Allow non-rectangular lat-lon grids

Allow non-rectangular lat/lon grids.

dyn_grid.F90 API:

1. **subroutine get_block_bounds_d(block_first,block_last)** : A block is assumed to be the basic organizing unit of columns in the dynamics (as viewed by the physics), with a unique global index within the range {block_first, ..., block_last} and no unused indices in the range. A block is assumed to be restricted to a single process in the dynamics. The physics uses the block index to request information that applies to all columns assigned to a block. This routine is called in phys_grid_init and in create_chunks, both in phys_grid.F90 .
2. **subroutine get_block_gcol_d(blockid,size,cdex)** : This routine returns the indices of all columns assigned to the indicated block. The number of columns (size) should first be determined by calling get_block_gcol_cnt_d . The physics assumes that there is a unique global index for each (unique) column. It is this index that is returned. This routine is called in phys_grid_init and create_chunks in phys_grid.F90; in infld_real_2d and infld_read_3d in ncdio_atm.F90; in get_dyn_decomp in homme/inidat.F90 .
3. **subroutine get_gcol_block_d(gcol,cnt,blockid,bcid)** : For a given global (dynamics) column index, returns the block ids and local index within that block. The current design allows the dynamics to "expose" a decomposition of columns over the vertical dimension to the physics. Multiple block ids and local index would be returned if the column were decomposed over the vertical, and thus spread over multiple blocks. The number of dynamics blocks over which a column is spread (cnt) should be determined first by calling get_gcol_block_cnt_d . This routine is called in phys_grid_init, create_chunks, find_partners, find_twin, and assign_chunks in phys_grid.F90; in d_p_coupling and p_d_coupling in homme /dp_coupling.F90 . Note the additional structure assumed within a block, that there is also a column ordering that can be referred to explicitly. Note that none of the current dycores expose a vertical decomposition to the physics, and cnt is always "1" at the moment. Finally note that phys_loadbalance = 2 is the only option guaranteed to work if the dycore does decompose over the vertical, as this is the only option that would be able to piece the decomposed columns back together for use in the physics for any arbitrary dynamics decomposition.
4. **subroutine get_block_levels_d(blockid, bcid, lvlsiz, levels)** : This routine returns the vertical level indices for the indicated column (bcid) in this block (blockid). The number of levels (lvlsiz) should be determined first by calling get_block_lvl_cnt_d . This routine is called in phys_grid_init in phys_grid.F90 when determining the remapping between dynamics and physics.
5. **subroutine get_horiz_grid_d(size,clat_d_out,clon_d_out,area_d_out, wght_d_out)** : The physics needs to know the coordinates for each of the columns. The number of columns in the computational grid (size) is determined from calling get_horiz_grid_dim_d . All other parameters are optional, but will return cos(latitude), cos(longitude), column area, and/or weight to use for quadrature rules for each column. This routine is called in phys_grid_init in phys_grid.F90 . The data is stored locally, so other physics routines that need these data get them from phys_grid.
6. **subroutine get_horiz_grid_dim_d(hdim1_d,hdim2_d)** : This routine returns the horizontal dimensions of the (assumed rectangular) dynamics computational grid. For non-lon/lat dycores, hdim1_d is the total number of (unique) columns in the dynamics computational grid and hdim2_d is == 1. This routine is called many places, including in phys_grid.F90, phys_gmean.F90, restart_physics.F90, restart_dynamics.F90 (sld, homme, fv, eul), homme/inidat.F90, pio_utils.F90, cam_history.F90 .
7. **integer function get_block_gcol_cnt_d(blockid)** : Number of columns represented in the indicated dynamics block. See get_block_gcol_cnt_d .
8. **integer function get_block_lvl_cnt_d(blockid,bcid)** : Number of vertical levels in column in indicated block (blockid) and at local column index bcid. See get_block_levels_d .
9. **integer function get_gcol_block_cnt_d(gcol)** : Number of blocks over which column indicated by global column index gcol is spread. See get_gcol_block_d .
10. **integer function get_block_owner_d(blockid)** : This routine returns the id of the MPI process that "owns" the indicated dynamics block during the dynamics. More precisely, this is the process that the physics needs to communicate with when copying the columns out of or into the corresponding block. This routine is called in phys_grid_init, create_chunks, find_partners, find_twin, and assign_chunks in phys_grid.F90 and in homme/inidat.F90 .
11. **integer function get_dyn_grid_parm(name) result(ival)** : This function can be used to retreve any integer value from a dyn grid given it's string name. Many of these names are common to all dycores, others are unique to a given dycore - if a dycore does not recognize a name the function returns -1.

Goal