Project 3

Deadline: Sunday, 5 June 2022

In this project you will implement a multigrid method for the numerical solution of the following elliptic boundary value problem in $\mathbb{R}^d$, $d = 1, 2$.
Let $\Omega = (-2, 2)^d \subset \mathbb{R}^d$. Given the functions $f, g \in C^0(\overline{\Omega})$ find a function $u \in C^2(\overline{\Omega})$ such that

$$\begin{aligned} -\Delta u + u &= f \quad \text{in } \Omega\,, \\ u &= g \quad \text{on } \partial\Omega\,. \end{aligned}$$

Throughout this project we fix

$$g(x) = \tfrac{1}{10}|x|^2\,.$$

Let $x_0 = -2 < x_1 < x_2 < \ldots < x_{N-1} < 2 = x_N$ be a uniform partitioning of the interval $[-2, 2]$ with mesh size $h = \frac{4}{N}$, $N = 2^\ell$, $\ell \geq 1$. We define the uniform grids $\Omega_\ell = \{x_0, x_1, \ldots, x_N\}$ in 1d and similarly $\Omega_\ell = \{(x_0, x_0), (x_0, x_1), \ldots, (x_0, x_N), \ldots, (x_N, x_0), \ldots, (x_N, x_N)\}$ in 2d.
Let

$$\widehat{A}_h \widehat{u}_h = \widehat{f}_h, \qquad \text{where } \widehat{A}_h \in \mathbb{R}^{(N+1)^d \times (N+1)^d}, \tag{1}$$

be the standard finite difference approximation to the given boundary value problem.
We also consider the sequence of grids $\Omega_1, \ldots, \Omega_\ell$, where Ω_1 is the trivial grid with only a single interior node.
For the implementation of the following tasks you should use Python, and you are free to use all available standard types and methods. You are only allowed to import the external package NumPy, as well as Matplotlib for the preparation of your report.

Multigrid in 1d

For the one-dimensional simulations, we choose

$$f(x) = (x-1)^3\,.$$

1. Implement a function `gs_step_1d(uh, fh)` that performs one step of the Gauss–Seidel method for the linear system (1). The inputs are `uh`, the initial guess $\widehat{u}_h^{(k)}$, and `fh`, the right hand side from (1). Recall that $\widehat{f}_h = (\gamma_0, f(x_1), \ldots, f(x_{N-1}), \gamma_1)$. The outputs are the new iterate $\widehat{u}_h^{(k+1)}$ (in place of `uh`), and the pseudo-residual $|\widehat{u}_h^{(k+1)} - \widehat{u}_h^{(k)}|_\infty$, which acts as the return value of the function.

   Note that the function should infer the value of N, and hence $h = \frac{4}{N}$, from the length of the arrays `uh` and `fh`.

2. Investigate the number of iterations needed for the Gauss–Seidel method, and the effect on the CPU time, applied to the linear system (1) for the stated model problem, as h decreases.

   Here the iteration, starting from $u_h^{(0)} = 0$, should stop when the pseudo-residual satisfies $|u_h^{(k+1)} - u_h^{(k)}|_\infty < \texttt{tol} = 10^{-8}$. Your sequence of grid sizes should include at least $N = 2^\ell$, $\ell = 3, \ldots, 7$.

3. Implement the two-grid correction scheme, where instead of solving $\widehat{A}_{2h}\widehat{e}_{2h} = \widehat{r}_{2h}$, you perform five Gauss–Seidel steps on the coarse grid. Investigate the improvement on the

number of iterations, and on the CPU time, compared to the standard Gauss-Seidel iterations you have used above.

Note that the residual equation on the coarse grid is solved for the error, and that the boundary conditions on the coarse grid must always be homogeneous.

4. Implement a function `v_cycle_step_1d(uh, fh, alpha1, alpha2)` that performs one V-cycle for the linear system (1), using α_1 Gauss–Seidel pre-smoothing steps, and α_2 post-smoothing steps. On the coarsest grid Ω_1 use an exact solve. The function should return the pseudo-residual of the final post-smoothing step, unless it is called on the coarsest grid Ω_1, when it can return zero.

   You can make the same assumptions on `uh` and `fh` as for the function `gs_step_1d()`. But here, in addition, you may assume that $N = 2^\ell$, for some $\ell \geq 1$.

5. Investigate the number of iterations needed for the implemented V-cycle with $\alpha_1 = \alpha_2 = 1$, and with $\alpha_1 = \alpha_2 = 2$, and the effect on the CPU time, applied to the linear system (1) for the stated model problem, as h decreases. Here the iteration, starting from a zero initial guess as usual, should stop when the pseudo-residual satisfies $|s_h^{(k+1)}|_\infty < \texttt{tol} = 10^{-8}$ for the final post-smoothing step on the finest grid. Your sequence of grid sizes should include at least $N = 2^\ell$, $\ell = 3, \ldots, 14$.

6. Implement a function `full_mg_1d(uh, fh, alpha1, alpha2, nu)` that performs a full multigrid step, utilizing your V-cycle implementation to perform ν V-cycle steps, with α_1 Gauss–Seidel pre-smoothing steps and α_2 post-smoothing steps, on each grid, for the linear system (1). You can make the same assumptions on `uh` and `fh` as for the function `v_cycle_step_1d()`. The function should return the result of the final call to `v_cycle_step_1d()` on the finest grid.

   Note that because of the inhomogeneous boundary conditions, special care has to be taken when defining the right hand side vectors $\widehat{f}_{2h}$, $\widehat{f}_{4h}$, etc., on the coarser grids. To keep things simple, we recursively define $\widehat{f}_{2h} = \widehat{I}_h^{2h} \widehat{f}_h$, where $\widehat{I}_h^{2h} : \mathbb{R}^{N+1} \to \mathbb{R}^{\frac{N}{2}+1}$ is the natural *injection operator*, i.e. we do not(!) employ full weighting here.

7. Investigate the CPU time needed, and the sizes of the final pseudo-residual $|s_h^{(k+1)}|_\infty$ as well as the size of the residual $|r_h|_\infty = |f_h - A_h u_h|_\infty$ after one full multigrid step with $(\alpha_1, \alpha_2, \nu) \in \{(1,1,1), (1,1,2), (2,2,1), (2,2,2)\}$, as h decreases. Your sequence of grid sizes should include at least $N = 2^\ell$, $\ell = 3, \ldots, 14$.

8. Plot an approximation to the solution u over $\overline{\Omega}$ and provide an accurate estimate of $\min_{x \in \overline{\Omega}} u(x)$.

Multigrid in 2d

For the two-dimensional simulations, we choose

$$f(x) = f(x_1, x_2) = x_1^2 - x_2^2 .$$

1. Implement the natural 2d variants of the previously discussed functions:

   - `gs_step_2d(uh, fh)`,
   - `v_cycle_step_2d(uh, fh, alpha1, alpha2)`,
   - `full_mg_2d(uh, fh, alpha1, alpha2, nu)`.

   In each case you can assume that `uh` and `fh` are two-dimensional NumPy arrays of size $(N+1) \times (N+1)$, where for the latter two functions $N = 2^\ell$ for some $\ell \geq 1$.

2. Repeat the analogous numerical investigations for the 1d model problem now in 2d. For the sequences of mesh sizes you should include at least $N = 2^\ell$, $\ell = 2, \ldots, 6$ for the Gauss–Seidel iterations, and at least $N = 2^\ell$, $\ell = 2, \ldots, 8$ for the multigrid methods.

3. Plot an approximation to the solution u over $\overline{\Omega}$ and provide an accurate estimate of $\min_{x \in \overline{\Omega}} u(x)$.

Submission via Moodle

Prepare a single Python source file called `project3.py` that contains the six requested functions you implemented, together with any routines or classes that they depend on, but nothing else. Importing your source code via `import project3` should **not** execute any code. The docstring of the file `project3.py` must contain your full name. In addition, prepare a short report in PDF format, called `project3.pdf`, that summarizes your investigations and reports on your results.