL variables which have a spectral representation. All horizontal derivatives are required in an Eulerian model but some derivatives are useless in a semi-Lagrangian model.

* **Input variables in grid-point space to be transformed into spectral space:** Provisional $t+\Delta t$ quantities before semi-implicit scheme and horizontal diffusion scheme for GMV variables equations (momentum equation, temperature equation (virtual temperature equation if $\text{LSPRT}=\text{.T.}$), additional NH variables for non-hydrostatic models), GMVS variables equations (continuity equation), GFL equations for GFL variables which have a spectral representation. For momentum equation input variables are reduced wind components, output variables are reduced divergence and vorticity.

* **Additional spectral transforms necessary for iterative centred-implicit schemes:** The iterative part of these schemes do spectral transforms as for the predictive part (transforms are done on the provisional $t+\Delta t$ quantities obtained after each iteration); transforms involves all variables for the centred-implicit scheme coded.

6.2.2 Shallow-water 2D model.

* **Input variables in spectral space to be transformed into grid-point space:** Reduced divergence $D'$, reduced vorticity $\zeta'$, equivalent height $\Phi$.

* **Output variables to be obtained in grid-point space:** Variable $X$, meridian derivative $\frac{1}{a \cos \Theta} \frac{\partial X}{\partial \Theta}$ and zonal derivative $\frac{1}{a} \frac{\partial X}{\partial \Lambda}$ for the following variables: reduced divergence $D'$, reduced vorticity $\zeta'$, reduced wind components $U'$ and $V'$, equivalent height $\Phi$. All horizontal derivatives are required in an Eulerian model but some derivatives are useless in a semi-Lagrangian model.

* **Input variables in grid-point space to be transformed into spectral space:** Provisional $t+\Delta t$ quantities before semi-implicit scheme and horizontal diffusion scheme for momentum equation and continuity equation. For momentum equation input variables are reduced wind components, output variables are reduced divergence and vorticity.

6.3 Spectral transforms necessary for particular applications.

One finds some other \texttt{TRANSDIR...} (which call \texttt{DIR_TRANS}) and \texttt{TRANSINV...} (which call \texttt{INV_TRANS}) routines (resp. \texttt{ETRANSDIR...} and \texttt{ETRANSINV...} routines for LAM models) which are designed for specific applications. For example, for ARPEGE, one finds \texttt{TRANSNRHCONV} and \texttt{TRANSNRHCONV_WAVELET} and \texttt{TRANSINWAVELET}. 

17
7 Some distributed memory features.

* Message passing division into processors: The total number of processors is NPROC. There are two levels of parallelisation. Distribution on these two levels is different in the different spaces of calculations.

* Grid-point computations: The total number of processors involved in the A-level parallelisation is NPRGPNS. The total number of processors involved in the B-level parallelisation is NPRGPEW. The total number of processors in NPROC=NPRGPNS*NPRGPEW. One 2D model field has NGPTOTG points divided into NPRGPNS*NPRGPEW sets of NGPTOT points treated by each processor. NGPTOT may be processor-dependent (with very small variations): the maximum value of NGPTOT is NGPTOTMX. A processor works on complete columns, but latitudes are not always complete (split among several processors). For one given processor, the NGPTOT points are divided into packets of length NPROMA (the useful number of values in each packet is lower or equal than NPROMA). NPROMA is identical for all processors. NPROMA-packet does not always contain a set of complete latitudes. Data reorganisation and transpositions are necessary between the grid-point space (complete columns, incomplete latitudes) and the Fourier space (complete latitudes, possibility to have incomplete columns), using routines TRGTOL and TRLTOG.

* Fast Fourier transforms: NPROC=NPRTRNS*NPRTRV. The total number of processors involved in the A-level parallelisation is NPRTRNS. The total number of processors involved in the B-level parallelisation is NPRTRV. Fourier transforms are done latitude by latitude for each NPROMA-packet. A processor treats a subset of latitudes, one latitude at the time. A processor works on complete latitudes. A processor can treat a subset of the vertical levels (if NPRTRV>1). Data reorganisation and transpositions are necessary in the Fourier space between the zonal wave structure necessary for Legendre transforms, and the latitudinal structure necessary for Fourier transforms (routines TRMTOL and TRLTOM).

* Legendre transforms (North-South Fourier for LAM models): NPROC=NPRTRW*NPRTRV. The total number of processors involved in the A-level parallelisation is NPRTRW. The total number of processors involved in the B-level parallelisation is NPRTRV. Legendre transforms are done zonal wave number by zonal wave number. A processor treats a subset of zonal wave numbers, one zonal wave number at the time. A processor can treat only a subset of the vertical levels (if NPRTRV>1).

* Spectral computations: The total number of processors involved in the A-level parallelisation is NPRTRW. The total number of processors involved in the B-level parallelisation is NPRTRV for horizontal diffusion, NPRTRN for semi-implicit scheme. Spectral computations are done zonal wave number by zonal wave number. A processor treats a subset of zonal wave numbers. For more details see documentations (IDSI) about semi-implicit scheme and (IDDH) about horizontal diffusion scheme.

* Computation of Legendre polynomials: This computation is done latitude by latitude for all the spectral coefficients. Data reorganisation and transpositions (done by SUTRLE) are necessary for Legendre transforms to gather all latitudes and split into zonal wave numbers because for Legendre transforms one processor treats a subset of zonal wave numbers and all latitudes.

* Additional remarks about LEQ_REGIONS. This key controls the way of optimising B-level parallelisation in grid-point space for reduced Gaussian grid (global models). More details are given in documentation (IDEMUL).
8 Specific variables in modules and namelists.

8.1 Specific variables of the “trans” and “etrans” libraries.

Variables (and comments) can be found in the following modules of the “trans” (modules TPM_...) and “etrans” (modules TPMALD_...) libraries, see comments in the code for more details. They may have mirror counterparts in “arpifs/module”.

* “trans”:
  - TPM_CONSTANTS: constants.
  - TPM_CTL.
  - TPM_DIM: dimensions.
  - TPM_DISTR: variables describing distributed memory parallelization.
  - TPM_FFT: setup for fast Fourier transforms. TRIGS contains the sines and cosines which are used in the matricial product which defines the Fourier transform.
  - TPM_FIELDS: Legendre polynomials and some variables linked to the Gaussian grid. For a given total wave number \( j_n \), \( NTN(j_n) \) is the number of available Legendre coefficients \( P(n = j_n, m) \).
  - TPM_FLT: variables for fast Legendre transforms.
  - TPM_GEN: general control variables.
  - TPM_GEOMETRY: data describing Gaussian grid.
  - TPM_POL: encapsulated routines for Legendre polynomials.
  - TPM_TRANS: variables "local" to a specific call to a transform.

* “etrans”:
  - TPMALD_DIM: dimensions.
  - TPMALD_DISTR: variables describing distributed memory parallelization.
  - TPMALD_FFT: setup for fast Fourier transforms. TRIGSE contains the sines and cosines which are used in the matricial product which defines the Fourier transform.
  - TPMALD_FIELDS: contains for example the eigen-values of the inverse Laplace operator.
  - TPMALD_GEO: data describing plane projection grid.
  - TPMALD_TCDIS (useless).

8.2 Specific variables of the “arpifs” library.

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:

- IFS_INIT object:
  - YOMARG (0-level control, former command line) and YOMCT0 (0-level control): for example NCONF, LSPRT. Some of these variables are in namelist NAMCT0.
  - YOMMP0 ("ifs_init" distributed memory environment, see documentation (IDDM) for more details).

- Geometry:
  - YOMDIM, YOMDIMV (geometry dimensioning): most of variables. Some of these variables are in namelist NAMDIM.
  - YOMGEM (computational space grid-point horizontal geometry): all variables. Some of these variables are in namelist NAMGEM.
  - YOMMLAP (constants related to the Laplace space).
  - YOMLEG (description of Legendre polynomials).
  - YOMMP (geometry-dependent distributed memory environment, see documentation (IDDM) for more details).
- **YOMTRANS** (spectral transforms control variables). Some of these variables are in namelist **NAMTRANS** and **NAMTRANS0**.
- **TYPE_GEOMETRY**: geometry (head structure of some geometry modules listed above).

**Other modules:**
- **YOMDIMF** (field dimensioning): most of variables.

### 8.3 Additional remarks.

- In a spectral array \( X \) dimensioned with **NSPEC2**, order of storing spectral coefficients is the following:
  \[
  \Re(X)(n=0,m=0), \\Im(X)(n=0,m=0), \Re(X)(n=1,m=0), \Im(X)(n=1,m=0), \Re(X)(n=2,m=0), \Im(X)(n=2,m=0), \ldots, \Re(X)(n=N_s,m=0), \Im(X)(n=N_s,m=0),
  \Re(X)(n=0,m=1), \Im(X)(n=0,m=1), \ldots, \Re(X)(n=N_s,m=N_s), \Im(X)(n=N_s,m=N_s).
  \]

- In the array **RPNM** containing the Legendre polynomials, order of storing Legendre coefficients for a given Gaussian latitude is the following for \((n,m)\): \((N_s+1,0), (N_s,0), \ldots, (1,0), (0,0), (N_s+1,1), (N_s,1), \ldots, (1,1), \ldots, (N_s+1,N_s), (N_s,N_s), \ldots, (N_s,N_s)\).

- The reader can also look at modules linked to the GMV and GFL structures: modules have names containing root **gfl** or **gmv**. See comments inside these modules and documentation (IDEUL) for more details.
9 Bibliography.
A lot of references are available in note ARPEGE nr 30.

9.1 Publications.

For non-hydrostatic models aspects:

9.2 Some internal notes and other ARPEGE notes.
- (IDBAS) Yessad, K., 2018: Basics about ARPEGE/IFS, ALADIN and AROME in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDEU1) Yessad, K., 2018: Integration of the model equations, and Eulerian dynamics, in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDSL) Yessad, K., 2018: Semi-Lagrangian computations in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDSI) Yessad, K., 2018: Semi-implicit spectral computations in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDDH) Yessad, K., 2018: Horizontal diffusion in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDDM) Yessad, K., 2018: Distributed memory features in the cycle 46t1 of ARPEGE/IFS (internal note).

21
Appendix 1: Expression of the laplacian in spectral space (spherical geometry).

Equation:

\[
\nabla^2 f \equiv (a \cos \Theta)^2 \frac{\partial^2 f}{\partial \Lambda^2} = \frac{1}{(a \cos \Theta)^2} \frac{\partial^2 f}{\partial \Lambda^2} + \left( \frac{1}{a \cos \Theta} \frac{\partial (\cos \Theta \frac{\partial f}{\partial \Theta})}{\partial \Theta} \right)_{(n,m)}
\]

(31)

can be rewritten:

\[
(a \cos \Theta)^2 \nabla^2 f \equiv \frac{\partial^2 f}{\partial \Lambda^2} + \left( \cos \Theta \frac{\partial (\cos \Theta \frac{\partial f}{\partial \Theta})}{\partial \Theta} \right)_{(n,m)}
\]

(32)

One assumes that this equation can be rewritten as:

\[
\nabla^2 f \equiv Z_{(n,m)} f_{(n,m)}
\]

(33)

Combining equations (32) and (33) yields:

\[
(a \cos \Theta)^2 Z f \equiv \frac{\partial^2 f}{\partial \Lambda^2} + \left( \cos \Theta \frac{\partial (\cos \Theta \frac{\partial f}{\partial \Theta})}{\partial \Theta} \right)_{(n,m)}
\]

(34)

* LHS of equation (34): One uses the following relationships:

\[
(a \cos \Theta)^2 = a^2 (1 - \mu^2)
\]

(35)

and:

\[
[\mu X]_{(n,m)} = e_{(n,m)} X_{(n-1,m)} + e_{(n+1,m)} X_{(n+1,m)}
\]

(36)

Applying \( \mu \) to equation (36) yields:

\[
[\mu^2 X]_{(n,m)} = e_{(n,m)} \left( e_{(n-1,m)} X_{(n-2,m)} + e_{(n+1,m)} X_{(n+2,m)} \right) + e_{(n+1,m)} \left( e_{(n+1,m)} X_{(n+1,m)} + e_{(n+2,m)} X_{(n+3,m)} \right)
\]

(37)

i.e.

\[
[\mu^2 X]_{(n,m)} = e_{(n,m)} e_{(n-1,m)} X_{(n-2,m)} + \left( e_{(n,m)}^2 + e_{(n+1,m)}^2 \right) X_{(n,m)} + e_{(n+2,m)} X_{(n+3,m)}
\]

(38)

Introducing the results of equations (35) and (38) in the LHS of equation (34) yields:

\[
(a \cos \Theta)^2 Z f \equiv \left( a^2 (1 - \mu^2) Z f \right)_{(n,m)} = -a^2 e_{(n,m)} e_{(n-1,m)} Z f_{(n-2,m)} - a^2 \left( 1 + e_{(n,m)}^2 + e_{(n+1,m)}^2 \right) Z f_{(n,m)} - a^2 e_{(n+1,m)} e_{(n+2,m)} Z f_{(n+2,m)}
\]

(39)

* RHS of equation (34): Equation (9) yields:

\[
\frac{\partial^2 f}{\partial \Lambda^2} = -m^2 f_{(n,m)}
\]

(40)

Equation (7) yields:

\[
\cos \Theta \frac{\partial \left( \cos \Theta \frac{\partial f}{\partial \Theta} \right)}{\partial \Theta} = -(n-1) e_{(n,m)} \left( -e_{(n-1,m)} f_{(n-2,m)} + (n+1) e_{(n,m)} f_{(n,m)} \right) + (n+2) e_{(n+1,m)} \left( -n e_{(n+1,m)} f_{(n+3,m)} + (n+3) e_{(n+2,m)} f_{(n+2,m)} \right)
\]

(41)

i.e.

\[
\cos \Theta \frac{\partial \left( \cos \Theta \frac{\partial f}{\partial \Theta} \right)}{\partial \Theta} = e_{(n,m)} e_{(n-1,m)} (n-1) f_{(n-2,m)} +
\]

\[
\left( -(n-1)(n+1) e_{(n,m)}^2 - n(n+2) e_{(n+1,m)}^2 \right) f_{(n,m)} + e_{(n+1,m)} e_{(n+2,m)} (n+2)(n+3) f_{(n+2,m)}
\]

(42)

The sum of equations (40) and (42) yield:

\[
\frac{\partial^2 f}{\partial \Lambda^2} + \cos \Theta \frac{\partial \left( \cos \Theta \frac{\partial f}{\partial \Theta} \right)}{\partial \Theta} = \left( -\frac{m^2}{n^2} - \frac{(n-1)}{n} e_{(n,m)}^2 - \frac{(n+2)}{n+1} e_{(n+1,m)}^2 \right) n(n+1) f_{(n,m)} + e_{(n+1,m)} e_{(n+2,m)} (n+2)(n+3) f_{(n+2,m)}
\]

(43)


*Identity of the LHS and the RHS:* After some strenuous calculations which are not detailed in this documentation and using the expression of $e_{(n,m)}$ (see formula (8)), one can show that the two expressions:

$$-1+e_{(n,m)}^2+e_{(n+1,m)}^2$$

and:

$$-\frac{m^2}{n(n+1)}-\frac{n-1}{n}e_{(n,m)}^2-\frac{n+2}{n+1}e_{(n+1,m)}^2$$

are equal. So equation (43) can be rewritten:

$$\left(\frac{\partial^2 f}{\partial \Lambda^2}\right)_{(n,m)}+\left(\cos \Theta \frac{\partial \left(\cos \Theta \frac{\partial f}{\partial \Theta}\right)}{\partial \Theta}\right)_{(n,m)}=e_{(n,m)}e_{(n-1,m)}(n-1)(n-2)f_{(n-2,m)}$$

$$+\left(-1+e_{(n,m)}^2+e_{(n+1,m)}^2\right)n(n+1)f_{(n,m)}+e_{(n+1,m)}e_{(n+2,m)}(n+2)(n+3)f_{(n+2,m)}$$

(44)

The identity of the LHS and RHS of equation (34) can be satisfied taking:

$$Z_{(n,m)}=-\frac{n(n+1)}{a^2}$$

(45)

So equation (33) can be rewritten:

$$\left(\nabla^2 f\right)_{(n,m)}=-\frac{n(n+1)}{a^2}f_{(n,m)}$$

(46)
Appendix 2: Computation of Gaussian latitudes and Gaussian weights (spherical geometry).

Appendix 2.1: First method.

Method used until CY35T2.

* Gaussian latitudes: *ndglg* is the number of Gaussian latitudes (for convenience it is assumed to be an even number), *jgl* is the index numbering the latitudes. Θ is the latitude on the computational sphere, $\mu = \sin \Theta$. $P_{(n,m)}$ are the unnormalized Legendre polynomials. The relationship between $P_{(n,m)}$ and $P_{(n,m)}$ is:

\[
P_{(n,m)}(\mu) = \frac{1}{\sqrt{(2n+1) \frac{(n-m)!}{(n+m)!}}} P_{(n,m)}(\mu)
\]

(47)

The Gaussian latitudes $\Theta_{jgl}$ match the following relationship:

\[
P_{(n=ndglg,m=0)}(\mu_{jgl}) = 0
\]

(48)

A first guess approximation for $\Theta_{jgl}$ (which are “nearly” equidistant) yields:

\[
\Theta_{jgl}^{(i=0)} = \frac{\pi}{2} - \frac{4 jgl - 1}{4 ndlg + 2} \pi - \frac{1}{8 ndlg} \cos \left( \frac{4 jgl - 1}{4 ndlg + 2} \pi \right)
\]

(49)

and thus $\mu_{jgl}^{(i=0)} = \sin \Theta_{jgl}^{(i=0)}$. An iterative Newton method is then applied to find the “final” value of $\mu_{jgl}$.

\[
\mu_{jgl}^{(i+1)} = \mu_{jgl}^{(i)} - P_{(n=ndglg,m=0)}(\mu_{jgl}^{(i)}) \frac{d P_{(n=ndglg,m=0)}}{d \mu}(\mu_{jgl}^{(i)}) 
\]

(50)

This algorithm is used to find the Northern hemisphere latitudes (jgl from 1 to ndglg/2). The southern hemisphere latitudes are given by:

\[
\mu_{ndlg+1-jgl} = -\mu_{jgl}
\]

One can see that this algorithm needs the calculation of $P_{(n=ndglg,m=0)}$ and $\frac{d P_{(n=ndglg,m=0)}}{d \mu}$. This calculation is done with some recurrence formulae which will be detailed in appendix 3 for the normalized Legendre polynomials.

The derivatives of the unnormalized Legendre polynomials are given by the following formula, once knowing the unnormalized Legendre polynomials:

\[
\frac{d P_{(n,m)}}{d \mu} = -\frac{n \mu}{1 - \mu^2} P_{(n,m)} + \frac{n + m}{1 - \mu^2} P_{(n-1,m)}
\]

(51)

* Gaussian weights: They are given by the following formula:

\[
\omega_{jgl} = \frac{1 - \mu_{jgl}^2}{ndlg \cdot P_{(n=ndglg,m=0)}(\mu_{jgl})^2}
\]

(52)

Appendix 2.2: Second method.

Method used by the current version of SUGAW. Uses a method described in (Swarztrauber, 2002).
Appendix 3: Computation of the normalized Legendre polynomials (spherical geometry).

Appendix 3.1: First method.

Method used until CY35T2.

* Introduction: The algorithm uses recurrence formulae. There are several possible solutions, the following one is used in the code. The order is the following one:

- \( P_{n=m=0} \); \( P_{n=1,m=0} \); \( P_{n=1,m=1} \).
- \( P_{n,m=0} \) where \( n > 1 \).
- \( P_{n,m=1} \) where \( n > 1 \).
- \( P_{n,m=n} \) where \( n > 2 \).
- The remaining Legendre polynomials in the following order: \( P_{n=3,m=2} \); \( P_{n=4,m=2} \); \( P_{n=4,m=3} \); \( P_{n=5,m=2} \); \( P_{n=5,m=3} \); \( P_{n=5,m=4} \); \( P_{n,m=2} \); \( P_{n,m=3} \); \( P_{n+m=n-1} \); \( P_{n+1,m=n-1} \); \( P_{n+1,m=n} \); etc.

* Wave numbers 0 and 1:

\[
\begin{align*}
P_{n=0,m=0} &= 1 \\
P_{n=1,m=0} &= \sqrt{\delta n} \\
P_{n=1,m=1} &= \sqrt{1.5(1-\mu^2)}
\end{align*}
\]

* Other polynomials for \( m = 0 \): \( P_{n,m=0} \) is computed knowing \( P_{n-1,m=0} \) and \( P_{n-2,m=0} \).

The following recurrence formula is used (formula (1.20) of (Rochas and Courtier, 1992)):

\[
\mu P_{n,m} = (n^2 - m^2)/(4n^2 - 1) P_{n-1,m} + \sqrt{(4(n + 1)^2 - m^2)/(4(n + 1)^2 - 1)} P_{n+1,m} \quad (53)
\]

Changing \( n \) into \( n-1 \) in equation (53), then putting \( P_{n,m} \) in the LHS, yields:

\[
P_{n,m} = \mu \sqrt{(4n^2 - 1)/(n^2 - m^2)} P_{n-1,m} - \sqrt{(n - 1)^2 - m^2)/(4(n - 1)^2 - 1)} \sqrt{4(n^2 - 1)/(n^2 - m^2)} P_{n-2,m} \quad (54)
\]

For \( m = 0 \) equation (54) becomes:

\[
P_{n,m=0} = \mu \sqrt{(4n^2 - 1)/n^2} P_{n-1,m=0} - \sqrt{n - 1)^2 - 1}/\sqrt{4(n - 1)^2 - 1)} \sqrt{4(n^2 - 1)/n^2} P_{n-2,m=0} \quad (55)
\]

* Other polynomials for \( m = 1 \): \( P_{n,m=1} \) is computed knowing \( P_{n,m=0} \) and \( P_{n-1,m=0} \).

The following recurrence formula is used (formula (1.22) of (Rochas and Courtier, 1992)):

\[
\sqrt{1 - \mu^2} P_{n,m} = \sqrt{[2(n+1)(n+m-1)]/[2(n-1)(n+m)]} P_{n-1,m-1} + \sqrt{(n-m+1)/(n+m)} P_{n,m-1} \quad (56)
\]

For \( m = 1 \) equation (56) becomes:

\[
\sqrt{1 - \mu^2} P_{n,m=1} = \sqrt{[2(n+1)]/[2(n-1)(n+1)]} P_{n-1,m=0} + \sqrt{n/(n+1)} P_{n,m=0} \quad (57)
\]

* Other polynomials for \( m = n \): The following recurrence formula is used:

\[
P_{n,n} = \sqrt{1 - \mu^2} \sqrt{2n+1}/(2n) P_{n-1,n-1} \quad (58)
\]

* Other polynomials: The following recurrence formula is used (formula (1.31) of (Rochas and Courtier, 1992)):

\[
P_{n,m} = \sqrt{(2n+1)(n+m-1)/(2n-3)(n+m)(n+m-2)} P_{n-2,m-2} - \sqrt{(2n+1)(n+m-1)/(2n-1)(n+m)(n+m-2)} P_{n-1,m-2} + \sqrt{(2n+1)(n-m)/2n-1}(n+m)} P_{n-1,m} \quad (59)
\]

Appendix 3.2: Second method.

Method used by the current version of SUPOL. Uses a method described in (Swarztrauber, 2002) and briefly described in (Wedi et al., 2013).
Appendix 4: Fair numbers between 1 and 50000 writing \(2^{(1+p)q}5^r\).

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>24</th>
<th>30</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>54</td>
<td>60</td>
<td>64</td>
<td>72</td>
<td>80</td>
<td>90</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>108</td>
<td>120</td>
<td>128</td>
<td>144</td>
<td>150</td>
<td>160</td>
<td>162</td>
<td>180</td>
<td>192</td>
<td>200</td>
<td>216</td>
<td>240</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>256</td>
<td>270</td>
<td>288</td>
<td>300</td>
<td>320</td>
<td>324</td>
<td>360</td>
<td>384</td>
<td>400</td>
<td>432</td>
<td>450</td>
<td>480</td>
<td>486</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>512</td>
<td>540</td>
<td>576</td>
<td>600</td>
<td>640</td>
<td>648</td>
<td>720</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>768</td>
<td>800</td>
<td>810</td>
<td>864</td>
<td>900</td>
<td>960</td>
<td>972</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>1024</td>
<td>1080</td>
<td>1152</td>
<td>1200</td>
<td>1250</td>
<td>1280</td>
<td>1296</td>
<td>1350</td>
<td>1440</td>
<td>1458</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>1536</td>
<td>1600</td>
<td>1620</td>
<td>1728</td>
<td>1800</td>
<td>1920</td>
<td>1944</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2048</td>
<td>2160</td>
<td>2250</td>
<td>2304</td>
<td>2400</td>
<td>2430</td>
<td>2500</td>
<td>2560</td>
<td>2592</td>
<td>2700</td>
<td>2880</td>
<td>2916</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>3072</td>
<td>3200</td>
<td>3240</td>
<td>3456</td>
<td>3600</td>
<td>3750</td>
<td>3840</td>
<td>3888</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>4096</td>
<td>4320</td>
<td>4374</td>
<td>4500</td>
<td>4608</td>
<td>4800</td>
<td>4860</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>5120</td>
<td>5184</td>
<td>5400</td>
<td>5760</td>
<td>5832</td>
<td>6000</td>
<td>6144</td>
<td>6250</td>
<td>6400</td>
<td>6480</td>
<td>6750</td>
<td>6912</td>
<td>7200</td>
<td>7290</td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td>7680</td>
<td>7776</td>
<td>8000</td>
<td>8100</td>
<td>8192</td>
<td>8640</td>
<td>8748</td>
<td>9000</td>
<td>9216</td>
<td>9600</td>
<td>9720</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>10240</td>
<td>10368</td>
<td>10800</td>
<td>11250</td>
<td>11520</td>
<td>11664</td>
<td>12000</td>
<td>12150</td>
<td>12288</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12500</td>
<td>12800</td>
<td>12960</td>
<td>13122</td>
<td>13500</td>
<td>13824</td>
<td>14400</td>
<td>14580</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>15360</td>
<td>15552</td>
<td>16000</td>
<td>16200</td>
<td>16384</td>
<td>17280</td>
<td>17496</td>
<td>18000</td>
<td>18432</td>
<td>18750</td>
<td>19200</td>
<td>19440</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>20250</td>
<td>20480</td>
<td>20736</td>
<td>21600</td>
<td>21870</td>
<td>22500</td>
<td>23040</td>
<td>23328</td>
<td>24000</td>
<td>24300</td>
<td>24576</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25000</td>
<td>25600</td>
<td>25920</td>
<td>26244</td>
<td>27000</td>
<td>27648</td>
<td>28800</td>
<td>29160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30000</td>
<td>30720</td>
<td>31104</td>
<td>31250</td>
<td>32000</td>
<td>32400</td>
<td>32768</td>
<td>33750</td>
<td>34560</td>
<td>34992</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36000</td>
<td>36450</td>
<td>36864</td>
<td>37500</td>
<td>38400</td>
<td>38880</td>
<td>39366</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40000</td>
<td>40500</td>
<td>40960</td>
<td>41472</td>
<td>43200</td>
<td>43740</td>
<td>45000</td>
<td>46080</td>
<td>46656</td>
<td>48000</td>
<td>48600</td>
<td>49152</td>
<td>50000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

26