COSMO-ICON Physics

Status

- Already implemented:
  - Microphysics: not really the same code as in ICON, because ICON uses code with vector optimizations (but the same as regards contents)
  - Radiation: the version that does not use the coarser radiation grid is even running on GPUs!
- On a good way:
  - Turbulence: code still needs some clean-up
  - IFS Tiedtke-Bechtold scheme: implemented by Jochen, tested by Lucio
- Next on the list
  - all the other parameterization from COSMO: SSO, Tiedtke and shallow convection, TERRA, seaice, FLAKE

The Blocked Data Structure

- Memory layout and data structure:
  - all parameterizations now implement a two-dimensional data structure:

    `(number of grid points=nproma, vertical dimension=ke)`
  - Small `nproma` works well on cache-based architectures, long `nproma` is good for vectorization
- All necessary fields in the blocked data structure have to be defined in the module `data_block_fields.f90`

Copy in / Copy out Infrastructure

- For the COSMO-Model data has to be copied to / from the blocked data structure before / after the physics
- This is handled by the „copy-to-block infrastructure“, which consists of 2 modules
  - `src_block_fields.f90`: contains methods to register fields and do the copy and also the correspondence table
  - `src_block_fields_org.f90`: contains block fields allocation / deallocation and organization methods

Correspondance Table and Copy Lists

- The correspondence table contains pointers to fields in the ijk- and the blocked data structure + additional meta data
  - It is built with the method `register_block_field`
  - all CALLs to `register_block_field` are in the subroutine `block_fields_register_all` in module src_block_fields_org.f90
    - `CALL register_block_field („hhl“, hhl, hhl_b)`
    - `CALL register_block_field („t“, t, t_b, nnow)`
- The copy lists: Every package has to create a copy list:
  - `CALL init_copy_list (turCopyList)`
  - for every field a `register_copy` has to be added
  - `CALL register_copy(hhl_b,turCopyList, copyToBlockF)`
  - `CALL register_copy(gz0_b,turCopyList, copyFromBlockF)`

Doing the Copying

- The copying for a special list is requested by the method
  - `IF (ltur) CALL request_copy (turCopyList,ierror,yerrmsg)`
  - it has to be called within the block loop and indicates that a parameterization is executed within this time step
- Finally the copy to / from the block structure is executed by the methods
  - `CALL copy_to_block (turCopyList,ierror,yerrmsg)`
  - `CALL copy_from_block (turCopyList,ipend,ib,ierror,yerrmsg)`
- To verify that all requested copies for this time step have been executed, the method finalize_copy has to be called after the block loop:
  - `CALL finalize_copy (ierror, yerrmsg)`

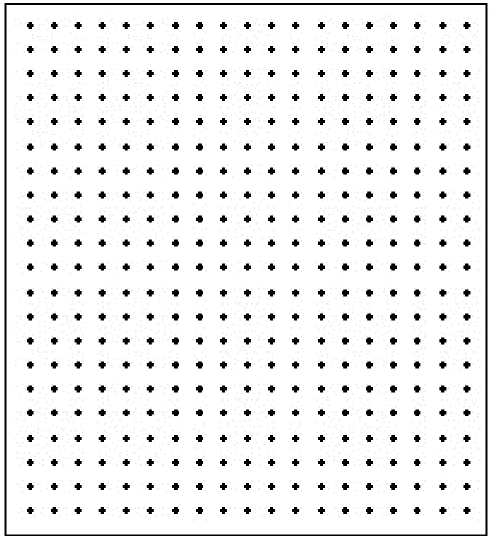
Allocation of Local Memory

- In principle we use local memory within a subroutine by the means of automatic arrays.
- But memory allocation is very expensive on GPUs, therefore the use of automatic arrays shall be avoided in the physical packages.
- Solution: All automatic arrays are replaced by allocatable arrays and routines are provided to allocate / deallocate them.
  - When running on GPUs, the allocation-routines are called at the beginning of the program.
  - When running on CPUs, they are called just before the physical package.

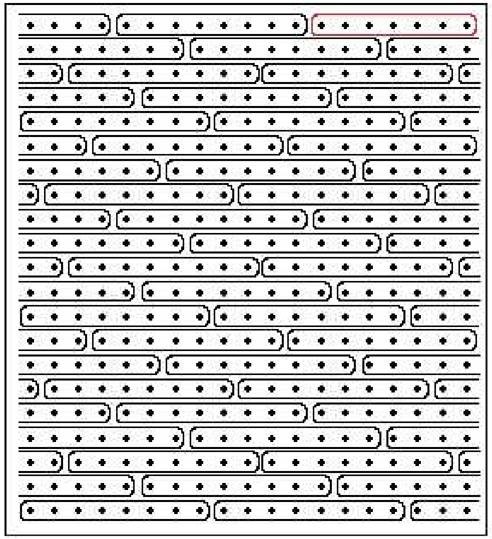
The Coarse Radiation Grid

- ➔ To save computational time, the COSMO-Model offers the possibility to run the radiation only on a coarse grid. How does this fit in the blocked data structure?
- ➔ Illustration of the blocked data structure for `nproma=8`:

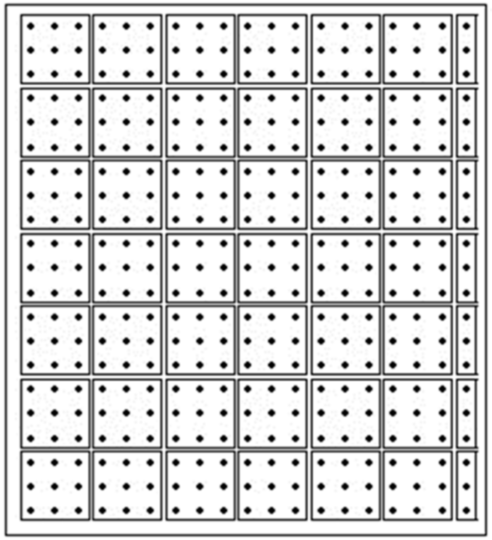
Grid points in the ij(k)-data structure



Grouped together in the blocked data structure

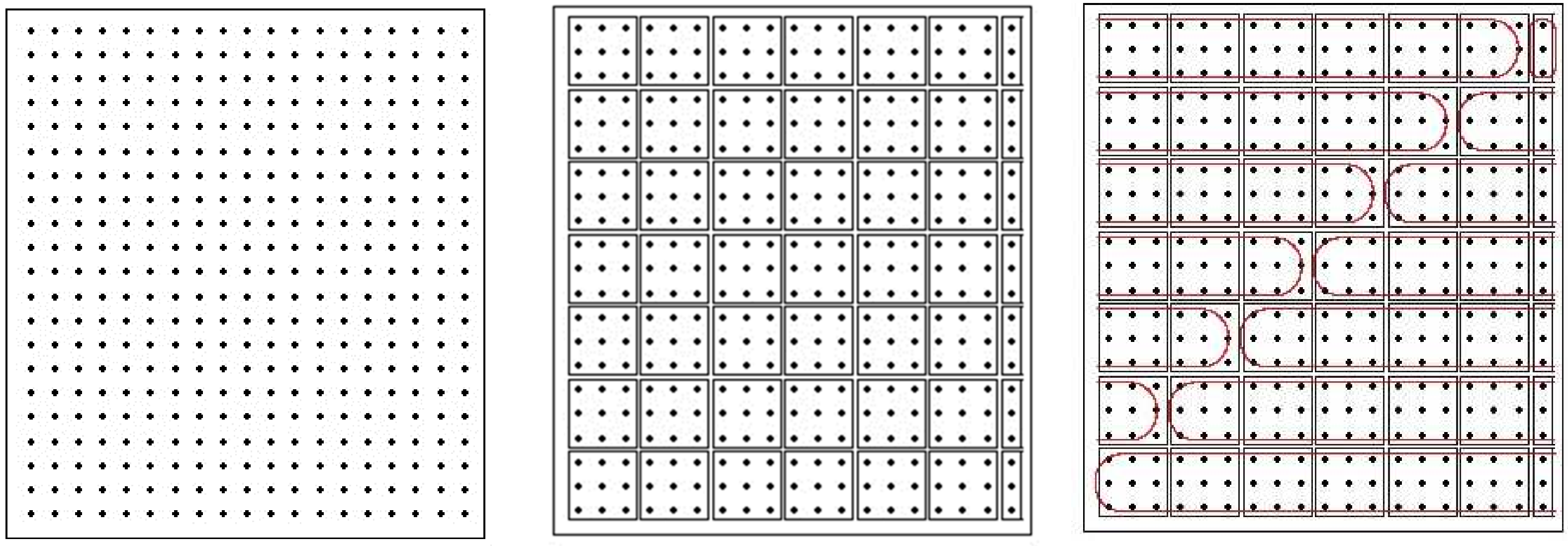


Grouped for coarse radiation grid



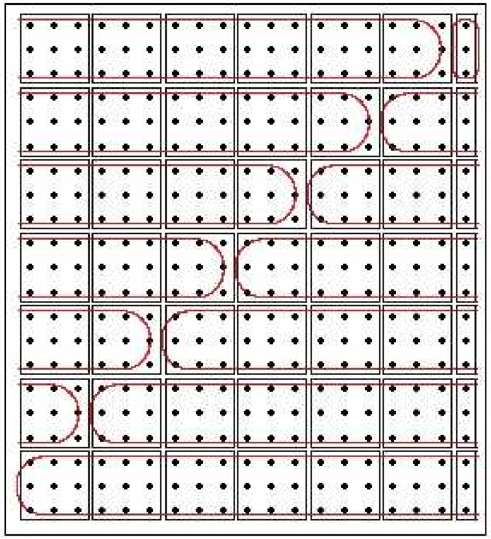
The Coarse Radiation Grid

- ➔ In the blocked data structure it is (nearly) impossible to compute the input values for the coarse radiation grid
- ➔ Therefore the radiation does not use the copy-in / copy-out mechanism, but the input is computed directly from the ijk-data structure but is also provided in blocked structure:



The Coarse Radiation Grid

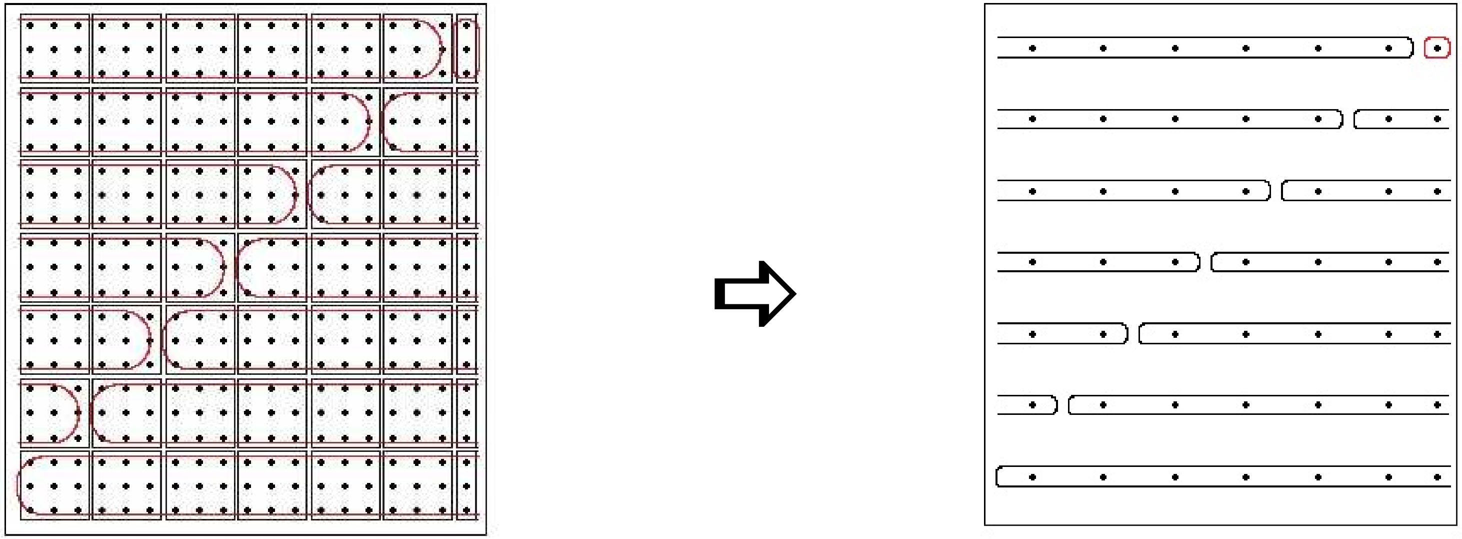
```
Do jp=1, nradcoarse
  DO ip=1, ipdim  !=nproma*nradcoarse
  ! get i/j indices for 2D COSMO data structure
    i = mind_ilon_rad(ip,jp,ib)
    j = mind_jlat_rad(ip,jp,ib)
    zti(ip,ke1,jp) = t_g(i,j,ntl)
  ENDDO
ENDDO
```



Note: `zti` is computed for every COSMO grid point. For `nradcoarse=1` it is just the „usual“ blocked data structure.

The Coarse Radiation Grid

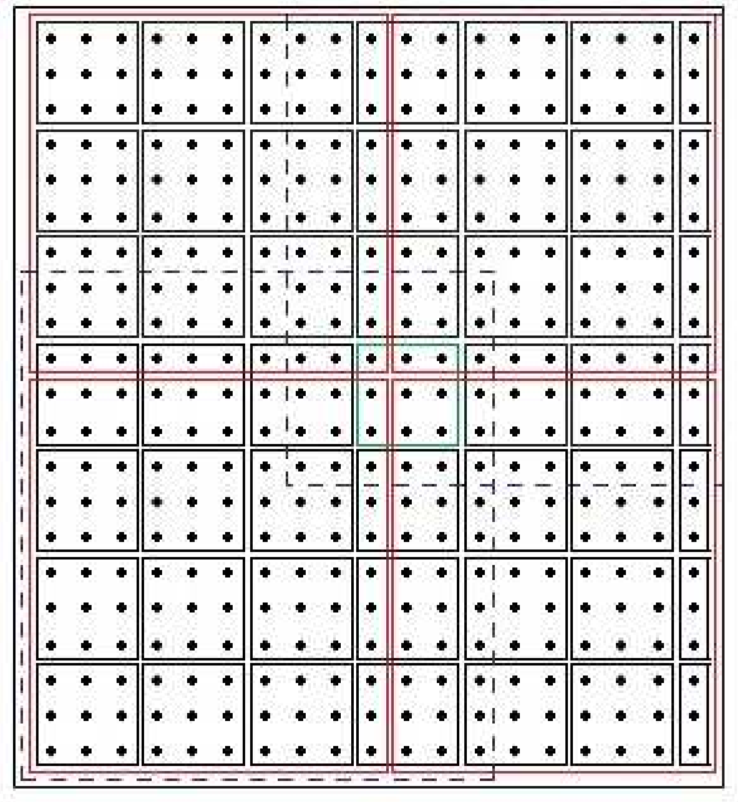
- ➔ In an average step, the input values for the coarser grid are computed.



- ➔ Note that most input- and output-variables of the radiation are not used in other parameterizations (but: `t, qv, qc, qi` and `sobs, pabs, thbs`)
- ➔ There are difficulties, if all packages are run within one block loop and if the microphysics is executed before the radiation.

The Coarse Radiation Grid

➔ And if you want to know the situation in the parallel program:



Impacts on COSMO-ART and 2-Moment Scheme?

Thank you

very much

for your

attention