Curvature computation in Volume of Fluid

Stéphane Zaleski[1], Ruben Scardovelli[2] & Yue Ling[1]

1. Institut Jean Le Rond d'Alembert,
CNRS & Université Pierre et Marie Curie -- UPMC – Paris 6

2. DIENCA, Laboratorio di Montecuccolino, Universita di Bologna

web site http://www.ida.upmc.fr/~zaleski

VOF methodology

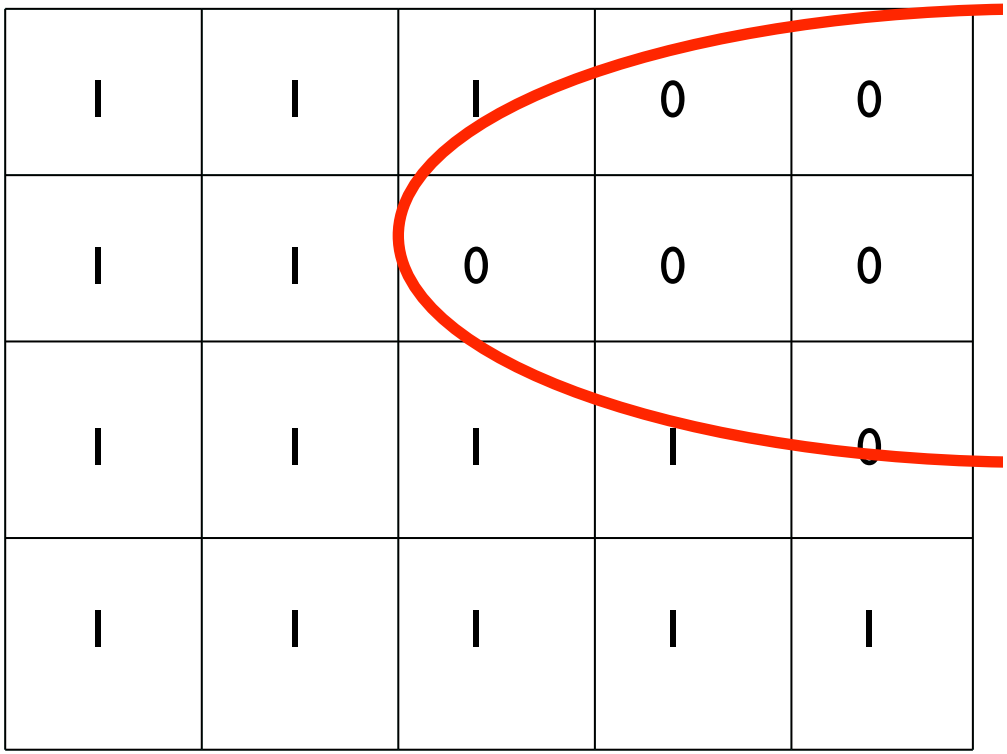
The never-ending attraction of simple ideas :

The simplest discrete representation of a two-phase flow

I	I	I	0	0
I	I	0	0	0
I	I	I	I	0
I	I	I	I	I

The never-ending attraction of simple ideas :

The simplest discrete representation of a two-phase flow



This is the idea in Ising models, lattice-gas liquid-gas or two-color models, and also an idea of Launder for two-phase flow.

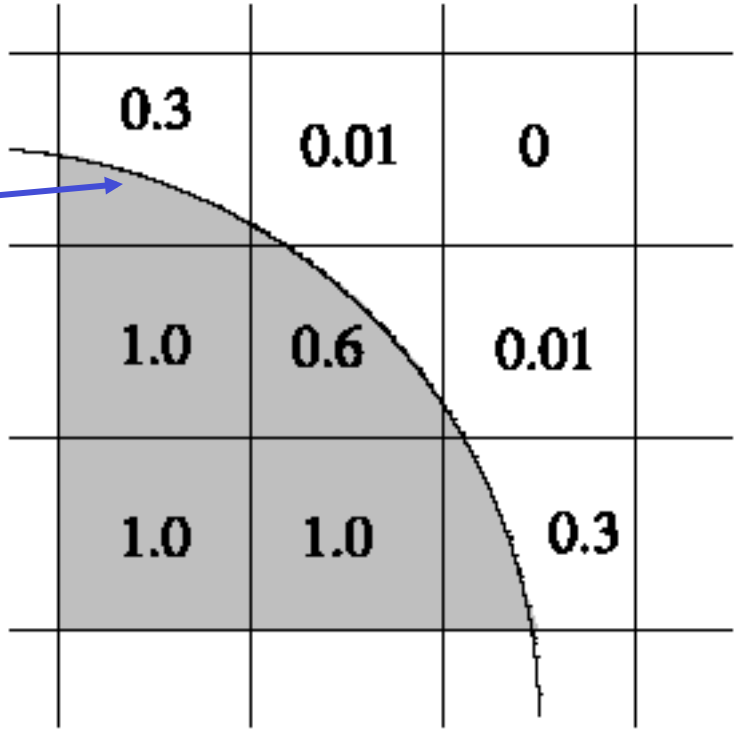
But it does not work very well. The interface is too noisy,
the curvature is poorly computed,
surface tension computation is plagued by large errors.

Make the number in each cell a real number marking the cell phase.

I	I	I	0	0
I	I	0	0	0
I	I	I	I	0
I	I	I	I	I

The Volume-of-Fluid method

True interface



C_{ij} = Volume of « fluid » in cell ij

A bit of history

Second-order (linear) reconstruction methods:
de Bar, Kraken code, 1974

Momentum conserving
Rudman 1996

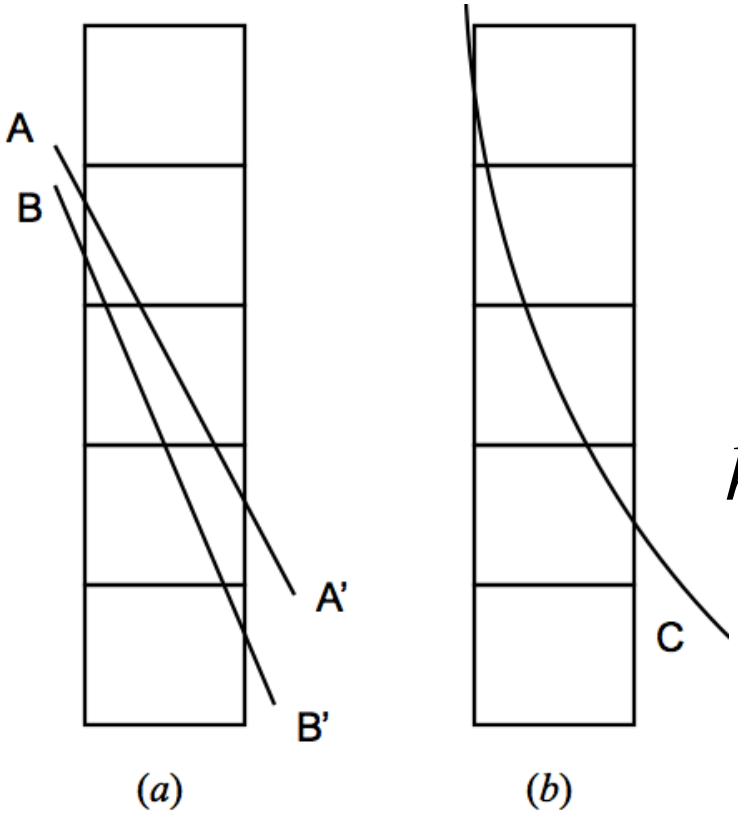
Height-Function methods yield better surface tension & curvature:
Sussman, 2003 , Popinet 2009

Coupled methods : with Level Sets or Front Tracking
Sussman, 2003, Aulisa, Manservisi, Scardovelli 2003

Exactly (machine accuracy) mass conserving in 3D
Weymouth & Yue 2010

Exactly mass, momentum and energy conserving
Le Chenadec, to appear.

Height function method



$$H_{j,k} = \sum_{i=-n}^{i=n} C_{i,j,k}$$

$$h = H_{j,k} - \frac{1}{24}\frac{\partial^2 h}{\partial x^2}(\Delta x)^2 + O(\Delta x^4)$$

A quick estimates says that finite differences of heights give second derivatives to zeroth order. But notice that there are 3 different kinds of heights: x, y, and z.

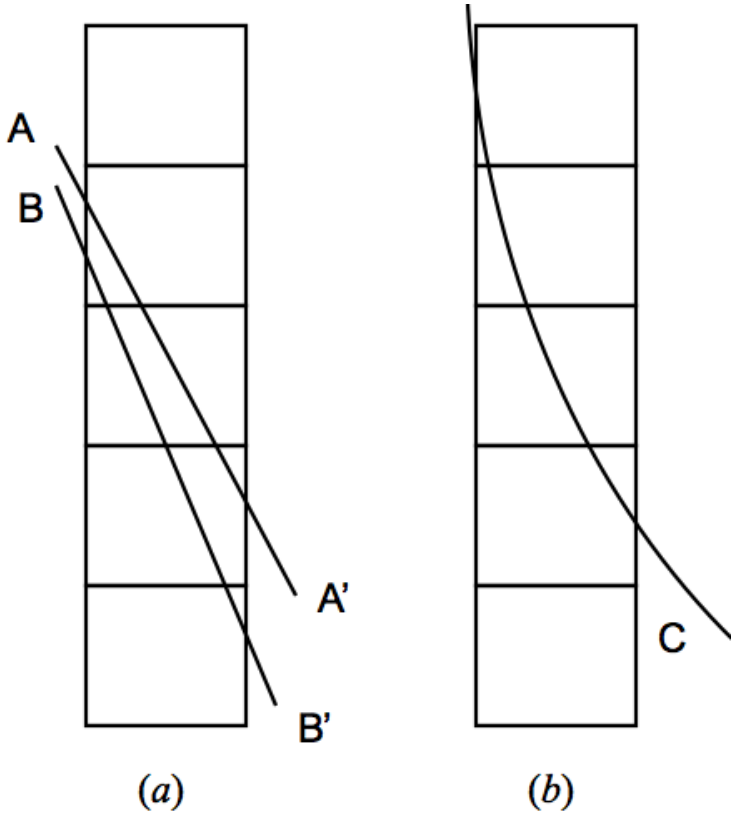
When using three consecutive heights of the same kind, there is a cancellation and the result is still of order 2

Define the first and second derivatives by finite differences. Compute the curvature. There is a cancellation and the result for cirvature and second derivatives are still second order !!

In 2D the succcess of the height function method depends on the slope of the line. For 45 degree lines it is most difficult to get a « good » case with three consecutive height functions of the same kind.

There are two ways out:

1) Increase the number n that controls the height of the « stack » of points.
2) use « mixed » heights with different kinds of heigts x, y and z.



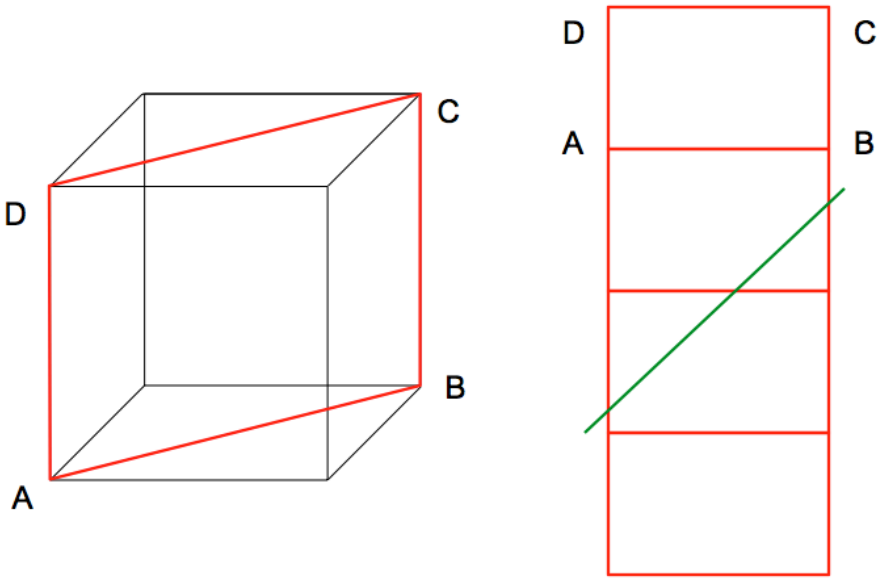
1) can only do so much: if the curvature us high, extending the height of the stencil doesn not help.

In 2D, a height of 5 cells should be sufficient for all straight lines, so 7 cells are used to account for curved interfaces.

2) Using mixed heights reduces the order of the approximation.

In 3D the most dangerous case is the direction **n** = (1,1,1).

This requires to look in the « diagonal » plane. A stack of diagnonal plane shouls have a height of 5 for planar interfaces thus 7 to account for curvature and a thus a stencil of 9 x 3 x 3 is needed accounting for the full and empty cells at top and bottom.



A third option is to use centroids:

- find the normal direction
- reconstruct the interface cutting the cube
- find the center of mass of the interface in the cube.

Then find the elliptic paraboloid-(*)that fits through all the centroid points in a 3 x 3 x 3 assembly of cells. The elliptic paraboloid is

$$h = a + bx + cy + dx^2 + exy + fy^2$$

It requires 6 points so 6 centroids at least are needed.

The full method used by Gerris is

1) attempt to find nine heights of the same kind
2) if not: Find mixed heights
(mixing height of x, y and z types). Avoid height that give almost the same point (points less than dx/2 of each other). If 6 heights found: rotate the coordinate system. fit the elliptic paraboloid.
3) If not working: find centroids (still rotating the coordinate system). Fit the elliptic paraboloid
4) cap the curvature to some maximum value.

Simplified method:

1) attempt to find nine heights of the same kind
2) if not: find centroids (do not rotate the coordinate system). Fit the elliptic paraboloid.
3) cap the curvature to some maximum value.

How to test this ? Initialize randomly a large number N of disks or spheres.

Problem: initialization must be exact. An approximate initialzation would yield randome $O(dx^2)$ errors that would destroy the curvature computation.

Exact initialization is not available in Gerris. So we initialize on a much smaller subgrid (typicall 8 times smaller in each directions, so 512 more computations to do).

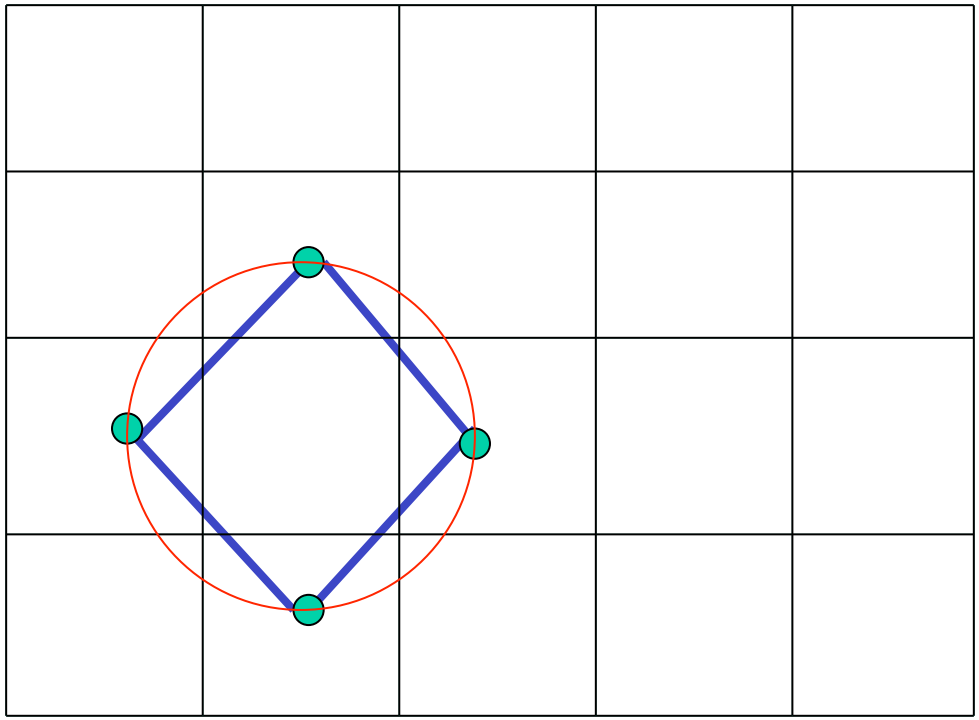
```
Define ABS(x) (((x) > 0 ? (x) : (-x) )
#Define CIRCLE (ellipse (-RADIUS,RADIUS,RADIUS,RADIUS))  apparently not better
Define CIRCLE (x*x + y*y + z*z - RADIUS*RADIUS)


1 0 GfsSimulation GfsBox GfsGEdge {} {
  Time { iend = IEND }
  Refine ((LEVEL+INIT))

  AdvectionParams { scheme = none }
#  VariableTracerVOFHeight T  # apparently identical in 2D
  VariableTracerVOF T
  VariableCurvature K T

  InitFraction {} T CIRCLE

  AdaptGradient { istep = 1 } { cmax = 1e-6 minlevel = ( t > 0.5 ? LEVEL : (LEVEL + INIT))  maxlevel = (
 t > 0.5 ? LEVEL : (LEVEL + INIT)) } T

  OutputSimulation { step = 1 }  curvature-ft-%g-LEVEL-RADIUS-D.gfs  { depth = 5 }
  OutputSimulation { istart = 0 istep = 1 } curvature-%ld-LEVEL-RADIUS-D.gfs

  OutputScalarNorm { start = end } {  awk '{ print "L2 norm ",$7*RADIUS/2., "Linfty norm ",$9*RADIUS/2.
}' } { v = ( K - (D-1)/RADIUS) }
  OutputScalarStats { istart = 1 istep = 1 } stderr { v = K }
}
GfsBox {
    left   =  Boundary   { BcDirichlet T 0 }
    right  =  Boundary   { BcDirichlet T 0 }
    top    =  Boundary   { BcDirichlet T 0 }
    bottom =  Boundary   { BcDirichlet T 0 }
#    front =  Boundary  { BcDirichlet T 0 } # uncomment for 3D runs
#    back =   Boundary  { BcDirichlet T 0 }
 }

--:--  curvature.gfs   All (1,0)      (Gerris[bash])----------------------------------------------------------
Beginning of buffer
```

Typically let the refinement decrease during iend=2-3 time steps.

This is extremely time consuming (hours): no time to initialize several droplets.

To accelerate the initialization, Ruben Scarodvelli and Simone Bna wrote a VOF initialization library called « Vofi » available at the same web site as parissimulator: http://parissimulator.sf.net

The continuous,
piecewise linear
polygonal line in 2D
or a polyhedron in 3D
obtained with elementary
constructions can be very different
from the exact surface with
some cells completely missed.



Internals of the Vofi library (Bnã, S, S Manservisi, R Scardovelli, Ph. Yecko, and S Zaleski. Computers and Fluids, 2014, online, doi:10.1016/j.compfluid.2014.04.010.)

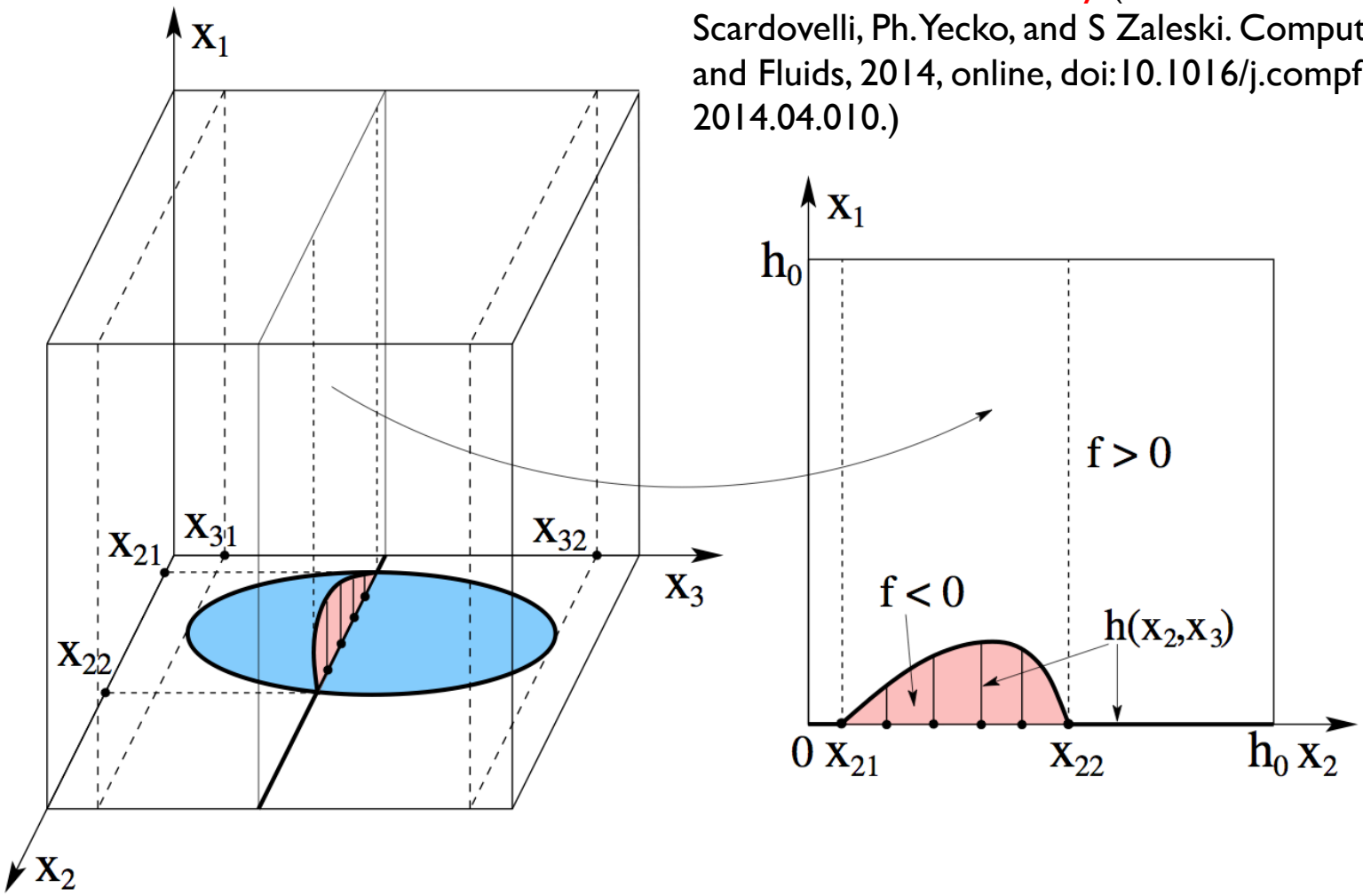
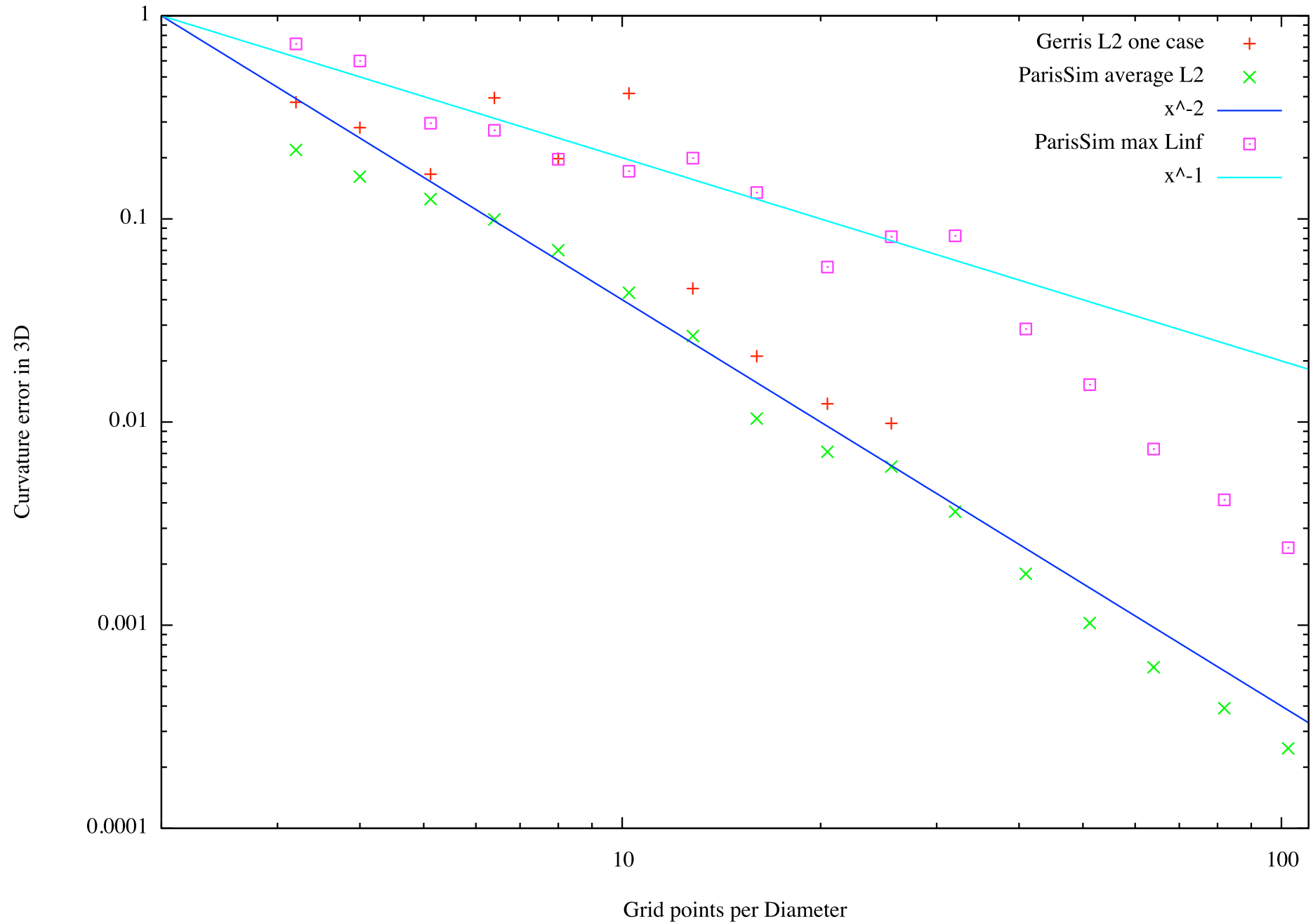


Figure 1: The cubic cell is subdivided in three right hexahedra after the computation of the two external limits of integration x_{31} and x_{32}, with the interface being present only in the central one (left); the internal integration requires the computation of the two internal limits of integration x_{21} and x_{22} and of the local height function h (right)

Initialize the spheres and measure L_1 and L_2 norms of the errors (norms are computed by summing over all points of a given sphere and all random spheres, except for gerris where a single sphere was initialized because of the cost).

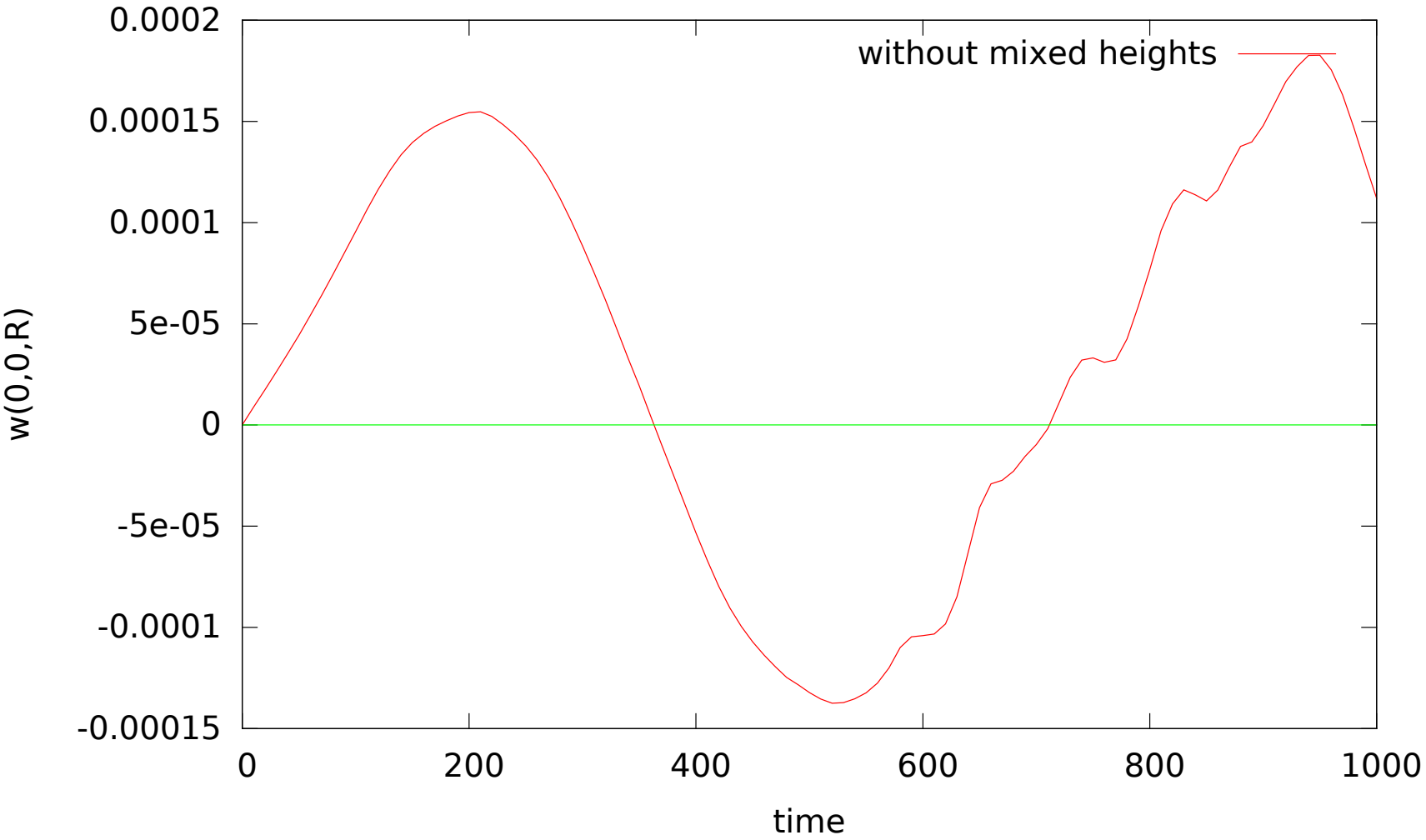
First show the results in
3D for the simplified algorithm (no mied heights)

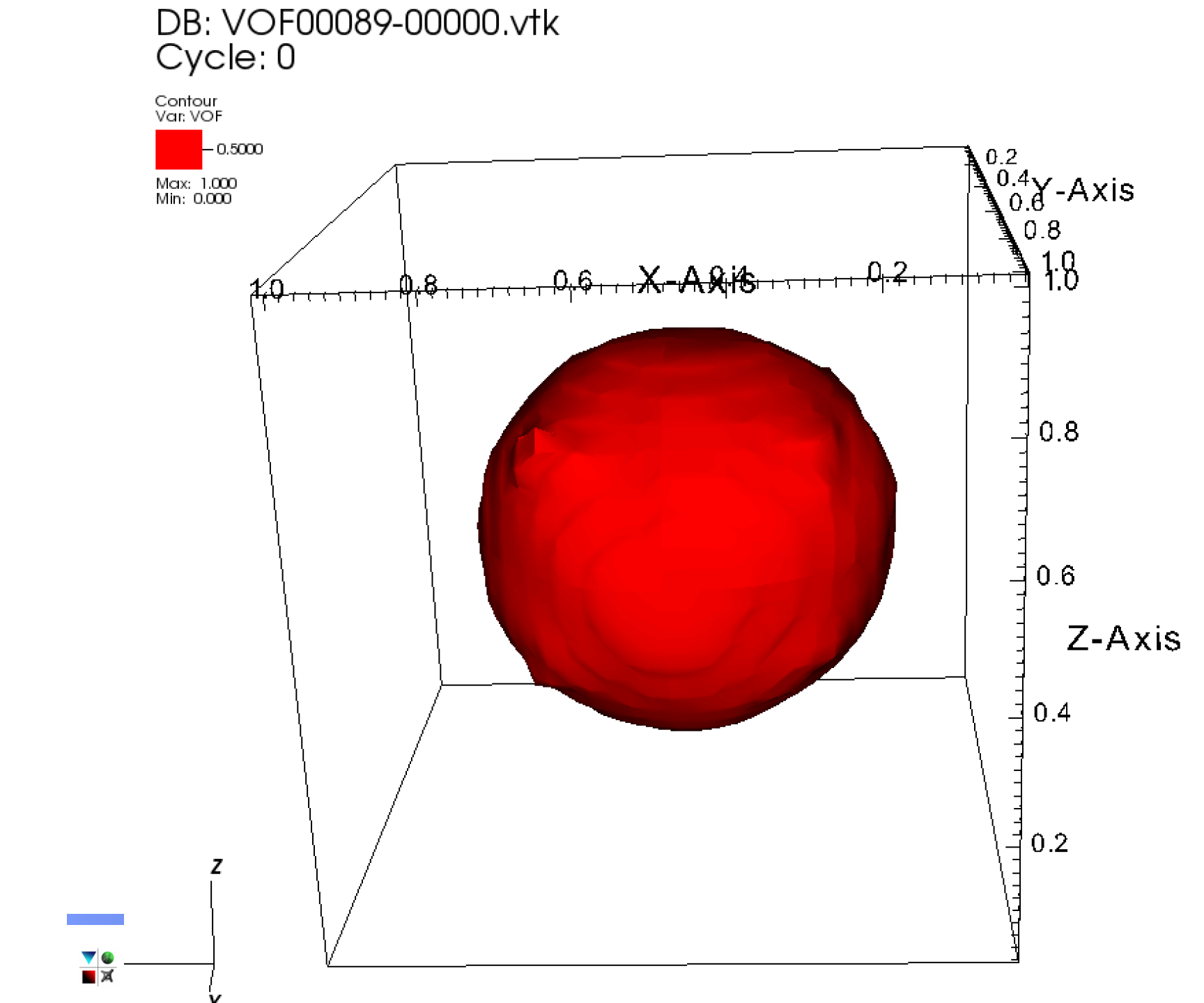

Gerris L2 one case
ParisSim average L2
x^-2
ParisSim max Linf
x^-1
Curvature error in 3D
Grid points per Diameter
1
0.1
0.01
0.001
0.0001
10
100

Then use the curvature method to find droplet oscillations.

Use a 3mm droplet of water in air. Large density ratio,
Laplace number La=216,000, density ratio $r=1.2\ 10^{-3}$
viscosity ratio m=0,017, points per diameter D/dx = 19.

The capillary oscillation computation fails.

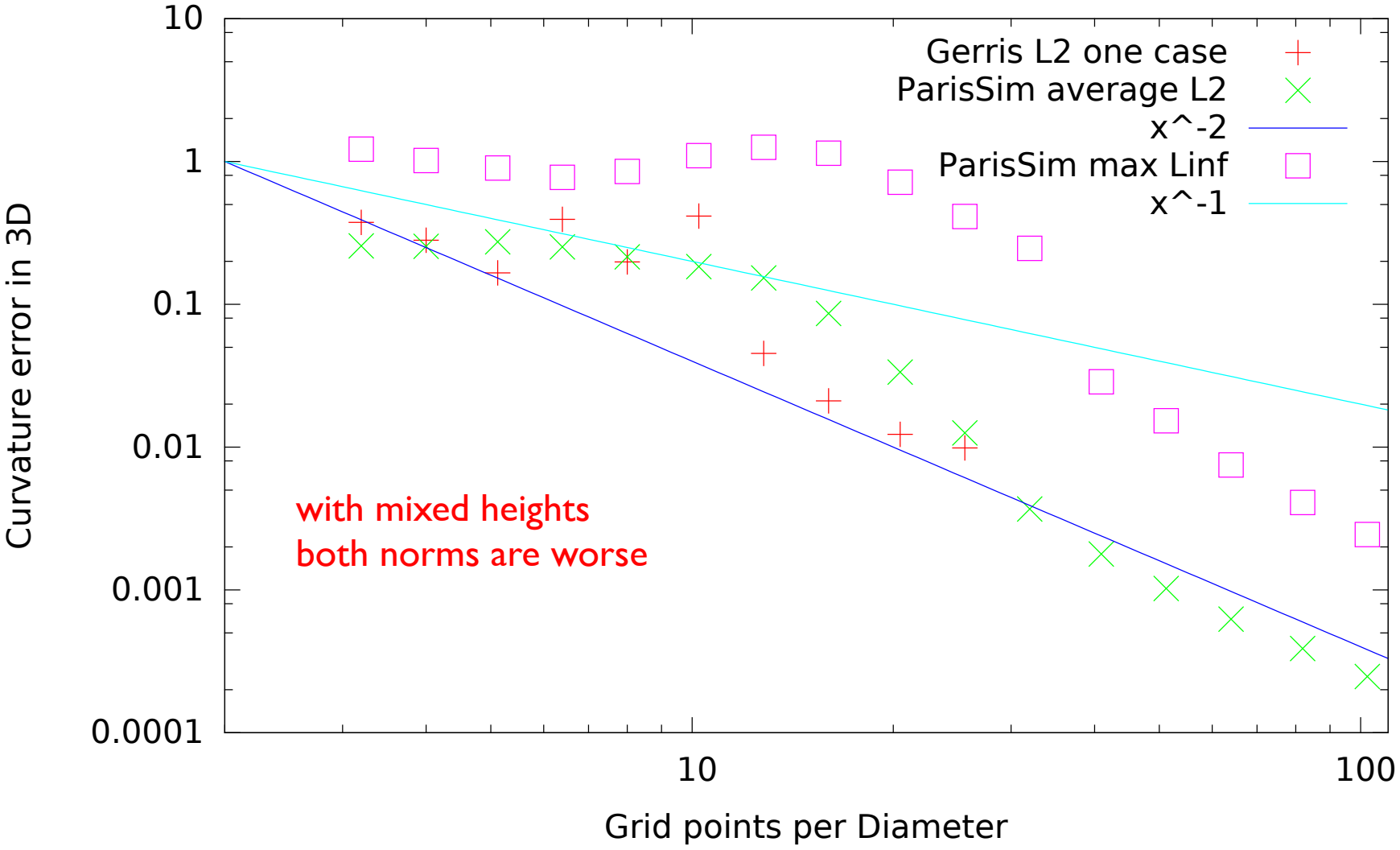

without mixed heights
0.0002
0.00015
0.0001
5e-05
0
-5e-05
-0.0001
-0.00015
w(0,0,R)
0
200
400
600
800
1000
time

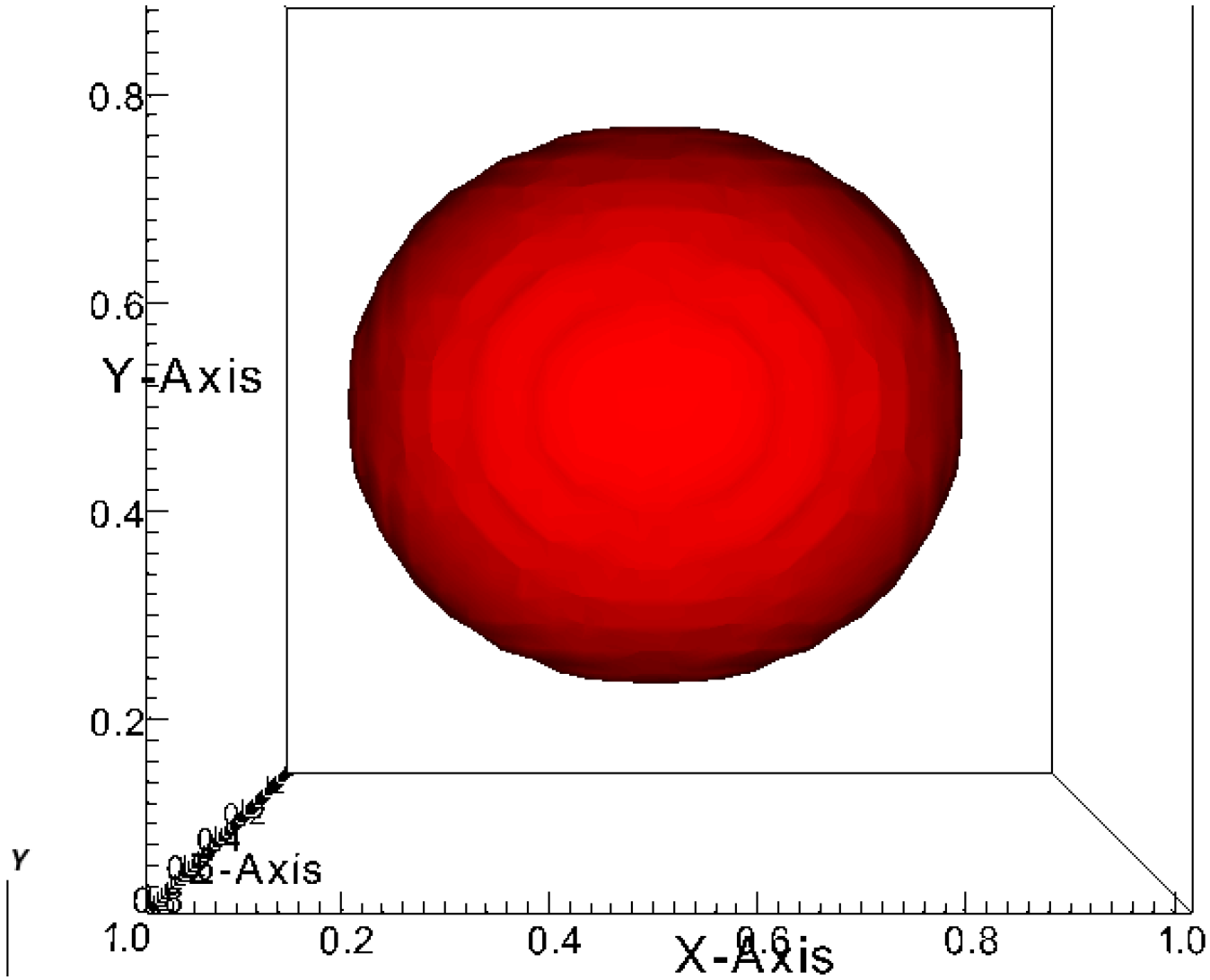

DB: VOF00089-00000.vtk
Cycle: 0
Contour
Var: VOF
0.5000
Max: 1.000
Min: 0.000
1.0
0.8
0.6
X-Axis
0.2
0.2
0.4
0.6
0.8
1.0
Y-Axis
1.0
0.8
0.6
Z-Axis
0.4
0.2
Z
Y

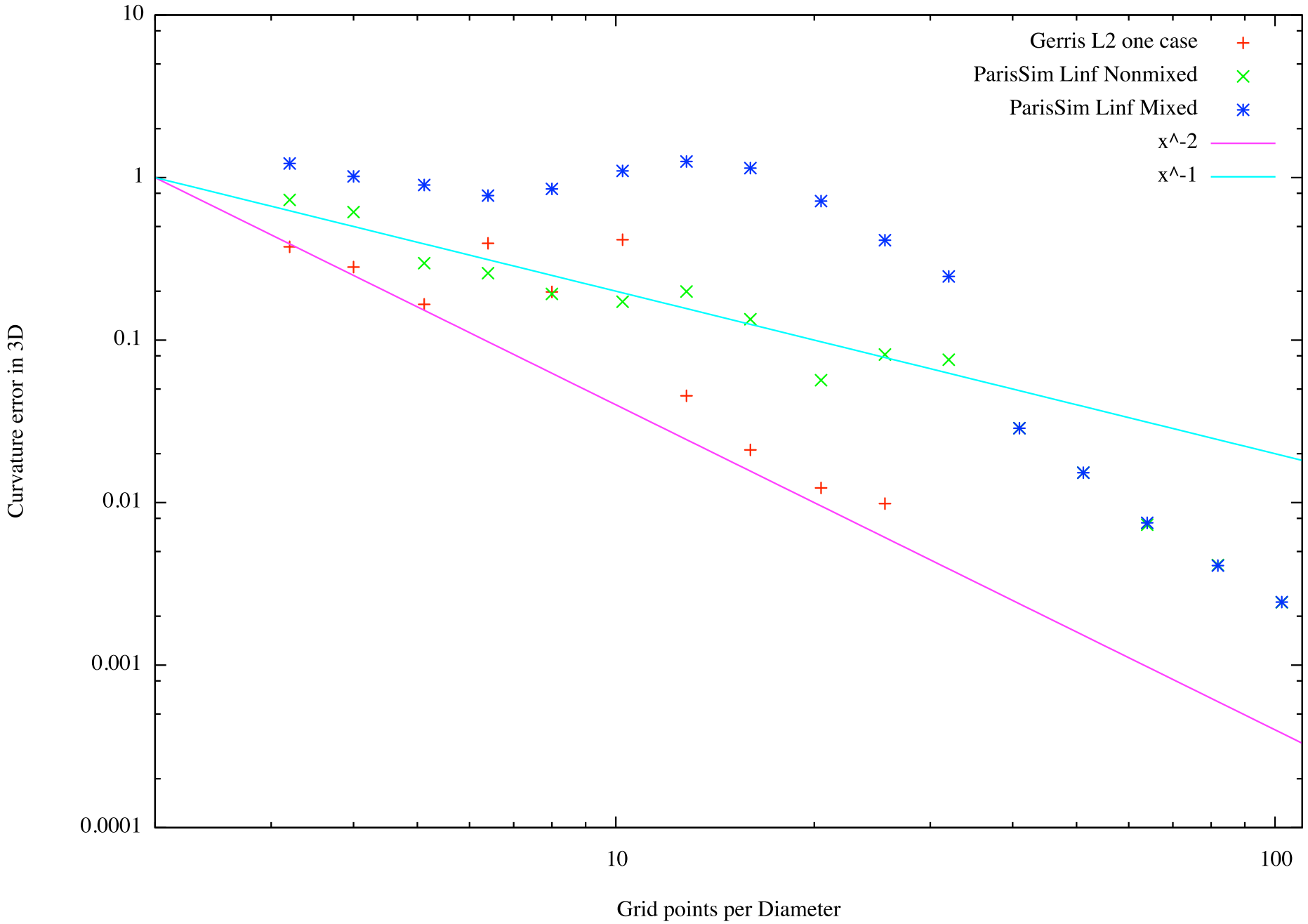
However, reinstating the mixed heights, that is

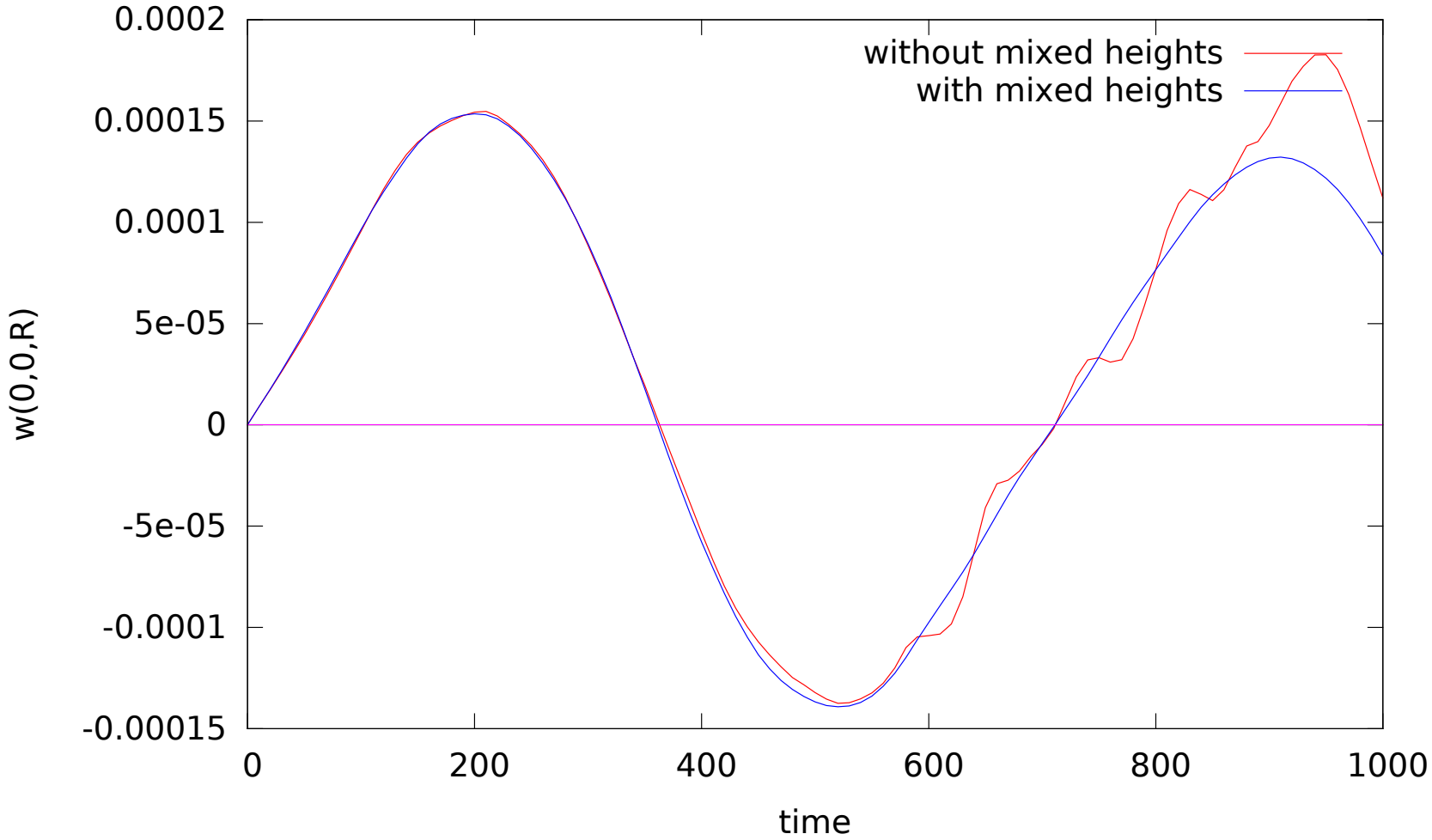
1) attempt to find nine heights of the same kind
2) if not: find mixed heights
(mixing height of x, y and z types). Avoid height that give almost the same point (points less than dx/2 of each other). If 6 heights found: fit the elliptic paraboloid.
3) If not working: find centroids (still rotating the coordinate system). Fit the elliptic paraboloid
4) cap the curvature to some maximum value.

yields worse results for the curvature but better results for theoscillations


Gerris L2 one case
ParisSim average L2
x^-2
ParisSim max Linf
x^-1
with mixed heights
both norms are worse
Curvature error in 3D
Grid points per Diameter
10
1
0.1
0.01
0.001
0.0001
10
100


0.8
0.6
Y-Axis
0.4
0.2
Y
Z-Axis
1.0
0.2
0.4
0.6
0.8
1.0
X-Axis

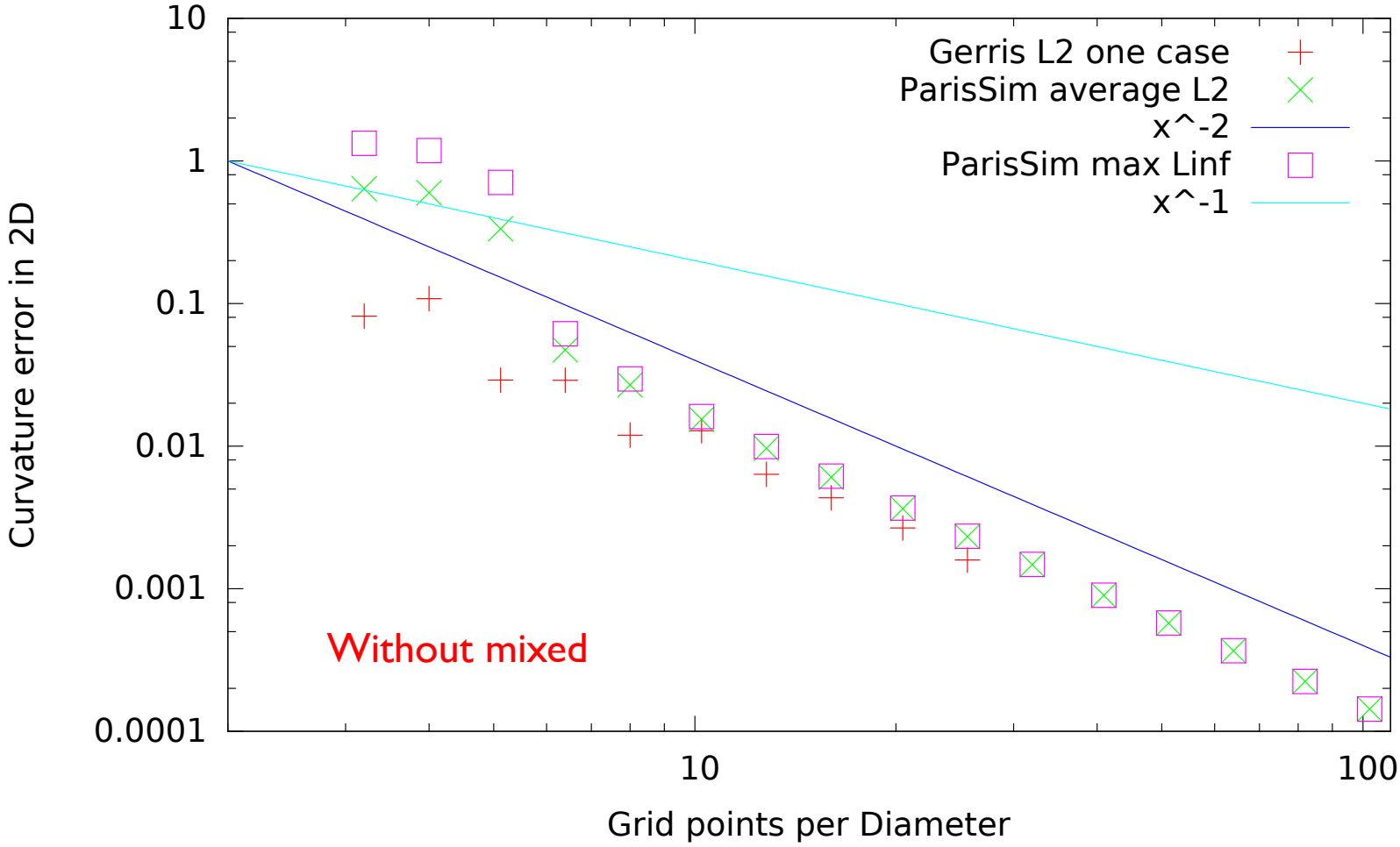

Gerris L2 one case
ParisSim Linf Nonmixed
ParisSim Linf Mixed
x^-2
x^-1
Curvature error in 3D
Grid points per Diameter
10
1
0.1
0.01
0.001
0.0001
10
100


0.0002
0.00015
0.0001
5e-05
0
-5e-05
-0.0001
-0.00015
w(0,0,R)
0
200
400
600
800
1000
time
without mixed heights
with mixed heights

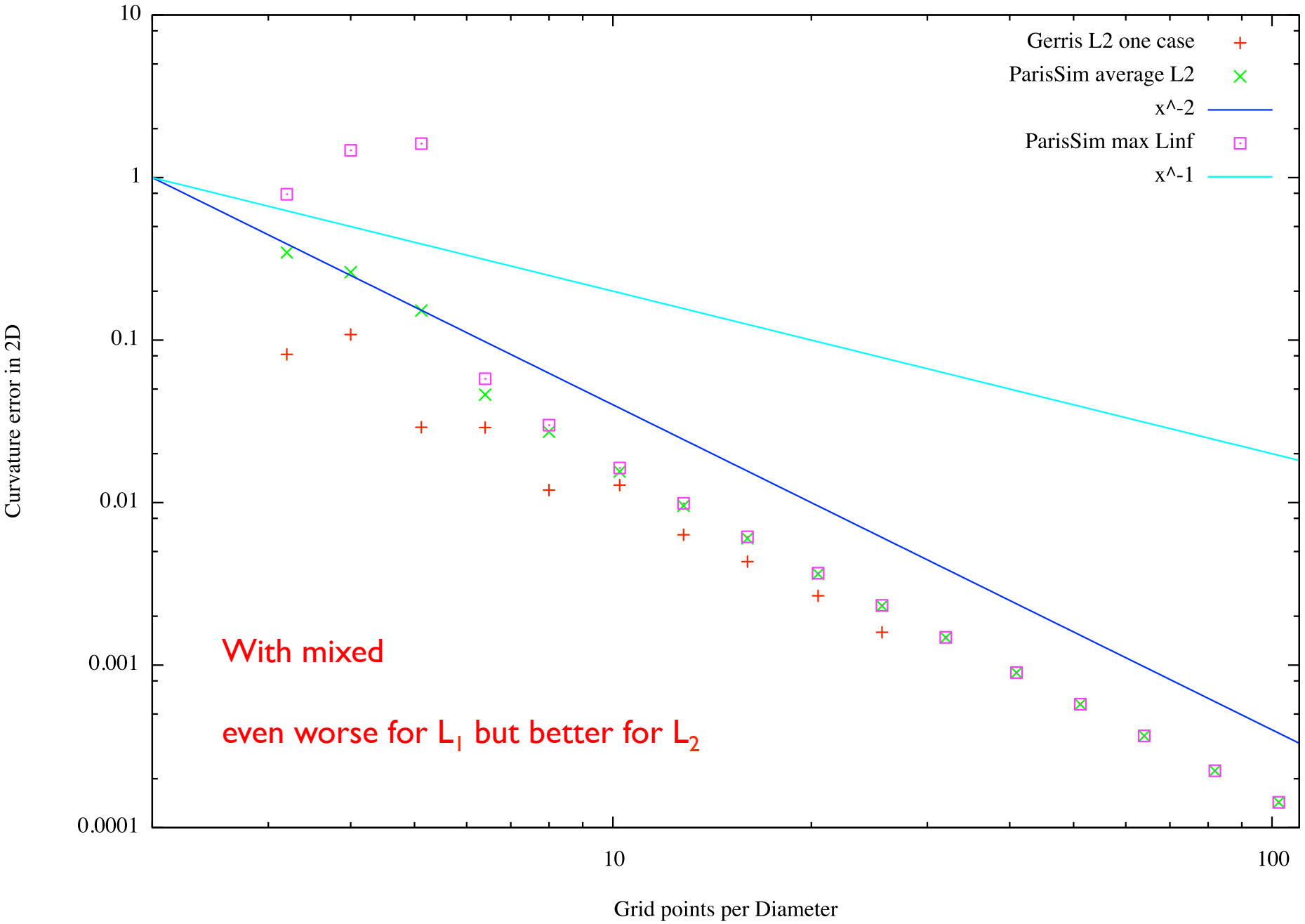
Conclusion: mixed-heights are inaccurate,
but give better capillary oscillations.

Another interesting point is that stencils 7x3x3 are sufficient (9x3x3 is needed in the theory above).

Finally go back to the 2D with mixed heights.


Curvature error in 2D
Grid points per Diameter
10
1
0.1
0.01
0.001
0.0001
10
100
Gerris L2 one case
ParisSim average L2
x^-2
ParisSim max Linf
x^-1
Without mixed

The advantage of Paris over Gerris has disappeared !


Gerris L2 one case
ParisSim average L2
x^-2
ParisSim max Linf
x^-1
With mixed
even worse for L_1 but better for L_2
Curvature error in 2D
Grid points per Diameter
10
1
0.1
0.01
0.001
0.0001
10
100

Conclusion: we still do not fully know what is the secret of Gerris.

The end