Exercise 6 – Poisson's equation

Write a Fortran program that iteratively solves the two-dimensional Poisson equation using the multi-grid method:

$$\nabla^2 u = f \tag{1}$$

1. On page 3 (and on Moodle / JupyterHub) you can find a recursive function called `Vcycle_2DPoisson` which performs the steps of the V-cycle. The function calls itself to calculate the correction on the next coarser grid. It also calls other functions and subroutines that have the following tasks but are still missing:

   - The function `iteration_2DPoisson(u,f,h,alpha)` performs a single Gauss-Seidel iteration on $\tilde{u}$:

     $$R = f_{i,j} - \nabla^2 \tilde{u}_{i,j} \tag{2}$$

     $$\tilde{u}_{i,j} = \tilde{u}_{i,j} - \alpha R \frac{h^2}{4} \tag{3}$$

     where $\nabla^2 \tilde{u}_{i,j}$ is calculated with finite differences. It returns the corrected $\tilde{u}$ and the root mean square of the residue (R) as a result.

   - The subroutine `residue_2DPoisson(res)` calculates the residue $R_{i,j} = f_{i,j} - \nabla^2 \tilde{u}_{i,j}$ at every grid point.

   - The subroutine `restrict(fine,coarse)` copies every second grid point in `fine` to `coarse`.

   - The subroutine `prolongate(coarse,fine)` copies every grid point in `coarse` to `fine` and interpolates linearly in between to fill the remaining grid points.

   Your task is to fill in the missing functions / subroutines and create a Poisson solver module from them together with the recursive function `Vcycle_2DPoisson`. The boundary conditions for u and f are 0.

2. Write a main program that tests the Poisson solver by performing several iterations until the root mean square residue is less than 10^{-5} of f (note: 64-bit precision may be needed to satisfy this condition).

   The program should have the option to call either the iteration function directly (without the multigrid method) or the V-cycle function so it is possible to compare the speed of the two methods. It should perform the following steps:

   - Read in input variables from a namelist:
     - Number of grid points `nx`, `ny` (should be $2^n + 1$).
     - Initialization `init_type` (`random` or `spike`)

  - Switch for multigrid `multigrid` (`.TRUE.` or `.FALSE.`)
  - Relaxation factor `alpha`

- Initialize variables:

  - Grid spacing `h=1./(ny-1.)`
  - Array `f` of size `(nx,ny)` with random numbers or delta function
  - Array `u` of size `(nx,ny)` with `0`.

- Repeatedly call either `iteration_2DPoisson` or `Vcycle_2DPoisson` until the solution converges (according to `res_rms` compared to `f`)

- Write out `f` and `u` for visualization.

Deadline: Please hand in your solutions (`.f90` files and plots of `f` and `u`) by **Tuesday, 30 April 2024, 23:59**.

```fortran
RECURSIVE FUNCTION Vcycle_2DPoisson(u,f,h,alpha) RESULT (res_rms)
IMPLICIT NONE
REAL, INTENT(INOUT) :: u(:,:)
REAL, INTENT(IN) :: f(:,:), h, alpha
REAL :: res_rms ! root mean square residue
INTEGER :: nx, ny, nxc, nyc, i ! local variables
REAL, ALLOCATABLE :: res_c(:,:), corr_c(:,:), res_f(:,:), corr_f(:,:)

nx = SIZE(u,1); ny = SIZE(u,2) ! must be power of 2 plus 1
nxc = (nx+1)/2; nyc = (ny+1)/2 ! coarse grid

IF (MIN(nx,ny) > 5) THEN

  ALLOCATE(res_f(nx,ny), corr_f(nx,ny))
  ALLOCATE(res_c(nxc,nyc), corr_c(nxc,nyc))

  ! take two iterations on the fine grid
  res_rms = iteration_2DPoisson(u,f,h,alpha)
  res_rms = iteration_2DPoisson(u,f,h,alpha)

  ! restrict residue to the coarse grid
  CALL residue_2DPoisson(u,f,h,res_f)
  CALL restrict(res_f,res_c)

  ! solve for the coarse grid correction
  corr_c = 0.
  res_rms = Vcycle_2DPoisson(corr_c,res_c,h*2,alpha) ! recursion

  ! prolongate (interpolate) the correction to the fine grid
  CALL prolongate(corr_c,corr_f)

  ! correct the fine-grid solution
  u = u + corr_f

  ! take two more smoothing iterations on the fine grid
  res_rms = iteration_2DPoisson(u,f,h,alpha)
  res_rms = iteration_2DPoisson(u,f,h,alpha)

  DEALLOCATE(res_f,corr_f,res_c,corr_c)

ELSE ! coarsest level (ny=5): iterate to get "exact" solution

  DO i = 1,100
    res_rms = iteration_2DPoisson(u,f,h,alpha)
  END DO

END IF

END FUNCTION Vcycle_2DPoisson
```