Résolution de Systèmes Linéaires issus des EDP

ModIA

TP4 Méthodes Multigrilles

Année scolaire 2024–2025

Carola Kruse   Ronan Guivarch

1 Introduction

In this TP, we will implement a V-cycle multigrid scheme with $L+1$ grid levels and using this, we will also implement a Full Multigrid method.

2 Multigrid in 1D (Matlab)

We keep the same problem setting as in TP1, but we use a different analytic solution. We search for a solution of a 1D Poisson problem

$$\begin{aligned} -u'' &= f, \qquad \text{in } \Omega = [0,1] \\ u(0) &= u(1) = 0, \end{aligned}$$

We choose as analytic solution

$$u(x) = x^2(1-x)^2 \tag{1}$$

and we can compute the right-hand site

$$f = -u'' = -2 + 12x - 12x^2. \tag{2}$$

We are then able to compute the discretization error between the numerical and the exact solution of the PDE. As you know, this error depends on the discretization method and the mesh size h.

2.1 The grid

We will implement multigrid solvers on a hierachy of grid levels. Let therefore N be the number of elements on the finest grid of the interval $[0,1]$ with mesh size $h = \frac{1}{N}$. For a multigrid method with $L+1$ grid levels, we define the following grids

$$\begin{array}{llll} \Omega_h, & x_i = ih, & i = 0,..,N & \text{(finest grid)}, \\ \Omega_{2h}, & x_i = 2^1 ih, & i = 0,..,N/(2^1) & \\ \vdots & \vdots & \vdots & \\ \Omega_{2^L h}, & x_i = 2^L ih, & i = 0,..,N/(2^L) & \text{(coarsest grid)}. \end{array}$$

2.2 Finite difference discretization

We use centered finite differences with

$$u_i'' \approx \frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i, \qquad i = 1, .., N-1 \tag{3}$$

and obtain the linear system

$$A\vec{u} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix} \tag{4}$$

We do not include the boundary nodes into the system, as these are given by $u_0 = u_N = 0$. The matrix is thus of size $(N-1) \times (N-1)$. You find the code of TP1 that is useful for TP2 on Moodle.

3 V-cycle multigrid

We first extract the two-level multigrid step from the sample code of TP1 and save it as a function V_cycle.m. We also include post smoothing with ν_2 steps of weighted Jacobi. Therefore, please write a function containing the following elements

```
function v = V_cycle(Ah,rhsf,u0,omega,nu1,nu2,N)
 % 1. Setup of intergrid transfer operators
 % 2. nu1 steps of weighted Jacobi smoothing
 % 3. Computation of residual
 % 4. Restriction of residual
 % 5. Construction of coarse grid matrix via Galerkin projection
 % 6. Coarse grid solve
 % 7. Interpolation of coarse grid error
 % 8. Update of solution
 % 9. nu2 steps of weighted Jacobi smoothing
```

To avoid code complexity, we will built the interpolation operators inside the V_cycle multigrid function. Also, we will pass only the matrix A_h of the finest grid. The matrices on the coarser grids are then constructed by the Galerkin projection.

The script (*script_vcycle_1.m*) to solve the linear system (4) can then be written as

```
clear all;
% Setup maillage
N = 256;
h = 1/N;
% Interior mesh points
xi=h:h:1-h; xi = xi'; % row vector to column vector

% Jacobi weighting parameter
omega = 2/3;
% Setup of the fine grid matrix and the right-hand side
Ah = getMatrixA(N);
rhsf= -2 + 12*xi - 12*xi.^2;

% Initial vector
v(1:N-1,1) = 0;

% We do 10 iterations.
maxit = 10;
res = zeros(maxit+1,1);
% Initial residual
res(1) = norm(rhsf - Ah*v);

nu1=1;
nu2=1;

for i=1:10
   v = V_cycle(Ah,rhsf,v,omega,nu1,nu2,N);
   res(i+1) = norm(rhsf - Ah*v);
 end
```

3.1 Discrete L_2-error

In the following, we want to compute the discretization error for the solution u and the numerical approximation u^h on a given grid with mesh size h. For this we need a discrete version of the continuous L_2-error

$$\|u\|_2 = \left(\int_\Omega u(x)^2 \, dx \right)^{1/2} .$$

The discrete L_2 error is given by

$$\|u^h\|_2^h = \left(h^d \sum_{i=1}^{N} u(x_i)^2 \right)^{1/2}$$

with d being the dimension of the domain. This factor h^d is necessary to have the correct scaling when passing from the discrete to the continuous L_2-norm, thus if $h \to 0$.

Exercise as a supplement for the students studying for a MSc degree (and for those interested in it of course):
Let $g(x) = x$, $\Omega = [0, 1]$, $h = \frac{1}{N}$. Show that

$$\|g\|_2^2 = \frac{1}{3},$$
$$\|g^h\|_2^2 \overset{h \to 0}{=} \frac{1}{3},$$

Hint: $\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$.

3.2 Computation of the discretization error

To compute the L_2-norm of the error $e^{(m)} = u - u_h^{(m)}$, we write a function

```
function err = compute_L2_error(N,u,uh)
% 1. Get mesh size h
h = 1/N;
% 2. Error vector
em = ?
% 3. Compute the error with the formula above
err = ?
end
```

Next, we compute the residual error and the discretization error after each iterate of the V-cycle. For this we introduce a new variable errorL2, in which we store the discretization error at each iteration.

Exercise: Modify the script (*script_vcycle_2.m*) by adding the exact solution evaluated on the interior grid points and adding the error computation.

```
% Script to run the V-cycle
%...
%...
% Evaluate the solution at each mesh point
xi=h:h:1-h; xi = xi'; % row vector to column vector
usol = ?
rhsf= -2 + 12*xi - 12*xi.^2;
% ...
% ...
maxit = 10

% Initialize errors variable and compute initial errors
res = zeros(maxit+1,1);
errorL2 = zeros(maxit+1,1);
% Initial residual and L2 error
res(1) = norm(rhsf - Ah*v);
errorL2(1) = ?

nu1=1;
nu2=1;

for i=1:10
   v = V_cycle(Ah,rhsf,v,omega,nu1,nu2,N);
% Compute errors after each iteration
   errorL2(i+1) = ?
   res(i+1) = norm(rhsf - Ah*v);
 end
```

Exercise: Run the script for $N = 32, 64, 128, 256$ elements.

1. What do you notice for the discretization error in errorL2?
2. What do you notice for the residuals in res?
3. Write down the converged discretization error for the four tests. What is the ratio between two following errors? What does this tell you?

3.3 From 2-grid to multigrid

The next step is to modify the function V_cycle.m to obtain a (L+1)-level multigrid method. Before continue reading, think about

1. Which part of the above algorithm has to be modified?
2. What is a reason to use more than two levels?

Indeed, if we haven't reached the coarsest level yet, then we do not do the coarse grid solve at line 6, but we will recursively continue with another V-cycle. Modify the function including a parameter L (*V_cycle_L.m*) that indicates the number of grid level, such that

```
function v = V_cycle_L(Ah,rhsf,u0,omega,nu1,nu2,N, L)
 % 1. Setup of intergrid transfer operators
 % 2. nu1 steps of weighted Jacobi smoothing
 % 3. Computation of residual
 % 4. Restriction of residual
 % 5. Construction of coarse grid matrix via Galerkin projection

 % If the coarsest level is reached (L=1) then
 % coarse level solve
 % else
 % L = L-1
 % Do another V-cycle
 % end

 % 7. Interpolation of coarse grid error
 % 8. Update of solution
 % 9. nu2 steps of weighted Jacobi smoothing
```

Exercise: Use $N = 256$ and vary the number of levels.

1. What is the maximal level L that can be used?
2. Verifiy that you obtain the same discretization error for varying L as for the two-level method.
3. What do you observe for the error reduction?

4 Full Multigrid

In the following section, we look into the *Full multigrid method*. It is based on a nested iteration technique, where we use a solution on a coarser grid, interpolate it to the next finer grid and use it as initial guess for a V-cycle.

In FMG, we first solve the linear system with a direct method on a coarse grid. This solution then respects the discretization error for the given mesh and used discretization method. It is interpolated up to the next finer grid, where it is used as the initial guess for a (1+1)-grid V-cycle.

If all parameters are well chosen (e.g. the number of smoothing steps) then the new solution at the next finer level should also be of the quality of the discretization error on this level. The first iteration can be written as

```
% Script Full multigrid One Level

clear all;
N = 2^2; %coarsest grid with 4 elements
h = 1/N;
xi=h:h:1-h; xi = xi'; % Mesh points

% Setup Jacobi
omega = 2/3;
nu1 = 2; % Presmoothing
nu2 = 1; % Postsmoothing

% Setup the matrix and the right-hand side at the coarsest level
Ah = getMatrixA(N);
rhsf= -2 *(1-xi).^2 + 8*xi.*(1-xi) - 2*xi.^2;

% Get the intial solution on the coarsest grid
sol_ref = Ah \ rhsf;

% Get solution u and the L2-error of the error
usol = xi.^2.*(1-xi).^2;
err(1) = compute_L2_error(N,usol,sol_ref);
v = sol_ref;
% Move to next finer grid
NN = 2*N; h = 1/NN; % 8 elements
xi=h:h:1-h; xi = xi'; %overwrite mesh
% Get interpolation operators, right-hand side and the stiffness matrix at ↩
    ↪ the next higher level
I2hh = interpol(NN);
Ah = getMatrixA(NN);
rhsf= -2 *(1-xi).^2 + 8*xi.*(1-xi) - 2*xi.^2;

% Interpolate the last solution as initiel guess to next higher level
v = I2hh * v;

% Run a (1+1)-level V-cycle with initial guess v
v = V_cycle(Ah,rhsf,v,omega,nu1,nu2,NN,1);
% Get solution u and the L2-error of the error
usol = xi.^2.*(1-xi).^2;
err(2) = compute_L2_error(NN,usol,v);
```

What we have just seen is one step of FMG. This principle can be repeated and the solution after this first step can again be interpolated to the next finer grid where it is used as initial guess

for a (2+1)-level V-Cycle thus $L = 2$. The levels are chosen, such that the coarse grid solve is kept at the same coarse level.
Exercise: Take the above code and extend it to more than the next higher level. Introduce a variable *target* that indicates the finest level with $N = 2^{2+target}$ elements.

```
% Script Full multigrid

clear all;
N = 2^2; %coarsest grid with 4 elements
h = 1/N;
xi=h:h:1-h; xi = xi'; % Mesh points
%...
%... As given before
%...
% Get solution u and the L2-error of the error
usol = xi.^2.*(1-xi).^2;

% Define target level for the finest discretization 2^target * N
% Here: 2^(target+2)
target = 4

% Define error variable
errorL2 = zeros(target+1,1);
err(1) = compute_L2_error(N,usol,sol_ref);
v = sol_ref;


for i=1:target
% YOUR CODE HERE
end
```

Now, that the framework is set up, we do experiments to see the behavior of the discretization error that we reach at each new level. If for a given problem the interpolation and the parameters in the multigrid cycle are well chosen, then we expect to reach the discretization error after one single $(k + 1)$-level V-Cycle at level $k = 0, ..., target$, with $k = 0$ beging the coarsest level.

1. Run the FMG cycle with $(\nu_1, \nu_2) = (1, 0)$, $(\nu_1, \nu_2) = (1, 1)$, $(\nu_1, \nu_2) = (2, 1)$ and up to N = 256 elements. Copy the errors on each level in a spreadsheet.

2. Look into the ratio from the errors from a coarser to a finer grid. We know that for a finite difference discretization, the L_2-error should be divided by a factor of 4. What do you see for the three cases above?

3. Run the V-cycle scheme of the previous section for N=256 elements, $(\nu_1, \nu_2) = (2, 1)$ and $L = 6$, so that the coarsest grid matrix is of the same size. Save also the L_2- error in each iteration into the spread sheet.

4. How many iterations does the 7-level V-cycle need to reach discretization error?

5. Evaluate the computational work (as seen during the lecture), using only the total number of smoothing steps at each level, that are needed by both methods with $(\nu_1, \nu_2) = (2, 1)$ to reach the discretization error. Which method is more efficient?