# 1 Iterative solvers in 2D

This section goes through necessary parts needed to build a V-cycle solver on a 2D grid with the same assumptions as in section 2. Without further ado, let's take a look at a pseudoalgorithm first and then explain what individual functions do in detail. The name V-cycle comes from the recursive part of the algorithm – the 2

```
1  function Vcycle(U, ω, n_smooth, m, F)
2  if m = 1 then
3  |   directly solve at the coarsest level
4  else
5  |   pre-smoothing: smooth
6  |   compute the residual: r = F − Au
7  |   coarsen residuals: coarsen
8  |   recursive part to obtain error: Vcycle(0, ω, n_smooth, −R_coarse)
9  |   interpolate the error: interpolate
10 |   update the solution: U = U + E
11 |   post-smoothing: smooth
12 end
13 return U
```

**Algorithm 1:** V-cycle algorithm pseudocode, full reference can be found on slide 8 of Lecture 20 or in LeVeque 4.6.2 "The multigrid approach"

lines indicate the path of pre- and post-smoothing on coarse/interpolated grids, whereas the point of the intersection indicates directly finding the solution in the first `if` branch.

It is worth noting that due to coarsened grid the depth of the recursion will be $\log_2(m)$ per one cycle. The cycle will be repeated until a desired tolerance is reach or the maximum number of iterations has passed.

## 1.1 Matrix free 5-point Laplacian

As requested in the handout, matrix-free method `Amult` will be used to calculate the term $-\mathbf{Au}$ without storing the $\mathbf{A}$ matrix in memory (as it is of size $m^2 \times m^2$ and somewhat sparse). The implementation can be tested with Matlab's Conjugate Gradient algorithm `pcg`, which can accept a function handle as the first argument, so `Amult` shall be provided. The reason why $-\mathbf{Au}$ is used instead of the positive alternative is that PCG requires a positive definite system and the $\mathbf{A}$ matrix is in fact negative definite. To check the difference between the implementation of the matrix free method and the direct multiplication of $u$ with the Laplacian matrix,

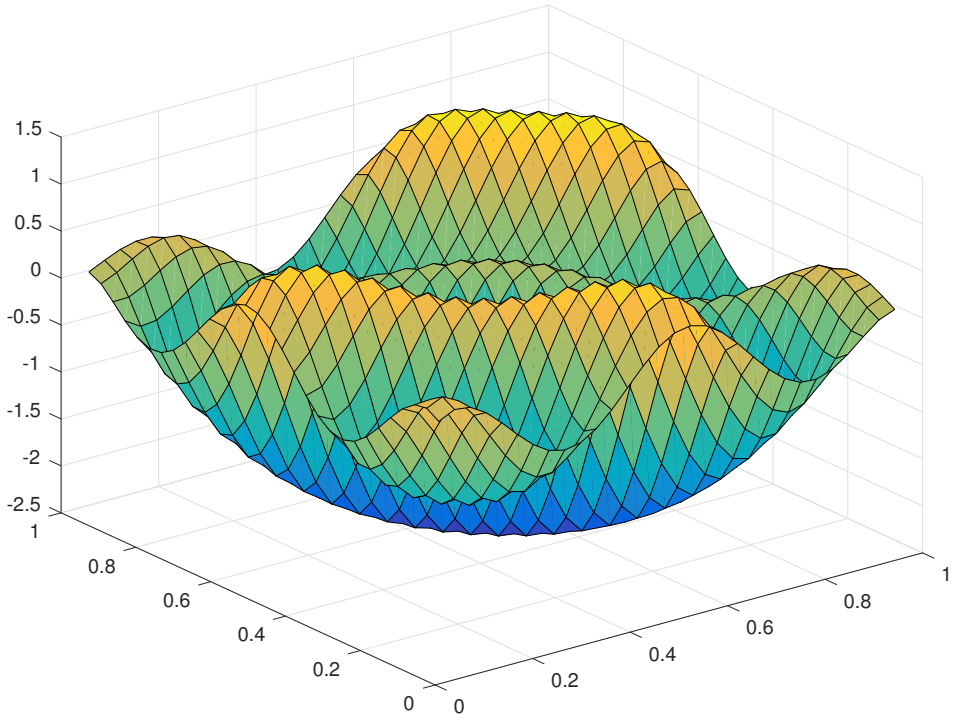


Figure 1: Solution to $-\mathbf{Au} = -\mathbf{f}$ using PCG

one can for example use `max(max(abs(Amult(u,m) - -1*poisson5(m)*u)))` in Matlab. The maximum of the absolute difference is roughly $10^{-12}$.

All the Matlab code for this section is provided in the appendix.

## 1.2 Relaxation and smoothing

The main motivation to use underelaxed Jacobi is that while the high frequency components of the error decay after a few iterations when applying the ordinary Jacobi method, the lower frequency components remain longer and negatively affect the convergence of the algorithm.

Thereby, by including an extra parameter $\omega$ we can manage how far the current iterate moves towards the true solution, and thus control high frequency harmonics.

$$G_\omega = (1-\omega)I + \omega G \tag{1}$$

where $G$

$$G = I + \frac{h^2}{2}A \tag{2}$$

besides, considering that the eigenvectors of $G$ are the same as the eigenvectors of $A$

$$\rho_{p,q} = 2 + \frac{h^2}{2}\lambda_{p,q} \tag{3}$$

from 3.15 in Leveque we know that the eigenvalues of $A$ are

$$\lambda_{p,q} = \frac{2}{h^2}((cos(p\pi h) - 1) + (cos(q\pi h) - 1)) \tag{4}$$

thus, combining the equations above we obtain the following expression for the eigenvalues of $G_\omega$

$$\gamma_{p,q} = (1 - \omega) + \omega(cos(p\pi h) + cos(q\pi h)) \tag{5}$$

and since we would like to choose $\omega$ so that we get optimal smoothing of high frequencies

$$\omega_{opt} = min(max|\gamma_{p,q}|) \tag{6}$$

Figure 2 shows $max|\gamma_{p,q}|$ for different values of the smoothing factor. We see in this case that $\omega_{opt}$ lies very close to $0.5$.

We have implemented a script in MATLAB that plots $max|\gamma_{p,q}|$ as a function of $\omega$ and finds the minimum of that function. We also wrote the function smooth which applies the under/overrelaxed Jacobi method for a given value of the smoothing parameter.

## 1.3 Coarsing / interpolating

One approach to solve this is issue is taking a fine grid to a coarse one and back (V). As is mentioned in the book, the convergence rate for some components of the error is greatly improved by transferring the error to a coarser grid. An example is given that a component's frequencies can be shifted to the right, i.e. more in the middle of what range the coarse grid has to offer, resulting in the improvement of the damping factor (in some cases as high as one order of the magnitude).

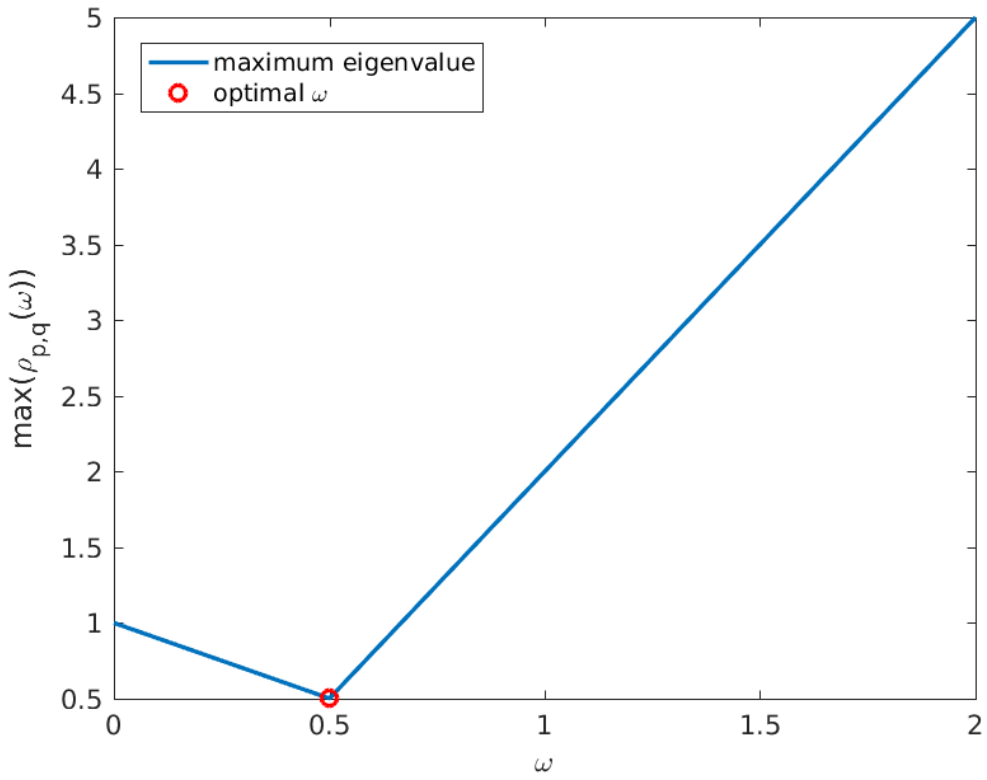


Figure 2: $max|\gamma_{p,q}|$ for different values of $\omega$ and $m = 1000$

## 1.4 Recursion

Now the problem is refolmulated in order to transfer the remaining part to the coarser grid. The approach taken uses the recursive call to find the error vector by solving $\mathbf{Ae} = -\mathbf{r}$ via the `vcycle` method.

Since the result is obtained on a coarse grid, interpolating is used in order to project the error vector back to the original $m \times m$ grid. In order to find a better approximation of $U$ the error vector is added and post-smoothing is performed. Once all the recursive results are popped from the stack a solution to the given problem is returned. As mentioned before this process is repeated until a desired tolerance or maximum number of iterations is reached.