Un code AMR (self - Adaptive Mesh Refinement) en différences finies pour les écoulements (magnéto-)fluides : projet en cours

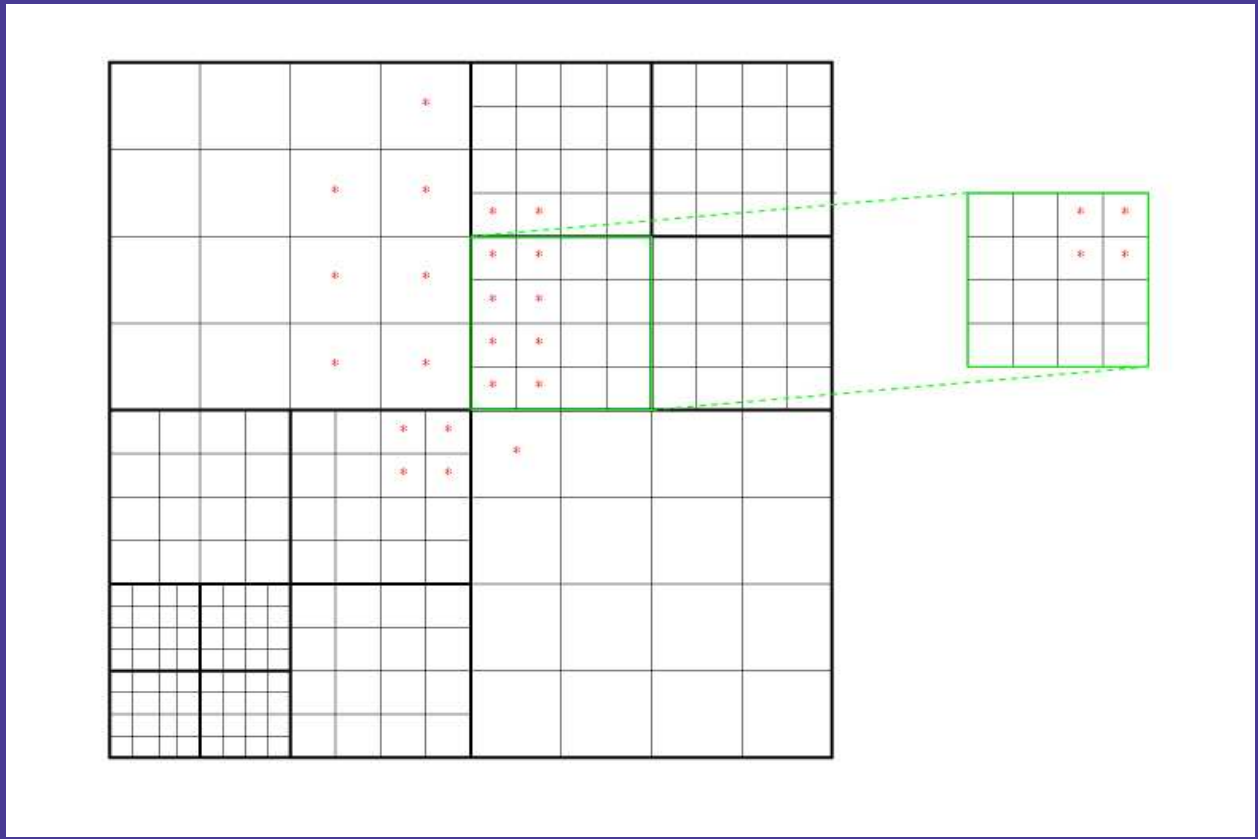
Dimitri Laveder

A. Noullez, H. Politano, Y.Ponty

Laboratoire Cassini - Observatoire de la Côte d'Azur - Nice

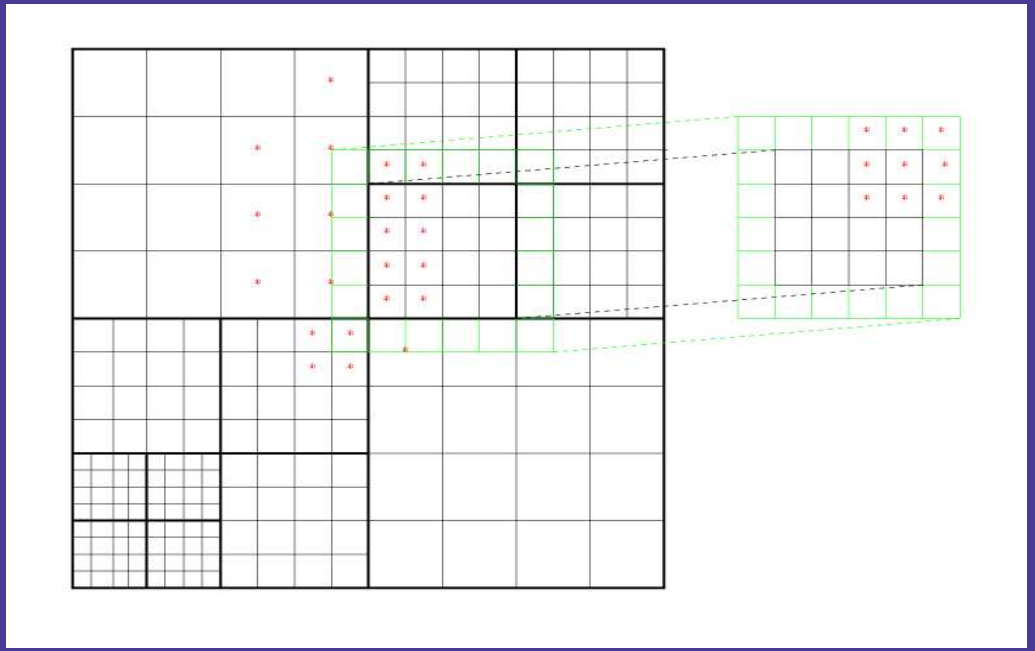
- AMR: integrate a PDE over a grid, and refine the computational grid where it is needed. Overview
- Focus on refinement criterion and interpolation

Our approach: block-structured grid



- Hierarchy of grids at different refinement levels, organized in blocks
- Cartesian mesh inside blocks
- Cell-centered data (or face-centered, or vertex ...)
- Dyadic refinement of a whole block

Integration: blocks as **almost** independent units

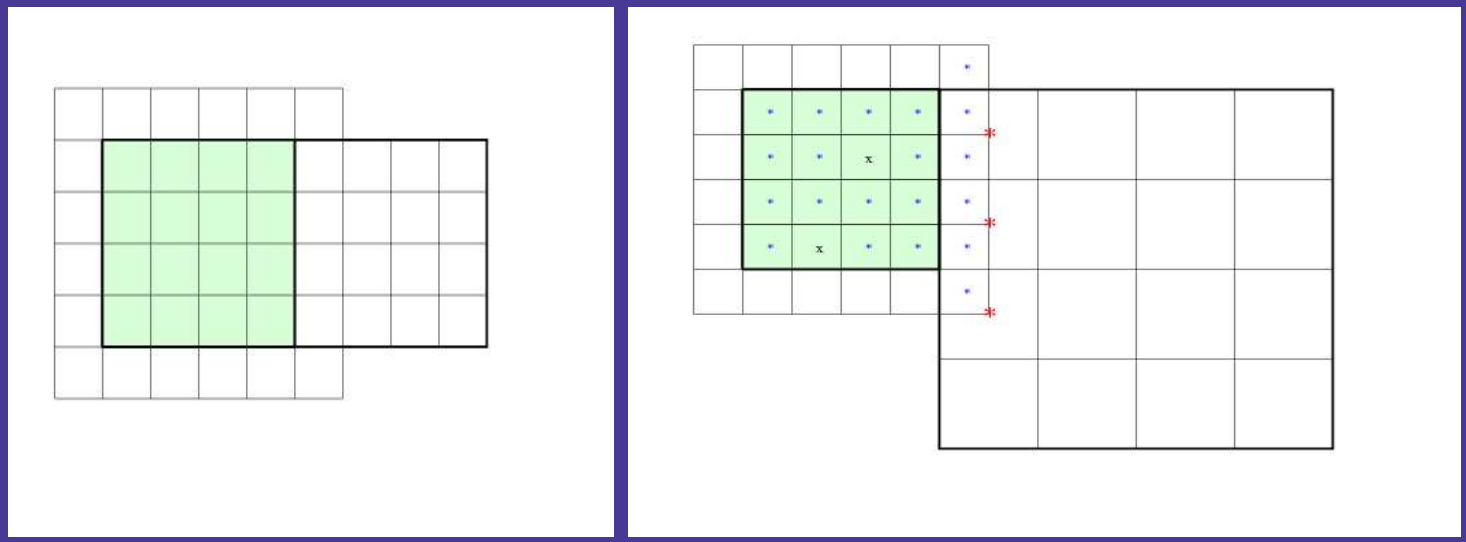


From the point of view of the integration, each block is an independent unit: only boundary informations coming from neighbor blocks need to be provided.

- Integration inside each block with a standard method (... finite differences, finite volumes/conservative FD ...)
- Each block gets boundary conditions from a surrounding buffer zone (guardcells)
- Guardcells are filled from neighbors
- Cumbersome with large stencils ...

Exchanging information between blocks: guardcells

- Neighbors at the same refinement level: copy
- Neighbors at different refinement level : **interpolate**



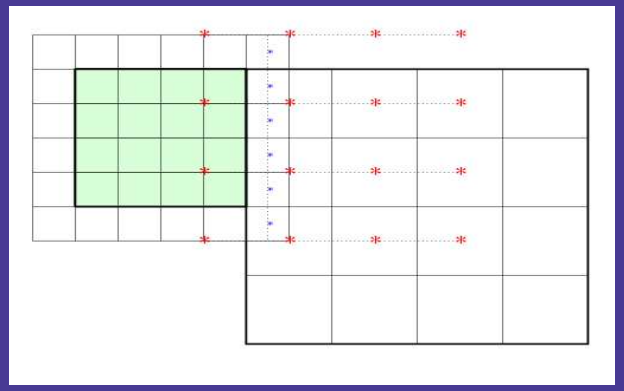
Left: copy. **Right: interpolate from coarser neighbors**

- Interpolation: piecewise polynomial, spline ...
- Time-dependent boundary conditions (each block influences close blocks)
  - Implicit schemes are difficult to use
  - Explicit schemes: possibility of using different timesteps at different refinement levels

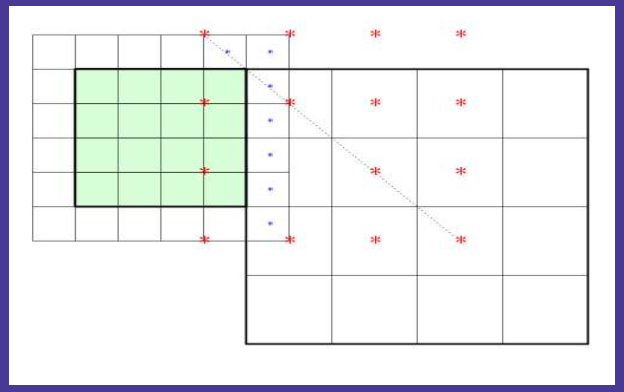
Filling guardcells : interpolate 1

Each block at a given refinement level is covered by a parent at a coarser refinement level

To fill an external layer by interpolation, use parents :



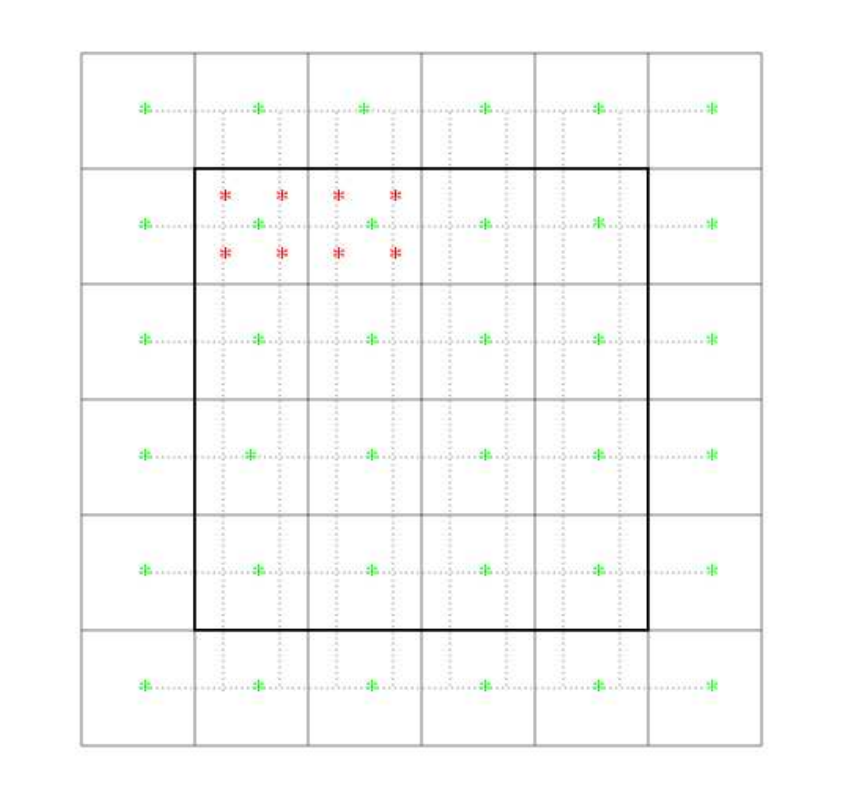
Or interpolate along diagonals?



The higher the order, the higher the interpolation stencil ...

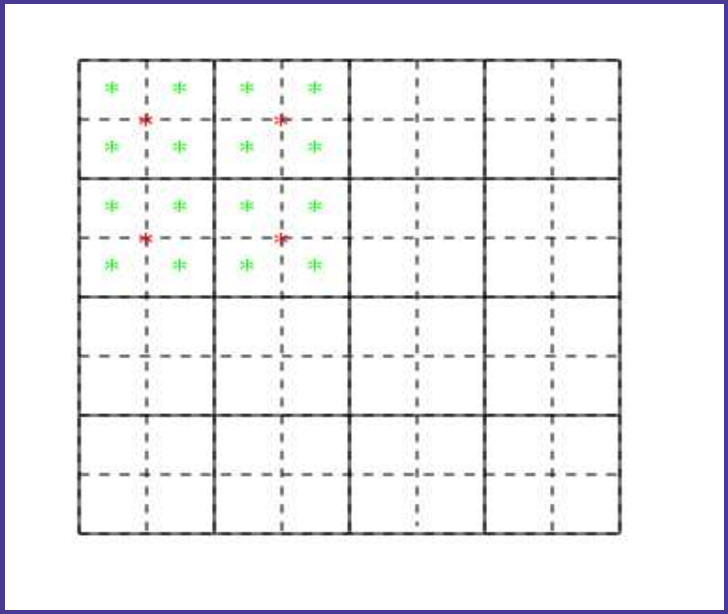
Refinement : interpolate 2

- Avoid extrapolation $\longrightarrow$ to interpolate the interior of a block you need to fill an external layer of guardcells
- Regular grid : interpolate first on one direction and then in the other one



To fill guardcells: you have to interpolate: (interpolate $\Rightarrow$ fill) $\Rightarrow$ (interpolate $\Rightarrow$ refine)

De-refinement : interpolate 3



$\Longrightarrow$ **interpolation : for refinement, de-refinement and guardcells filling.**
Apply in the right order from level to level.

- From piecewise linear to splines: different issues (precision, conservation ...)
- Cell-centered, vertex, face-centered formulation: different procedures. All mixed when using staggered grids (incompressible NS for example)

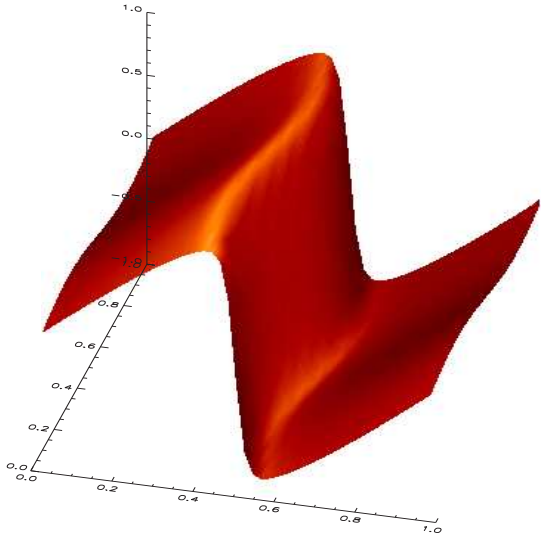
So far, piecewise polynomial with cell-centered data.

A 2D test case: Hwa-Kardar equation

$$\partial_t u + u\partial_x u - \nu_{\|}\partial^2_{xx} u - \nu_{\perp}\partial^2_{yy} u = 0$$

- Time discretization: second - order Runge-Kutta
- Space discretization : second - order centered finite differences (simpler for a test even if weakly unstable)
- Interpolation: linear, piecewise cubic $\Longleftarrow$
- Refinement criterion: define some $Error$.
  - $Error > threshold \longrightarrow$ refine
  - $Error < threshold' < threshold \longrightarrow$ de-refine

  Error: combination of derivatives; local truncation error of the scheme $\Longleftarrow$



Grid1: cubic + $Error = \alpha|\nabla u| + \beta|\kappa_{Gauss}|$

Refinement criterion: local truncation error of the numerical scheme

One can use the full truncation error (time+space discretization) or simply concentrate on the spatial part. With explicit schemes the CFL condition will impose time-refinement anyway.

Truncation error on the **spatial part of the scheme:** a simple example

Diffusion equation: $\boxed{\partial_t u - \alpha \partial^2_{xx} u = 0}$

Take as a starting equation the Euler time-discretized diffusion equation (with more precise, multi-step schemes it works the same way).

$$u_i^{n+1} = u_i^n + \Delta t\, \alpha (\partial^2_{xx} u)_i^n \equiv \mathcal{L}(\Delta t) u|_i^n$$

Now the space-discretized equation reads:

$$v_i^{n+1} = v_i^n + \Delta t\, \alpha \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} \equiv L_{\Delta x}(\Delta t) v|_i^n$$

Truncation error ($u_i^n = v_i^n$):

$$\mathcal{T}_i^n \equiv u_i^{n+1} - L_{\Delta x}(\Delta t) u|_i^n =$$
$$u_i^{n+1} - L_{\Delta x}(\Delta t) v|_i^n =$$
$$u_i^{n+1} - v_i^{n+1} =$$
$$\alpha \Delta t \left[(\partial_{xx}^2 u)_i^n - \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} \right] =$$
$$-\frac{1}{12} \alpha \Delta t \, \Delta x^2 \, \partial_{xxxx}^4 u|_i^n + \mathcal{O}(\Delta x^4) = \mathcal{O}(\Delta x^2)$$

- Interpolate $\longrightarrow$ reduce Δx $\longrightarrow$ get a better approximation for the derivative $\longrightarrow$ reduce the truncation error.
- What precision for interpolation? Enough to **effectively reduce the truncation error** and **preserve the order of the scheme**.

Truncation error and interpolation

Truncation error before interpolation:

$$\alpha \Delta t \left[(\partial^2_{xx} u)^n_i - \frac{v^n_{i+1} - 2v^n_i + v^n_{i-1}}{\Delta x^2} \right]$$

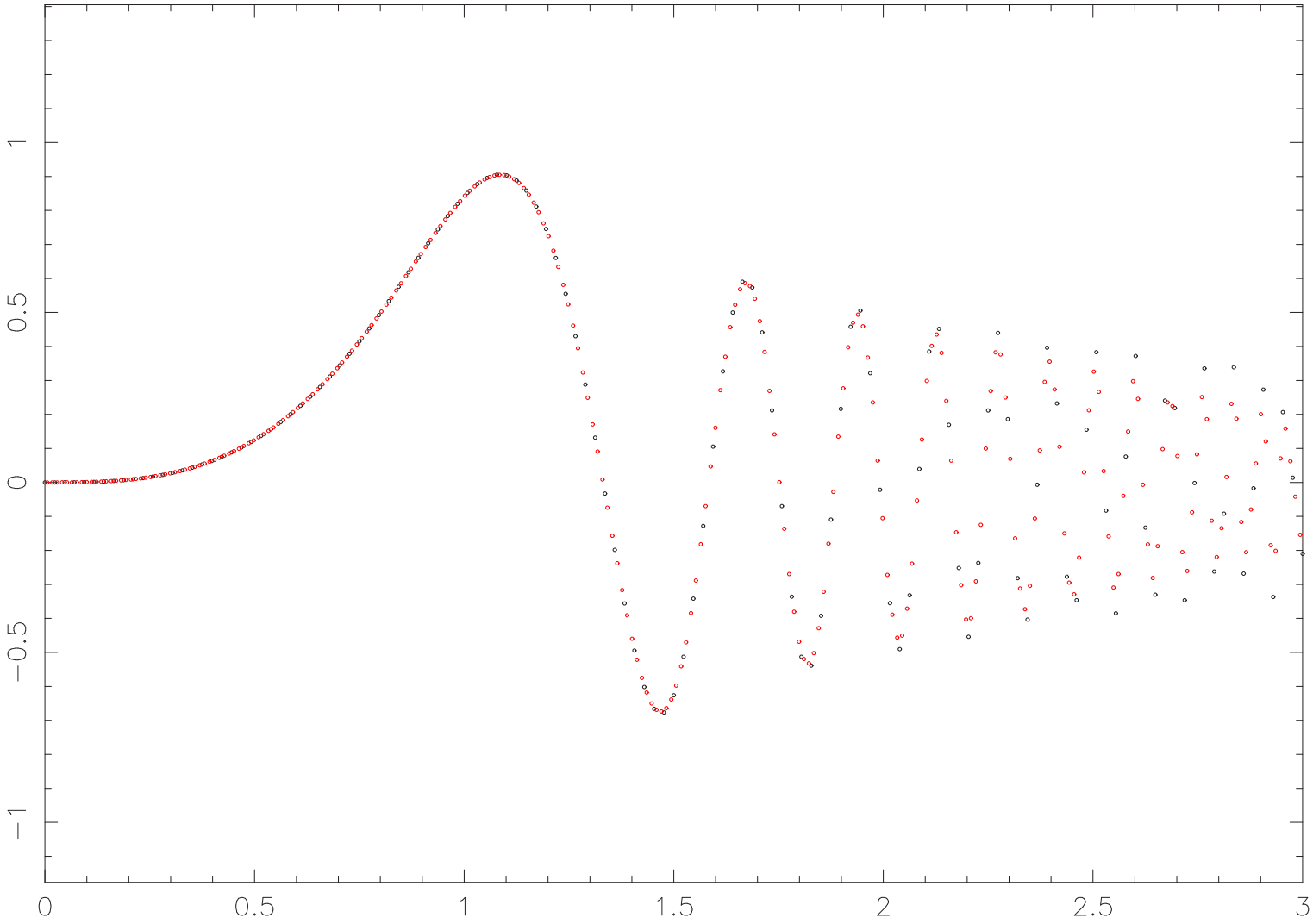
Truncation error after interpolation:

$$\alpha \Delta t \left[(\partial^2_{xx} u)^n_{i+1/2} - \frac{\tilde{v}^n_{i+1} - 2\tilde{v}^n_{i+1/2} + \tilde{v}^n_{i-1}}{(\Delta x/2)^2} \right]$$

($\tilde{v}$ are the interpolated values, not the function values and carry an error)

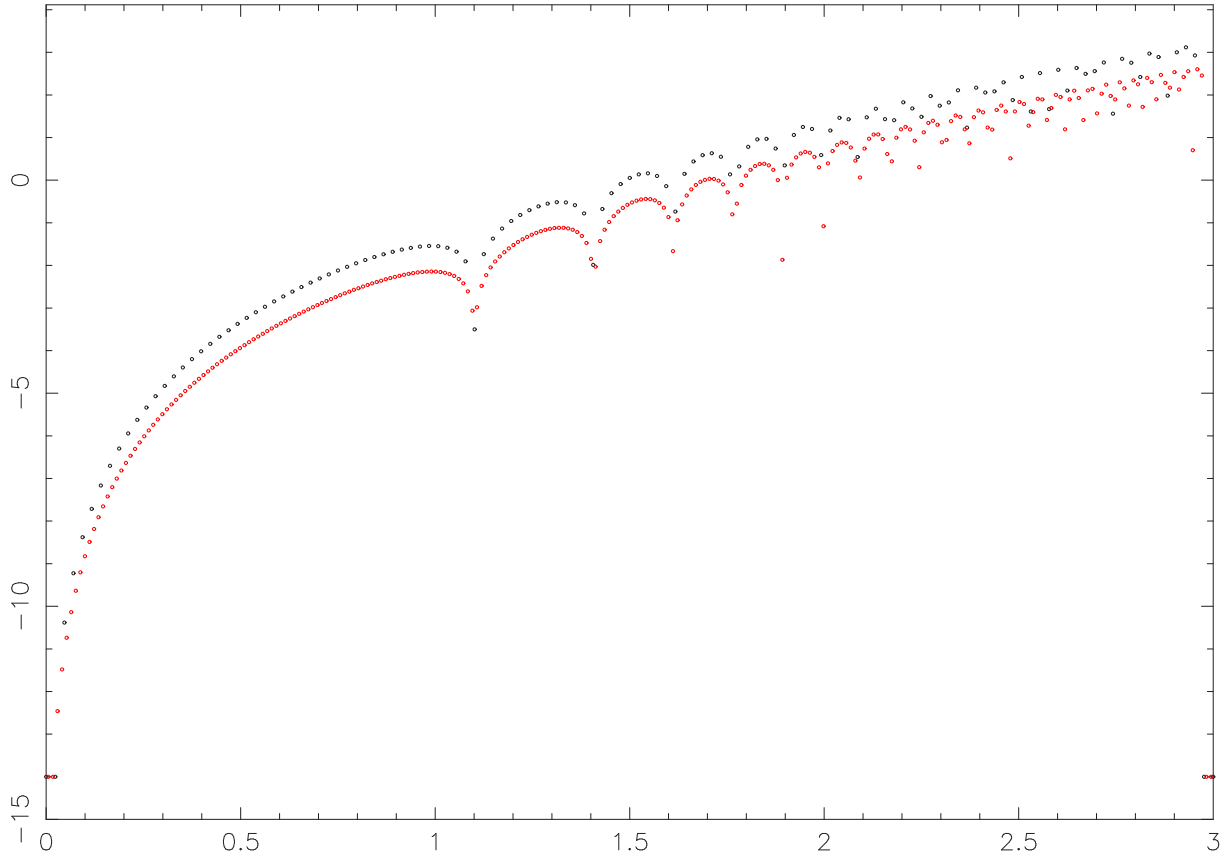
If $\delta_i = v_i - \tilde{v}_i$ interpolation error and $\mathcal{T} = \mathcal{O}(\Delta x^2)$, a minimal requirement is affecting the truncation error with an error of the same order, so at least $\delta = \mathcal{O}(\Delta x^4)$ a priori (cubic piecewise polynomial for example).

Interpolation of $(\sin x)^4/x$



Black : original values. Red : interpolated values (linear interpolation)

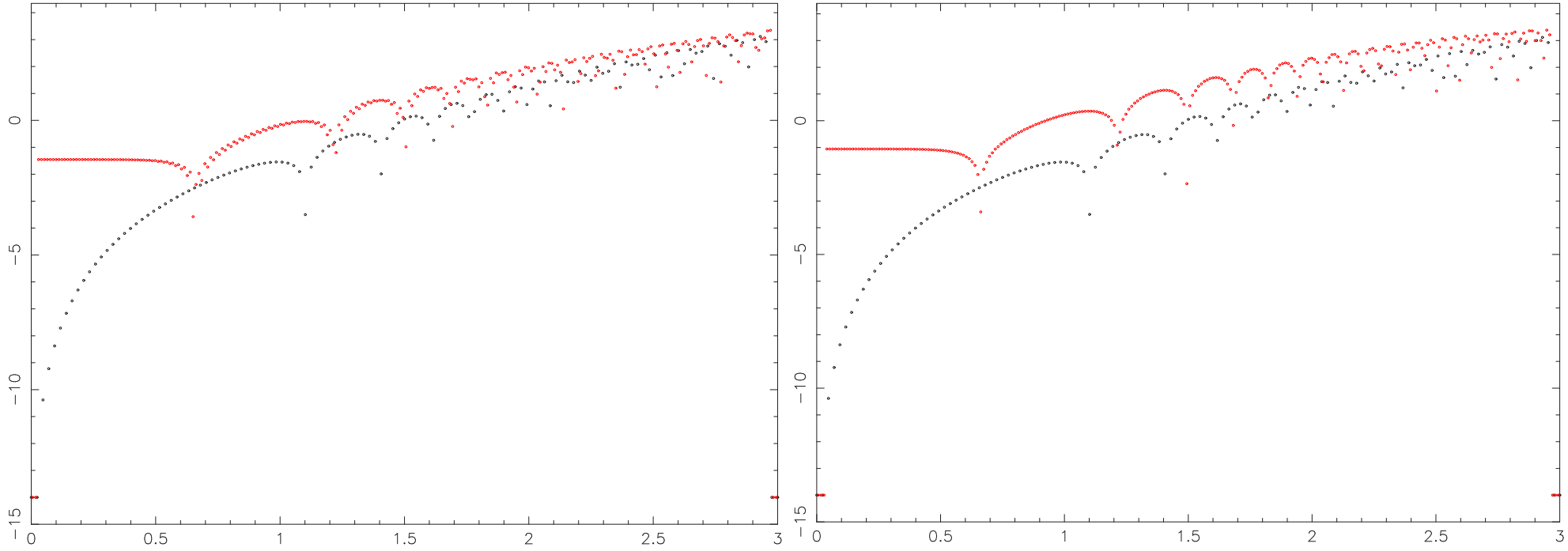
Truncation error at original and interpolated positions with exact function values:



Black: truncation error at the original positions calculated with the exact function values

Red: truncation error at the interpolating postions calculated with the exact function values

Linear and quadratic interpolation : truncation error on the second derivative

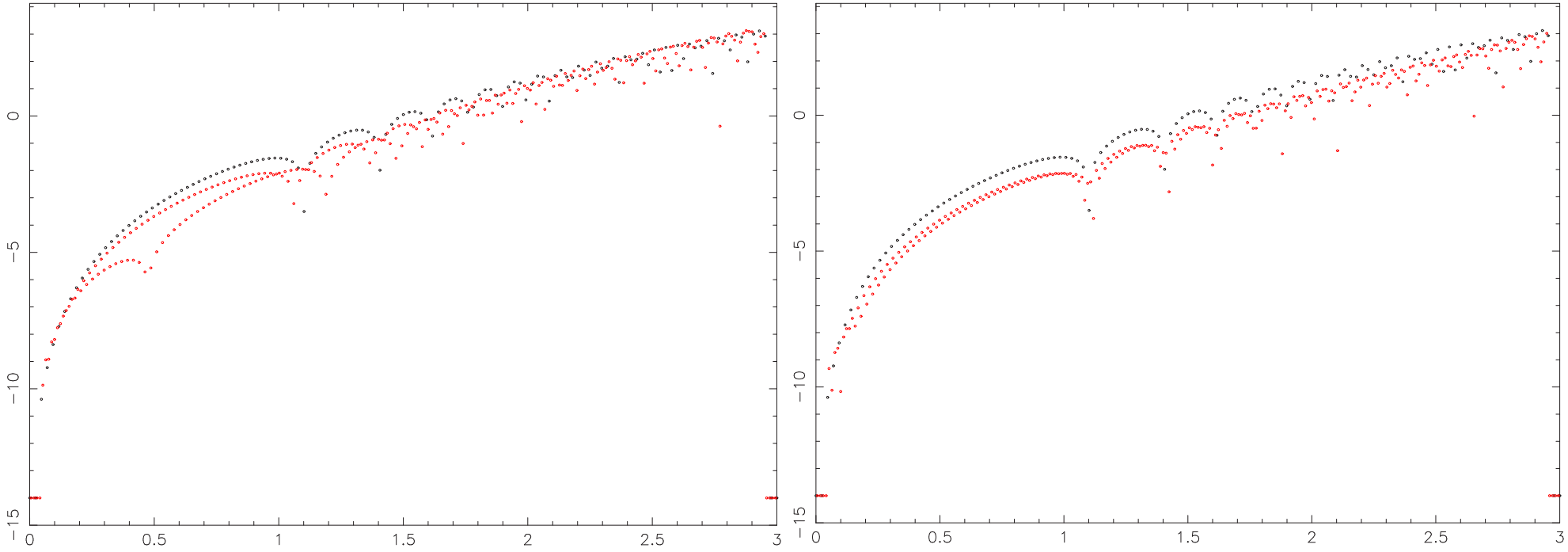


Black: truncation error at the original positions calculated with the exact function values

Red: truncation error at the interpolating postions, calculated with the interpolated values

Left: linear ; Right : quadratic

Cubic interpolation : truncation error on the second derivative



Black: truncation error at the original positions calculated with the exact function values

Red: truncation error at the interpolating postions, calculated with the interpolated values

Left: piecewise polynomial ; Right : spline

Convergence of the interpolated truncation error towards the exact one

Once interpolated, the truncation error has to be estimated with the interpolated values.
A stronger requirement is the convergence of the estimated truncation error towards the true one.

Truncation error before interpolation:

$$\alpha\Delta t \left[(\partial^2_{xx}u)^n_i - \frac{v^n_{i+1} - 2v^n_i + v^n_{i-1}}{\Delta x^2}\right] = \mathcal{O}(\Delta x^2)$$

Truncation error after interpolation (δ_i: interpolation error):

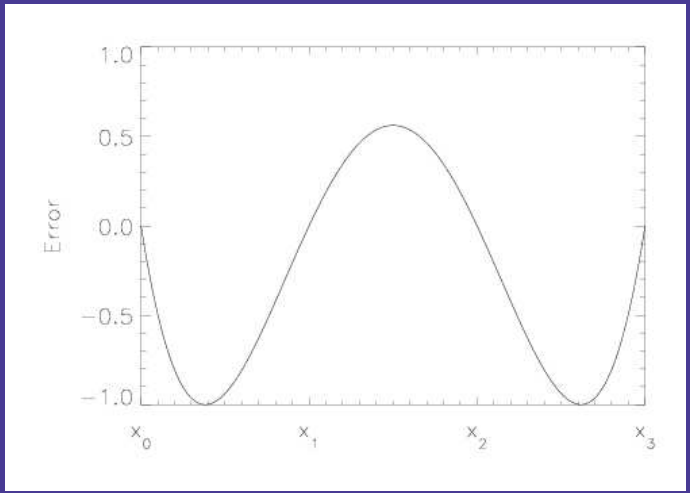
$$\alpha\Delta t \left[(\partial^2_{xx}u)^n_{i+1/2} - \frac{\tilde{v}^n_{i+1} - 2\tilde{v}^n_{i+1/2} + \tilde{v}^n_{i-1}}{(\Delta x/2)^2}\right] = \alpha\Delta t \left[(\partial^2_{xx}u)^n_{i+1/2} - \frac{v^n_{i+1} - 2v^n_{i+1/2} + v^n_{i-1}}{(\Delta x/2)^2}\right] + \frac{\delta_i}{\Delta x^2}$$

$\mathcal{T} = \mathcal{O}(\Delta x^2)$: for convergence you would need at least:
$\delta/\Delta x^2 = \mathcal{O}(\Delta x^3) \Longrightarrow \delta = \mathcal{O}(\Delta x^5)$ (4^{th} degree polynomial), **BUT:**

1. N-polynomial interpolation : error $\mathcal{O}(\Delta x)^{N+1}$ given by :

$$y(x) - P_N(x) = [(N+1)!]^{-1}(x - x_0)(x - x_1)\cdots(x - x_N)y^{(N+1)}(\xi)$$

$\longrightarrow$ N odd: $(x - x_0)\cdots(x - x_N)$ symmetric in the interpolating interval:



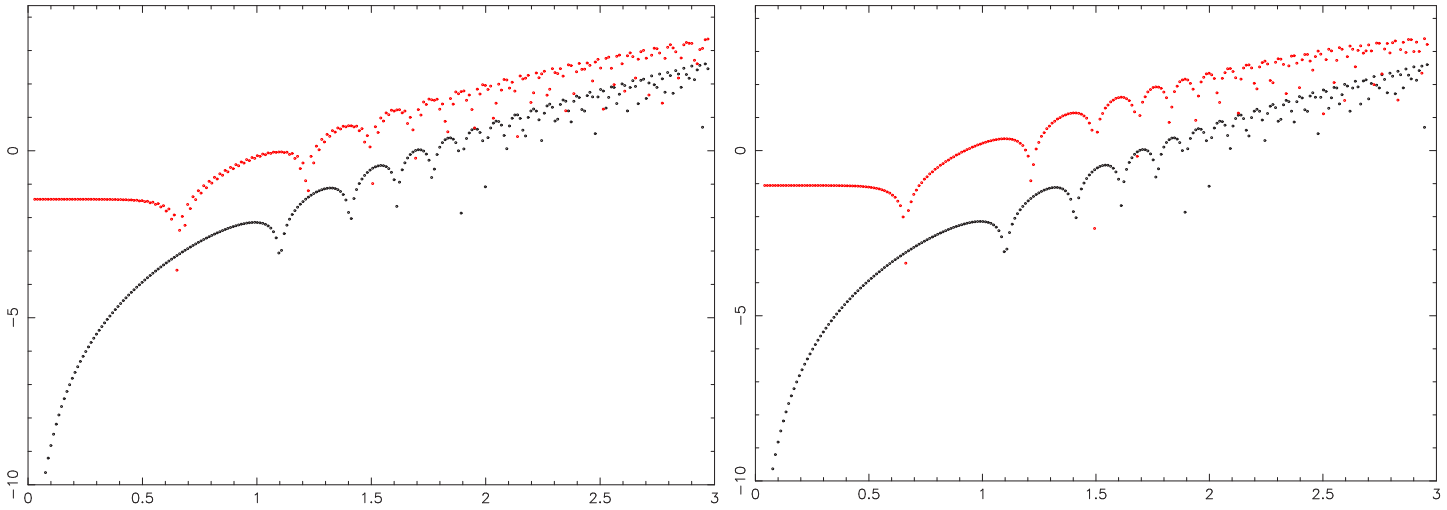
$\longrightarrow$ for $\sum_{j=-p}^{q} C_j \tilde{q}_{i+j}$ the interpolation error is $\mathcal{O}(\Delta x)^{N+2}$ if $\sum_{j=-p}^{q} C_j = 0$

$\Longrightarrow$ with an odd polynomial you gain an order on the FD derivative:

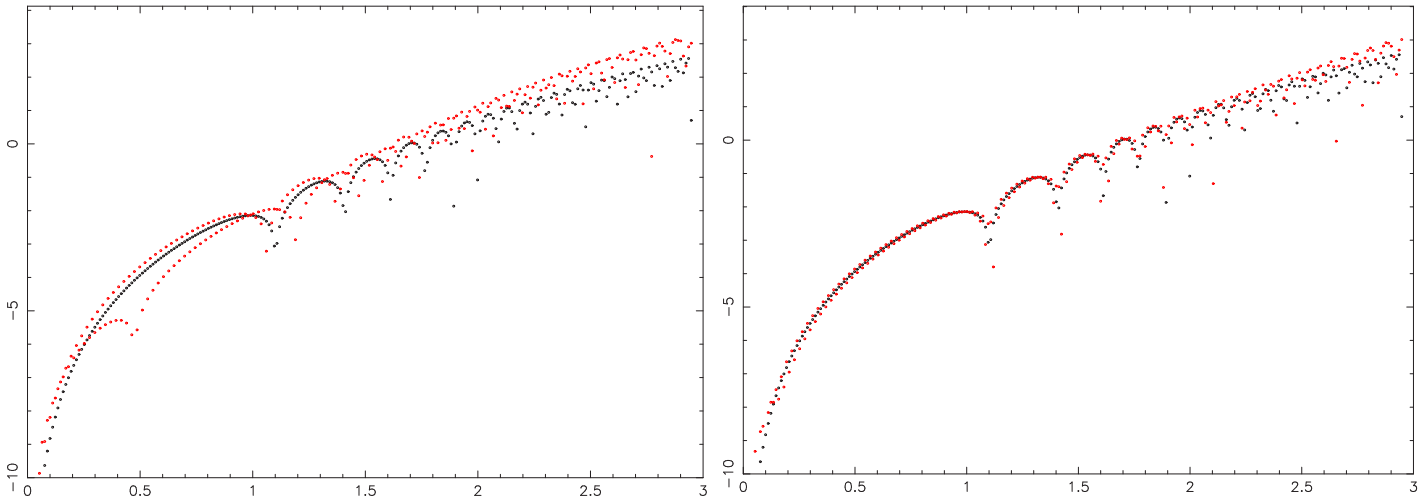
- linear : $\mathcal{O}(\Delta x^3)/(\Delta x^2) = \mathcal{O}(\Delta x)$
- quadratic : $\mathcal{O}(\Delta x^3)/(\Delta x^2) = \mathcal{O}(\Delta x)$
- cubic : $\mathcal{O}(\Delta x^5)/(\Delta x^2) = \mathcal{O}(\Delta x^3)$

2. You can also interpolate the derivatives directly instead of the grid points

Exact truncation error (black) versus interpolated one(red): $(\sin x)^4/x$

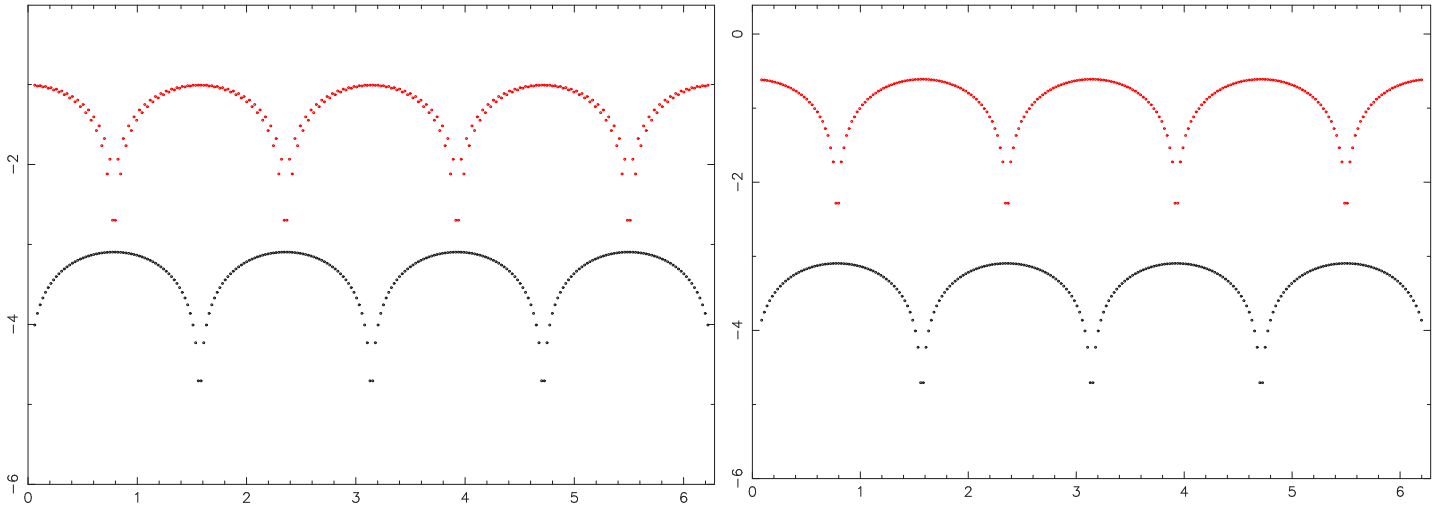


Left : linear piecewise. Right: quadratic piecewise.

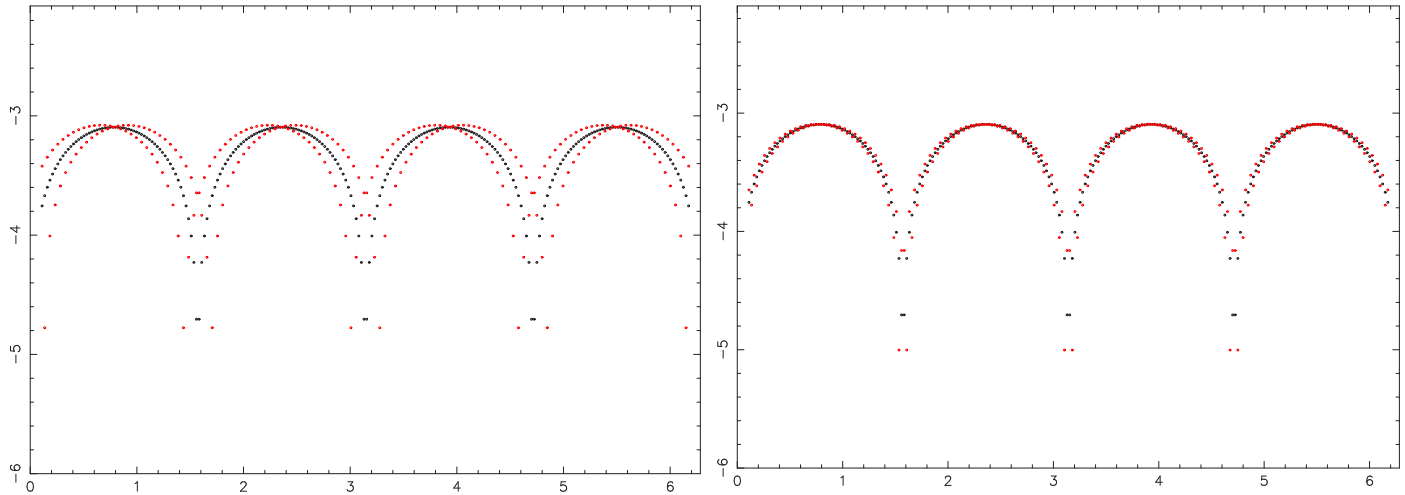


Left : cubic piecewise. Right: cubic spline.

Exact truncation error (black) versus interpolated one(red) : $\sin x$



Left : linear piecewise.Right: quadratic piecewise.



Left : cubic piecewise. Right: cubic spline.

Estimate the leading part of the truncation error : Richardson criterion

$$u_i^{n+1} - L_{\Delta x}(\Delta t) v|_i^n \equiv \mathcal{T}_i^n$$
$$u_i^{n+1} - L_{2\Delta x}(\Delta t) v|_i^n \sim 4\mathcal{T}_i^n$$

$$\mathcal{T}_i^n \sim \mathcal{TR}_i^n \equiv \frac{1}{3}(L_{\Delta x}(\Delta t) v|_i^n - L_{2\Delta x}(\Delta t) v|_i^n) =$$
$$\left(v_i^n + \Delta t\, \alpha \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} \right) - \left(v_i^n + \Delta t\, \alpha \frac{v_{i+2}^n - 2v_i^n + v_{i-2}^n}{(2\Delta x)^2} \right) =$$
$$\alpha \Delta t \left(\delta_{xx}^0(\Delta x) - \delta_{xx}^0(2\Delta x) \right) v|_i^n$$

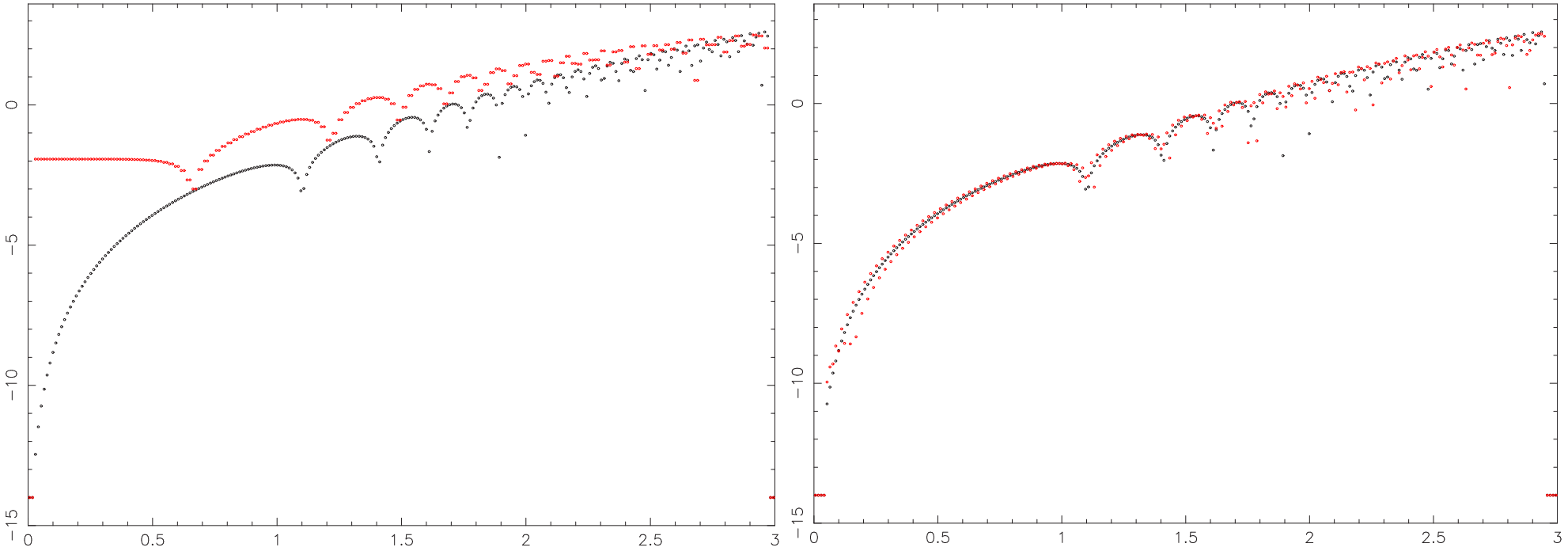
Now use what you have, i.e. the interpolated values:

$$\widetilde{\mathcal{TR}}_i^n = \left(\tilde{v}_i^n + \Delta t\, \alpha \frac{\tilde{v}_{i+1}^n - 2\tilde{v}_i^n + \tilde{v}_{i-1}^n}{\Delta x^2} \right) - \left(\tilde{v}_i^n + \Delta t\, \alpha \frac{\tilde{v}_{i+2}^n - 2\tilde{v}_i^n + \tilde{v}_{i-2}^n}{(2\Delta x)^2} \right) =$$
$$\alpha \Delta t \left(\delta_{xx}^0(\Delta x) - \delta_{xx}^0(2\Delta x) \right) \tilde{v}|_i^n = \mathcal{TR}_i^n + \delta_i$$

$\mathcal{T} = \mathcal{O}(\Delta x^2) \Longrightarrow \delta/\Delta x^2 = \mathcal{O}(\Delta x^3) \Longrightarrow \delta = \mathcal{O}(\Delta x^5)$

$\Longrightarrow$ cubic is enough because of simmetry.

Truncation error calculated with the exact values (black)
versus Richardson estimate calculated with interpolated values (red)



Left : linear piecewise. Right : cubic piecewise

Remarks

In general for a scheme of type:

$$U_i^{n+1} = U_i^n - \alpha \frac{\Delta t}{\Delta x^q}(F_{i+1}^n - F_i^n)$$

after interpolation you get for the estimated truncation error:

$$[\mathcal{O}(\Delta x^p)] + \frac{\delta}{\Delta x^q}$$

(p is the order of the scheme and $\delta = \mathcal{O}(\Delta x^r)$ is the interpolation error).

To properly estimate the truncation error you need at least $\delta = \mathcal{O}(\Delta x^{p+q+1})$ (for example if $q = 1$,$p = 2$ and you interpolated directly F instead of U you need a cubic polynomial).

Conclusions

- The project of an AMR, block-structured code for fluid-like system is growing.
- We aim at incompressible hydro and MHD, and problems with localized structures.
- So far, physical refinement criteria and piecewise polynomial interpolation have been merged in a working code.
- The focus now is on criteria based on the evaluation of truncation errors. The kind of interpolation to use is strictly dependent on these criteria and on precision (and conservation) constraints. Moreover, is the continuity of the interpolant (i.e. splines) necessary?
- More work on the way: algorithms, tests of accuracy and convergence, parallelization ...