Prague 10.7.–12.8.2006

Study of semi-Lagrangian interpolators in idealized framework

Report from RC LACE scientific stay

Scientific supervisor: Filip Váňa

Ján Mašek

SHMÚ, Jeséniova 17
833 15 Bratislava
Slovak Republic

E-mail: jan.masek@shmu.sk

Typeset by LaTeX

Contents

1 Introduction **3**

2 Checks of model code **3**
2.1 Check of interpolation subroutines 3
2.2 Check of data flow 4

3 Theory **4**
3.1 Global versus local interpolators 4
3.2 Important 4-point interpolators 5
3.3 General 4-point interpolator 6
3.4 Family of cubic 4-point interpolators 7
3.5 Accuracy of cubic 4-point interpolators 9
3.6 Diffusivity of cubic 4-point interpolators 11
3.7 Equivalent diffusion 16

4 Idealized 2D experiments **18**
4.1 Bubble test #1 19
4.2 Bubble test #2 22

5 Conclusions **24**

A Family of cubic 4-point interpolators – case with irregular nodes **25**

1 Introduction

Semi-Lagrangian advection enables having timestep length controlled by accuracy of the scheme, not its stability (as is the case for Eulerian advection). This property, together with small phase errors makes it very attractive for operational NWP models [5].

On the other hand, semi-Lagrangian interpolations are responsible for the scale selective damping, which is not present with Eulerian advection. This feature is exploited by semi-Lagrangian horizontal diffusion (SLHD), which combines high order and low order interpolators in order to achieve right amount of damping [6, 8]. Unfortunately, this approach has undesired side effect, since reducing the accuracy of interpolator increases positive bias evolving in MSLP field. The reason is lack of conservation properties for semi-Lagrangian scheme [7].

As a remedy, replacing of cubic Lagrange interpolator by more accurate natural cubic spline was tried in ALADIN. It indeed reduced MSLP bias for SLHD scheme [7], but it increased temperature and geopotential bias in some parts of troposphere. Since such behaviour was quite unexpected, it was speculated that there might be a bug in model code. Therefore, objectives of this study were:

1. Check ARPEGE/ALADIN implementation of spline interpolator.
2. Analyze relation between accuracy and diffusivity of various interpolators.
3. Try to better understand spline behaviour using suitable 2D academic tests.

2 Checks of model code

All presented work uses ARPEGE/ALADIN cycle 29t2. First task was to check whether spline interpolator is implemented correctly in model. It was splitted in two steps. First step was done outside of model in order to check interpolation subroutine itself. Second step employed integration of vertical plane 2D model in order to verify data flow.

2.1 Check of interpolation subroutines

3D semi-Lagrangian interpolator in ARPEGE/ALADIN uses 32-point stencil which can be obtained from cube $4\times4\times4$ points by removing corners and edges. Interpolation point is always inside central $2\times2\times2$ cube. In order to evaluate given function at interpolation point, three sets of 1D interpolations are performed: first set in x direction, second set in y direction and last interpolation on vertical. 1D interpolations along 4-point sections are high order (cubic Lagrange polynomial or natural cubic spline), remaining interpolations along 2-point sections are low order (linear). In total, each 3D interpolation requires 7 high order and 10 low order 1D interpolations. More details can be found in [9].

32-point 3D interpolations are performed by low level subroutines LAITRI and LAITSP, which employ cubic Lagrange polynomials, resp. natural cubic splines. They were checked using external program, which initialized 32-point stencil with zeros, except from one 4-point section. Interpolation point was placed in central interval of this section, so that result could have been compared against 1D interpolation performed independently. Choosing different 4-point sections enabled to check all 7 high order interpolators employed (4 in x direction, 2 in y direction, one on vertical).

Subroutines LAITRI and LAITSP have also their 2D horizontal versions LAIDDI and LAIDSP, working on 12-point stencil derived from square 4×4 points by omitting

corners. They are used for interpolating fields without vertical dependence. Each 2D interpolation requires 3 high order and 2 low order 1D interpolations (see [9] for details).

Subroutines LAIDDI and LAIDSP were tested in similar manner as their 3D counterparts. Tests were negative, i.e. subroutines LAITRI, LAITSP, LAIDDI and LAIDSP correctly reproduced all independently computed values.[1]

2.2 Check of data flow

Second test verified data flow when spline interpolations are activated (switches LRSPLINE_[x] in namelist NAMDYN[2]). It integrated vertical plane 2D version of ALADIN-NH, using bubble test of Robert [3]. Data flow was checked by simple trick – coding Lagrange cubic polynomials into spline subroutines LAITSP/LAIDSP (the only simplification used was assumption of regularly spaced nodes, after checking that it has very weak impact in bubble test). Results were then compared against those obtained by calling true subroutines LAITRI/LAIDDI. The test was again negative.

Having these results it was concluded that implementation of spline interpolator in ARPEGE/ALADIN is most probably correct[3] and deeper understanding of spline behaviour is needed.

3 Theory

In section 2 it was seen that heart of 2D and 3D semi-Lagrangian interpolators in ARPEGE/ALADIN are 4-point 1D interpolators. It is therefore crucial to understand behaviour of these basic "bricks". Therefore, this section restricts to 1D interpolators and analyzes their behaviour in detail. It first demonstrates change of some properties as one goes from global interpolator to its local 4-point version. After listing important 4-point interpolators it introduces general concept of such interpolator. Then it derives 2-parametric family of cubic interpolators[4] with required properties. Finally, this family is investigated for accuracy and diffusivity.

3.1 Global versus local interpolators

Most of us probably remember from school that high order Lagrange interpolation polynomials tend to oscilate and overshoot, that is why they are seldom used for interpolating functions. On the other hand, behaviour of lower order piecewise polynomials (typically cubic splines) is much better in this respect. Figure 1a compares 9^{th} order Lagrange interpolation polynomial with natural cubic spline for simple jump function. It can be seen that in the vicinity of jump Lagrange polynomial overshoots *more* than spline. Moreover, amplitude of polynomial oscillations increases as one departs from jump, while for spline it decays.

So far, both interpolators were *global*. It means that outside nodes interpolated value $y(x)$ depends on all values $y_1, y_2, \ldots, y_N$. This property is undesirable in NWP models,

[1] Each of these subroutines has also quasi-monotonic version(s), which were not tested.

[2] This was the case in cycle 29t2. In cycle 31, spline interpolations for GFL fields are controlled via attribute LRSPLINE.

[3] One might argue that problem still can be hidden in x direction, since vertical plane 2D model is coded in yz plane.

[4] More precisely, *at most* cubic interpolators.

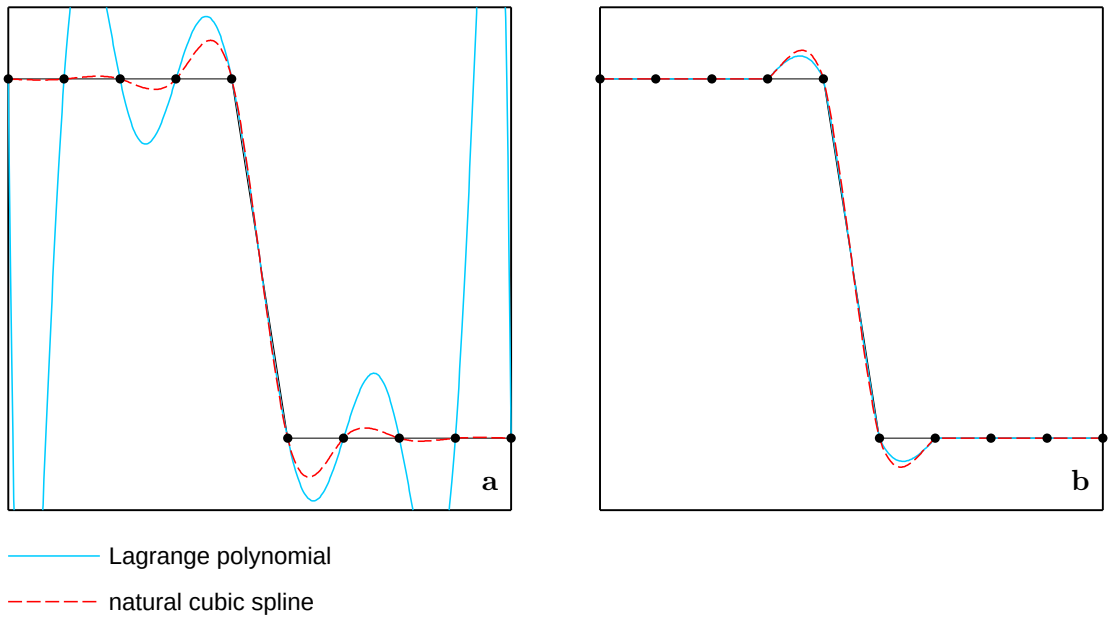

Figure 1: Comparison of global interpolators (a) with their local 4-point versions (b).

since it prevents efficient parallelization. That is why more local strategy is usually adopted. For example, value $y(x)$ on interval (x_n, x_{n+1}) can be obtained by 4-point interpolator applied on nodes $x_{n-1}, x_n, x_{n+1}, x_{n+2}$. Global solution $y(x)$ is then glued together from local solutions on individual intervals. Figure 1b shows that with this approach cubic Lagrange polynomial overshoots *less* than natural cubic spline. Price paid for use of local interpolators is loss of global smoothness. For example, while global cubic spline is C^2 function (i.e. continuous up to second derivative), its local version is only C^0 function (i.e. continuous). Discontinuity of first derivative can occur in nodes, since different local solutions meet here. This can be observed on figure 1b.

3.2 Important 4-point interpolators

In subsequent text, four "traditional" 4-point interpolators will be mentioned frequently. They are defined on interpolation stencil x_0, x_1, x_2, x_3, where function to be interpolated has values y_0, y_1, y_2, y_3. Formulas given below are valid for $x \in [x_1, x_2]$:

1. Linear interpolator (lin)

   Strictly speaking, this is just 2-point interpolator, since it does not use information from outer nodes. It is defined as straight line connecting points (x_1, y_1) and (x_2, y_2):

   $$y(x) = (1 - \xi)y_1 + \xi y_2 \qquad \xi \equiv \frac{x - x_1}{x_2 - x_1}$$

2. Cubic Lagrange polynomial (lag)

   This interpolator is defined as third order polynomial passing through interpolation points (x_0, y_0), (x_1, y_1), (x_2, y_2) and (x_3, y_3):

   $$y(x) = \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} y_0 + \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} y_1 +$$
   $$+ \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} y_3$$

3. Natural cubic spline (spl_n)

   It is defined as piecewise cubic polynomial passing through interpolation points (x_0, y_0), (x_1, y_1), (x_2, y_2) and (x_3, y_3), continuous up to second derivative. Third derivative can have jumps in inner nodes x_1, x_2. This requirement defines whole class of cubic splines. In order to get unique definition, two more constraints must be specified. For natural cubic spline it is required that second derivative in outer nodes x_0, x_3 is zero, i.e. $y_0'' = y_3'' = 0$. This definition gives:

$$y(x) = (1-\xi)y_1 + \xi y_2 + \frac{1}{6}(x_2 - x_1)^2[-\xi(\xi-1)(\xi-2)y_1'' + (\xi+1)\xi(\xi-1)y_2'']$$

   Second derivatives y_1'' and y_2'' are solution of the system:

$$\frac{x_2 - x_0}{3}y_1'' + \frac{x_2 - x_1}{6}y_2'' = \frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}$$
$$\frac{x_2 - x_1}{6}y_1'' + \frac{x_3 - x_1}{3}y_2'' = \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}$$

4. Quasi-cubic spline (spl_c1)

   Contrary to natural cubic spline, quasi-cubic spline is defined as third order polynomial passing through interpolation points (x_1, y_1) and (x_2, y_2) with prescribed first derivatives y_1' and y_2':

$$y(x) = (1 - 3\xi^2 + 2\xi^3)y_1 + (3\xi^2 - 2\xi^3)y_2 + (x_2 - x_1)[\xi(\xi-1)^2 y_1' + \xi^2(\xi-1)y_2']$$

   Derivative y_1' is prescribed as $f'(x_1)$ where f is quadratic function passing through points (x_0, y_0), (x_1, y_1) and (x_2, y_2). Analogically, derivative $y_2' = g'(x_2)$ with g being quadratic function passing through points (x_1, y_1), (x_2, y_2) and (x_3, y_3):

$$y_1' = \frac{x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)}y_0 + \frac{2x_1 - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)}y_1 + \frac{x_1 - x_0}{(x_2 - x_0)(x_2 - x_1)}y_2$$
$$y_2' = \frac{x_2 - x_3}{(x_1 - x_2)(x_1 - x_3)}y_1 + \frac{2x_2 - x_1 - x_3}{(x_2 - x_1)(x_2 - x_3)}y_2 + \frac{x_2 - x_1}{(x_3 - x_1)(x_3 - x_2)}y_3$$

   When this procedure is applied to more than 4 nodes, result is piecewise cubic polynomial continuous up to first derivative. That is why name *quasi*-cubic spline was adopted (global cubic spline would be continuous up to second derivative).

3.3 General 4-point interpolator

From now on it is assumed that interpolation nodes x_0, x_1, x_2, x_3 are *regular*. This assumption greatly simplifies the analysis, but main conclusions should remain valid also for case with irregular nodes.

General 4-point interpolator can be viewed as function $y(x) = F(x, \mathbf{y})$, where $\mathbf{y} \equiv (y_0, y_1, y_2, y_3)$ are values of function being interpolated at nodes $(x_0, x_1, x_2, x_3) \equiv (-1, 0, 1, 2)$. With this choice of nodes interpolation point x lies in interval $[0, 1]$.

Function $F(x, \mathbf{y})$ cannot be completely arbitrary, but it should respect several basic requirements:

1. Linearity with respect to $\mathbf{y}$:

$$F(x, \mathbf{y_1} + c\mathbf{y_2}) = F(x, \mathbf{y_1}) + cF(x, \mathbf{y_2})$$

2. Invariance with respect to horizontal mirroring:

$$F(1-x, y_3, y_2, y_1, y_0) = F(x, y_0, y_1, y_2, y_3)$$

3. Invariance with respect to vertical shift:

$$F(x, y_0 + c, y_1 + c, y_2 + c, y_3 + c) = F(x, y_0, y_1, y_2, y_3) + c$$

4. Reproducing of values y_1, y_2:

$$\begin{aligned} F(0, y_0, y_1, y_2, y_3) &= y_1 \\ F(1, y_0, y_1, y_2, y_3) &= y_2 \end{aligned}$$

5. Reproducing of linear function $y = x$:

$$F(x, -1, 0, 1, 2) = x$$

Requirement 1 restricts shape of $F(x, \mathbf{y})$ to:

$$F(x, \mathbf{y}) = w_0(x)y_0 + w_1(x)y_1 + w_2(x)y_2 + w_3(x)y_3$$

Having this form, requirements 2–5 translate into constraints for interpolation weights w_0, w_1, w_2, w_3:

$$\begin{aligned} 2: \quad & w_2(x) = w_1(1-x) \\ & w_3(x) = w_0(1-x) \\ \\ 3: \quad & w_0(x) + w_1(x) + w_2(x) + w_3(x) = 1 \\ \\ 4: \quad & w_0(0) = 0 \qquad w_0(1) = 0 \\ & w_1(0) = 1 \qquad w_1(1) = 0 \\ & w_2(0) = 0 \qquad w_2(1) = 1 \\ & w_3(0) = 0 \qquad w_3(1) = 0 \\ \\ 5: \quad & -w_0(x) + w_2(x) + 2w_3(x) = x \end{aligned}$$

Remark 1:

Interpolation weights w_0, w_1, w_2, w_3 can be negative for some values of x. They are called weights just because they sum up to one.

Remark 2:

Requirements 1, 3 and 5 guarantee that $F(x, \mathbf{y})$ reproduces any linear function. In other words, it is first order accurate.

3.4 Family of cubic 4-point interpolators

As the next step, 4-point interpolator $F(x, \mathbf{y})$ can be further restricted by requiring that weights w_0, w_1, w_2, w_3 are polynomials of degree at most 3. Then, constraints 2–5

enable to eliminate 14 of 16 coefficients, so that at the end one is left with only two free parameters $(a_1, a_2) \in \mathbb{R}^2$:

$$\begin{aligned} w_0(x) &= a_1 x + a_2 x^2 - (a_1 + a_2) x^3 \\ w_1(x) &= 1 + (a_2 - 1)x - (3a_1 + 4a_2)x^2 + 3(a_1 + a_2)x^3 \end{aligned}$$

$$F(x, \mathbf{y}) = w_0(x) y_0 + w_1(x) y_1 + w_1(1 - x) y_2 + w_0(1 - x) y_3 \tag{1}$$

Case with irregular nodes is examined in appendix A.

Requirement that $F(x, \mathbf{y})$ reproduces also quadratic function $y = x^2$ (which implies second order accuracy) defines straight line in (a_1, a_2) plane:

$$6a_1 + 2a_2 = -1 \tag{2}$$

It means that space of second order accurate 4-point cubic interpolators is only one dimensional.

Since all "traditional" 4-point interpolators are at most cubic, they can be classified in terms of (a_1, a_2). This is done in table 1, which summarizes also their basic properties. It can be observed that linear interpolator and natural cubic spline are only first order accurate. On the other hand, cubic Lagrange polynomial is the only third order accurate interpolator in (a_1, a_2) space. As for global smoothness, quasi-cubic spline is continuous up to first derivative, all other interpolators are only continuous. The only interpolator preserving monotonicity is the linear one. To say something nice also about natural cubic spline, it will be seen later that it is most precise for short waves.

a_1	a_2	name	global smoothness	order of accuracy	monoto-nicity
0	0	linear interpolator	C^0	1	yes
$-\frac{1}{3}$	$\frac{1}{2}$	cubic Lagrange polynomial	C^0	3	no
$-\frac{7}{15}$	$\frac{4}{5}$	natural cubic spline	C^0	1	no
$-\frac{1}{2}$	1	quasi-cubic spline	C^1	2	no

Table 1: Classification and basic properties of "traditional" 4-point interpolators.

Remark:

It can be seen from relations (1) that when F_1 and F_2 are interpolators from (a_1, a_2) family, so is their weighted average $(1 - \kappa)F_1 + \kappa F_2$ where $\kappa \in \mathbb{R}$. For example, every second order accurate (a_1, a_2) interpolator can be expressed as weighted average of cubic Lagrange polynomial (lag) and quasi-cubic spline (spl_c1). This property can be advantageous for code implementation.

To go one step further, when interpolators F_1, F_2 and F_3 form a triangle in (a_1, a_2) plane, every interpolator from this plane can be obtained as weighted average $\kappa_1 F_1 + \kappa_2 F_2 + (1 - \kappa_1 - \kappa_2)F_3$ where $\kappa_1, \kappa_2 \in \mathbb{R}$.[5] It means that complete family of cubic 4-point interpolators can be generated e.g. from linear interpolator (lin), cubic Lagrange polynomial (lag) and quasi-cubic spline (spl_c1). To illustrate this, natural cubic spline (spl_n) can be decomposed as:

$$F_{\text{spl\_n}} = -\frac{1}{5} F_{\text{lin}} + \frac{4}{5} F_{\text{lag}} + \frac{2}{5} F_{\text{spl\_c1}} \tag{3}$$

[5]This result is specific for regular nodes. With irregular nodes additional degrees of freedom appear, see appendix A for details.

3.5 Accuracy of cubic 4-point interpolators

Important criterion for comparing performance of various interpolators is their accuracy or precision. Intuitively, the higher precision the better. Problem with this approach is that precision can be defined in many ways, since interpolation error depends on function being interpolated. Therefore, it is necessary to define representative sample of test functions on which interpolation error will be evaluated.

Traditional approach is to use polynomials as test functions. Interpolator is said to be n-th order accurate if it is exact for polynomials up to order n. Precision measured in this way is very broad, since it says nothing about interpolation error when interpolator is applied on function for which it is not exact. For example, various second order accurate interpolators can perform very differently when applied on cubic polynomial or some more general function.

In order to provide finer measure of precision, sample of harmonic test functions was used. Method was inspired by work [7], namely by procedure leading to results shown on figure 3 therein. It was adopted in following way:

Basic grid had $N = 100$ points. Test functions of the form $y(x) = \sin(2\pi mx/N)$ with $x \in [0, N]$ were taken. Index $m = 1, 2, \ldots, M$ denotes number of wavelengths in the domain. Linear truncation with $M = 49$ was chosen. Waves shorter than $2\Delta x$ were ignored, since they are undersampled and aliased to longer waves.

Each grid interval was divided into $I = 20$ subintervals. Test function was interpolated onto this finer grid and its values were compared against exact ones. Cost function measuring overall precision was mean absolute error with weights proportional to $\exp(-\beta m/M)$, where β was tuning parameter used to control significance of shortest waves.[6]

Described method was applied to interpolators from (a_1, a_2) family. Results are displayed on figure 2, which shows contours of weighted mean absolute error in part of (a_1, a_2) plane. Plots on the left are for normal interpolators, plots on the right for their quasi-monotonic versions.[7] Thin dashed red line on the plots denotes second order accurate interpolators. Second order accuracy is lost for quasi-monotonic treatment (plots b, d, f), where the red line applies to original (a_1, a_2) interpolator.

Plots a, b show results for $\beta = 25$, when very little weight is given to short waves. Error surface forms a V-shaped valley with round bottom, oriented from NW to SE. Line of second order accuracy coincides with the axis of valley, where the error is smallest. This is no surprise, since for long waves local Taylor expansion up to second order provides sufficient accuracy within $3\Delta x$ interpolation stencil. Natural cubic spline (spl_n) is slightly displaced from the line of minimum error, because of its first order accuracy. Linear interpolator (lin) is displaced much more, having largest error from the four interpolators shown.

Plots c, d ($\beta = 5$) and e, f ($\beta = 1$) show what happens when more weight is given to short waves. Error rises, axis of valley separates from the line of second order accuracy and moves to the left. Originally straight error contours deform into elliptic shape. Result is that natural cubic spline (spl_n) becomes most precise of all four interpolators. This is due to its overshooting tendency, which improves performance for poorly sampled short waves. Accuracy of quasi-cubic spline (spl_c1) remains close to that of cubic Lagrange polynomial (lag).

Comparison of plots on the left with those on the right shows impact of quasi-

[6] One should realize that 50% of test functions have wavelengths between $2\Delta x$ and $4\Delta x$. With $\beta = 0$ cost function would be completely dominated by short waves.

[7] In quasi-monotonic case, interpolated value y is truncated when it lies outside of interval $[y_1, y_2]$.

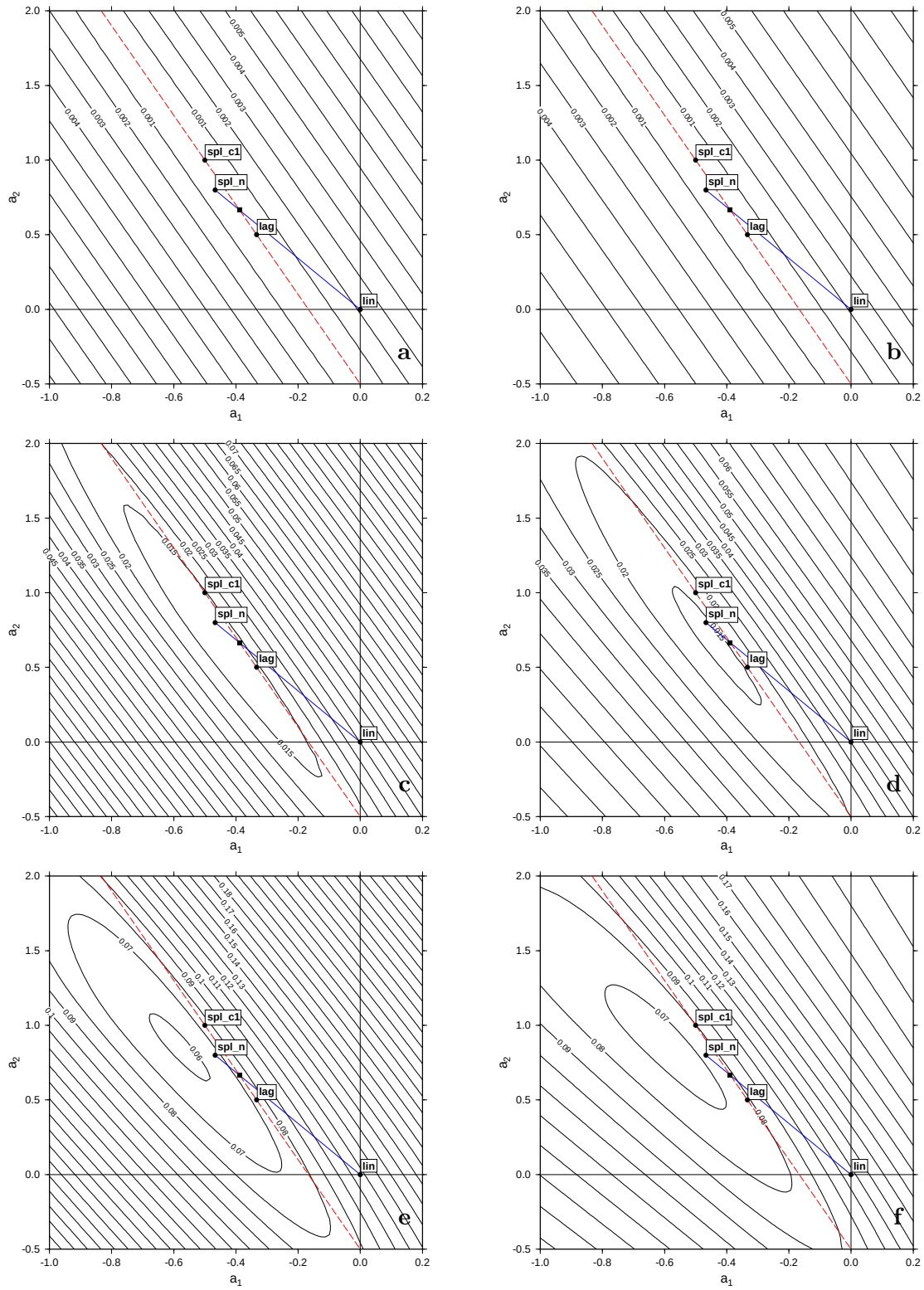

Figure 2: Weighted mean absolute error of 4-point cubic interpolators (left) and their quasi-monotonic versions (right). Going from top to bottom, more weight is given to short waves by decreasing tuning parameter β. Results obtained with $\beta = 25$ (a, b), $\beta = 5$ (c, d) and $\beta = 1$ (e, f). Thin dashed red line denotes second order accurate interpolators, thin blue line represents class of interpolators used by current SLHD scheme with natural cubic splines (spl_n) activated.

monotonic treatment. It increases error close to the valley's axis, so that accuracy of cubic Lagrange polynomial (lag) and cubic spline interpolators (spl_n, spl_c1) is deteriorated. It does not touch error of linear interpolator (lin) and reduces error on distant slopes. However, performance of interpolators from these regions is so poor that they are out of interest.

Precision of various (a_1, a_2) interpolators evaluated on sample of harmonic functions revealed several things:

- For long waves, second order accurate interpolators are optimal in terms of precision. Natural cubic spline (spl_n) is only slightly less precise, linear interpolator (lin) being much worse.
- When more weight is given to short waves, natural cubic spline (spl_n) outperforms second order accurate interpolators.
- In studied portion of (a_1, a_2) plane, moving along second order accuracy line changes overall precision only slightly, keeping it much better than for linear interpolator (lin).
- Quasi-monotonic treatment slightly deteriorates performance of the most precise interpolators.

3.6 Diffusivity of cubic 4-point interpolators

As was mentioned in introduction, semi-Lagrangian interpolators suffer from scale selective damping which is not present when Eulerian advection is used [5]. Since this unphysical source of damping deteriorates precision of semi-Lagrangian scheme, question rises how much it can be reduced by choice of suitable interpolator. On the other hand, diffusivity of semi-Lagrangian interpolators is exploited by SLHD scheme which uses them to control short waves [6, 8]. Important question is how selectively this can be done, since one does not want to touch longer waves much. When interpolator should be accurate for long waves, how much space is left for tuning of its diffusivity?

In order to answer these questions, diffusivity of interpolator have to be defined. First idea was to solve constant speed 1D linear advection with semi-Lagrangian scheme and to look how does amplitude of harmonic wave evolve with time:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{4}$$

$$u(x, t_0) = \sin(kx) \qquad k = 2\pi m/N \qquad m = 1, 2, \ldots, M$$

Computational domain was periodic with $N = 100$ points. Truncation was again linear with $M = 49$. Non-dimensional coordinates $x \in [0, N]$ and t were used, such that velocity $c = 1$. In order to simulate random distribution of origin points, timesteps $\Delta t_i \equiv t_i - t_{i-1}$ were initialized with pseudo-random sequence of numbers uniformly distributed in interval $[0, 1)$. With this distribution of timesteps, mean Courant number was 0.5.

Semi-Lagrangian integration of equation (4) is trivial:

$$u(n, t_i) = u(n - \Delta t_i, t_{i-1}) \qquad i = 1, 2, \ldots, I \tag{5}$$

RHS of equation (5), i.e. value at origin point $x = n - \Delta t_i$, is evaluated using interpolator. Having time evolution for wavenumber k enables to obtain corresponding damping rate $\gamma(k)$, supposing that wave amplitude decays as $\exp[-\gamma(k)t]$.

Application of this method on interpolators from (a_1, a_2) family showed that amplitude of advected harmonic wave decays exponentially with time up to the point where the solution becomes dominated by numerical noise. This happens when amplitude is reduced to ε times its original value, ε being machine precision (on machine with 15 digits $\varepsilon = 10^{-15}$). It has therefore no meaning to go beyond this point. Moreover, it was seen that natural cubic spline (spl_n) is slightly *amplifying* long waves.

Problem appeared when the method was applied to quasi-monotonic versions of *overshooting* interpolators (lag, spl_c1, spl_n). For some waves amplitude evolution ceased to obey exponential law very soon, *slowing down* the decay rate.

Explanation of this phenomenon is very simple and it reveals serious limitation of proposed method. Figure 3a shows amplitude evolution for wave with $m = 27$ ($\lambda \approx 3.7\Delta x$) when it is advected using natural cubic spline (spl_n) and its quasi-monotonic version. In early stage quasi-monotonic solution decays faster, but after few tens of timesteps its decay rate reduces significantly, becoming much smaller than decay rate of original interpolator. Reason can be understood from figure 3b, which shows shape of advected wave after 100 timesteps. While for original interpolator wave with $m = 27$ is still apparent, quasi-monotonic solution is dominated by longer waves with $m = 11$ and $m = 3$ ($\lambda \approx 9.1\Delta x$ and $\lambda \approx 33.3\Delta x$). It means that less precise quasi-monotonic interpolator gradually distorted original short wave, converting it into longer waves which are much less damped afterwards. This is direct consequence of the fact that while harmonic waves are eigenfunctions of RHS operator (5) based on (a_1, a_2) interpolator, it is not true for its quasi-monotonic version as soon as overshoots appear. In such case damping rate $\gamma(k)$ is not well defined concept.

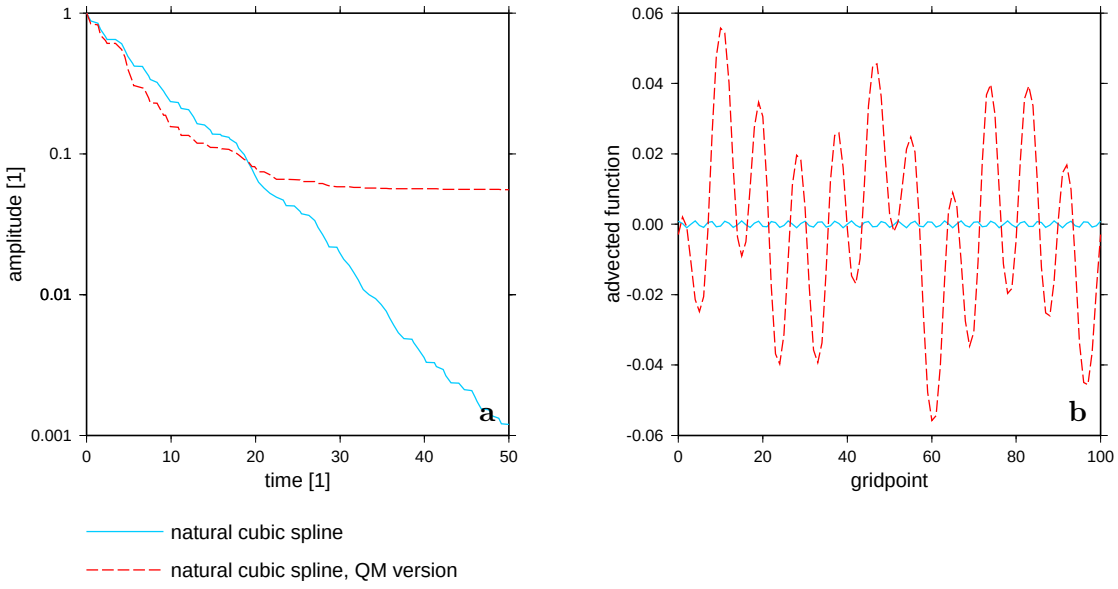

Figure 3: Evolution of amplitude for advected harmonic wave with $m = 27$ (a) and wave shape after 100 timesteps (b).

Even if problematic for long evolution times, damping rate $\gamma(k)$ of quasi-monotonic interpolator can be defined in the limit of single timestep, where the wave distortion remains negligible. New method still uses multiple interpolations, but always starts from initial wave. Test function is interpolated onto grids shifted by $\alpha \equiv i/I$, where $i = 1, 2, \ldots, I$.[8] For every shift amplitude of interpolated wave A_{int} is compared

[8]Shift α can be interpreted as residual Courant number.

to amplitude of exact solution A_{exact} sampled onto shifted grid. Ratio $A_{\text{int}}/A_{\text{exact}}$ is then averaged over shifts α and mean damping rate $\gamma(k)$ is determined assuming dimensionless evolution time $t = 0.5$.[9] Results are not sensitive to the number of shifts I once it is large enough. For number of gridpoints $N = 100$ and linear truncation $M = 49$ it was sufficient to use $I = 40$.

New method is very close to that of McCalpin [1]. Main difference is that McCalpin based his approach on amplification factors, which restricts its applicability only to interpolators for which RHS operator (5) has harmonic eigenfunctions. On the other hand, he uses analytical formulas and does not average the results through residual Courant number α.

Using the new method, damping rates for 5 different wavelengths were computed in part of (a_1, a_2) plane. They are shown on figure 4. For longest wave (plot a), second order accurate interpolators are neutral. Natural cubic spline (spl_n) is slightly amplifying and linear interpolator (lin) is damping. For $10\Delta x$ wave (plot b), neutrality line starts to tilt from the line of second order accuracy (thin dashed red line), so that quasi-cubic spline (spl_c1) and cubic Lagrange polynomial (lag) become sligthly damping, while natural cubic spline (spl_n) remains amplifying. Situation becomes different for $3.0\Delta x$ wave (plot c). Neutrality line is shifted to the left and originally straight isolines start to bend. All four interpolators are now damping. Moving along line of second order accuracy from NW to SE increases damping rate. This remains true also for $2.5\Delta x$ and $2.0\Delta x$ waves (plots d, e), but shape of damping surface changes and orientation of its slope in central part of domain turns clockwise. Consequence of this turning is that while for $3.0\Delta x$ wave natural cubic spline (spl_n) was least damping of the four interpolators, for $2.5\Delta x$ and $2.0\Delta x$ it is quasi-cubic spline (spl_c1). Most damping is always linear interpolator (lin).

Figure 5 shows damping rates with quasi-monotonic treatment. Most important difference from figure 4 is that regions of amplification ($\gamma < 0$) disappear. Otherwise, qualitative behaviour is the same. There is one curiosity worth mentioning: For very inaccurate interpolators (which are therefore out of question), quasi-monotonic treatment can really *decrease* damping rate for some wavelengths. In order to understand why, it is enough to look how such interpolator works on wave concerned. One example is given on figure 6.

Remark:

It must be remembered that diffusivity of semi-Lagrangian scheme depends on timestep length. Presented results were obtained for mean Courant number 0.5. Mean Courant number 1.5 would lead to 3 times weaker damping rates, since the number of interpolations within given time window (having the same residual Courant number α) would be reduced by factor 3. Damping rates remain bounded in the limit of zero timestep, because increasing number of interpolations is compensated by α tending to zero (small α means weak damping, since interpolation points are close to the nodes).

In this section it was seen that:

- Diffusivity of interpolator can be characterized by its damping rate $\gamma(k)$. However, for interpolators which distort advected harmonic waves this can be done only in the limit of single timestep.

[9] This choice of evolution time corresponds to mean Courant number 0.5.

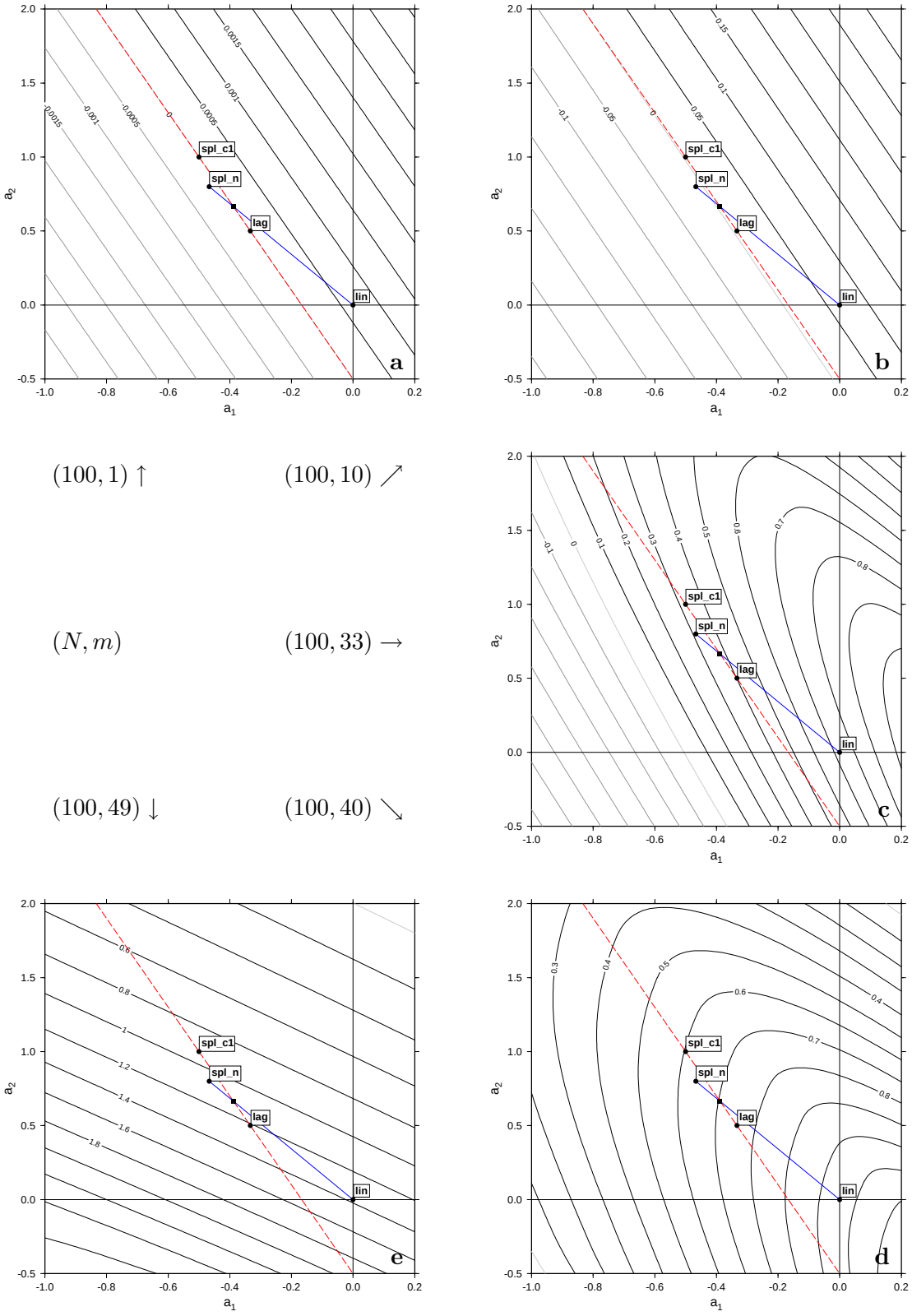

Figure 4: Dimensionless damping rate γ of 4-point cubic interpolators for waves with $m = 1$ (a), $m = 10$ (b), $m = 33$ (c), $m = 40$ (d) and $m = 49$ (e) in domain with $N = 100$ points. Corresponding wavelengths are approximately $100\Delta x$ (a), $10\Delta x$ (b), $3.0\Delta x$ (c), $2.5\Delta x$ (d) and $2.0\Delta x$ (e). Meaning of red and blue lines is the same as on figure 2.

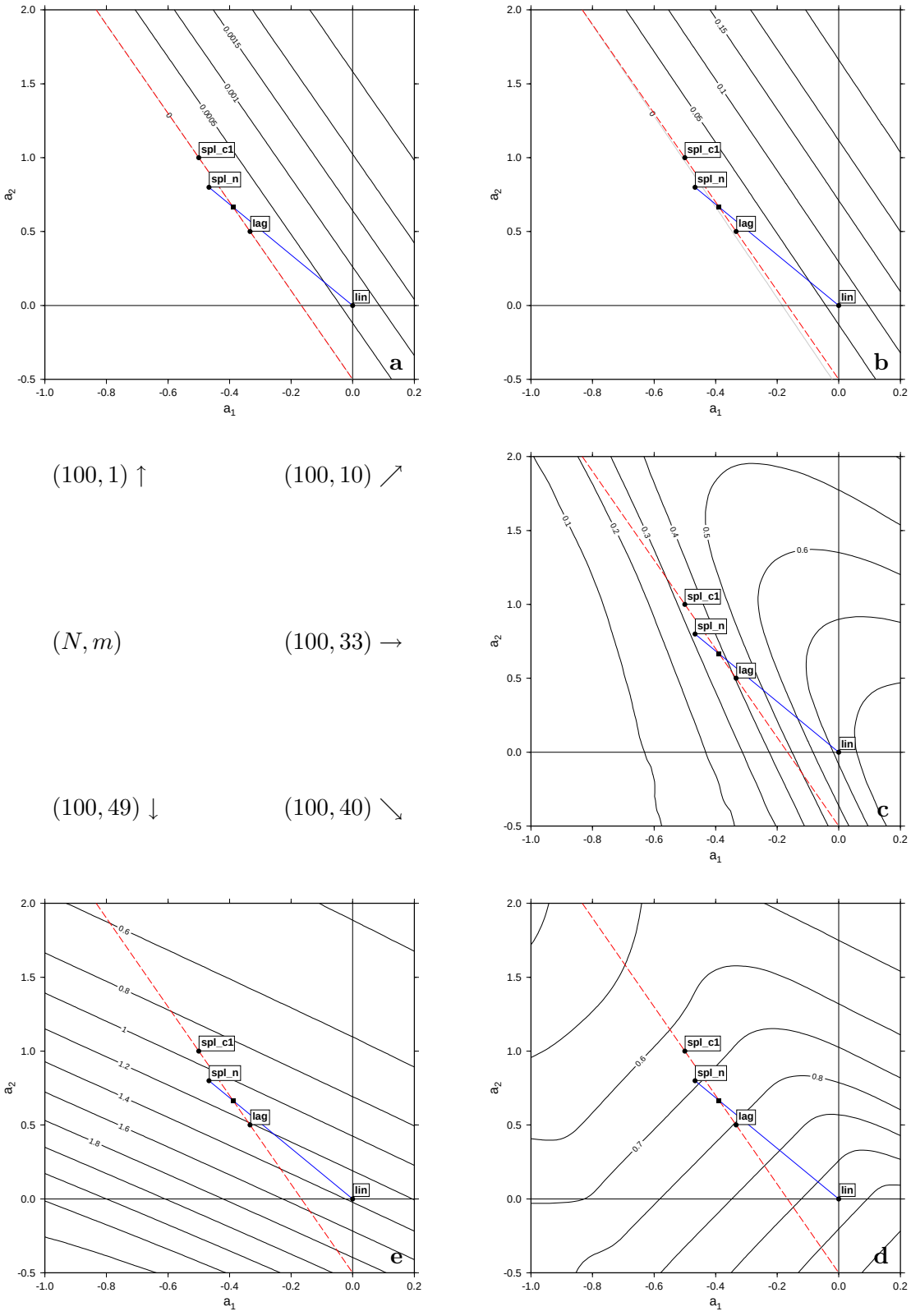

Figure 5: Same as figure 4, but with quasi-monotonic treatment.

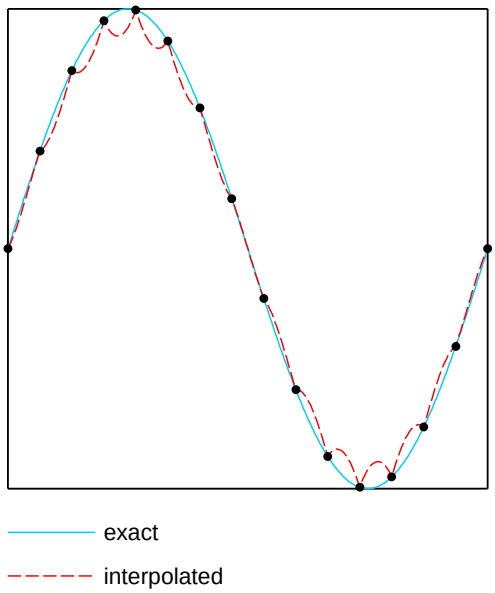

Figure 6: Performance of 4-point cubic interpolator $(0, 2)$ for longest wave in domain with 15 points.

- Second order accurate interpolators are optimal for long waves, being neutral or slightly damping. Natural cubic spline (spl_n) is sligtly amplifying long waves while linear interpolator (lin) is strongly damping.
- Diffusivity of interpolators from (a_1, a_2) family varies in a wide range. There exist second order accurate interpolators which are damping short waves much less than traditional interpolators like cubic Lagrange polynomial (lag) or even natural cubic spline (spl_n), having only slightly worse overall precision. At the same time, there are second order accurate interpolators damping short waves stronger than linear interpolator (lin), while retaining much better precision.
- Quasi-monotonic treatment removes unstable regions from (a_1, a_2) plane. It increases damping rate of accurate interpolators, which results in slight deterioration of their precision.

3.7 Equivalent diffusion

Diffusive properties of interpolator are described by its damping rate $\gamma(k)$. Sometimes they can be characterized in terms of equivalent diffusion. The idea is to fit damping rate $\gamma(k)$ with function $D \times k^p$, where fitting parameters D and p are equivalent diffusion coefficient and order. The higher equivalent diffusion order p, the more scale selective interpolator is.

Function $D \times k^p$ is nothing else but damping rate of generalized 1D diffusion, driven by equation:

$$\frac{\partial u}{\partial t} = -D \left(-i \frac{\partial}{\partial x} \right)^p u \tag{6}$$

$$D > 0 \qquad p \geq 0$$

For $p = 2$ it reduces to standard Laplacian diffusion:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

Diffusion order p in equation (6) is not necessarily integer. Derivative of non-integer order p is defined through its action on harmonic functions $\exp(ikx)$:

$$\left(\frac{\partial}{\partial x}\right)^p \exp(ikx) \equiv (ik)^p \exp(ikx)$$

When the fitting procedure was blindly applied to "traditional" interpolators, it provided following equivalent diffusion orders:

linear interpolator	$p \approx 2$
cubic Lagrange polynomial, quasi-cubic spline	$p \approx 4$
natural cubic spline	$p \approx 6$

According to this it seems that natural cubic spline (spl_n) is most scale selective from the four interpolators, but some care is needed when interpreting the results. In case of linear interpolator (lin), cubic Lagrange polynomial (lag) and quasi-cubic spline (spl_c1), fitted dependency was close to power law at least for long waves. However, this was not the case for natural cubic spline (spl_n), where amplifying long waves had to be excluded since the fitting uses log-log scaling. Moreover, for the remaining wavenumbers there was significant departure from power law. Therefore, equivalent diffusion order is not well defined concept for natural cubic spline (spl_n). For other interpolators it has meaning mainly in longwave part of spectrum.

Diffusivity of cubic spline interpolator was investigated for example by Purnell [2] and McCalpin [1]. They found it to be stable for all wavenumbers. Moreover, McCalpin states that in longwave approximation cubic spline interpolator is equivalent to fourth order diffusion and gives following asymptotic formula for amplification factor $|\lambda|$ when residual Courant number $\alpha \ll 1$:

$$|\lambda|^2 = 1 - \frac{1}{12}\alpha^2(k\Delta x)^4 \tag{7}$$

In formula (7) k is wavenumber and Δx is grid spacing; longwave approximation assumes $k\Delta x \ll 1$. (In original paper there is a typo – missing second power for $|\lambda|$.)

How to explain this discrepancy? Problem is hidden in the fact that mentioned authors analyzed global cubic spline with *periodic* boundary conditions and their results cannot be extrapolated to 4-point *natural* cubic spline. When the analysis is repeated for this case, amplification factor obtained in $k\Delta x \ll 1$ and $\alpha \ll 1$ limit is:

$$|\lambda|^2 = 1 + \frac{1}{5}\alpha(k\Delta x)^2 - \frac{3}{20}\alpha(k\Delta x)^4 + \cdots \tag{8}$$

This is consistent with numerical results, since dominant second order term (in k) has positive sign, so that long waves are amplified. Next fourth order term has negative sign, but it cannot change amplification to damping when $k\Delta x$ is small.

Classification of 4-point cubic interpolators derived in this section enabled to map their accuracy and diffusivity. It was shown that diffusivity of second order accurate interpolator can vary in wide range, while its overall precision remains much better than that of linear interpolator (lin). This property makes the class of second order accurate interpolators very attractive for use in SLHD scheme.

It was also shown that 4-point natural cubic spline (spl_n) has outstanding performance for short waves, but it is slightly unstable for long ones. This weak instability might be the source of problem seen in 3D model.

SLHD scheme controls diffusivity by contaminating high order interpolator with more diffuse linear one [6, 8]. When original interpolator is second order accurate, blending with linear interpolator (lin) distracts it from the line of second order accuracy towards origin, which results in loss of accuracy and overall precision. With natural cubic spline (spl_n) situation is sligthly different, since it is originally displaced to the other side. When it is blended with linear interpolator (lin), it moves along thin blue line (figures 2, 4 and 5) *towards* line of second order accuracy and for $\kappa = \frac{1}{6}$ becomes second order accurate (black square on the plots). For larger κ accuracy is reduced again. Anyway, as can be seen from plots 2c and 2e, blended interpolator based on natural cubic spline (spl_n) remains more precise than its equivalent based on cubic Lagrange polynomial (lag). The only exception is on plot 2a for κ small.

In SLDH scheme value of κ is controlled by horizontal deformation of the flow. Should the spline instability be responsible for problem seen in 3D model, it would be regions with weak flow deformation which trigger it. If this is really the case, SLHD scheme based on class of second order accurate interpolators could be the solution.

4 Idealized 2D experiments

Results of previous section were obtained in very simplified context. In this section, behaviour of selected semi-Lagrangian interpolators is tested in more realistic framework. 2D vertical plane version of model ALADIN-NH is used for this purpose. As for test case, bubble experiments of Robert [3] were taken, because of their high sensitivity to discretization details.[10]

Experiments used resting and neutrally stratified initial state, with constant background potential temperature $\overline{\theta} = 300\,\mathrm{K}$ and constant surface pressure 101 325 Pa. Circular bubble perturbations were superposed to initial $\overline{\theta}$ field, with details given below:

test	$\theta'_{\max}$ [K]	x_0 [m]	z_0 [m]	a [m]	s [m]
#1	−0.50	500	700	150	50
	+0.15	560	360	0	50
#2	+0.50	500	260	250	

Bubble with radius a was placed in point (x_0, z_0). For the first test bubbles had smooth edge with gaussian profile (r denotes distance from the centre):

$$\theta' = \begin{cases} \theta'_{\max}, & r \le a \\ \theta'_{\max} \exp\left(-r^2/s^2\right), & r > a \end{cases}$$

[10] Moreover, they produce nice and in some way intuitive results. When the model is OK, of course.

For the second test bubble edge was sharp ($s \to 0^+$).

Both tests were done in domain 1×1 km, with mesh size $\Delta x = \Delta z = 10$ m.[11] Domain was periodic in x direction, without any coupling or extension zone. Additional 30 layers with gradually increasing thickness had to be added on the top, since current ALADIN does not enable top pressure $p_T > 0$. No sponge was used on vertical. For the first test truncation was quadratic, for the second linear.

As for domain boundaries, Robert uses rigid walls everywhere. This was not possible with ALADIN, but for short evolution times difference is negligible. In order to be directly comparable with [3], first test was vertically reverted with change of sign for θ'. In this way, rigid top acting on rising hot bubble was replaced by rigid bottom acting on its cold sinking mirror image.[12]

Initial state was integrated using fully compressible model. Advection scheme was either two-time level non-extrapolating semi-Lagrangian with iterated centered implicit time treatment and timestep 5 s, or three-time level Eulerian with semi-implicit time treatment and timestep 1 s. In case of semi-Lagrangian scheme, LGWADV option (i.e. advection of vertical velocity w) had to be used.[13] Simulations were adiabatic, without any explicit source of damping (diffusion, decentering, physics). Asselin filter was used for three-time level scheme.

For testing family of 4-point cubic interpolators, dirty version of the code was prepared. It replaced spline interpolators in subroutine LAITSP by general (a_1, a_2) interpolators, assuming regular nodes in all directions. Comparison of results obtained with Lagrange interpolator (dirty versus clean version) showed, that impact of this simplification on bubble evolution is very weak.

All presented results were obtained with option N[x]LAG = 3, which means that non-linear terms were interpolated using low order interpolator. Few tests performed with N[x]LAG = 2 (all terms interpolated using high order interpolator) showed very little sensitivity to this option.

4.1 Bubble test #1

Reference solution for bubble test #1 was obtained with cubic Lagrange polynomial (lag). It is shown on figure 7. After reverting upside-down it is in quite good agreement with figure 8 of Robert [3] (please note that contour levels are not the same). He proves that this solution is accurate by integrating on twice finer grid (with halved timestep), showing that high resolution results are essentially the same.

Figure 8 shows solutions for different schemes after 10 min. It is obvious that natural cubic spline (spl_n) has some problem, since the solution is distorted and noisy (plot a). First suspicion falls on overshoots. However, turning on quasi-monotonic treatment which eliminates all overshoots does not help much. Some noise is reduced, but the distortion remains (plot b). It means that amplification of long waves can be responsible only for small part of the problem. Second order accurate interpolator $\left(-\frac{2}{3}, \frac{3}{2}\right)$ which has very low diffusivity gives less distorted but noisy solution (plot c). The noisiest of all is Eulerian scheme (plot d), which should have minimal diffusivity only due to time filter. This suggests that main reason for bad behaviour of the four schemes could be their *insufficient* diffusivity. In other words, solution 7d probably cannot be obtained in

[11] Vertical mesh size is only approximate and varies slightly in time, since ALADIN uses mass based hybrid eta coordinate.

[12] This works thanks to special symmetry which holds for Boussinesq flows.

[13] Advection of vertical divergence d produces distortion in bubble experiments and it is still not understood why.

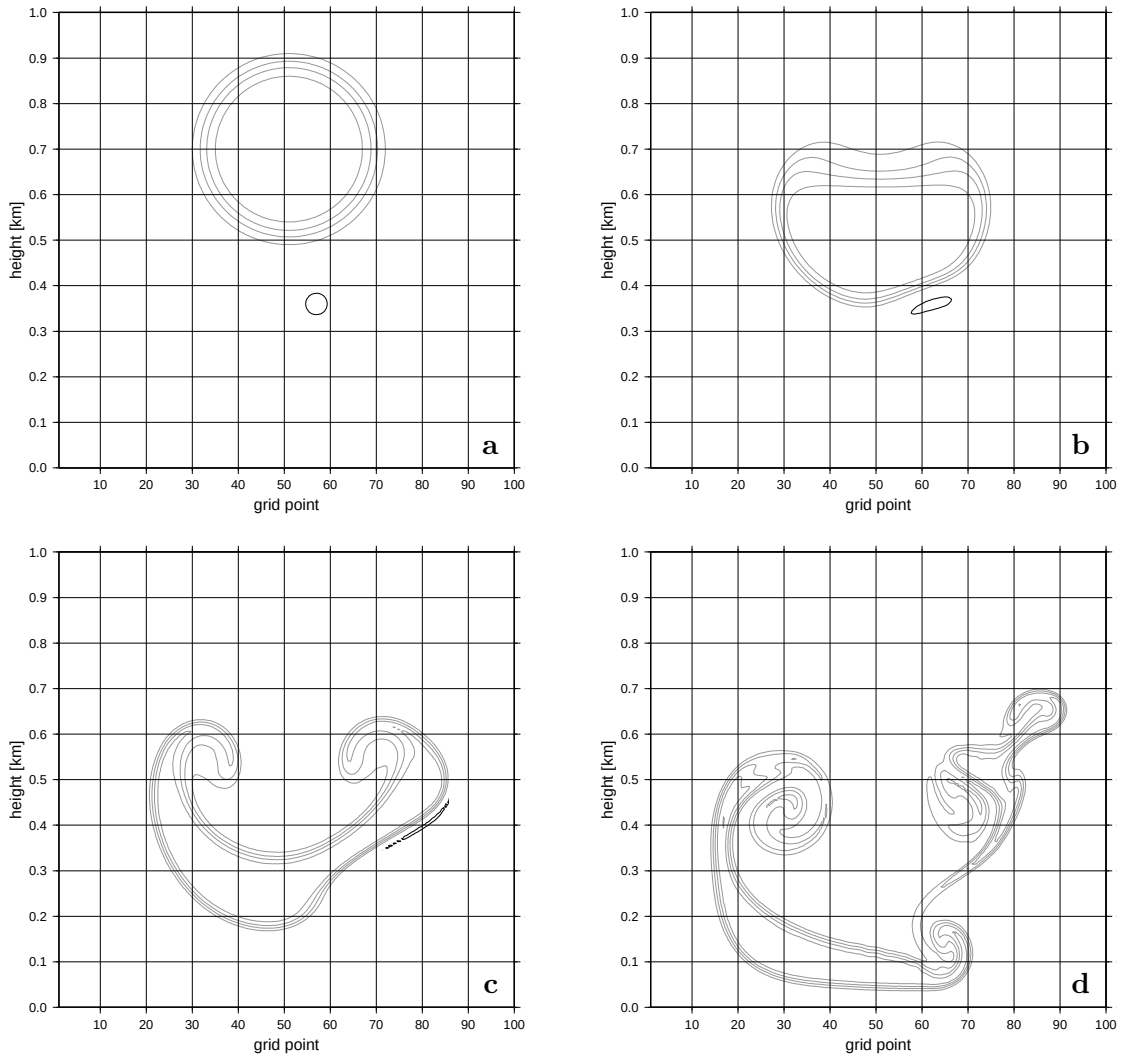

Figure 7: Bubble test #1. Perturbation of potential temperature in initial state (a) and after 4, 7 and 10 min (b, c, d). Reference solution with iterated non-extrapolating 2TL SL scheme and cubic Lagrange interpolator.

completely inviscid case. And by chance, cubic Lagrange polynomial (lag) supplies right amount of "viscosity" to mimick energy dissipation into unresoved scales. Without this dissipation, physically sound solution cannot be obtained.

It is interesting that Robert [3] does not pay much attention to this fact. He compares some of his results with those of Smolarkiewicz and Pudykiewicz [4], remarking that they needed monotonicity constraint in order to get clean solution, while he got it without any such constraint, explicit time filter or diffusion. On one hand he states that his scheme contains some inherent diffusion due to bicubic interpolations, but on the other hand he does not mention that Smolarkiewicz and Pudykiewicz used *non-interpolating* semi-Lagrangian scheme, which has diffusive properties comparable to Eulerian scheme [5]. It seems that their noisy results were caused by using the scheme which was "too decent".

In order to prove that diffusivity is really the key to reliable bubble results, spectral diffusion was turned on for Eulerian scheme. It improved results only partially, probably because it is restricted to horizontal direction (not shown). Another test used second order accurate interpolator $\left(0, -\frac{1}{2}\right)$ which has comparable damping of shortest waves as

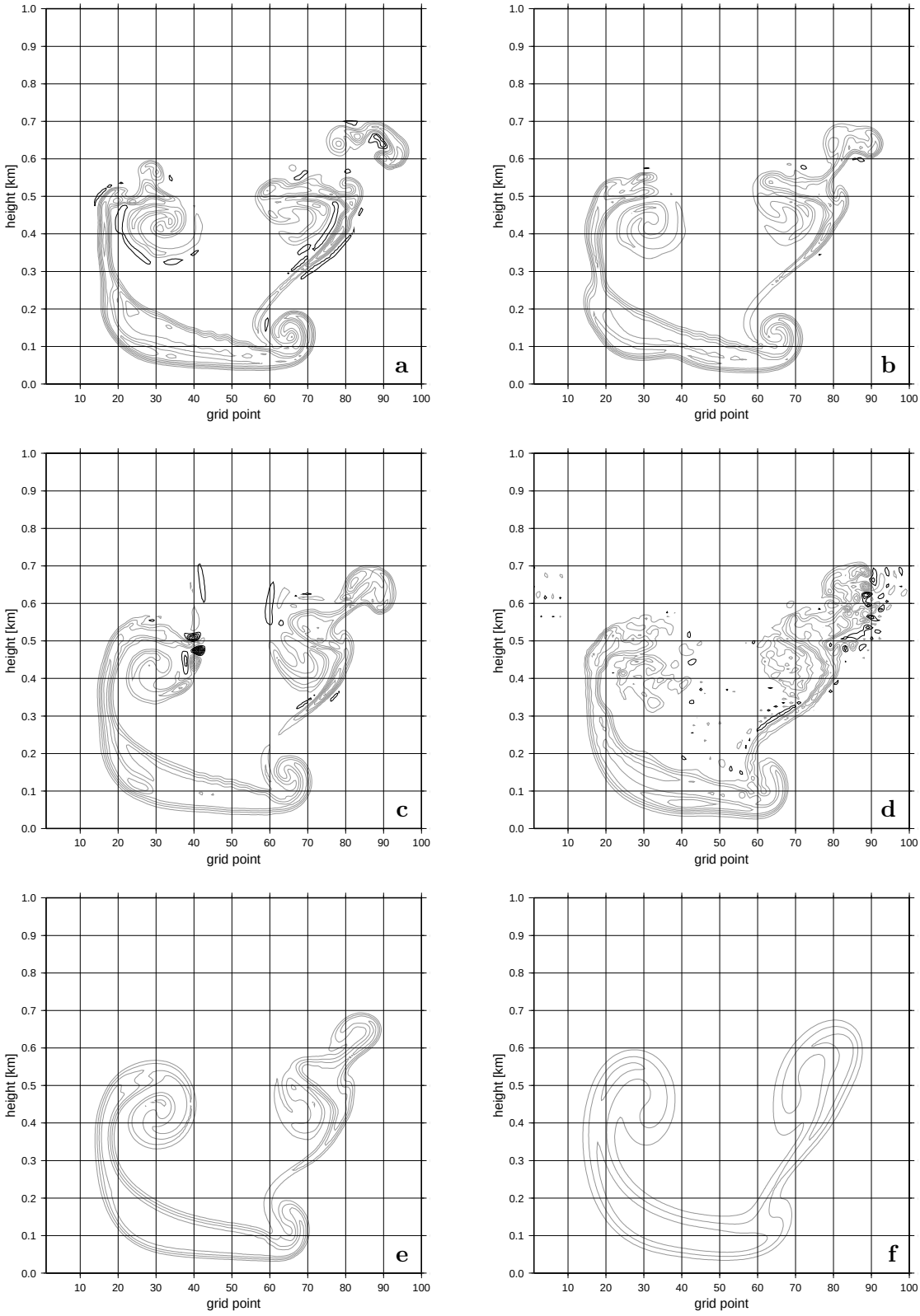

Figure 8: Bubble test #1. Perturbation of potential temperature after 10 min. Results for SL scheme with natural cubic spline (a) and its quasi-monotonic version (b), SL scheme with interpolator $\left(-\frac{2}{3}, \frac{3}{2}\right)$ (c), Eulerian scheme (d), SL scheme with interpolator $\left(0, -\frac{1}{2}\right)$ (e) and with linear interpolator (f).

the linear one. It gives solution very similar to reference 7d (plot e), while the linear interpolator (lin) is not able to keep sharp gradients and smoothes out most of details (plot f). This is not surprising, since it damps least selectively of all tested interpolators.

Bubble test #1 showed two things. First, purely adiabatic simulation cannot provide reliable bubble results. Certain amount of damping is needed in order to parameterize energy dissipation into unresolved scales. For sufficiently precise scheme with very weak internal damping results are noisy in the absence of explicit diffusion. Second, theoretical properties of (a_1, a_2) interpolators obtained in idealized 1D framework are consistent with results of 2D bubble simulations. When scale selective damping is needed, class of second order accurate interpolators provides much better alternative than blending with linear interpolator (lin).

4.2 Bubble test #2

Bubble test #2 demonstrates selectivity of damping for different interpolators. Bubble with sharp edge and linear truncation were used, so that initial state contains also some $2\Delta x$ waves.[14] Robert [3] explains, that this test is not suitable for testing model accuracy, since analytical solution is not known and the results are sensitive to resolution (gradient on the jump becomes sharper as the resolution increases). He also demonstrates that numerical solution is sensitive to small perturbations in initial conditions, especially in upper part of bubble. This must be taken into account when comparing results of different models.[15]

Figure 9 shows bubble after 7 min. Semi-Lagrangian scheme with natural cubic spline (spl_n) keeps sharpest gradients, but it again suffers from noise (plot a). It becomes much more pronounced after 10 min, but here upper part of bubble already escaped from 1×1 km domain (not shown). Cubic Lagrange polynomial (lag) provides smooth solution with less sharp gradients (plot b). The two solutions are qualitatively similar (horseshoe shape, number and placement of main vortices), but some finer details are significantly different. Plots c, d and e illustrate what happens as one moves along second order accuracy line towards more and more diffusive interpolators. Gradients are slightly reduced, vortices become less developed and finer details change, but the resemblance with plot b remains good even for very strong diffusivity. This cannot be said about linear interpolator (lin), which again smoothes out almost every detail (plot f). It is confirmed once again that second order accurate interpolators greatly outperform linear interpolator (lin) in terms of damping selectivity.

Key result of this section is finding that properties of semi-Lagrangian interpolators analyzed in strongly simplified 1D framework are qualitatively valid also for much more realistic 2D bubble simulations. Therefore, class of second order accurate interpolators remains a hot candidate for the use with SLHD scheme.

[14] In previous experiment $2\Delta x$ waves were not allowed by quadratic truncation, but $2\Delta z$ waves could be present in either case.

[15] Simply stated, this is not good benchmark test.

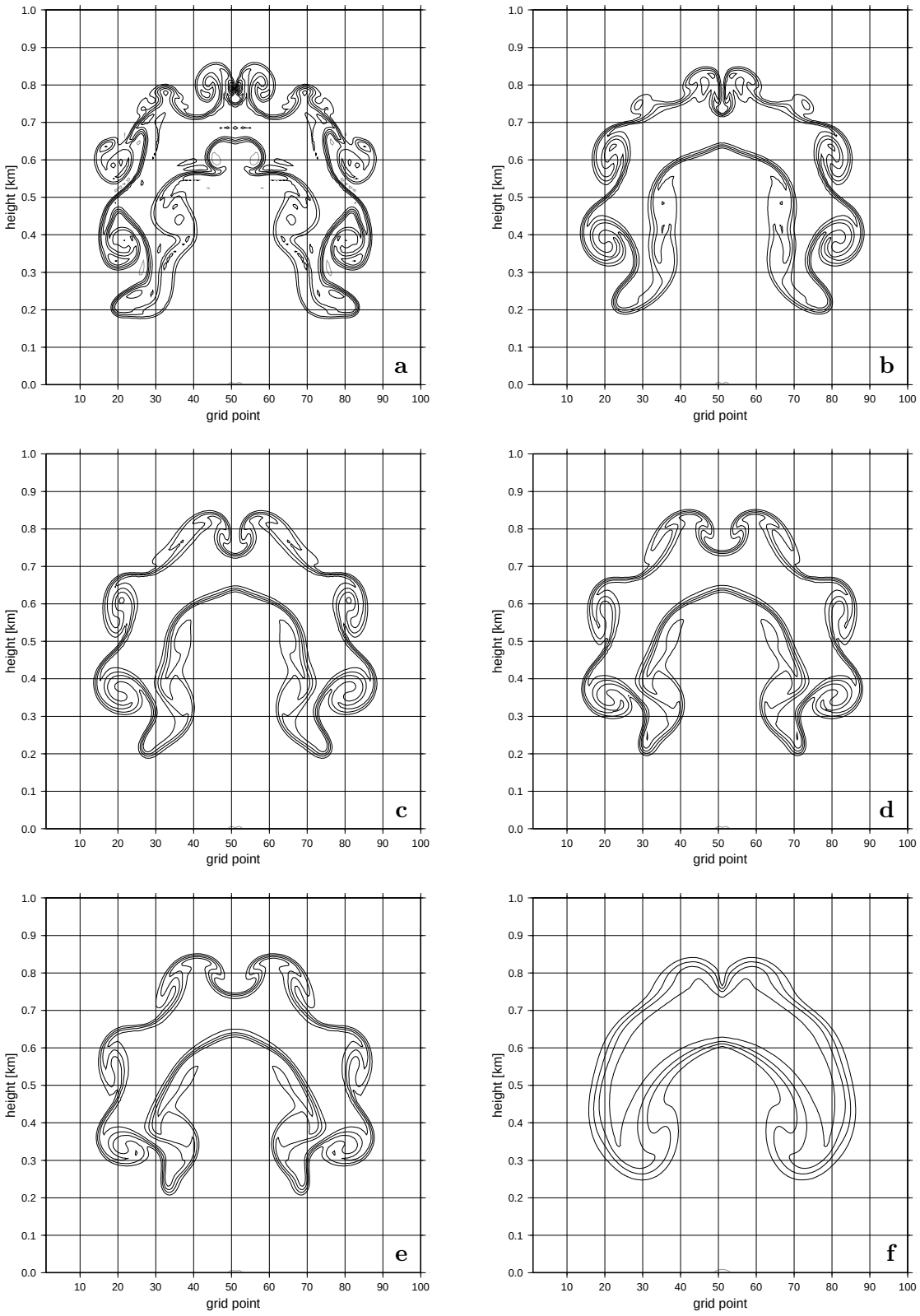

Figure 9: Bubble test #2. Perturbation of potential temperature after 7 min. Results for SL scheme with natural cubic spline (a), cubic Lagrange interpolator (b), interpolator $\left(-\frac{1}{6}, 0\right)$ (c), interpolator $\left(0, -\frac{1}{2}\right)$ (d), interpolator $\left(\frac{1}{6}, -1\right)$ (e) and with linear interpolator (f).

5 Conclusions

This study showed that ARPEGE/ALADIN implementation of 12-point (2D) and 32-point (3D) semi-Lagrangian interpolators based on natural cubic spline (spl_n) is correct, so that problem observed in 3D model must be related to spline properties. Then it concentrated on 4-point 1D interpolators, revealing that four "traditional" interpolators belong to more general 2-parametric family of 4-point cubic interpolators. Properties of this family were studied in detail. It was understood that natural cubic spline (spl_n) outperforms other interpolators for short waves, but the price paid is stronger overshooting leading to slight instability of long waves and only first order accuracy. These deficiencies might be potential source of problem.

Most important finding is that for short waves diffusivity of interpolator decouples from its accuracy. Fixing second order accuracy still enables to adjust diffusivity in a wide range, keeping overall precision much better than for linear interpolator (lin). In longwave approximation, class of second order accurate interpolators has damping equivalent to fourth order diffusion.

Subsequent bubble tests with 2D vertical plane version of model ALADIN-NH revealed problem for schemes without sufficient damping. It was concluded that completely inviscid bubble simulation does not have physical meaning and certain amount of damping is needed in order to mimick energy dissipation into unresolved scales. Experiments with strongly damping second order accurate interpolators showed their clear superiority over linear interpolator (lin), which is used to control diffusivity in SLHD scheme.

Practical output from the study is a recommendation to base SLHD scheme on class of second order accurate interpolators. However, it must be stressed that for the time being it is not sure whether their use will really remove problem with temperature and geopotential bias seen in 3D model with full physics. Anyway, retaining second order accuracy seems to be very promising feature of the proposed scheme. It should guarantee that increase in MSLP bias will not be reintroduced.

If the new SLHD treatment turns to be successful, one may ask whether natural cubic splines (spl_n) should be removed from model code. It is believed that such step would be a bit short-sighted, since they can still prove to be useful for fields which are dominated by short scales (these are typically fields with strong gradients, like humidity or prognostic cloud water and ice). Moreover, slight instability of long waves can be insignificant when there is sufficiently strong physical dissipation. For this reason it would be desirable to keep natural cubic splines (spl_n) in the code.

Even better idea seems to be replacing of current semi-Lagrangian interpolators by more universal ones with tunable properties. This could be achieved by weighting three basic interpolators: linear interpolator (lin), cubic Lagrange polynomial (lag) and quasi-cubic spline (spl_c1). There are other choices but this one is particularly suitable, since first two interpolators are already in the code and the third one will have to be added if SLHD scheme is to be changed. Moreover, linear interpolations are very cheap, so the main increase in CPU consumption would be caused by adding quasi-cubic spline interpolations. Alternatively, blending of the three interpolators could be done at the level of coefficients a_1, a_2. In such case, more complicated computation of interpolation weights would be compensated by only one call to interpolation subroutine. More efficient of the two strategies can be recommended for code implementation.

Acknowledgements. I would like to thank Filip Váňa for initiating and supervising the work, as well as for all comments, remarks and corrections leading to substantial improvment of this report.

Appendix

A Family of cubic 4-point interpolators – case with irregular nodes

In section 3, family of cubic 4-point interpolators was derived assuming regular nodes $(-1, 0, 1, 2)$. Since in code implementation such assumption might not be acceptable, formulas valid for general nodes (x_0, x_1, x_2, x_3) are listed here.

Because general 4-point interpolator should be invariant with respect to linear transformation of x coordinate, it can be written as function $y(x) = F(\xi, p, q, \mathbf{y})$ where ξ is relative position of interpolation point x inside central interval (x_1, x_2) and p, q are relative widths of outer intervals (x_0, x_1) and (x_2, x_3):

$$\xi \equiv \frac{x - x_1}{x_2 - x_1} \qquad p \equiv \frac{x_1 - x_0}{x_2 - x_1} \qquad q \equiv \frac{x_3 - x_2}{x_2 - x_1}$$

Meaning of vector $\mathbf{y} = (y_0, y_1, y_2, y_3)$ is same as before, i.e. values of function being interpolated at nodes (x_0, x_1, x_2, x_3).

With these notations, requirements imposed on general 4-point interpolator can be written as:

1. Linearity with respect to $\mathbf{y}$:

$$F(\xi, p, q, \mathbf{y_1} + c\mathbf{y_2}) = F(\xi, p, q, \mathbf{y_1}) + cF(\xi, p, q, \mathbf{y_2})$$

2. Invariance with respect to horizontal mirroring:

$$F(1 - \xi, q, p, y_3, y_2, y_1, y_0) = F(\xi, p, q, y_0, y_1, y_2, y_3)$$

3. Invariance with respect to vertical shift:

$$F(\xi, p, q, y_0 + c, y_1 + c, y_2 + c, y_3 + c) = F(\xi, p, q, y_0, y_1, y_2, y_3) + c$$

4. Reproducing of values y_1, y_2:

$$\begin{aligned} F(0, p, q, y_0, y_1, y_2, y_3) &= y_1 \\ F(1, p, q, y_0, y_1, y_2, y_3) &= y_2 \end{aligned}$$

5. Reproducing of linear function $y = \xi$:

$$F(\xi, p, q, -p, 0, 1, 1 + q) = \xi$$

Requirement 1 restricts shape of $F(\xi, p, q, \mathbf{y})$ to:

$$F(\xi, p, q, \mathbf{y}) = w_0(\xi, p, q)y_0 + w_1(\xi, p, q)y_1 + w_2(\xi, p, q)y_2 + w_3(\xi, p, q)y_3$$

When interpolation weights w_0, w_1, w_2, w_3 are constrained to be at most cubic polynomials in ξ, requirements 2–5 lead to general shape of cubic 4-point interpolator:

$$\begin{aligned} w_0(\xi, p, q) &= a_1\xi + a_2\xi^2 - (a_1 + a_2)\xi^3 \\ w_1(\xi, p, q) &= 1 - [1 + (1 + p)a_1 - 2q\bar{a}_1 - q\bar{a}_2]\xi - \\ &\quad - [(1 + p)a_2 + 3q\bar{a}_1 + 2q\bar{a}_2]\xi^2 + [(1 + p)(a_1 + a_2) + q(\bar{a}_1 + \bar{a}_2)]\xi^3 \\ F(\xi, p, q, \mathbf{y}) &= w_0(\xi, p, q)y_0 + w_1(\xi, p, q)y_1 + w_1(1 - \xi, q, p)y_2 + w_0(1 - \xi, q, p)y_3 \quad (9) \end{aligned}$$

Coefficients a_1, a_2 are arbitrary functions of p, q. Bar indicates exchanged p and q:

$$\begin{aligned} a_1 &= a_1(p,q) \qquad \bar{a}_1(p,q) \equiv a_1(q,p) \\ a_2 &= a_2(p,q) \qquad \bar{a}_2(p,q) \equiv a_2(q,p) \end{aligned}$$

For $a_1, a_2 = \text{const}$, formula (9) reduces to (1).

If interpolator F should reproduce also quadratic function $y = \xi^2$ (this implies second order accuracy), functions $a_1(p,q)$ and $a_2(p,q)$ are further restricted:

$$2p(p+1)a_1 + q(q+1)\bar{a}_1 + p(p+1)a_2 = -1 \tag{10}$$

Equation (10) is generalization of constraint (2).

Formula (9) enables to classify "traditional" 4-point interpolators in terms of functions $a_1(p,q), a_2(p,q)$. Results are summarized in table 2. Values a_1, a_2 listed in table 1 can be obtained as special case for $p = q = 1$.

$a_1(p,q)$	$a_2(p,q)$	name
0	0	linear interpolator
$-\frac{q+1}{p(p+1)(p+q+1)}$	$\frac{q+2}{p(p+1)(p+q+1)}$	cubic Lagrange polynomial
$-\frac{1}{p} \cdot \frac{4q+3}{4(p+1)(q+1)-1}$	$\frac{1}{p} \cdot \frac{6(q+1)}{4(p+1)(q+1)-1}$	natural cubic spline
$-\frac{1}{p(p+1)}$	$\frac{2}{p(p+1)}$	quasi-cubic spline

Table 2: Classification of "traditional" 4-point interpolators in case with irregular nodes.

Remark:

Dependency of coefficients a_1, a_2 on parameters p, q provides additional degrees of freedom, so that family of cubic 4-point interpolators can be no longer represented by plane. For this reason, weighted combinations of three independent interpolators do not cover whole family, as was the case with regular nodes. Specific consequence is that decomposition of natural cubic spline (spl_n) given by formula (3) is not valid for irregular nodes. Nevertheless, weighted combinations of linear interpolator (lin), cubic Lagrange polynomial (lag) and quasi-cubic spline (spl_c1) generate sufficiently universal 2-parametric family of interpolators, attractive for code implementation.

References

[1] McCalpin, J. D., 1988: A quantitative analysis of the dissipation inherent in semi-Lagrangian advection. *Mon. Wea. Rev.*, **116**, 2330–2336

[2] Purnell, D. K., 1976: Solution of the advective equation by upstream interpolation with a cubic spline. *Mon. Wea. Rev.*, **104**, 42–48

[3] Robert, A., 1993: Bubble convection experiments with a semi-implicit formulation of the Euler equations. *J. Atmos. Sci.*, **50**, 1865–1873

[4] Smolarkiewicz, P. K., and J. A. Pudykiewicz, 1992: A class of semi-Lagrangian approximations for fluids. *J. Atmos. Sci.*, **49**, 2082–2096

[5] Staniforth, A., and J. Côté, 1991: Semi-Lagrangian integration schemes for atmospheric models – a review. *Mon. Wea. Rev.*, **119**, 2206–2223

[6] Váňa, F., 2003: Semi-Lagrangeovské advektivní schéma s kontrolovanou difuzivitou – alternativní formulace nelineární horizontální difuze v numerických předpovědních modelech. *Charles University, Prague, PhD thesis (in Czech with English and French abstracts)*

[7] Váňa, F., 2005: Spline interpolation in semi-Lagrangian advection scheme of ALADIN/ARPEGE/IFS. *RC LACE/CHMI/ONPP internal document*

[8] Váňa, F., P. Bénard, J.-F. Geleyn, A. Simon and Y. Seity, (2006): Semi-Lagrangian advection scheme with controlled damping – an alternative way to nonlinear horizontal diffusion in a numerical weather prediction model. *To be submitted to QJRMS*

[9] Yessad, K., 2005: Semi-Lagrangian computations in the cycle 30 of ARPEGE/IFS. *Météo-France/CNRM/GMAP/ALGO internal document*