UNIVERSITÄT GRAZ
UNIVERSITY OF GRAZ

Time-stepping methods (continued) and numerical derivatives

(Pencil Code School, Geneva/CERN, 21st of September 2025)

Philippe-A. Bourdin
Philippe.Bourdin@uni-graz.at

Overview:

* Summary on common time-stepping methods

* “Creative” methods for time stepping

* Numerical derivatives

* Numerical curiosities due to precision

* Numerical curiosities from the initial condition

* Numerical curiosities from “switching on” the simulation

Time integration methods

Time integration methods

Euler method:

* very simple to implement (single-step integration)
* may have infinitely growing error

Time integration methods

Euler method:

* very simple to implement (single-step integration)
* may have infinitely growing error

Velocity-Verlet algorithm:

* simple to implement (two-step integration)
* more accurate than the simple Euler method

Time integration methods

Euler method:

* very simple to implement (single-step integration)

* may have infinitely growing error

Velocity-Verlet algorithm:

* simple to implement (two-step integration)

* more accurate than the simple Euler method

Runge-Kutta scheme:

* harder to implement (order of accuracy = number of integration substeps)

* very accurate and computation efficient (larger dt is possible)

Time integration methods

Euler method:

* very simple to implement (single-step integration)
* may have infinitely growing error

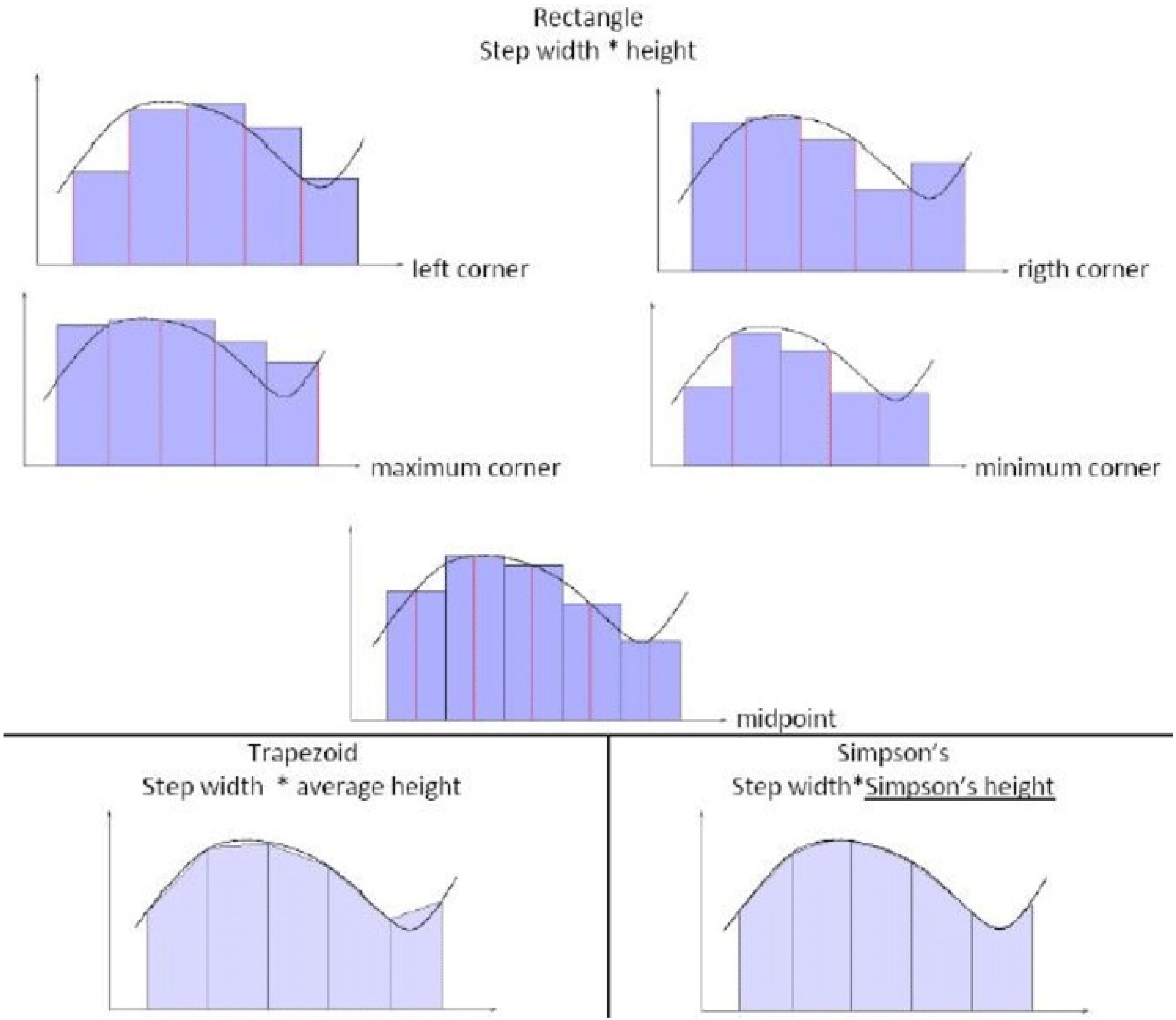
Velocity-Verlet algorithm:

* simple to implement (two-step integration)
* more accurate than the simple Euler method

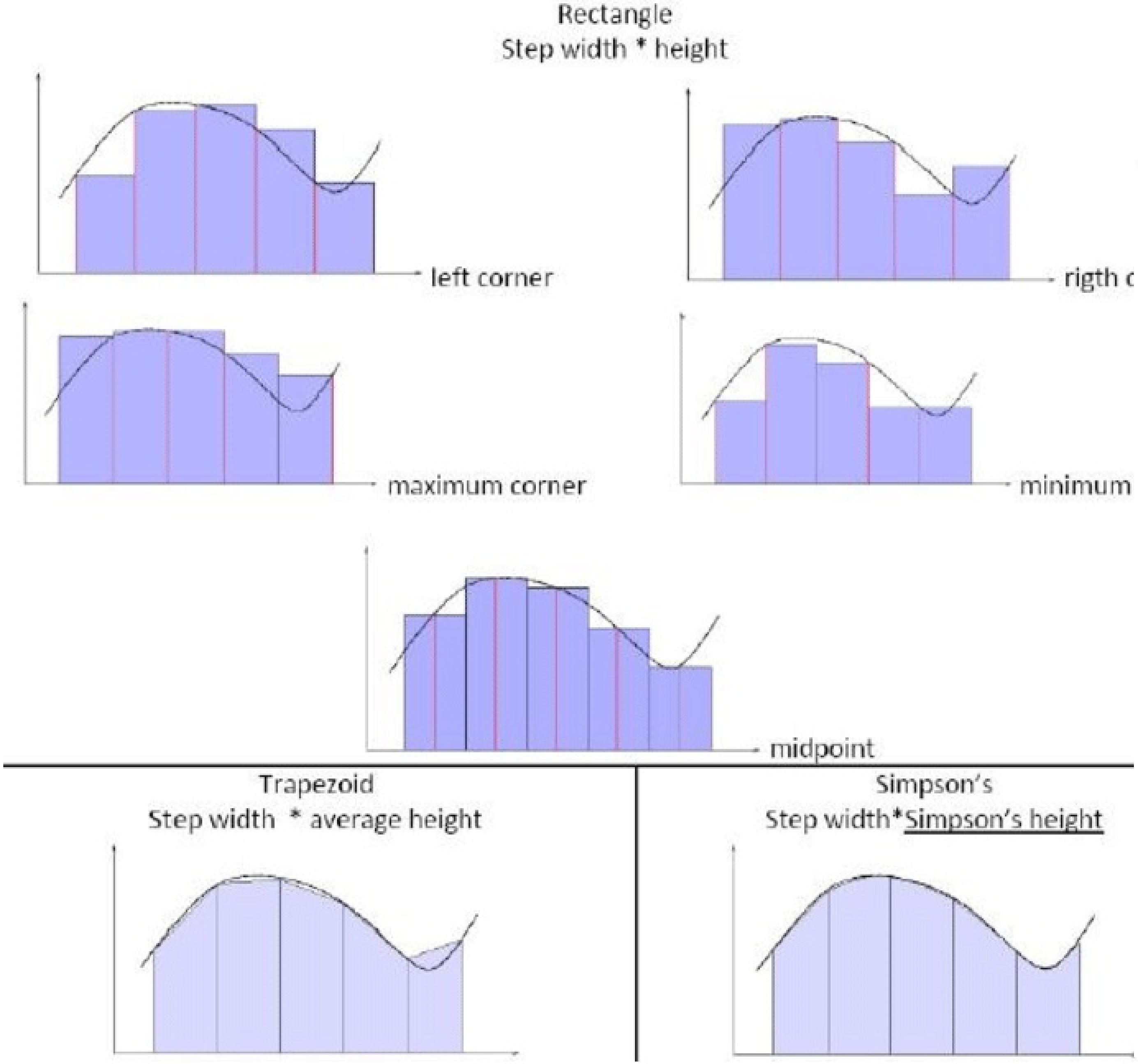
Runge-Kutta scheme:

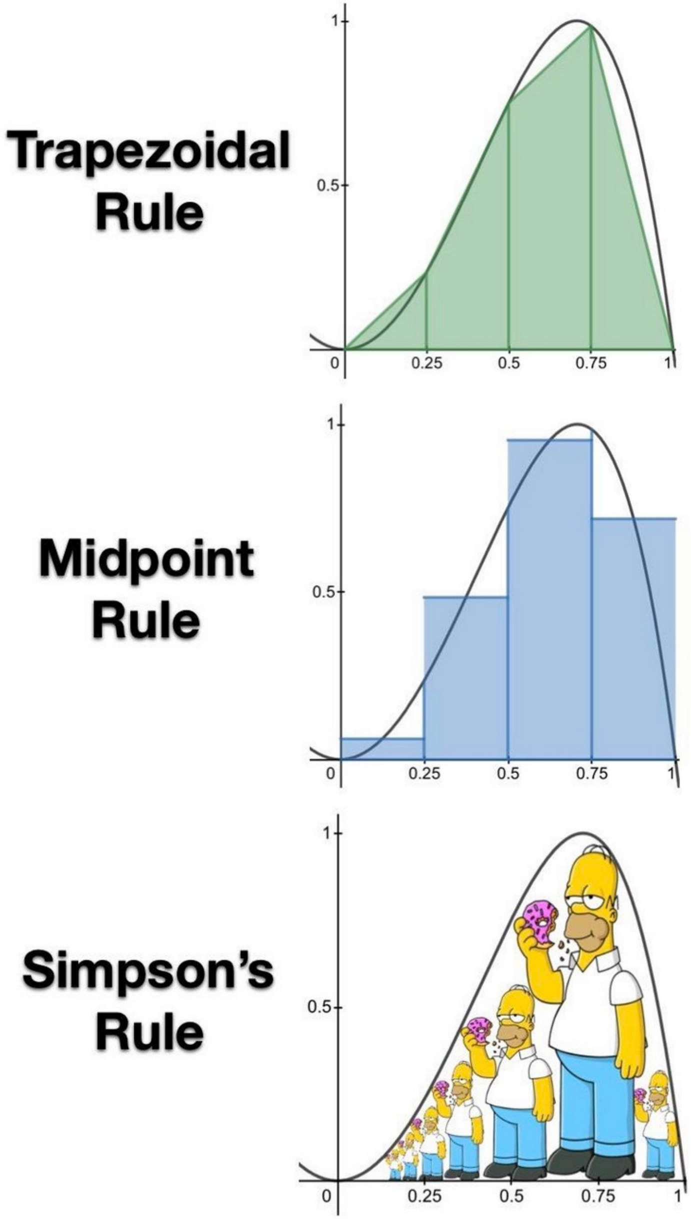
* harder to implement (order of accuracy = number of integration substeps)
* very accurate and computation efficient (larger dt is possible)
* substeps require additional function evaluations and derivatives (=> costs)

Time integration methods



Time integration methods





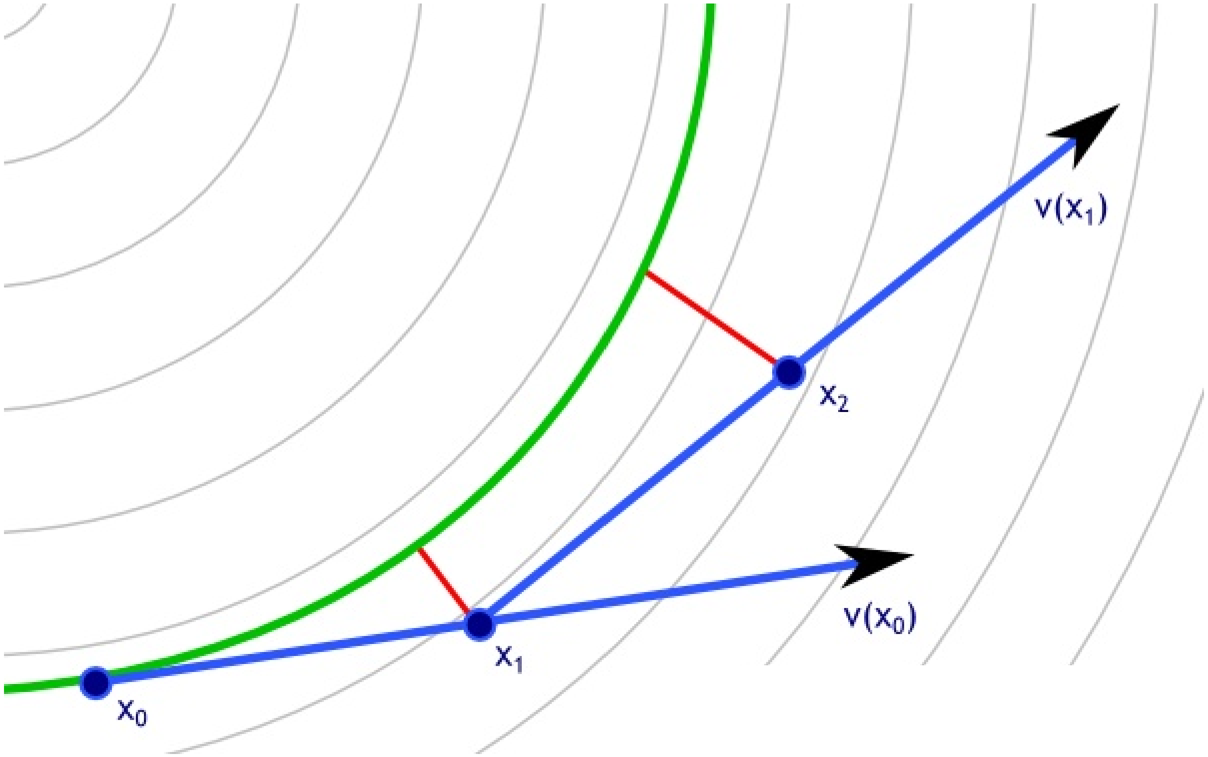
Euler method

Euler method

1) start from x_0
2) calculate derivatives (tangential)

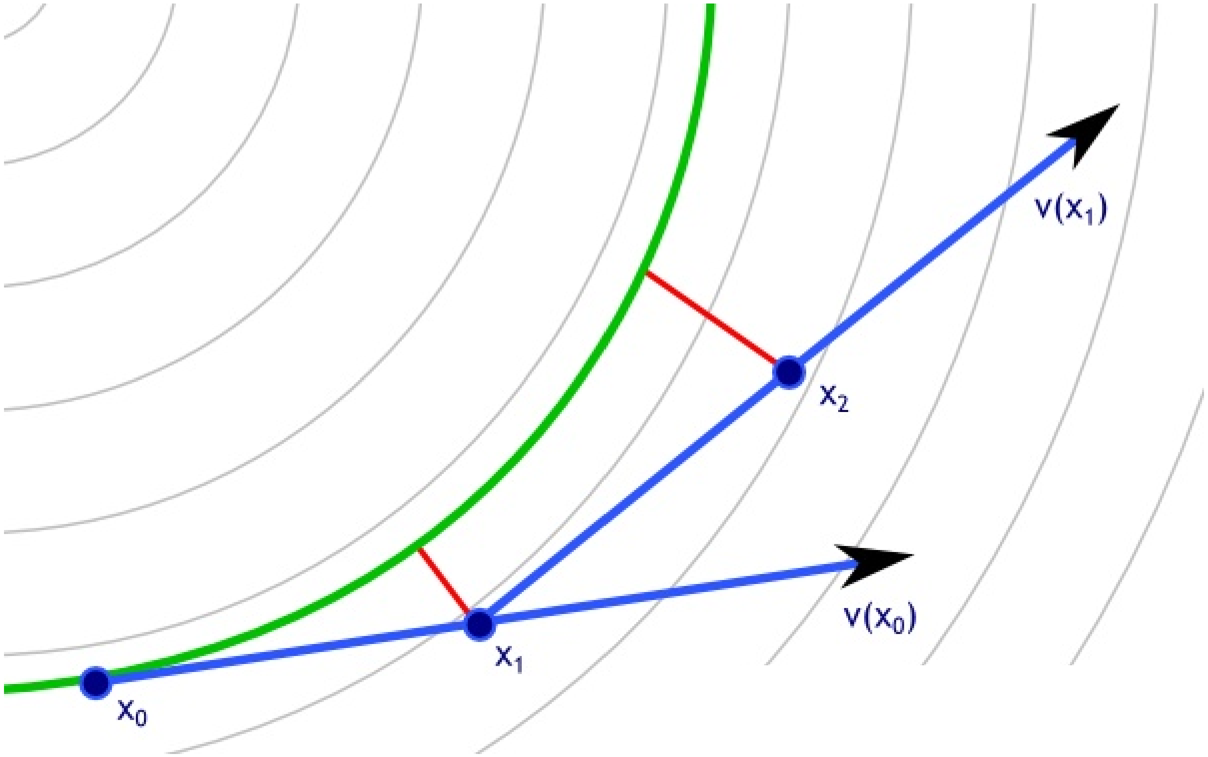
gray: true field

green: exact solution



Euler method

1) start from x_0
2) calculate derivatives (tangential)
3) iterate to x_1 with dt=0.5



green: exact solution

blue: tangential derivative

red: deviation

$$x_{n+1} = x_n + dt \cdot v(x_n) + O(dt^2)$$

x_n : current position

x_{n+1} : next position

dt : time step

$v(x_n)$: vector field at position x_n

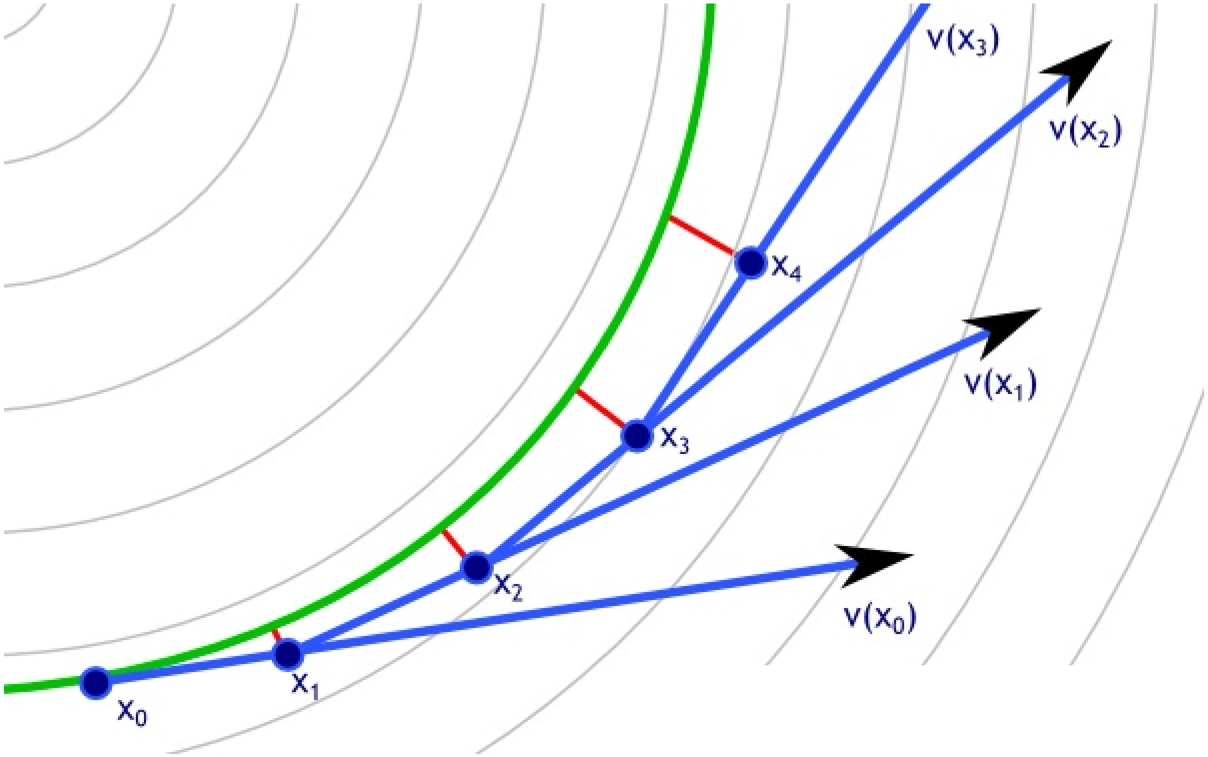
$O(dt^2)$: error

=> error grows infinitely…

=> make dt smaller?

Euler method

1) start from x_0
2) calculate derivatives (tangential)
3) iterate to x_1 with dt=**0.25**



green: exact solution

blue: tangential derivative

red: deviation

$$x_{n+1} = x_n + dt \cdot v(x_n) + O(dt^2)$$

x_n : current position

x_{n+1} : next position

dt : time step

$v(x_n)$: vector field at position x_n

$O(dt^2)$: error

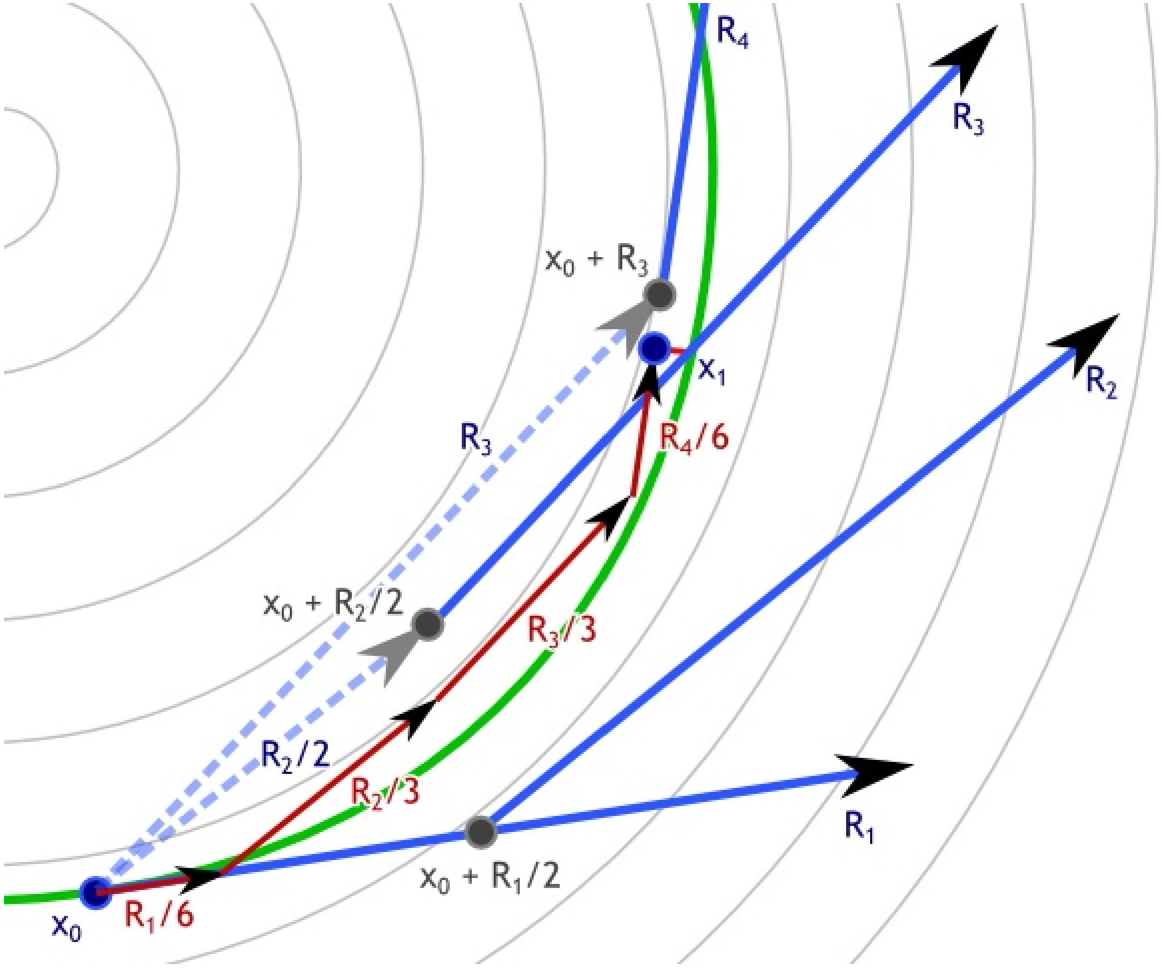
=> accuracy can be improved by making dt smaller

=> but error still grows...

Runge-Kutta method (4th order)

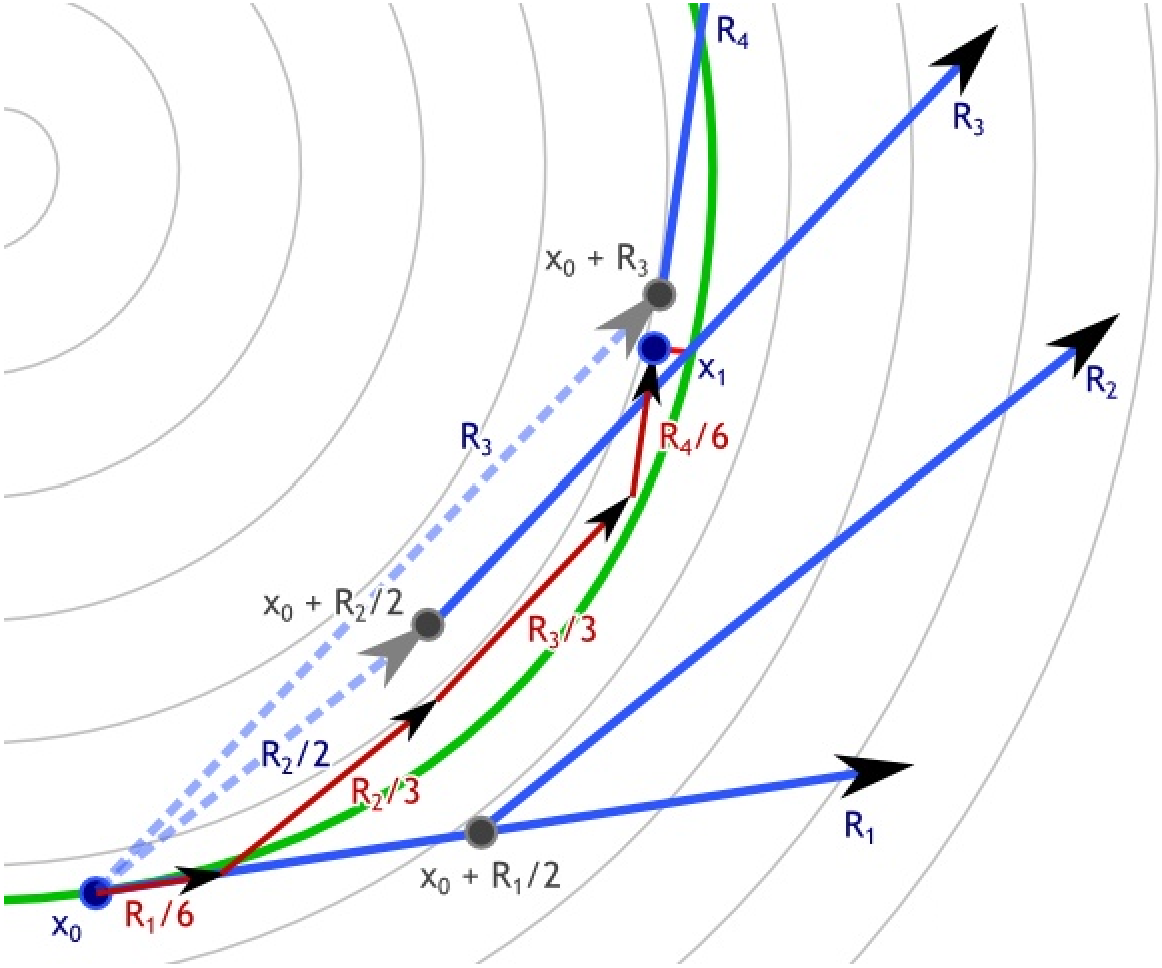
1) start from x_0
2) calculate derivatives (tangential)
3) iterate substeps and recalculate derivatives (tangential)
4) reach x_1 with substeps

blue: tangential derivative
red: deviation



Runge-Kutta method (4th order)

1) start from x_0
2) calculate derivatives (tangential)
3) iterate substeps and recalculate derivatives (tangential)
4) reach x_1 with substeps



blue: tangential derivative

red: deviation

=> much better accuracy!

=> larger computational cost

Runge-Kutta method (4th order)

RK4 scheme:

$$k_1 = dt \cdot v(x_n)$$

$$k_2 = dt \cdot v(x_n + \frac{k_1}{2})$$

$$k_3 = dt \cdot v(x_n + \frac{k_2}{2})$$

$$k_4 = dt \cdot v(x_n + k_3)$$

$$x_{n+1} = x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(dt^5)$$

x_n : current position

x_{n+1} : next position

dt : time step

$v(x)$: vector field at position x

$O(dt^5)$: error

“Creative” time-stepping methods

“Creative” time-stepping methods

- Stiff differential equations?

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

Hope: few super steps much larger than from Courant (CFL) criterion

are later compensated by following smaller steps => overall speed up!

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

Hope: few super steps much larger than from Courant (CFL) criterion

are later compensated by following smaller steps => overall speed up!

Hope: irregular time steps avoid certain frequencies (wiggles) in result.

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

Hope: few super steps much larger than from Courant (CFL) criterion

are later compensated by following smaller steps => overall speed up!

Hope: irregular time steps avoid certain frequencies (wiggles) in result.

=> effect: works surprisingly well in 1D – but needs a lot of diffusion in 2D.

“Creative” time-stepping methods

- Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

- Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

Hope: few super steps much larger than from Courant (CFL) criterion

are later compensated by following smaller steps => overall speed up!

Hope: irregular time steps avoid certain frequencies (wiggles) in result.

=> effect: works surprisingly well in 1D – but needs a lot of diffusion in 2D.

=> 3D: either huge diffusion or it will need the same amount of time steps.

“Creative” time-stepping methods

Stiff differential equations?

Small timestep only for fastest dynamics (**timestep_subcycle** with **corona**).

=> effect: back-reaction to system is delayed => slower overall dynamics!

more timesteps needed & finally not such a big saving of computing time.

Mix of larger and smaller steps – in the hope the error is small?

Idea: Tschebychev polynomials for super time steps (**timestep_STS**),

Hope: few super steps much larger than from Courant (CFL) criterion

are later compensated by following smaller steps => overall speed up!

Hope: irregular time steps avoid certain frequencies (wiggles) in result.

=> effect: works surprisingly well in 1D – but needs a lot of diffusion in 2D.

=> 3D: either huge diffusion or it will need the same amount of time steps.

=> Speed up and accuracy granted only in 1D.

“Creative” time-stepping methods

Split Runge-Kutte time step into two halves (**timestep_strang**).

Idea: ...?

=> effects?

“Creative” time-stepping methods

- Split Runge-Kutte time step into two halves (**timestep_strang**).

Idea: ...?

=> effects?

- Algorithms for stiff PDEs (**timestep_stiff** from Numerical Recipies).

Idea: ...?

=> effects?

Numerical derivatives (6th order)

Numerical derivatives (6^{th} order)

Pencil Code manual:

H Numerical methods

H.1 Sixth-order spatial derivatives

Spectral methods are commonly used in almost all studies of ordinary (usually incompressible) turbulence. The use of this method is justified mainly by the high numerical accuracy of spectral schemes. Alternatively, one may use high order finite differences that are faster to compute and that can possess almost spectral accuracy. Nordlund & Stein [32] and Brandenburg et al. [16] use high order finite difference methods, for example fourth and sixth order compact schemes [28].[19]

The sixth order first and second derivative schemes are given by

$$f_i' = (-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3})/(60\delta x), \tag{235}$$

$$f_i'' = (2f_{i-3} - 27f_{i-2} + 270f_{i-1} - 490f_i + 270f_{i+1} - 27f_{i+2} + 2f_{i+3})/(180\delta x^2), \tag{236}$$

Numerical derivatives (6^{th} order)

Pencil Code manual:

H Numerical methods

H.1 Sixth-order spatial derivatives

Spectral methods are commonly used in almost all studies of ordinary (usually incompressible) turbulence. The use of this method is justified mainly by the high numerical accuracy of spectral schemes. Alternatively, one may use high order finite differences that are faster to compute and that can possess almost spectral accuracy. Nordlund & Stein [32] and Brandenburg et al. [16] use high order finite difference methods, for example fourth and sixth order compact schemes [28].[19]

The sixth order first and second derivative schemes are given by

$$f_i' = (-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3})/(60\delta x), \qquad (235)$$

$$f_i'' = (2f_{i-3} - 27f_{i-2} + 270f_{i-1} - 490f_i + 270f_{i+1} - 27f_{i+2} + 2f_{i+3})/(180\delta x^2), \qquad (236)$$

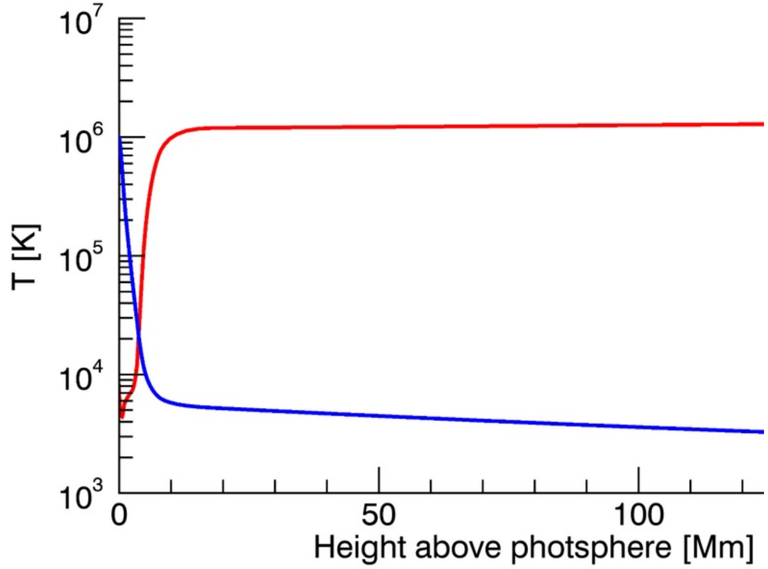
=> Coefficients can be obtained analytically.

Resolving steep gradients in the solar corona

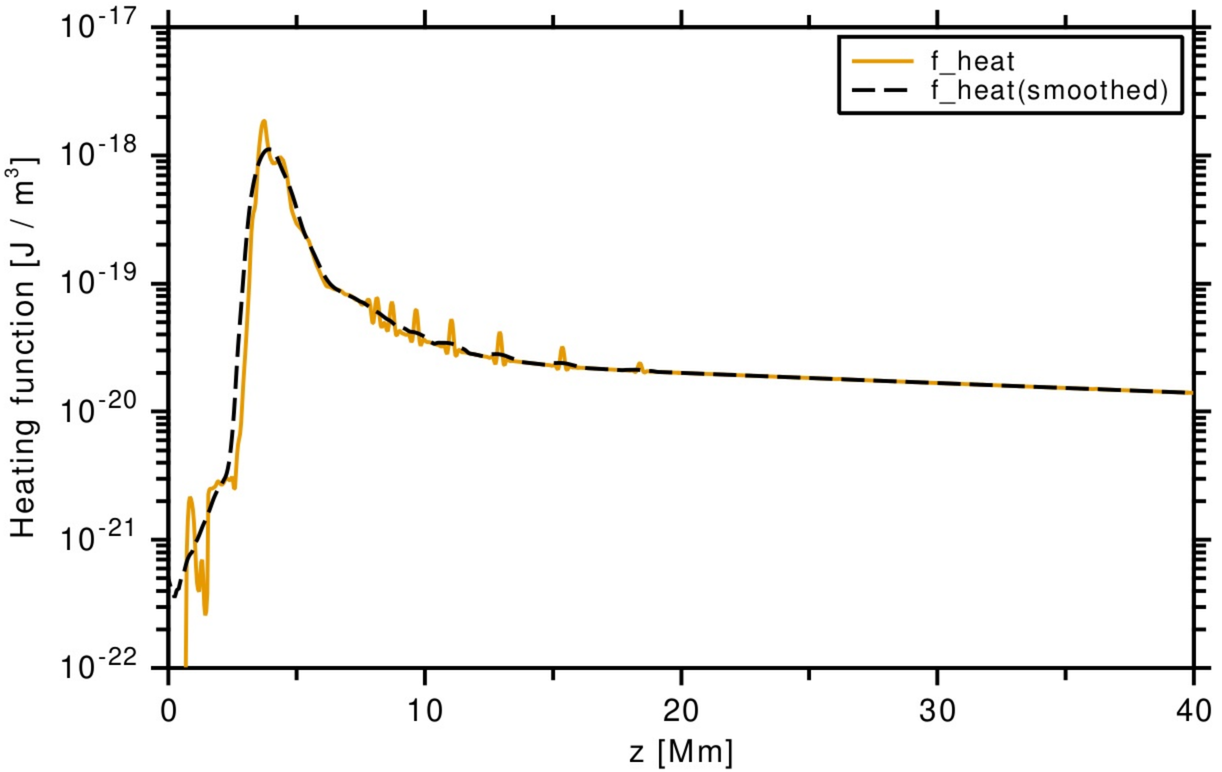
(Vartika Pandey)

Coronal heating in 1D MHD models:

- Initial condition:

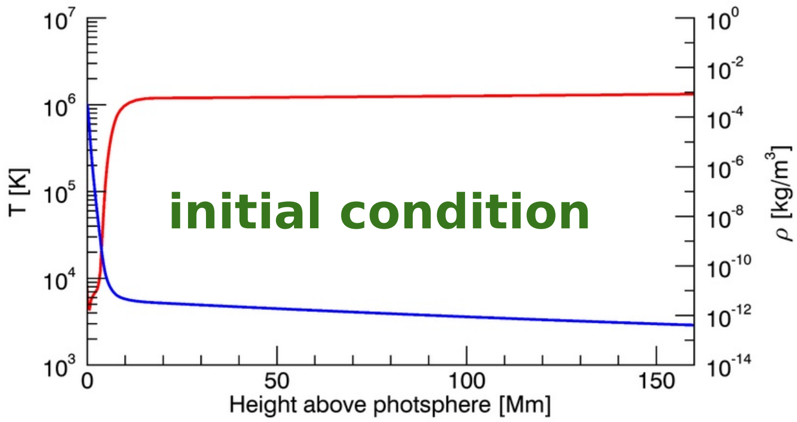


=> After **one** iteration:

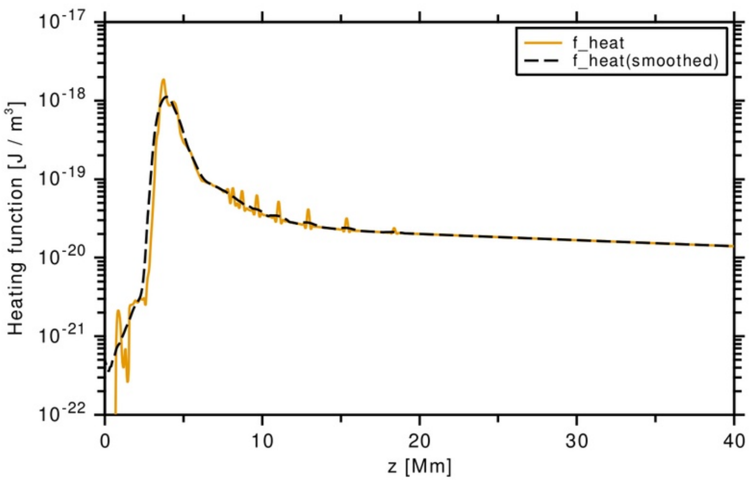


(Pandey & Bourdin, 2025)

Numerical stability analysis:

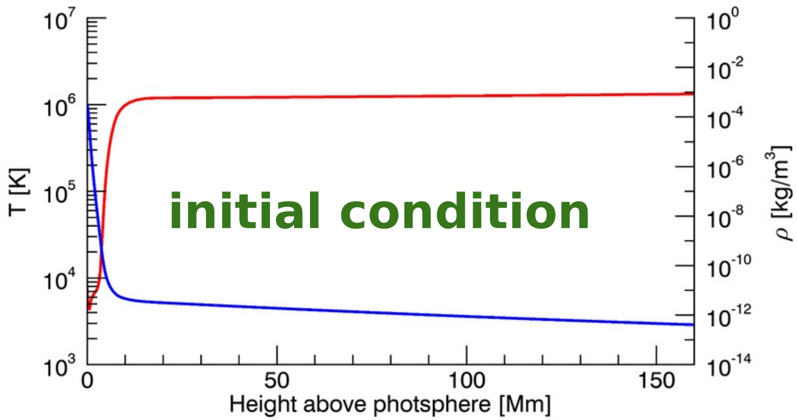


+ heating

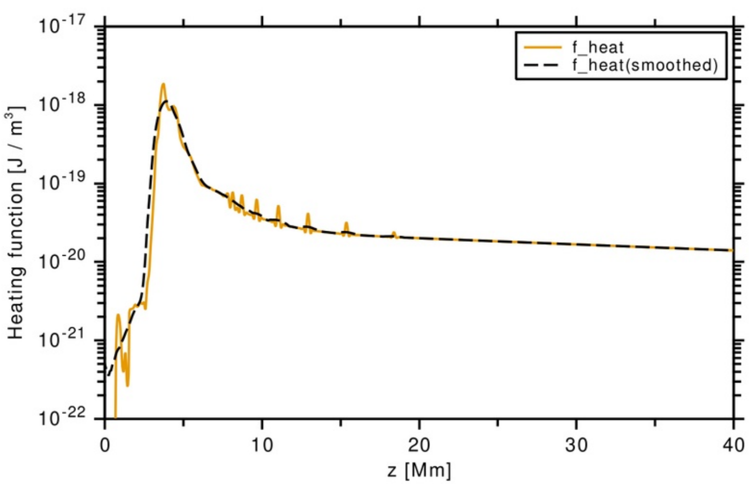


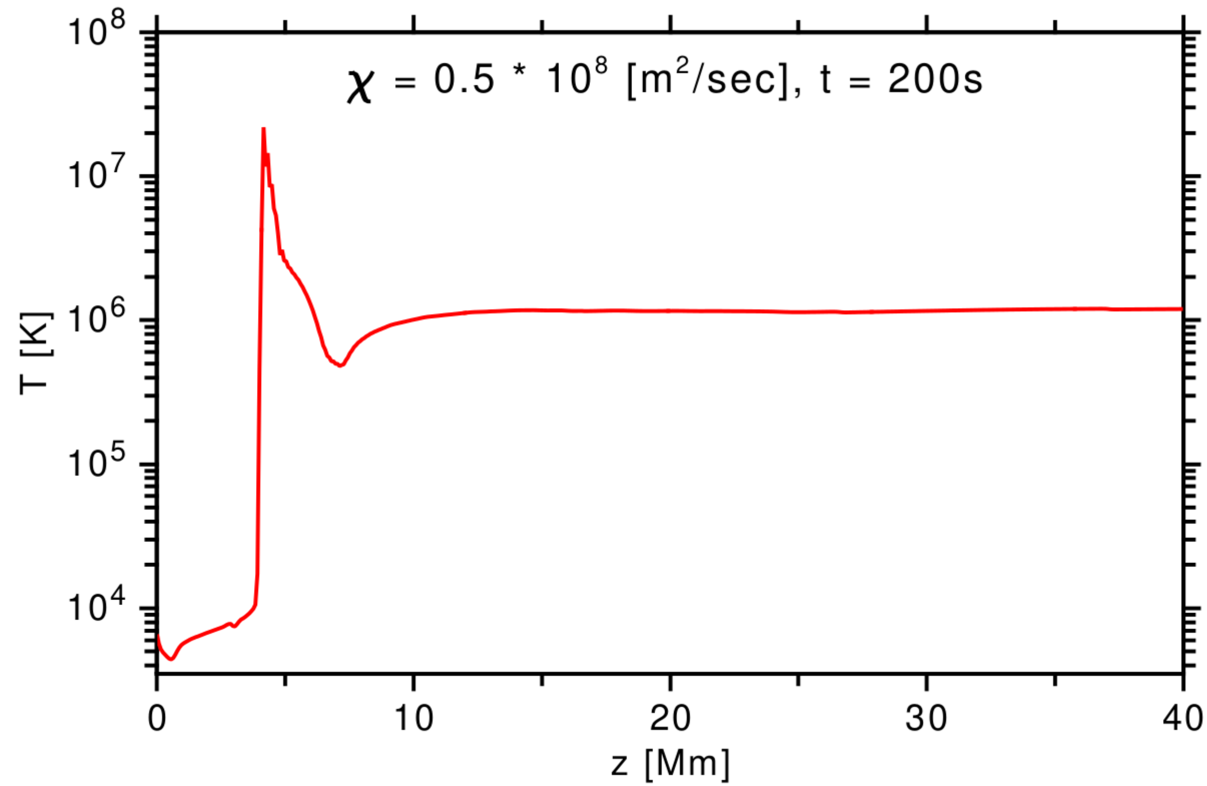
(Pandey & Bourdin, 2025)

Numerical stability analysis:



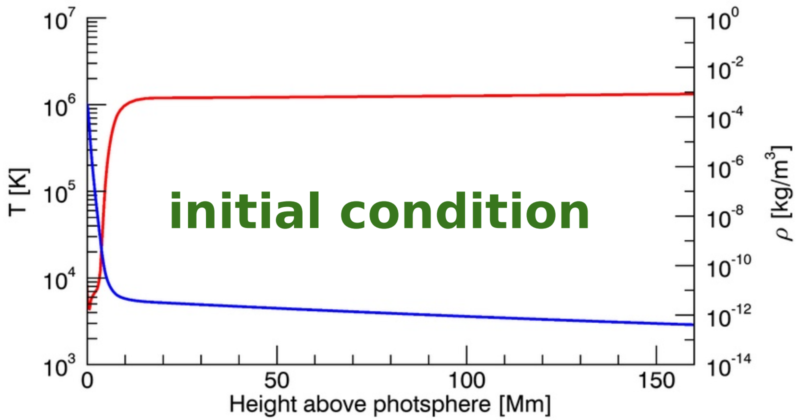
+ heating =>





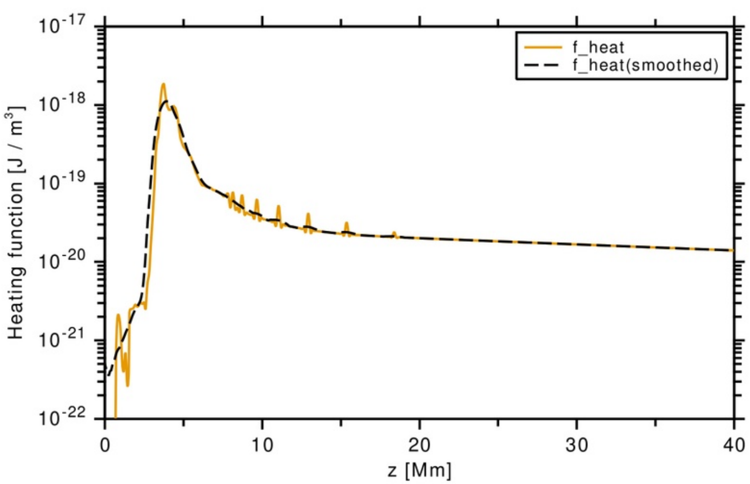
(Pandey & Bourdin, 2025)

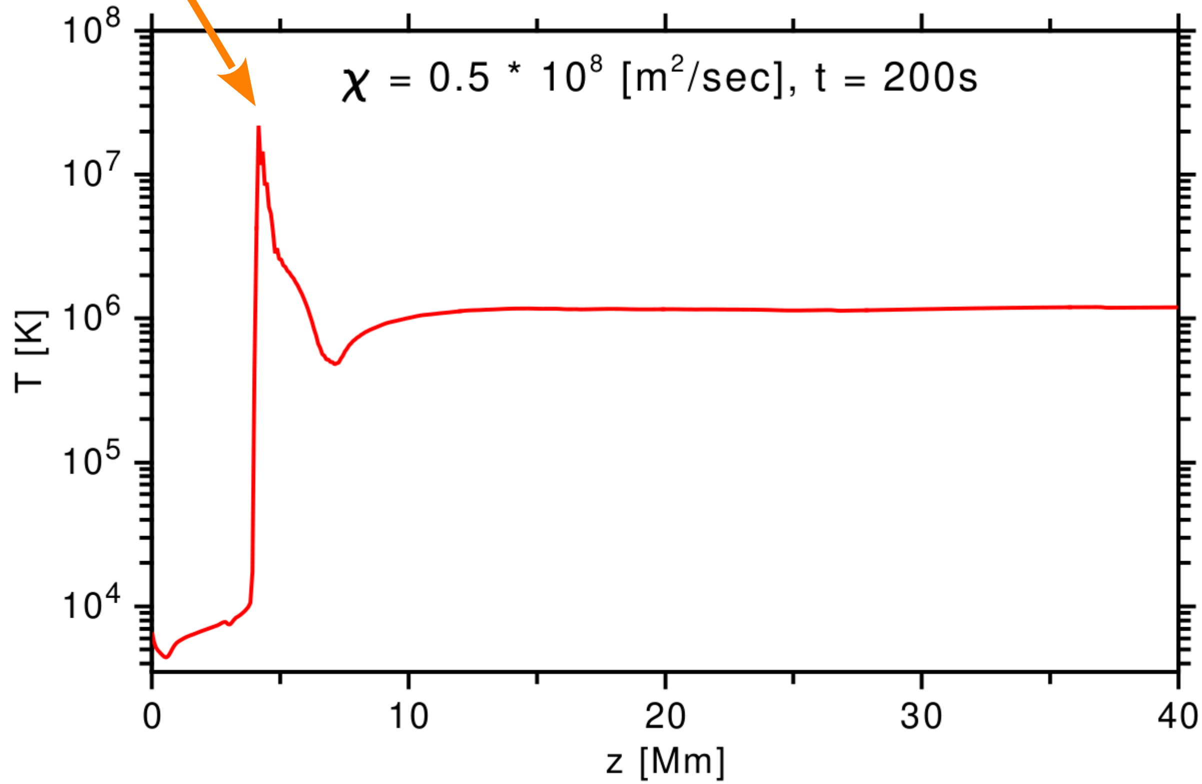
Numerical stability analysis:



- ? -

+ heating =>



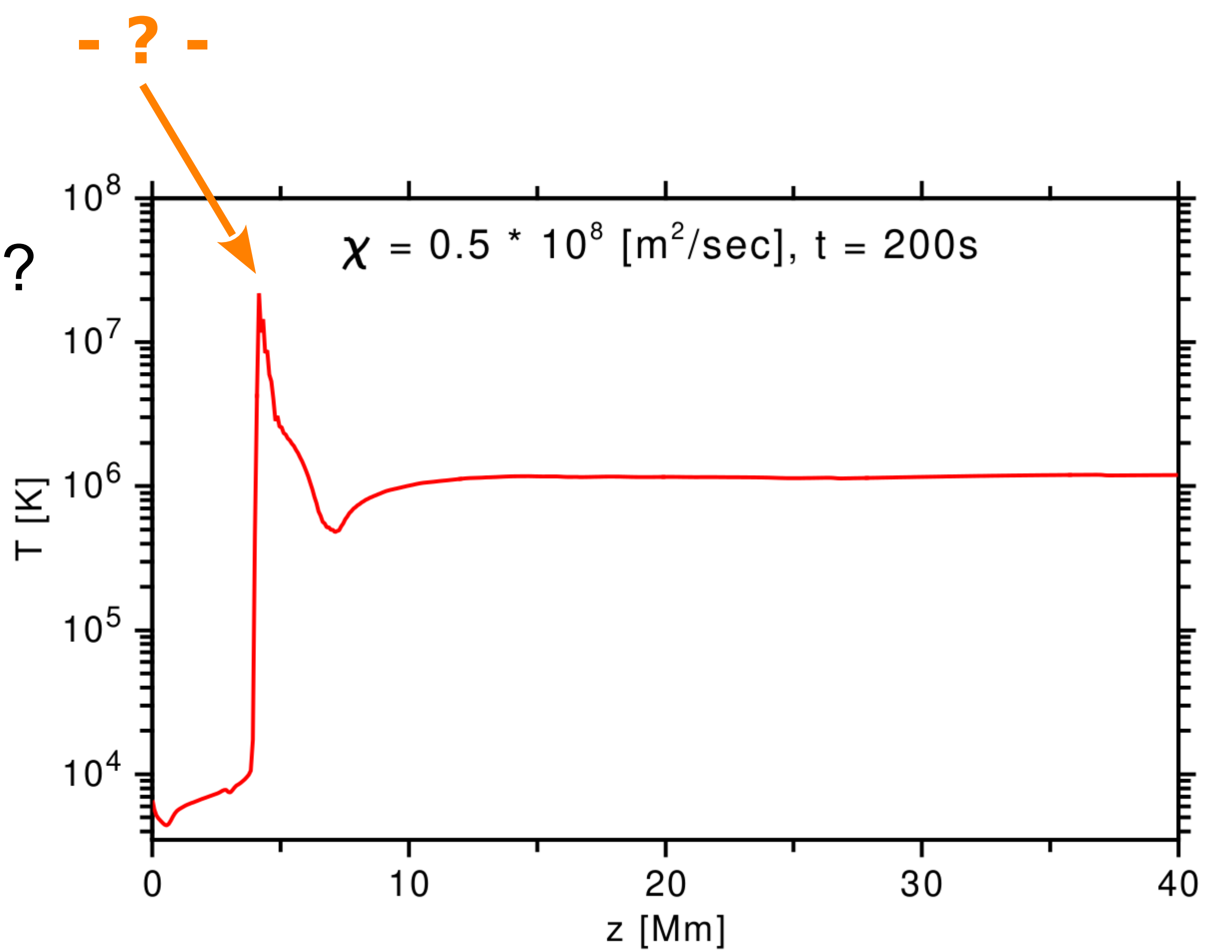


(Pandey & Bourdin, 2025)

Numerical stability analysis:

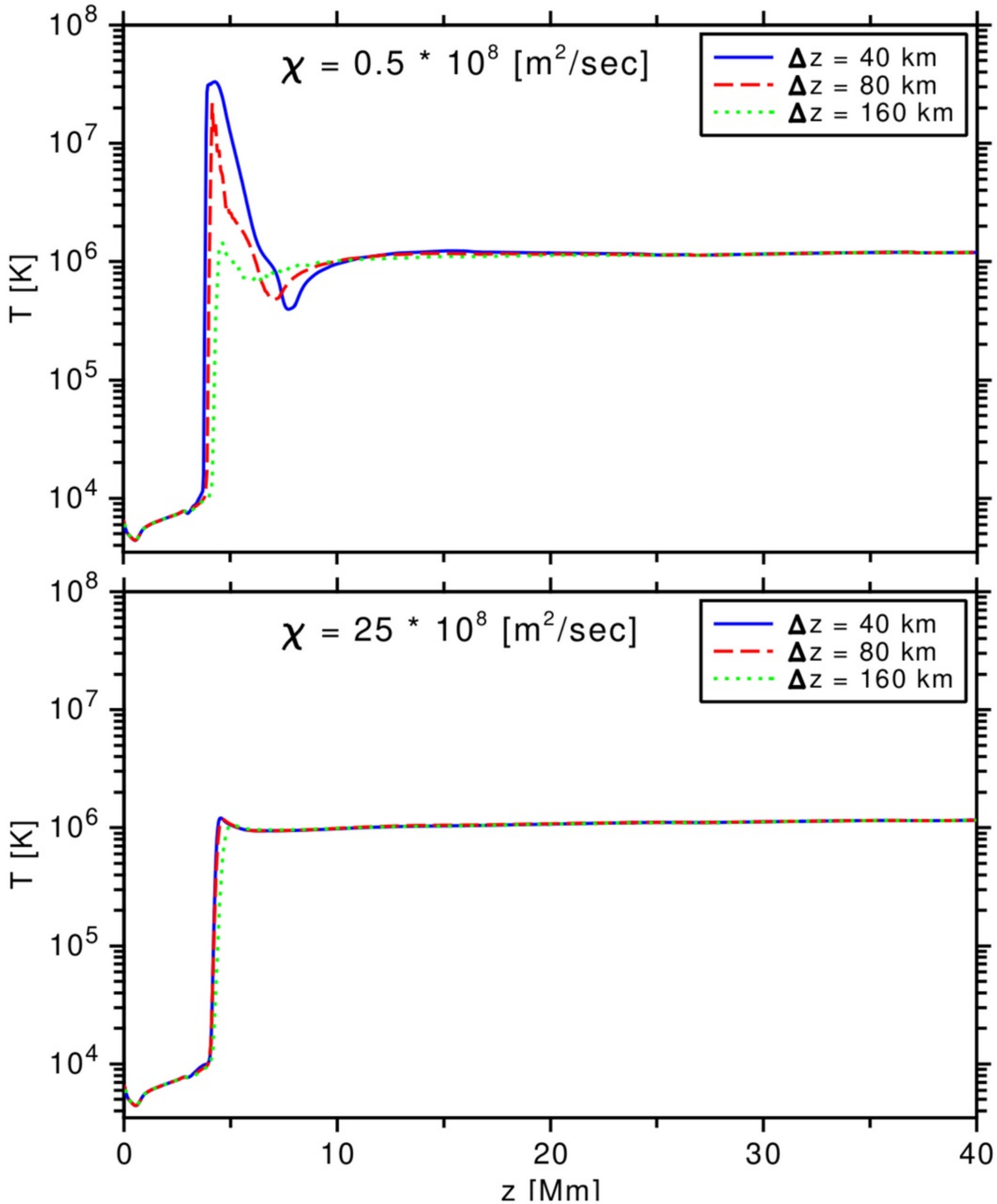
<u>Possible problems:</u>

1) too coarse grid?
2) insufficient diffusion?
3) absence of magnetic field?



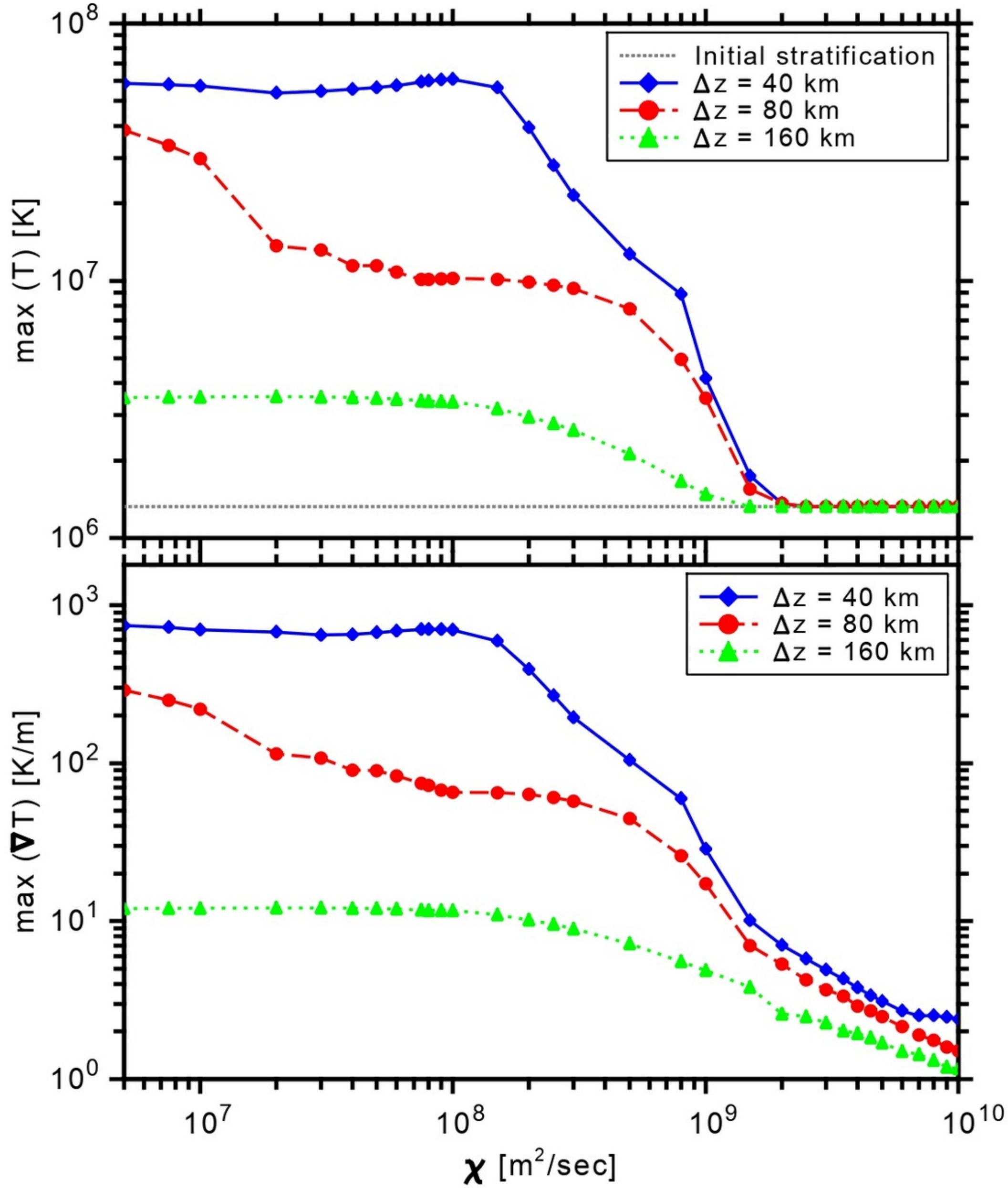
(Pandey & Bourdin, 2025)

Effects due to the grid resolution and heat conduction:



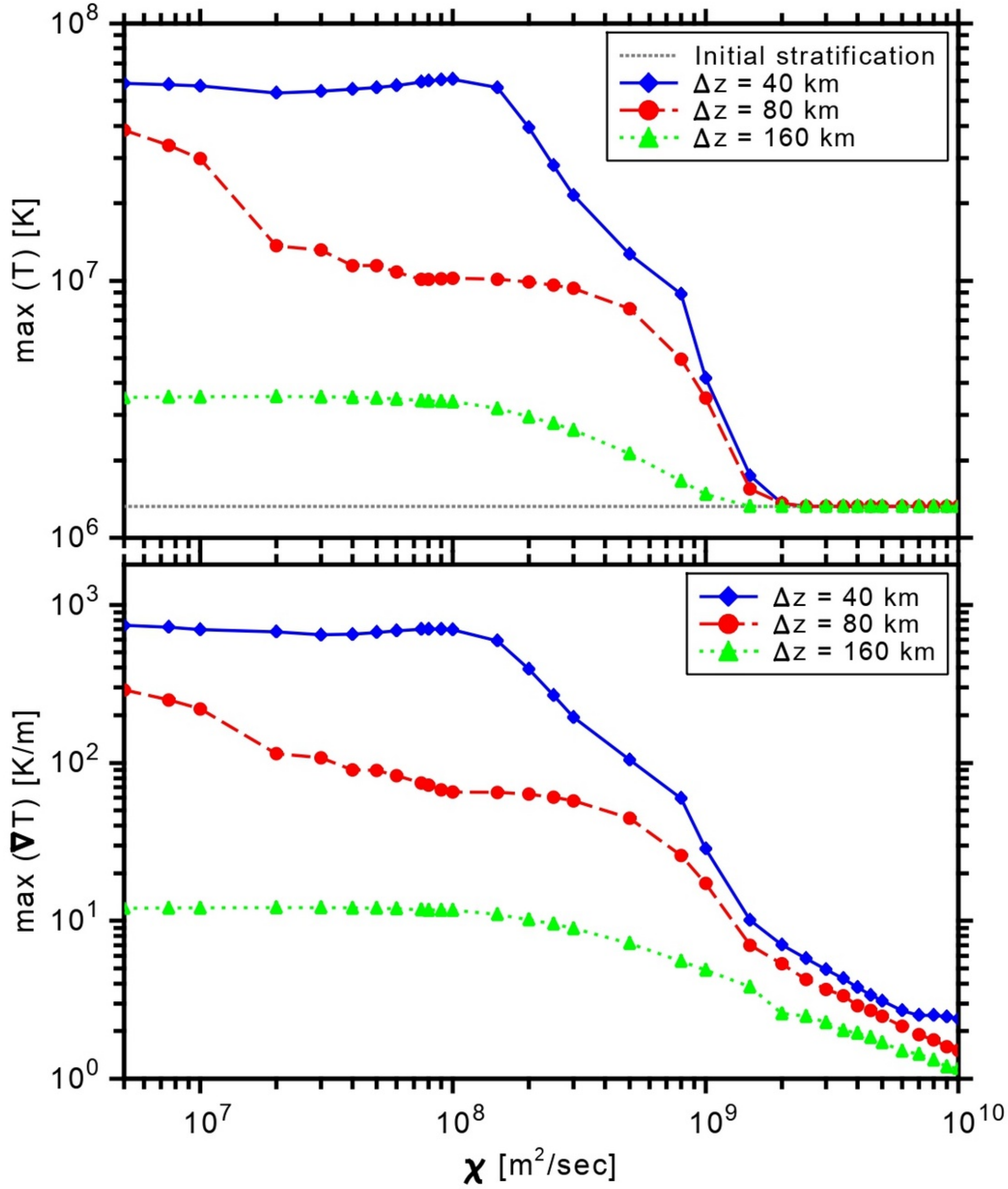
(Pandey & Bourdin, 2025)

Scaling the heat conduction:



(Pandey & Bourdin, 2025)

Scaling the heat conduction:

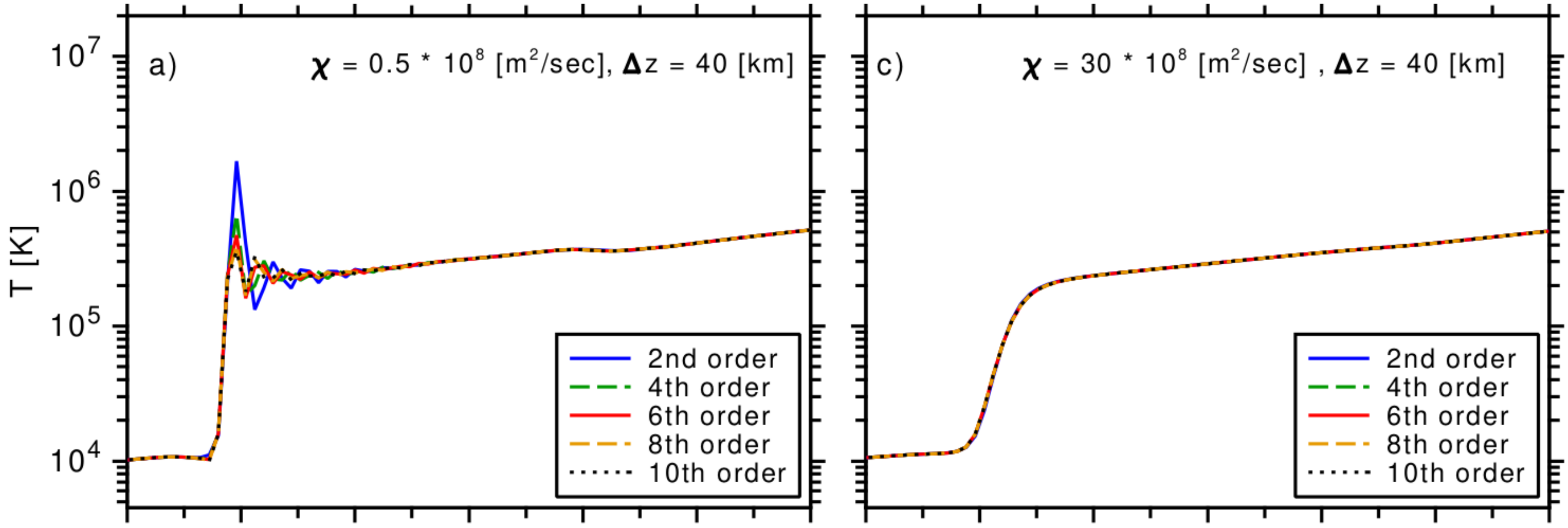


=> higher resolutions do not always imply models are “better”

=> diffusion constants need to fit to the grid resolution

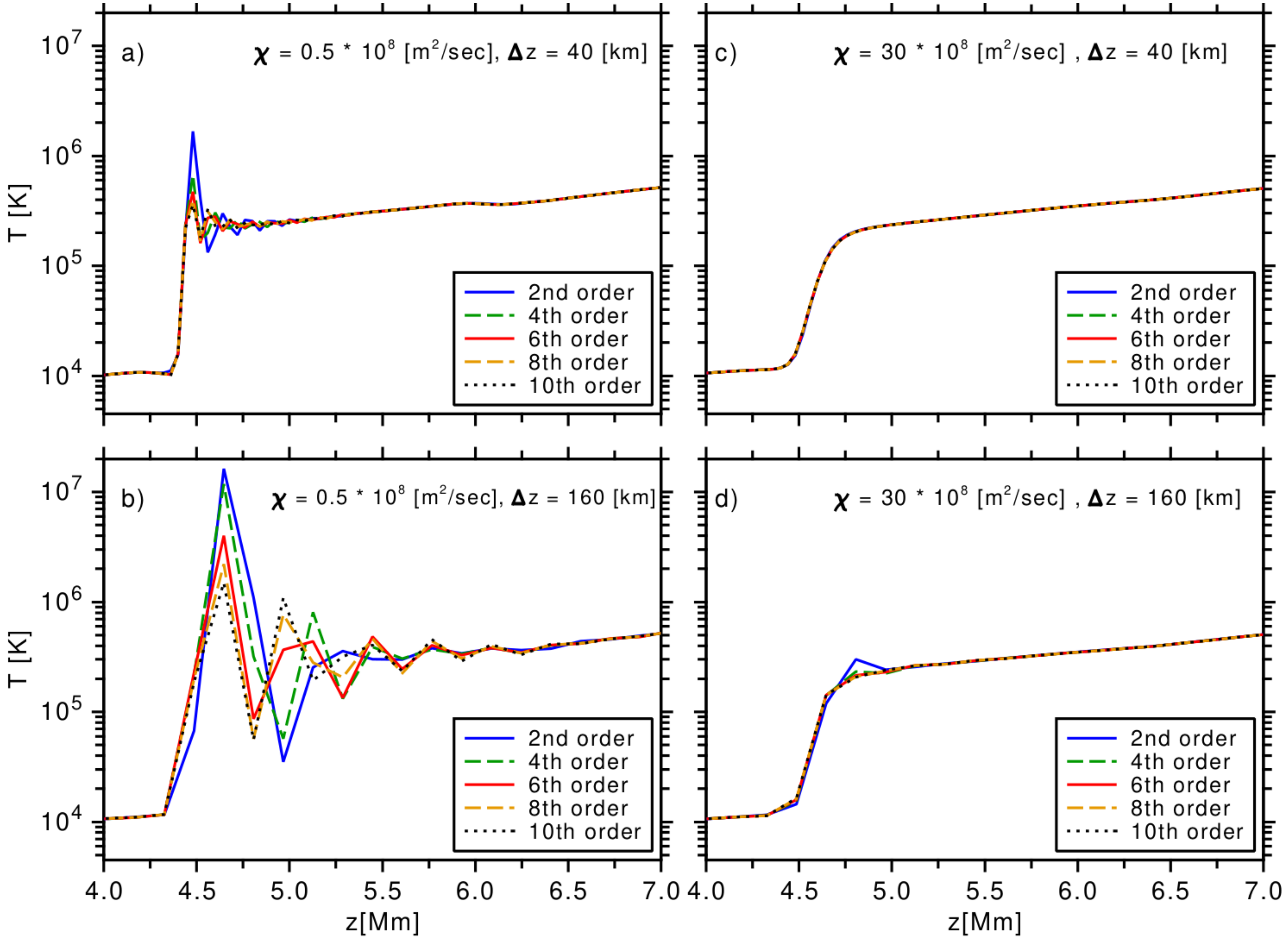
(Pandey & Bourdin, 2025)

Effects due to the order of numerical derivatives:



Pandey & Bourdin (2025)

Effects due to the order of numerical derivatives:



Pandey & Bourdin (2025)

Numerical curiosities
(due to precision & truncation errors)

Precision and machine epsilon

Numerical curiosities cabinet:

$$(x+y)+z \neq x+(y+z)$$

Precision and machine epsilon

Numerical curiosities cabinet:

$$(x+y)+z \neq x+(y+z)$$

Machine epsilon:

$$1.0+\epsilon=1.0$$

Precision and machine epsilon

Numerical curiosities cabinet:

$$(x+y)+z \neq x+(y+z)$$

Machine epsilon:

$$1.0+\epsilon=1.0 \qquad (*)$$

- Estimation of the granularity:

(***b = 2; p₀ = 0.5***)

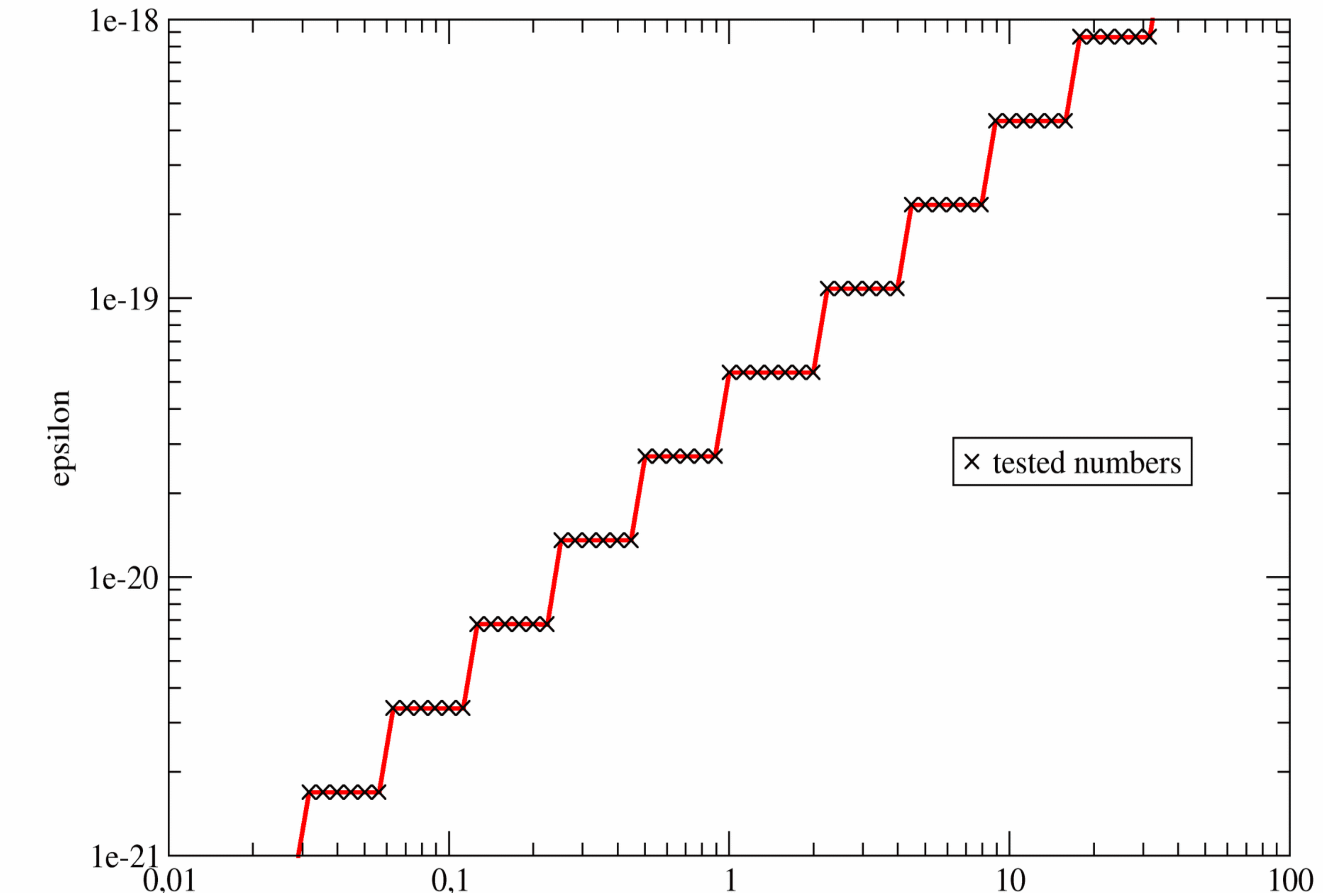
$$\epsilon_N=\frac{b}{2}\cdot b^{-p_N}$$

$$p_{N+1}=p_N\cdot b$$

Stop, if (*) is true

Precision and machine epsilon

Machine epsilon in Fortran 90 (double precision):



Granularity of numbers

Numerical curiosities cabinet: **(continued...)**

Example: simple increment of simulation time:

$$T_N + dT = T_{N+1}$$

Numerical curiosities cabinet: **(continued...)**

Example: simple increment of simulation time:

$$T_N + dT = T_{N+1}$$

$$4096 + 4 \cdot 10^{-4} = 4096$$

Numerical curiosities cabinet: **(continued...)**

Example: simple increment of simulation time:

$$T_N + dT = T_{N+1}$$

$$4096 + 4 \cdot 10^{-4} = 4096$$

$$\Rightarrow \frac{T_N}{dT} \approx 10^{-7}$$

Granularity of numbers

Numerical curiosities cabinet: **(continued...)**

<u>Example:</u> simple increment of simulation time:

$$T_N + dT = T_{N+1}$$

$$4096 + 4 \cdot 10^{-4} = 4096$$

$$\Rightarrow \frac{T_N}{dT} \approx 10^{-7}$$

<u>But:</u>

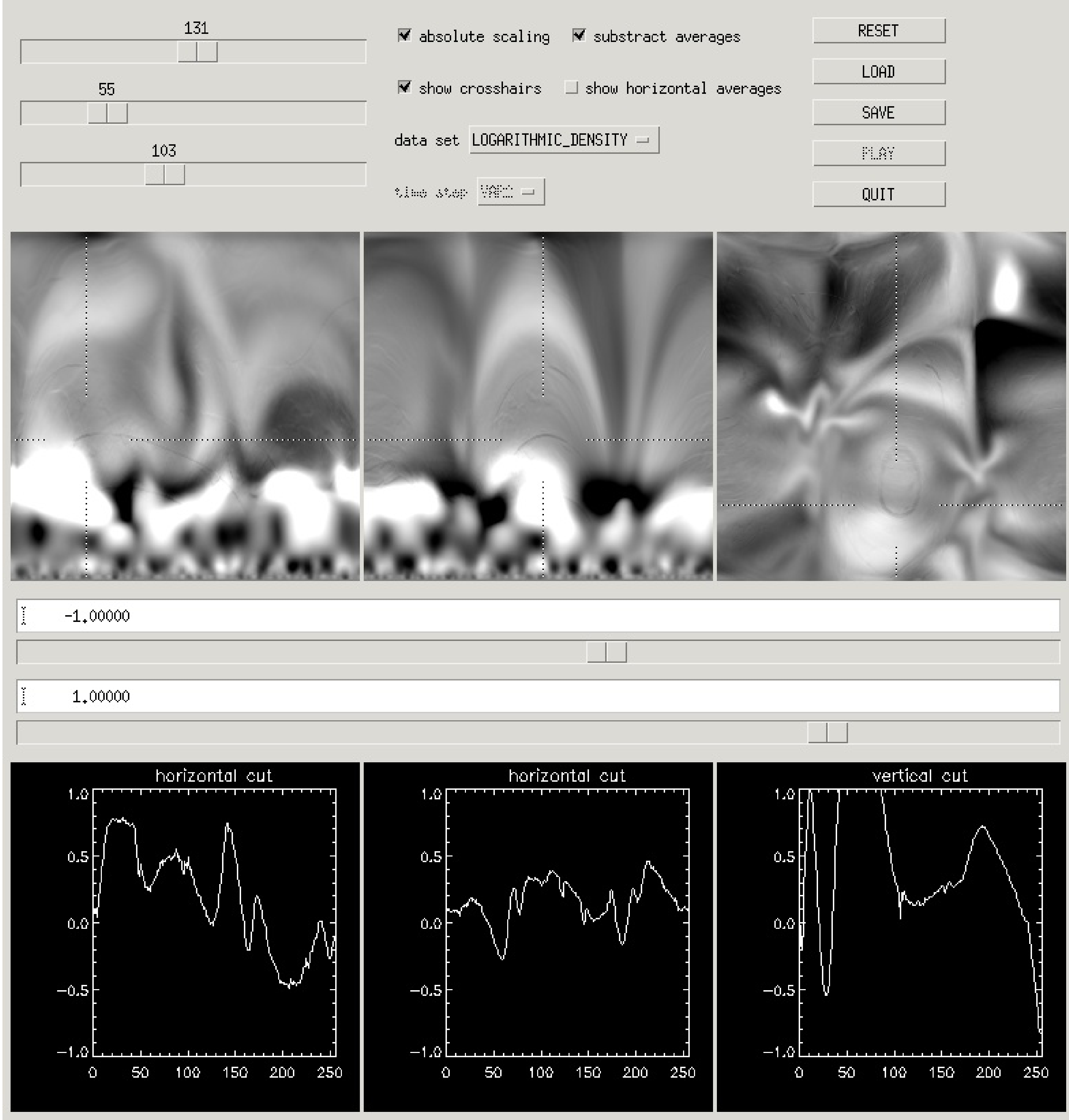
$$4096 + 8 \cdot 10^{-4} \neq 4096$$

Granularity effects in physical observables

Density in a coronal 3D MHD simulation:

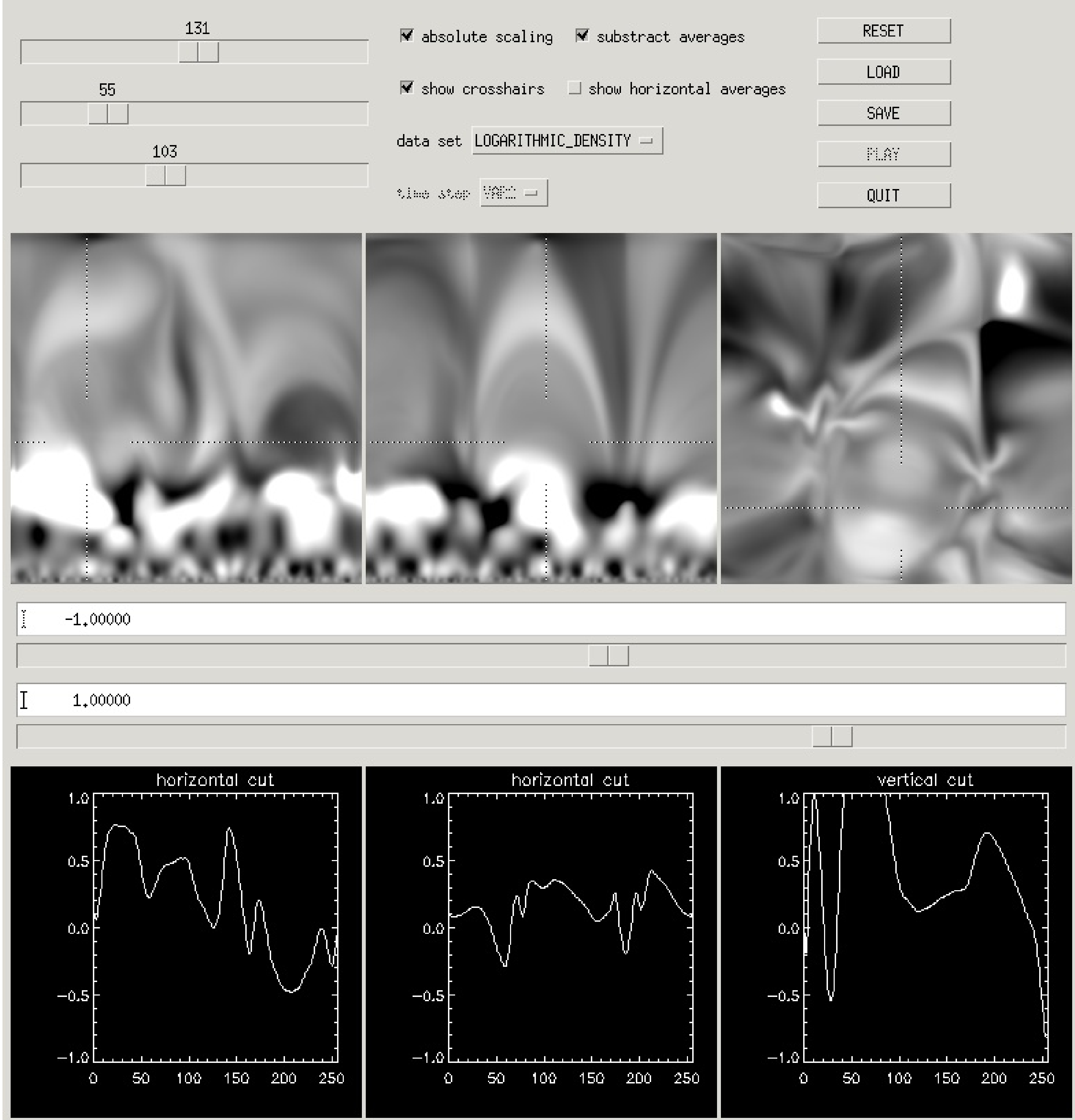
ln(ρ):

SP


131
55
103
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set LOGARITHMIC_DENSITY
RESET
LOAD
SAVE
PLAY
QUIT
-1.00000
1.00000
horizontal cut
horizontal cut
vertical cut
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250

ln(ρ):

DP

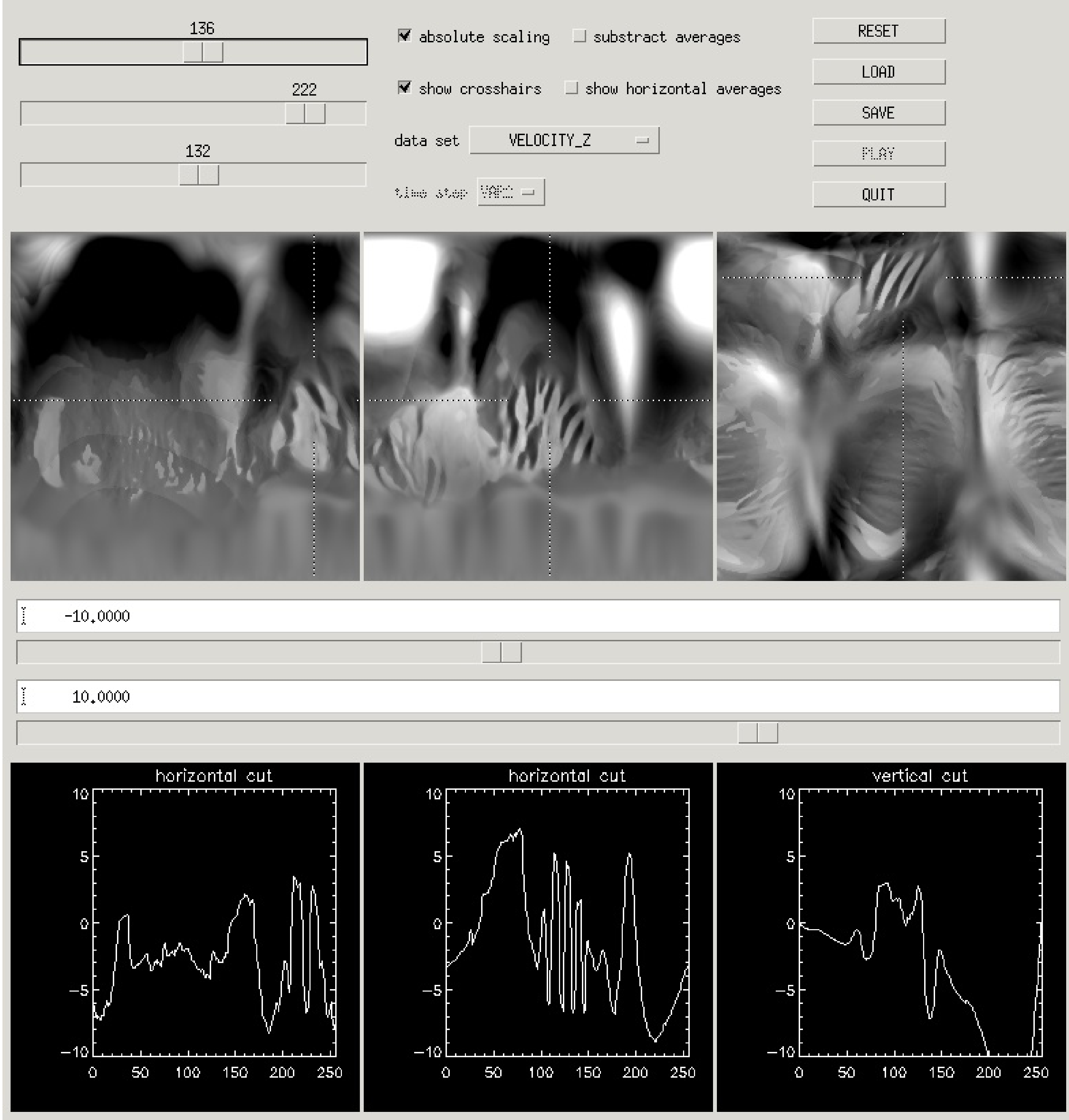

131
55
103
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set
LOGARITHMIC_DENSITY
RESET
LOAD
SAVE
PLAY
QUIT
-1.00000
1.00000
horizontal cut
horizontal cut
vertical cut
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250

Granularity effects in physical observables

Vertical velocity:

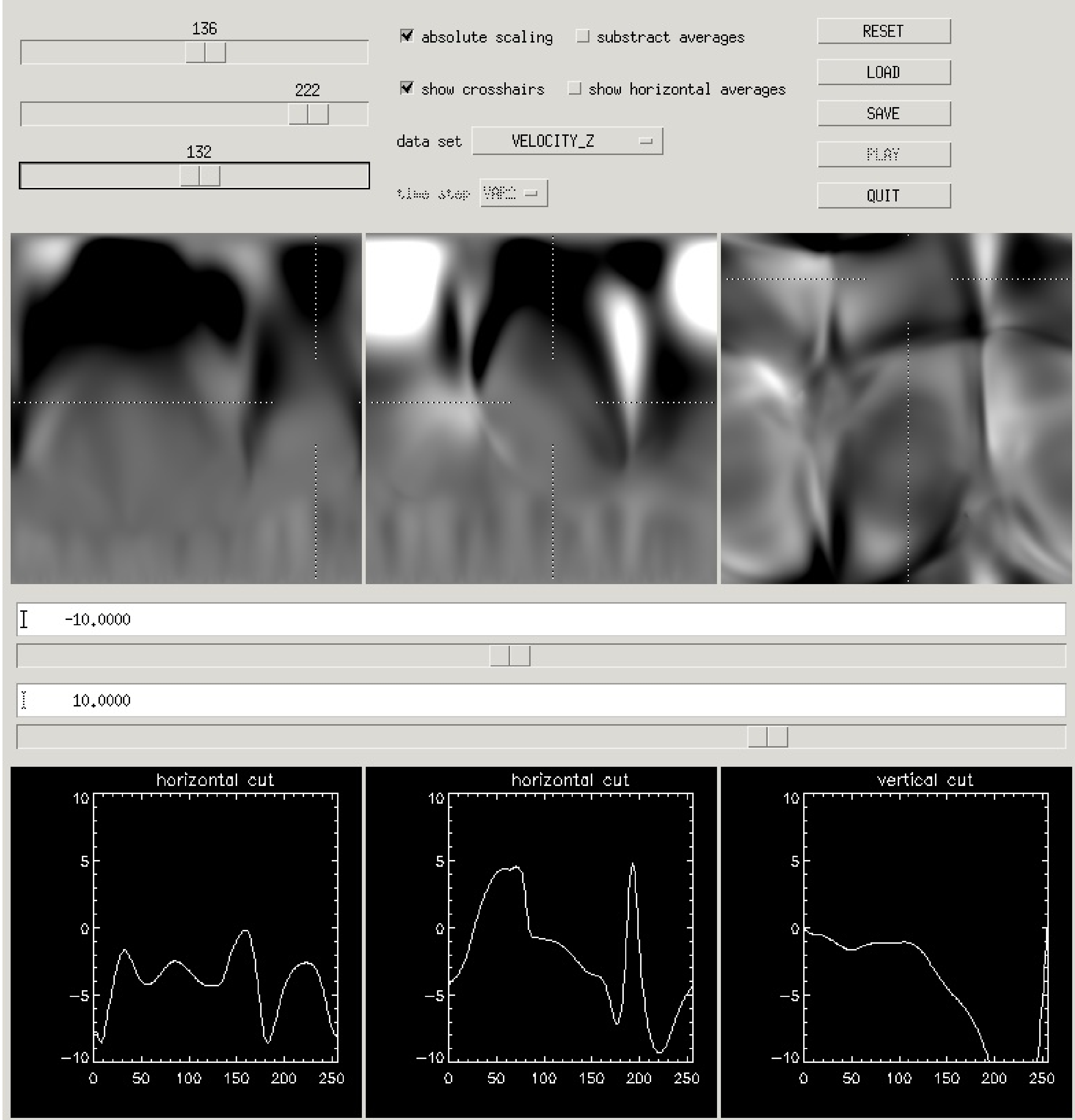
Vz:

SP


136
222
132
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set
VELOCITY_Z
RESET
LOAD
SAVE
PLAY
QUIT
-10.0000
10.0000
horizontal cut
horizontal cut
vertical cut
10
5
0
−5
−10
0
50
100
150
200
250

Vz:

DP



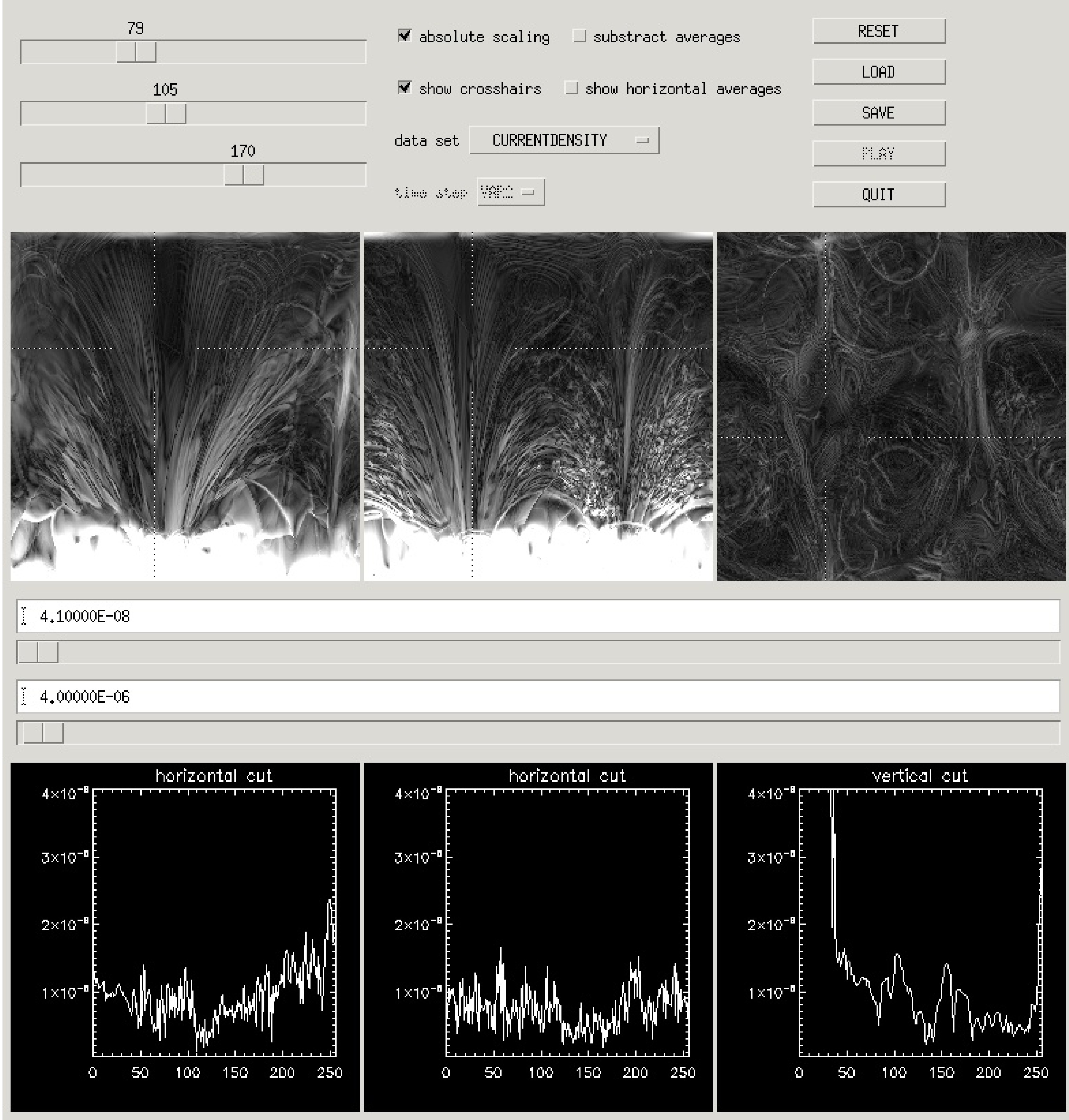
136
222
132
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set
VELOCITY_Z
RESET
LOAD
SAVE
PLAY
QUIT
-10.0000
10.0000
horizontal cut
horizontal cut
vertical cut

Granularity effects in physical observables

Current density / Heating rate: $H = \mu_0 \eta \, \boldsymbol{j}^2 \sim \nabla \times \left(\nabla \times A \right)$

|j|:

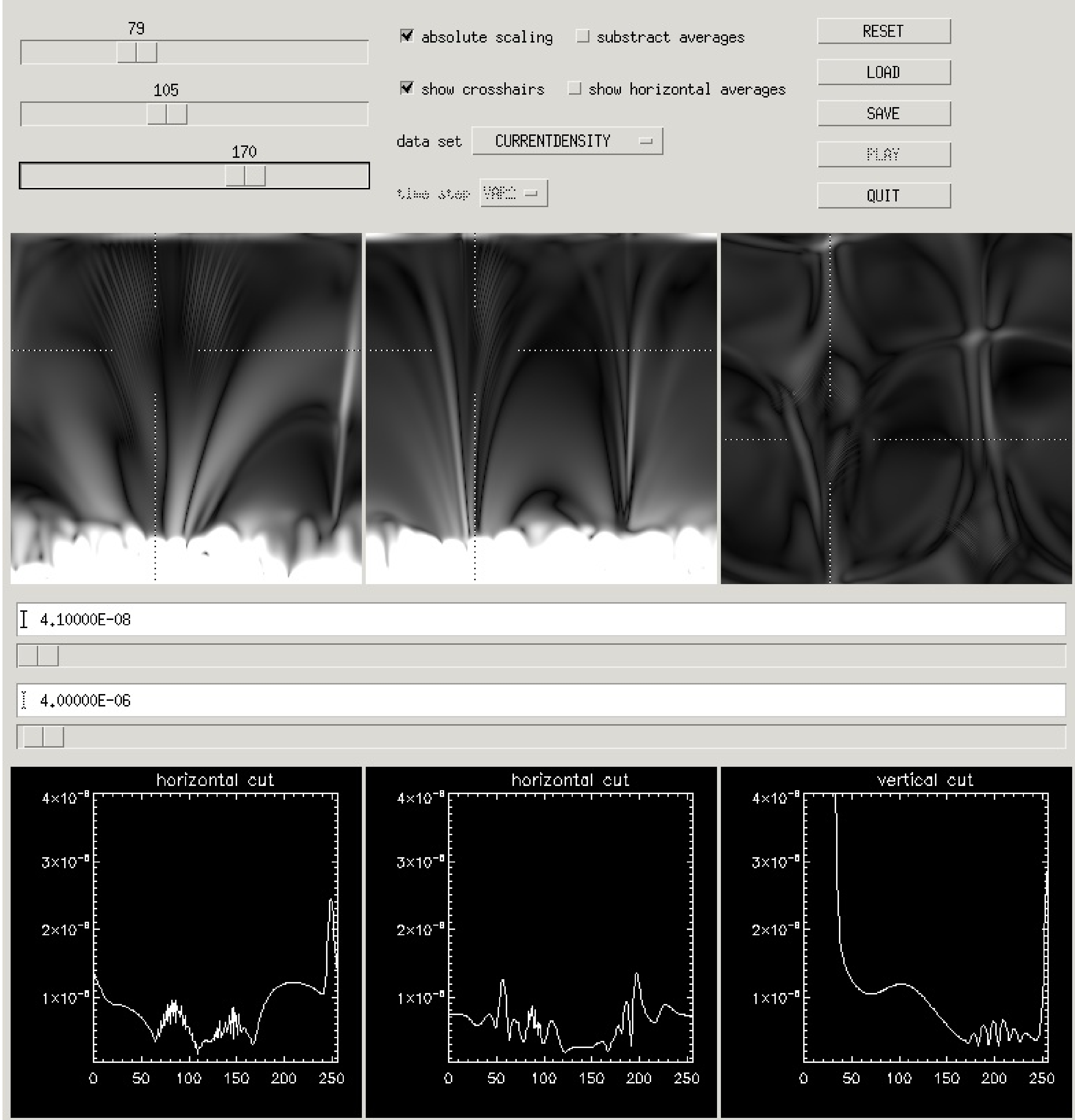
SP



79
105
170
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set
CURRENTDENSITY
RESET
LOAD
SAVE
PLAY
QUIT
4.10000E-08
4.00000E-06
horizontal cut
horizontal cut
vertical cut
4×10⁻⁸
3×10⁻⁸
2×10⁻⁸
1×10⁻⁸
0
50
100
150
200
250

|j|:

DP



79
105
170
absolute scaling
substract averages
show crosshairs
show horizontal averages
data set
CURRENTDENSITY
RESET
LOAD
SAVE
PLAY
QUIT
4.10000E-08
4.00000E-06
horizontal cut
horizontal cut
vertical cut
4×10⁻⁸
3×10⁻⁸
2×10⁻⁸
1×10⁻⁸
0
50
100
150
200
250

Catastrophic cancellation:

“What Every Computer Scientist Should Know About Floating-Point Arithmetic”

(David Goldberg, 1991)

$$\boldsymbol{\nabla} \times \boldsymbol{A} \sim \boldsymbol{B} \quad ; \quad \boldsymbol{\nabla} \times \boldsymbol{B} \sim \boldsymbol{j}$$

Catastrophic cancellation:

“What Every Computer Scientist Should Know About Floating-Point Arithmetic”

(David Goldberg, 1991)

$$\nabla \times \boldsymbol{A} \sim \boldsymbol{B} \quad ; \quad \nabla \times \boldsymbol{B} \sim \boldsymbol{j}$$

$$j_Z \sim \frac{\partial B_Y}{\partial x} - \frac{\partial B_X}{\partial y}$$

$$B_X \approx B_Y$$

Catastrophic cancellation:

“What Every Computer Scientist Should Know About Floating-Point Arithmetic”

(David Goldberg, 1991)

$$\nabla \times \boldsymbol{A} \sim \boldsymbol{B} \quad ; \quad \nabla \times \boldsymbol{B} \sim \boldsymbol{j}$$

$$j_Z \sim \frac{\partial B_Y}{\partial x} - \frac{\partial B_X}{\partial y}$$

$$B_X \approx B_Y \quad \Rightarrow \quad \partial B_X \approx \partial B_Y$$

Granularity effects in physical observables

Catastrophic cancellation:

“What Every Computer Scientist Should Know About Floating-Point Arithmetic”

(David Goldberg, 1991)

$$\nabla \times \boldsymbol{A} \sim \boldsymbol{B} \quad ; \quad \nabla \times \boldsymbol{B} \sim \boldsymbol{j}$$

$$j_Z \sim \frac{\partial B_Y}{\partial x} - \frac{\partial B_X}{\partial y}$$

$$B_X \approx B_Y \quad \Rightarrow \quad \partial B_X \approx \partial B_Y$$

$$\Rightarrow \quad j_Z \approx 2 \cdot \epsilon$$

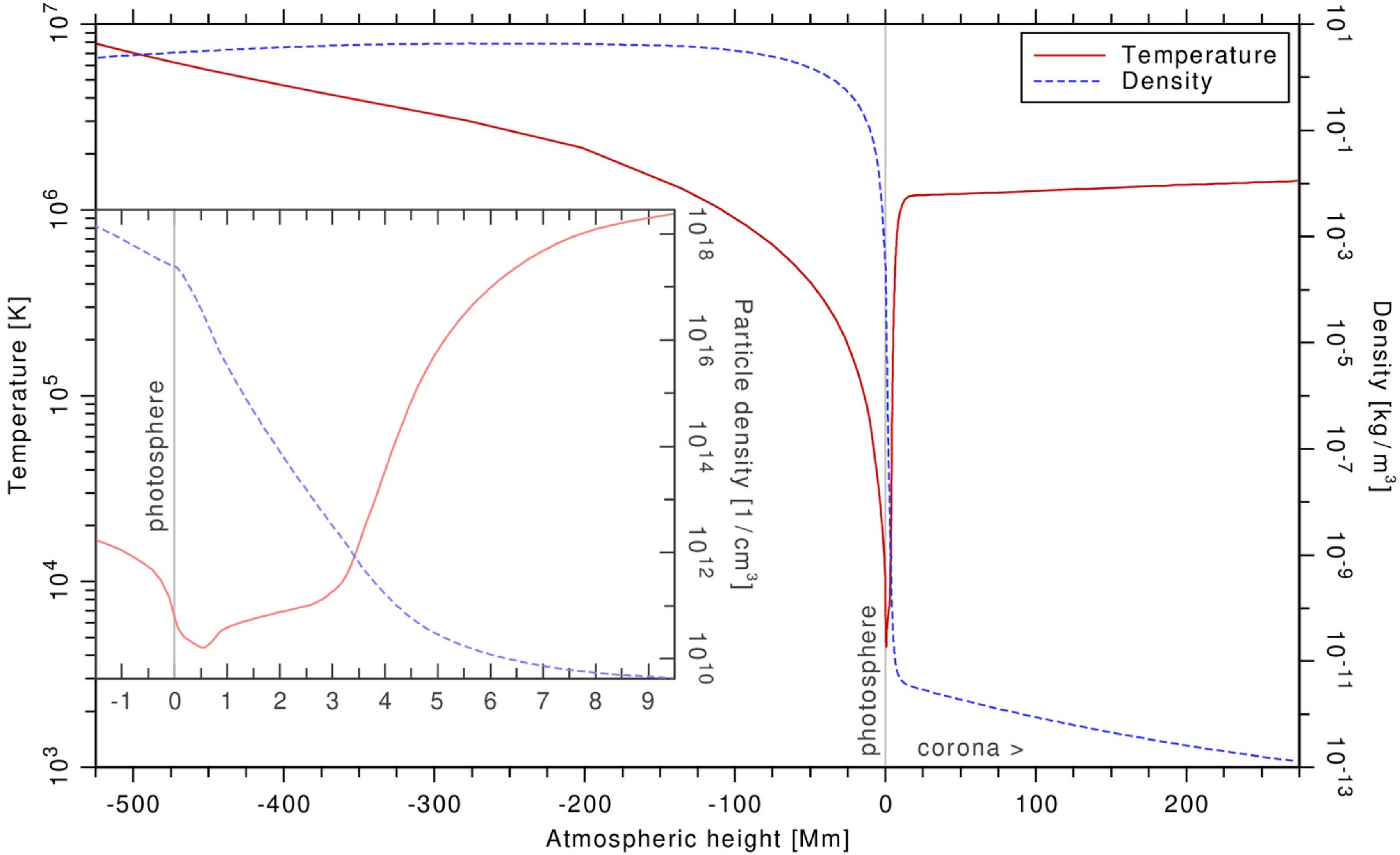
Initial condition

Initial condition

Problem: initial condition might not be in „numerical equilibrium“

Initial condition

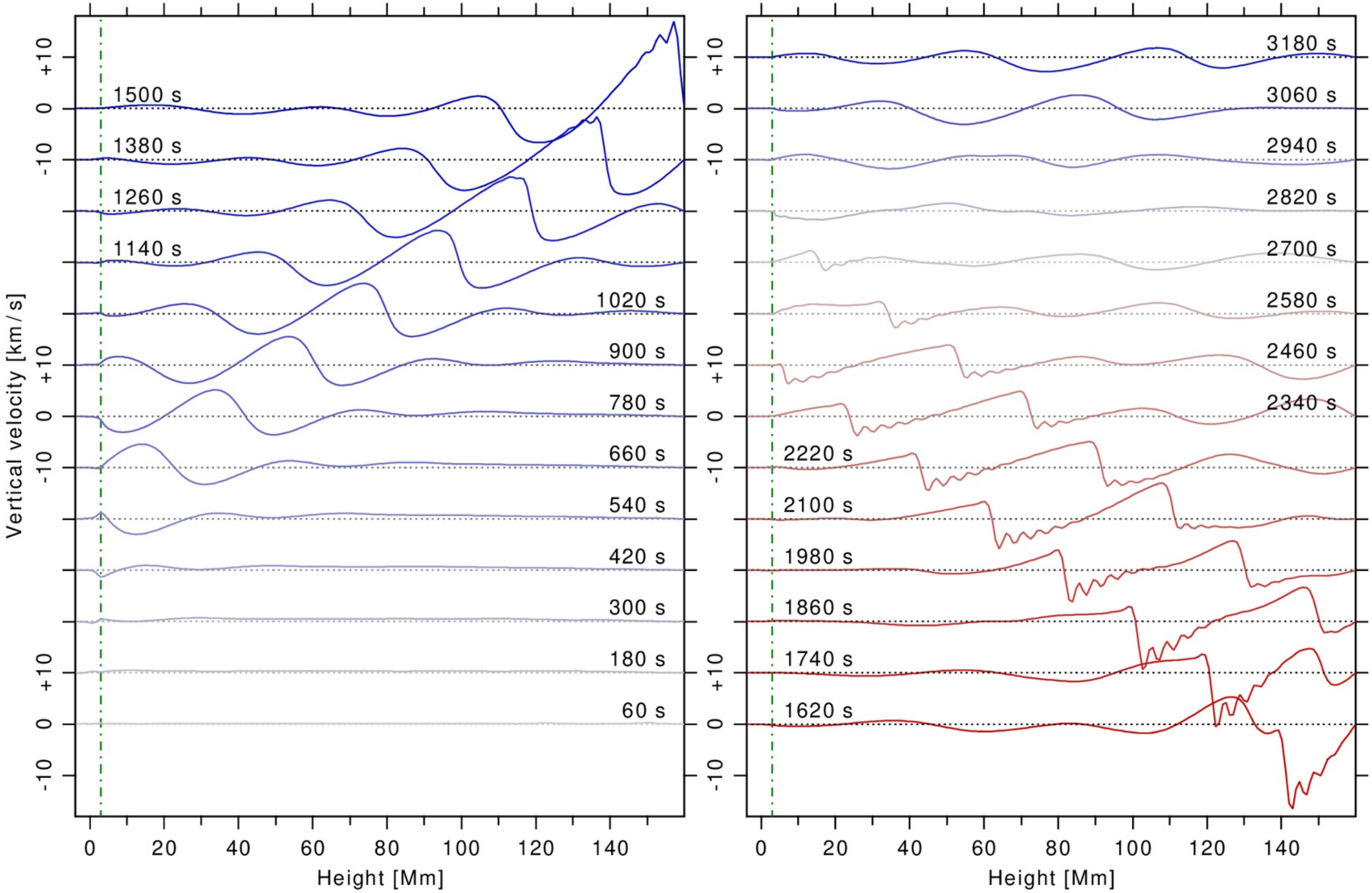
Problem: initial condition might not be in „numerical equilibrium“



=> Solar atmosphere in hydrostatic equilibrium (Bourdin, 2014, CEAB)

Initial condition

Problem: initial condition might not be in „numerical equilibrium“



=> Analytic solution needs time to equilibrate (Bourdin, 2014, CEAB)

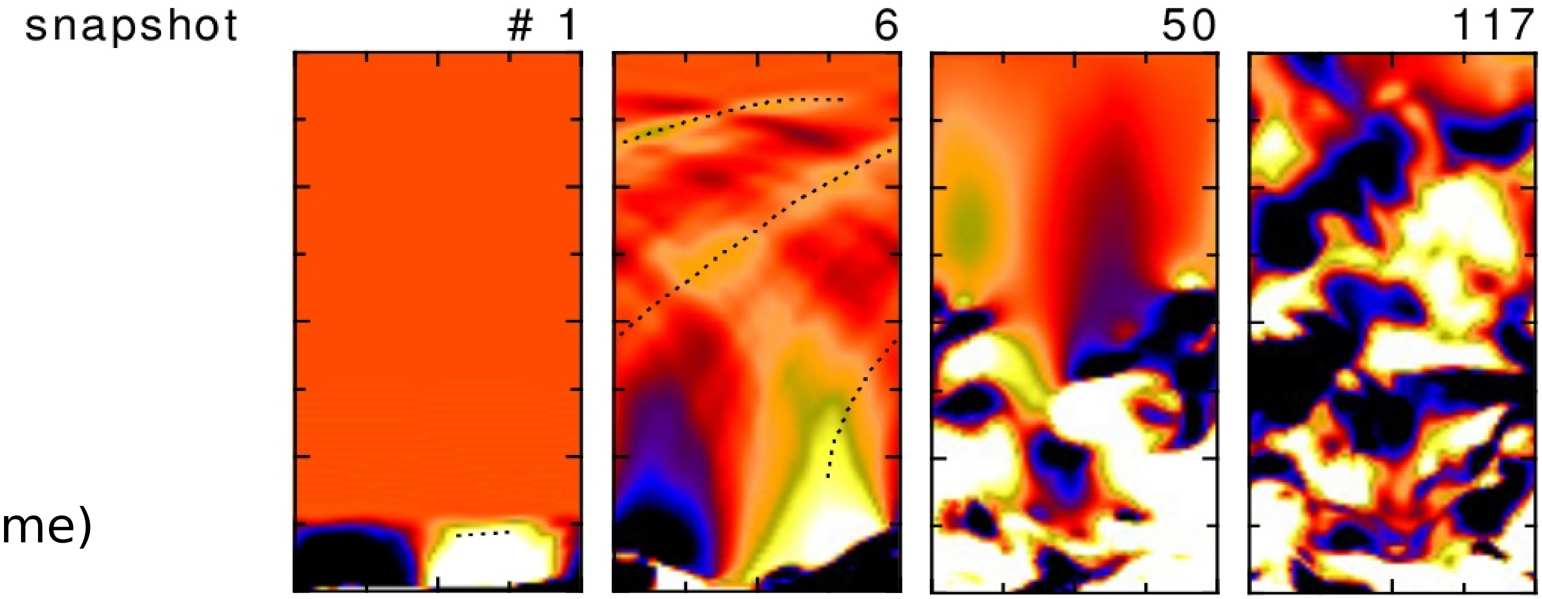
Driving a simulation with external forces

Driving a simulation with external forces

Problem: switching on over small dt => infinite momentum

Driving a simulation with external forces

<u>Problem:</u> switching on over small dt => infinite momentum => shock!

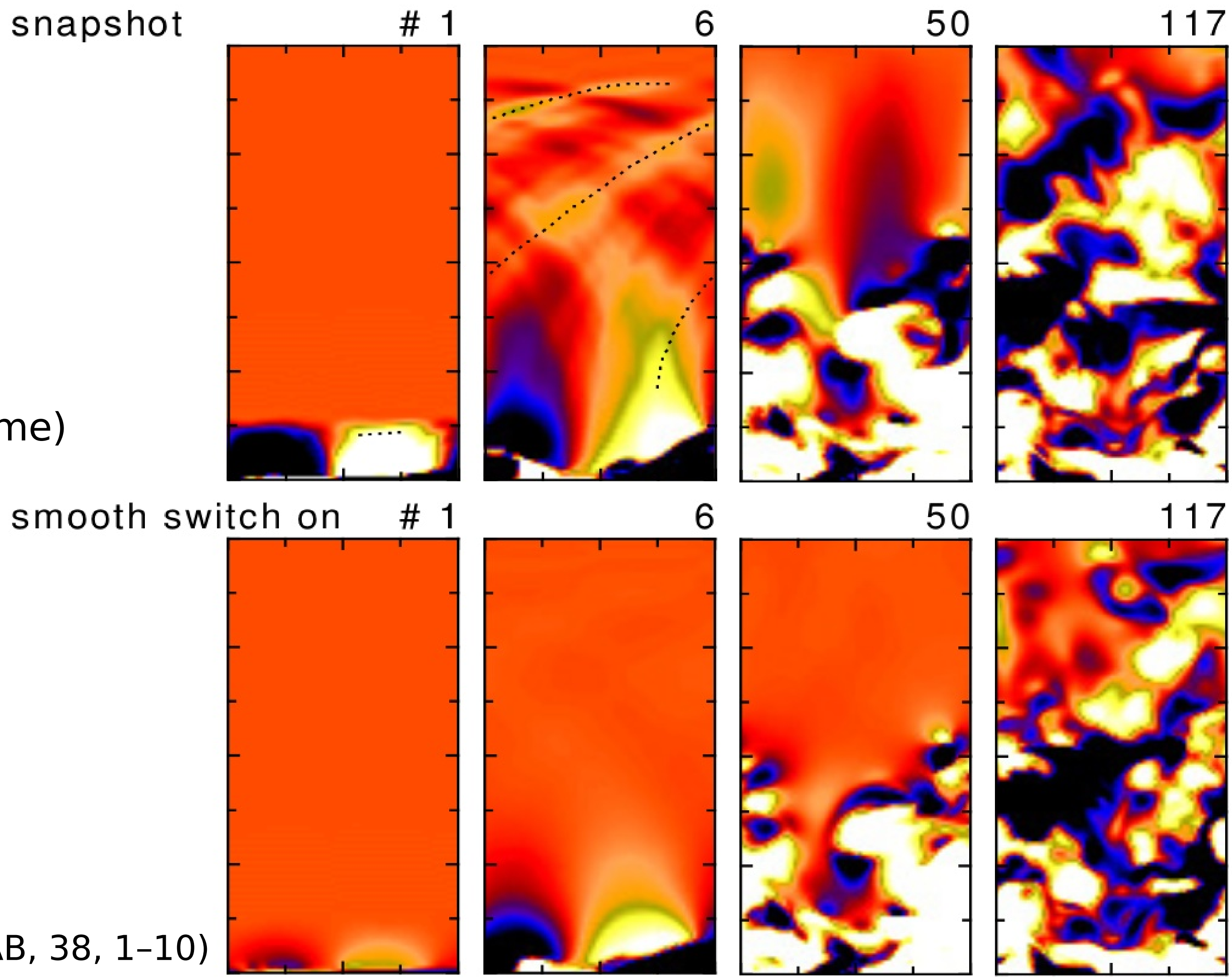


(samples/
turbulent_flame)

(Bourdin, 2014, CEAB, 38, 1–10)

Driving a simulation with external forces

<u>Problem:</u> switching on over small dt => infinite momentum => shock!



(samples/

turbulent_flame)

(Bourdin, 2014, CEAB, 38, 1–10)