

An Introduction to PALM: Numerics and boundary conditions

Heiko Jansen

Zingst, July 2005

Overview

- ① Introduction
 - ▣ Finite differences
- ② Temporal discretization
 - ▣ Time-step-schemes
- ③ Spatial discretization
 - ▣ Advection-schemes
- ④ Boundary conditions
- ⑤ Ensuring incompressibility

Introduction

Starting point: The Navier-Stokes equation

Momentum equations:

$$\frac{\partial \bar{u}_i}{\partial t} = -\frac{\partial(\bar{u}_j \bar{u}_i)}{\partial x_j} - \frac{1}{\rho_0} \frac{\partial \bar{p}^*}{\partial x_i} - (\varepsilon_{ijk} f_j \bar{u}_k - \varepsilon_{i3k} f_3 u_{kgeo}) + \frac{g}{\theta_{v00}} (\bar{\theta}_v - \theta_{v00}) \delta_{i3} - \frac{\partial \tau_{ij}}{\partial x_j}$$

Liquid water potential temperature:

$$\frac{\partial \bar{\theta}_l}{\partial t} = -\frac{\partial(\bar{u}_i \bar{\theta}_l)}{\partial x_i} - \frac{\partial W_i}{\partial x_i} + S_{\text{rad}} + S_{\text{prec}}$$

Total water content:

$$\frac{\partial \bar{q}}{\partial t} = -\frac{\partial(\bar{u}_i \bar{q})}{\partial x_i} - \frac{\partial H_i}{\partial x_i} + S_{\text{prec}}$$

The Navier-Stokes equations form a set of coupled, non-linear partial differential equations

Equation of continuity:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0$$

Objective: to find a solution (numerically) consistent with given initial and boundary conditions

Problem: the equations cannot be solved analytically

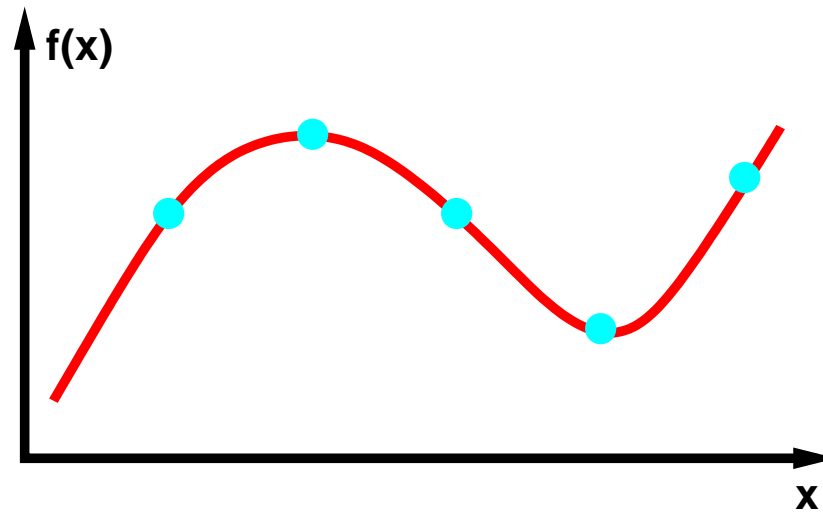
In the following we will consider a general form of the prognostic equations: $\frac{d\psi}{dt} = F$

(ψ any variable, F forces affecting ψ)

Introduction (I)

- Aim: Design methods to convert the original differential equations into a set of solvable algebraic equations
- As a part of this task continuous functions must be represented by finite set of numbers
- One basic strategy ➤ **the grid-point method**
 - Transformation of the spatial and temporal differential equations to difference equations

Continuous functions ➤ functions defined at discrete points



- In PALM the finite difference method is used

Introduction – Finite Differences

- Finite differences are based on truncated Taylor series:

$$\psi(x \pm \Delta x) = \psi(x) \pm \Delta x \frac{\partial \psi(x)}{\partial x} + \frac{(\Delta x)^2}{2} \frac{\partial^2 \psi(x)}{\partial x^2} \pm \dots \frac{(\Delta x)^n}{n!} \frac{\partial^n \psi}{\partial x^n} \dots$$

- Example:** Finite-difference representation of derivatives:

$$\left(\frac{\partial \psi}{\partial x} \right) \approx \frac{\psi(x + \Delta x) - \psi(x)}{\Delta x} + \mathcal{O}(\Delta x) \quad \text{forward differences}$$

$$\left(\frac{\partial \psi}{\partial x} \right) \approx \frac{\psi(x) - \psi(x - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x) \quad \text{backward differences}$$

$$\left(\frac{\partial \psi}{\partial x} \right) \approx \frac{\psi(x + \Delta x) - \psi(x - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x^2) \quad \text{central differences}$$

$$\left(\frac{\partial^2 \psi}{\partial x^2} \right) \approx \frac{\psi(x + \Delta x) - 2\psi(x) + \psi(x - \Delta x)}{\Delta x^2} + \mathcal{O}(\Delta x^2) \quad \text{2nd derivatives}$$

Discretization of the time-axis

Objective:

- Prediction of the temporal evolution of ψ :

$$\frac{\partial \psi(t)}{\partial t} = F(t)$$

- For discretization of the above equation the time-axis is divided into (equidistant) segments with spacings Δt :

$$t - 2\Delta t, t - \Delta t, t, t + \Delta t, t + 2\Delta t$$

- Based on the truncated Taylor series different time-step schemes can be defined

$$\psi(t \pm \Delta t) = \psi(t) \pm \Delta t \frac{\partial \psi(t)}{\partial t} + \frac{(\Delta t)^2}{2} \frac{\partial^2 \psi(t)}{\partial t^2} \pm \dots \frac{(\Delta t)^n}{n!} \frac{\partial^n \psi}{\partial t^n} \dots$$

Time-step schemes in PALM (I)

Euler-scheme:

- Truncate the Taylor series at term of the first order
- Forward differences:

$$\psi(t + \Delta t) = \psi(t) + \Delta t \cdot F(t)$$

- Entirely unstable and diffusive

$$\frac{\partial \psi}{\partial t} \approx \frac{\psi(t + \Delta t) - \psi(t)}{\Delta t} + \mathcal{O}(\Delta t)$$

Leap-frog scheme:

- Scheme of second order accuracy
- Central differences:

$$\psi(t + \Delta t) = \psi(t - \Delta t) + 2\Delta t \cdot F(t)$$

- Unstable for large Δt
($\Delta t \leq 0.1 \cdot \Delta t_{\text{CFL}}$ for stability)
- "Time-splitting" requires a weak time filter
(Asselin Filter)

$$\frac{\partial \psi}{\partial t} \approx \frac{\psi(t + \Delta t) - \psi(t - \Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2)$$

Sidestep: The CFL Criterion

⇒ Courant, Friedrich, Lewy (1928).

⇒ For stability, $c \cdot \frac{\Delta t}{\Delta x} < 1$ must hold, where c is the velocity of the solution.

⇒ Thus, $\Delta t < \Delta t_{\text{CFL}} = \frac{\Delta x}{c}$.

⇒ Consequence for grid refinement: $(\Delta x)^* = \frac{1}{2}\Delta x \Rightarrow (\Delta t_{\text{CFL}})^* = \frac{1}{2}\Delta t_{\text{CFL}}$.
Halving the meshsize means 4-fold computation time!

⇒ In PALM, $\Delta t_{\text{CFL}} = \min \left(\frac{\Delta x}{u_{\text{max}}}, \frac{\Delta y}{v_{\text{max}}}, \frac{\Delta z}{w_{\text{max}}} \right)$.

⇒ Here, halving the horizontal meshsize means 8-fold computation time!

Time-step schemes in PALM (II)

Runge-Kutta time-step scheme

$$k_1 = F(\psi_n),$$

$$k_2 = F\left(\psi_n + \frac{1}{3} \Delta t k_1\right),$$

$$k_3 = F\left(\psi_n - \frac{3}{16} \Delta t k_1 + \frac{15}{16} \Delta t k_2\right),$$

$$\psi_{n+1} = \psi_n + \frac{1}{30} \Delta t (5k_1 + 9k_2 + 16k_3)$$

Advantages:

- Scheme of second order accuracy
- no time-splitting appears
- Large time-steps are possible ($\Delta t \leq 0.9 \cdot \Delta t_{\text{CFL}}$)

Disadvantages:

- Requires more function evaluations than the leapfrog scheme

Time-step schemes in PALM (III)

- ▶ In the PALM-code the time-step schemes are written in compact notation

$$u_p(k,j,i) = (1 - c1) * u_m(k,j,i) + c1 * u(k,j,i) + dt * (c2 * tend(k,j,i) + c3 * tend_m(k,j,i))$$

- ▶ The switches $c1 - c3$ distinguish between the time-step schemes:

$c1=c2=1, c3=0$: Euler-scheme

$$u_p(k,j,i) = u(k,j,i) + dt * tend(k,j,i)$$

$c1=0, c2=2, c3=0$: Leap-frog scheme

$$u_p(k,j,i) = u_m(k,j,i) + 2 * dt * tend(k,j,i)$$

Time-step schemes in PALM (III)

c1=1, c2=1/3, c3=0 : Runge-Kutta scheme, first part

$$u_p(k,j,i) = u(k,j,i) + 1/3 * dt * tend(k,j,i)$$

c1=1, c2=15/16, c3=-25/48 : Runge-Kutta scheme, second part

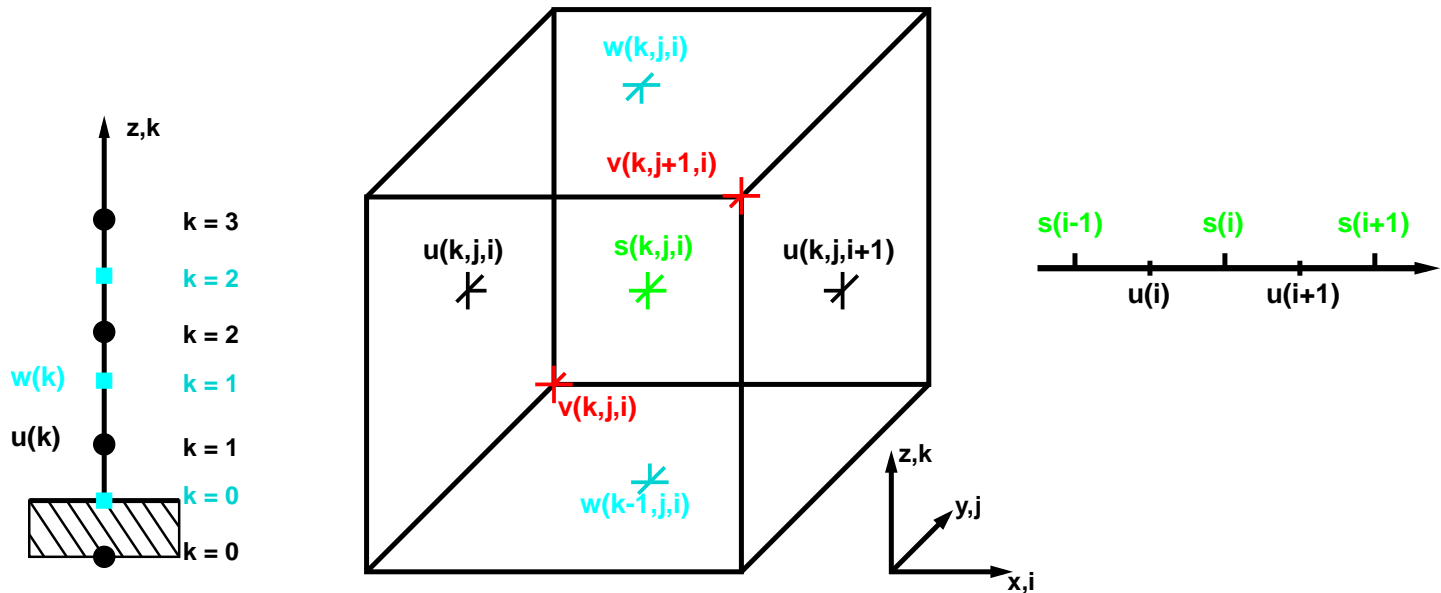
$$u_p(k,j,i) = u(k,j,i) + dt (15/16 * tend(k,j,i) - 25/48 * tend_m(k,j,i))$$

c1=1, c2=8/15, c3=1/15 : Runge-Kutta scheme, third part

$$u_p(k,j,i) = u(k,j,i) + dt (8/15 * tend(k,j,i) + 1/15 * tend_m(k,j,i))$$

NAMELIST-parameter	parameter kind	possible values
timestep_scheme	&INIPAR	'euler' 'leapfrog' 'runge-kutta-2' 'runge-kutta-3' (default)

Spatial discretization – The numerical grid



- ▣ The equations are spatially discretized on an Arakawa-C grid
 - ▣ Regular staggered grid
- ▣ Scalar variables (e.g. θ , p^* , e , K_m , K_h) are defined in the cell centers
- ▣ Velocity components (u , v , w) are shifted by a half of the grid spacing
- ▣ Spacings are equidistant

Spatial discretization - advection terms (1)

- The investigation will be focused on the discretization of the advection term:

$$\frac{\partial \psi}{\partial t} = -u_k \frac{\partial \psi}{\partial x_k} + S$$

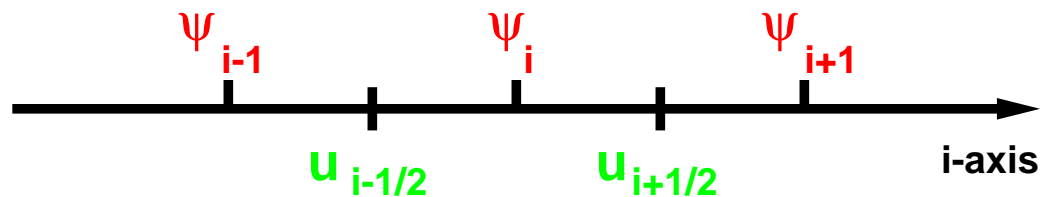
(respectively: $-\frac{\partial(u_k \psi)}{\partial x_k}$ in case of an incompressible flow)

- Advection schemes used in PALM:
 - Scheme after Piacsek-Williams
 - Upstream-scheme
 - Upstream-spline-scheme
 - Bott-Chlond-Scheme (scalar variables only)

NAMELIST-parameter	parameter kind	possible values
scalar_advec	&INIPAR	'pw-scheme' (default) 'bc-scheme' 'ups-scheme'
momentum_advec	&INIPAR	'pw-scheme' (default) 'ubs-scheme'

Advection scheme of Piacsek und Williams

- The default advection scheme in PALM
- Scheme C3 after Piacsek and Williams (1970, J. Comput. Phys., 6, 392)
- Scheme of 2nd order accuracy
- Conservation of integrals of linear and quadratic quantities
- Requires incompressibility ➤ flux form of advection term
- Low computational costs
- Used in combination with the Runge-Kutta time-step-scheme



$$\left. \frac{\partial}{\partial x} (u \psi) \right|_i = \frac{1}{2\Delta x} \left(u_{i+\frac{1}{2}} \psi_{i+1} - u_{i-\frac{1}{2}} \psi_{i-1} \right)$$

Annotation: In case of velocity advection (e.g., $\psi = u$), $u_{i+\frac{1}{2}}$ and $u_{i-\frac{1}{2}}$ have to be obtained by linear interpolation

Advection scheme of Piacsek and Williams – Example

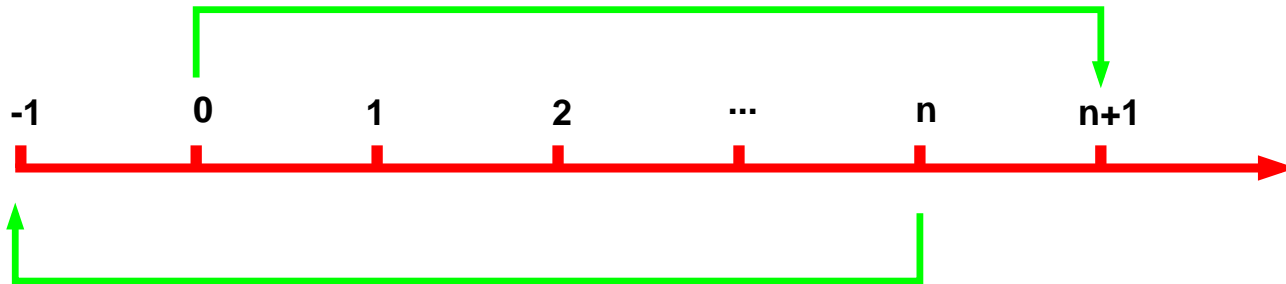
Example: A piece of code taken from `advec_u_pw.f90`

```
gu = 2.0 * u_gtrans      ! Galilei-transformation
gv = 2.0 * v_gtrans      ! Galilei-transformation
DO  i = nxl, nxr
  DO  j = nys, nyn
    DO  k = nzb+1, nzt
      tend(k,j,i) = tend(k,j,i) - 0.25 * (                                &
        ( u(k,j,i+1) *( u(k,j,i+1) +u(k,j,i)      -gu )                &
          - u(k,j,i-1) *( u(k,j,i)  +u(k,j,i-1)    -gu ) ) * ddx      &
        + ( u(k,j+1,i) *( v(k,j+1,i) +v(k,j+1,i-1) -gv )                &
          - u(k,j-1,i) *( v(k,j,i)   +v(k,j,i-1)   -gv ) ) * ddy      &
        + ( u(k+1,j,i) *( w(k,j,i)   +w(k,j,i-1) )                      &
          - u(k-1,j,i) *( w(k-1,j,i) +w(k-1,j,i-1) ) ) * ddzw(k) )
    ENDDO
  ENDDO
ENDDO
```

Boundary conditions (I)

The boundary conditions are applied to all variables after each time-step.

▣ Lateral boundary conditions are cyclic by default.



$$\psi(-1) = \psi(n)$$

$$\psi(n+1) = \psi(0)$$

Boundary conditions (II)

► non-cyclic boundary can be chosen by the namelist parameters `bc_lr` and `bc_ns`

NAMELIST-parameter	possible values	meaning
<code>bc_lr</code>	'cyclic' (default)	cyclic boundaries
	'dirichlet/neumann'	inflow from left, outflow to the right
	'neumann/dirichlet'	inflow from right, outflow to the left
<code>bc_ns</code>	'cyclic' (default)	cyclic boundaries
	'dirichlet/neumann'	inflow from north, outflow to the south
	'neumann/dirichlet'	inflow from south, outflow to the north

Warning: When non-cyclic boundary conditions are desired, one has to adjust other model parameters as well! Refer to the model documentation in this case.

Boundary conditions (III)

Bottom boundary condition:

Describes a physical/real boundary (impermeable wall)

u, v, w : Dirichlet-conditions : $\psi = 0$.

due to the staggered grid : $w(0) = 0, \quad u(0) = -u(1), \quad v(0) = -v(1)$

e, p^*, K_m, K_h : Neumann-conditions : $\partial\psi/\partial z = 0$, i.e. $\psi(0) = \psi(1)$.

θ, q :

Prescribed fluxes: Neumann-conditions

Monin-Obukhov theory: Dirichlet-conditions

Boundary conditions (IV)

▣▣▣ Top boundary condition:

▣▣▣ u, v, w : Dirichlet-conditions : $u(z_{\text{top}}) = u_g, v(z_{\text{top}}) = v_g, w(z_{\text{top}}) = 0$.

▣▣▣ e : Neumann-conditions : $\partial e / \partial z = 0$, i.e. $e(z_{\text{top}}) = e(z_{\text{top}} - 1)$.

▣▣▣ θ, q : temporal constant gradients

‘Neumann-conditions’ : $\partial \psi / \partial z = \partial \psi / \partial z|_0$.

Ensuring incompressibility (I)

- Governing equations of PALM require incompressibility
- Incompressibility is reached by a predictor-corrector method
 - ① Prediction of a preliminary momentum field by taking into account the pressure from the previous time step

$$\tilde{u}_i^{t+\Delta t} = u_i^{t-\Delta t} + 2\Delta t \cdot (\dots) + \frac{\Delta t}{\rho_0} \frac{\partial p^{*t}}{\partial x_i}$$

- ② The final momentum field has to guarantee mass conservation (fulfilling the equation of continuity):

$$\frac{\partial u_i^{t+\Delta t}}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\tilde{u}_i^{t+\Delta t} - \frac{\Delta t}{\rho_0} \frac{\partial p^{*t}}{\partial x_i} \right) \stackrel{!}{=} 0$$

- ③ Resulting in a Poisson-equation for the perturbation pressure

$$\frac{\partial^2 p^*}{\partial x_i^2} = \frac{\rho_0}{\Delta t} \frac{\partial \tilde{u}_i^{t+\Delta t}}{\partial x_i}$$

Ensuring incompressibility (II)

⇒ The Poisson equation forms a set of linear equations:

$$\frac{\partial^2 p^*}{\partial x_i^2} = f \quad \Rightarrow \quad \mathbf{A} \cdot \vec{p}^* = \vec{f}$$

with $\vec{p}^*, \vec{f} \in R^n$, $\mathbf{A} = D - L - R \in R^{n \times n}$

⇒ Two general possibilities are provided by PALM to solve the Poisson equation

① Direct solving by means of a Fast-Fourier-transformation (FFT)

② Iterative solving

(a) Successive over-relaxation-solver (SOR-solver)

(b) Multi-grid scheme

NAMELIST-parameter	parameter kind	possible values
psolver	&INIPAR	'poisfft' (default) 'multigrid' 'sor'

Ensuring incompressibility (III) – FFT-solver

Procedure:

- ① Discretization of the Poisson-equation by central differences
- ② 2D discrete FFT in both horizontal directions
- ③ Solving the resulting tridiagonal set of linear equations
- ④ Inverse 2D discrete FFT in both horizontal directions leading to the perturbation pressure
- ⑤ Correction of the preliminary velocity field, e.g. $u_i^{t+\Delta t} = \tilde{u}_i^{t+\Delta t} - \Delta t \frac{p_i^* - p_{i-1}^*}{\Delta x}$

Annotations:

- ▣ Highly effective solver
- ▣ Due to non-locality of the FFT, transpositions are required on distributed machines
- ▣ The use is linked to periodic boundary conditions and uniform grids

Ensuring incompressibility (III) – iterative solvers

Basic idea of iterative solvers:

Poisson equation is transformed to a fixed point problem: $\vec{p}^{k+1} = \mathbf{T} \cdot \vec{p}^k + \vec{c}^k$

Starting from a first guess the solution will be improved by repeated execution of the fixed point problem:

$$\vec{p}^1 = T \cdot \vec{p}^0 + \vec{c}^0$$
$$\vec{p}^2 = T \cdot \vec{p}^1 + \vec{c}^1$$

⋮

$$\vec{p}^k = T \cdot \vec{p}^{k-1} + \vec{c}^{k-1}$$
$$\vec{p}^{k+1} = T \cdot \vec{p}^k + \vec{c}^k$$

Implementation:

Depending of the structure of the matrix T and vector \vec{c} different iterative solvers can be defined, e.g.: Jacobi-scheme (2D-uniform grid):

$$p_{i,j}^{k+1} = \frac{1}{4} \cdot (p_{i-1,j}^k + p_{i+1,j}^k + p_{i,j-1}^k + p_{i,j+1}^k - \Delta x^2 f(i, j, k))$$

- With each iteration step k the improved solution converges towards the exact solution, the residual is a measure for the error of intermediate solution: $\vec{r}^k = \vec{c}^k - T \cdot \vec{p}^k$
- Iterative scheme are 'local schemes' \implies exchange of information takes places only between neighboring grid-points
- The convergence depends on the number of grid-points n
 $\mu = 1 - \mathcal{O}(1/n)$ ($0 < \mu < 1$)
the larger n and thus μ , the slower the residual will be reduced by an iteration-step.

- With regard to the number of grid-points typically used in PALM the convergence of all iterative solvers is uneconomical! (SOR-solver are be used for tests only)

Ensuring incompressibility (IV) – Multi-grid-method

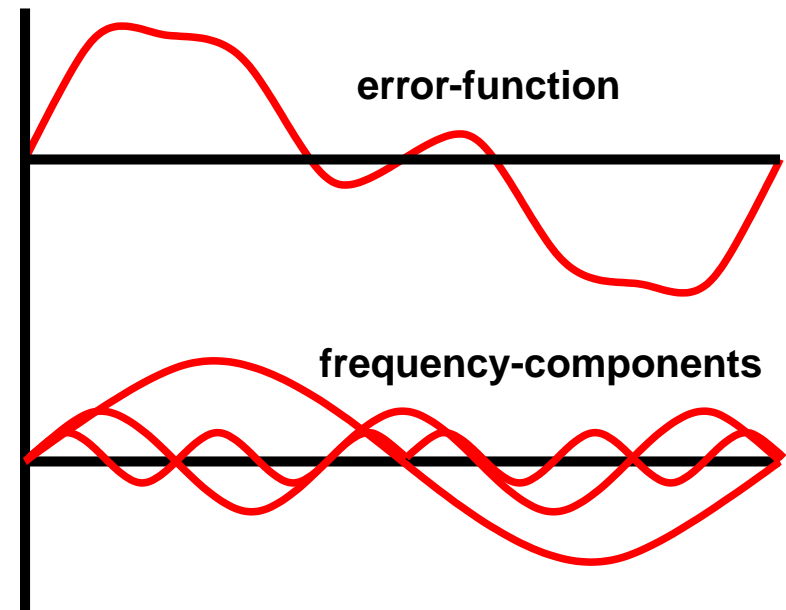
Motivation:

- Standard iteration methods have (very) slow speed of convergence
- Iterative methods show a frequency-dependent reduction of the residual ➤ low frequencies are reduced slower than high-frequencies (due to locality)
- FFT method does not work with non-cyclic boundaries

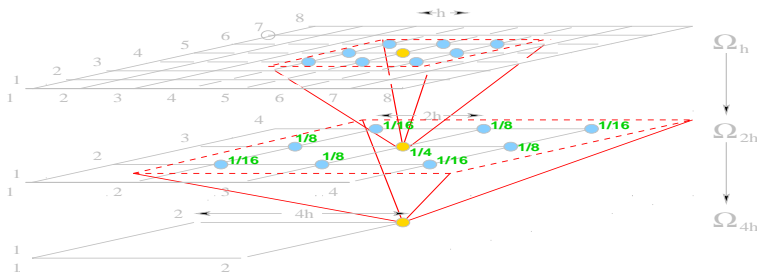
Idea:

Reducing the error frequencies on grids with different grid-spacing

- Errors of low frequency are reduced on coarse grids
- Errors of high frequency are reduced on fine grids



Ensuring incompressibility (V) – Multi-grid-method



- On each grid-level an approximate solution of the fixed point equation is obtained performing a few iterations
- The solution is transmitted to the next coarser grid-level where it is used as the first guess to solve the fixed point problem
- This procedure is performed up to the coarsest grid-level containing two grid-points in each direction
- From the coarsest grid-level the procedure is passed in backward order to get the final solution