

# High-Order Finite Difference Methods



**Wladimir Lyra**

New Mexico State University

# Discretization of Equations

- The first step in solving any computational fluid dynamics problem is to discretize the equations
- Usually all fluid dynamics equations are in a partial differential equation format
- These differential equations must be transformed into algebraic form, so that they can become solvable
- The most straightforward discretization technique for partial differential equations is the finite difference method

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

# Numerical methods: properties

Finite differences

robust, simple concept, easy to parallelize, regular grids, explicit method

Finite elements

implicit approach, matrix inversion, well founded,  
irregular grids, more complex algorithms, engineering problems

Finite volumes

robust, simple concept, irregular grids, explicit method

# What is a finite difference?

Common definitions of the derivative of  $f(x)$ :

$$\partial_x f = \lim_{dx \rightarrow 0} \frac{f(x + dx) - f(x)}{dx}$$

$$\partial_x f = \lim_{dx \rightarrow 0} \frac{f(x) - f(x - dx)}{dx}$$

$$\partial_x f = \lim_{dx \rightarrow 0} \frac{f(x + dx) - f(x - dx)}{2dx}$$

These are all correct definitions in the limit  $dx \rightarrow 0$ .

But we want  $dx$  to remain **FINITE**

# What is a finite difference?

The equivalent **approximations** of the derivatives are:

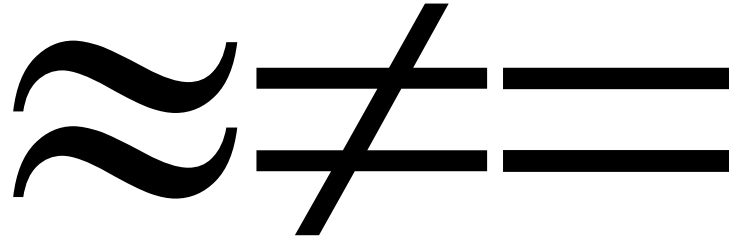
$$\partial_x f^+ \approx \frac{f(x + dx) - f(x)}{dx} \quad \text{forward difference}$$

$$\partial_x f^- \approx \frac{f(x) - f(x - dx)}{dx} \quad \text{backward difference}$$

$$\partial_x f \approx \frac{f(x + dx) - f(x - dx)}{2dx} \quad \text{centered difference}$$

# The **BIG** question

How good are the finite difference approximations?



This leads us to Taylor series....

# Taylor Series

Taylor series are expansions of a function  $f(x)$  for some finite distance  $dx$  to  $f(x+dx)$

$$f(x \pm dx) = f(x) \pm dx f'(x) + \frac{dx^2}{2!} f''(x) \pm \frac{dx^3}{3!} f'''(x) + \frac{dx^4}{4!} f^{(4)}(x) \pm \dots$$

What happens, if we use this expression for

$$\partial_x f^+ \approx \frac{f(x+dx) - f(x)}{dx} \quad ?$$

# Taylor Series

... that leads to :

$$\begin{aligned}\frac{f(x+dx) - f(x)}{dx} &= \frac{1}{dx} \left[ dx f'(x) + \frac{dx^2}{2!} f''(x) + \frac{dx^3}{3!} f'''(x) + \dots \right] \\ &= f'(x) + O(dx)\end{aligned}$$

The error of the first derivative using the *forward* formulation is *of order dx*.

Is this the case for other formulations of the derivative?  
Let's check!



# Taylor Series

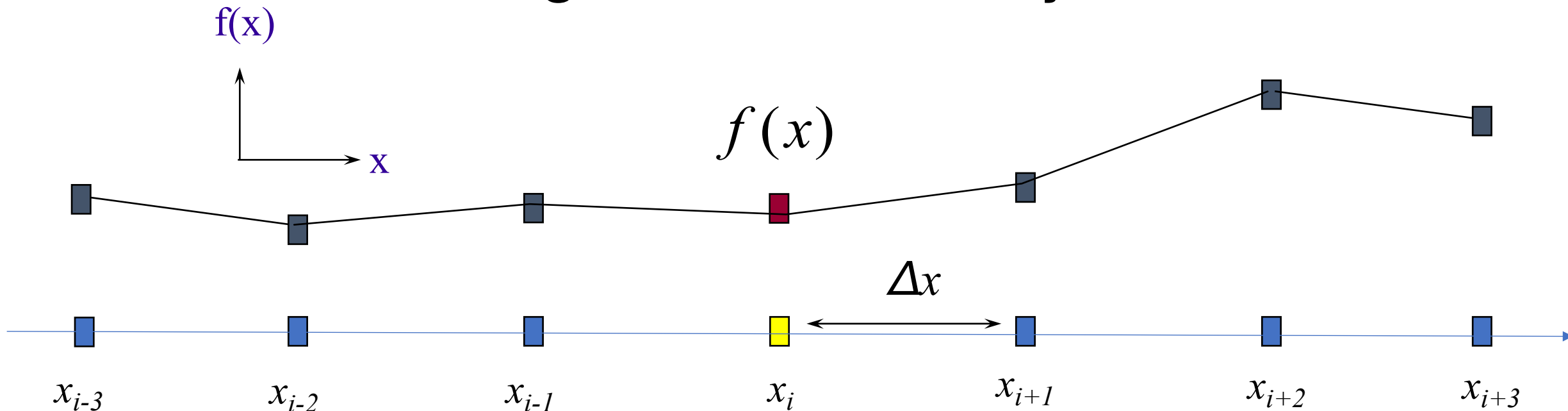
... with the *centered* formulation we get:

$$\begin{aligned}\frac{f(x + dx/2) - f(x - dx/2)}{dx} &= \frac{1}{dx} \left[ dx f'(x) + \frac{dx^3}{3!} f'''(x) + \dots \right] \\ &= f'(x) + O(dx^2)\end{aligned}$$

The error of the first derivative using the centered approximation is *of order*  $dx^2$ .

This is an **important** results: it DOES matter which formulation we use. The centered scheme is more accurate!

# Higher-order accuracy



What is the (approximate) value of the first derivative at the desired location?

The first derivative, to 2nd order accuracy is

$$f'_i = \frac{-f_{i-1} + f_{i+1}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$

to 4th order accuracy, it is

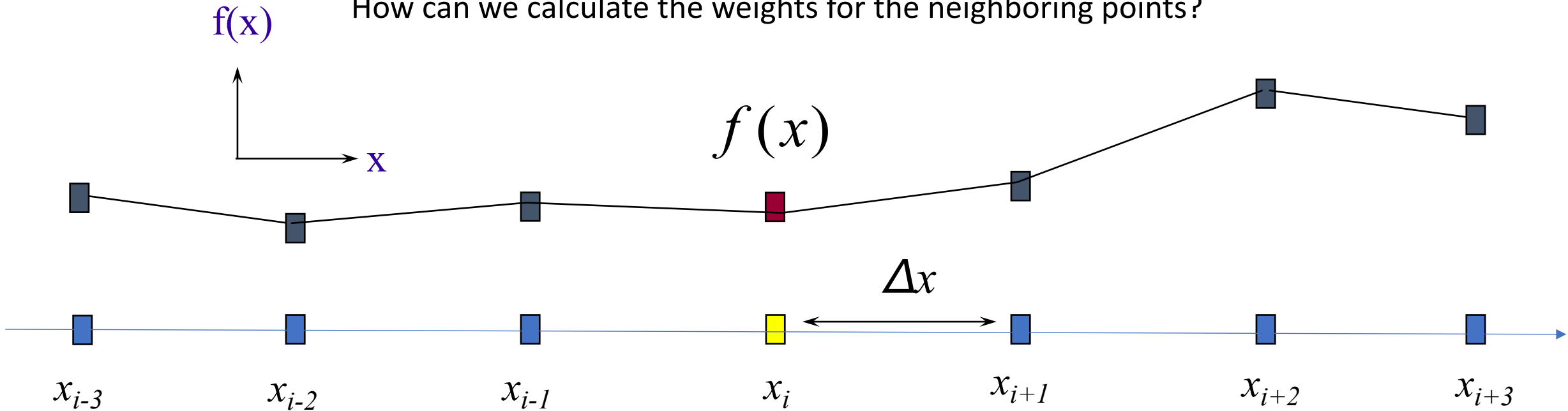
$$f'_i = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12\Delta x} + \mathcal{O}(\Delta x^4)$$

to 6th order accuracy, it is

$$f'_i = \frac{-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3}}{60\Delta x} + \mathcal{O}(\Delta x^6)$$

# Finite Difference Coefficients

How can we calculate the weights for the neighboring points?



$$\sum_{i=0}^{N-1} s_i^n c_i = d! \delta(n - d) \quad \text{for } 0 < n < N - 1$$

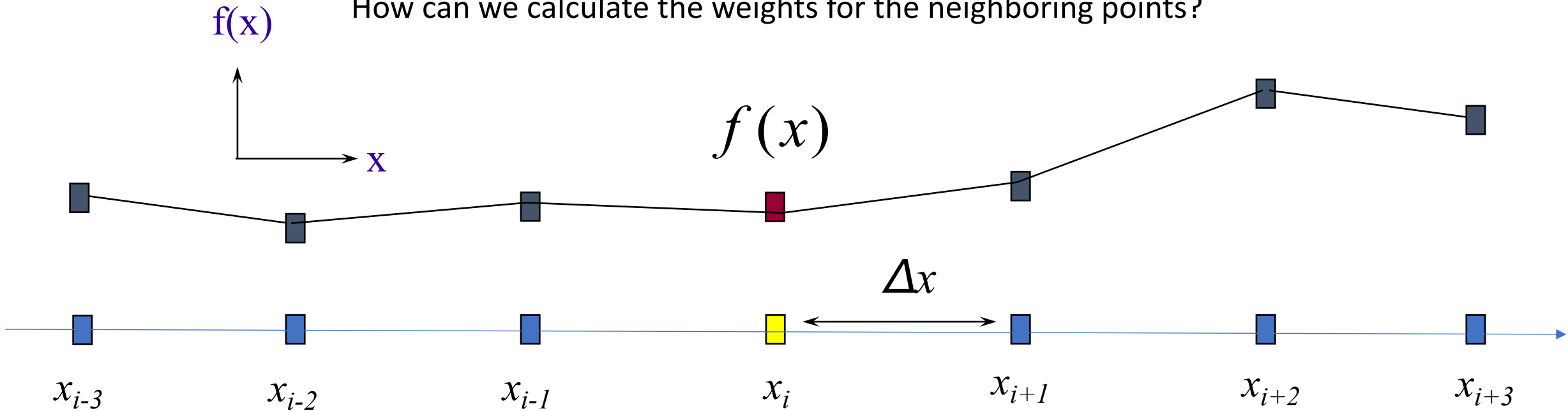
$s = (-3, -2, -1, 0, 1, 2, 3)$  stencil positions

$N$  = size of stencil

$d$  = order of the derivative

# Finite Difference Coefficients

How can we calculate the weights for the neighboring points?



$$\sum_{i=0}^{N-1} s_i^n c_i = d! \delta(n - d) \quad \text{for } 0 < n < N - 1$$

$s = (-3, -2, -1, 0, 1, 2, 3)$  stencil positions

$N$  = size of stencil

$d$  = order of the derivative

# Finite difference coefficient

7 languages

Article Talk

Read Edit View history Tools

From Wikipedia, the free encyclopedia

In mathematics, to approximate a derivative to an arbitrary order of accuracy, it is possible to use the **finite difference**. A finite difference can be **central**, **forward** or **backward**.

## Central finite difference [edit]

This table contains the coefficients of the **central** differences, for several orders of accuracy and with uniform grid spacing:<sup>[1]</sup>

Derivative	Accuracy	−5	−4	−3	−2	−1	0	1	2	3	4	5
1	2					−1/2	0	1/2				
	4				1/12	−2/3	0	2/3	−1/12			
	6			−1/60	3/20	−3/4	0	3/4	−3/20	1/60		
	8		1/280	−4/105	1/5	−4/5	0	4/5	−1/5	4/105	−1/280	
2	2					1	−2	1				
	4				−1/12	4/3	−5/2	4/3	−1/12			
	6			1/90	−3/20	3/2	−49/18	3/2	−3/20	1/90		
	8		−1/560	8/315	−1/5	8/5	−205/72	8/5	−1/5	8/315	−1/560	
3	2				−1/2	1	0	−1	1/2			
	4			1/8	−1	13/8	0	−13/8	1	−1/8		
	6		−7/240	3/10	−169/120	61/30	0	−61/30	169/120	−3/10	7/240	
4	2				1	−4	6	−4	1			
	4			−1/6	2	−13/2	28/3	−13/2	2	−1/6		
	6		7/240	−2/5	169/60	−122/15	91/8	−122/15	169/60	−2/5	7/240	
5	2			−1/2	2	−5/2	0	5/2	−2	1/2		
	4		1/6	−3/2	13/3	−29/6	0	29/6	−13/3	3/2	−1/6	
	6	−13/288	19/36	−87/32	13/2	−323/48	0	323/48	−13/2	87/32	−19/36	13/288
6	2			1	−6	15	−20	15	−6	1		
	4		−1/4	3	−13	29	−75/2	29	−13	3	−1/4	
	6	13/240	−19/24	87/16	−39/2	323/8	−1023/20	323/8	−39/2	87/16	−19/24	13/240

$$\sum_{i=0}^{N-1} s_i^n c_i = d! \delta(n - d) \quad \text{for } 0 < n < N - 1$$

s = (-3,-2,-1,0,1,2,3) stencil positions

N = size of stencil

d = order of the derivative

# Finite Difference Coefficients

These coefficients come from Taylor expansion. Suppose that we want to compute  $df/dx$  to 2nd order. Using a 3-point stencil, we have

$$\frac{df}{dx} = \frac{1}{h} [Af(x-h) + Bf(x) + Cf(x+h)]$$

where  $h = \Delta x$ . According to the table above, we expect to find  $A = -1/2$ ,  $B = 0$  and  $C = 1/2$ . Let us prove this.

If we Taylor expand around  $x$ ,

$$Af(x-h) = Af(x) + Af'(x)(-h) + Af''(x)\frac{h^2}{2}$$

$$Bf(x) = Bf(x)$$

$$Cf(x+h) = Cf(x) + Cf'(x)(h) + Cf''(x)\frac{h^2}{2}$$

Summing them all

$$h\frac{df}{dx} = f(x)(A+B+C) + f'(x)(A-C)h + f''(x)\frac{h^2}{2}(A+C)$$

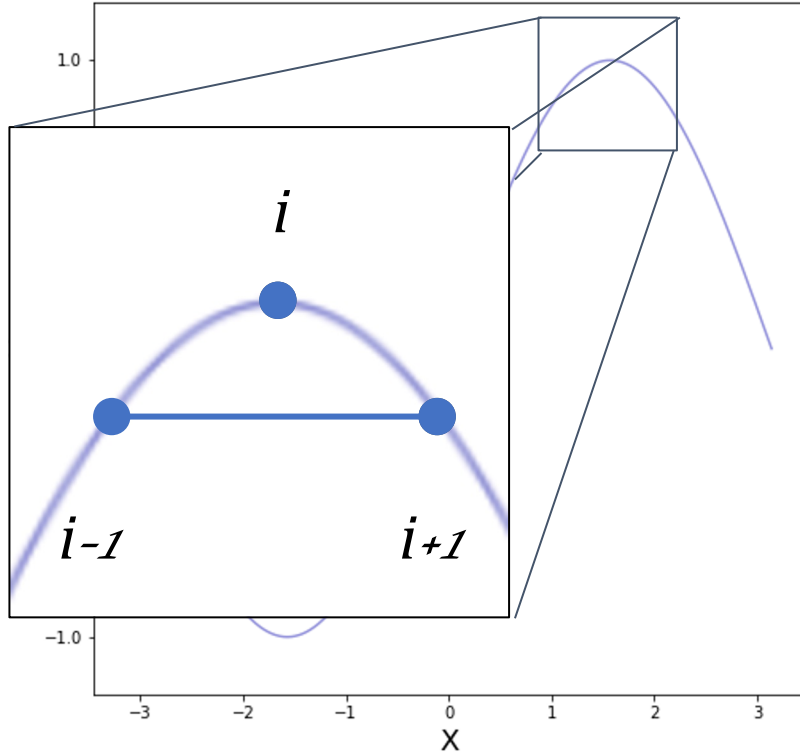
Since only the first derivative should survive in the RHS, this leads to the conditions

$$A + B + C = 0$$

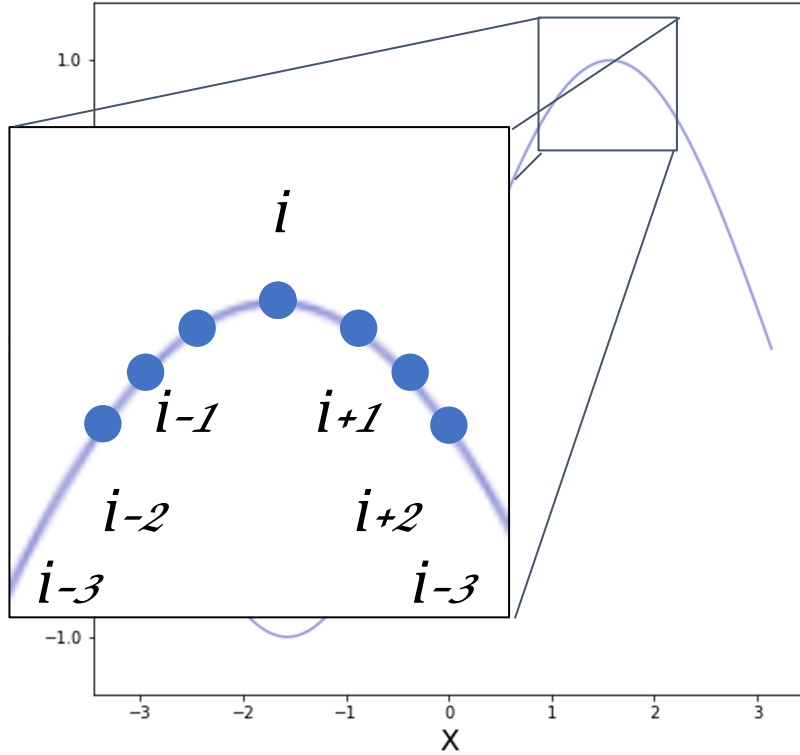
$$A - C = 1$$

$$A + C = 0$$

Leading to  $A = -1/2$ ,  $C = 1/2$ ,  $B = 0$ , as expected.



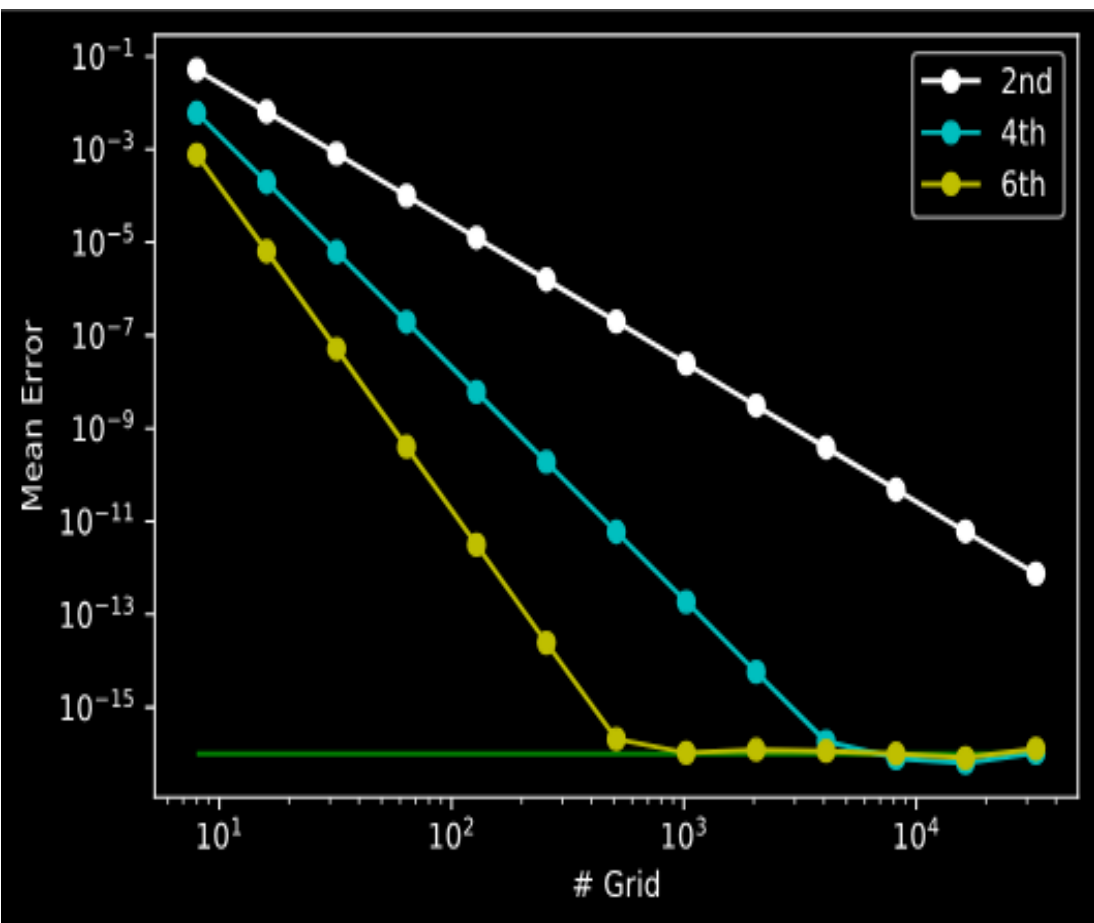
$$f'_i = \frac{-f_{i-1} + f_{i+1}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$



$$f'_i = \frac{-f_{i-3} + 9f_{i-2} - 45f_{i-1} + 45f_{i+1} - 9f_{i+2} + f_{i+3}}{60\Delta x} + \mathcal{O}(\Delta x^6)$$



# Higher-order accuracy



- Higher order derivative = smaller truncation error
- Higher orders naturally require a larger stencil (more ghost zones)
- Higher order derivatives approach machine precision faster.
- **Alternative to spectral schemes**

# High-order finite-difference vs Spectral

- Calculate  $k_{\text{eff}}$  that is returned by the numerical derivative

$$k_{\text{eff}} = -\frac{1}{\sin kx} \frac{d(\cos kx)}{dx}$$

$$k_{\text{eff}}^2 = -\frac{1}{\cos kx} \frac{d^2(\cos kx)}{dx^2}$$

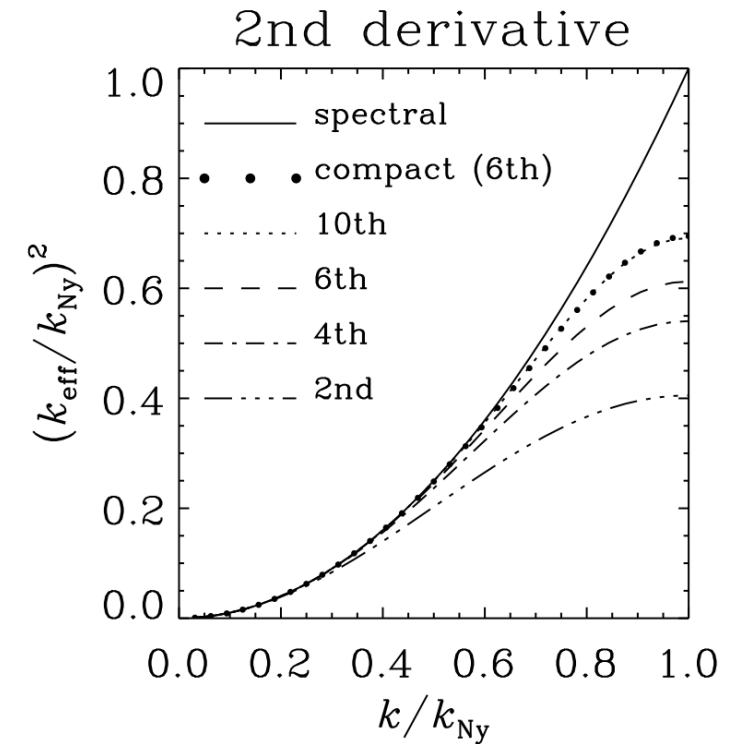
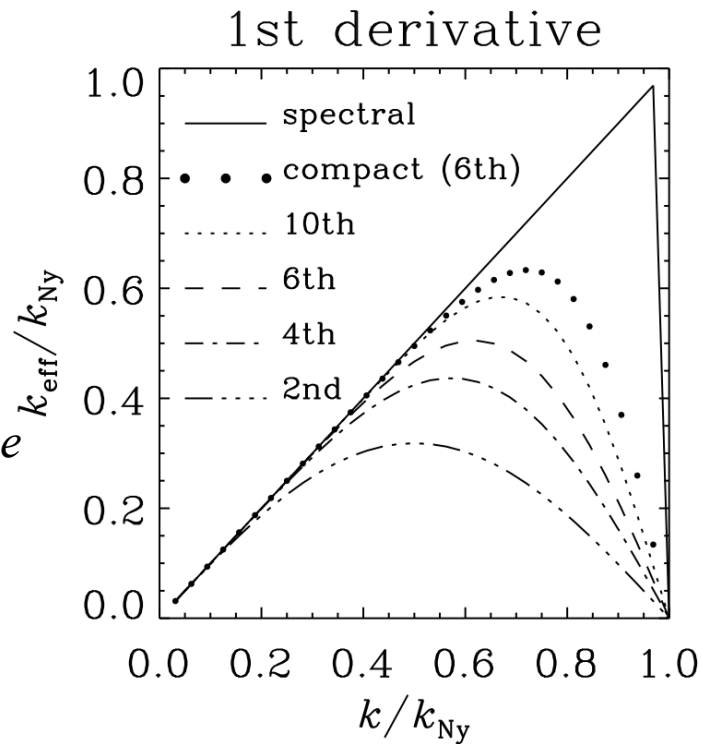
Or, to avoid dividing by zero

- Define a vector  $\mathbf{A} = (0, \sin kx, \cos kx)$ .
- Note that  $\mathbf{B} = \nabla \times \mathbf{A} = k\mathbf{A}$ .
- Evaluate numerically  $\mathbf{A} \cdot \mathbf{B}$
- Since  $|\mathbf{A}|=1$  we find immediately the *effective* wavenumber as  $k_{\text{eff}} = \langle \mathbf{A} \cdot \mathbf{B} \rangle$ .

Similarly for the second derivative

- $k_{\text{eff}}^2 = \langle \mathbf{A} \cdot \mathbf{J} \rangle$ .

where  $\mathbf{J} = -\nabla^2 \mathbf{A}$ ,



# High-order finite-difference vs Spectral

- Calculate  $k_{\text{eff}}$  and  $k_{\text{eff}}^2$  returned by the numerical derivatives

$$k_{\text{eff}} = -\frac{1}{\sin kx} \frac{d(\cos kx)}{dx}$$

$$k_{\text{eff}}^2 = -\frac{1}{\cos kx} \frac{d^2(\cos kx)}{dx^2}$$

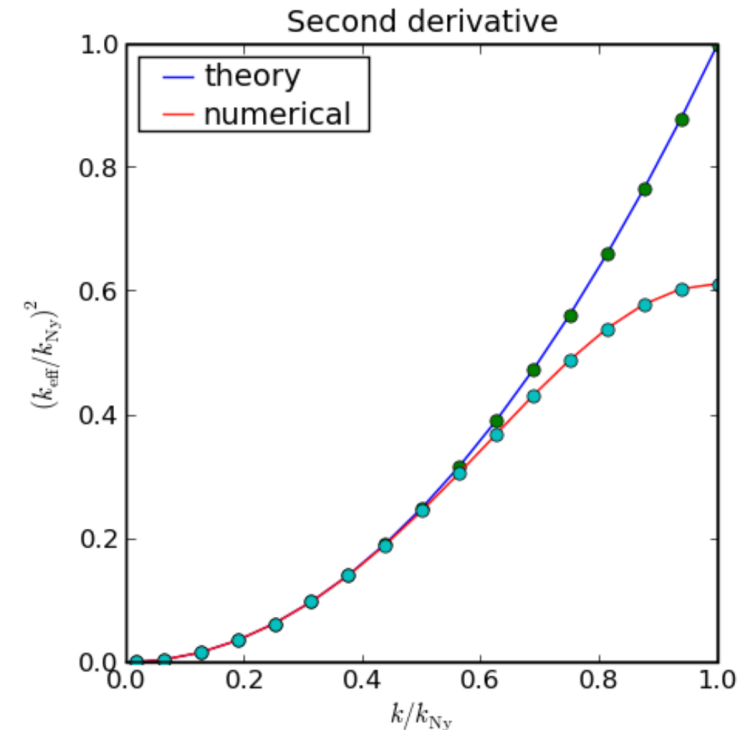
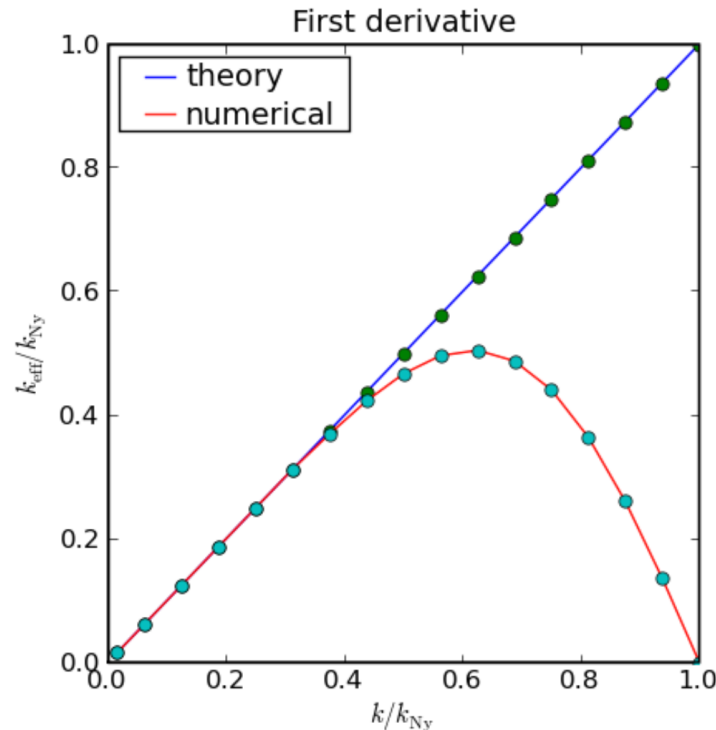
Or, to avoid dividing by zero

- Define a vector  $\mathbf{A} = (0, \sin kx, \cos kx)$ .
- Note that  $\mathbf{B} = \nabla \times \mathbf{A} = k\mathbf{A}$ .
- Evaluate numerically  $\mathbf{A} \cdot \mathbf{B}$
- Since  $|\mathbf{A}|=1$  we find immediately the *effective* wavenumber as  $k_{\text{eff}} = \langle \mathbf{A} \cdot \mathbf{B} \rangle$ .

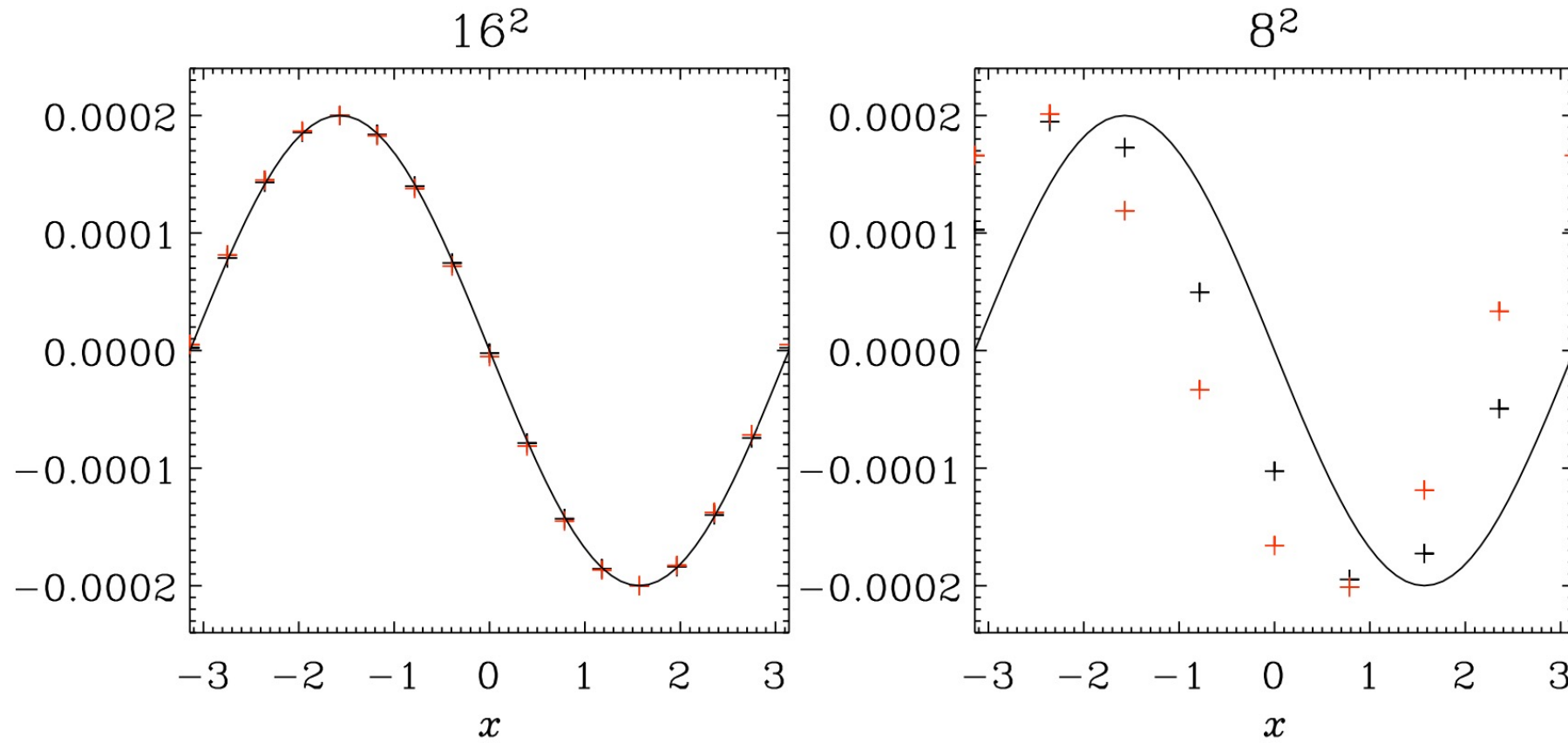
Similarly for the second derivative

- $k_{\text{eff}}^2 = \langle \mathbf{A} \cdot \mathbf{J} \rangle$ .

where  $\mathbf{J} = -\nabla^2 \mathbf{A}$ ,



# Advection test – Alfvén wave



# Gibbs Phenomena

A common criticism of high order schemes is their tendency to produce Gibbs phenomena (ripples) near discontinuities. Consequently one needs a small amount of diffusion to damp out the modes near the Nyquist frequency. Thus, one needs to replace eq. (50) by the equation

$$\dot{f} = -uf' + \nu f''. \quad (56)$$

The question is now how much diffusion is necessary, and how this depends on the spatial order of the scheme.

In figure 2 we plot the result of advecting the periodic step-like function,  $f(kx)$ , over 5 wavelengths, corresponding to a time  $T = L/u$ . The goal is to find the minimum diffusion coefficient  $\nu$  necessary to avoid wiggles in the solution. In the first two panels one sees that for a 6th order scheme the diffusion coefficient has to be approximately  $\nu = 0.01u\delta x$ . For  $\nu = 0.005u\delta x$  there are still wiggles. For a 10th order scheme one can still use  $\nu = 0.005u\delta x$  without producing wiggles, while for a spectral scheme of nearly infinite order one can go down to  $\nu = 0.002u\delta x$  without any problems.

We may thus conclude that all these schemes need some diffusion, but that the diffusion coefficient can be much reduced when the spatial order of the scheme is high. In that sense it is therefore not true that high order schemes are particularly vulnerable to Gibbs phenomena, but rather the contrary!

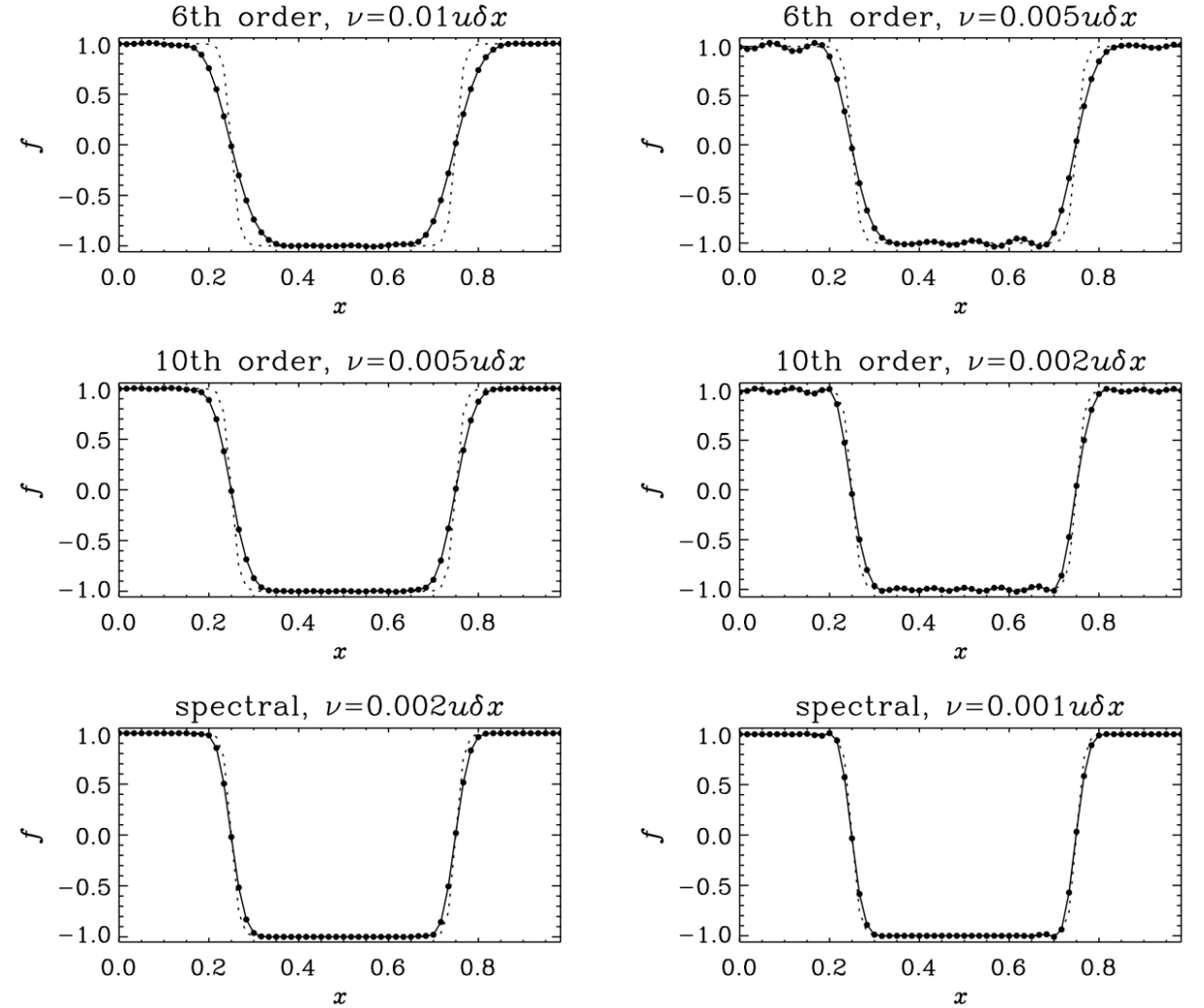
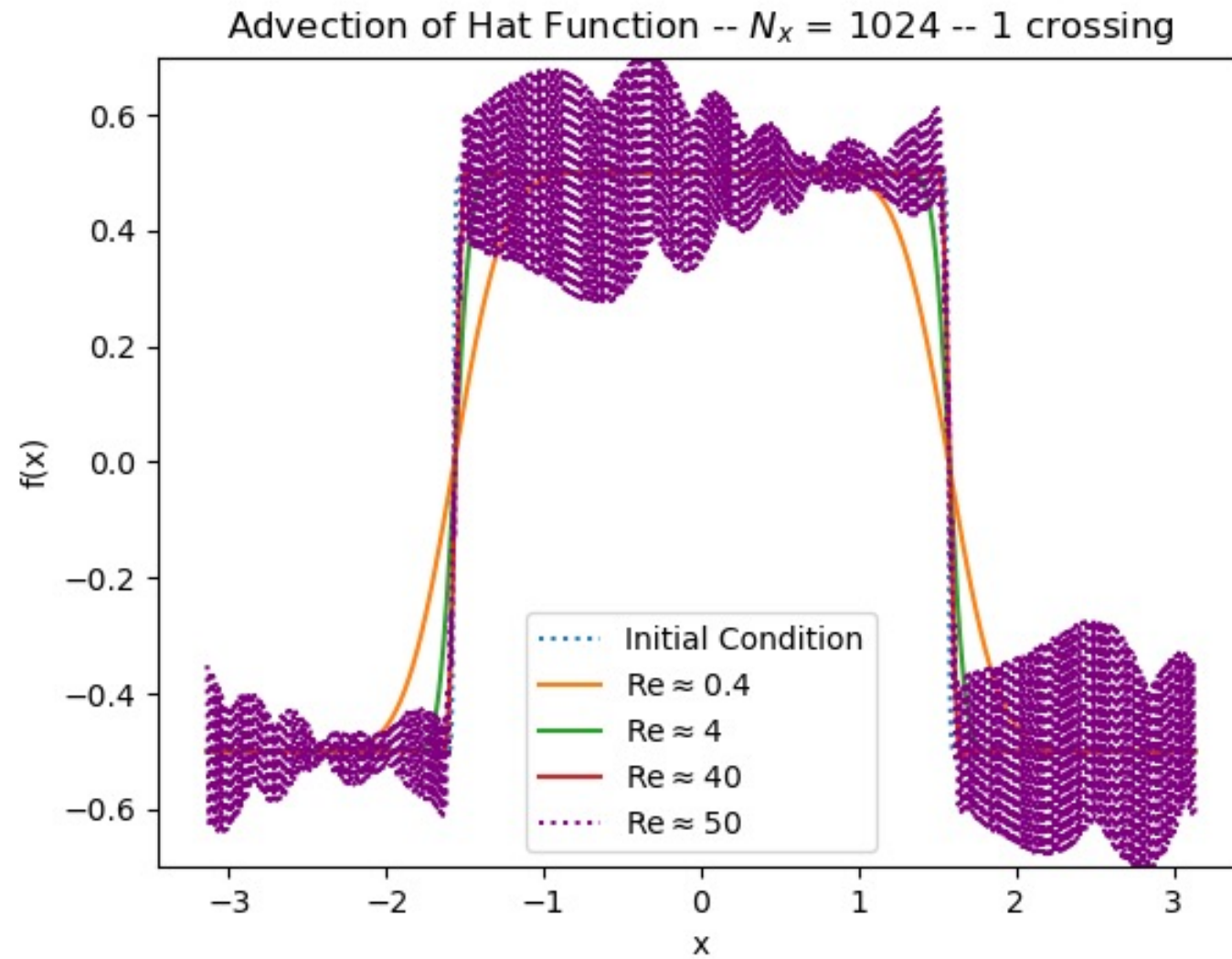
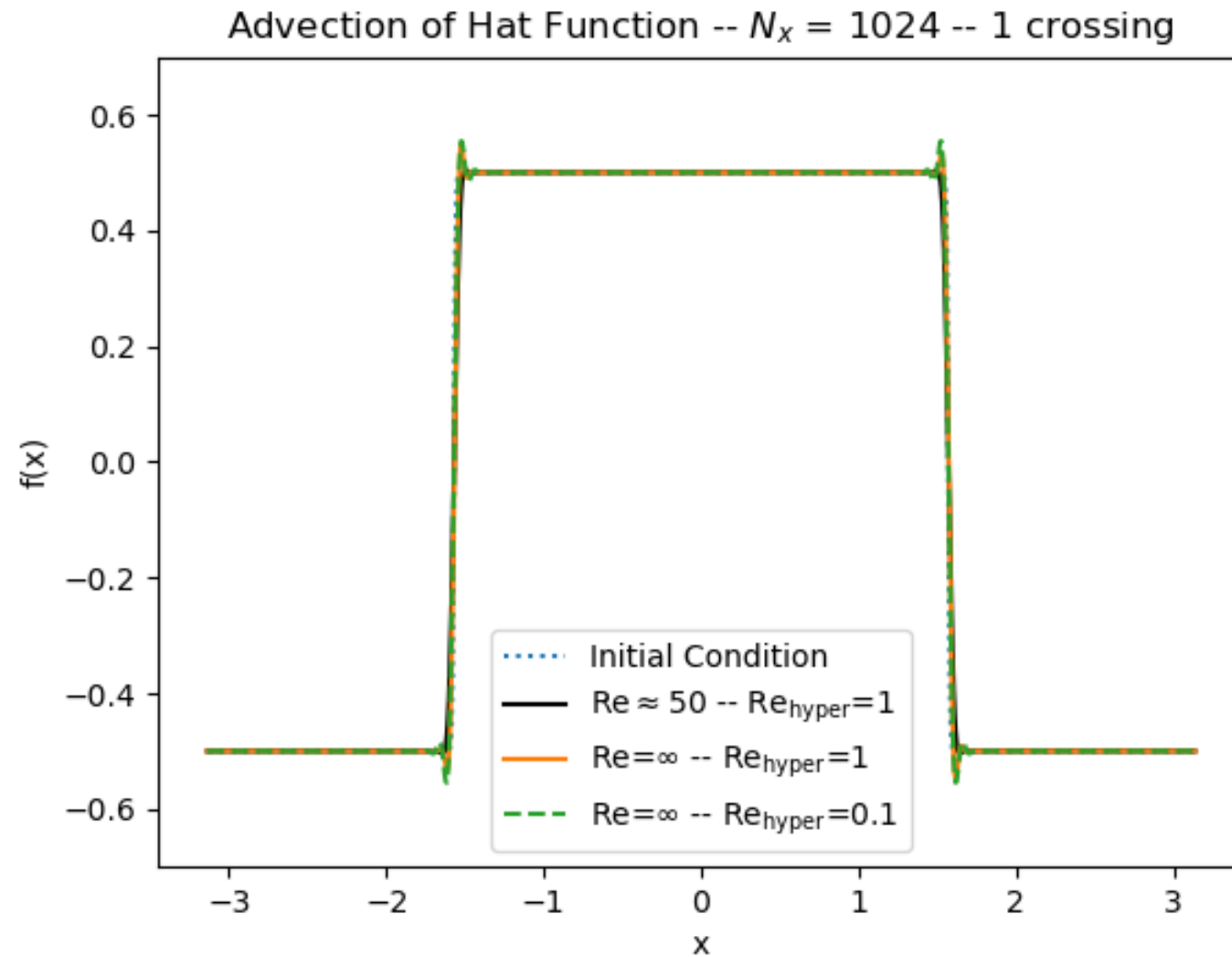


Figure 2: Resulting profile after advecting a step-like function 5 times through the periodic mesh. The dots on the solid line give the location of the function values at the computed meshpoints and the dotted line gives the original profile. For the panels on the right hand side the diffusion coefficient is too small and the profile shows noticeable wiggles.  $\delta x = 1/60$ .

# Hyperdiffusion

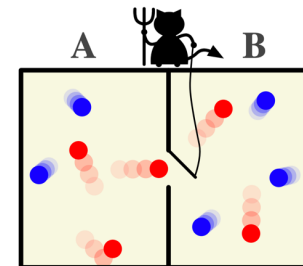


# Hyperdiffusion



# Hyperdiffusion

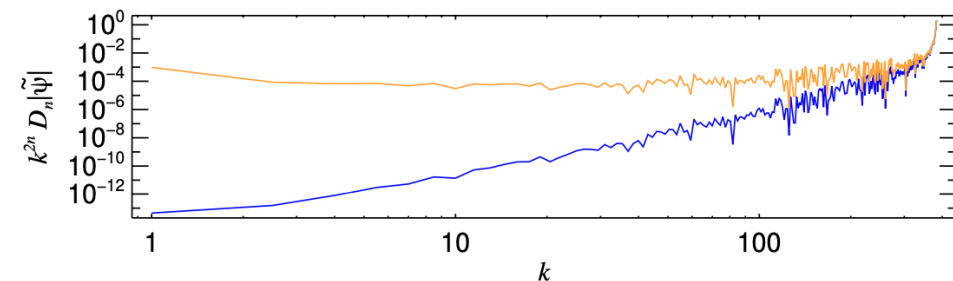
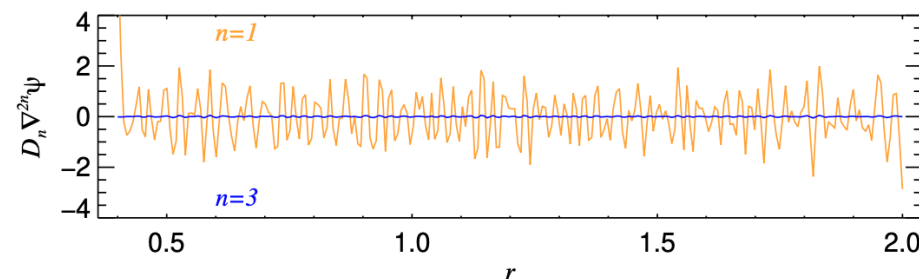
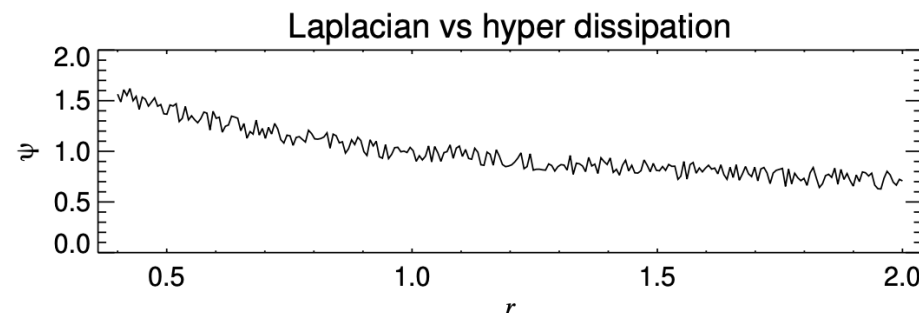
Hyper-dissipation allows **low Reynolds number** at the **grid scale** while maintaining **high Reynolds number** at the **inertial range**.



$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathcal{J} = 0.$$

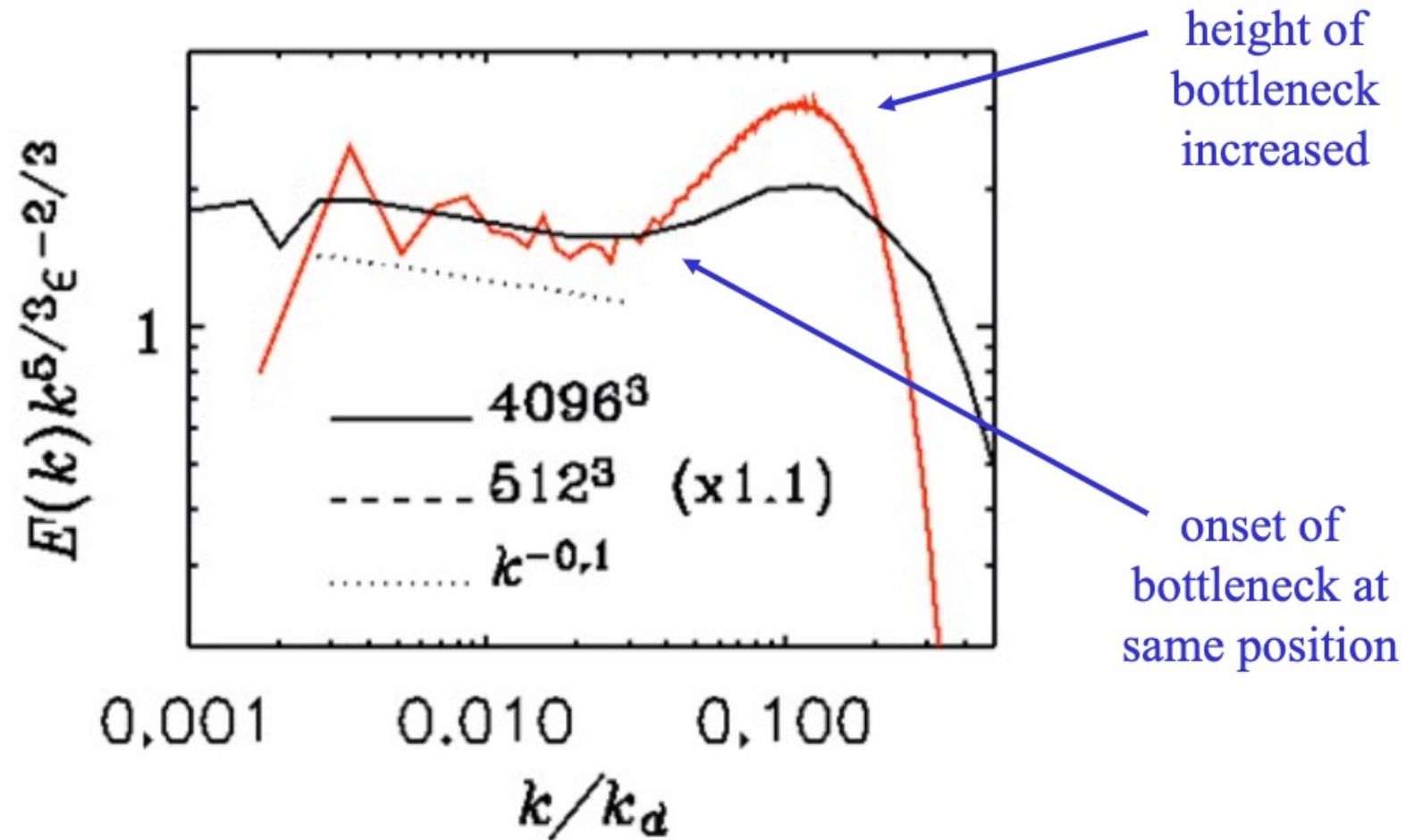
Laplacian dissipation	Hyper dissipation
$\mathcal{J} = -D \nabla \rho$ $\frac{\partial \rho}{\partial t} = D \nabla^2 \rho$ (n=1) $\propto k^2$	$\mathcal{J}^{(n)} = (-1)^n D^{(n)} \nabla^{2n-1} \rho.$ $\frac{\partial \rho}{\partial t} = D^{(3)} \nabla^6 \rho,$ (n=3) $\propto k^{2n}$

$$\text{Re}_{\text{grid}} = u_{\text{rms}} / (\nu_n k_{\text{Ny}}^{2n-1})$$

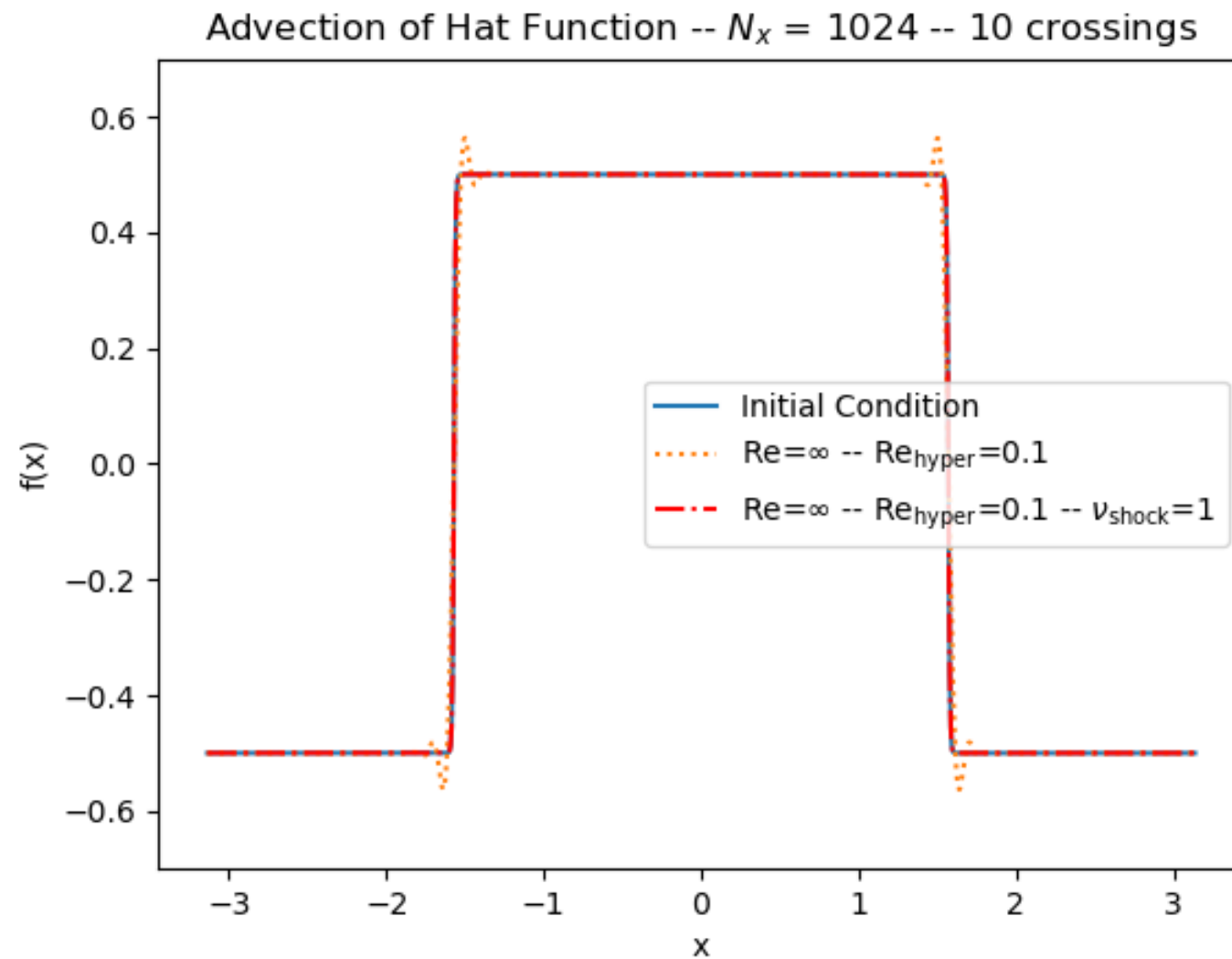




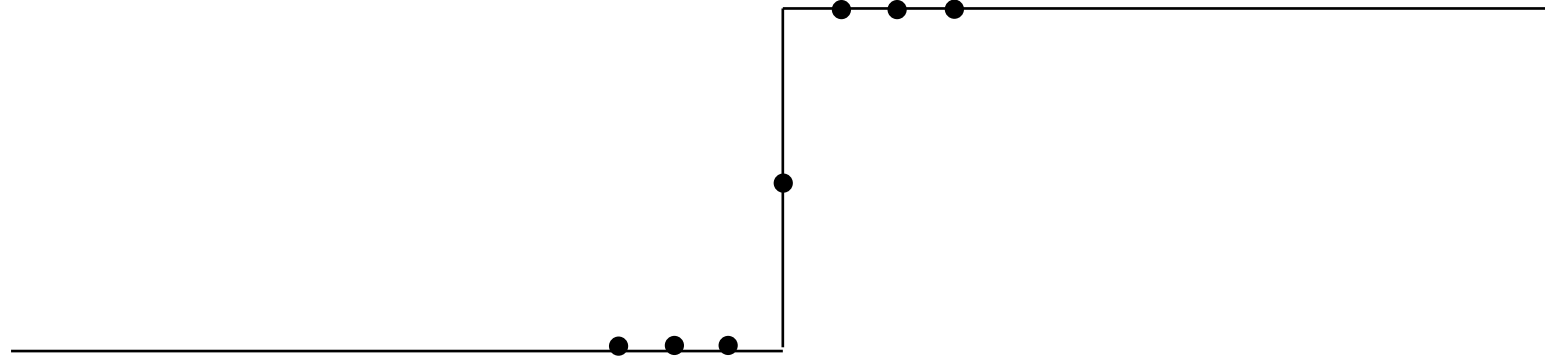
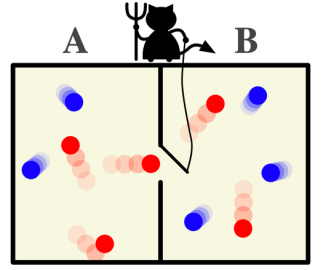
# Perils of hyperviscosity: bottleneck effect



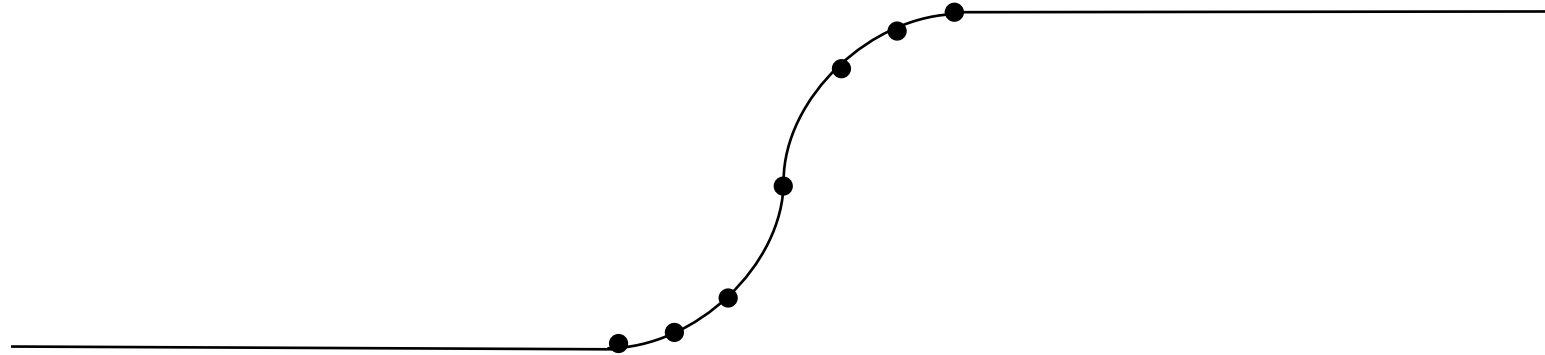
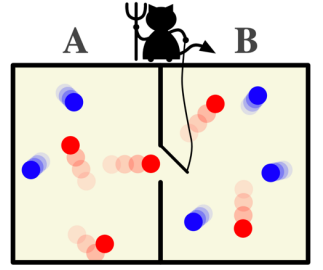
# Shock Viscosity



# Shock Viscosity

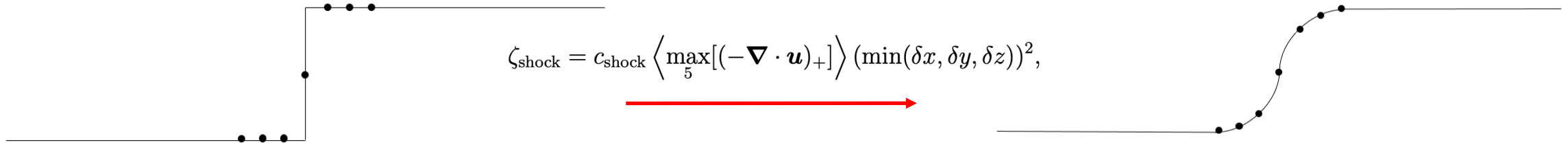


# Shock Viscosity



# Shock Viscosity

Shock viscosity smooths the discontinuity  
so that the stencil can resolve it.



Shock viscosity takes the form of a bulk viscosity  
(only relevant in regions of strong convergence)

$$\tau_{ij} = 2\rho\nu S_{ij} + \rho\zeta_{\text{shock}}\delta_{ij}\nabla \cdot \mathbf{u}.$$

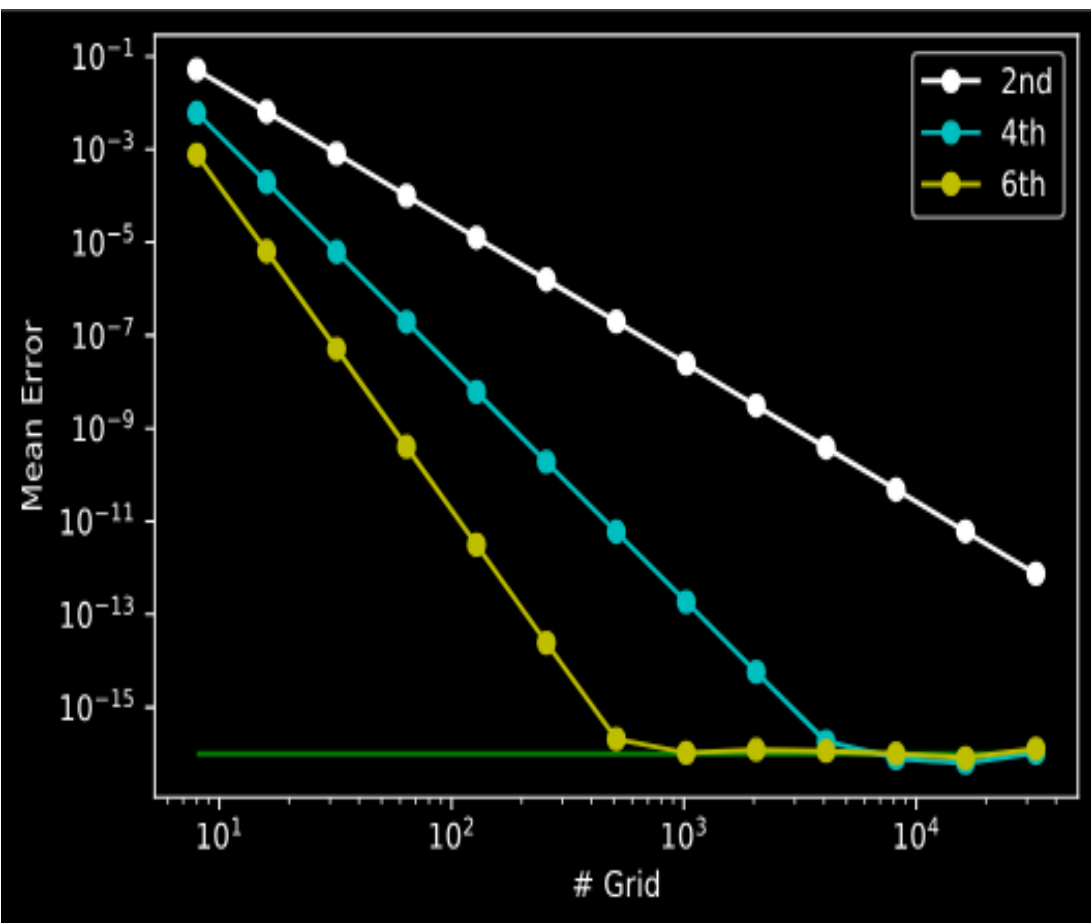
Viscous force

$$\rho^{-1}\mathbf{F}_{\text{visc}} = \nu \left( \nabla^2 \mathbf{u} + \frac{1}{3} \nabla \nabla \cdot \mathbf{u} + 2\mathbf{S} \cdot \nabla \ln \rho \right) + \zeta_{\text{shock}} [\nabla \nabla \cdot \mathbf{u} + (\nabla \ln \rho + \nabla \ln \zeta_{\text{shock}}) \nabla \cdot \mathbf{u}].$$

Viscous heating

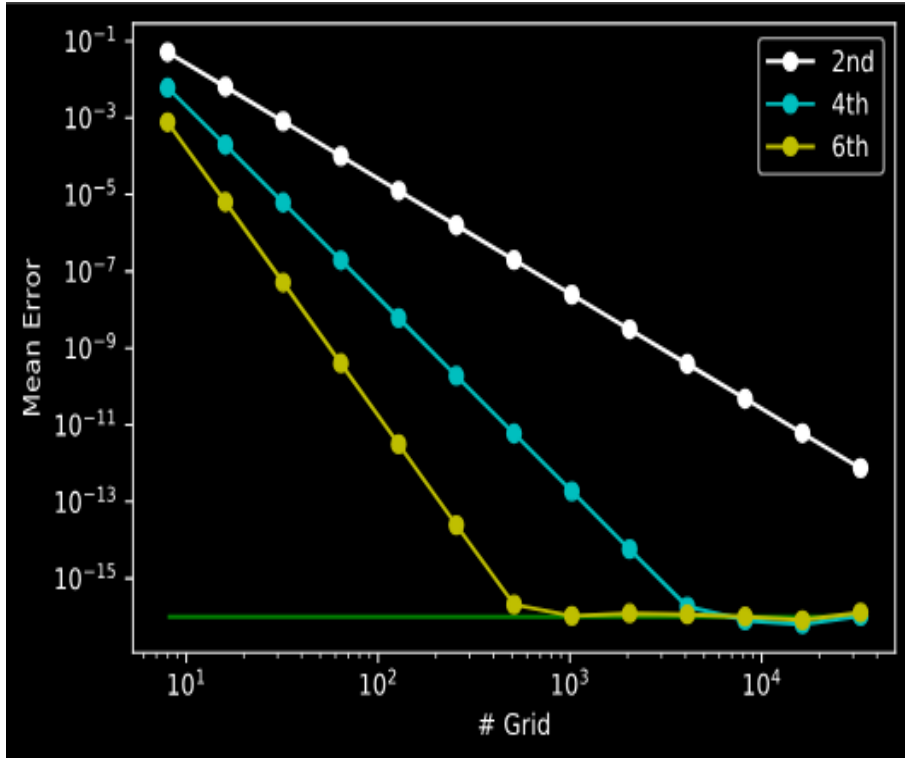
$$\rho^{-1}\Gamma_{\text{visc}} = 2\nu \mathbf{S}^2 + \zeta_{\text{shock}}(\nabla \cdot \mathbf{u})^2.$$

# Higher-order accuracy



- Higher order derivative = smaller truncation error
- Higher orders naturally require a larger stencil (more ghost zones)
- Higher order derivatives approach machine precision faster.
- **Alternative to spectral schemes**
- **Non-conservative methods become feasible because of the high accuracy.**

# Non-conservative method



- Conservation is not enforced, the method relies on accuracy of numerical derivative (but monitor the numerical solution).
- Possible to solve for
  - $\ln \rho$  (density never goes negative)
  - **vector potential  $\mathbf{A}$**  ( $\mathbf{B} = \nabla \times \mathbf{A}$ , automatically gives  $\nabla \cdot \mathbf{B} = 0$ )
  - **Entropy** (pdV work eliminated from equations)
  - Any quantity really (Pencil is essentially a general PDE solver)

$$\frac{D \ln \rho}{Dt} = -\nabla \cdot \mathbf{u}$$

$$\begin{aligned} \frac{D \mathbf{u}}{Dt} = & -c_s^2 \nabla \left( \frac{s}{c_p} + \ln \rho \right) - \nabla \Phi_{\text{grav}} + \frac{\mathbf{j} \times \mathbf{B}}{\rho} \\ & + \nu \left( \nabla^2 \mathbf{u} + \frac{1}{3} \nabla \nabla \cdot \mathbf{u} + 2 \mathbf{S} \cdot \nabla \ln \rho \right) + \zeta (\nabla \nabla \cdot \mathbf{u}); \end{aligned}$$

$$\rho T \frac{Ds}{Dt} = \mathcal{H} - \mathcal{C} + \nabla \cdot (K \nabla T) + \eta \mu_0 \mathbf{j}^2 + 2 \rho \nu \mathbf{S} \otimes \mathbf{S} + \zeta \rho (\nabla \cdot \mathbf{u})^2$$

$$\frac{\partial \mathbf{A}}{\partial t} = \mathbf{u} \times \mathbf{B} - \eta \mu_0 \mathbf{j}$$

$$D/Dt \equiv \partial/\partial t + \mathbf{u} \cdot \nabla$$

# Finite Differences - Summary

- Conceptually the **simplest** of the numerical methods and can be learned quite quickly
- Depending on the physical problem FD methods are **conditionally stable** (relation between time and space increment)
- High-order FD methods have difficulties concerning **damping at the grid scale**
- FD methods are usually **explicit** and therefore very easy to implement and efficient on **parallel computers**
- FD methods work best on regular, rectangular grids




# The Pencil Code

← → ↻ github.com/pencil-code

pencil-code

Type ↵ to search


🏠 Overview 📁 Repositories 4 📁 Projects 📦 Packages 👥 Teams 2 👤 People 108 ⚙️ Settings



## Pencil Code

Pinned

Customize pins

 **pencil-code** Public

A high-order finite-difference code for compressible hydrodynamic flows with magnetic fields and particles

Fortran ☆ 168 🔗 96

📁 Repositories

Find a repository...

Type ▾

Language ▾


Sort ▾

New

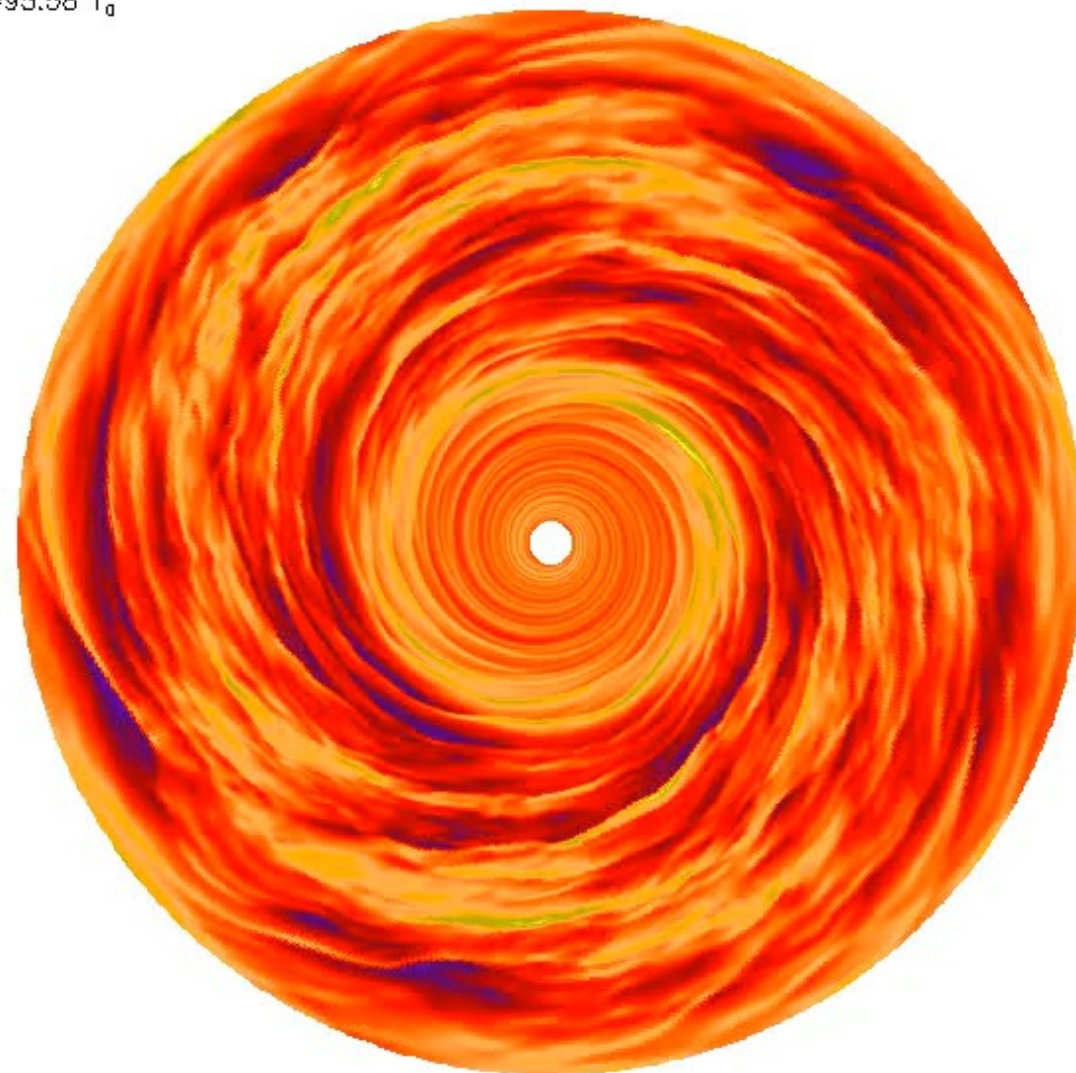
**pencil-code** Public

A high-order finite-difference code for compressible hydrodynamic flows with magnetic fields and particles

Fortran ☆ 168 🔗 96 🕒 5 🔗 3 Updated 2 hours ago



$t=95.58 T_0$



# History

- Started in Sept 2001 by Axel Brandenburg and Wolfgang Dobler
- Finite difference high order (default 6<sup>th</sup> in space, 3<sup>rd</sup> in time)
- Fully compressible code written in Fortran 90/2003
- Cache & memory efficient
- MPI parallelized
- Maintained/developed by ~50 people (Github) # of users > 200
- Live repository on GitHub
- Automatic daily validation of ~80 samples

# Capabilities

- Advanced particle module
- Advanced MHD module
- Selfgravity (FFT)
- Radiation (long characteristics)
- Cartesian, Cylindrical, Spherical geometries
- N-body
- Chemistry, combustion
- Embarrassingly parallel

# Pencil Code Philosophy

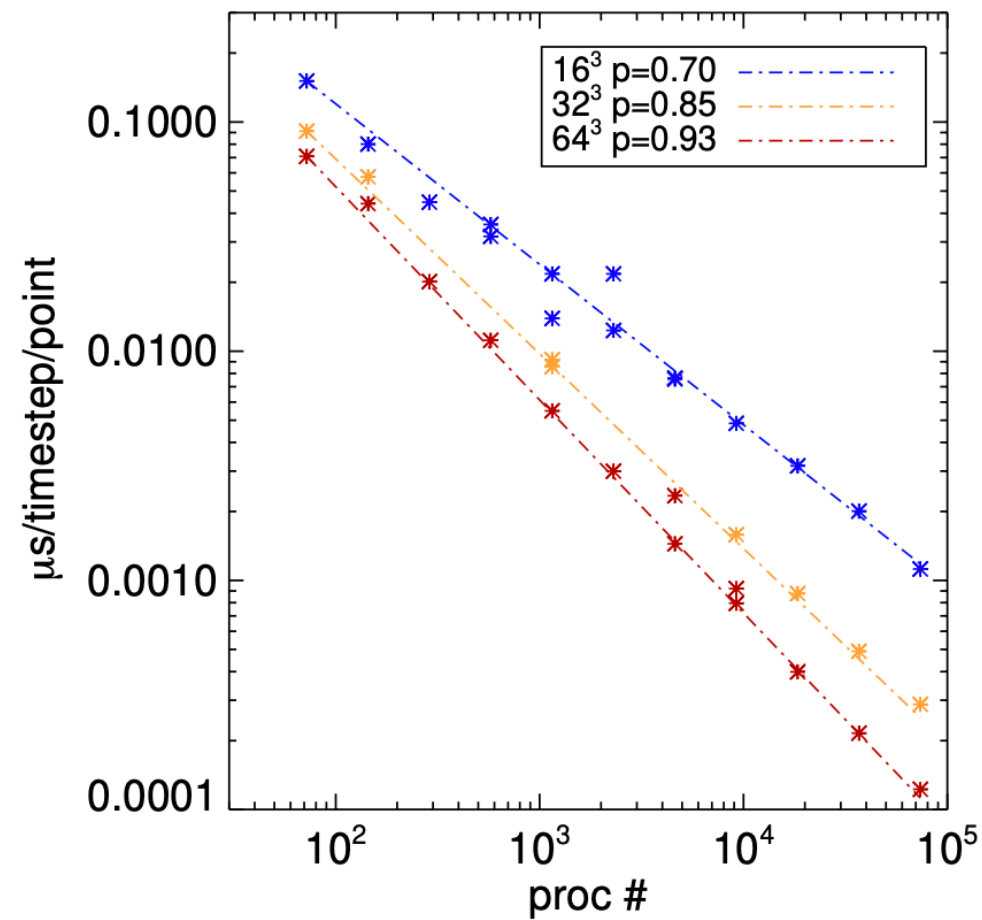
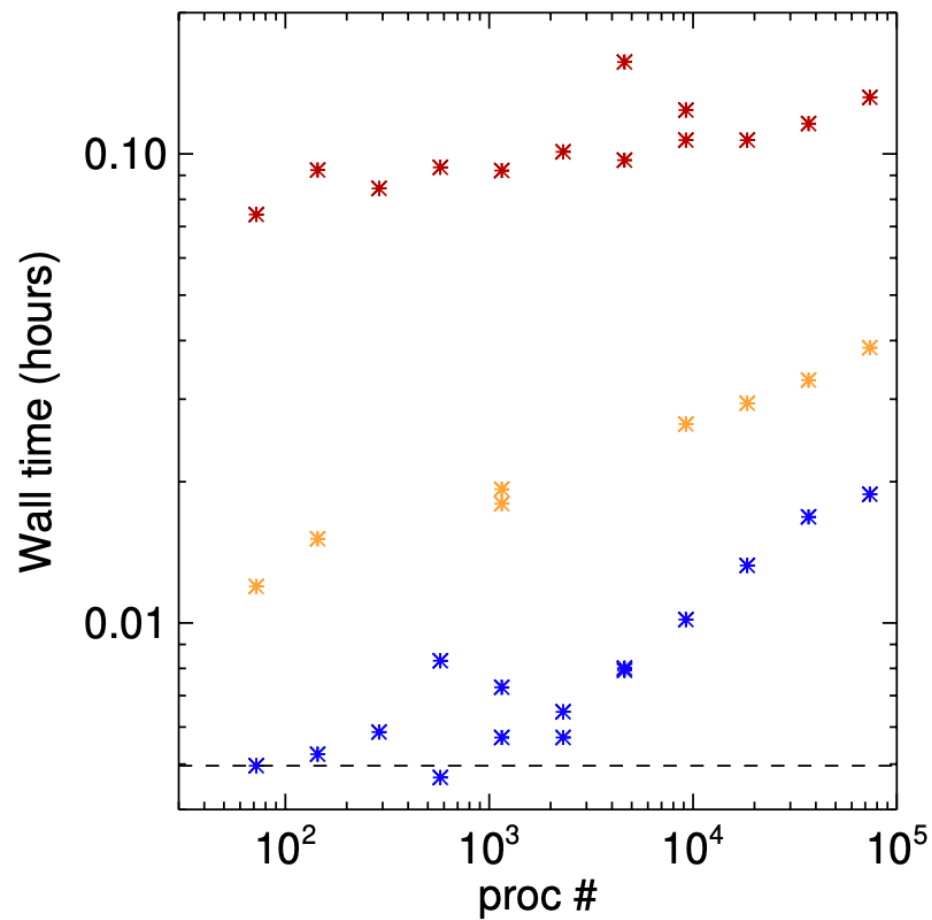
- Hands-off
  1. Code coverage is done in nightly auto-tests,
  2. which are also the only gatekeeping
- Research-driven – “Forever beta”
- Minimum interconnectivity
  1. Pencil is an engine and a toolbox
  2. The engine is clean and streamlined
  3. The toolbox is like a patchwork quilt (“it’s like walking into someone’s attic”)
- Only one version – Minimal duplication

# Versatility

- Interstellar medium
  - Galaxy clusters, Early Universe
- Planet formation
  - Inertial particles, planet-disk interaction
- Accretion disks
  - Shear flows
  - Dynamical instabilities,
- Solar Physics
  - Coronal heating, dynamos, spot formation
  - Convection, global convective dynamos
- Miscellanea
  - Test-field method, Hydro turb, turb combustion, covid pandemics evolution.

# Scaling

Weak Scaling



# Scaling

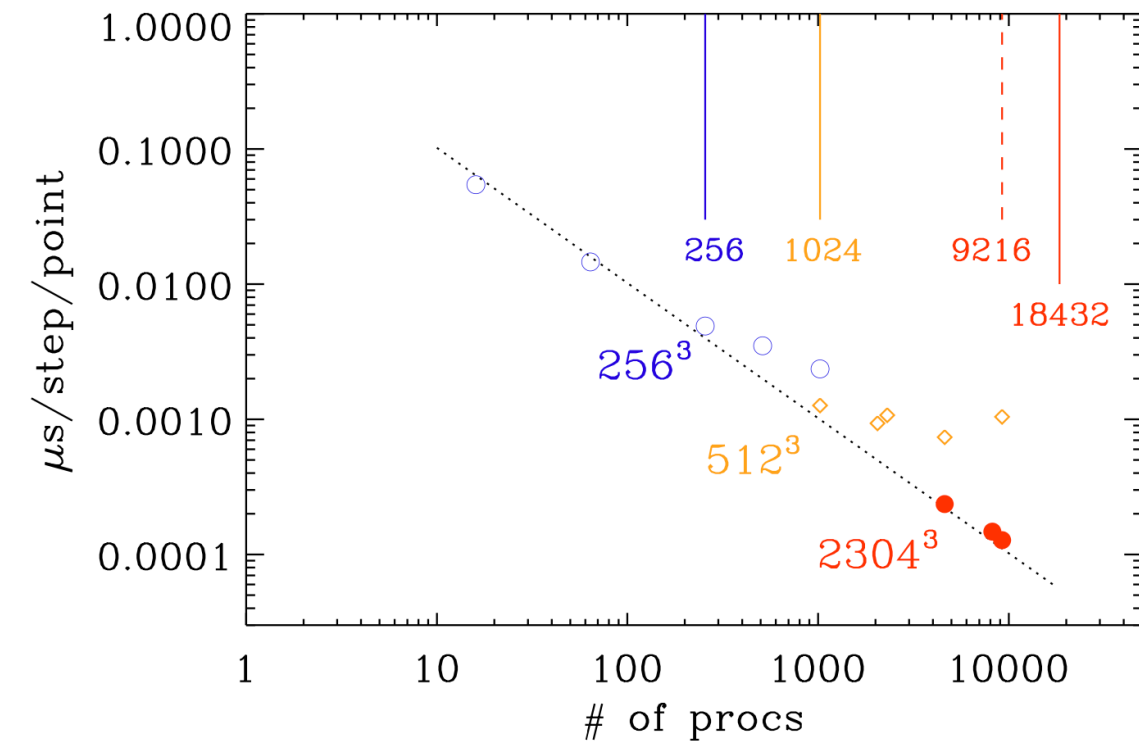


Figure 17: Strong scaling on Triolith (2014).

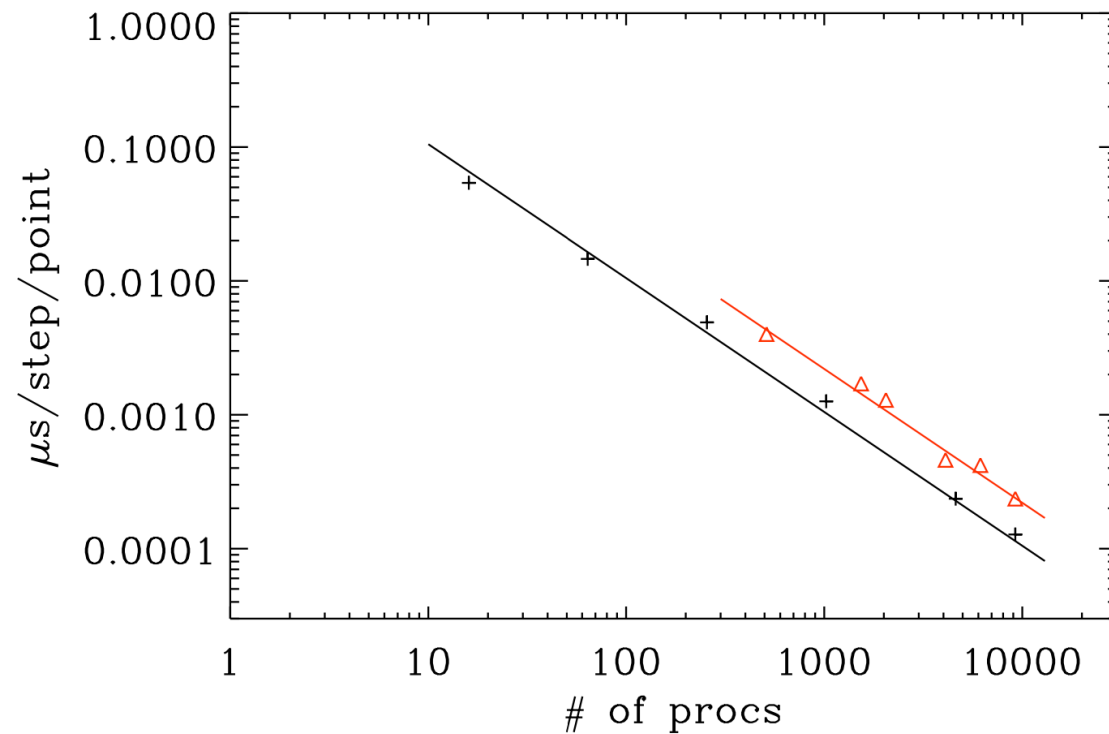
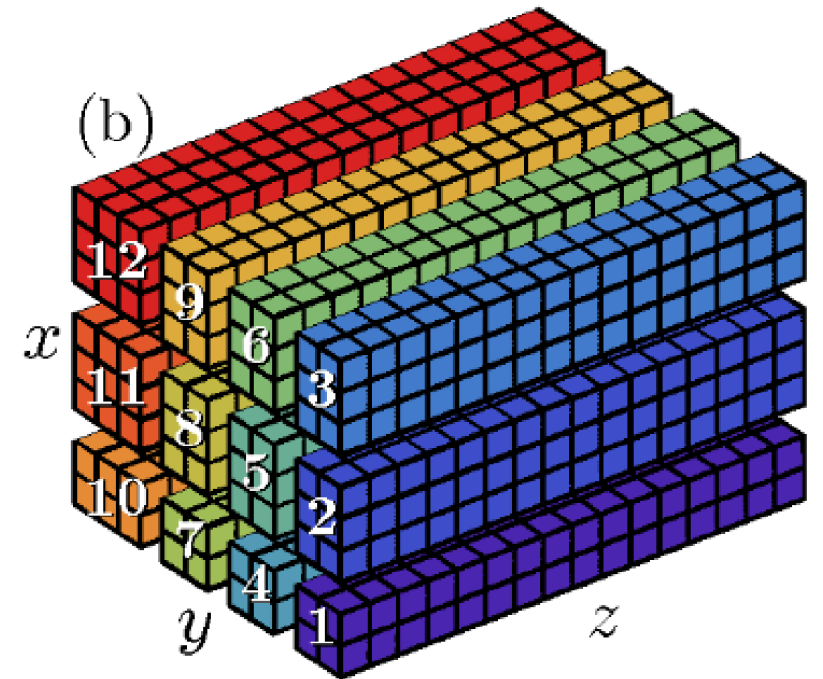


Figure 18: Comparison Triolith (black, plus signs) and Lindgren (red, triangles). Weak scaling (2014).

# Pencil formulation

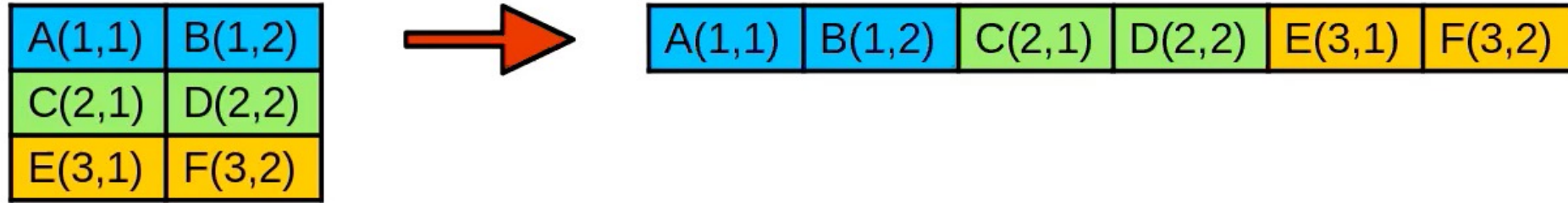
- In CRAY days: worked with full chunks  $f(nx,ny,nz,nvar)$ 
  - On modern CPUs, nearly 100% cache misses
- Instead work with  $f(nx,nvar)$ , i.e. one  $nx$ -pencil
- No cache misses, negligible work space, just  $2N$ 
  - Can keep all components of derivative tensors
- Communication before sub-timestep
- Then evaluate all derivatives, e.g. *call*  $curl(f,iA,B)$ 
  - Vector potential  $A=f(:, :, :, iAx:iAz)$ ,  $B=B(nx,3)$





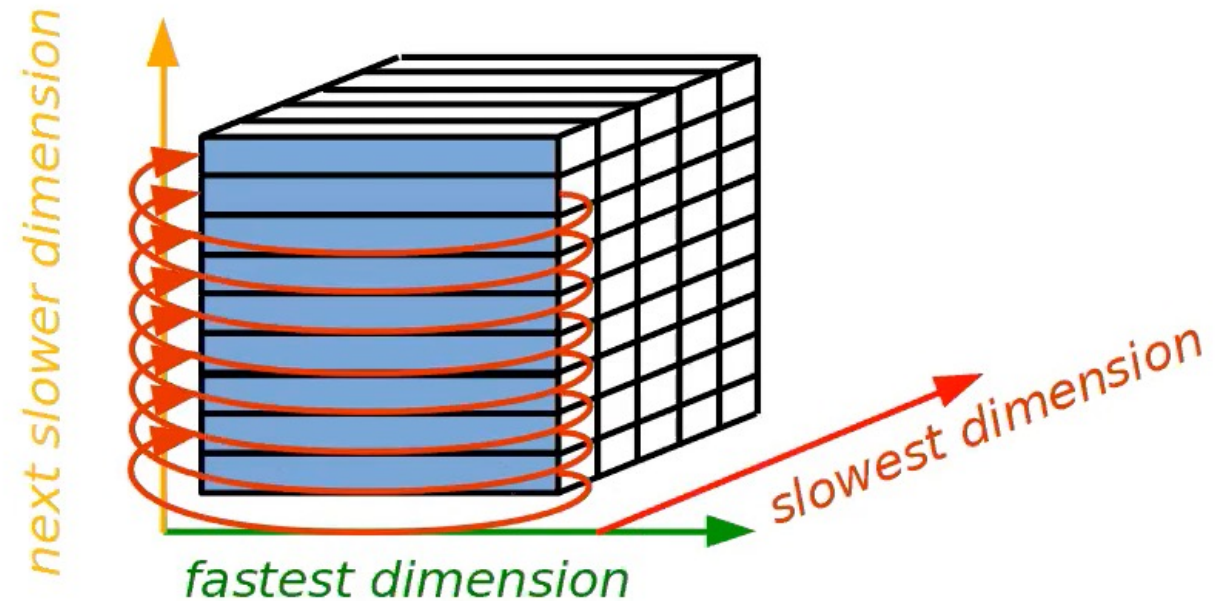
# Memory alignment and cache efficiency

## Alignment of data in memory:



## Cache efficiency:

=> both is achieved by  
elongated subdomains  
and optimal data access



## Unaligned access:

C / C++ / NumPy:

```
for iy = 1...3 {  
    for ix = 1...2 {  
        f[ix,iy] = ix+iy*2-2  
    }  
}
```

A(1,1)	B(2,1)
C(1,2)	D(2,2)
E(1,3)	F(2,3)

=> **“Row Major”**



A(1,1)	B(2,1)	C(1,2)	D(2,2)	E(1,3)	F(2,3)
1	2	3	4	5	6

Fortran / IDL / Julia / Matlab / R:

```
for ix = 1...2 {  
    for iy = 1...3 {  
        f[ix,iy] = ix+iy*2-2  
    }  
}
```

=> **“Column Major”**

## Aligned access:

C / C++ / NumPy:

```
for ix = 1...2 {  
    for iy = 1...3 {  
        f[ix,iy] = ix+iy*2-2  
    }  
}
```



A(1,1)	B(2,1)	C(1,2)	D(2,2)	E(1,3)	F(2,3)
1	2	3	4	5	6

Fortran / IDL / Julia / Matlab / R:

```
for iy = 1...3 {  
    for ix = 1...2 {  
        f[ix,iy] = ix+iy*2-2  
    }  
}
```

# Block domain decomposition

